NEURO-FUZZY MODELS FOR CLASSIFICATION AND RULE GENERATION

Sushmita Mitra
Machine Intelligence Unit
Indian Statistical Institute
Calcutta 700035, India

A thesis submitted to the *Indian Statistical Institute*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
1994

To my Mother

ACKNOWLEDGEMENTS

First of all I express my gratitude to my supervisor, *Prof. Sankar K. Pal*, without whose guidance, encouragement and affection this thesis could not have been completed. It was he who introduced me to the interesting field of *Neuro-Fuzzy Computing* when I was an M. Tech. student, and continued helping me in every stage of my endeavours to delve into its depths.

The final part of this work was carried out when I was in the R.W.T.H., Aachen, Germany with *Prof. Dr. H. -J. Zimmermann*. His help and guidance during that phase of my work is gratefully acknowledged. I take this opportunity to thank all my friends at the *European Laboratory for Intelligent Techniques and Engineering* for all that they have done for me during my stay there.

I must also thank Dr. M. K. Kundu and Prof. D. Dutta Majumder for their encouragement and co-operation during this work. I owe a lot to my colleagues and friends at the Machine Intelligence Unit, I.S.I., Calcutta. Thanks are also due to the C.S.S.C., I.S.I., for allowing me to use the computing facilities of the Institute available there. I express my gratitude to Mr. S. Chakraborty for drawing some of the diagrams and Mr. S. Bhattacharya for photocopying the manuscript.

Finally, I should acknowledge the constant help, encouragement and love that I have received in abundance from my family members, without whose co-operation it would have been impossible for me to have done this.

ISI, Calcutta
October, 1994.

Sushmita Mitra

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

Machine recognition [1, 2] of patterns can be viewed as a two-fold task, consisting of learning the invariant and common properties of a set of samples characterizing a class, and of deciding a new sample as a possible member of the class by noting that it has properties common to those of the set of samples. In other words, pattern recognition by computers can be described as a transformation from the measurement space M to the feature space F and finally to the decision space D [1], *i.e.*,

$$M \rightarrow F \rightarrow D.$$

Here, the mapping $\delta : F \rightarrow D$ is the decision function and the elements $d \in D$ are termed as decisions.

When the input pattern is an image, the measurement space involves processing tasks such as enhancement, filtering, contour extraction and noise reduction, with the objective being to extract salient features from the pattern. This is termed as image processing [3, 4]. The ultimate goal is the understanding, recognition and interpretation using the processed information available from the image pattern. A complete image recognition/ interpretation scheme is called a vision system [5, 6] and can be viewed as consisting of three levels, *viz.*, low level, mid level and high level.

Artificial intelligence is the field that investigates how computers can be made to exhibit intelligence in different aspects of thinking, reasoning, perception or action. In other words, it involves the study of the mental faculties using computational models [7]. A key observation in this direction is that knowledge should not be represented in heavily encoded forms that suppress the structure and constraints. This has lead to the development of more explicit, symbolic, highly flexible forms of representation of knowledge, such as semantic nets, frames and production rules which can be efficiently processed. An expert system can be viewed as a rule-based application program which provides the user with the facility for posing and obtaining answers (that require expertise) to questions related to the information stored in its knowledge base [8, 9]. Basically it consists of the knowledge base, the inference engine and an user interface linking the external environment to the system. The model typically functions in a narrow domain dealing with specialized knowledge generally possessed by human experts. Such systems possess a non-trivial inferential capability and are expected to be capable of directing the acquisition of new information in an efficient manner. The

knowledge base of an expert system is problem-dependent and contains information that controls inferencing. Traditional rule-based models encode this information as *If-Then* rules. In *classification expert systems* [10], the outputs are represented by class variables each of which can assume continuous values. Programs for diagnosis, fault detection and pattern recognition are examples of applications that can be represented as classification problems and are handled by this type of expert systems. In the medical domain it can be used for diagnosing a particular symptom set as affliction by one or more disease(s).

Fuzzy sets were introduced in 1965 by Zadeh [11] as a new way of representing vagueness in everyday life. This theory [12]-[21] provides an approximate and yet effective means for describing the characteristics of a system which is too complex or ill-defined to admit precise mathematical analysis. It attempts to model the human thinking process and behaviour, and is reputed to handle, to a reasonable extent, uncertainties (arising from deficiencies of information) in various applications particularly in decision making models under different kinds of risks, subjective judgement, vagueness and ambiguity. The deficiencies may result from various reasons, *viz.*, incomplete, imprecise, not fully reliable, vague or contradictory information depending on the problem. Since this theory is a generalization of the classical set theory, it has greater flexibility to capture various aspects of incompleteness or imperfection in information about a situation.

The relevance of fuzzy sets for pattern recognition and image processing problems has been adequately reported in literature [22]-[32]. It is found that the concept of fuzzy sets can be exploited in representing linguistically phrased input features for processing, in extracting ill-defined image regions, primitives, properties and relations among them, in measuring image information, in providing an estimate/ representation of missing or contradictory information and multiclass membership for ambiguous patterns; thereby reducing uncertainty in a recognition system.

As the knowledge base of an expert system is a repository of human knowledge and since much of this is imprecise in nature, often this results in a collection of rules and facts which for the most part are neither totally certain nor totally consistent. The model is also likely to be required to infer from premises that are imprecise, incomplete or not totally reliable. The uncertainty of information in the knowledge base of the question-answering system thus induces some uncertainty in the validity of

its conclusions [33]-[36]. Therefore, the answer to a question must be associated with an assessment of its reliability. Hence a basic problem in the design of expert systems is the analysis of the transmitted uncertainty from the premises to the conclusion and the association of a certainty factor [37]. Fuzzy expert systems [37, 38], incorporating the concept of fuzzy sets at various stages, help in the management of uncertainty in such situations.

Artificial neural networks [39]-[46] are signal processing systems that try to emulate the behaviour of biological nervous systems, by providing a mathematical model of combination of numerous neurons connected in a network. These can be formally defined as *massively parallel interconnections of simple (usually adaptive) processing elements that interact with objects of the real world in a manner similar to biological systems.* The origin of artificial neural networks can be traced to the work of Hebb [47], where a local learning rule was proposed. This rule assumed that correlations between the states of two neurons determined the strength of the coupling between them. Thereby, a synaptic connection that was very active grew in strength and *vice versa.* The benefit of neural nets lies in the high computation rate provided by their inherent massive parallelism. This allows real-time processing of huge data sets with proper hardware backing. All information is stored distributed among the various connection weights. The redundancy of interconnections produces a high degree of robustness resulting in a *graceful degradation* of performance in case of noise or damage to a few nodes/ links. Neural network models have been studied for many years with the hope of achieving human like performance (artificially), particularly in the field of pattern recognition, by capturing the key ingredients responsible for the remarkable capabilities of the human nervous system. Note that these models are extreme simplifications of the actual human nervous system.

For any pattern recognition system, one desires to achieve robustness with respect to random noise and failure of components and to obtain output in real time. It is also desirable for the system to be adaptive to changes in environment. Moreover, a system can be made artificially intelligent if it is able to emulate some aspects of the human processing system. Connectionist approaches [39, 41, 43, 45] to pattern recognition are attempts to achieve these goals. The architecture of the network depends on the goal one is trying to achieve. Similarly, neural networks are also used in designing expert systems. Such models are called connectionist expert systems [10], and they

use the set of connection weights of a *trained* neural net for encoding the knowledge base for the problem under consideration. These models are usually suitable in data-rich environment. They help in minimizing human interaction and associated inherent bias during the phase of knowledge base formation (which is time-consuming in the case of traditional models) and also reduce the possibility of generating contradictory rules.

We see that fuzzy set theoretic models try to mimic human reasoning and the capability of handling uncertainty, whereas the neural network models attempt to emulate the architecture and information representation schemes of the human brain. Integration of the merits of fuzzy set theory and neural network theory therefore promises to provide, to a great extent, more intelligent systems (in terms of parallelism, fault tolerance, adaptivity and uncertainty management) to handle real life recognition/ decision making problems. For the last five to seven years, there have been several attempts [23],[48]-[53] by researchers over the world in making a fusion of the merits of these theories under the heading *neuro-fuzzy computing* for improving the performance in decision making systems.

The present thesis provides some results of investigation demonstrating the effectiveness of the neuro-fuzzy approach for pattern classification, rule generation and inferencing. Some new connectionist models have been developed where the concept of fuzzy sets is used in the input level for handling impreciseness in information in terms of linguistic properties, in the output level for representing ambiguous decisions in terms of membership values, and also during learning. Rule generation and inferencing methodologies have been built into these models so that they can be used as classificatory expert systems. Fuzziness has also been incorporated at the neuronal level, by using *And-Or* operations. The effectiveness of these models has been demonstrated on synthetic as well as real-life problems, and comparisons provided with other related algorithms. In Section 1.2 the preliminaries of pattern recognition and the relevance of fuzzy sets are discussed in short. A general study on traditional expert systems and the utility of fuzzy sets is provided in Section 1.3. Section 1.4 deals with artificial neural network approaches and their usefulness in pattern recognition and expert system models. A review on neuro-fuzzy computing in these fields is included in Section 1.5. In Section 1.6, we present a brief discussion on the promise of genetic algorithms in this regard. The scope of the thesis is provided in Section 1.7.

1.2 Pattern recognition and fuzzy set theory

In a general setting the process of pattern recognition [24],[54]-[57] is visualized as a sequence of few steps, *viz.*, (i) data acquisition, (ii) feature selection, and (iii) classification. At the first step, depending on the environment within which the objects are to be classified, data are gathered via a set of sensors. Afterwards, a feature space is constituted in order to reduce the space dimensionality. However, in a broader perspective this stage significantly influences the entire recognition process. Finally, at the third stage of the scheme the classifier is constructed, or in other words, a transformation relationship is established between features and classes. This can be, for instance, a Bayesian rule of computing *a posterior* class probability, linear or nonlinear discriminant function, nearest neighbor rule, nearest prototype (mean) classification rule, etc. [1, 2, 17, 22, 23, 57, 58].

The approach to pattern recognition can be decision theoretic or syntactic. In the decision theoretic approach, once a pattern is transformed through feature selection to a vector in the feature space, its characteristics are expressed only by a set of numerical values. Classification can be done by using deterministic or probabilistic techniques [1, 2, 57]. On the other hand, when a pattern is rich in structural information (*e.g.*, picture recognition, character recognition, scene analysis) *i.e.*, the structural information plays an important role in describing and recognizing the patterns, it is convenient to use syntactic approaches [59] which deal with the representation of structures via sentences, grammars and automata.

In real life, the complete description of the classes is not known. We have instead, a finite and usually smaller number of samples which often provides partial information for optimal design of feature extractor or classifier. Under such circumstances, it is assumed that these samples are representative of the classes (usually consisting of an infinite number of samples). Such a set of typical patterns is called a *training set.* On the basis of the information gathered from the samples in the training set, the pattern recognition systems are designed, *i.e.*, we decide the values of the parameters of various pattern recognition methods. Design of a classification scheme can be made with labeled or unlabeled data. When the computer is given a set of objects with known classifications (*i.e.*, labels) and is asked to classify an unknown object based on the information acquired by it during training, we call the design scheme supervised learn-

ing; otherwise we call it unsupervised learning (or clustering) [60]. Details of different aspects of pattern recognition, *viz.*, data acquisition and preprocessing, classifier design, cluster analysis, feature selection/ extraction, learning, scene analysis, and their various applications are available in [1, 2, 17],[22]-[24].

There are several properties that a good pattern classifier should possess. These are – on-line adaptation, nonlinear separability, handling of overlapping classes, fast decision making, generation of soft and hard decisions, verification and validation mechanisms for evaluating its performance, and minimizing the number of parameters in the system that have to be tuned.

Fuzzy set theory has proved itself to be of significant importance in pattern recognition problems both using decision theoretic and syntactic approaches. On the one hand, fuzzy algorithms for different decision theoretic recognition and image processing problems are being developed, and on the other hand, fuzzy language, fuzzy grammars and fuzzy automata theories have been developed and applied. In Sections 1.2.1-1.2.3 we describe fuzzy set theory, its relevance to pattern recognition and some of the existing fuzzy models for classification.

1.2.1 Fuzzy sets

A fuzzy set A in a space of points $R = \{r\}$ is a class of events with a continuum of grades of membership and is characterized by a membership function $\mu_A(r)$ which associates with each element in R a real number in the interval $[0, 1]$ with the value of $\mu_A(r)$ at r representing the grade of membership of r in A. Formally, a fuzzy set A with its finite number of supports $r_1, r_2, \ldots, r_t$ is defined as a collection of ordered pairs

$$A = \{(\mu_A(r_i), r_i), i = 1, 2, \ldots, t\},$$

where the support of A is an ordinary subset of R and is defined as

$$S(A) = \{r | r \in R \text{ and } \mu_A(r) > 0\}.$$

Here μ_i, the grade of membership of r_i in A, denotes the degree to which an event r_i may be a member of A or belong to A. Note that $\mu_i = 1$ indicates the strict containment of the event r_i in A. If, on the other hand, r_i does not belong to A then $\mu_i = 0$.

Fuzzy logic is based on the theory of fuzzy sets and, unlike classical logic, it aims at modeling the imprecise (or inexact) modes of reasoning and thought processes (with linguistic variables) that play an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision. This ability depends, in turn, on our ability to infer an approximate answer to a question based on a store of knowledge that is inexact, incomplete, or not totally reliable. In fuzzy logic everything, including truth, is a matter of degree [37]. Zadeh has developed a theory of approximate reasoning based on fuzzy set theory. By approximate reasoning we refer to a type of reasoning that is neither very exact nor very inexact. This theory aims at modeling the human reasoning and thinking process with linguistic variables [12] in order to handle both soft and hard data, as well as various types of uncertainties. Many aspects of the underlying concept have been incorporated in designing decision-making systems [35].

Because fuzzy sets are a generalization of the classical set theory, the embedding of conventional models into a larger setting endows fuzzy models with greater flexibility to capture various aspects of incompleteness or imperfection (*i.e.*, deficiencies) in whatever information and data are available about a real process. Assignment of membership functions of a fuzzy subset is subjective in nature, and reflects the context in which the problem is viewed. It cannot be assigned arbitrarily. In many cases, it is convenient to express the membership function of a fuzzy subset in terms of standard S and π functions [17]. Note that fuzzy membership function and probability density function are conceptually different. Probabilities convey information about relative frequencies of objects while fuzzy membership represents similarities of objects to imprecisely defined properties.

1.2.2 Relevance of fuzzy sets

Uncertainties always exist either explicitly or implicitly in each and every phase of a pattern recognition system. Some of these, which one encounters while designing such a system, are discussed here in short. Let us consider, first of all, the case of decision theoretic approach to pattern classification. With the conventional probabilistic and deterministic classifiers [1, 2, 57, 58], the features characterizing the input patterns are considered to be quantitative (numerical) in nature. The pattern vectors having imprecise or incomplete specification are usually ignored or discarded from the design

and test sets. The impreciseness (or ambiguity) may arise from various reasons. For example, instrumental error or noise corruption in the experiment may lead to partial or partially reliable information available on a feature measurement. Again, in some cases the expense incurred in extracting a very precise exact value of a feature may be high, or it may be difficult to decide on the most salient features to be extracted. For these reasons, it may become convenient to use the linguistic variables and hedges (*e.g., low, medium, high, very, more or less*, etc.) in order to describe the feature information. In such cases, it is not appropriate to give exact representation to uncertain feature data. Rather, it is reasonable to represent uncertain feature information by fuzzy subsets.

Again, the uncertainty in classification or clustering of patterns may arise from the overlapping nature of the various classes. This overlapping may result from fuzziness or randomness. In the conventional classification technique, it is usually assumed that a pattern belongs to only one class, which is not necessarily realistic physically, and certainly not mathematically. A pattern can and should be allowed to have degrees of membership in more than one class. It is therefore necessary to convey this information while classifying a pattern or clustering a data set.

Similarly, consider the problem of determining the boundary or shape of a class from its sampled points (*i.e.*, training samples). There are various approaches [61, 62] described in literature which attempt to estimate an exact shape for the area in question by determining a boundary that contains (*i.e.*, passes through) some or all of the sample points. This is not necessarily true in practice. It may be necessary to extend the boundaries to some extent to represent the possible uncovered portions by the sampled points. The extended portions should have lower possibility to be in the class than the portions explicitly highlighted by these points. The size of the extended regions should also decrease with an increase in the number of sample points. This leads one to define a multivalued or fuzzy shape and boundary of a pattern class.

Let us now consider the problem of processing and recognizing a gray tone image pattern. In a conventional vision system, each operation in low level, mid level and high level involves crisp decision to make regions, features, primitives, relations, and interpretations crisp. Since the regions in an image are not always crisply defined, uncertainty can arise at every phase of recognition tasks. Therefore it becomes convenient, natural and appropriate to avoid committing ourselves to specific (hard) decision by allowing the segments or contours to be fuzzy subsets of the image; the

subsets being characterized by the possibility (degree) of a pixel belonging to them. Similarly, for describing and interpreting ill-defined structural information in a pattern, it is natural to define primitives (line, corner, curve, *etc.*) and relations among them using labels of fuzzy sets. The production rules of a grammar may similarly be fuzzified to account for the fuzziness in physical relation among the primitives; thereby increasing the generative power of a grammar for syntactic recognition of a pattern.

From the afore-mentioned examples we see that the concept of fuzzy sets can be used at the *feature level* in representing input data as an array of membership values denoting the degree of possession of certain properties, as well as in representing linguistically phrased input features for their processing, in weakening the strong commitments for extracting ill-defined image regions, properties, primitives, and relations among them; and at the *classification level* for representing class membership of objects, and for providing an estimate (or a representation) of missing information in terms of membership values.

1.2.3 Fuzzy models for pattern recognition

It may be mentioned that from the very beginning of the development of fuzzy set theory, its application to pattern recognition played a very significant role. It is two fold : (i) a methodological one - this leads to treatment of fuzzy sets as a well-suited theory within which one can establish a plausible tool for modeling and mimicking cognitive process of the human being, especially those concerning recognition aspects; (ii) secondly, fuzzy sets offer a lot of novel algorithms which are useful for the designing of feature analysis and classification procedures. Recently, Bezdek and Pal [23] have provided an excellent review on various approaches which helped in the evolution of fuzzy pattern recognition. One may also consult, in this context, the review article of Pedrycz [63]. The present survey is in the line of that reported in [23].

Research on the application of fuzzy set theory to supervised pattern recognition was started in 1966 in the seminal note of Bellman, Kalaba and Zadeh [64] where the two basic operations, *viz.*, abstraction and generalization were proposed. Abstraction in fuzzy set theory means estimation of a membership function μ of a fuzzy class from the training samples. Having obtained the estimate, generalization is performed when this estimate is used to compute the values of μ for unknown objects not contained in

the training set. Consideration of linguistic features and fuzzy relations in representing a class has also been suggested by Zadeh. The work by Pal and Dutta Majumder [65] outlines an early application of fuzzy sets for decision theoretic classification, where a pattern is considered as an array of linguistically phrased features denoting certain properties and where each of these features is a fuzzy set. The variation of the recognition score (for speech data) with the change of fuzziness in the linguistically phrased feature values has subsequently been investigated [17]. These classifiers have also been used for designing a self-supervised recognition system [66]. Nath *et al.* [67, 68] proposed a classification model applicable in the soft sciences (*e.g.*, medical diagnostics) where enough *a priori* knowledge about the classifier is available from the expert in linguistic form.

A fuzzy version of the well known *k-nearest neighbor (k-NN)* classifier has been provided by Keller *et al.* [69]. In the conventional approach [1, 2, 57, 58], each of the labelled samples is given equal importance in deciding the class membership of an unknown pattern; this frequently causes problems in places where the labelled samples overlap. This is tackled in [69] by providing fuzzy label vectors for the samples as an indication of their class representativeness and subsequently leads to a fuzzy classification rule. The fuzzy version of the k-NN rules seems to offer better performance (lower error rates) than crisp rules.

An adaptive system can be viewed as a learning machine, in which the system's decisions gradually approach the optimal decisions by acquiring the necessary information from the observed patterns. The approaches considered so far use a specified set of labelled data for the training of a classifier before it is used for classifying unknown patterns. Devi and Sarma [70] proposed an adaptive algorithm using a fuzzy approximation to the gradient descent technique for training a classifier sequentially. They suggested a method for eliminating or discarding doubtful or unreliable samples from the training procedure.

Although the task of feature selection plays an important role in designing a pattern recognition system, the research in this area using fuzzy set theory has not been significant. Bezdek and Castalez [71] showed an application of the fuzzy c-means clustering algorithm to select an optimum feature subset from the available features so that there is no appreciable loss of classifier performance with the reduced set of features. Pal and Chakraborty explained in [72] an application of fuzziness measures (the index

of fuzziness, entropy, and π-ness) of a set in selecting features without going through classification. This work has then been extended to evaluate the importance of any subset of features to provide an average quantitative index of goodness.

Chang and Pavlidis [73] incorporated the concept of fuzzy decision trees in developing an efficient algorithm for making decisions in pattern recognition problems. Fuzzy tree automata are defined by Lee [74] for processing fuzzy tree representations of patterns using syntactic recognition. This work shows how membership functions for structural patterns can be defined and how fuzzy language can be used for handling imprecision in structural pattern recognition.

There have been many attempts showing the application of fuzzy set theoretic approaches to real life recognition problems. Some attempts can be found in literature [17, 23, 75] for recognizing speech patterns which are biological in origin and the patterns manifest a considerable amount of fuzziness (vagueness). Pathak and Pal [76, 77] demonstrated an application of fuzzy and fractionally fuzzy grammars in syntactic recognition of ages of different bones from x-ray image patterns. They have shown that incorporation of the concept of fuzziness in defining sharp, fair, and gentle curves and the production rules used enable one to work with a smaller number of primitives and to use the same set of rules and nonterminals at each stage. Furthermore, these grammars need not be unambiguous, whereas nonambiguity is an absolutely necessary requirement for the nonfuzzy approach.

Automatic recognition of handwritten characters is another area where ambiguity occurs because of imprecision in writing rather than from randomness, and the fuzzy set theory has been used quite extensively both in feature extraction and in classification. Some details may be obtained in [23, 78]. Recently, rule based systems have gained popularity in pattern recognition activities. By modeling the rules and facts in terms of fuzzy sets, it is possible to make interfaces using the concept of approximate reasoning. Such a system has been designed for automatic target recognition using forty rules [79].

An extensive discussion on the application of fuzzy k-NN classifier for diagnosing gastric cancer has been provided in [80]. Defining degrees of membership to correspond to the severity of *metastases*, the authors have formulated a fuzzy pattern recognition task with multiple class membership values. Another recent idea in pattern recognition is the partitioning of the initial feature space into regions and the application of different classification rules to them [81, 82]. A partition may be based on the geomet-

ric properties of the classes detected by a preliminary clustering method [81]. In the fuzzy classification rule described in [82], the partitioning is uniform, *i.e.*, the regions continue to be split until a sufficiently high certainty of the rule, generated by each region, is achieved. Ishibuchi *et al.* extended this work in [83] by using an idea of sequential partitioning of the feature space into fuzzy subspaces, until a pre-determined stopping criterion is satisfied, and studied its application for solving various pattern classification problems.

A multivalued approach for supervised classification has recently been developed by Pal and Mandal [84] and Mandal *et al.* [85], based on approximate reasoning where the system can accept imprecise input in linguistic form and provide output in multiple states. The feature space is decomposed by using linguistic property [84] and geometrical structures [85] of training samples. The performance of the algorithms has been demonstrated on both speech recognition problem and analysis of satellite imagery for detecting man-made objects [86]. The concept of multistate decision is found to be effective in connecting road-like structures. The theoretical analysis of the methods including convergence property and relation with Bayes' decision regions has also been studied [87]. The concept of determining multiclass (fuzzy) boundary and shape of a pattern class from its sampled points (training samples) has been introduced in another study of Mandal *et al.* [88], in order to avoid committing oneself to a specific determination of boundary.

Besides the above-mentioned supervised classification methods, fuzzy set theory has been extensively used in clustering problems where the task is to provide class labels to input data (partitioning of feature space) under unsupervised mode based on certain criterion. A seminal contribution to cluster analysis was Ruspini's concept of a fuzzy partition [60]. A new direction in this line was initiated by Bezdek and Dunn in their work on fuzzy ISODATA and the fuzzy c-means algorithms [89, 90]. Another important branch called fuzzy image processing also grew up in parallel, based on the realization that many of the basic concepts in pattern analysis, e.g., the concept of an edge or a corner, do not lend themselves to precise definition. Applications of fuzzy set theory to unsupervised classification (clustering), feature extraction and image processing have not been discussed here, as they do not fall within the interest of this thesis. Readers may consult [23, 89],[91]-[93] for further details on these aspects.

1.3 Expert systems : some problems and the utility of fuzzy sets

The major components of an expert system [8, 94] are the *knowledge base, inference engine* and *user interface.* The knowledge base contains the expert-level information necessary to solve problems in a specific domain. This information is generally represented in the form of a set of rules, although frames [95], semantic nets [96, 97] and belief networks [98] are also in vogue. We shall consider rule-based systems in this discussion. Knowledge bases, being domain-specific, are nontransferable. The inference engine interacts both with the knowledge base and a *working memory* (that records *facts* about the current problem and is updated with the availability of new information). Pattern matching occurs between the *rules* in the knowledge base and the *facts* in the working memory to select the relevant rules applicable. Note that when no matching occurs, no rule is selected, whereas when multiple rules apply, *conflict resolution* strategies are used to select the *most specific* one. The same inference engine can be used with different knowledge bases. The expert system should be able to justify its reasoning since in some domains, like medicine, a doctor must accept the ultimate responsibility for a diagnosis even if it was arrived at with considerable help from a program.

1.3.1 Involvement of uncertainties

The knowledge base itself is a major source of uncertain information [99] in expert systems, the causes being :

- unreliable information,
- imprecise descriptive languages,
- inferencing with incomplete information, and
- poor combination of knowledge from different experts.

Firstly, ill-defined domain concepts or inaccurate data result in unreliable information leading to weak correlation between a rule's premise and its conclusion. *Mycin* [100]

uses numeric *certainty factors* to quantify the degree of this correlation. Secondly, the numerous ambiguities in natural language often result in imprecise expression of rules in formal language. Therefore the meanings of the facts have to be *approximately* matched with those of the premises. Thirdly, when the available information is incomplete the system accepts the value *unknown* while evaluating the premise's degree of certainty during the approximate pattern matching. Finally, generation of a *consensus* knowledge base by combining the views of multiple experts is often difficult, especially when the experts have contradictory viewpoints. Moreover, problems occur as human beings generally do not have uniform levels of expertise throughout a domain. Some conflict resolution strategies are followed, like (say) separately weighting the knowledge of each expert, to generate a composite conclusion.

Good expert systems are expected to be capable of handling uncertainty, as in most cases the data are inherently inexact, incomplete or imprecise. Some of the existing methods used for dealing with such uncertainties are : the subjective probability theory, Dempster-Shafer theory [101], possibility theory [13] and certainty factors [100]. Using the subjective Bayesian method, *Prospector* [102] has been designed to provide advice on mineral exploration. *Mycin* is a rule-based expert system (using certainty factors) [103] which attempts to recommend appropriate therapies for patients with bacterial infections. It interacts with the physician to acquire the clinical data it needs. The associated *Teiresias* system enables the doctors to interact with *Mycin* to ask it questions about its reasoning and to modify and augment its knowledge base. In most of these systems, uncertainty is dealt with a combination of predicate logic and probability-based methods. A serious shortcoming of these techniques is that they are not capable of handling the pervasive fuzziness of information in the knowledge base, and are mostly *ad hoc* in nature [37].

The knowledge base of an expert system contains *human* knowledge, most of which is imprecise and qualitative. To describe situations where the boundary between competing hypotheses is vaguely defined, human experts use terms such as *very likely, likely, more or less likely, low, medium, high*, etc. Encoding this sort of expertise by probabilities results in the loss of information about this *fuzziness* or imprecision. Fuzzy logic may be used to express these vague terms. Generally linguistic variables are used here for the representation of the experts' knowledge or rules. Such representation schemes enable a knowledge engineer to capture the essence of the experts' experience

and judgement without attempting to over-quantify intuition. Moreover, facts about the world are rarely known with certainty. Conventional rule-based systems, with two-valued logic, usually evade this issue of partial matching.

1.3.2 The role of fuzzy logic

The importance of fuzzy logic to the management of uncertainty in expert systems mainly lies in its ability for dealing with fuzzy quantifiers and modifiers. Fuzzy logical systems allow a proposition or conclusion to range over fuzzy subsets (like *very true, more or less true, likely true*, etc.) of truth-value sets characterised by their possibility distributions. Fuzzy modifiers like *not, very, more or less, extremely, slightly, much, a little,* etc. can also be represented. A fuzzy *certainty factor* is associated with the conclusion to analyse the transmission and cumulation of uncertainty from the premises to the conclusion. Deduction of conclusions from observations and rules in the knowledge base are made using either *truth value restriction* or *compositional rule of inference*. Hence, partial match can occur between the antecedent of a rule and a fact supplied by the user.

In short, fuzzy logic or approximate reasoning [37] provides a natural conceptual framework for knowledge representation and inferencing from knowledge bases that are imprecise, incomplete or not totally reliable. The advantage of using fuzzy reasoning is that it can yield an approximate answer even when probabilistic theories are not applicable, as the latter often require idealized assumptions such as the independence of evidence and the mutual exclusiveness and exhaustiveness of hypotheses.

The various approaches in fuzzy inferencing for traditional expert systems include the approximate analogical reasoning based on similarity measures by Turksen and Zhong [104], the problem-reduction method of Ishizuka *et al.* [105], modeling of physicians' decision processes by Esogbue and Elder [106] and inferencing in the framework of *inflammatory protein variations* by Sanchez and Bartolin [107] (using weighting). Wang and Mendel [108] developed a slightly different method for creating a fuzzy rule base made up of a combination of rules generated from numerical examples and linguistic rules supplied by human experts. The input and output domain spaces are divided into a number of linguistic subspaces. Human intervention is sought to assign degrees to the rules, and conflicts are resolved by selecting those rules yielding the maximum

of a computed measure corresponding to each linguistic subspace. Some mathematical analysis is also provided.

1.3.3 A few other problems

Note that the various uncertainty management schemes of traditional expert models share some common problems. For example, a willing human expert able to accurately quantify expertise is needed. The transfer of the knowledge takes place gradually through many interviews between the expert and the system and is therefore very time consuming. Usually humans are prone to be easily biased and thus the quality of knowledge extracted from the experts depends greatly on the methods used for assessment. Moreover, large knowledge bases need to be searched quickly and it is also very important to check that this knowledge base remains consistent as more information is accummulated. It would therefore be welcome if knowledge assessment could be automated by freeing it from human intervention, thereby avoiding human bias and subjectivity.

It is worth mentioning that the most difficult, time-consuming and expensive task in building an expert system is constructing and debugging its knowledge base. In practice, the knowledge base construction can be said to be the *only* real task in building an expert system considering the proliferating presence of *expert shells*. Several approaches have been explored for easing this knowledge-acquisition bottleneck. However, in the process of overcoming these problems one should also try to incorporate the basic characteristics of expert systems, *viz.*,

- capability of dealing with non availability of data, and enquiring the user for additional data when necessary, and
- capability of explaining and justifying solutions or recommendations to convince the user that its reasoning is, in fact, correct.

In the medical domain, for instance, data may be missing for various reasons, *viz.*, some examinations can be risky for the patient or contra-indications can exist; an urgent diagnostic decision may need to be made and some very informative but prolonged test results may have to be excluded from the feature set; or, appropriate technical equipment may not be available. Besides, the final responsibility for any diagnostic

decision has to be accepted by the medical practitioner. So the doctor may want to verify the justification behind the decision reached, based on her expertise. This requires the system to be able to explain its mode of reasoning for any inferred decision.

1.4 Connectionist approach to pattern recognition and expert systems

During the mid 1950's and early 1960's a class of machines called perceptrons, proposed by Rosenblatt [109, 110], seemed to offer what many researchers thought was a natural and powerful model of machine learning. However, interest in these tapered off when Minsky and Papert [111] proved that the simple single-layer networks were not capable of discriminating between linearly nonseparable classes. Work continued on linear and piecewise linear machines, providing the mathematical foundation for further research. In 1986, Rumelhart, Hinton and Williams [112] presented the generalized delta rule which provided a practicable means of training even multilayer perceptrons, and removed the one major stumbling block. The advent of this algorithm rejuvenated a stagnating area of research and lead to a veritable surge of interest in neural network models.

Artificial neural networks (ANNs) [39]-[41],[113, 114] attempt to replicate the *computational* power (low-level arithmetic processing ability) of biological neural networks and, thereby, hopefully endow machines with some of the (higher-level) *cognitive abilities* that biological organisms possess (due in part, perhaps, to their low-level computational prowess). These networks are reputed to possess the following basic characteristics.

- adaptivity : the ability to adjust the connection strengths to new data/ information,
- speed : due to massive parallelism,
- robustness : to missing, confusing and/or noisy data,
- ruggedness : to failure of components, and
- optimality : as regards the error rates in performance.

The various models are designated by the network topology, node characteristics, and the status updating rules. Network topology refers to the structure of interconnections among the various nodes (neurons) in terms of layers and/or feedback or feedforward links. Node characteristics mainly specify the operations it can perform, like summing the weighted inputs incident on it and then amplifying or applying some aggregation operators on it. The updating rules may be for weights and/or states of the processing elements (neurons). Normally an objective function, representing the status of the network, is defined such that its set of minima correspond to the set of stable states of the network.

The networks can be trained by examples (as is often required in real life) and sometimes generalize well for unknown test cases. The worthiness of a network lies in its inferencing or generalization capabilities over such test sets. Connectionist learning procedures are suitable in domains with several graded features that collectively contribute to the solution of a problem. In the process of learning, a network may discover important underlying regularities in the task domain. The network architecture is selected depending upon the objective of the problem to be tackled. Typically the models are based on parallel and distributed working principles, *i.e.*, all neurons work independently and in parallel.

A brief review on the application of connectionist models to the fields of pattern recognition, inferencing and rule generation (connectionist expert systems) will be provided in this section.

1.4.1 Connectionist pattern recognition

For any pattern recognition system, it is always desirable for a model to possess characteristics such as adaptivity, speed, robustness, ruggedness and optimality. As these are easily possible with neural network models, the usefulness of such networks becomes evident. Moreover, there exists some direct analogy between the working principles of many pattern recognition tasks and neural network models.

The task of pattern recognition in real life problems involves searching a complex decision space. This becomes more complicated particularly when there is no *a priori* information on class distribution. Neural network based systems use adaptive learning procedures, learn from examples and attempt to find a relation between input and

output, however complex it may be, for decision making problems. Neural networks are also reputed to model complex nonlinear boundaries and in discovering important underlying regularities in the task domain. These characteristics demand that methods are needed for constructing and refining neural network models for various recognition tasks. For example, consider the case of supervised classification. Here each pattern is characterized by a number of features. Different features usually have different amount of weightage in characterizing the classes. A collective decision, taking into account all the features, is made for assignment of class labels to an input. A multi-layer perceptron in which the input layer has neurons equal to the number of features and the output layer has neurons equal to the number of classes, can therefore be used to tackle this classification problem. Here the importance of different features will automatically be encoded in the connection links during training. The nonlinear decision boundaries are modeled and class labels are assigned by taking collective decisions.

Some of the commonly known neural networks generally used for pattern recognition and optimization tasks include the single-layer perceptron [110, 115], Boltzmann machine [116, 117], multi-layer perceptron using back propagation of errors [112, 118, 119], Hopfield nets [120, 121], Hamming nets [40], Grossberg/Carpenter nets (ART) [122]-[124], Kohonen's self-organizing feature maps [41, 125, 126], Counterpropagation networks [127, 128] and Neocognitron [129, 130]. The Hopfield and Hamming nets are primarily used with fixed weights for optimization tasks while the single and multi-layer perceptrons undergo supervised learning, basically for classifier design. On the other hand, the Grossberg/Carpenter net and Kohonen's feature map perform unsupervised learning or clustering. In addition, the Kohonen's net has a good application to feature analysis due to its self-organizing ability. Combinations of the different neural algorithms have also been tried to design connectionist systems [131, 132] for pattern recognition. More details about the multi-layer perceptron (MLP) and Kohonen's net models are provided in Chapters 2 and 3 respectively.

Applications of the various neural network models have been made in diverse spheres of pattern recognition. A good survey can be found in [133]-[135]. Some of the notable areas encompass

- invariant recognition schemes [136, 137],
- partially occluded object recognition [138],

- applications to text recognition [139, 140],
- character recognition [141, 142],
- speech recognition [143]-[145],
- recognition of hand gesture [146],
- sonar target recognition [147],
- knowledge representation [148], and
- image processing and object recognition [149]-[153].

Some studies on the comparison of the abilities of neural nets with conventional pattern recognition techniques have been provided by Huang and Lippmann [154], Barnard and *Casasent* [155], *Bounds et al.* [156], Chen [157] and Sethi and Jain [158]. Efforts at improving the speed of convergence and/or efficiency of the back propagation algorithm were reported by Silverman and Noetzel [159], Franzini [160], Chan and Fallside [161], Jacobs [162] and Tollenaere [163]. Some other recent approaches in this direction include the works of Bello [164] (using nonlinear least squares and quasi-Newton optimization techniques), *Wessels and Barnard [165] (by properly initializing connection* weights), Sakaue *et al.* [166] (incorporating a weighted error function), Hwang *et al.* [167] (employing network inversion) and Drucker and Le Cun [168] (utilizing double backpropagation).

Automated generation of appropriate/ optimal neural network topology have been investigated by Fahlman *et al.* [169] (employing the cascade-correlation learning algorithm), Sietsma and Dow [170] (by pruning and/or adding units), Geva and Sitte [171] (by combining local functions, selected using a nearest neighbor classification technique), Bauer and Pawelzik [172] (utilizing a topographic product to indicate perfect neighborhood preservation of the optimal self-organizing feature maps), Zollner *et al.* [173] (by systematically collecting patterns into smaller subsets), and Smyth [174]. However *completely satisfactory solutions* to these issues, concerning the optimal design of the networks for particular applications, are yet to be offered. This is mainly because of the large space of possible network architectures that need to be probed in the process of generating an optimal structure. Besides, the problem to be solved and the constraints on the solutions, *viz.*, fast learning, low connectivity and

high accuracy, severely restrict such design. In addition, a large number of variables interact in a complex manner in a neurocomputing system.

Mathematical and/or geometrical analysis of the processes involved in decision making as well as experimental studies on the performance of various network structures have been adequately reported in literature [175]-[187]. Some work on the hardware implementation of the various neural network models can also be found in [188]-[191]. Parallel versions of the different algorithms involved have been implemented in [192, 193].

1.4.2 Connectionist expert systems

Expert systems having connectionist networks as their knowledge bases are called connectionist expert systems [10]. These models seem to be capable of overcoming the problem of the *knowledge acquisition bottleneck* of traditional expert systems. Connectionist expert systems are usually suitable in data-rich environment. They help in minimizing human interaction and associated inherent bias during the phase of knowledge base formation (which is time-consuming in the case of traditional models) and also reduce the possibility of generating contradictory rules. Powerful learning techniques exist for generating connectionist networks from training samples. This enables us to automate the construction of knowledge bases for *classification-type* expert systems.

Connectionist expert systems offer an alternative approach both to the knowledge base construction as well as the inferencing phase, providing interaction with the user accompanied by justification(s) of the conclusion(s) reached. Rules are not required to be supplied by humans. Instead, the connection weights encode among themselves, in a distributed fashion, the information conveyed by the input-output combinations of the training set. The problems faced by traditional expert systems regarding the difficulties in normalizing across different experts' scales, conversion from human expressions to numerical terms, bias of the expert(s), generation of contradictory rules by the experts, etc., may be overcome here. The use of the learning technique of connectionist networks enables the model to extract the information inherent in the data (that is not utilised in the traditional models) and allows dynamical adjustments to changes in the environment. It also enables one to handle a complicated environment for which

either no mathematical model exists, or, even if it exists is so strongly nonlinear that a design method does not exist. Besides, the various characteristics of neural nets, *viz.*, generalization, tolerance to noise, graceful degradation at the border of the domain of expertise, ability to discover new relations between variables, etc., are in-built and hence can be exploited by the connectionist expert systems.

Here we discuss a few of the existing layered connectionist expert system models (mostly in the medical domain). The model by Gallant [10], dealing with *sacrophagal* problems, uses *crisp* inputs/ outputs and a linear discriminant network (with no hidden nodes) that is trained by the simple *Pocket Algorithm*. The absence of the hidden nodes and nonlinearity limits the utility of the system in modeling complex decision surfaces [40]. Dependency information regarding the variables, in the form of an adjacency matrix, is provided by the expert. Each variable (symptom, disease or treatment) corresponds to some node of the network. The model incorporates inferencing/ forward chaining, confidence estimation, question generation/ backward chaining and explanation of conclusions by *If-Then* rules. In order to generate a rule, the attributes with greater inference strength (magnitude of connection weights) are selected and a conjunction of the premises is formed to justify the output concept.

Yin and Liang [194] employed a *gradually-augmented-node* learning algorithm, with binary inputs and outputs, to incrementally build a dynamic knowledge base capable of both acquiring new knowledge as well as relearning existing information. The rules are explicitly represented among the *condition nodes, rule nodes* and *action nodes* and the algorithm gradually builds the multilayer feedforward network. This connectionist incremental expert model is used as an *animal identification system* whose network structure is changed dynamically according to the new environment or through human intervention.

Saito and Nakano [195] used a multi-layer network to design a medical diagnostic expert system for detecting *headache*. A patient responds to a questionnaire regarding her perceived symptoms and these constitute the input to the network. The doctor is supplied with information regarding possible diagnoses based on the output node values. Relation factors, estimating the strength of the relationship between symptom(s) and disease(s), are extracted from the network and used to help doctors. Rules are generated from the changes in levels of input and output units; the connection weights are not involved in the process. These rules are then used to allow the patient

to confirm the symptoms initially provided by her to the system, in order to eliminate noise from the answers.

1.4.3 Knowledge-based networks

Recently, there have been some attempts in improving the performance of connectionist expert systems using knowledge-based networks which use the domain knowledge to determine the initial structure of the network. Such a model has the capability of outperforming a standard MLP as well as other related algorithms including symbolic and numerical ones. In fact, the work of Yin and Liang [194] (as mentioned before), can be grouped under this category. However, in the absence of knowledge one has to resort to a purely data-driven mode of learning. If the initial knowledge is *strong* then one can simply map it into the neural network. But if this knowledge is *weak* or *incomplete*, additional hidden units and connections need to be added and the network needs to be trained with the data set. When the initial knowledge fails to explain many instances, one is inclined to deduce that the knowledge is weak. The number of additional hidden units may be determined empirically. The initial encoded knowledge may be refined with experience by performing learning in the data environment.

Fu [196] formulated a model where the initial domain knowledge (in terms of rules) is used to generate the network topology, while the links are weighted to maintain the semantics. Hidden units and additional connections are introduced appropriately when the network performance stagnates during training using backpropagation. Weight decay, pruning of weights and clustering of hidden units are incorporated to improve the generalization of the network. The objective of the training algorithm is to encode information more compactly in the hidden layers while maintaining the network performance. The knowledge of the trained neural model is then used to extract the revised rules for the problem domain. Its application to a control problem is also demonstrated.

1.5 Neuro-fuzzy computing for pattern recognition and expert systems

So far we have described various fuzzy set-theoretic and connectionist approaches for pattern recognition and expert system design. For the last few years, researchers all over the world [23],[48]-[53],[197]-[201] have been trying to combine the merits of these approaches under the heading *neuro-fuzzy computing* for building more intelligent decision making systems. The uncertainties involved in the input description and output decision are taken care of by the concept of fuzzy sets while the neural net theory helps in generating the required (linearly nonseparable) decision regions. This section provides a brief review of such models for pattern recognition, inferencing and rule generation.

1.5.1 Neuro-fuzzy pattern recognition

The state-of-the-art for the various techniques of combining neuro-fuzzy concepts involves synthesis at various levels. In general, these methodologies can be broadly categorized as follows.

1. incorporating fuzziness into the neural network framework : fuzzifying the input data, assigning fuzzy labels to the training samples, possibly fuzzifying the learning procedure and obtaining neural network outputs in terms of fuzzy sets [202]-[204];

2. designing neural networks guided by fuzzy logic formalism : designing neural networks to implement fuzzy logic and fuzzy decision making, and to realize membership functions representing fuzzy sets [205]-[213];

3. changing the basic characteristics of the neurons : neurons are designed to perform various operations used in fuzzy set theory (like fuzzy union, intersection, aggregation) instead of the standard multiplication and addition operations [214]-[218];

4. making the individual neurons fuzzy : the input and output of the neurons are fuzzy sets and the activity of the networks involving the fuzzy neurons is also a

fuzzy process [219, 220]; and

5. using measures of fuzziness as the error or instability of a network : the fuzziness/ uncertainty measures of a fuzzy set are used to model the error or instability or energy function of the neural network based system [221].

In the approaches under category 1, the integration can be viewed as incorporating the concept of fuzziness into a neural network framework for building *fuzzy neural network* classifiers. For example, the output of the neurons in the output layer, during both the training and testing phases, can be fuzzy label vectors. Besides, the input can be modeled as some fuzzy properties and the learning procedure can also be fuzzified. In this case the network itself functions as a fuzzy classifier. A good example is the work of Keller and Hunt [202]. They were the first to suggest a way of incorporating the concept of fuzzy sets in the single layer perceptron for pattern recognition applications. A method is described for fuzzifying the labeled target data for training the perceptron. Assignment of membership functions to the label vectors is found to provide a good stopping criterion for linearly nonseparable classes (cases where the classical perceptron usually oscillates). A method was suggested by Kammerer [203] to incorporate known uncertainty of the data in the computational process of neural networks. A measure of certainty is used on each input element in order to modulate the element's contribution to the whole input activity. Some improvements on the classification accuracy have been demonstrated on optical character recognition problems. The technique basically shows an effect of fuzzifying input on the classification accuracy.

Approaches under category 2 deal with neural networks that are used for a variety of computational tasks within the framework of a pre-existing fuzzy model (*i.e.*, implementation of fuzzy logic formalism using neural networks). Huntsberger and Ajjimarangsee [205] were the first to modify Kohonen's network for generating the fuzzy self-organizing feature map. Fuzziness was also incorporated in the learning process by replacing the learning rate with fuzzy membership of the nodes in each class. Further modifications on the rate of learning, shrinking of neighborhood and termination conditions of the algorithm were reported by Bezdek *et al.* [206]. A relationship between the fuzzy version of Kohonen's algorithm and the fuzzy c-means algorithm [22] was also established. A neural network architecture that can be used for fuzzy clustering and classification was suggested in [211]. The system uses a control structure similar to that found in adaptive resonance theory of Carpenter and Grossberg [43]; and em-

ploys a learning strategy, similar to that of the fuzzy c-means algorithm, to update the centroid position of the clusters. Functionally the architecture is similar to the leader clustering algorithm. A supervised neural network classifier that utilizes min-max hyperboxes as fuzzy sets (which are aggregated into fuzzy set classes) was introduced in [212]. The network has a three layered architecture consisting of the input, hidden and output layers. Each hidden layer neuron represents a hyperbox fuzzy set having two types of connections from the input layer representing the *min* and *max* points of the inputs. Learning is a single pass procedure. The model is capable of finding reasonable decision boundaries in overlapping classes and for learning highly nonlinear relations.

Fusion made in category 3 replaces the integration/ transformation operation at each node by the fuzzy aggregation operation (*i.e.*, fuzzy union, intersection, etc.). Pedrycz [222] used logical operators *max* and *min*, with the *crisp* implication operator, for designing two-layered neural nets capable of solving two-class problems. This was extended in [216] to handle multi-class problems using a different performance index. Application of the model for solving a system of relational equations is demonstrated. The concept of reference neurons at the hidden layer, corresponding to the number of clusters, was introduced in [217, 223] to design pattern classifiers. The input to the network consists of the logical combinations of the input variables, and the *Lukasiewicz* implication operator is used. Watanabe *et al.* [224] also used *min-max* operations but with two kinds of weight vectors and a different scheme of backpropagation for a three-layered network.

Another related work in this category is that reported by Krishnapuram and Lee [214]. They used fuzzy aggregation connectives with compensatory behaviour (lying in the range between the two extremes, *viz.*, *min* and *max*) as the activation functions of the neurons. A modified version of backpropagation is used to determine the proper type of the aggregation at each node and its parameters, given an approximate dependency structure of the network. Various union, intersection, generalized mean and multiplicative hybrid operators (which are used in fuzzy sets literature to aggregate imprecise information in order to arrive at a decision in uncertain environments) are implemented by the layered networks. The model of Zimmermann and Zysno [225] is used in the hybrid (compensatory) operator. An iterative algorithm to determine the type of aggregation function and its parameters at each node in the network is also provided; thereby making the network more flexible. The approach provides a tool for

modeling and managing uncertainty in the process of combination of evidence from complementary and supplementary knowledge sources. The technique also provides a mechanism for selecting powerful features and discarding irrelevant features via the detection of redundancy. To achieve faster convergence the additive hybrid operator was studied, under the above framework, by Keller and Chen [218] as an alternative connective in such networks. Recently, Hirota and Pedrycz [226] designed a three layer network topology consisting of two types of generic *Or* and *And* neurons. The logical connectives use standard triangular norms for their realization, whereas their compensatory character is achieved by developing some structural relationships between the layers of the network.

The idea of making the individual neuron fuzzy falls under category 4, and was first promulgated in 1975 by Lee and Lee [219]. Some of the concepts of fuzzy set theory are employed to define a fuzzy neuron, which is a generalization of the classical neuron. The activity of a fuzzy neuron is a fuzzy process. The input to such a neuron is a fuzzy set and the outputs are equal to some positive numbers μ_j's ($0 < \mu_j \leq 1$), if it is firing and zero if it is quiet. μ_j denotes the degree to which the j^{th} output is fired. Unlike conventional neurons, such a neuron has multiple outputs (set). The utility of neural networks with such fuzzy neurons has been demonstrated for synthesizing fuzzy automata. This concept, although introduced long back, has not been explored much as compared to others.

The work of Ghosh *et al.* [221] fall under category 5, where an attempt was made to incorporate various fuzziness measures in a multi-layer network for making it able to perform (unsupervised) self-organizing task in image processing, in general, and object extraction in particular. The network architecture is basically a feed forward one with a feedback path. In each layer every neuron corresponds to an image pixel. Each neuron is connected to the corresponding neuron in the previous layer and its neighbors. The status of neurons in the output layer is described as a fuzzy set representing object regions. A fuzziness measure (*e.g.*, index of fuzziness and entropy [17]) of this set is used to quantify system error (instability of the network) and is backpropagated to correct weights. After the weights have been adjusted the output of the neurons in the output layer is fed back to the corresponding neurons in the input layer. The second pass is then continued with this as input. The iteration (updating of weights) is continued until the network stabilizes, *i.e.*, the error value (measure of fuzziness)

becomes negligible. This integration has made a layered network (which is normally used as a supervised classifier) capable of acting as an unsupervised one, in addition to providing robust noise insensitive segmentation algorithm.

1.5.2 Neuro-fuzzy expert systems

In this section we provide a discussion on the various existing neurofuzzy approaches to inferencing and expert system design. Takagi [227] provided a review on related work in this direction. Fuzzy logic expresses the qualitative reasoning capability of humans and thereby enables easier formulation of substantially smaller number of rules (for the problem under consideration). The membership function that maps the relationship between quantitative data and fuzzy sets may be learned and adapted by training a neural network with real-life data suitable for the environment. Neuro-fuzzy expert systems use fuzzy neural nets for encoding the knowledge base, thereby enabling one to incorporate the advantages of approximate reasoning into the connectionist expert system model.

Inferencing in neurofuzzy systems

The MLP-based approach reported by Keller and Tahani [228] falls under category 2 of the fusion methodologies described (in Section 1.5.1). It receives the possibility distributions of the antecedent clauses at the input, uses a hidden layer to generate an internal representation of the relationship and finally produces the possibility distribution of the consequent at the output. The model is expected to function as an inference engine with each small sub-network learning the functional input-output relationship of a rule. Trapezoidal possibility distributions, sampled at discrete points, are used to represent fuzzy linguistic terms and modifiers. The network is supposed to be able to extrapolate to other inputs (for a rule) following *modus ponens*. Conjunctive antecedent clauses are also modeled using separate groups of hidden nodes for each clause. In [207], Keller *et al.* explicitly encoded the knowledge of each rule among the connection weights of the neural net. A measure of disagreement between the input possibility distribution and the antecedent clause distribution is used at the *clause-checking* and *combination* layers to determine the uncertainty in the consequent part of the *fired* rule.. Theoretical properties of the various combination schemes are also

investigated. Ishibuchi *et al.* [210, 229], on the other hand, used interval vectors to represent fuzzy input and output in a MLP. A modified version of the *backpropagation* algorithm is used to incorporate interval arithmetic. Different fuzzy *If-Then* rules are interpolated from a few sample rules (used during training). Ishibuchi *et al.* [230] also reported learning methods of neural networks for utilizing expert knowledge represented by fuzzy If-Then rules. Both numeric and linguistic inputs are represented in terms of fuzzy numbers and intervals that can be learned by their fuzzy neural network model. Applications to pattern classification and control problems are described.

Keller reported an extension [215] to his earlier work [207, 228]. Note that this approach falls under category 3. The model uses a fixed network architecture which employs parametrized families of operators, like the generalized mean and multiplicative hybrid operators [18, 225, 231]. The hybrid operator can behave as union, intersection or mean operator for different sets of parameters, that can be learned during training. These networks possess extra predictable properties and admit a training algorithm which produces sharper inference results. It is worth mentioning, in this connection, that all the other neuro-fuzzy techniques for inferencing, reviewed here, belong to category 2 (Section 1.5.1) of the fusion methodologies.

Fuzzy inferencing was used by Nakanishi and Takagi [232] for the recognition of noisy patterns (English alphabets distorted by ink marks). An alphabet is divided into n blocks $\vec{X} = (X_1, X_2, \ldots, X_n)$, with each block being processed by a separate neural network. Each block is divided into m smaller subblocks such that the neural network NN_j for block X_j has m inputs $x_j^1, x_j^2, \ldots, x_j^m$. Each NN_j has the number of outputs equal to the number of patterns to be classified and each output is mapped into membership values to the three fuzzy sets *zero, positive medium,* and *positive big.* The fuzzy outputs from the n neural nets are then used to formulate the antecedents of fuzzy inference rules to recognize a noisy pattern when its typical feature is still visible in a small block of the ink-blotched pattern.

An approach to fuzzy inferencing by Zhuang *et al.* [233], for controlling a mathematically intractable system, used a multilayer network with each node standing for a fuzzy subset of a linguistic variable. The truth values flow from the input nodes to the output nodes according to the rule of propagation being modeled. The interconnection weights are associated with ordered pairs of real numbers lying in the range [-1,1] and are indicative of the certainty factors used to represent the reliability of the fuzzy con-

trol rules. The basic operations in fuzzy logic, *viz., union, intersection* and *negation*, are implemented at the neuron level. The linguistic variables used are *positive large, positive medium, positive small, zero, negative small, negative medium* and *negative large.*

The role of fuzzy logic in the control of the activation, training, reliability and performance of neural networks was investigated by Yen [234] using a hybrid architecture. Fuzzy rules are used to detect situations under which certain actions need to be invoked for neural network modules based on their performance measures. The networks process data obtained either from external sensor devices or from the knowledge base of the symbolic system. The global knowledge base consists of a fuzzy database and a neural network taxonomy (of neural net classes) that describes meta-level knowledge about the neural nets themselves (using *crisp* or *fuzzy* attribute values). The fuzzy database stores data and hypotheses that can be uncertain, imprecise or vague. A production system that takes into account the degree of partial matching of the fuzzy action rules thus enables the system to respond in a robust way even in the face of incomplete or noisy data.

Yager [235] employed membership neural modules for the antecedents, inverse membership modules for the consequents and a rule neural module with a combiner (using *min* or *product* functions) for modeling the rules of fuzzy logic controllers. The various weights are learned and the importance of the antecedent clauses simulated.

The neural network based fuzzy reasoning scheme by Takagi and Hayashi [208] was capable of learning the membership function of the *If* part and determining the amount of control in the *Then* part of the inference rules. The input data is clustered to find the best number of partitions corresponding to the number of inference rules applicable to the reasoning problem, a single neural net block modeling one rule. The optimum number of cycles required is determined to avoid *overlearning* and the minimal number of input variables selected for inferring the control values. This was extended in [209] to develop the neural network architecture based on the structure of the fuzzy inference rules involved. The identification error can be analyzed to improve the performance of the structured network. For this, the appropriate region of the feature space is further clustered and the corresponding *Then* parts accordingly added.

It may be noted in this connection that the methods described so far were mainly for rule learning and inferencing. Let us now describe a few neuro-fuzzy expert systems.

Connectionist expert models

Neurofuzzy techniques have recently been used by Isshiki and Endo [236] and Rocha [237, 238] to design expert systems. Hayashi *et al.* [239] showed that any fuzzy expert system may be approximated by a neural network and *vice versa*. An interesting comparison between a connectionist expert system and an *induction type* conventional expert model was provided by Krishnamaraju *et al.* in [240]. It is to be noted that the induction type expert system can also build a rule base from examples. Linguistic approximation of relative preference values is used to represent the consequent part specifying the degree of recommendation of a rule.

A *Distributed single-layer perceptron-based* model using the *Pocket-Algorithm* was used as an expert system for diagnosing *hepatobiliary* disorders by Yoshida *et al.* [241]. Here real-life fuzzy data are *defuzzified* using the *Level Set* representation to produce the *crisp* inputs {+1,−1, 0} required by the algorithm. All contradictory training data are excluded, as these cannot be tackled by the model. The model is used for the extraction of fuzzy *If-Then* production rules. Hayashi [242, 243] extended this to include linguistic relative importance terms like *very important* and *moderately important* in each proposition; linguistic truth values like *completely true, true, possibly true, unknown, possibly false, false* and *completely false* are also assigned by the domain experts. The input layer consists of both fuzzy and crisp cell groups while the output is modeled only by fuzzy cell groups. The crisp cell groups are represented by m cells taking on two values in $\{(+1,+1,\ldots,+1),(-1,-1,\ldots,-1)\}$. Fuzzy cell groups, on the other hand, use binary m-dimensional vectors, each taking on values in $\{+1,-1\}$. Multiple correct pattern classes, using different linguistic truth values, is allowed.

The approach reported by Hudson *et al.* [244]-[246] is for detecting *carcinoma of the lung*. They use a feed-forward neural network model to extract information directly from the accummulated data and then combine it with a rule-based expert system incorporating approximate reasoning techniques. The input nodes represent data values for signs, symptoms and test results (may be continuous or discrete). The interactive nodes account for the interactions which may occur between these parameters. The learning method is an adaptation of the *potential function* approach to pattern recognition and is used to determine the weighting factors as well as the relative strengths of rules for two-class problems.

Sanchez [204] has associated primary linguistic weights and secondary numerical weights to generate the knowledge base for a *biomedical application (inflammatory protein variations)* using a feedforward network. Triangular membership functions like *negative large, negative medium, negative small, approximately zero, positive small, positive medium* and *positive large;* or, *decreased, normal* and *increased* account for the linguistic weights while the quantitative weights lie in the range [0,1]. The linguistic weights are tuned according to the information provided from the input-output examples while the numeric weights and the network topology are determined by solving *fuzzy relation equations.* Note that the aforesaid neuro-fuzzy approaches for connectionist expert system design fall under category 1 of the fusion methodologies (Section 1.5.1).

A *cell recruitment* learning algorithm, capable of forgetting previously learned facts by learning new information, was employed by Romaniuk and Hall [247] to build a fuzzy connectionist expert system for determining the *creditworthiness of credit applicants.* This approach falls under category 3 of the fusion schemes. Fuzzy functions *maximum, minimum* and *negation* are applied at the neuronal levels depending upon the corresponding bias values. This incremental learning algorithm can be used either in conjunction with an existing knowledge base or alone. Learned knowledge can be extracted from the network in the form of rules (during an explanation phase) by a top-down traversal involving analysis of the cell activations, their bias and the associated link weights. The network consists of *positive* and *negative collector cells* along with *unknown* and *intermediate* cells and can handle *fuzzy* or *uncertain* data.

Rhee and Krishnapuram [248] reported a method for rule generation from minimal approximate fuzzy aggregation networks. Note that this model can also be categorized under group 3 of the fusion methodologies. The linguistic labels and the membership functions for the input features are estimated and the network is trained using the gradient-descent technique. Pruning of redundant features and/or hidden nodes helps in generating appropriate rules in terms of And-Or operators that are represented by hybrid functions possessing compensatory behaviour. Applications to pattern recognition and computer vision domains are also provided.

1.5.3 Neuro-fuzzy knowledge-based systems

Some work on neuro-fuzzy approaches to the design of knowledge-based systems have also been recently reported. A brief survey on this field is provided here. Masuoka *et al.* [249] used knowledge in the form of membership functions and fuzzy rules (in *And-Or* form), extracted from experts, to build and preweight the structured neural network which is then tuned using selected learning data. This neural model consists of the input variable membership net, the rule net, and the output variable net. Modified fuzzy rules, extracted from the trained neural network using pruning, are then evaluated and unsuitable rules corrected using relearning. Okada *et al.* [250] used a similar approach to examine the bond rating of investors. Changes in the weights and threshold levels of the neurons (after training) are interpreted as adaptations in the membership functions and fuzzy rules.

Fuzzy signed digraph with feedback, termed fuzzy cognitive map, was used by Kosko [251] to represent knowledge. Additive combination of augmented connection matrices are employed to include the views of a number of experts for generating the knowledge network. It is to be mentioned that all these approaches belong to category 1 of the neuro-fuzzy fusion techniques.

Machado and Rocha [252, 253] used a connectionist knowledge base involving fuzzy numbers at the input layer, fuzzy *And* at the hidden layers and fuzzy *Or* at the output layer. This approach falls under category 3 of the fusion methodologies. The input data, in symbolic or numeric forms, are converted to possibility degrees. The hidden layers chunk input evidences into clusters of information for representing regular patterns of the environment. The output layer computes the degree of possibility of each hypothesis. The initial network architecture is generated using *knowledge graphs* elicited from experts by the application of the knowledge acquisition technique of [254]. The experts express their knowledge about each hypothesis of the problem domain by selecting an appropriate set of evidences and building an acyclic weighted *And-Or* graph to describe how these must be combined to support decision making. Inference, inquiry and explanation are possible during consultation.

Pedrycz and Rocha [255] used basic aggregation neurons (And/Or) and referential processing units (matching, dominance and inclusion neurons) to design knowledge-based networks, employing category 3 of the neuro-fuzzy fusion methodologies. The

inhibitory and excitatory characteristics are captured by embodying direct and complemented input signals. Applications in decision-making, diagnostic and control problems are outlined, employing fully supervised learning. Another related approach by Hirota and Pedrycz [256] incorporated the use of fuzzy clustering for developing the geometric constructs leading to the design of knowledge-based networks. Its application for solving classification problems is also described. ♣

Before describing the scope of the thesis, we provide a brief discussion on genetic algorithms as a recent tool for solving various pattern recognition problems.

1.6 Promise of genetic algorithms

Like neural networks, genetic algorithms (GAs) [257, 258] are also based on powerful metaphors from the natural world. They mimic some of the processes observed in natural evolution, which include cross-over, selection and mutation, leading to a stepwise optimization of organisms. GAs try to apply these principles to technical problems.

From the optimization point of view, GAs are highly parallel, robust and adaptive search processes which generally lead to approximately global optimal solutions even along multidimensional and multimodal surfaces. They consider many points in the search space simultaneously, and hence have less chance of converging to local optima. Given a fitness function and some relations among objects and classes, a GA can search the space to find the best solution in determining a class to be associated most appropriately with each object. Various attempts have been made on the application of GA's to different fields like classification, segmentation, image enhancement, primitive extraction, membership function selection and tuning [259]-[261]. Genetic algorithms have also been applied to the problem of neural network design in order to generate the optimal topology and for improving convergence, for avoiding local minima during learning by MLP, in cloning templates, and also in integrating neuro-fuzzy concepts [262]-[271]. A good review on the integration of GAs and connectionist models is provided by Yao [272].

1.7 Scope of the thesis

The objective of this thesis is to present some results of investigation that demonstrate the effectiveness of the neuro-fuzzy approach by developing some new models for pattern classification, rule generation and inferencing. The connectionist systems can handle uncertainty at the input feature level and provide output decision in terms of class membership values. Both the multilayer perceptron (as a fully supervised classifier) and the Kohonen's net (as a partially supervised classifier) have been used in this regard. The models are capable of querying the user in the case of incomplete or partial inputs and can generate rules to justify an inferred decision. The logical operators *And* and *Or* have also been used in the neuronal level of the aforesaid fuzzy MLP. Note that the integration (Section 1.5.1) made here falls under category 1 in the case of fuzzy MLP and fuzzy Kohonen's net of Chapters 2-5 [273]-[276] and a combination of categories 1 and 3 in the case of fuzzy logical MLP of Chapter 6 [277].

The effectiveness of these models is demonstrated on both synthetic data and real life problems, such as speech recognition and medical diagnosis. In some cases, the results are compared with those obtained using conventional/ classical systems. The results of the investigations are summarized below under different chapter headings.

1.7.1 Multilayer perceptron, fuzzy sets and pattern classification [273,278]

One of the most exciting developments in the early days of pattern recognition was the multilayer perceptron (MLP) which consists of multiple layers of simple sigmoid processing elements (nodes) or neurons that interact using weighted connections. The MLP discovers the input-output mapping in a sequence of forward and backward passes through the training set, thereby efficiently modeling complex decision regions.

Chapter 2 describes a fuzzy version of the multilayer perceptron, using the *back-propagation* algorithm. It describes a method of integrating the uncertainty handling capability of fuzzy set theory with the ability of layered neural networks in generating highly nonlinear decision boundaries to design an intelligent system for pattern recognition. The system can accept input features in linguistic form (low, medium and high), and can provide confidence values to soft (fuzzy) decision for ambiguous

patterns. The concept of fuzzy sets has been exploited in representing linguistically phrased input features, both for their processing as well as in neural learning. The output vector for a sample point coming from overlapped regions is not committed to any particular class (but given unconstrained memberships to more than one class), and this helps to reduce oscillations of the decision boundaries caused by patterns from overlapping regions. During training, the *learning rate* is gradually decreased in discrete steps. This also helps the network in reducing oscillations in the process of convergence to a minimum error solution. Minimization of both the mean square error and the cross entropy are studied to demonstrate their effect on the performance of the model.

The merits of this neuro-fuzzy integration have been demonstrated on speech recognition problem where the classes have significant amount of overlapping, and the features (formant frequencies) are sometimes linguistic and imprecise/ incomplete in nature. The performance of the system has been compared with those of the conventional MLP, the Bayes classifier and other related models. Effect of different numbers of hidden layers, hidden nodes and training set size, on the performance of the model has also been investigated.

1.7.2 Kohonen's net, fuzzy sets and pattern classification [274]

Self-organization by a neural net refers to the ability to create abstractions and symbolic representations by properly organizing itself on its own, so that the connection weights get appropriately updated in the process. In Kohonen's model [41], the network automatically performs a mapping transformation from an input space to generally a lower dimensional output space such that the latter acquires the same topological ordering as the former.

Chapter 3 presents a fuzzy version of a self-organizing Kohonen's model, that is capable of handling fuzzy input and providing *fuzzy* classification of patterns. Like the fuzzy MLP of Chapter 2, this algorithm also incorporates fuzzy set-theoretic concepts at various stages. The input vector consists of membership values for *linguistic* properties along with some *contextual class membership* information which is used during self organization to permit efficient modeling of *fuzzy* (ambiguous) patterns. A new

definition of gain factor for weight updating is provided. An *index of disorder* involving mean square distance between the input and weight vectors is used to determine a measure of the *ordering* of the output space. This controls the number of sweeps required in the process. Incorporation of the concept of *fuzzy partitioning* allows natural self organization of the input data, especially when they have ill-defined boundaries.

The output of unknown test patterns is generated in terms of class membership values. Incorporation of fuzziness in input and output is seen to provide better performance as compared to the original Kohonen's model and its hard version. The effectiveness of this algorithm is demonstrated on the speech data for various network array sizes, training sets and gain factors. It is worth mentioning that while the fuzzy MLP-based model (Chapter 2) is fully supervised, the fuzzy Kohonen's net-based model admits of partial supervision with some contextual class information (embedded in the input vector) during learning.

1.7.3 Applications of fuzzy MLP and fuzzy Kohonen's model to linearly nonseparable nonconvex pattern classes [275]

An attempt is made in Chapter 4 in demonstrating the ability of the fuzzy versions of the MLP and the Kohonen's net (described in Chapters 2 and 3 respectively) for classification of certain linearly nonseparable pattern classes with nonconvex decision regions. A comparative study is also provided. Representing the input information in terms of *low, medium* and *high* enables these models to utilize more local information of the feature space, while the neural net theory helps in generating the required concave and/or disconnected decision regions. Superiority of these fuzzy models (over the respective conventional connectionist versions, the k-nearest neighbors classifier and seven other existing neural algorithms) has been adequately established when they are implemented on three sets of linearly nonseparable pattern classes. Effect of fuzzification at the input membership functions has been investigated for both the models. The contribution of the *a priori* probabilities of the pattern classes in the backpropagation procedure for weight updating has also been studied.

1.7.4 Rule generation and inferencing using fuzzy MLP and fuzzy Kohonen's models [276,279,280]

In Chapter 5 we develop some methodologies based on fuzzy version of the MLP (Chapter 2) and the fuzzy Kohonen's net (Chapter 3), for inferencing and rule generation. This leads to the design of fuzzy connectionist expert systems.

Based on the output class membership value(s) of unknown patterns, the networks can generate some measure of certainty expressing confidence in the decision. In the case of partial inputs they are capable of querying the user for *important* input feature information, if and when required. Justification for an inferred decision may be produced in rule form, when so desired by the user. The magnitudes of the connection weights of the *trained* neural network are utilised in every stage of the inferencing procedure. The antecedent and consequent parts of the justificatory rules are provided in *natural* forms. Note that the input can be in quantitative, linguistic or set forms or a combination of these.

The algorithms allow selection of those *currently active* input neurons, which contribute *most* to the final conclusion, as the clauses of the antecedent part of a rule. They also enable the *currently active* test pattern inputs (current evidence) to influence the generated *knowledge base* (connection weights learned during training) in producing a rule to justify the *current* inference (decision regarding the current test pattern). This helps extracting rules relevant to that region of the feature space which is local to the area pointed by the feature values of the test pattern. The service of a human expert may be used at this stage to verify the logic of the rule generated by the model in support of the currently inferred decision. This may help in building the confidence of the user in the efficiency of the system. These rules could also be used as the knowledge base for a traditional classification-type expert system for the problem under consideration.

The effectiveness of the algorithms is tested on the speech data and the three sets of artificially generated data consisting of linearly nonseparable pattern classes.

1.7.5 Use of logical neuron in fuzzy MLP for rule generation and inferencing [277,281,282]

A fuzzy layered neural network for classification and rule generation using *logical neurons* is described in Chapter 6. The input and output representations are the same as that of the fuzzy MLP (Chapter 2). Instead of the standard weighted sum and sigmoid functions, the system uses the logical *And* and *Or* operators. The conventional *backpropagation algorithm* is accordingly modified to incorporate these operators. Two cases of the conjugate pairs of *t-norm* T and *t-conorm* S, *viz.*, *max-min* and *product-probabilistic sum*, are utilised to model the *And* and *Or* operations. The hidden layer consists of *And* nodes while the output layer is made up of *Or* nodes. Fuzzy implication operators are employed to incorporate various amounts of mutual interaction during error propagation.

Unlike the methods described in Chapter 5, the built-in *And-Or* structure of the network enables the generation of appropriate rules expressed as the disjunction of conjunctive clauses. This is specially true in case of problem domains that can be represented in terms of *And-Or* combinations of the features. (However, it has been observed that the fuzzy MLP (using sigmoidal nonlinearities) usually performs better for classification). The effectiveness of the model is tested on the speech data and on some artificially generated pattern sets.

1.7.6 Application of the fuzzy MLP to medical diagnosis [276,283]

Some applications of the fuzzy MLP, for both classification and rule generation, have been provided in Chapters 2, 4 and 5 on synthetic and speech data. In Chapter 7 we demonstrate two rigorous applications of the fuzzy MLP in a different domain, *viz.*, medical diagnosis of *hepatobiliary disorders* and *kala-azar*. In order to handle the skewed distribuibution of the data set on *hepatobiliary disorders*, the input to the network is modified. The input vector is modeled in terms of linguistic π-sets, whose centres and radii along each feature axis are automatically determined from the distribution of the training data. It is observed that for more complex feature spaces (*e.g.*, medical diagnosis) it is appropriate to employ the fuzzy MLP (with no

logical operators) for producing more accurate classification and also better inferencing and/or rule generation. A comparative study of the performance of the model with that of Hayashi's network [242] is also provided.

1.7.7 Conclusions and scope for further work

The concluding remarks with further scope of research are presented in Chapter 8.

Chapter 2

Multi-Layer Perceptron, Fuzzy Sets and Pattern Classification

2.1 Introduction

This chapter presents a fuzzy version of the multilayer perceptron (MLP) [273, 278] using the gradient-descent based backpropagation algorithm [39] by incorporating concepts from fuzzy sets at various stages. Unlike the conventional system, this model is capable of handling input available in linguistic form. Besides, conventional two-state neural net models generally deal with the ideal condition where any input feature is either present or absent, and any pattern belongs to either one class or another. They do not consider cases where an input feature may possess a property with a certain degree of confidence, or a pattern may belong to more than one class with a finite degree of belongingness. The fuzzy MLP model incorporates these concepts and is capable of classification of fuzzy patterns. In other words, this investigation provides a way of integrating the merits of fuzzy set theory in handling uncertainty in input and the ability of MLP in generating highly nonlinear decision boundaries for designing an efficient system for classification.

Broadly the network passes through two phases, *viz.*, training and testing. During the training phase supervised learning is used to assign output membership values lying in the range [0,1] to the training vectors. Hence each output class (node in the output layer) may be assigned a non-zero membership instead of choosing the single class (node) with the highest activation. This allows modeling of fuzzy data when the feature space involves overlapping pattern classes such that a pattern point may belong to more than one class with non-zero membership.

During training, each error in membership assignment is fed back and the connection weights of the network are appropriately updated. The backpropagated error is computed with respect to each desired output which is a membership value denoting the degree of belongingness of the input vector to that class. Hence the error (which is backpropagated for weight updating) has inherently more weight in case of nodes with higher membership values. The contribution of ambiguous or uncertain vectors to the weight correction is automatically reduced. This is natural as vectors that are more *typical* of their class should have more influence in determining the position and shape of the decision surface.

The utility of this approach to model output values may be further appreciated by considering a point lying in a region of overlapping classes in the feature space.

In such cases its membership in each of these classes may be nearly equal. Then there is no reason why we should follow the *crisp* approach of classifying this pattern as belonging to the class corresponding to that output neuron with a slightly higher activation, and thereby neglect the smaller yet significant response(s) obtained for the other overlapping class(es).

After a number of cycles the neural net may converge to a minimum error solution. The network now encodes the input space information in its connection weights. In the second phase, a part of the same fuzzy data (kept aside for testing during random selection of the training set) is applied as input and the network performance verified at the output. Note that the output is obtained in terms of fuzzy membership values to the various pattern classes. A confusion matrix is generated to evaluate the classification efficiency of the neural network. Here each test data contributes a count at a particular position in this matrix, its row corresponding to the class to which it belongs (as determined from the *hard* labels attached to the input data set) and its column indicating the class corresponding to the neuronal output providing the *best match.*

The fuzzy neural network model is capable of handling input features presented in quantitative and/or linguistic form. The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties *low, medium* and *high* corresponding to each input feature. This provides scope for incorporating linguistic information in both the training and testing phases of the said model and increases its robustness in tackling imprecise or uncertain input specifications. It is to be noted that the impreciseness (or ambiguity) at the input may arise from various reasons. For example, instrumental error or noise corruption in the experiment for computing a feature, or the high expense incurred in extracting its precise/ exact value, may lead (as mentioned in Section 1.2.2) to such uncertainties. In these situations, it becomes convenient to use the linguistic variables (*low, medium* and *high*) in order to describe the feature information.

During training, the learning rate and the damping coefficient are gradually decreased until the network hopefully converges to a minimum error solution. This heuristic helps to avoid spurious local minima and usually prevents oscillations of the mean square error in the weight space.

The effectiveness of the model is demonstrated on a speech recognition problem

where the classes have ill-defined, fuzzy boundaries. A comparison is made with the standard Bayes' classifier and the conventional multilayer perceptron, and the performance of the fuzzy MLP model is found to be better.

2.2 Multi-layer perceptron using backpropagation of error

The multilayer perceptron (MLP) [39] consists of multiple layers of simple, two-state, sigmoid processing elements (nodes) or neurons that interact using weighted connections. After a lowermost input layer there are usually any number of intermediate or *hidden* layers followed by an output layer at the top. There exist no interconnections within a layer while all neurons in a layer are fully connected to neurons in adjacent layers. Weights measure the degree of correlation between the activity levels of neurons that they connect.

An external input vector is supplied to the network by clamping it at the nodes in the input layer. For conventional classification problems, during training, the appropriate output node is clamped to state 1 while the others are clamped to state 0. This is the desired output supplied by the *teacher*.

Consider the network given in Fig. 2.1. The total input x_j^{h+1} received by neuron j in layer $h+1$ is defined as

$$x_j^{h+1} = \sum_i y_i^h w_{ji}^h - \theta_j^{h+1} \tag{2.1}$$

where y_i^h is the state of the i^{th} neuron in the preceding h^{th} layer, w_{ji}^h is the weight of the connection from the i^{th} neuron in layer h to the j^{th} neuron in layer $h+1$ and θ_j^{h+1} is the threshold of the j^{th} neuron in layer $h+1$. Threshold θ_j^{h+1} may be eliminated by giving the unit j in layer $h+1$ an extra input line with a fixed activity level of 1 and a weight of $-\theta_j^{h+1}$.

The output of a neuron in any layer other than the input layer ($h > 0$) is a monotonic non-linear function of its total input and is given as

$$y_j^h = \frac{1}{1 + e^{-x_j^h}} \tag{2.2}$$

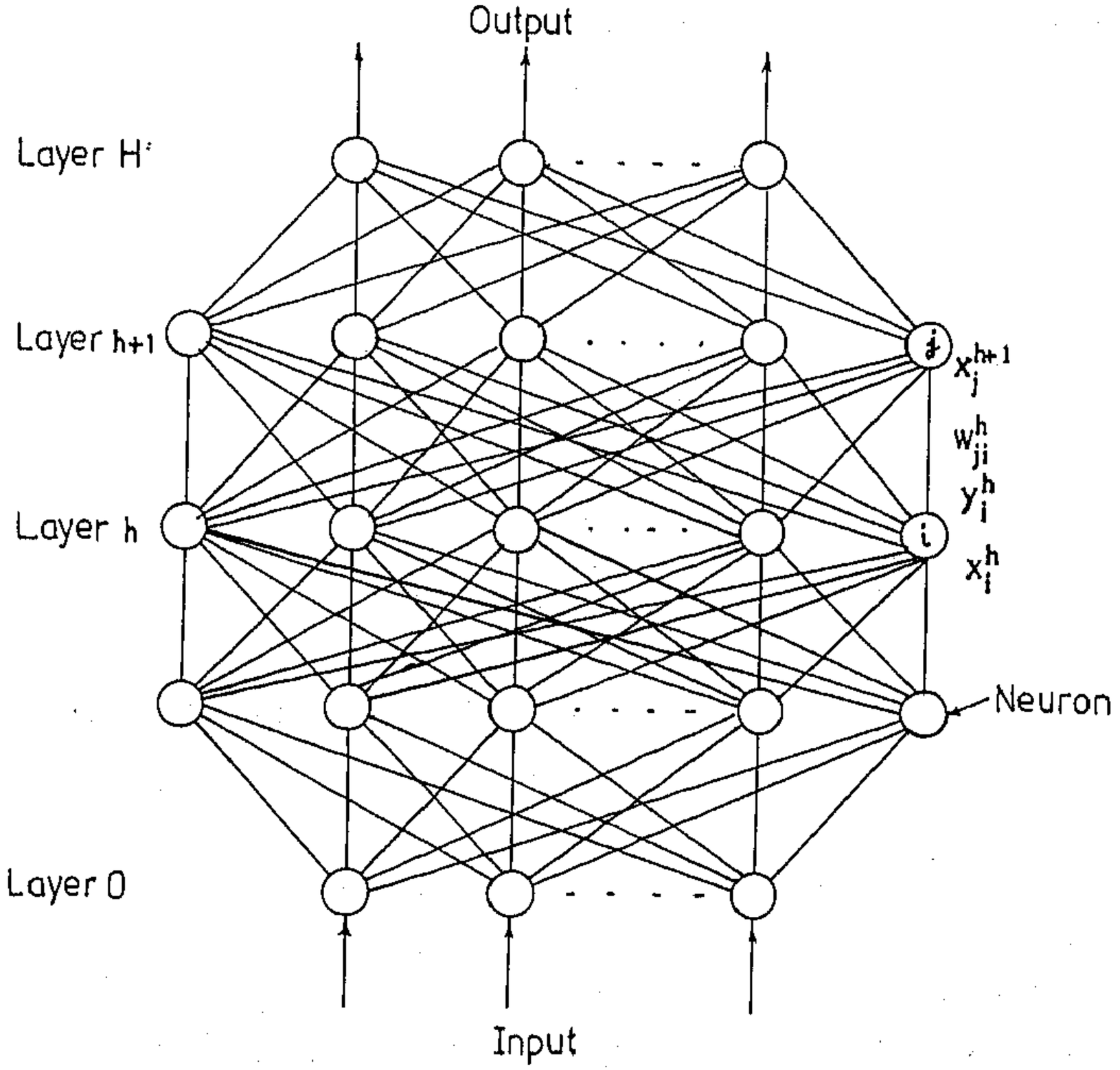

Figure 2.1: A neural network with three hidden layers

For nodes in the input layer

$$y_j^0 = x_j^0 \tag{2.3}$$

where x_j^0 is the j^{th} component of the input vector clamped at the input layer. All neurons within a layer, other than the input layer, have their states set by eqns. (2.1)-(2.2) in parallel while different layers have their states set sequentially in a *bottom-up* manner until the states of the neurons in the output layer H are determined. The learning procedure has to determine the internal parameters of the hidden units based on its knowledge of the inputs and desired outputs. Hence learning consists of searching a very large parameter space and therefore is usually rather slow.

The Least Mean Square (LMS) error in output vectors, for a given network weight vector $\boldsymbol{w}$, is defined as

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{j,c} (y_{j,c}^H(\boldsymbol{w}) - d_{j,c})^2 \tag{2.4}$$

where $y_{j,c}^H(\boldsymbol{w})$ is the state obtained for output node j in layer H in input-output case c and $d_{j,c}$ is its desired state specified by the teacher. One method for minimization of $E(\boldsymbol{w})$ is to apply the method of gradient-descent by starting with any set of weights and repeatedly updating each weight by an amount

$$\Delta w_{ji}^h(t) = -\epsilon \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}^h(t-1) - hdec.\, w_{ji}^h(t-1) \tag{2.5}$$

where the positive constant ϵ controls the descent, $0 \leq \alpha \leq 1$ is the damping coefficient or momentum, *hdec* is the *percent* decay coefficient and t denotes the number of the iteration currently in progress. Using a decay factor $0.01 > hdec > 0$ enables only those weights doing *useful* work in reducing the error to survive and hence improves the generalization capabilities of the network [284](p.8).

From eqns. (2.1)-(2.2) and (2.4), we have

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \frac{dy_j}{dx_j} \frac{\partial x_j}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} y_j^h (1 - y_j^h) y_i^{h-1} \tag{2.6}$$

For the output layer ($h = H$), we substitute in eqn. (2.6)

$$\frac{\partial E}{\partial y_j} = y_j^H - d_j \tag{2.7}$$

This assigns credit proportionately to those weights most responsible for the error, thus implementing gradient steepest descent in the weight space. The central idea is

to first use a forward pass for each input-output case c, starting at the input neurons, to compute the activity levels of all the neurons in the network. Then a backward pass, starting at the output neurons, is used to compute the error derivative $\frac{\partial E}{\partial y_j}$ and backpropagate to enable weight updating until the input layer is reached. For the other layers, using eqn. (2.1), we substitute in eqn. (2.6)

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dx_k} \frac{\partial x_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dx_k} w_{kj}^h \tag{2.8}$$

where units j and k lie in layers h and $h+1$ respectively.

Consider a multi-dimensional weight-space with an axis for each weight and one extra axis corresponding to the error measure. This weight space typically contains *ravines* with steep sides and a shallow gradient along the *ravine*. Acceleration methods used in eqn. (2.5) force the gradient to change the velocity of the current point in the weight space, help to speed convergence and rarely get stuck at poor local minima. The roles of ϵ and α in eqn. (2.5) have natural interpretations in terms of physical movement along this error surface, composed of *hills, valleys, ravines, ridges, plateaus* and *saddle points*, in the weight space.

During training, each pattern of the training set is used in succession to clamp the input and output layers of the network. A sequence of forward and backward passes using eqns. (2.1)-(2.8) constitute a *cycle* and such a cycle through the entire training set is termed a *sweep*. After a number of sweeps through the training data, the error $E(\boldsymbol{w})$ in eqn. (2.4) may be minimized. At this stage the network is supposed to have discovered (learned) the relationship between the input and output vectors in the training samples.

In the testing phase the neural net is expected to be able to utilize the information encoded in its connection weights to assign the correct output labels for the test vectors that are now clamped only at the input layer. It should be noted that the optimal number of hidden layers and the number of units in each such layer are mostly determined empirically. This is demonstrated in Section 2.6. The number of units in layer H correspond to the number of output classes.

2.3 Pattern representation in linguistic form

In conventional statistical designs, the input patterns are quantitatively exact to within the resolution of the sensors used to collect them. However, real processes also may possess imprecise or incomplete input features. In such cases it may become convenient to use linguistic variables and hedges [17] like *low, medium, high, very, more or less,* etc. to augment or even replace numerical input feature information.

The fuzzy MLP model is capable of handling exact (numerical) and/or inexact (linguistic) forms of input data. Any input feature value can be described in terms of some combination of membership values in the linguistic property sets *low, medium* and *high.*

2.3.1 Fuzzy sets

In traditional classifiers an element r either belongs or does not belong to a given class A. Hence the characteristic function of A is expressed as

$$\mu_A(r) = \begin{cases} 1 & \text{if } r \in A \\ 0 & \text{otherwise} \end{cases}$$

But in real-life problems the classes are often ill-defined, overlapping or fuzzy. A pattern may belong to more than one class with a non-zero degree of membership. Using fuzzy set-theoretic techniques [17, 18, 22] a pattern point r, belonging to the universe R, may be assigned a grade of membership with the membership function $\mu_A(r) \in [0, 1]$ to a fuzzy set A (as defined earlier in Section 1.2.1).

2.3.2 π membership function

The $\pi-$function, with range [0,1] and $\boldsymbol{r} \in I\!R^n$, is defined as [285]

$$\pi(\boldsymbol{r}; \boldsymbol{c}, \lambda) = \begin{cases} 2\left(1 - \frac{\|\boldsymbol{r}-\boldsymbol{c}\|}{\lambda}\right)^2, & \text{for } \frac{\lambda}{2} \leq \|\boldsymbol{r}-\boldsymbol{c}\| \leq \lambda \\ 1 - 2\left(\frac{\|\boldsymbol{r}-\boldsymbol{c}\|}{\lambda}\right)^2, & \text{for } 0 \leq \|\boldsymbol{r}-\boldsymbol{c}\| \leq \frac{\lambda}{2} \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

where $\lambda > 0$ is the radius of the $\pi-$function with $\boldsymbol{c}$ as the central point and $\|.\|$ denotes the Euclidean norm. This is shown in Fig. 2.2 for $\boldsymbol{r} \in I\!R^2$. Note that when the pattern

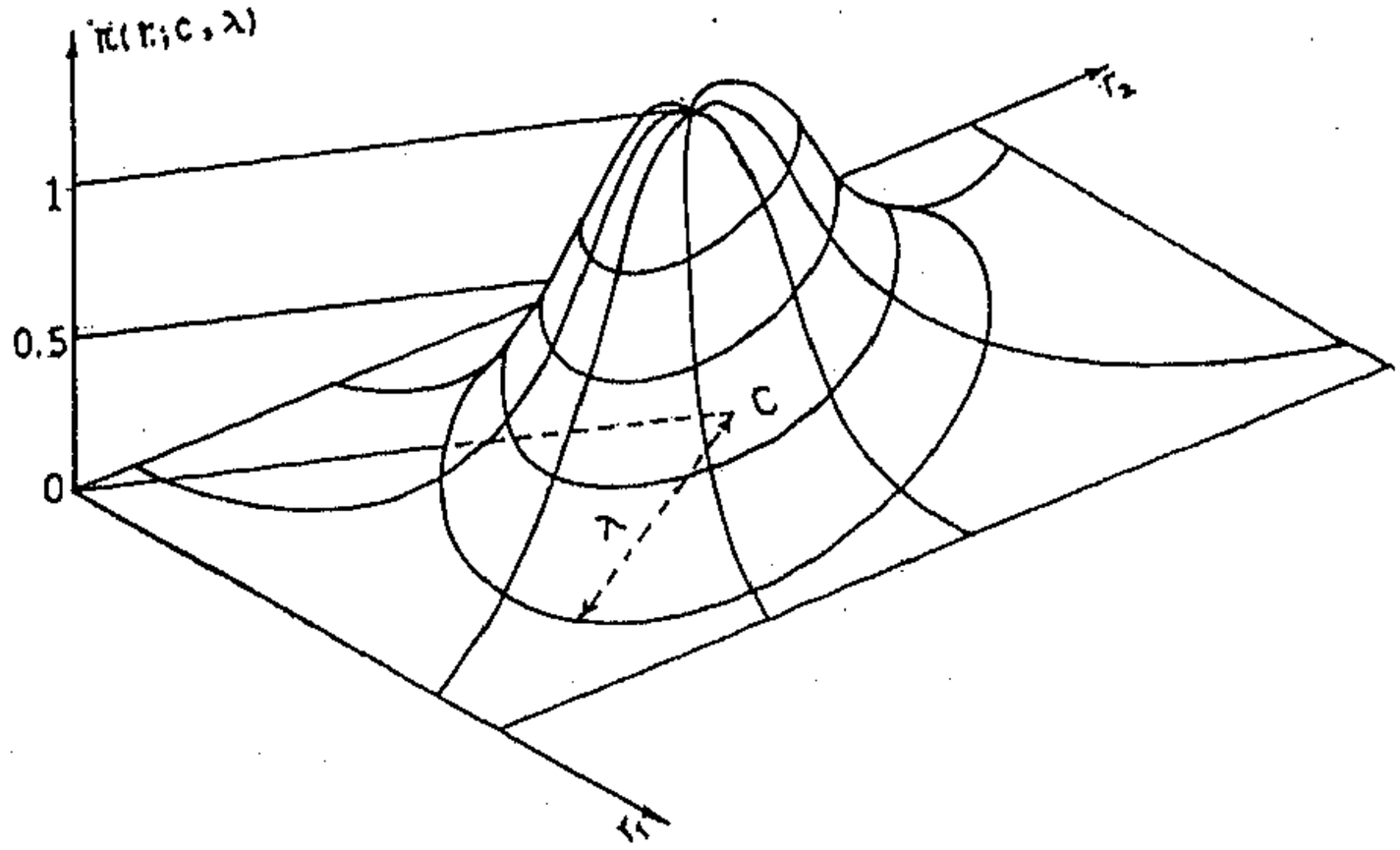

Figure 2.2: $\pi-$ function when $r \in R^2$

$\boldsymbol{r}$ lies at the central point $\boldsymbol{c}$ of a class, then $\|\boldsymbol{r} - \boldsymbol{c}\| = 0$ and its membership value is maximum, *i.e.*, $\pi(\boldsymbol{c}; \boldsymbol{c}, \lambda) = 1$. The membership value of a point decreases as its distance from the central point $\boldsymbol{c}$, *i.e.*, $\|\boldsymbol{r} - \boldsymbol{c}\|$ increases. When $\|\boldsymbol{r} - \boldsymbol{c}\| = \frac{\lambda}{2}$, the membership value of $\boldsymbol{r}$ is 0.5 and this is called a *cross-over* point.

A fuzzy set with membership function $\pi(\boldsymbol{r}; \boldsymbol{c}, \lambda)$ therefore represents a set of points clustered around $\boldsymbol{c}$. In the fuzzy MLP model, the $\pi-$function (in the one-dimensional form) is used to assign membership values for the input features.

2.3.3 Incorporation of the linguistic concept

Each input feature F_j (in quantitative and/or linguistic form) can be expressed in terms of membership values to each of the three linguistic properties *low, medium* and *high*. Therefore an $n-$dimensional pattern $\vec{F}_i = [F_{i1}, F_{i2}, \ldots, F_{in}]$ may be represented as a $3n-$dimensional vector [84]

$$\vec{F}_i = \left[\mu_{low(F_{i1})}(\vec{F}_i), \mu_{medium(F_{i1})}(\vec{F}_i), \mu_{high(F_{i1})}(\vec{F}_i), \ldots, \mu_{high(F_{in})}(\vec{F}_i)\right] \qquad (2.10)$$

When the input feature F_j is linguistic, its membership values for the $\pi-$sets *low, medium* and *high* are quantified as

$$\begin{aligned} low &\equiv \left\{\frac{0.95}{L}, \frac{0.6}{M}, \frac{0.02}{H}\right\} \\ nedium &\equiv \left\{\frac{0.7}{L}, \frac{0.95}{M}, \frac{0.7}{H}\right\} \\ high &\equiv \left\{\frac{0.02}{L}, \frac{0.6}{M}, \frac{0.95}{H}\right\} \end{aligned} \qquad (2.11)$$

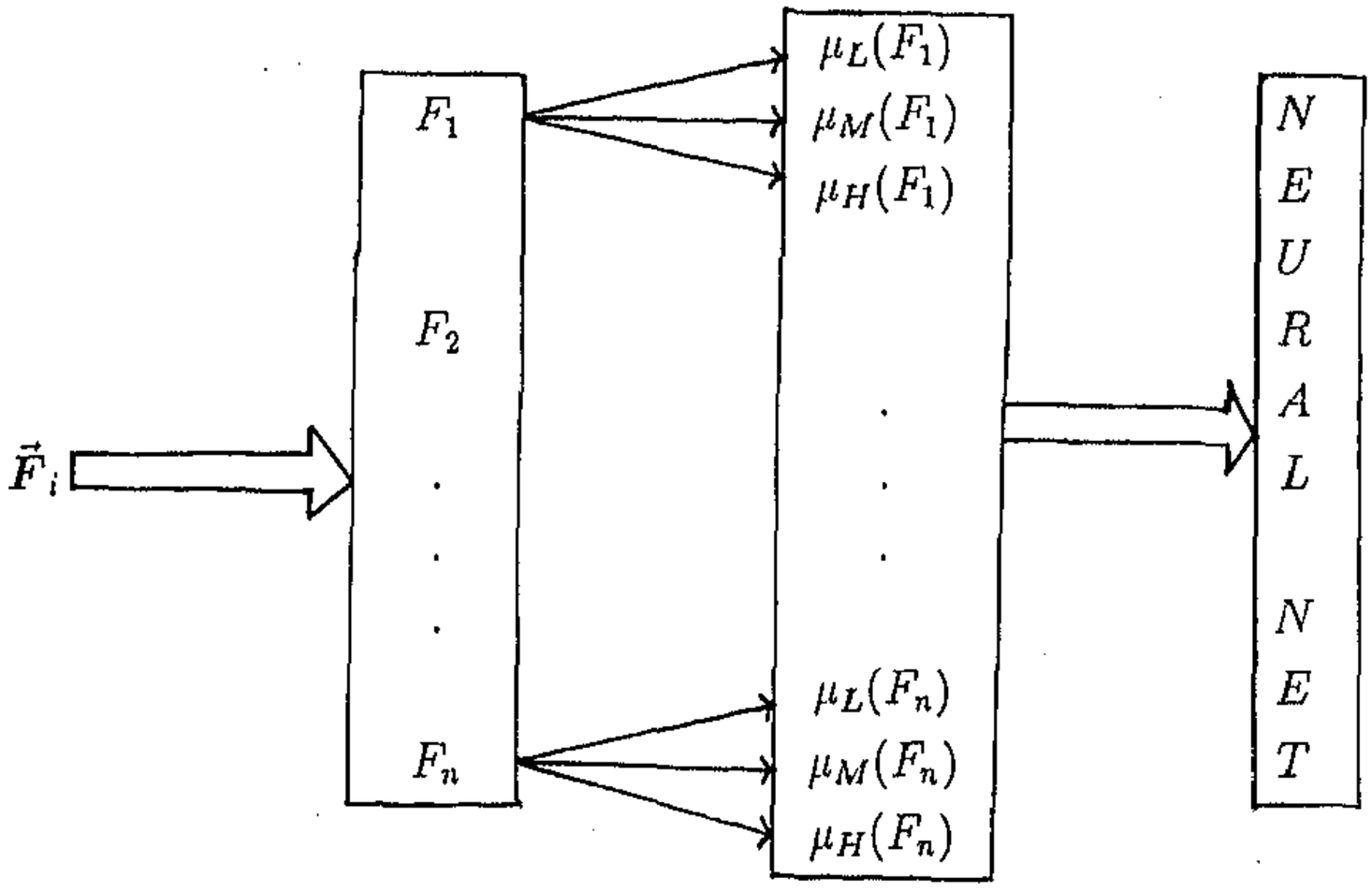

Figure 2.3: Block diagram of the input phase

For example, if F_j is *low*, then its membership values for the primary properties *low (L)*, *medium (M)* and *high (H)* are 0.95, 0.6 and 0.02 respectively. Similar are the cases of F_j is *medium* or *high*.

When F_j is numerical, we use the $\pi-$fuzzy set of eqn. (2.9) with appropriate c and λ chosen as explained in Section 2.3.4.

The processing at the input layer is summarized in the block diagram in Fig. 2.3. Depending upon the numerical or linguistic nature of the input feature F_j, eqn. (2.9) or eqn. (2.11) is used to convert F_j to its 3−dimensional form given by eqn. (2.10). Note that for the training set the numerical F_j's are kept aside to also compute the mean and standard deviation vectors of eqn. (2.15). However in case of the test set, once the membership values have been computed, the actual numerical values are no longer needed in future computations.

Hence in trying to express an input $\vec{F}_i$ with linguistic properties we are effectively dividing the dynamic range of each feature into three overlapping partitions. The

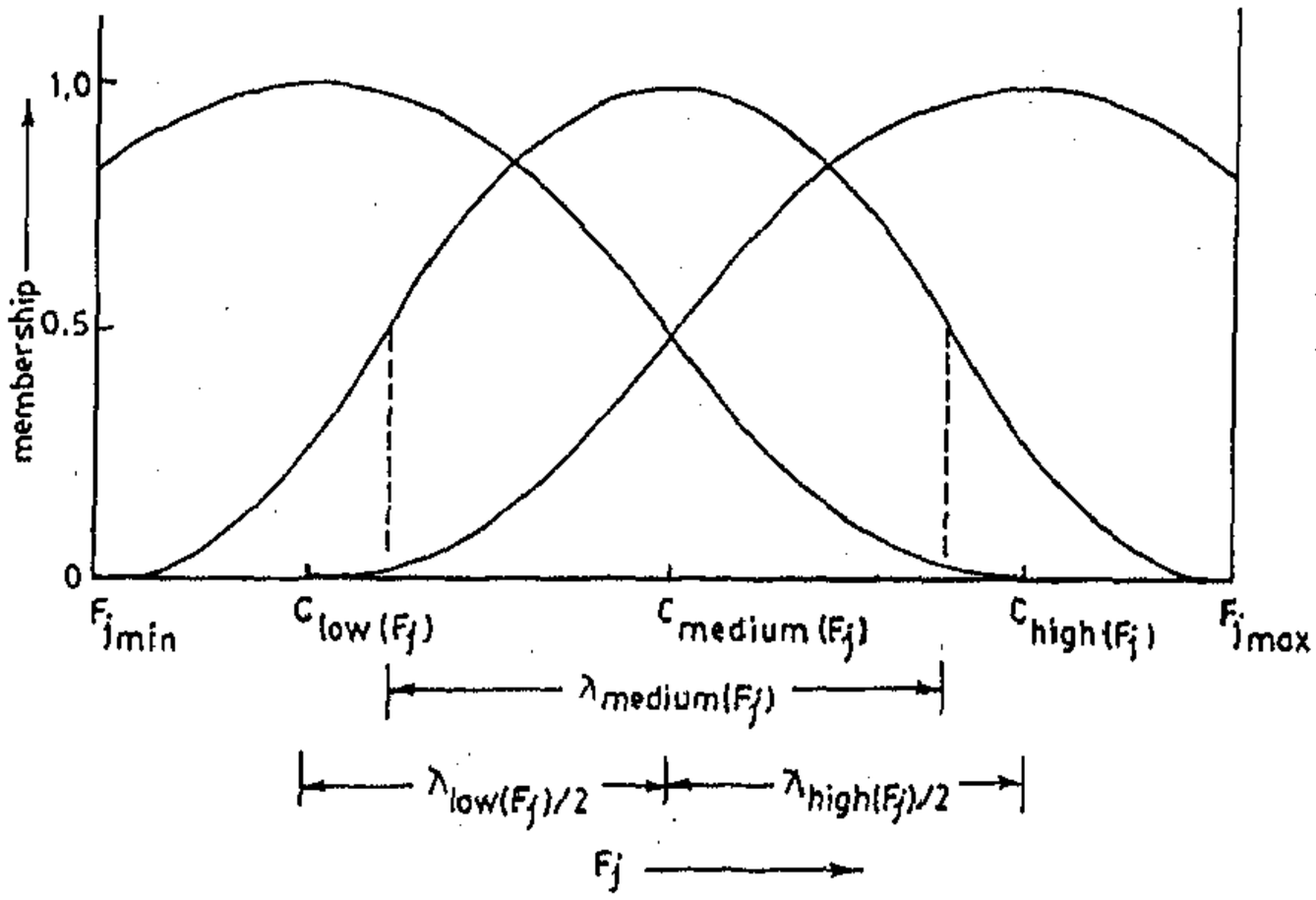

Figure 2.4: Overlapping structure of the $\pi-$ functions

sets *low, medium and high* for each feature are represented by $\pi-$functions. Fig. 2.4 shows the overlapping structure of the three $\pi-$functions for a particular input feature F_j. It should be noted in this context that triangular functions, having properties of eqn. (2.9), can also be used in generating the input features of the network.

2.3.4 Choice of parameters of $\pi-$functions for numerical features

Let $F_{j_{max}}$ and $F_{j_{min}}$ denote the upper and lower bounds of the observed range of feature F_j in all L pattern points, considering numerical values only. Then for the three linguistic property sets we define

$$\begin{aligned} \lambda_{medium(F_j)} &= \frac{1}{2}\left(F_{j_{max}} - F_{j_{min}}\right) \\ c_{medium(F_j)} &= F_{j_{min}} + \lambda_{medium(F_j)} \end{aligned} \tag{2.12}$$

$$\lambda_{low(F_j)} = \frac{1}{fdenom}\left(c_{medium(F_j)} - F_{j_{min}}\right)$$

$$c_{low(F_j)} = c_{medium(F_j)} - 0.5 * \lambda_{low(F_j)} \tag{2.13}$$

$$\lambda_{high(F_j)} = \frac{1}{fdenom}\left(F_{j_{max}} - c_{medium(F_j)}\right)$$

$$c_{high(F_j)} = c_{medium(F_j)} + 0.5 * \lambda_{high(F_j)} \tag{2.14}$$

where $0.5 \leq fdenom \leq 1.0$ is a parameter controlling the extent of overlapping.

This combination of choices for the λ's and c's automatically ensures that each

quantitative input feature value r_j along the j^{th} axis for pattern $\vec{F}_i$ is assigned membership value combinations in the corresponding 3−dimensional linguistic space of eqn. (2.10) in such a way that at least one of $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ or $\mu_{high(F_{ij})}(\vec{F}_i)$ is greater than 0.5. This allows a pattern $\vec{F}_i$ to have strong membership to at least one of the properties *low*, *medium* or *high*.

2.4 Class memberships as output vectors

In the conventional MLP, used for pattern classification, the number of output nodes corresponds to the number of pattern classes present. During training, the output node corresponding to the class of a pattern vector is kept clamped at state 1 while the others are clamped to state 0. Hence the components of the desired output vector take on *crisp* two-state values. During testing, a *winner-take-all* mechanism causes the test pattern to be classified as belonging to that class corresponding to the output node with the highest activation.

In real-life problems, the data are generally ill-defined with overlapping or fuzzy class boundaries. Each pattern used in training may possess non-zero belongingness to more than one class. To model such data we clamp the desired membership values, lying in the range [0,1], at the output nodes during training. Then the network backpropagates the error(s) with respect to these desired membership value(s) at the output(s). In the process, the network may become able to detect regularities in the input-output membership relation of the training set. Then, when a separate set of test patterns is presented at the input layer, the output nodes automatically generate the class membership values of the patterns to the corresponding classes. This procedure of assigning *fuzzy* output membership values, instead of the more conventional binary output values, enables the fuzzy layered neural net model to more efficiently classify the fuzzy data having overlapping class boundaries.

2.4.1 Membership functions

Consider an $l-$class problem domain such that we have l nodes in the output layer. Let the $n-$dimensional vectors O_k and V_k denote the mean and standard deviation respectively of the numerical training data for the k^{th} class. The weighted distance of

the training pattern $\vec{F}_i$ from the k^{th} class is defined as

$$z_{ik} = \sqrt{\sum_{j=1}^{n} \left[\frac{F_{ij} - o_{kj}}{v_{kj}}\right]^2} \text{ for } k = 1, \ldots, l \tag{2.15}$$

where F_{ij} is the value of the j^{th} component of the i^{th} pattern point and C_k is the k^{th} class. The weight $\frac{1}{v_{kj}}$ is used to take care of the variance of the classes so that a feature with higher variance has less weight (significance) in characterising a class. Note that when all the feature values of a class are the same, then the standard deviation will be zero. In that case, we consider $v_{kj} = 1$ such that the weighting coefficient becomes one. This is obvious because any feature occurring with identical magnitudes in all members of a training set is certainly an *important* feature of the set and hence its contribution to the membership function should not be reduced [17, 56].

The membership [17] of the i^{th} pattern to class C_k is defined as follows

$$\mu_k(\vec{F}_i) = \frac{1}{1 + \left(\frac{z_{ik}}{f_d}\right)^{f_e}} \tag{2.16}$$

where z_{ik} is the weighted distance from eqn. (2.15) and the positive constants f_d and f_e are the denominational and exponential fuzzy generators controlling the amount of fuzziness in this class-membership set (*i.e.*, in the distance set). Obviously $\mu_k(\vec{F}_i)$ lies in the interval [0,1]. Here eqn. (2.16) is such that the higher the distance of a pattern from a class, the lower is its membership value to that class. It is to be noted that when the distance is zero, the membership value is one (maximum) and when the distance is infinite, the membership value is zero (minimum).

As the training data have fuzzy class separation, a pattern point $\vec{F}_i$ may correpond to one or more classes in the input feature space. So a pattern point belonging to two classes (say, C_{k_1} and C_{k_2}) corresponds to two *hard* labels in the training data, with $\vec{F}_i$ tagged to classes C_{k_1} and C_{k_2} respectively. In other words, there are two or more occurrences of point $\vec{F}_i$ in the training set such that sometimes it is tagged to class C_{k_1} and sometimes to class C_{k_2}. In this case $\vec{F}_i$ is used in computing O_{k_1}, O_{k_2}, V_{k_1} and V_{k_2} only. Here the $l-$dimensional vector Z_i has only two non-zero components, *viz.*, z_{ik_1} and z_{ik_2}. However in the *hard* case $\vec{F}_i$ corresponds to only one *hard* label in the training data, say C_{k_1}, such that $\vec{F}_i$ is used in computing O_{k_1} and V_{k_1} only. Note that Z_i has l non-zero components in the *fuzziest* case and only one non-zero component in the hard case.

In the *fuzziest* case, we may use the fuzzy modifier INT to enhance contrast [17] in class membership. We have

$$\mu_{INT(k)}(\vec{F}_i) = \begin{cases} 2\left[\mu_k(\vec{F}_i)\right]^2 & \text{for } 0 \leq \mu_k(\vec{F}_i) \leq 0.5 \\ 1 - 2\left[1 - \mu_k(\vec{F}_i)\right]^2 & \text{otherwise} \end{cases} \tag{2.17}$$

This is needed to increase the contrast within class membership values, *i.e.*, to decrease the ambiguity in taking a decision.

When the training input is linguistic, we use

$$z_{ik} = \sqrt{\sum_{j=1}^{n} \left[\sum_{p=1}^{3} \frac{1}{3} \left(\mu_p(F_{ij}) - \mu_p(o_{kj})\right)^2 \right]} \text{ for } k = 1, \ldots, l \tag{2.18}$$

where $\mu_1(F_{ij})$, $\mu_2(F_{ij})$, $\mu_3(F_{ij})$ correspond to the membership values given by eqn. (2.11) and $\mu_p(o_{kj})$ correspond to the $3n-$dimensional representation by eqns. (2.10) and (2.12)-(2.14) of the class mean O_k computed using the numerical input values.

2.4.2 Applying the membership concept

For the i^{th} input pattern we define the desired output of the j^{th} output node as

$$d_j = \begin{cases} \mu_{INT(j)}(\vec{F}_i) & \text{in the } \textit{fuzziest} \text{ case} \\ \mu_j(\vec{F}_i) & \text{otherwise} \end{cases} \tag{2.19}$$

where $0 \leq d_j \leq 1$ for all j. The l output neurons are clamped with the fuzzy output membership values of eqn. (2.19) during training and the error backpropagated for appropriate weight updating by eqns. (2.5)-(2.8).

The block diagram given in Fig. 2.5 illustrates the various stages in computing the desired output vector to be clamped at the output nodes of the proposed model. Note that the enhancement phase (3^{rd} stage in the figure) is required in the *fuzziest* case only.

During testing the output of the j^{th} output neuron y_j^H of eqn. (2.2), computed in a forward pass after clamping the $3n-$dimensional input vector from eqn. (2.10) for the test pattern $\vec{F}_t$ at the input layer, indicates the inferred membership value of the pattern $\vec{F}_t$ to the j^{th} class. So the output is generated as a measure of finite belongingness to each class. This is unlike the conventional *crisp* concept of either belonging or not belonging to a class.

$$\vec{F}_i \xrightarrow{\text{weighted distance}} \begin{pmatrix} z_{i1} \\ \vdots \\ z_{il} \end{pmatrix} \xrightarrow{\text{membership}} \begin{pmatrix} \mu_1(\vec{F}_i) \\ \vdots \\ \mu_l(\vec{F}_i) \end{pmatrix} \xrightarrow{\text{enhancement}} \begin{pmatrix} \mu_{INT(1)}(\vec{F}_i) \\ \vdots \\ \mu_{INT(l)}(\vec{F}_i) \end{pmatrix} \xrightarrow{\text{desired vector}} \vec{d}$$

Figure 2.5: Block diagram of the output phase

2.5 Fuzzy extension to the MLP model

Consider an $(H+1)$–layered MLP with $3n$ neurons in the input layer and l neurons in the output layer, such that there are $H-1$ hidden layers. The input vector, with components x_j^0 of eqn. (2.3) represented as in eqn. (2.10), is clamped at the input layer while the desired l–dimensional output vector with components d_j from eqn. (2.19) is clamped during training at the output layer. The outputs of the neurons y_j^H in the various layers, $h = 0, 1, \ldots, H$, of the network are computed using eqns. (2.1)-(2.3) to obtain the error E at the output layer by eqn. (2.4). It is to be noted that the y_j^H's in eqn. (2.2) and d_j's in eqn. (2.4) are expressed as continuous fuzzy membership values lying in the interval [0,1]. This is an extension to the conventional MLP model that deals with actual input values and *crisp* (binary) output values. This model is found to classify fuzzy data with overlapping class boundaries in a more efficient manner.

2.5.1 Weight Updating

Initially the connection weights w_{ji}^h between each pair of neurons i in layer h and j in layer $h+1$ are set to small random values lying in the range [-0.5,0.5]. The error between the desired output d_j and computed output y_j^H in the output layer (for all output neurons) is evaluated using eqn. (2.4) and backpropagated for appropriate weight updating by eqns. (2.5)-(2.8) such that the weights are modified in proportion to their contribution to the error. The updating occurs after each cycle and is not accumulated over the entire sweep in order to offset the problem caused by the inherent higher redundancy of *natural* data or, in other words, to counter the repetitive occurrence of very similar training samples [286](p.7). It should be mentioned that we adjust $\Delta w_{ji}^h(t)$ to ensure that $-0.1 \leq \Delta w_{ji}^h(t) \leq 0.1$ for each updating step, so that the possibility of *overshooting* of the weights may be minimized in the course of

smoothly approaching a minimum error solution.

For a pattern point lying in a region of overlapping in the feature space, there may be more than one output node with (say) $d_j > 0.5$ indicating appreciable belongingness to more than one pattern class. In the conventional MLP for such a case only that node with a slightly higher activation is clamped to state 1 while the others are clamped to state 0. This may result in oscillations on the decision surface separating the pattern classes while training the network, because nearly identical patterns may be clamped to different classes. In such a case, terminating the algorithm at an arbitrary point may or may not yield a *"good"* weight vector.

In other words, the conventional *crisp* algorithm may not terminate, nor necessarily converge to a *useful* solution when the data are nonseparable with overlapping fuzzy classes. The pattern vectors that cause the classes to overlap are probably responsible for the oscillatory behaviour of the algorithm because these *"ambiguous"* vectors, although relatively less characteristic of their classes, are given full weightage in one class while backpropagating the corresponding errors for weight updating.

The fuzzy MLP model overcomes this problem by assigning the class membership values to the corresponding output nodes. Thus, the error to be backpropagated has more weightage in case of nodes with higher membership values and hence can induce greater weight corrections in favour of that class for input data that demand such an adjustment. This is desirable, as points that have a larger belongingness to a particular pattern class and possess less ambiguity should influence positively the positioning of the decision surface separating the pattern classes. The contribution of ambiguous or uncertain vectors (*i.e.*, those with output membership close to 0.5) to the weight correction in favour of any class is automatically reduced. As the actual output vectors are modified to approximate the corresponding *ambiguous* desired output vectors, it helps to prevent oscillation on the decision surface in such cases and thus enables the model to function efficiently in environments with fuzzy class separation. Note the possible reverse but significant impact of pattern vectors in inducing weight updating along interconnections corresponding to classes for which they have lower output membership values (closer to 0). This is because such low values signify the degree of *not belonging* to a class and hence involve less ambiguity.

2.5.2 Learning rate, mean square error and cross entropy

In conventional MLP the two-state output is either 0.8 (true) or 0.2 (false) [284, 287] and the learning rate ϵ and momentum α in eqn. (2.5) of the backpropagation algorithm are constant throughout training. In this model we introduce output membership values distributed over the range [0,1], depending on the degree of belongingness of a pattern vector to a class. These output values are actually distributed over a suitable subinterval, say, [0.2,0.8], as attainment of 0 or 1 by y_j^H in eqn. (2.2) requires an infinite input. We usually encounter a large number of intermediate output values during the training phase. To distinguish finer intricacies in the desired outputs, ϵ is gradually decreased in discrete steps, taking values from the chosen set {2, 1, 0.5, 0.3, 0.1, 0.05, 0.01, 0.005, 0.001}. Further, α is also decreased from an initial value of 0.9 for $\epsilon = 2$ to a final value of 0.5 for all other values of ϵ.

In the course of training, at each sweep we use the mean square error (mse) and the cross entropy (s) as performance measures of learning. We define,

$$mse = \left[\sum_{\vec{F} \in trainset} \sum_{j=1}^{l} \left(d_j - y_j^H \right)^2 \right] / \left(l * |trainset| \right) \tag{2.20}$$

and

$$s = \left[\sum_{\vec{F} \in trainset} \sum_{j=1}^{l} \left\{ -d_j \ln y_j^H - (1 - d_j) \ln(1 - y_j^H) \right\} \right] / \left(\ln 2 * l * |trainset| \right) \tag{2.21}$$

Here $|trainset|$ refers to the number of input pattern vectors in the training set, and d_j corresponds to the j^{th} desired output component. It is to be noted that both mse and s decrease as y_j^H approaches d_j for all j.

In our experiments, both mse and s are found to steadily decrease and reach a local minimum, for each value of ϵ, after several sweeps (say, 20 to 300). This is the appropriate time to decrease ϵ, as otherwise generally oscillation sets in. The necessity of modeling a number of intermediate output values in the range [0,1] seems to cause the weight space to have ravines with a corresponding error surface of great complexity, so that there exist a large number of local minima. One typically starts a long way from the minimum error solution, and spends a lot of time oscillating across ravines in the weight space before numerical convergence is attained. A larger ϵ is initially

necessary to cause weight updating along the gradient to occur fast in roughly the right direction without getting stuck at poor local minima. The momentum term with α helps maintain movement along a stable direction, and hence should be high in the early stages to prevent the weights from accidentally attaining large values due to the initially high gradient that may often be in incorrect directions. As the weight vectors move closer towards the correct orientation both the *mse* and s may decrease. After several sweeps, (varying from 20 to 300, say), when the weight vectors have settled somewhat, ϵ appears to be too large for the Δw_{ji}^h's to change by the small amount now required to attain the necessary minimum along the more shallow gradient encountered here. The model may oscillate between local minima, as the weights tend to *overshoot* the minimum error solution; at this point ϵ needs to be decreased. At this time it is preferrable to decrease α to speed progress along the most effective direction. This allows more weight to the gradient (now with lower ϵ) to model finer intermediate output values while also maintaining some damping (via α) to prevent oscillation. The same procedure is repeated for the discrete set of ϵ's until it stops at the lowest chosen value of ϵ. This scheme of decreasing ϵ and α is somewhat different from the approach used in conventional MLP, but is somewhat analogous to the method reported in [159]. A comparison in performance of the proposed model with the latter (termed Variation *O*) is given in Table 2.3. Keeping ϵ and α constant as in the *crisp* model, or increasing it gradually [288], seems to lead to oscillation in our model, because the fuzzy version of MLP apparently is incapable of modeling intermediate output values encountered in the fuzzy data space so that the weights tend to overshoot.

We use two measures of *percent* correct classification for the training set. The output, after a number of updatings, is considered a *perfect match* if the value of each output neuron y_j^H is within a margin of 0.1 of the desired membership value d_j. This is a *stricter* criterion than the *best match*, where we test whether the j^{th} neuron output y_j^H (for a particular training pattern) has the highest (or maximum) activation when the j^{th} component d_j of the desired output vector also has the highest value, provided $y_j^H > 0.5$. Each pattern in the training set is supposed to have $y_j^H > 0.5$ for some j, *i.e.*, possess finite positive belongingness to at least one class. As training proceeds, the percent perfect match p and percent best match b gradually increase as the neural net moves towards a minimum error solution. Note that for b we consider only that class to which the pattern belongs *best*.

Let the various values of ϵ be indicated by $\epsilon_0 = 2$, $\epsilon_1 = 1$, ..., $\epsilon_q = 0.001$ such that ϵ_i indicates the $(i+1)^{th}$ value of ϵ. Let $\alpha_0 = 0.9$ and $\alpha_1 = \alpha_2 = \ldots = \alpha_q = 0.5$. Note that α close to zero is avoided because small values of α are unable to prevent unwanted oscillations. We use

$$i = \begin{cases} i+1 & \text{if } mse(nt-kn) - mse(nt) < \delta \\ & \text{Or } s(nt-kn) - s(nt) < \delta \\ i & \text{otherwise} \end{cases} \quad (2.22)$$

where $i = 0$ initially, $|\epsilon| = q+1$ and $0 < \delta \leq 10^{-2}$. Here $mse(nt)$ and $s(nt)$ denote the mean square error by eqn. (2.20) and cross entropy by eqn. (2.21) respectively at the end of the nt^{th} sweep through the training set and kn is a positive integer such that mse and s are sampled at intervals of kn sweeps. The process is terminated when $i > q$ and $\epsilon_q = 0.001$. At this stage the network is said to have converged to a *good* minimum error solution if $90 \leq b \leq 100$. Obviously $p \leq b$ always. The corresponding value of nt indicates the maximal number of sweeps required in the process. It is to be mentioned that the relations $\Delta mse_j < \Delta mse_i$ and $\Delta s_j < \Delta s_i$ hold when $\epsilon_j < \epsilon_i$.

2.5.3 Testing phase

After training, the fuzzy MLP has presumably encoded in its weights information regarding class discrimination in the feature space. This is the desired fuzzy classifier. In the second stage, a separate set of test patterns (initially kept aside from the same fuzzy data) is supplied as input to the neural network model and its performance is evaluated.

During this phase a labeled input test vector $\vec{F}$ in the $3n-$dimensional linguistic space of eqn. (2.10) is applied to the network. The output neurons yield values by eqns. (2.1)-(2.3) indicating fuzzy memberships of the test pattern in the corresponding output classes. The normalised membership values in the range [0.2,0.8] are reconverted to the actual range [0,1].

Let the j_1^{th} and j_2^{th} neurons in the output layer H generate the highest and second highest outputs $y_{j_1}^H$ and $y_{j_2}^H$ respectively, for test pattern $\vec{F}_t$. These represent, respectively, membership values to class j_1 using the best choice and to class j_2 using the second choice. It may be that $y_{j_1}^H \simeq y_{j_2}^H$ for a pattern $\vec{F}_t$ lying in overlapping regions in the input feature space, so that there is some ambiguity of decision in such cases.

A confusion matrix is generated for the test set to evaluate the network performance. To determine the row of an entry in the said confusion matrix, we use the class j of the particular test data as obtained from the maximum of the desired outputs d_j's by eqns. (2.15)-(2.19). Note that the non-zero components of Z_t are determined from the *hard* labels attached to the test pattern $\vec{F}_t$ as well as the training patterns having the same feature values. The column corresponds to the neuronal output class providing the *best match* using eqns. (2.1)-(2.3) and (2.10). The generalization capability of the model in correctly classifying unseen patterns is then verified.

The mean square error for the test patterns, mse_t, is defined from eqn. (2.20) as

$$mse_t = \left[\sum_{\vec{F} \in testset} \sum_j \left(d_j - y_j^H \right)^2 \right] / \left(l * |testset| \right) \tag{2.23}$$

where $|testset|$ corresponds to the number of pattern vectors used during testing and the l components of y_j^H in the output layer are computed from the $3n$ components x_j^0 of pattern $\vec{F}$ using eqns. (2.1)-(2.3). The l components of the desired output d_j are generated by using eqns. (2.18)-(2.19). This mse_t is a measure of the amount of misclassification for the set of test patterns. Analogously, the cross entropy for the test patterns, s_t, is defined from eqn. (2.21) as

$$s_t = \left[\sum_{\vec{F} \in testset} \sum_j \left\{ -d_j \ln y_j^H - (1 - d_j) \ln(1 - y_j^H) \right\} \right] / \left(\ln 2 * l * |testset| \right) \tag{2.24}$$

The closer y_j^H is able to approach d_j, the lower is the value of s_t. Hence this is also a measure of the amount of misclassification for the test set.

2.6 Implementation and results

The above-mentioned algorithm has been tested on a set of 871 Indian Telugu vowel sounds collected by trained personnel [65]. These were uttered by three male speakers in the age group of 30 to 35 years, in a Consonant-Vowel-Consonant context. The data set has three features; F_1, F_2 and F_3 corresponding to the first, second and third vowel formant frequencies obtained through spectrum analysis of the speech data. Thus the dimension of the input vector in eqn. (2.10) for the proposed model is 9. Note that the boundaries of the classes in the given data set are seen to be ill-defined (fuzzy). Fig. 2.6

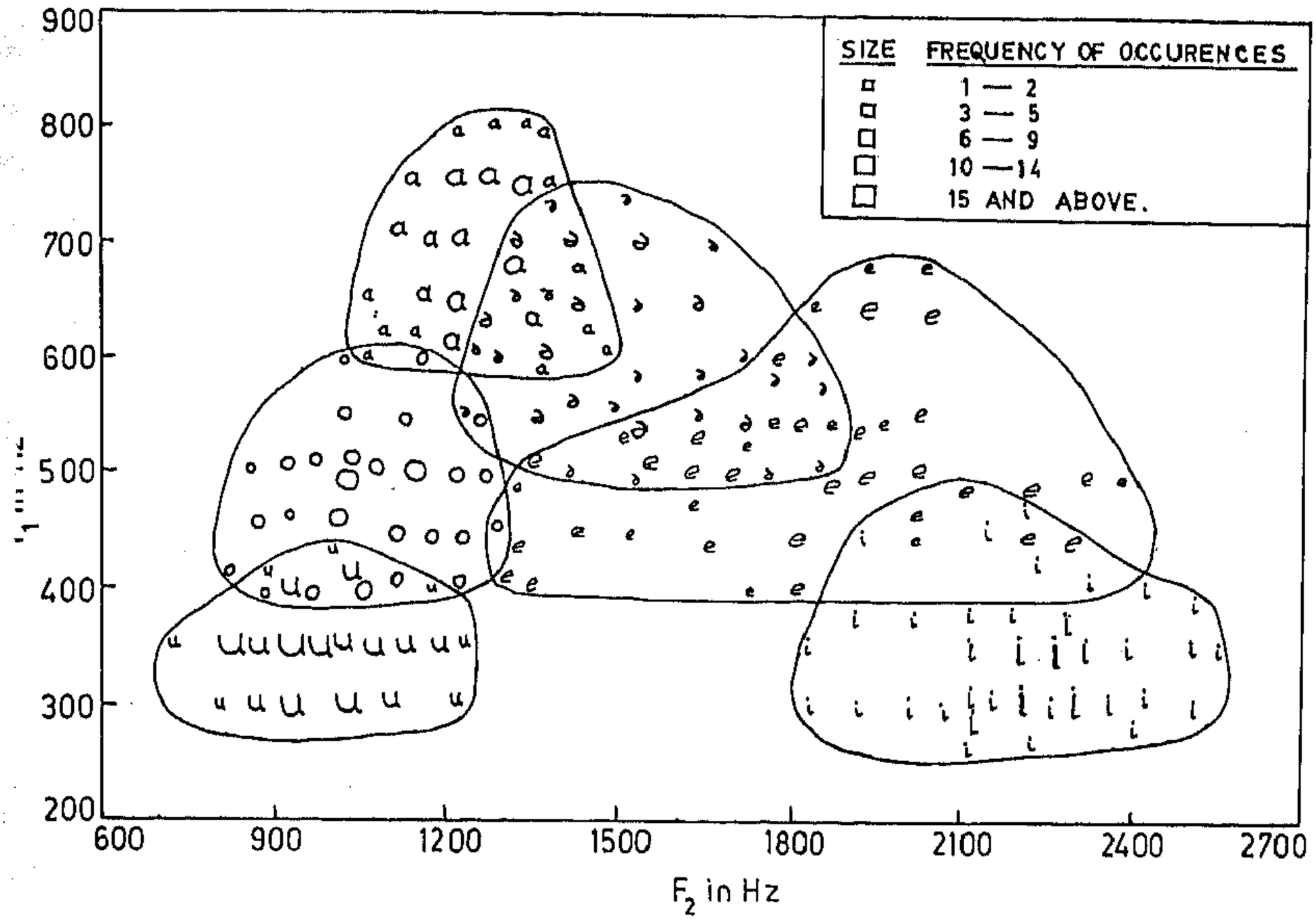

Figure 2.6: Vowel diagram in the $F_1 - F_2$ plane

shows a 2D projection of the 3D feature space of the six vowel classes $(\partial, a, i, u, e, o)$ in the $F_1 - F_2$ plane (for ease of depiction).

It may be mentioned that the conventional *crisp* MLP has also been used in the vowel classification problem [154] to generate hard partitions in the formant frequency space of English vowel data. Besides, a radically different approach has been used in [144], where context-dependent information is used to generate *hard* decision boundaries in an attempt to separate a different set of overlapping English vowel data in the formant frequency space.

On the other hand, this model, incorporating fuzzy concepts at various levels, has been used here to demonstrate its effectiveness in classifying the fuzzy Telugu vowel data. In this case overlapping fuzzy decision regions are generated in the output space.

The model has been tested for different numbers of hidden layers $(H-1)$ (like 1, 2, 3), and with different number of neurons m (like 10, 15, 20), in each such layer. During learning, various sizes of training sets have been used by randomly choosing

different percentages *perc* of samples (like 10%, 50%), from each vowel class. In each case the remaining percentage ($100 - perc$) of data has been used as the test set. The parameters $fdenom$, f_d, f_e, kn and δ in eqns. (2.13)-(2.14), (2.16) and (2.22) have been chosen as $fdenom = 0.8$, $f_d = 5$, $f_e = 1$, $kn = 10$ and $\delta = 0.0001$ after several experiments. Further, we have observed that with $hdec > 0$ in eqn. (2.5), $\epsilon_q = 0.1$ at eqn. (2.22) gives best results whereas with $hdec = 0$ we require $\epsilon_q = 0.001$.

In Figs. 2.7-2.8, part(a) plots the percent correct classification using the best match criterion, while part (b) shows the variation of mse of eqn. (2.20), s of eqn. (2.21) and mse_t of eqn. (2.23). In part(a), the class numbers j (1 for ∂, 2 for a, 3 for i, 4 for u, 5 for e, 6 for o) indicate the class-wise correct classification of the test set. The variables b and p correspond to the *best* match and *perfect* match criteria respectively obtained for the training set while t indicates the percent *best* match performance over the entire test set as a whole. In part(b), the variation of the cross entropy s is shown by the dotted curve while the mean square error for the training (mse) and test (mse_t) set are plotted using solid curves. Note that the left vertical axis corresponds to the mean square error while the right vertical axis indicates the cross entropy. In both these cases, 10% of the samples have been used for training the model.

Fig. 2.7 is drawn to illustrate the effect of varying the number of neurons in the hidden layers of the model. We considered three hidden layers with a distribution $m : m : m$ of neurons such that $m = 10$, 15 and 20. It has been observed that $m = 10$ gives the best results. A larger size is seen to provide better performance over the training set (higher b, p in part(a) and lower mean square error mse and cross entropy s in part(b)), perhaps due to over-learning. This is probably due to *"memorization"*, with poor generalization, that leads to poorer performance over the test set (lower t in part(a) and higher mean square error mse_t in part(b)). A smaller size of (m) is incapable of handling all the information required, perhaps due to insufficient capacity.

In Fig. 2.8 we observe the effect of introducing different numbers of hidden layers (H−1) with $m = 10$ neurons in each layer. A single hidden layer is obviously incapable of capturing the intricacies of the feature space. Two hidden layers lead to a much better performance, as expected. However three hidden layers seem to give an overall best performance both over the training and test sets. The network fails to converge with four hidden layers perhaps due to the resulting very large number of complicated

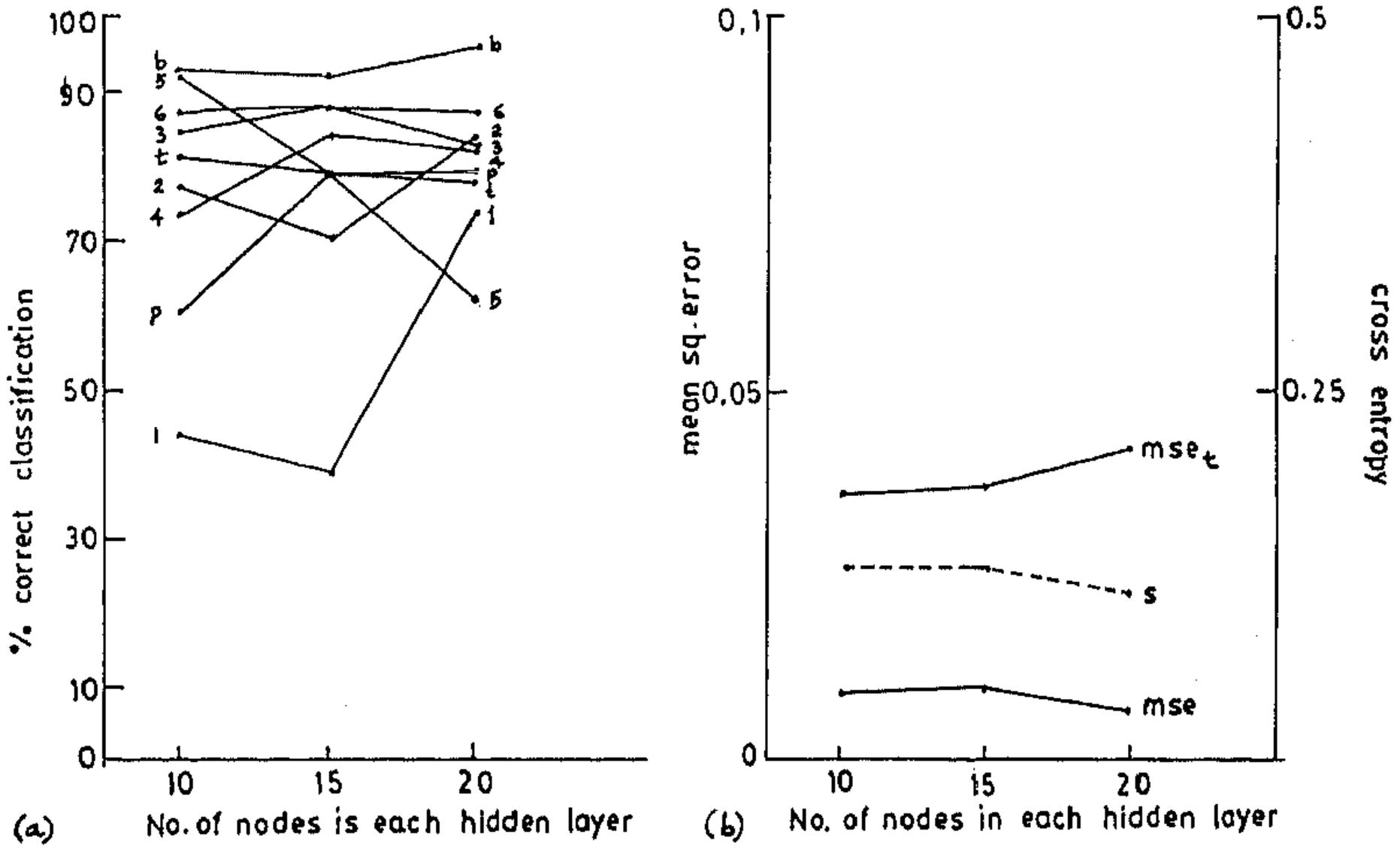

Figure 2.7: Five-layered fuzzy MLP with $hdec = 0$. (a)Correct classification (%), (b)Mean square error and cross entropy

interconnections and the comparatively limited set of data available for training the net.

Fig. 2.9 demonstrates the effects of ϵ and α of eqn. (2.5) on the performance of a five-layered network (with $m = 10$ nodes in each hidden layer) trained using $perc = 10\%$ samples. A fixed low learning rate of $\epsilon = 0.1$ and high momentum of $\alpha = 0.9$, as in the conventional MLP, results in oscillation and the worst classification rates (worst recognition for class 1, *i.e.*, ∂, and nil *perfect* match). Varying ϵ from higher to lower values in discrete steps leads to notably better overall performance. Keeping α constantly low at 0.5 for this ϵ does not provide enough damping and results in a comparatively poor performance as oscillations still cannot be avoided. Clamping α high at 0.9 with more damping, while ϵ varies as proposed, improves the situation. However changing α from 0.5 to 0.9 shows some improvement in the recognition rate for class 1 (∂) at the expense of the neighboring overlapping class 5 (e). The best overall performance is obtained by decreasing α from 0.9 to 0.5, while simultaneously also decreasing ϵ in steps. This has already been explained earlier in Section 2.5. Increasing ϵ from lower to higher values leads to instability such that the network becomes incapable of classifying the fuzzy patterns. This is in sharp contrast to some

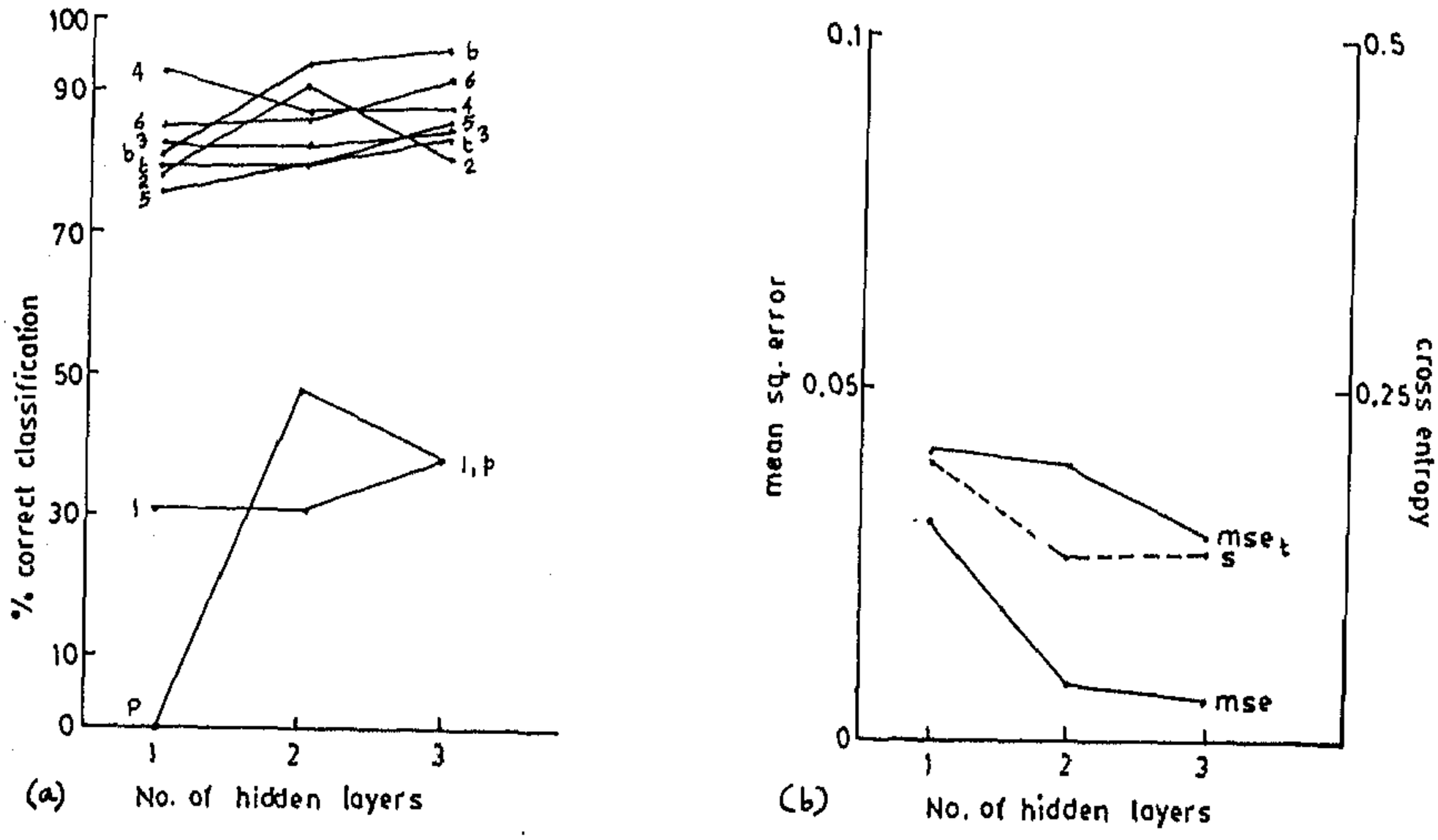

Figure 2.8: Fuzzy MLP with $hdec = 0.001$. (a)Correct classification (%), (b)Mean square error and cross entropy

models [288].

Fig. 2.10 is drawn to illustrate the (crisp) output partitioning capability of the fuzzy MLP with three hidden layers using *alpha-cuts*. The network consists of $m = 10$ nodes in each hidden layer and is trained using $perc = 50\%$ of the samples with $hdec = 0$. An alpha-cut of a fuzzy set A is defined as $A_{\alpha'} = \{r | \mu_A(r) \geq \alpha'\}$, $1 \geq \alpha' > 0$. A pattern with output membership value $y_j^H > \alpha'(= 0.5)$ is plotted as a member of class j. The $l = 6$ vowel classes are labeled by their corresponding class numbers. Fig. 2.10.(a) depicts the resultant output map (class boundaries) over the entire pattern set (both training and testing data) in the two-dimensional formant frequency space showing the fuzzy overlapping, as expected. In Fig. 2.10.(b), the training samples for each class are superimposed on the generated partitions to demonstrate their generalization capability and impact on the classification performance of the model. Note that the topological ordering of the vowel classes with respect to each other and the amount of overlapping between them bear much similarity to the actual partitioning illustrated in Fig. 2.6. This shows that the fuzzy MLP model helps to satisfactorily preserve the structure of ambiguous (fuzzy) classes. (It is to be noted from Fig. 2.10.(b) that sometimes distortion and overshadowing of the symbols occur because of the overlapping

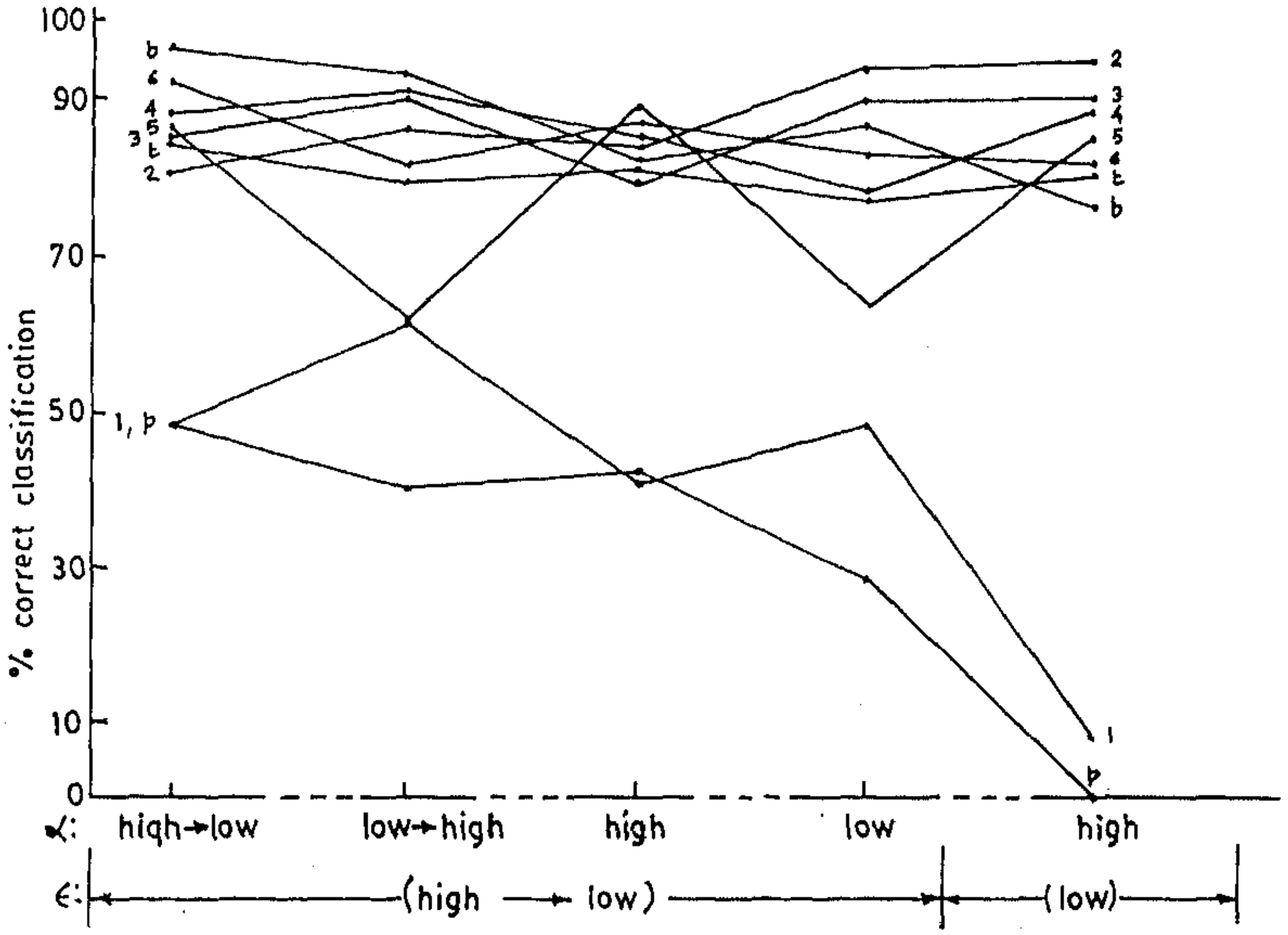

Figure 2.9: Effect of various combinations of ϵ and α on the correct classification (%) of a five-layered net

of the training points from the different pattern classes. This is especially true in case of the co-occurrence of points from classes 1 and 6 in the $F_1 - F_2$ plane, due to the nature of their symbols, as is evident from the figure).

Fig. 2.11 depicts the output partitioning obtained over the entire data set with two hidden layers. The network consists of $m = 10$ nodes in each hidden layer and is trained using $perc = 10\%$ of the samples with $hdec = 0.001$. Here also the training samples are shown superimposed on the generated partitions. This network is seen to be incapable (as compared to the one with three hidden layers) of properly classifying all pattern points, classes 1 (∂) and 2 (a) suffering the most. This may also be verified from Fig. 2.8.

In Fig. 2.12 we illustrate the variation of the best match b and perfect match p performance (denoted by solid curves) and the mean square error mse (plotted using a dotted curve) with the number of sweeps over the training set. A five-layered network with $perc = 50$, $hdec = 0$ and $m = 10$ nodes in each hidden layer is used. The solid points along the curves mark the sweep number at which ϵ_i is changed to ϵ_{i+1} by eqn. (2.22). It is observed that initially high ϵ and α are necessary to cause fast

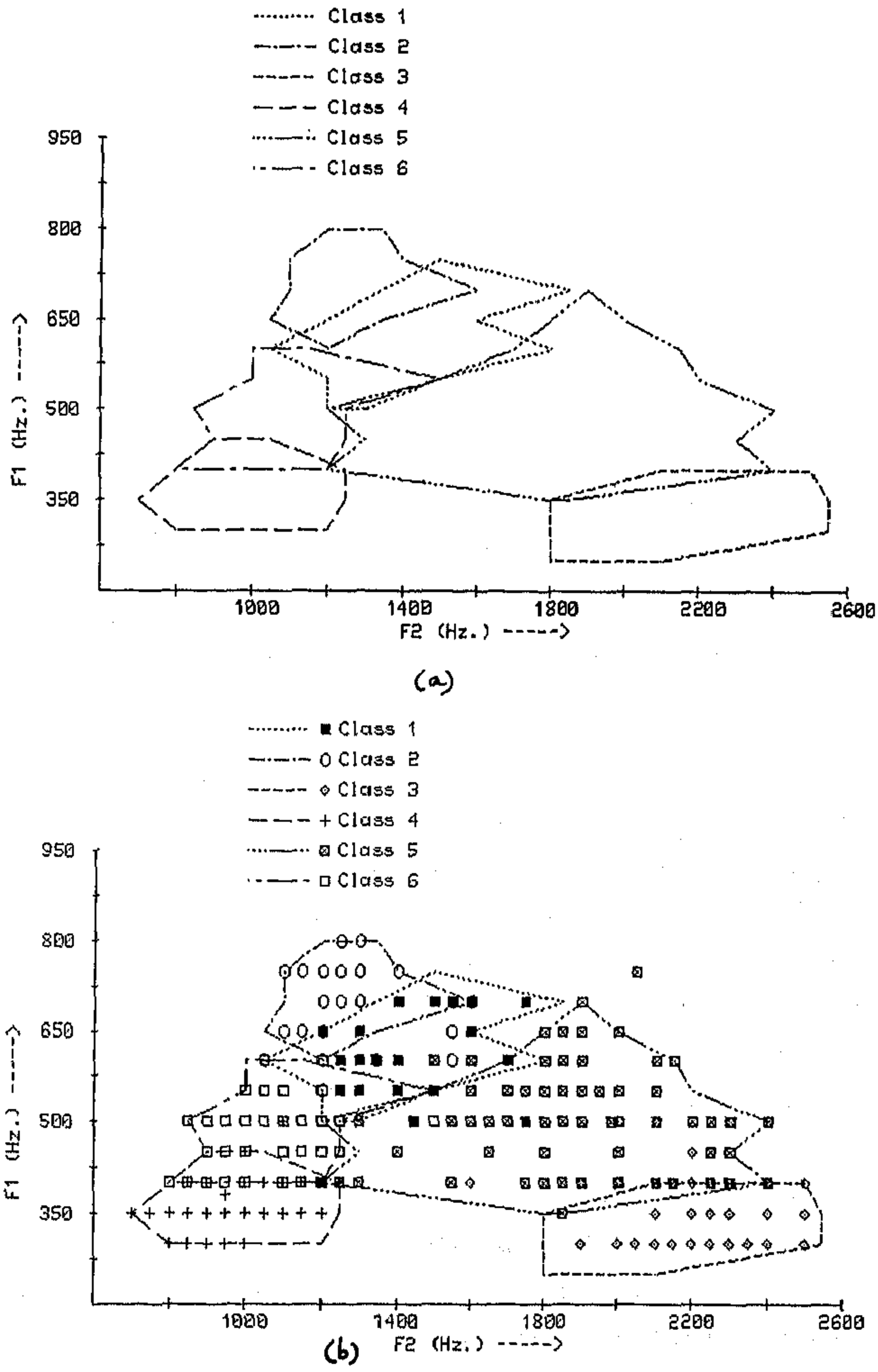

Figure 2.10: Output partitioning of pattern space generated by a five-layered net. (a) Class boundaries, with (b) the superimposition of the training samples

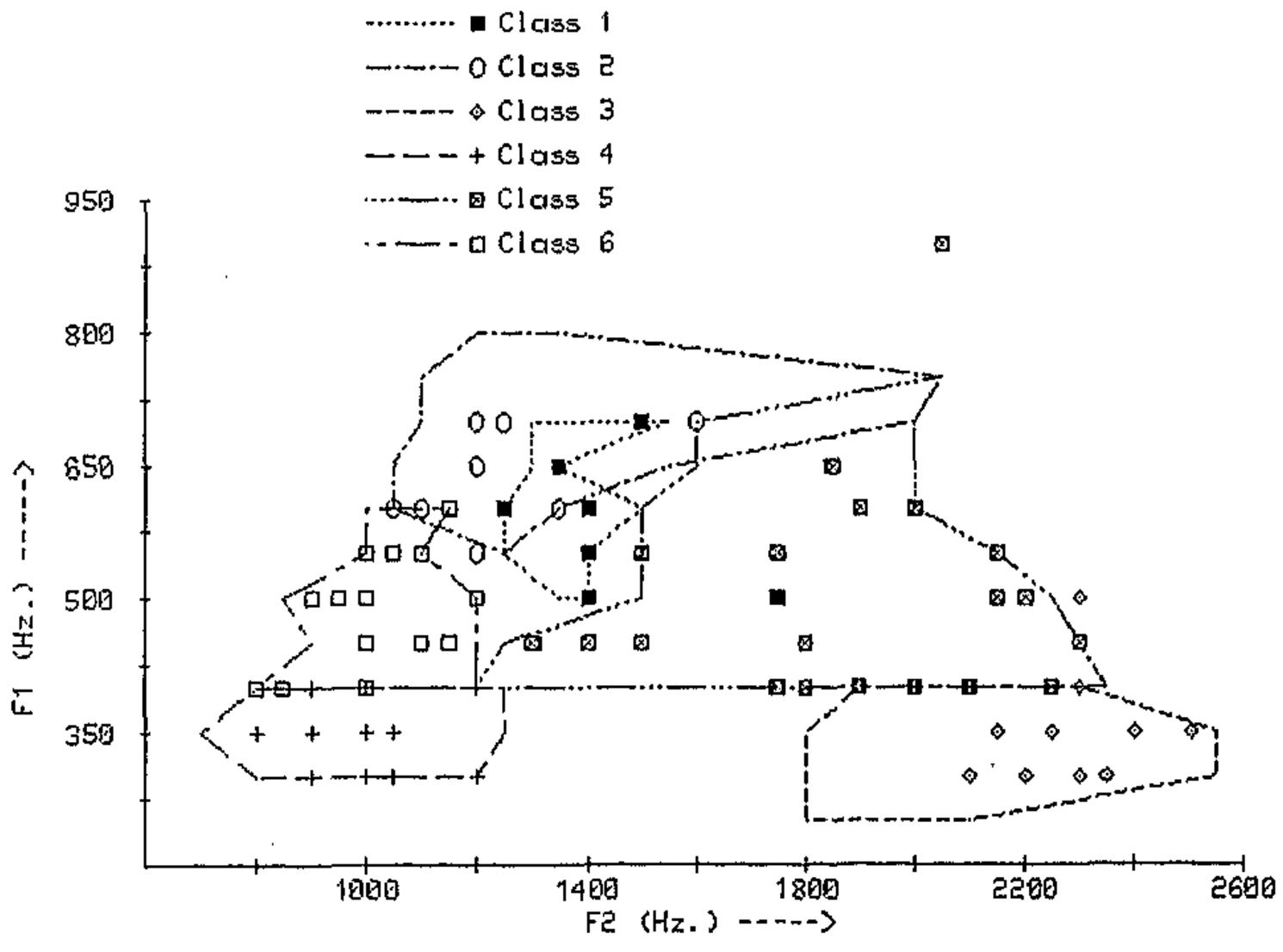

Figure 2.11: Output partitioning of pattern space generated by a four-layered net

weight updating in roughly the right direction without getting stuck at undesirable local minima. As the weight vectors approach a "correct" orientation, improvement in b, p and mse becomes slower, as indicated by the flatter nature of the curves for lower ϵ. The need to model finer intermediate output values requires ϵ to be gradually decreased, as explained in Section 2.5, until a suitably good solution is obtained. Note that the curves are neither smooth nor monotonically increasing/ decreasing. This is due to the *local* nature of the weight updating process that ultimately enables the neural net model to arrive at a *good global* solution by avoiding spurious local minima.

Table 2.1 is used to provide examples of input vectors $\vec{F}_i$, desired output vectors $\boldsymbol{d}$ and the actual output vectors $\boldsymbol{y}^H$ (both in the range [0,1]) for a set of sample patterns. Here the first three patterns correspond to training sets while the remaining three patterns belong to the test set. Note that although the desired output vector has a *hard* label for the 5^{th} entry (test pattern), indicating belongingness to class e alone, yet the proposed model generalizes to yield a fuzzy actual output vector in keeping with the vowel diagram of Fig. 2.6.

Table 2.2 compares the average percent correct recognition score (on the test set using the *best match* criterion, both class-wise and overall) of the fuzzy neural net model to that of a conventional MLP, its hard version and the standard Bayes' classifier. The

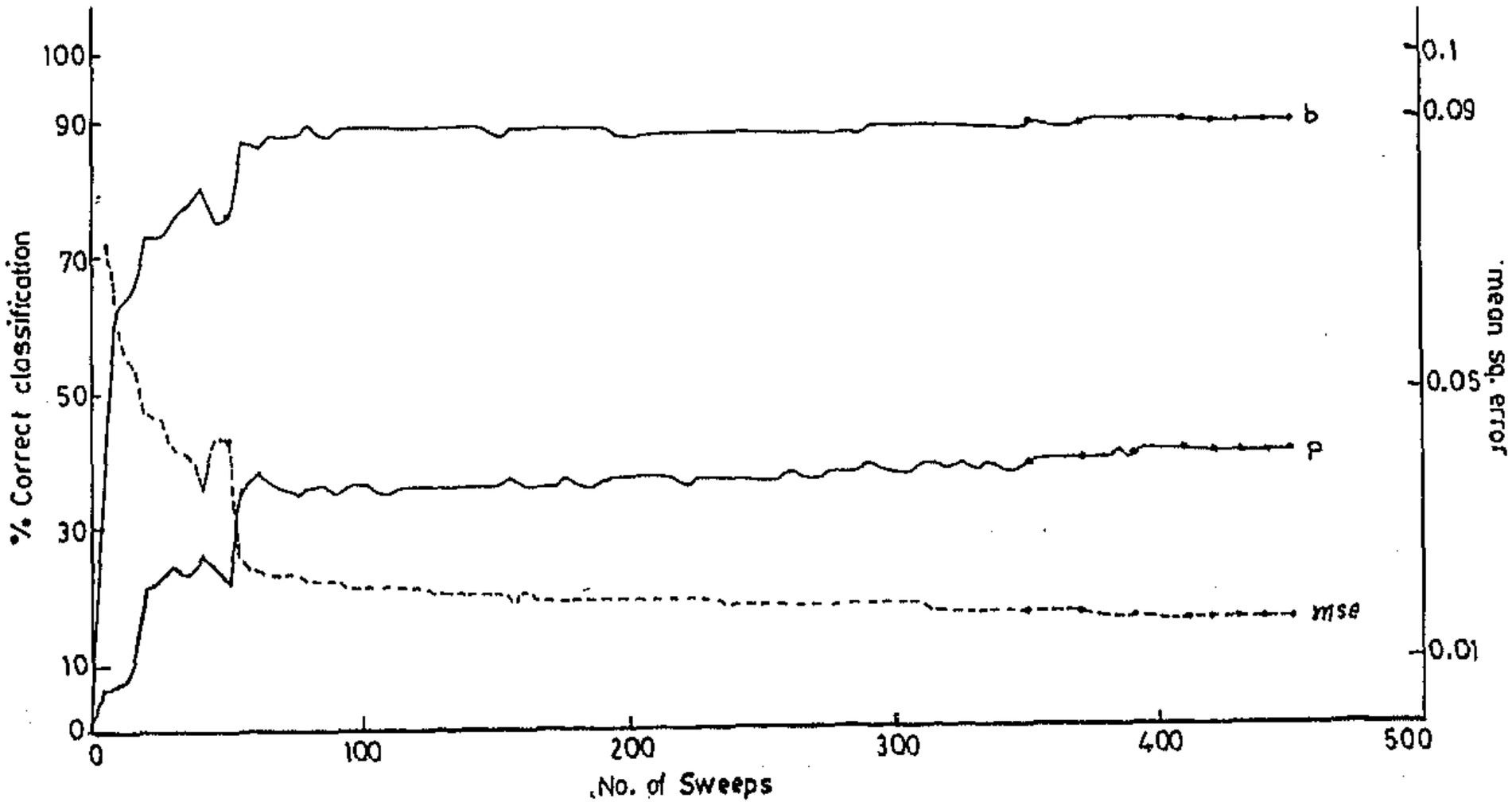

Figure 2.12: Variation of perfect match (%), best match (%) and mean square error with number of sweeps through the training set, as ϵ is decreased in discrete steps from $\epsilon_0 = 2$ to $\epsilon_q = 0.001$

Table 2.1: Input, desired output and actual output vectors for a set of sample patterns with the five-layered fuzzy MLP

Input features F_1, F_2, F_3	Input vector	Output vector	
		Desired	Actual
700,1500,2600	.07,.70,.93,.69,.96,.31,.30,.96,.70	.84,0.0,0.0,0.0,0.0,0.0,0.0	.87,.02,0.0,0.0,0.0,0.0
750,1150,2600	.01,.43,.99,.98,.47,.02,.30,.96,.70	0.0,.82,0.0,0.0,0.0,0.0,0.0	.03,.87,0.0,0.0,0.0,0.0
300,2200,2700	.94,.05,0.0,0.0,.29,.99,.15,.84,.85	0.0,0.0,.91,0.0,0.0,0.0,0.0	0.0,0.0,.96,0.0,0.0,0.0
350,700,2630	.99,.19,0.0,.82,0.0,0.0,.25,.93,.75	0.0,0.0,0.0,.76,0.0,0.0	.01,0.0,.03,.89,0.0,0.0
550,1500,2500	.62,.99,.38,.69,.96,.31,.50,1.0,0.5	0.0,0.0,0.0,0.0,.88,0.0	.58,.06,0.0,.04,.18,0.0
450,1200,2300	.93,.70,.07,.96,.58,.04,.85,.84,.15	0.0,0.0,0.0,0.0,0.0,.86	.14,.02,.06,.07,.07,.75

training and test sets each constitute 50% of the pattern data. A comparison of the overall performance on the training set between the various neural net model variations is also provided. For this both the *best match b* and *perfect match p* criteria are used. We have used the Bayes' classifier for multivariate normal patterns with the *a priori* probabilities $p_i = \frac{|C_i|}{N}$ where $|C_i|$ indicates the number of patterns in the i^{th} class and N is the total number of pattern points. The covariance matrices are different for each pattern class. The choice of normal densities for the vowel data has been found to be justified [289]. The MLP and its variations all use three hidden layers with m hidden nodes in each such layer. We have used $m = 10$ and 20 nodes in the current experiment. The first two MLP versions in Table 2.2 use *crisp* binary representation at the output. However in (a) the actual input features in the n-dimensional feature space are normalised to the range [0,1] and this is termed the *conventional* model. In (b) $3n$ linguistic input features with *crisp* values are used such that corresponding to a pattern $\vec{F}_i$, along the j^{th} axis, we clamp the highest of $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ and $\mu_{high(F_{ij})}(\vec{F}_i)$ of eqn. (2.10) to 1 while the remaining two are kept clamped at 0. This is referred to as the MLP model with hard linguistic input features. In (c) we use the fuzzy version of the MLP presented here.

Although model (a) is found to generate somewhat high recognition scores (%) using *best match*, yet we have observed that the outputs in most such *winning* cases were less than 0.5. This is perhaps an after-effect of the *crisp* output labeling used for the training data. Hence the output partitioning of the feature space using alpha-cut value $\alpha' = 0.5$ (as explained earlier) fail to generate the boundaries for classes ∂ and o (using $m = 20$) and for class o (using $m = 10$). Besides, the other class boundaries are also observed to be not very appropriate. This behaviour of models (a) and (b) correspond to the poorer recognition score (%) on the training set using the *perfect match* criterion p. Model (b) is found to have the worst efficiency. It is found to be able to generate all the output partitions, even though distorted in several cases. The *crisp* output labelings in (a) & (b) are perhaps therefore responsible for the resultant hindrance to *proper* convergence leading to *poor* output partitioning. However the fuzzy version of the MLP is found to provide a very satisfactory overall performance. Note also that although models (a) and (c) yield somewhat comparable performance on the test set, the fuzzy MLP (model (c)) has additional advantage of handling linguistic and imprecise inputs.

Table 2.2: Comparison of percent correct recognition score between Bayes' classifier and various neural net models

Model		Bayes' classifier	Neural net models					
			(a) Conventional		(b) Hard ling. input		(c) Fuzzy version	
			$m = 10$	$m = 20$	$m = 10$	$m = 20$	$m = 10$	$m = 20$
T	∂	41.6	55.5	44.4	24.4	34.1	51.2	55.5
e	a	91.1	86.6	88.8	64.4	60.0	84.4	80.0
s	i	93.0	81.4	81.4	88.8	90.9	81.9	91.5
t	u	94.7	88.1	90.7	76.2	55.7	86.8	86.6
s	e	71.1	92.3	86.5	69.7	65.6	92.4	85.9
e	o	71.1	86.6	85.5	85.9	91.4	89.5	81.3
t	Overall	79.2	84.6	82.8	72.5	70.2	84.2	83.6
perfect p		-	49.6	43.6	35.3	6.3	55.6	58.1
best b		-	86.0	87.6	77.7	71.5	92.2	92.2

It is to be noted that a good statistical (Bayes) classifier requires a lot of sequential computation and a large number of reference vectors while, on the other hand, a neural network is massively parallel and can generalize appreciably well. Incorporation of fuzzy concepts in the neural net further enhances its capability in handling the impreciseness in input patterns and the uncertainty arising from overlapping/ ill-defined regions.

Table 2.3: Comparison of recognition scores with Model O

Model		Fuzzy MLP	Variation O
T	∂	51.2	53.6
e	a	84.4	88.8
s	i	81.9	83.1
t	u	86.8	89.4
s	e	92.4	91.5
e	o	89.5	77.9
t	Overall	84.2	83.1
best b		92.2	91.7
perfect p		55.6	69.6
No. of sweeps		460	598

In Table 2.3 we demonstrate a comparison in performance of the fuzzy MLP with that of the model in [159] (referred to as Model O here). In Model O, the learning rate ϵ is modified dynamically by an additive increase when the error measure decreases and a multiplicative decrease otherwise. We use fuzzy input and output representations (as explained earlier) for both models (with three hidden layers having 10 nodes in each such layer, $hdec = 0$ and $perc = 50$), to specifically demonstrate the usefulness of our mode of variation of ϵ. The perfect match p and the best match b are provided for the training set. It is observed that the performance of the fuzzy MLP model is generally better in terms of the recognition scores (%). The output partitioning has also been found to be more appropriate in our model. Note that the total number of sweeps required is larger in the case of Model O.

2.7 Conclusions and discussion

A way of integrating the merits of fuzzy set theory in handling uncertainty in input and the ability of the MLP in generating highly nonlinear decision boundaries has been described in order to design a fuzzy version of the MLP. The model converts numerical and linguistic inputs to linguistic terms (*low, medium* and *high*), and provides fuzzy classification in terms of membership values. Backpropagated errors are assigned appropriate weightage for weight updating depending upon the membership values at corresponding outputs. The learning rate ϵ and damping coefficient α are gradually decreased to prevent oscillations as the neural network converges to a minimum error solution. The problem of vowel recognition has been considered to demonstrate the effectiveness of the model.

It may be noted that the parameters ϵ and α traverse a range of values in the course of the computations and, as such, no particular choices need to be made. However, in general, one may choose initially $0.6 < \alpha < 1.0$, $1.0 < \epsilon < 2.5$ and finally $0.2 < \alpha < 0.6$, $0.0001 < \epsilon \leq 0.1$ for good results. This is because initially high ϵ and α, and finally very low ϵ but not so low α are needed as explained in Section 2.5.2. Further, usually $0 \leq hdec < 0.01$.

It is worth mentioning that the fuzzy MLP in [290] used trapezoidal possibility distributions to represent each linguistic term, sampled at a fixed number of values over their respective domains of discourse, for fuzzy inferencing. Hence the sampling frequency had a direct bearing on the faithfulness of the representation of the linguistic terms as well as the cost of calculation required. On the other hand, we have considered the $\pi-$function to model the inputs in terms of the linguistic variables. This has resulted in a more cost-effective representation with less inputs dedicated to a particular input feature. The concept of using class membership functions for fuzzy modeling of multi-class problems is also basically different from the two-class single-layer-perceptron based fuzzy pattern classification algorithm [202].

Note that only three linguistic terms *low, medium* and *high* have been used in the input stage of the model. Incorporation of the fuzzy hedges like *more or less, very, nearly,* etc. as additional properties might enhance performance of the model, due to the resulting more detailed input description, but then the cost of nodes and interconnections would also increase. It is also to be noted that we have allowed significant

amount of overlapping between the functions *low, medium* and *high* (Figure 2.4) so that the results obtained from the system become more meaningful (valid).

In this connection, it may be mentioned that the MLP has been used to approximate continuous functions for prediction in time series [177]. This approach of input-output representation was basically different from our model, although in both cases the range of allowable values was [0,1]. We have used fuzzy membership concepts for modeling both the input and output vectors.

In the algorithm reported in [208], the least number of iterations was evaluated using a different approach. There an attempt was made to generate the *"best"* solution, and this was rather expensive, as it required computations involving results from various runs over several ratios of training and test set combinations. The fuzzy MLP model, on the other hand, has applied a heuristic in an attempt to obtain a *good* solution for any chosen ratio of training-test set combination in a single *run* (one sequence of *sweeps*) over this set.

Neural net performance for fuzzy classification of speech data has been found to compare favourably with that of the Bayes' classifier trained on the same data. In the fuzzy MLP model we have used many parallel interconnection links with simple processing elements. Therefore with appropriate parallel hardware this model should be able to perform much faster and hence more efficiently than serial techniques. The incorporation of fuzzy concepts at various levels has enhanced the performance and the capability in handling imprecise/ incomplete input. Representation of input in terms of *low, medium* and *high* will help the system in accepting feature information also in *set* form, as demonstrated in Chapter 5. Besides, *missing* input information can also be handled.

Regarding the relative merits of the different numbers of hidden layers and/or the total number of neurons, it might be noted that the approach in [170] used an algorithm making weight updates only once per sweep. Here, on the other hand, the weights have been updated after each pattern presentation as explained in Section 2.5.1 with reference to [286]. In this work we have demonstrated a few experimental observations in this direction. For this we have used m nodes in each hidden layer to bring uniformity in the design of the neural network. We have not considered different numbers of nodes in the various hidden layers in order to simplify the building procedure. Incorporation of pruning and further addition of layers as in [170], to improve the generalizing

capabilities of the network, seems non-trivial in the current context and might be an interesting topic for future investigation.

Chapter 3

Kohonen's Net, Fuzzy Sets and Pattern Classification

3.1 Introduction

The present chapter discusses a self organizing neural network model that performs fuzzy classification [274]. It is an attempt to extend Kohonen's model [41] by incorporating fuzzy set-theoretic concepts at various stages. In the process, a separate testing phase is added to evaluate the performance of the classifier in recognizing a separate set of test patterns. We consider a single layer two-dimensional rectangular array of neurons with short range lateral feedback interconnections between neighboring units.

The network under consideration passes through two stages, *viz.*, self organization and testing. In the first stage a set of training data is used by the network to initially self organize the connection weights and finally *calibrate* the output space. During this stage the weight vector most similar to the input pattern vector is rotated towards the latter. The neighboring weight vectors are also rotated, but by a lower amount. After a number of sweeps through the training set the output space becomes appropriately ordered. An index of disorder is computed to evaluate a measure of this ordering. The network is now supposed to encode the input space information among its connection weights. By calibration we refer to the labeling of the neurons, after self organization, relative to the training pattern classes. This procedure also provides some qualitative assessment of the topological ordering of the output space as compared to the input data space.

During training, the input vector also includes some contextual information regarding the finite output membership of the pattern to one or more class(es). Compared to the conventional two-state system, which assigns membership to one class only and uses no class information in the input, this technique produces a more efficient modeling in cases where the feature space has overlapping or ill-defined clusters. However, during self organization, this part of the input vector is assigned a lower weight to allow the linguistic and/ or quantitative input properties to dominate.

During calibration, only the class membership information in the input vector is used (in *crisp* form) while the input feature information is kept clamped at zero. In the conventional Kohonen's model, after self-organization, the training pattern vectors are used to label the neurons to which they are mapped. This gives the ordering of the pattern classes in the output space. In the fuzzy Kohonen's net-based model, the labeling of the output neurons is determined solely by the contextual class information

associated with the training pattern vectors. This is termed as *calibration* of the neurons. Each neuron is labeled by the pattern class for which it generates the highest response. This corresponds to a *hard* partitioning of the neurons. A *fuzzy* partitioning of the output space is also generated to produce an appropriate topological ordering with fuzzy data.

In the second stage a separate set of test patterns is supplied to the network and the resulting neuronal outputs verified against the calibrated output map. This is an extension to the conventional Kohonen's model which basically performs a clustering operation. Our model, on the other hand, is designed to be a classifier. The *calibrated* neurons *self-organized* by the training set, are used to evaluate the recognition capability (using *best* match) of the said trained neural net on the test set. Now the input vector contains only the feature information. A confusion matrix is generated to evaluate the classification performance (based on *best* match) of the network on the test set. The output is generated in terms of fuzzy class membership values.

The fuzzy Kohonen's network is capable of handling input features presented in quantitative and/ or linguistic form. As mentioned in the previous chapter, the components of the input vector consist of the membership values to the overlapping partitions of linguistic properties *low, medium* and *high* corresponding to each input feature. This creates the possibility of incorporating linguistic information into the model, if necessary, and enhances its robustness in handling imprecise or uncertain input specifications.

Like the previous chapter, the effectiveness of the model is demonstrated on the speech recognition problem where the classes have ill-defined, *fuzzy* boundaries. Comparison is made with the standard Bayes' classifier and the conventional Kohonen's net, and the performance of the said model is found to be quite satisfactory.

Given the burgeoning interest in fuzzy self-organizing maps [205, 206, 291], it is worth highlighting the major contribution of the present chapter. Basically, the Kohonen clustering network is used here as a symbol map. There are phenomena which are inherently fuzzy but which are associated with physical manifestations that can be characterized quite precisely by physical measurements. Clustering or classifying solely on the basis of these physical measurements is not useful, however, because meaningful clusters can be constructed only with the assistance of additional factors which cannot be elucidated directly from these physical measurements. Human language, probably

at all levels but especially in the area of phonology, is perhaps the best example of such a phenomenon. Thus, while a listener recognizes a phoneme from physical cues alone, exactly which phoneme class a particular conflation of physical features is assigned to by a listener depends on factors which are not inherent in these physical features (e.g., the formant values used here), but which depend on physically extraneous factors such as (in particular but not limited to) the language the listener assumes is being spoken. There are also, for many reasons, variations among speakers such as are evident in the vowel data used here (Fig. 2.6). Thus, assignment of speech sounds to phonemes yields clusters which are fuzzy at the very least in the sense that different listeners may disagree on what they believe themselves to be hearing and that different speakers may produce different physical manifestations of the same phoneme. The essential properties of phoneme clusters, therefore, must be elucidated by appeal to essentially psycholinguistic experimentation of one kind or another. Now, how can one build a self-organizing network which can perform this same classification? Simply by doing exactly what we have done, which is to replace the arbitrary encoding of the abstract portion of the data vectors with fuzzy class memberships. Note that this violates Ritter and Kohonen's *no information about similarities between the items* condition ([292], p. 247), but it does not matter, because a kind of orthogonality is maintained by the fact that $\boldsymbol{x}_a$ (attribute part) and $\boldsymbol{x}_s$ (symbol part) of the data vectors here are characterized by different *levels* of description (phonetic and phonemic). The value of this approach is manifested in calibration (clustering, labeling) and in classification, since the organized network yields a good fuzzy clustering of the neurons after calibration and functions as an effective fuzzy classifier. Thus, where there is reason to believe that the elements of $\boldsymbol{x}_s$ and $\boldsymbol{x}_a$ relate to each other not so much as purely arbitrary and purely physical (or at least arbitrary, in some sense) but rather as two levels of abstraction, and where there is reason to believe that at least one of the levels (the *higher* one) is fuzzy, the fuzzification of $\boldsymbol{x}_s$ is justifiable and yields excellent results. Attempts at crisp calibration and/or the use of purely arbitrary class labels (as in the pure Ritter and Kohonen approach, where the labels (the semantic concepts) are not connected to each other except through the data vectors they label) in such cases will prove to be fruitless. Note that this does indeed amount to a kind of partial supervision as we have suggested, but it is an extremely interesting kind of partial supervision in that it arises from reasonable assumptions about the nature of human language itself (*i.e.*, its multi-lingual properties) and not directly from expert intervention (i.e., the

learning is guided not by intelligence but by intuition)!

Applications of the semantic self-organizing Kohonen's feature maps have also been reported by Ichiki *et al.* [293] and Fritsch *et al.* [294] (for classifying patients suffering from Parkinson syndrome).

3.2 Kohonen's neural network model

The essential constituents of Kohonen's neural network model are as follows [41, 42, 143, 295, 296] :

- an array of neurons receiving coherent inputs and computing a simple output function,
- a mechanism for comparing the neuronal outputs to select the neuron producing maximum output,
- a local interaction between the selected neuron and its neighbors,
- an adaptive mechanism that updates the interconnection weights.

Consider the self-organizing network given in Fig. 3.1. Let M input signals be simultaneously incident on each of an $N \times N$ array of neurons. The output of the i^{th} neuron is defined as

$$\eta_i(t) = \sigma \left[[\boldsymbol{m}_i(t)]^T \; \boldsymbol{x}(t) + \sum_{k \in S_i} w_{ki} \; \eta_k(t - \Delta t) \right] \tag{3.1}$$

where $\boldsymbol{x}$ is the $M-$dimensional input vector incident on it along the connection weight vector $\boldsymbol{m}_i$, k belongs to the subset S_i of neurons having interconnections with the i^{th} neuron, w_{ki} denotes the fixed feedback coupling between the k^{th} and i^{th} neurons, $\sigma[.]$ is a suitable sigmoidal output function, t denotes a discrete time index and T stands for the transpose.

If the best match between vectors $\boldsymbol{m}_i$ and $\boldsymbol{x}$ occurs at neuron c, then we have

$$\|\boldsymbol{x} - \boldsymbol{m}_c\| = \min_i \|\boldsymbol{x} - \boldsymbol{m}_i\|, \;\; i = 1, 2, \ldots, N^2 \tag{3.2}$$

where $\|.\|$ indicates the Euclidean norm.

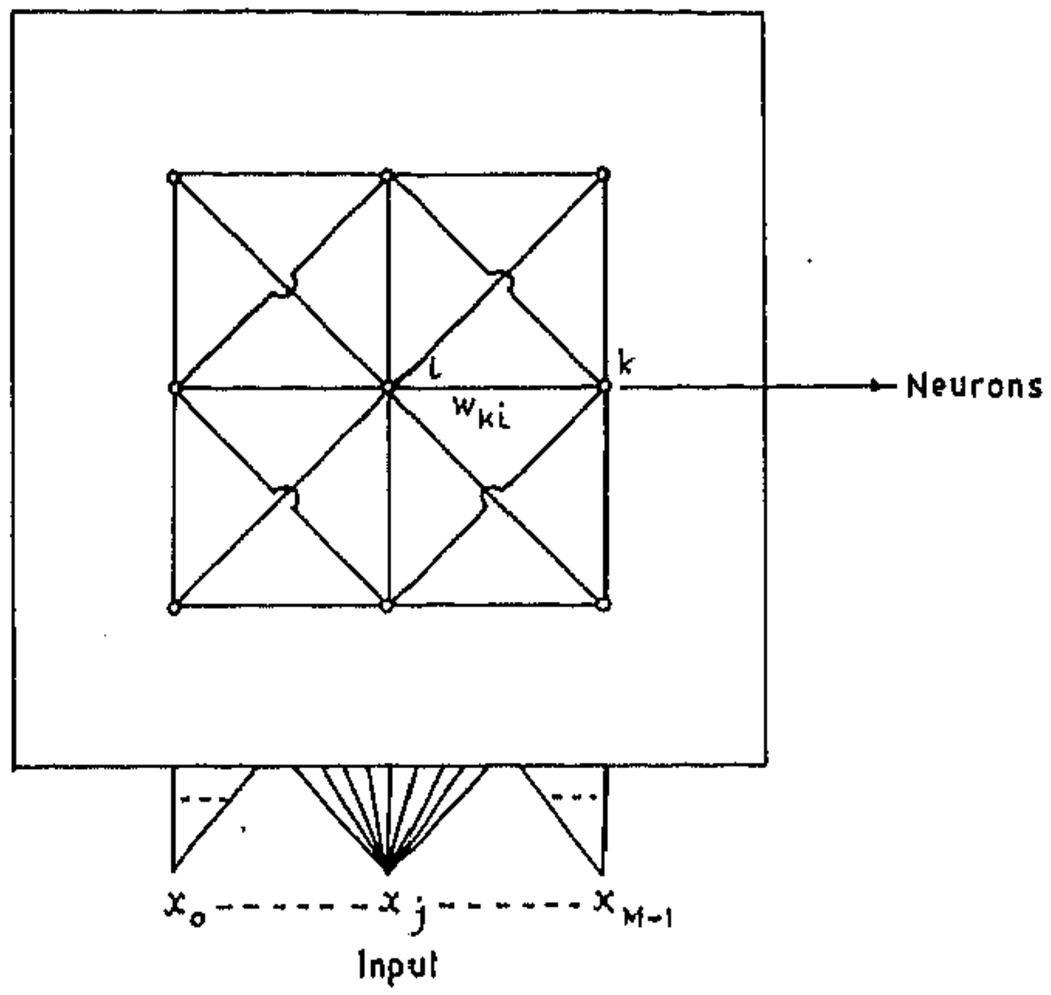

Figure 3.1: Kohonen's neural network model

The weight updating is given by [41, 143] as

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)\,(x(t) - m_i(t)) & \text{for } i \in N_c \\ m_i(t) & \text{otherwise} \end{cases} \tag{3.3}$$

where $\alpha(t)$ is a positive constant that decays with time and N_c defines a topological neighborhood around the maximally responding neuron c, such that it also decreases with time. Note that $\alpha(t)$ is a particular case of the more general Gaussian term $h(x,t)$ [296]. Different parts of the network become selectively sensitized to different inputs in an ordered fashion so as to form a continuous map of the signal space. After a number of sweeps through the training data, with weight updating at each iteration obeying eqn. (3.3), the asymptotic values of m_i cause the output space to attain proper topological ordering. This is basically a variation of *unsupervised* learning. The self organization using training patterns enables the ordering of the output neurons. These may then be calibrated with the class information by applying labeled training patterns at the input.

3.3 Incorporation of class information in input vector during training

The input to the fuzzy Kohonen's net consists of two portions. In addition to the linguistic properties (already discussed in Section 2.3 with reference to the fuzzy MLP), there is also some contextual information [292] regarding the fuzzy class membership [17] of each pattern used as training data during self organization of the network.

In the traditional Kohonen's net model [41, 42], the input vector consists of quantitative information only regarding the patterns. Generally the training patterns, used during self organization, are also used later for calibrating the output space. This refers to a *hard* labeling of the output neuron by the pattern class corresponding to a training pattern for which it elicits the maximum response. A qualitative measure of the topological ordering of the output space may be obtained from calibration. Note that during self-organization the model clusters the training patterns whereas during calibration it labels these clusters with some additional class information. So the training phase is completely unsupervised while calibration is not. Then, we could add a testing phase to obtain a hard classification of a set of test data by assigning a membership value of 1 to only that class corresponding to the partition of the neuron (labeled during calibration) eliciting the maximum response.

In many real-life problems, the data are generally ill-defined with overlapping or fuzzy class boundaries. Each pattern used in training may possess finite belongingness to more than one class. To model such data, it often becomes necessary to incorporate some contextual information regarding class membership as part of the input vector. However during self-organization this part of the input vector is assigned a lower weightage so that the linguistic properties dominate in determining the ordering of the output space. During calibration we use the contextual class membership information part of the input vector (in *crisp* form as in eqn. (3.6)) only for determining the *hard* labeling of the output space. A separate fuzzy partitioning, that allows scope for producing overlapping clusters, is also introduced. It has been observed that the inclusion of this contextual class membership information produces more efficient self-organization and is necessary in handling fuzzy or imprecise data. This is perhaps because in addition to the associated higher input space dimensionality, some sort of partial supervision is used here instead of the completely unsupervised functioning of

the conventional Kohonen's model.

While the traditional Kohonen's model was used for clustering purposes, this fuzzy version has been extended to function as a classifier. Therefore, here the output classes are known. We use partial supervision in the form of providing a *lower weighted* contextual class membership information during self-organization. We also use a testing phase to evaluate the recognition performance of the *calibrated* neurons on a separate set of test data. Note that only the class membership information (in *crisp* form as in eqn. (3.6)) is used during calibration. But during testing, only the input feature information of the test patterns is used. The calibrated partition of the neuron eliciting the maximum response for a test pattern is inferred to be its recognized class. The value of the inferred class membership of the test pattern is also dependent on the membership of the responding neuron (during calibration) to the appropriate partition. This scheme also enables better generalization so that the performance of the network while inferring the class membership of the test data is observably superior as compared to the conventional model. This claim is experimentally demonstrated in Section 3.5.

3.3.1 Class membership as contextual information

The pattern $\vec{F}_i$ is considered to be presented as a concatenation of the linguistic properties in eqn. (2.10) and the contextual information regarding class membership. Let the input vector be expressed as

$$\boldsymbol{x} = [\boldsymbol{x}', \boldsymbol{x}'']^T = [\boldsymbol{x}', 0]^T \cup [0, \boldsymbol{x}'']^T \tag{3.4}$$

where $\boldsymbol{x}'$ contains the linguistic information in the $3n$-dimensional space of eqn. (2.10) and $\boldsymbol{x}''$ covers the class membership information in an l-dimensional space for an l-class problem domain. So the input vector $\boldsymbol{x}$ lies in an $(3n + l)$-dimensional space. Both $\boldsymbol{x}'$ and $\boldsymbol{x}''$ are expressed as membership values. The representation of $\boldsymbol{x}'$ has already been discussed in Section 2.3 with reference to the fuzzy MLP. Here we consider the definition of $\boldsymbol{x}''$.

Applying the membership concept

For the i^{th} pattern we define

$$x'' = \begin{cases} s * \left[\mu_{INT(1)}(\vec{F}_i), \ldots, \mu_{INT(l)}(\vec{F}_i)\right]^T & \text{in the } \textit{fuzziest} \text{ case} \\ s * \left[\mu_1(\vec{F}_i), \ldots, \mu_l(\vec{F}_i)\right]^T & \text{otherwise} \end{cases} \tag{3.5}$$

where $0 < s \leq 1$ is the scaling factor. To ensure that the norm of the linguistic part x' predominates over that of the class membership part x'' in eqn. (3.4) during self-organization, we choose $s < 0.5$. The variables $\mu_{INT(k)}(\vec{F}_i)$ and $\mu_k(\vec{F}_i)$ are defined by eqns. (2.15)-(2.17), with reference to the output vector of the fully supervised fuzzy MLP of Chapter 2.

Fig. 3.2 provides a representation of the input vector x that is used by the model. Note that unlike the model in [292], we define the part x'' of the input vector x in terms of membership functions that attain values in the interval [0,1] and provide a measure of belongingness to the corresponding fuzzy set. During self organization we allow partial supervision involving $s(< 0.5)$ times the class membership information, such that this knowledge may also be incorporated into the connection weight values. This enables a training pattern with membership (say) 0.9 to class C_{k_1} be mapped perhaps to a neuron that is not the same as that to which another training pattern with membership (say) 0.5 to class C_{k_1} or (say) 0.5 to class C_{k_2} is mapped.

3.3.2 Modification of input during calibration

During calibration of the output space the input vector chosen is $x = [0, x'']$, where x'' is given by eqn. (3.5) such that

$$\mu_q(\vec{F}_i) = \begin{cases} 1 & \text{if } q = k \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

for $k \in \{1, \ldots, l\}$ and $s = 1$. The N^2 neuron outputs η_i are calibrated with respect to the l classes. Here the class information of the training patterns is given full weightage while the input feature information is suppressed. The primary objective of this stage is to label each neuron by the class (partition) for which it elicits the maximum response. The resulting *hard* (labeled) partitioning of the output space may be used to qualitatively assess the topological ordering of the pattern classes with respect to the

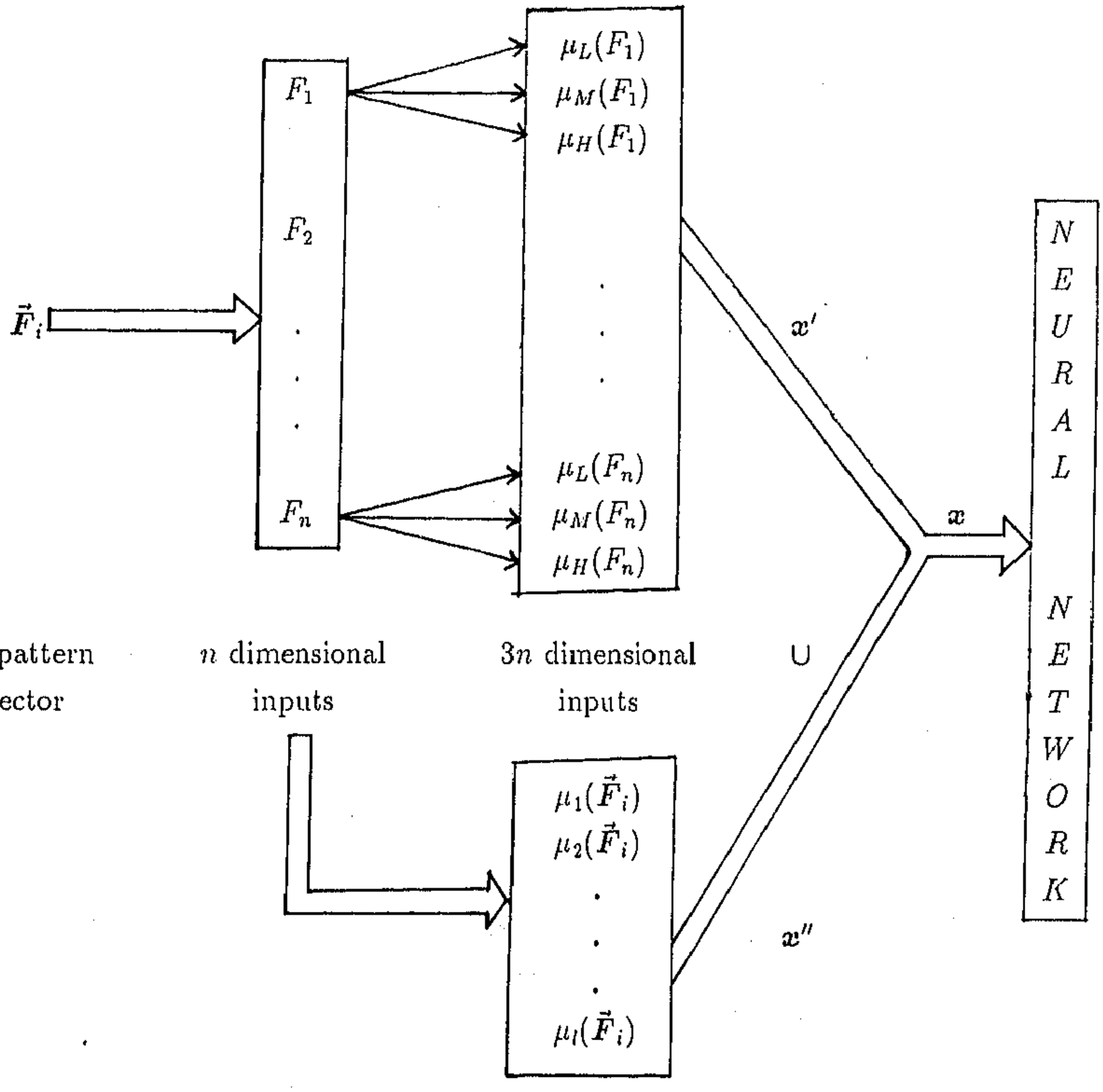

Figure 3.2: Block diagram of the input phase

input feature space. Note that while x'' contains class membership information during self organization, we use binary x'' at the input during calibration. We introduce a *fuzzy* partitioning of the output space by also labeling the output neurons with the fuzzy membership values of their output responses. This helps generate overlapping partitions of the output space and are hence closer to the input feature space representation in case of fuzzy data. This concept is explained in detail in Section 3.4.3.

Let us consider the following situation. A pattern having class memberships of (say) 0.52 to class C_{k_1} and 0.48 to class C_{k_2} may be mapped to a neuron i (eliciting maximum response) that is calibrated as belonging to the *hard* partition of class C_{k_1}. However we should note that the lower yet significant belongingness of the said pattern to class C_{k_2} ought not be ignored. Herein lies the utility of the *fuzzy* partitioning. By this, the particular neuron i may be calibrated as belonging to both the classes C_{k_1} and C_{k_2}, albeit with different membership values.

It is to be noted that the traditional Kohonen's net model uses unsupervised learning during self organization. During calibration, the training patterns or some reference vectors (in case of known samples) are used for the *hard* labeling of the neurons. This provides some insight into the topological ordering of the output space thus partitioned. In the semantic maps [292], on the other hand, the class information is used in this stage to generate the *hard* labeling of the partitions during calibration. Both these models deal with clustering problems. We, on the other hand, concentrate on classification. Our concept of fuzzy partitioning is an extension in this context. We introduce a separate testing phase where a different set of fuzzy test patterns (kept aside from the original data set while randomly selecting the training set for self organization) are classified using the input feature information of the test vector along with the above-mentioned fuzzy partitioning information. This procedure is explained in detail in Section 3.4.4.

3.4 Fuzzy extension to Kohonen's algorithm

Consider an $(3n + l)$-dimensional input space with the input vector $x = [x', x'']^T$ of eqn. (3.4) being incident simultaneously on the $N \times N$ array of neurons.

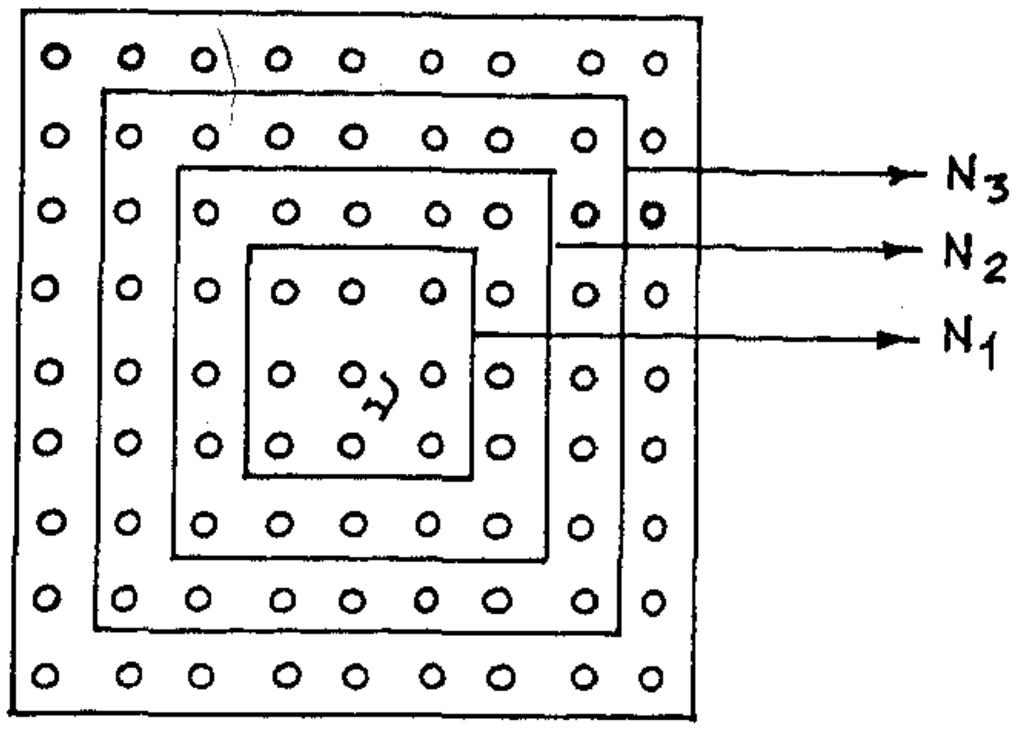

Figure 3.3: Topological r-neighborhoods N_r

Concept of r-neighborhood

Each neuron $\nu(ii, jj)$ has a topological r-neighborhood $N_r(ii, jj)$, as depicted in Fig. 3.3, where ii, jj denote the row and column numbers respectively of the neuron. We have

$$N_r(ii, jj) = \{\nu(u, v) | \max\{|u - ii|, |v - jj|\} = r\}, 1 \leq u, v \leq N \tag{3.7}$$

where $r = 0, 1, \ldots, 3$. The neighborhood starts large and slowly decreases in size over time from $r = 3$ to $r = 1$. Note that the indices ii and jj will be omitted in future reference to avoid clutter.

Output of a neuron

The output of the i^{th} neuron is computed using eqn. (3.1), with the subset S_i of neurons being defined as its r-neighborhood N_r. We choose

$$\sigma(q) = \begin{cases} 0 & \text{if } q < 0, \\ q & \text{otherwise} \end{cases} \tag{3.8}$$

This transformation ensures that $\sigma(q) \geq 0$. We also use

$$w_{ki} = \begin{cases} b & \text{for } r = 1, \\ -\frac{b}{2} & \text{for } r = 2, \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Here b is the mutual interaction weight such that the lateral coupling w_{ki} has the form of a *Mexican hat.*

3.4.1 Weight updating

Initially the components of the m_i's are set to small random values lying in the range [0,0.5]. Let the best match between vectors m_i and x, selected using eqn. (3.2), occur at neuron c. Using eqn. (3.3), the weight updating expression may be stated as

$$m_i(t+1) = \begin{cases} m_i(t) + h_{ci} * (x(t) - m_i(t)) & \text{for } i \in N_r,\ r = 0, 1, \ldots, 3 \\ m_i(t) & \text{otherwise} \end{cases} \tag{3.10}$$

where N_r defines a r-neighborhood by eqn. (3.7) around neuron c such that r decreases with time. Here the gain factor h_{ci} is considered to be bell-shaped like the $\pi-$function, such that $|h_{ci}|$ is the largest when $i = c$ and gradually decreases to zero with increasing distance from c. Besides, $|h_{ci}|$ also decays with time.

Gain factor

We define

$$h_{ci} = \frac{(1 - r * f) * \alpha}{\left[1 + \left(\frac{nt}{cdenom}\right)^2\right]} \tag{3.11}$$

where nt is the number of sweeps already made through the entire set of training samples at any point of time, *cdenom* is a positive constant suitably chosen and $0 < f, \alpha < 1$. The decay of $|h_{ci}|$ with time is controlled by nt. The slowly decreasing radius of the *bell-shaped* function h_{ci} and the corresponding change in $|h_{ci}|$ are controlled by the parameters r and f. The utility of this choice of h_{ci} is demonstrated later in Section 3.5. Due to the process of self-organization, the randomly chosen initial m_i's gradually attain new values according to eqns. (3.2) and (3.10) such that the output space acquires appropriate topological ordering.

3.4.2 Index of disorder

An index of disorder D may be defined to provide a measure of this ordering. Let *msd* denote the mean square distance between the input vector and the weight vectors in the r-neighborhood of neuron c. We define

$$msd = \frac{1}{|trainset|} \sum_{x \in trainset} \left[\sum_{r=0}^{3} \left\{ \left(\frac{1}{|N_r|} \sum_{i \in N_r} \|x - m_i\|^2 \right) * (1 - r * f) \right\} \right] \tag{3.12}$$

where $|trainset|$ refers to the number of input pattern vectors in the training set. This definition ensures that neurons nearer c (smaller r) contribute more to msd than those farther away. Also

$$f = \begin{cases} \frac{1}{4}, & 0 \leq r \leq 3 \quad \text{for } ncnt = 1, \\ \frac{1}{3}, & 0 \leq r \leq 2 \quad \text{for } ncnt = 2, \\ \frac{1}{2}, & 0 \leq r \leq 1 \qquad \text{otherwise} \end{cases} \tag{3.13}$$

Here $|N_r|$ denotes the number of neurons in the r-neighborhood of neuron c such that $|N_1| \leq 8$, $|N_2| \leq 16$ and $|N_3| \leq 24$ depending upon the position of c in the two-dimensional array. Note that N_0 implies neuron c itself.

The expression for the index of disorder is given as

$$D = msd(nt - kn) - msd(nt) \tag{3.14}$$

where $msd(nt)$ denotes the mean square distance by eqn. (3.12) at the end of the nt^{th} sweep through the training set and kn is a suitable positive integer such that D is sampled at intervals of kn sweeps. Initially $ncnt$ is set to 1. Then

$$ncnt = \begin{cases} ncnt + 1 & \text{if } D < \delta, \\ ncnt & \text{otherwise} \end{cases} \tag{3.15}$$

where $0 < \delta \leq 0.001$. The process is terminated when $ncnt > 3$, so that in eqn. (3.13) we always have $r \geq 1$. For good self organization, the value of msd and therefore D should gradually decrease. It is to be noted that r and f of eqns. (3.10)-(3.11), that control $|h_{ci}|$, obey eqn. (3.13). The parameter $ncnt$ of eqn. (3.15), depending on D of eqn. (3.14), controls r and f of eqn. (3.13).

3.4.3 Partitioning during calibration

During calibration the input vector $\boldsymbol{x} = [0, \boldsymbol{x}'']$ of eqn. (3.4) is applied to the neural network. Let the $(i1)_k^{th}$ neuron generate the highest output η_{f_k} for class C_k. We define a membership value for the output of neuron i when calibrated for class C_k simply as

$$\mu_k(\eta_i) = \frac{\eta_{i_k}}{\eta_{f_k}} \text{ for } i = 1, \ldots, N^2, \text{ and } k = 1, \ldots, l \tag{3.16}$$

such that $0 \leq \mu_k(\eta_i) \leq 1$ and $\mu_k(\eta_i) = 1$ for $i = (i1)_k$.

Each neuron i may be marked by the output class C_k, among all l classes, that elicits the maximal response η_{i_k}. This generates a hard partitioning of the output space that is used in Ritter and Kohonen's model [292].

Fuzzy partitioning

On the other hand, each neuron i has a finite belonging or output membership $\mu_k(\eta_i)$ to class C_k by eqn. (3.16). We may generate the *crisp* boundaries for the fuzzy partitioning of the output space by considering for each of the l classes the α-cut set $\{i|\mu_k(\eta_i) > \alpha'\}$, $0 < \alpha' \leq 1$, where α' is a suitably chosen value. This is done solely for the ease of depiction of the various partitions in the output space. Note that the generation of overlapping fuzzy partitions for the fuzzy input data demonstrates the utility of the process.

An ordered and unbroken map of the output space indicates good self organization and hence grouping of the patterns according to similarity. In cases where the data are fuzzy and overlapping classes exist, the hard partitioning contains apparent disorder and/or discontinuity. It has been observed that the incorporation of the fuzzy membership concept alleviates this problem. The utility of the fuzzy approach may be appreciated by considering a point lying in a region of overlapping classes in the feature space. In such cases its membership to each of these classes may be nearly equal. Then there is no reason why we should follow the *hard* approach of calibrating the corresponding neuron (generating maximum response for such a pattern point) with the class for which it elicits a slightly larger response and thereby neglect the smaller yet significant response(s) obtained for the other overlapping class(es).

3.4.4 Testing phase

After self organization, the fuzzy Kohonen's net encodes all input data information distributed among its connection weights. The class membership of the training patterns is also *learned* due to the partial supervision used in that stage. During calibration, the neurons are labeled by the pattern classes and the corresponding membership values are assigned. This is the desired fuzzy classifier. In the final stage, a separate set of test patterns is supplied as input to the neural network model and its performance evaluated.

During this phase the input test vector $x = [x', 0]^T$, consisting of only the linguistic information in the $3n$-dimensional space defined by eqn. (2.10), is applied to the network. Let the $p1^{th}$ and $p2^{th}$ neurons generate the highest and second highest outputs

η_{f_p} and η_{s_p} respectively, for test pattern p. Besides, let $\mu_{k_1}\left(\eta_{f_{pm}}\right)$ and $\mu_{k_2}\left(\eta_{s_{pm}}\right)$ be the highest and second highest output membership values generated during testing, with respect to classes C_{k_1} and C_{k_2} respectively. It is to be noted that $k_1 = k_2$ for both choices for pattern points not lying in regions of overlapping classes and there is no ambiguity of decision in such cases. We define

$$\begin{aligned} \mu_{k_1}\left(\eta_{f_{pm}}\right) &= \mu_{k1}\left(\eta_{p1}\right), \\ \mu_{k_2}\left(\eta_{s_{pm}}\right) &= \frac{1}{\eta_{f_p}}\mu_{k2}\left(\eta_{p2}\right) * \eta_{s_p}, \end{aligned} \tag{3.17}$$

and $k_1 = k1$, $k_2 = k2$, if $\mu_{k1}\left(\eta_{p1}\right) \geq \frac{1}{\eta_{f_p}}\mu_{k2}\left(\eta_{p2}\right) * \eta_{s_p}$. Otherwise,

$$\begin{aligned} \mu_{k_1}\left(\eta_{f_{pm}}\right) &= \frac{1}{\eta_{f_p}}\mu_{k2}\left(\eta_{p2}\right) * \eta_{s_p}, \\ \mu_{k_2}\left(\eta_{s_{pm}}\right) &= \mu_{k1}\left(\eta_{p1}\right), \end{aligned} \tag{3.18}$$

such that $k_1 = k2$ and $k_2 = k1$. Here $k1$ and $k2$ refer to the output classes (hard partitions) C_{k1} and C_{k2} that elicited maximal strength responses at the $p1^{th}$ and $p2^{th}$ neurons respectively during calibration. On the other hand, C_{k_1} and C_{k_2} are dependent both on the actual output responses during testing and the membership values evaluated during calibration with respect to classes C_{k1} and C_{k2}. The membership values on the right hand side of eqns. (3.17)-(3.18) are defined as

$$\mu_{k1}\left(\eta_{p1}\right) = \frac{\eta_{(p1)_{k1}}}{\eta_{f_{k1}}} \tag{3.19}$$

from eqn. (3.16), where $\eta_{f_{k1}}$ and $\eta_{(p1)_{k1}}$ are obtained during calibration for class C_{k1}. Hence pattern p may be classified as belonging to class C_{k_1} with membership $\mu_{k_1}\left(\eta_{f_{pm}}\right)$ lying in the interval [0,1], using the first choice and to class C_{k_2} with membership $\mu_{k_2}\left(\eta_{s_{pm}}\right)$ using the second choice. It is to be noted that classes C_{k_1} and C_{k_2} are determined from classes C_{k1} and C_{k2} by eqns. (3.17)-(3.18). A confusion matrix may be generated to evaluate the performance of this fuzzy classifier on the set of test patterns.

It is to be mentioned that if we consider the calibrated membership values instead of the calibrated strength values on the right hand side of eqn. (3.19) for substitution into eqns. (3.17)-(3.18), then we get membership-based recognition instead of the above-mentioned strength-based recognition scheme.

Mean square distance for test set

The mean square distance for test patterns is defined as

$$msd_t = \frac{1}{|testset|} \sum_{p \in testset} \|p - m_{p1}\| * \frac{3n + l}{3n} \tag{3.20}$$

where $|testset|$ corresponds to the number of pattern vectors used during testing, and $\boldsymbol{m}_{p1}$ consists of the first $3n$ components only of the weight vector of the neuron $p1$ generating the highest output response η_{f_p} for test pattern $\boldsymbol{p}$. This is a measure of the amount of mismatch between the two vectors while classifying pattern $\boldsymbol{p}$.

3.5 Implementation and results

The above-mentioned algorithm has been implemented on the set of 871 Indian Telugu vowel sounds depicted in Fig. 2.6. The dimension of the input vector here is 15. The model has been tested for various sized two-dimensional arrays of neurons. During self-organization, different sizes of training sets have been used by randomly choosing *perc* % samples from each vowel class. The remaining (100 − *perc*) % samples from the original data set have been used as the test set in each case. We have selected $s = 0.2$ in eqn. (3.5), $b = 0.02$ in eqn. (3.9), $\alpha = 0.9$ in eqn. (3.11) and $\delta = 0.0001$ in eqn. (3.15) after several experiments.

3.5.1 Output map

After self-organization and calibration the resulting output map is plotted using both hard and fuzzy partitioning. In Figs. 3.4-3.5, part(a) corresponds to the hard partitioning obtained by mapping each neuron to the vowel class to which it is most sensitive. The class number k (1 for ∂, 2 for a, 3 for i, 4 for u, 5 for e, 6 for o) marks the neuron eliciting the maximum response η_{f_k} for that class C_k while the neighboring dot indicates the neuron generating the second highest response. Parts(b)-(d) of the same figures indicate the boundaries for the fuzzy partitioning of the output space by eqn. (3.16) for the three pairs (drawn for the convenience of understanding) of the six classes using $\alpha' = 0.1$. It is to be noted that the topological ordering of the vowel classes in the two-dimensional output space (considering fuzzy partitioning) bears much similarity,

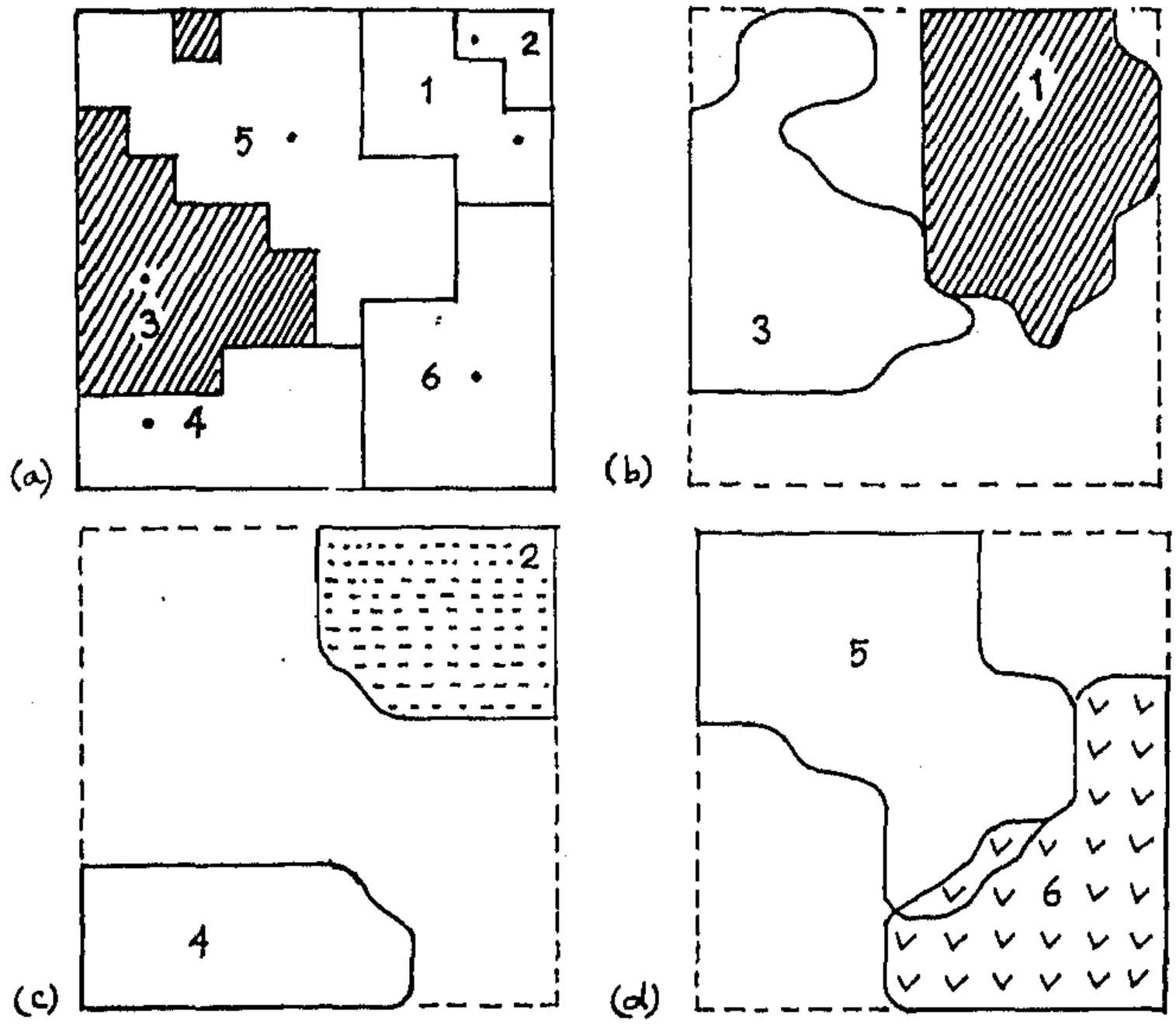

Figure 3.4: Fuzzy Kohonen's model, (a) Hard and (b)-(d) Fuzzy partitioning

including the amount of overlapping, to the original Fig. 2.6 in the two-dimensional feature space. The use of fuzzy partitioning is found to help in faithfully preserving the mapping of fuzzy or overlapping pattern classes.

Fig. 3.4 shows the output map generated for an 10×10 array of neurons with $perc = 15$ and $cdenom = 100$. The hard partitioning illustrates one discontinuous mapping for class 3. However the incorporation of fuzzy partitioning alleviates this problem and we find overlapping between classes 1,2; 1,5; 2,5; 2,6; 3,5; 4,5; 4,6; and 5,6. This compares favourably with the overlapping observed in the feature space of Fig. 2.6. It is to be noted that, unlike in Fig. 2.6, the classes 3 and 4 are seen to be adjacent in part(a) here. This is because there exist no pattern points between these two classes *in the input feature space* and in this sense they may be termed *adjacent*. To alleviate this problem, one may model the region of the feature space with *no patterns* as a separate class. This may result in mapping the actual ordering between the classes, along with their inter-class spaces, on to the neuronal array.

Fig. 3.5 shows the output for the conventional Kohonen's model (using the same parameters as in Fig. 3.4) with $s = 0$ in eqn. (3.5) but also incorporating the fuzzy

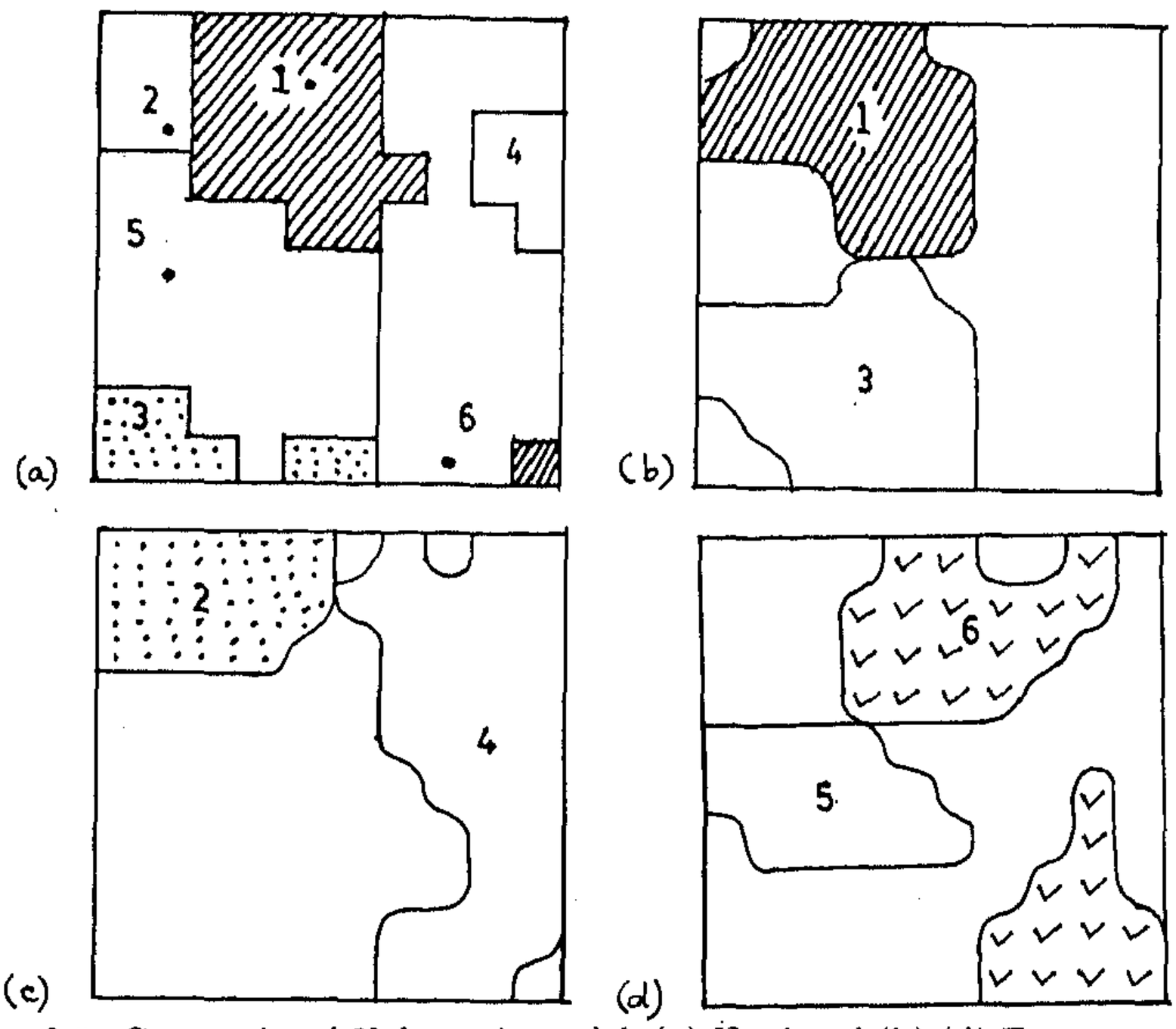

Figure 3.5: Conventional Kohonen's model, (a) Hard and (b)-(d) Fuzzy partitioning

partitioning concept as extension. The input feature information part $\boldsymbol{x}'$ of eqn. (3.4) is in the fuzzy linguistic form of eqn. (2.10) for ease of comparison with the fuzzy Kohonen's model while demonstrating the utility of the inclusion of the contextual class membership part $\boldsymbol{x}''$ in the input vector. Note the discontinuities among the hard partitions for classes 1 and 3. We also observe wrong topological ordering of the vowel classes (as compared to Fig. 2.6). In part(a) the partitions for classes 2,5 and 3,6 are adjacent, unlike that desired, while classes 2,6 and 4,5 are separated. Besides, the neurons eliciting the highest and second highest responses have been observed to lie in the wrong calibrated *hard* partitions for classes 3 and 4. This has an adverse effect on the recognition performance over the test set by eqns. (3.17)-(3.19). The use of fuzzy partitioning introduces discontinuities for class 6 in part(d) while eliminating the problems for classes 1 and 3 in part(b). However classes 1,3 and 2,4 are found to be adjacent in parts (b)-(c), unlike the case in Fig. 2.6. By comparing the output maps for Figs. 3.4 and 3.5, the usefulness of $s > 0$ to incorporate some contextual information while handling ill-defined, fuzzy data becomes evident.

3.5.2 Performance on test set

Finally, a separate set of test patterns has been applied to the model under consideration, using $cdenom = 100$ and $perc = 10$, and its performance evaluated. In Figs. 3.6-3.8, part(a) plots the *percent* correct classification while part(b) shows the variation of the mean square distance msd_t of eqn. (3.20) along the ordinate. In part(a), the class numbers ($k = 1, \ldots, 6$) indicate the class-wise correct classification of the test set. The symbols s and m correspond to the overall correct classification of the entire test set using the strength-based recognition by eqns. (3.17)-(3.18) and the related membership-based recognition schemes respectively.

Fig. 3.6 is drawn to illustrate the effect of varying the size of the network. The 10×10 array is observed to give best recognition rates in part(a). A smaller size of the network is seen to be incapable of handling all the information required while a larger size may result in poor performance over the test set. However the msd_t curve in part(b) demonstrates that the 8×8 array results in a much poorer topological ordering as compared to the other two network sizes while the 12×12 array yields a *slightly lower* value of msd_t as compared to the 10×10 network. In this connection, it may be noted that the recognition scores are dependent on the output membership values generated *both during calibration and testing* by eqns. (3.16)-(3.19) while the msd_t of eqn. (3.20) is dependent *only on the testing phase* values. Therefore *more significance* may be assigned to the recognition rate results.

Fig. 3.7 demonstrates the effect of using the index of disorder D of eqns. (3.12)-(3.15) to control the number of sweeps through the training samples during self organization of 10×10 networks. This is marked as *"usual iterations"* on the abscissa of the figure. In the traditional Kohonen's model, the network goes through a larger number of sweeps. The effect of using 200 iterations without considering the influence of D is also plotted. Our model is found to yield an improved performance (with only 90 iterations) over the more conventional design.

In Fig. 3.8 we illustrate the comparison between (i) the fuzzy Kohonen's net (marked *"usual"* along the abscissa), (ii) the *"hard"* version using a *crisp* linguistic representation for the input vector with $s > 0$ and (iii) the *"original"* Kohonen's model with $s = 0$ in eqn. (3.5) but using fuzzy linguistic representation for the input vector along with the fuzzy partitioning concept as an extension. The different

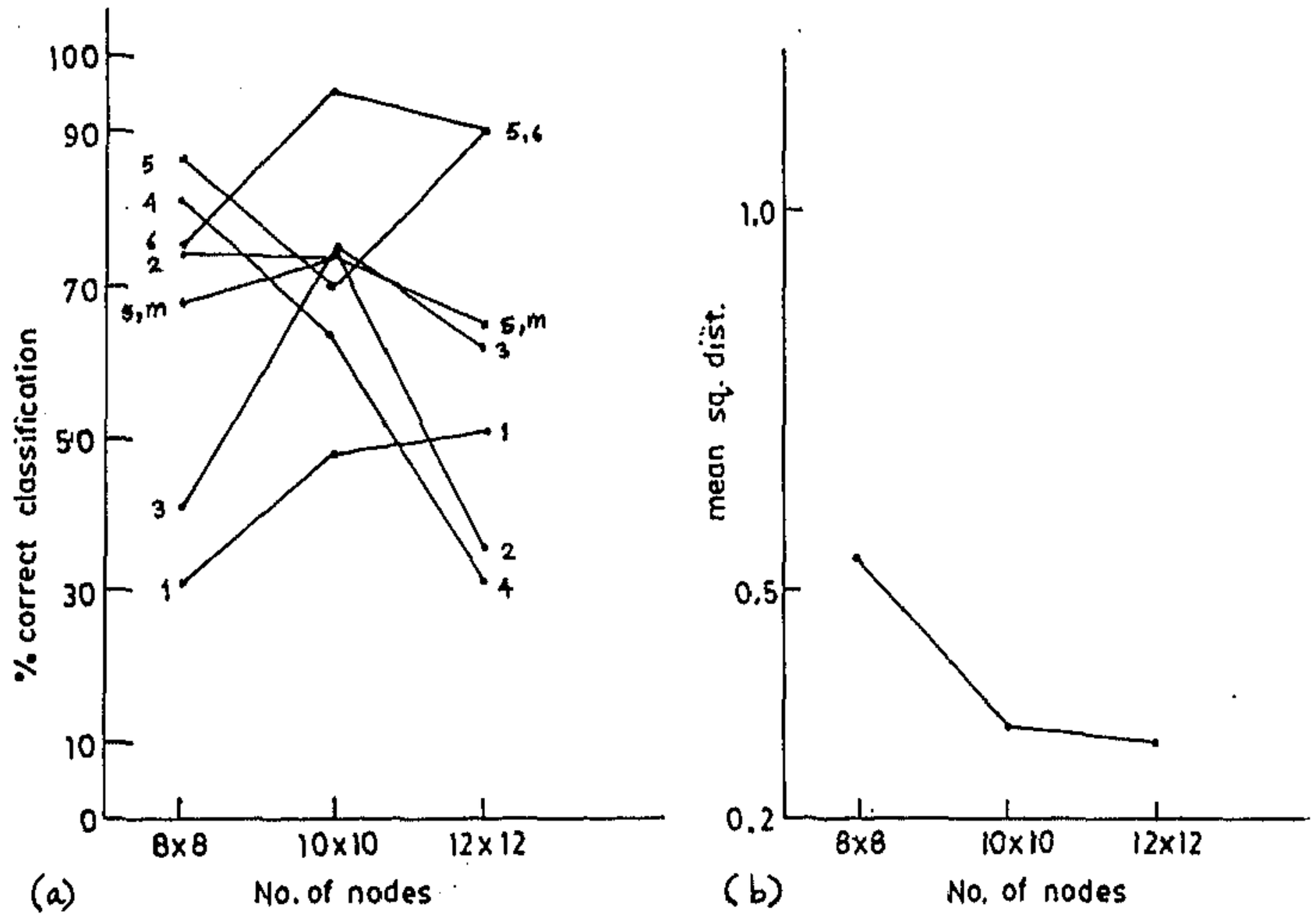

Figure 3.6: Fuzzy Kohonen's net showing (a)Correct classification (%) and (b)Mean square distance

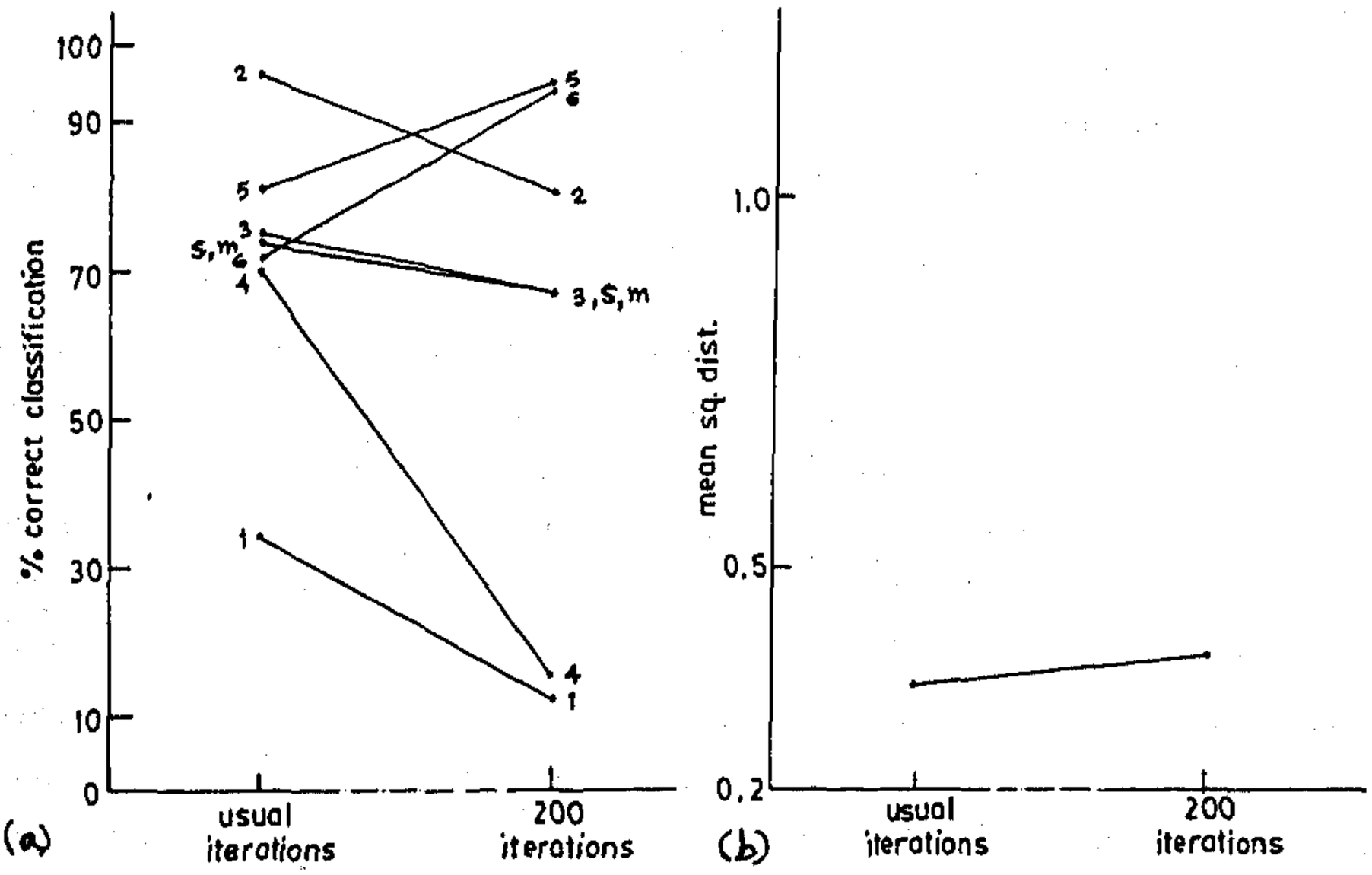

Figure 3.7: Effect of D on the performance, (a)Correct classification and (b)Mean square distance

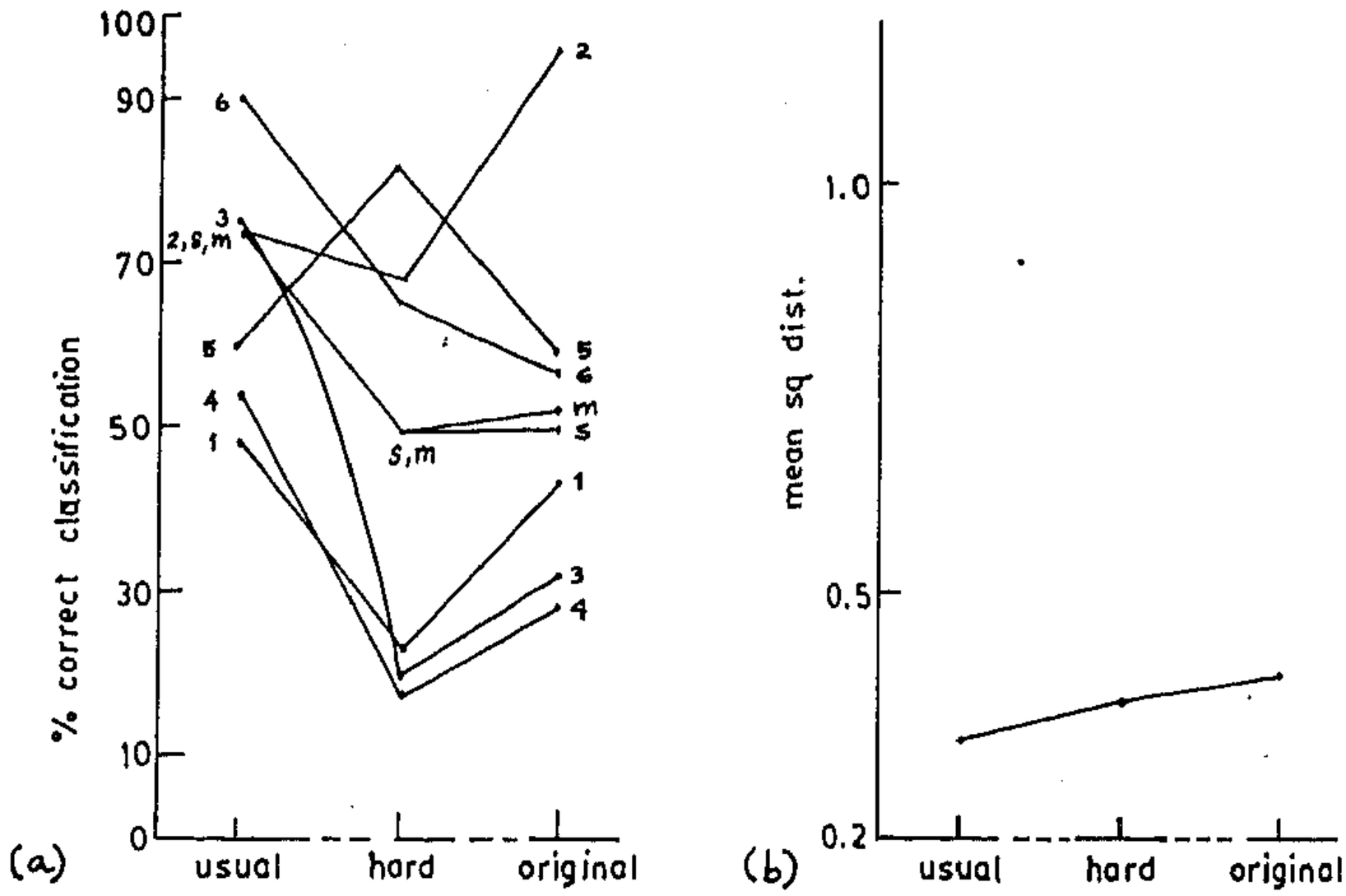

Figure 3.8: Comparison between the fuzzy self organizing network, its *hard* version and the *original* Kohonen's model. (a)Correct classification and (b)Mean square distance

features of these models are listed in Table 3.1. In all three cases, 10×10 network arrays are employed. In the hard model, the input feature information part in the $3n$-dimensional space is assigned *crisp* values such that corresponding to a pattern $\vec{F}_i$, along the j^{th} axis, we clamp the highest of $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ and $\mu_{high(F_{ij})}(\vec{F}_i)$ of eqn. (2.10) to 1 while the remaining two are kept clamped at 0. The gain factor h_{ci} from eqn. (3.11) is not *bell-shaped* and its *hard* version is defined as $h_{ci} = \alpha / \left[1 + (\frac{nt}{cdenom})^2\right]$. The contextual class information, though present, is not in the form of membership values but is expressed in *crisp* terms giving a membership of 1 to only one class. The original model (method (iii)) is used with the $3n$-dimensional fuzzy linguistic representation for the input feature information and the *bell-shaped* gain factor h_{ci} of eqn. (3.11).

Note that the hard model is seen to have the worst recognition rate. The fuzzy Kohonen's net is observed to result in the best overall classification efficiency. Inclusion of fuzzy concepts (as introduced in methods (i) and (iii)) is found to enhance the performance with respect to the *hard* version (method(ii)). On the other hand, the incorporation of class information with $s > 0$ enables the fuzzy Kohonen's net (method(i)) to score over the more conventional *original* version (method(iii)). This underscores the utility of involving fuzzy concepts in conjunction to partial class membership information in our model.

Table 3.1: Different features of the *Fuzzy (usual)*, *hard* version and the *original* Kohonen's network

Model	*Input feature* information	*Class information*		*Gain* factor	*Output* partitioning
		scale factor	membership		
Fuzzy (*usual*)	fuzzy linguistic	$s = 1$ for calibration $0.5 < s < 0$ for self-organization	fuzzy	Bell-shaped	fuzzy
Hard	crisp linguistic	$s = 1$ for calibration $0.5 < s < 0$ for self-organization	crisp	Pulse	fuzzy
Original	fuzzy linguistic	*nil*	*nil*	Bell-shaped	fuzzy

It is observed that the msd_t curve in part(b) exhibits better resultant topological ordering for the *hard* version as compared to the *original* model. This is in contrast to the findings for the recognition rate (%) in part(a) of the figure where it is seen to have poorer performance. We should note that the *hard* version uses partial supervision ($s > 0$) although with *crisp* input, output and partitioning. This contextual class information generates a better ordering of the output space (along with a lower msd_t value) although the recognition rate is poorer due to the *hard* representation used. However the fuzzy Kohonen's net has a superior performance with respect to both the recognition rate and msd_t, as it incorporates both fuzziness and partial supervision.

Table 3.2: Comparative study of the recognition scores (%) of the various models

Class	Bayes' classifier	Standard fuzzy classifier	Fuzzy neural model
∂	44.6	51.4	23.0
a	83.9	81.7	97.5
i	81.9	78.0	74.8
u	88.9	67.6	73.5
e	82.8	77.7	88.7
o	77.7	78.8	92.6
Overall	79.6	73.4	79.6

Table 3.2 compares the recognition score (on test set) of a 10×10 fuzzy Kohonen's net (with $perc = 10$ and $cdenom = 20$) to that of the Bayes' classifier [1, 2] and the standard fully supervised fuzzy approach [65]. We have used the Bayes' classifier

or multivariate normal patterns with the *a priori* probabilities $p_i = \frac{|C_i|}{N}$ where $|C_i|$ ndicates the number of patterns in class C_i and N is the total number of pattern points. The dispersion matrices are different for each pattern class. The overall performance of the fuzzy Kohonen's net is found to be quite satisfactory. It is to be noted that the Bayes' classifier is the best that is theoretically possible and neural nets should not do better. Besides, a good statistical classifier requires a lot of sequential computation and a large number of reference vectors while, on the other hand, a neural network is massively parallel and can generalize well with a smaller set of training patterns. This brings out the utility of our approach.

Table 3.3: Recognition score (%) with $cdenom = 60$ and $perc = 10$

Class	First choice	Second choice	Net score
∂	53.8	7.7	61.5
a	76.5	21.0	97.5
i	79.3	3.9	83.2
u	66.9	13.2	80.1
e	64.7	23.0	87.7
o	90.1	1.9	92.0
Overall	73.5	11.2	84.7

Generally the confusion in recognizing a test pattern, considering the first choice, arises only due to misclassification as one of the neighboring classes constituting a vowel triangle. This is due to the appreciable amount of overlapping present in the data under consideration. The correct classification rate for an 10×10 network considering both the first and second choices by eqns. (3.17) and (3.18) is illustrated in Table 3.3. The confusion matrix for this particular set of parameters, as shown in Table 3.4, also supports this claim.

In Table 3.5 we compare the performance of the 10×10 fuzzy Kohonen's net (with $cdenom = 100$ and $perc = 10$) for various choices of parameters r and f for the gain factor h_{ci} of eqn. (3.11). Model A refers to the case where $f = \frac{1}{4}$ always in eqn. (3.13) for all values of $ncnt$. Although both $|h_{ci}|$ and r decay with time, yet it presents a slight variation of our model D (due to the constant value of f). Network B uses

Table 3.4: Confusion matrix for the network

	∂	a	i	u	e	o
∂	35	16	0	0	13	1
a	18	62	0	0	0	1
i	0	0	123	0	32	0
u	4	0	0	91	3	38
e	57	0	7	1	121	1
o	4	9	0	0	3	146

Table 3.5: Recognition scores (%) for various choices of r and f in h_{ci}

Class	Model			
	A	B	C	D
∂	7.7	0.0	1.5	47.7
a	51.8	98.7	100.0	74.0
i	69.0	89.6	68.3	74.8
u	24.2	80.8	66.9	63.9
e	94.6	73.8	74.8	70.0
o	88.8	63.5	58.0	94.4
Overall	64.6	72.5	65.2	73.5

$0 \leq r \leq 3$ and $f = \frac{1}{4}$ for all values of *ncnt* in eqn. (3.13). Note that here only $|h_{ci}|$ decays with time by eqn. (3.11) while its radius remains constant. In model C the term $(1 - r * f)$ is eliminated from the numerator of eqn. (3.11) in addition to keeping the radius ($0 \leq r \leq 3$) of the gain function constant (as in B). Here the function h_{ci} is no longer *bell-shaped* and only $|h_{ci}|$ decays with time. The significance of the gain factor in model D, where both $|h_{ci}|$ and r decay with time, is obvious from the results.

Table 3.6: Recognition scores (%) of the fuzzy and conventional Kohonen's net for various training set sizes

Model	Conventional Kohonen's net					Fuzzy Kohonen's net				
perc	10	20	30	40	50	10	20	30	40	50
∂	43.0	44.8	39.2	77.2	80.5	47.7	43.1	43.1	61.3	47.2
a	96.3	65.2	90.4	40.7	0.0	74.0	59.7	47.6	0.0	37.7
i	31.6	26.8	53.7	36.5	34.8	74.8	78.9	57.0	78.8	62.8
u	27.9	44.6	0.0	39.5	31.5	63.9	72.7	49.0	39.5	48.6
e	59.3	83.7	76.5	79.2	77.8	70.0	74.1	91.0	88.0	98.0
o	56.8	79.8	57.9	47.2	92.2	94.4	62.5	95.2	95.3	93.3
Overall	50.3	59.8	53.2	53.2	56.5	73.5	68.3	69.4	68.0	71.1

Table 3.6 illustrates a comparison in performance on test set (using first choice) of our model with the conventional Kohonen's network (using fuzzy linguistic feature information $\boldsymbol{x}'$ of eqns. (2.10) and (3.4) only at the input) for various sizes of training data set *perc* (using 10×10 array with *cdenom* = 100). This is to demonstrate the necessity of incorporating the contextual class membership information $\boldsymbol{x}''$ at the input of the said network for modeling fuzzy data. It is observed that our model has a superior recognition score compared to its more conventional counterpart. Note that an increase in the size of the training set (abundance of attribute data) for the vectors under analysis has no appreciable impact on the performance of the conventional model. On the other hand, the incorporation of the contextual class membership information, with $s > 0$, seems to boost the efficiency of the fuzzy model (with identical parameter values) in classifying the same data. This further demonstrates the utility of using class membership information in the input vector.

Table 3.7: Recognition scores (%) of the fuzzy and conventional Kohonen's net for various number of input attributes

Model	Conventional Kohonen's net			Fuzzy Kohonen's net
Input Vector Components	fuzzy linguistic features with $s = 0$			fuzzy linguistic features with $0.5 > s > 0$
Dimension	9	18	27	15
∂	43.0	52.3	100.0	47.7
a	96.3	0.0	0.0	74.0
i	31.6	69.0	51.6	74.8
u	27.9	71.3	0.0	63.9
e	59.3	75.9	0.0	70.0
o	56.8	97.5	0.0	94.4
Overall	50.3	68.4	18.4	73.5

In Table 3.7 we demonstrate the effect (on the recognition efficiency) of using various number of input attributes (dimensions) on the standard Kohonen's net (with fuzzy linguistic input feature information as an extension) and compare with our fuzzy version using contextual class membership information at the input. Network arrays of size 10×10 with $cdenom = 100$ and $perc = 10$ are used. A very high input feature space dimensionality with too many attributes is found to hinder the efficiency of the conventional network. The primary linguistic properties *low, medium* and *high* corresponding to each input feature results in 9 attributes for the given data set. Incorporation of the hedge *very* (for each of the three linguistic terms) yields 18 attributes while further addition of the hedge *more or less* leads to 27 attributes for the conventional model. The latter version is seen to be incapable of classifying the given pattern set. Note that the incorporation of the contextual class membership information (with $s > 0$) in the fuzzy Kohonen's net results in the best performance, both overall and class-wise.

3.6 Conclusions and discussion

A neural network model based on self organization and capable of performing fuzzy classification has been described. The algorithm passes through two stages, *viz.* self organization and testing. The model has the flexibility of accepting linguistic input and can provide output decision in terms of membership values. The input vector incorporates partial class membership information during self organization. An index of disorder is used to determine a measure of the ordering of the output space and control the number of sweeps required in the process. Unlike Kohonen's conventional model, this network is capable of producing fuzzy partitioning of the output space and thereby provides a more faithful representation for ill-defined or fuzzy data with overlapping classes. Incorporation of fuzziness in the input and output of the said model is seen to result in better performance as compared to the original Kohonen's model and the hard version, when the vowel data is considered as input. The performance of the model for various network array sizes, training sets and gain factors is also investigated.

It has been observed that a *critical* size of the network is required for satisfactory performance. The fact that a larger size resulted in poorer recognition of the test patterns is favourable in the sense that more neurons lead to an increased cost.

Note that the fuzzy Kohonen's net described in this chapter employs partial supervision incorporating contextual class information in the input vector. Unlike the conventional Kohonen's model, which is basically used for clustering, the network here functioned as a classifier. In other words, the present investigation also provides a way how the Kohonen's net, which is usually used for clustering (unsupervised classification), can also act as a classifier (in partially supervised mode). The relevance of the partial supervision involved has already been discussed in the context of the vowel recognition problem in the Introduction of this chapter.

Since the fuzzy Kohonen's net described here acts as a partially supervised classifier, it provides relatively poor performance as compared to the fully supervised fuzzy MLP of Chapter 2. This is evident from the results of vowel recognition (Tables 2.2 and 3.6). In the following chapter we provide a comparative study of the performance of the fuzzy MLP and the fuzzy Kohonen's net for classifying certain linearly nonseparable nonconvex pattern sets. Their performance has also been compared with related techniques.

Chapter 4

Applications of Fuzzy MLP and Fuzzy Kohonen's Model to Linearly Nonseparable Nonconvex Pattern Classes

4.1 Introduction

The fuzzy multilayer perceptron (MLP) and the fuzzy Kohonen's net, described in Chapters 2 and 3, are capable of accepting input features in quantitative and/or linguistic form and providing fuzzy output decision in terms of class membership values. One can also obtain hard decisions (as a special case). The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties *low, medium* and *high* corresponding to each input feature. Thereby an n-dimensional feature space is decomposed into 3^n overlapping sub-regions corresponding to the three primary properties [84]. This enables the models to utilise more local information of the feature space and is found to be suitable in handling overlapping regions.

In this chapter we concentrate on demonstrating the classification ability of these models for concave and disconnected (non overlapping) pattern sets [275]. Here we have considered crisp output decision. Three sets of such linearly nonseparable pattern classes (artificially generated) are depicted in Figs. 4.1-4.3. There are two pattern classes (1 and 2) in each case. The region of *no pattern points* is modeled as the class *none (no class)*. In Figs. 4.1 and 4.3, class 2 consists of two disjoint regions and its *a priori* probability is much lower as compared to that of the other classes. The effect of fuzzification at the input has been investigated for both the models. Their performance is compared with those of the nonfuzzy versions of the two neural models, the layered neural algorithms of Rumelhart and McClelland [39], McClelland [160], Franzini [160], Chan and Fallside [161], Tollenaere [163], Silverman [159] and networks using second-order weight correction [39], and also the conventional k-nearest neighbor (k-NN) classifier. (Note that the k-NN classifier is capable of generating piecewise linear boundaries and is suitable for classifying the pattern sets in Figs. 4.1-4.3). The contribution of the *a priori* probabilities of the pattern classes in the error derivative of the backpropagation procedure for weight updating has also been studied.

Before providing the experimental results we describe, in brief, the salient features of various neural algorithms which are considered here for comparison.

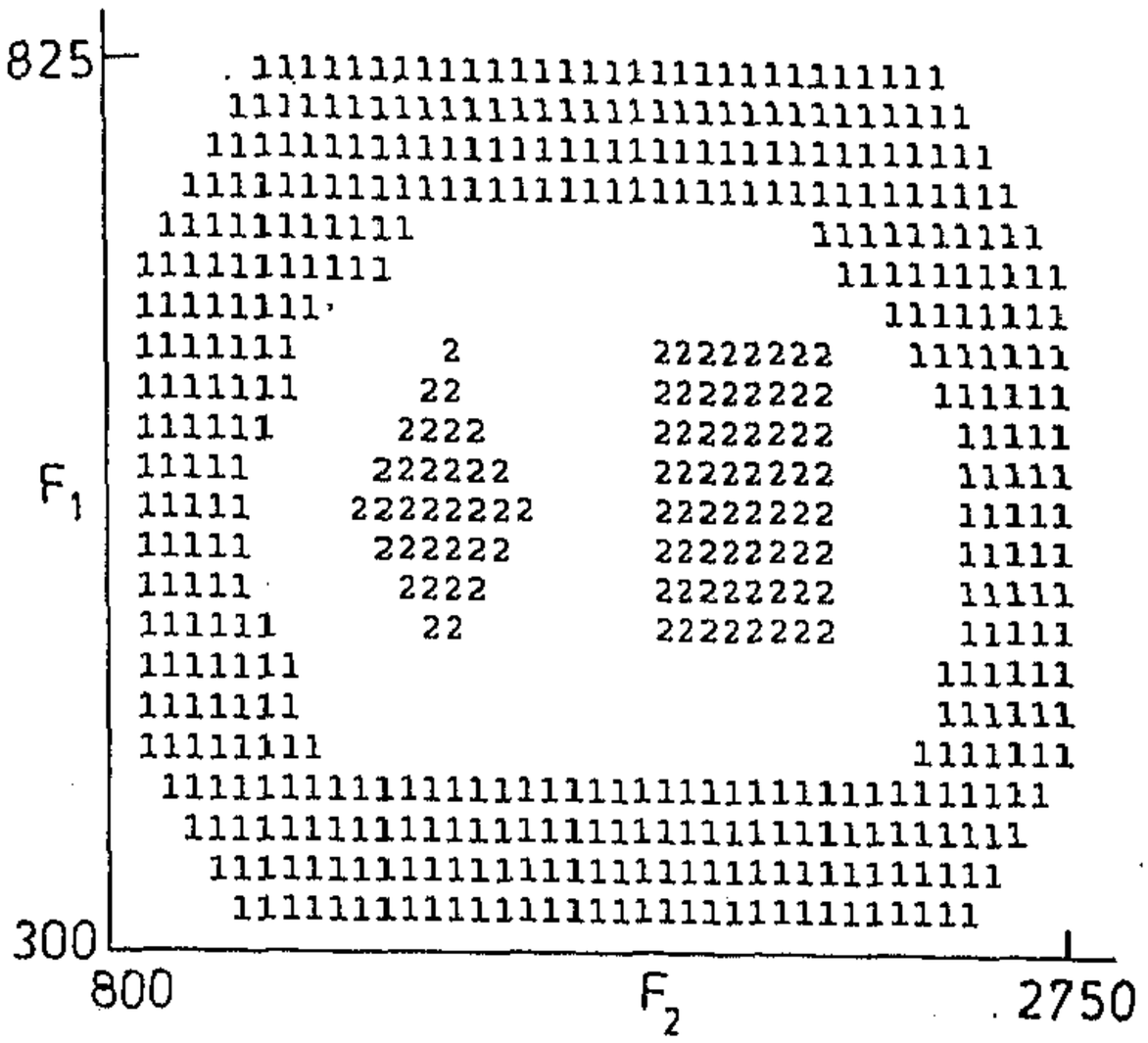

Figure 4.1: Pattern Set A in the $F_1 - F_2$ plane

4.2 Salient features of neural algorithms compared

The various algorithms considered here for comparison are based on the MLP and mainly implement its variations that use different techniques for adapting the learning rate ϵ of eqn. (2.5). The key features of these algorithms, indicating their difference with the conventional MLP, are described below.

4.2.1 McClelland's new error measure

The error measure defined by McClelland [160] is slightly different from that used in the conventional MLP. According to this, the total error of eqn. (2.4) is expressed as

$$E(w) = -\sum_{j,c} \ln \left[1 - (y_j^H(w) - d_{j,c})^2\right] \tag{4.1}$$

such that

$$\frac{\partial E}{\partial y_j} = \frac{1}{1 + (d_j - y_j^H)} - \frac{1}{1 - (d_j - y_j^H)} \tag{4.2}$$

for $h = H$ in eqn. (2.6). This error derivative is expected to speed up the movement of weights which had previously moved slowly because of small sigmoid derivatives. Hence in case of output units whose output is at the wrong end of the sigmoid (and

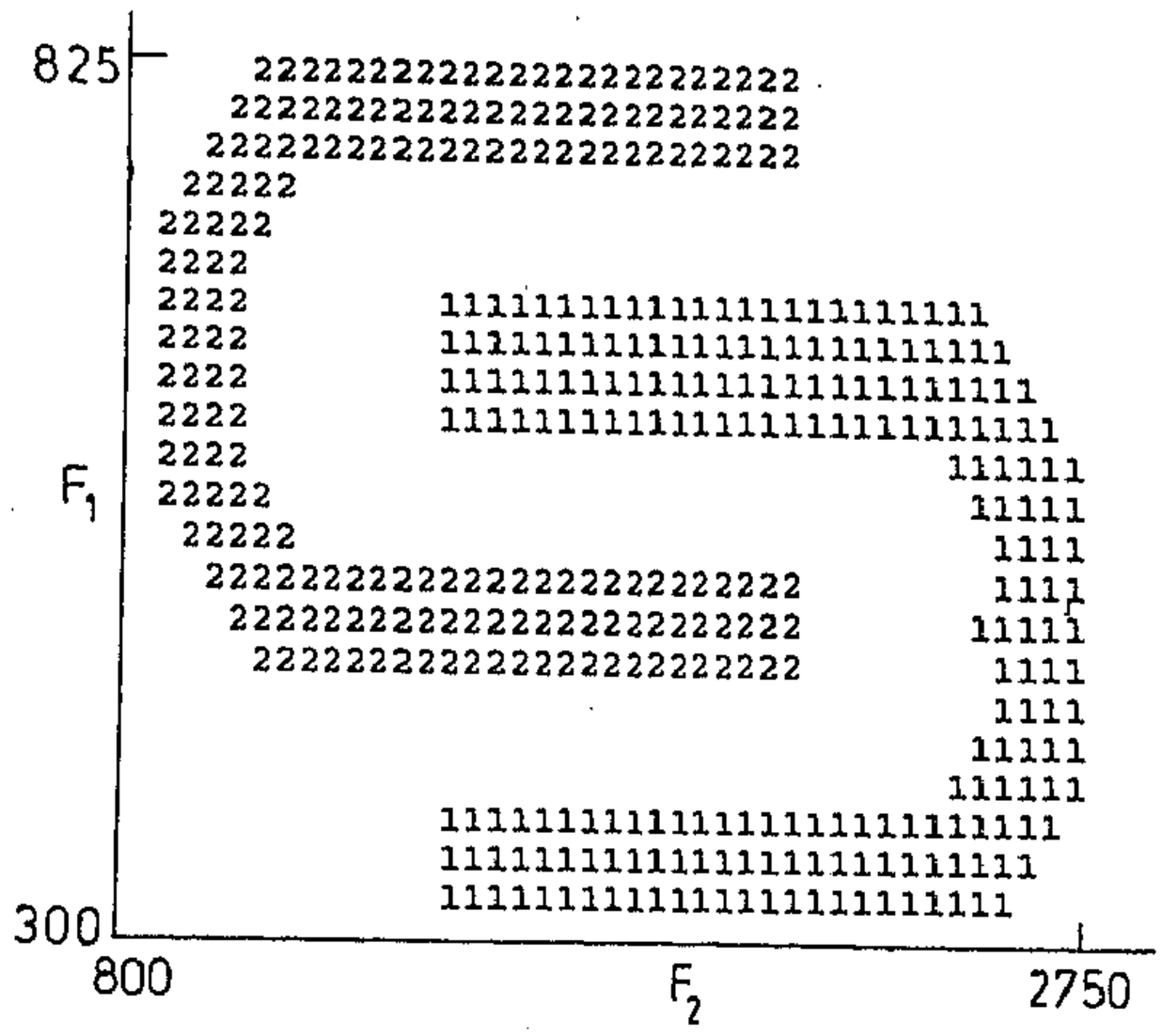

Figure 4.2: Pattern Set B in the $F_1 - F_2$ plane

close to zero or one), the weight change increases and thereby the learning time is reduced.

4.2.2 Learning rate adapted by angles

Chan and Fallside reported in [161] that useful information about the shape of the energy contour can be learned from the directions of the local gradient vector $\nabla \boldsymbol{E}(t)$, and the weight updates $\Delta \boldsymbol{w}(t-1)$ and $\Delta \boldsymbol{w}(t)$ using the vector version of eqn. (2.5). The angle $\theta(t)$, giving an indication of the nature of the energy surface during training, is defined as

$$\cos\theta(t) = -\frac{\nabla \boldsymbol{E}(t).\Delta \boldsymbol{w}(t-1)}{\|\nabla \boldsymbol{E}(t)\| \, \|\Delta \boldsymbol{w}(t-1)\|} \tag{4.3}$$

The learning rate is adapted as

$$\epsilon(t) = \epsilon(t-1)\left(1 + \frac{1}{2}\cos\theta(t)\right) \tag{4.4}$$

such that $\epsilon(t)$ decreases near the *ravine* walls when $\frac{\pi}{2} \leq \theta(t) \leq \frac{3\pi}{2}$ and increases at the *plateaus* when $\theta(t) \rightarrow 0$ or 2π. The damping coefficient is modified as

$$\alpha(t) = \lambda(t)\epsilon(t)$$

where

$$\lambda(t) = \lambda(0)\frac{\|\nabla \boldsymbol{E}(t)\|}{\|\Delta \boldsymbol{w}(t-1)\|}$$

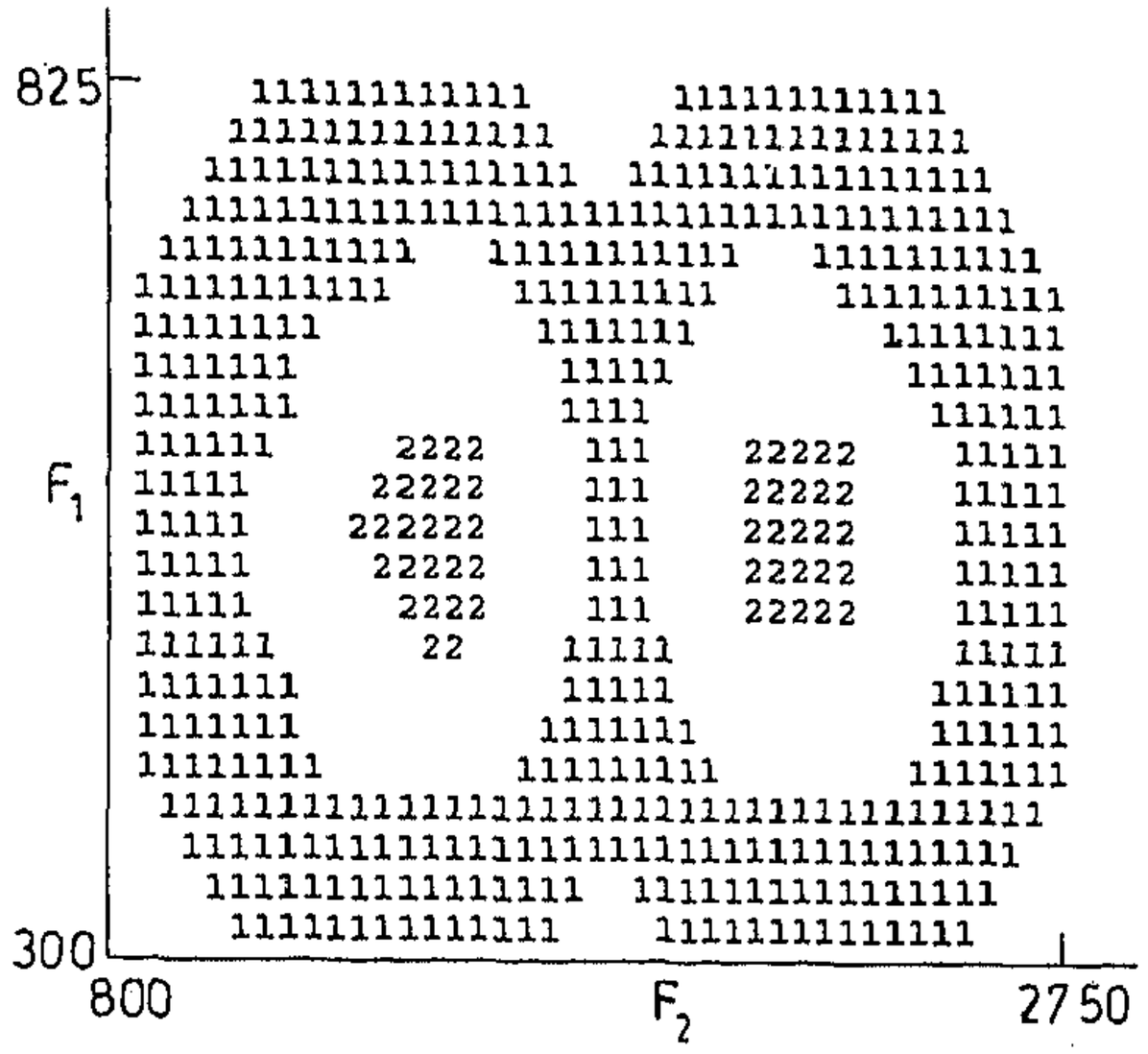

Figure 4.3: Pattern Set C in the $F_1 - F_2$ plane

and $0 \leq \lambda(0) \leq 1$. This scheme is supposed to reduce oscillations at the walls of the ravine.

4.2.3 Adaptive acceleration strategy SSAB

This is a modification of the strategy reported by Jacobs [162] for speeding up the backpropagation algorithm. In this approach by Tollenaere [163], (i) every weight w_{ij} has its own individual (adaptive) step size ϵ_{ij}; (ii) each step size ϵ_{ij} is allowed to vary over time; (iii) for each w_{ij}, as long as the w_{ij} derivative does not change sign the corresponding ϵ_{ij} is increased; and, (iv) when a change in the sign of the w_{ij} derivative is detected : (a) the previous weight update is undone and then ignored in the momentum term of the following step, and (b) the corresponding ϵ_{ij} is decreased.

Therefore as long as the weight derivative keeps changing sign, the step size is decreased until a step can be done without causing the weight derivative to change sign. This method requires a number of local computations and is supposed to be easier to implement on parallel architecture computers. However it also involves an increase in the total computational overhead in its attempt at increasing the speed of convergence.

4.2.4 Second order weight correction for sigma-pi units

Instead of the first-order weight correction of the conventional MLP, Rumelhart and McClelland also considered two element conjuncts in [39]. Here the output of the j^{th} neuron in layer h, from eqns. (2.1)-(2.2), is given as

$$y_j^h = f_j\left(\sum_{i,k} w_{jki}^{h-1} y_k^{h-1} y_i^{h-1}\right) \tag{4.5}$$

where $f_j(.)$ is the sigmoidal function. The error derivative corresponding to the j^{th} neuron in layer h, from eqn. (2.6), becomes

$$\frac{\partial E}{\partial w_{jki}} = \frac{\partial E}{\partial x_j} y_i^{h-1} y_k^{h-1} \tag{4.6}$$

such that

$$\frac{\partial E}{\partial y_j} = \sum_{j,k} \frac{\partial E}{\partial x_k} w_{kij}^h y_j^{h+1} \quad \text{for } 0 \leq h < H$$

Such nets are expected to result in better performance due to the higher order connection weight interactions. However it should be noted that each sweep through the training set involves a much larger number of weight updates and leads to a resultant increase in the computational overhead.

4.2.5 Learning rate adapted by error

The total error E of eqn. (2.4) is used by Silverman [159] to determine the learning rate ϵ of the backpropagation algorithm. In this algorithm when $E(t) < E(t-1)$, ϵ is increased additively. On the other hand, when $E(t) > E(t-1)$, ϵ is decreased multiplicatively. This scheme ensures that the decrease is faster than the increase.

There is a potential pitfall in such methods if in some dimension the energy landscape is such that the gradient never changes [163]. In such cases the learning rate may keep growing and the weights may become infinitely large.

4.2.6 Heuristic scaling of learning rate

Franzini [160] reported a technique that aims to maintain a maximum value of learning rate ϵ (of the backpropagation algorithm) such that the direction of weight change

remains nearly constant. The angle between the error derivative vector component $d_{ij} = \frac{\partial E}{\partial w_{ij}}$ at cycles t and $t-1$ is defined as

$$\cos\theta = \frac{\sum_{i,j} d_{ij}(t-1)d_{ij}(t)}{\sqrt{\sum_{i,j}[d_{ij}(t-1)]^2 \sum_{i,j}[d_{ij}(t)]^2}} \tag{4.7}$$

The epsilon-scaling rule is given as

$$\epsilon(t) = \begin{cases} \epsilon(t-1)\beta^{+}\cos\theta & \text{if } \cos\theta > 0 \\ \epsilon(t-1)\beta^{-} & \text{otherwise} \end{cases} \tag{4.8}$$

with $\beta^{+} \simeq 1.005$ and $\beta^{-} \simeq 0.8$. This rule is supposed to significantly reduce the learning time and avoid local minima (which fixed higher values of ϵ are likely to reach).

4.2.7 Fixed learning rate

The conventional MLP uses a fixed learning rate ϵ [39]. However it has been pointed out in [163] that there is an *optimal step size region* (*osr*) with the interval $[\epsilon_{opt} - \delta_l, \epsilon_{opt} + \delta_r]$ for every problem. For all ϵ lying in this region, the learning converges reasonably fast and remains stable. But one does not know *a priori* where the *osr* is located for a particular problem. The width of the *osr* scales with the absolute value of ϵ_{opt} [163]. The network size and training set seem to influence the *osr*.

To overcome these problems, various techniques are currently being used for heuristically adapting the learning rate ϵ. A few of these schemes are being discussed in this section. We have also implemented the MLP using different constant values of learning rate ϵ.

4.3 Implementation procedure, results and comparative study

The afore-mentioned neural algorithms along with the conventional as well as the fuzzy MLP and fuzzy Kohonen's net have been used to classify three sets (A, B, C) of artificially generated linearly nonseparable pattern classes (Figs. 4.1-4.3) involving nonconvex decision regions. Each of these pattern sets consists of 880 sample points.

The input vector is 6-dimensional for MLP-based models, and 9-dimensional for the Kohonen's net. Various training set sizes are used by randomly choosing $perc\%$ samples from each pattern class. The remaining $(100 - perc)\%$ samples from the original data set are used as the test set in each case.

The effect of fuzzification at the input is demonstrated for the fuzzy MLP and the fuzzy Kohonen's net, by varying the radius of the π-function corresponding to the linguistic set *medium* (eqn. (2.12)). In the cases of *Pattern Sets A* and C, class 2 consists of two disjoint regions with low *a priori* probability. The contribution of the *a priori* probability in the error derivatives of the backpropagation procedure for weight updating is also investigated for these two cases.

The performance of the fuzzy MLP has been compared with those of the conventional MLP and other seven fully supervised neural algorithms of Sections 4.2.1-4.2.7. As the self-organizing fuzzy Kohonen's model acts as a partially supervised classifier, its performance has been compared only with its conventional version (used as a classifier). An investigation on the performance of the k-nearest neighbors algorithm (having the capability of generating piecewise linear boundaries for these three pattern sets) has also been provided.

4.3.1 Using the fuzzy MLP

Tables 4.1-4.3 demonstrate the performance of the fuzzy MLP for three and more layers (having m nodes in each hidden layer) trained in the *batch mode* using $perc$ = 10% and 50% of the samples from the three pattern sets (A, B, C). The *perfect match* p, *best match* b and *mean square error* mse correspond to the training set while the individual classwise scores for classes 1, 2 and *none*, the *overall score* t and mse_t refer to the test set.

4.3.2 Using other neural algorithms

A comparison is provided with the several layered neural net models (variations of MLP) described in Sections 4.2.1-4.2.7, using the same number of hidden nodes as well as the *same set* of initial random weights as in the corresponding fuzzy MLP *(Model O)*. For the sake of convenience, we denote McClelland's error measure of Section 4.2.1

Table 4.1: Performance of fuzzy MLP for Pattern Set A

Layers		3		4		5	
perc		10	50	10	50	10	50
Nodes m		17	17	19	19	10	10
best b (%)		100.	98.7	100.	98.	97.7	95.9
perfect p (%)		34.5	3.4	81.6	53.6	77.	24.2
mse		.006	.018	.002	.016	.018	.022
T	mse_t	.076	.044	.067	.028	.064	.034
e	1 (%)	89.3	95.2	93.2	95.6	91.5	98.6
s	2 (%)	75.	93.9	71.6	91.8	62.5	93.8
t	*none* (%)	84.5	86.4	82.4	90.7	86.2	82.
	overall t (%)	86.0	91.8	86.9	93.4	86.4	92.

Table 4.2: Performance of fuzzy MLP for Pattern Set B

Layers		3		4		5	
perc		10	50	10	50	10	50
Nodes m		11	11	14	14	10	10
best b (%)		100.	86.6	100.	99.3	98.9	94.6
perfect p (%)		62.1	12.1	65.5	69.1	70.1	44.
mse		.007	.086	.004	.008	.012	.025
T	mse_t	.088	.09	.09	.041	.097	.08
e	1 (%)	78.6	93.7	88.5	96.4	83.1	91.9
s	2 (%)	84.	68.	77.7	91.7	74.8	64.9
t	*none* (%)	84.9	84.9	83.9	88.3	86.1	84.
	overall t (%)	83.1	83.4	83.7	91.1	82.8	81.8

Table 4.3: Performance of fuzzy MLP for Pattern Set C

Layers		3		4		5	
perc		10	50	10	50	10	50
Nodes m		13	13	12	12	10	10
best b (%)		100.	67.9	100.	97.3	95.4	90.9
perfect p (%)		32.2	0.	73.6	37.6	67.8	41.5
mse		.008	.159	.003	.024	.025	.053
T	mse_t	.143	.168	.162	.042	.121	.067
e	1 (%)	83.9	100.	79.7	95.7	80.3	98.4
s	2 (%)	84.8	0.	28.2	84.6	28.2	76.9
t	*none* (%)	59.5	0.	62.7	87.1	81.7	69.6
	overall t (%)	75.4	58.9	70.7	92.	77.8	87.

as *Model M*, Chan and Fallside's learning rate adapted by angles of Section 4.2.2 as *Model C*, Tollenaere's adaptive acceleration strategy SSAB of Section 4.2.3 as *Model A*, Rumelhart and McClelland's second-order weight correction for sigma-pi units of Section 4.2.4 as *Model S*, Silverman's learning rate adapted by error of Section 4.2.5 as *Model P*, Franzini's heuristic scaling of learning rate of Section 4.2.6 as *Model F*, and the conventional fixed learning rate of Section 4.2.7 as *Model R*.

Table 4.4: Comparative performance of the layered neural models on Pattern Set A

Model		O	R with $\epsilon =$					P	C	M	F	A	S	O'
			2.0	1.0	0.5	0.3	0.1							
best b (%)		100.	70.2	100.	98.9	96.6	78.2	100.	100.	100.	50.6	77.	93.1	73.6
perfect p (%)		34.5	0.	9.2	2.3	1.2	10.4	4.6	15.	5.8	0.	11.5	11.5	3.5
mse		.006	.064	.013	.015	.023	.081	.023	.018	.008	.162	.109	.06	.096
T	mse_t	.076	.13	.084	.087	.087	.132	.09	.088	.119	.161	.169	.12	.119
e	1 (%)	89.3	94.7	86.4	89.1	90.3	84.3	95.9	88.6	86.2	99.	53.8	87.2	97.8
s	2 (%)	75.	4.5	76.1	81.8	45.4	31.8	76.1	64.7	50.	7.9	21.6	82.9	0.0
t	*none* (%)	84.5	78.7	81.1	77.3	84.2	69.4	70.1	81.4	76.3	46.	79.7	64.2	72.1
	net t (%)	86.	78.8	83.3	84.	83.1	73.	84.2	83.3	78.5	69.5	59.7	78.3	77.5

It is to be noted that the above-mentioned models have been modified to incorporate fuzzy linguistic values in the $3n$-dimensional input space of eqn. (2.10) to facilitate a

more authentic comparison with the fuzzy MLP model O. The objective is mainly to demonstrate the utility of the heuristic adaptation of the learning rate ϵ (eqn. (2.22)). However, model M (using a different error measure) and model S (using a different network architecture) are compared incorporating the same scheme as used in model O for adapting ϵ. The performance of the conventional MLP (model O'), using the same learning rate variation scheme of model O, has also been studied.

Table 4.5: Comparative performance of the layered neural models on Pattern Set B

Model		O	P	C	M	F	A	S	O'
best b (%)		100.	97.7	94.3	100.	61.	60.9	100.	87.4
perfect p (%)		62.1	35.7	18.4	28.8	0.	2.3	47.2	1.2
mse		.007	.018	.039	.007	.133	.113	.007	.078
T	mse_t	.088	.105	.121	.099	.174	.142	.076	.152
e	1 (%)	78.6	83.1	85.	83.6	55.7	65.6	82.1	47.7
s	2 (%)	84.	78.8	64.5	74.3	36.	32.	89.7	72.0
t	*none* (%)	84.9	81.	79.3	84.4	78.1	88.9	84.6	82.0
	overall t (%)	83.1	81.1	77.5	81.9	63.1	70.5	85.1	71.1

Tables 4.4-4.6 compare the performance of these models on the three pattern sets (A,B,C) with m nodes in the single hidden layer and 10% training samples from each class. The choice of m is made after several runs with different numbers of hidden nodes and best results are obtained with m = 17, 11, 13 for pattern sets A, B, C respectively. In all three cases the nonfuzzy model O' gives poorer results. This is particularly evident on observing the very poor recognition score for class 2 in the cases of pattern sets A and C. It may be noted that the use of linguistic inputs causes the n-dimensional feature space of model O' to be decomposed to 3^n overlapping subregions in the case of model O, thereby enabling the fuzzy model to utilise more local information about the feature space and making it better equipped in handling the given pattern sets.

All models (except M and S that terminate by the criterion of eqn. (2.22)) have been run for the same number of sweeps as required by the corresponding model O before convergence. This allows us to assess the status of the other models at the

Table 4.6: Comparative performance of the layered neural models on Pattern Set C

Model		O	P	C	M	F	A	S	O'
best b (%)		100.	93.1	93.1	100.	70.1	82.8	100.	77.1
perfect p (%)		32.2	19.6	0.	33.4	10.4	9.2	10.4	8.1
mse		.008	.049	.041	.009	.116	.096	.011	.104
T	mse_t	.143	.139	.14	.149	.162	.163	.131	.16
e	1 (%)	83.9	85.4	84.8	79.7	85.	92.9	90.1	87.1
s	2 (%)	84.8	0.	0.	91.3	0.	0.	67.4	0.
t	*none* (%)	59.5	70.9	69.9	61.6	50.1	27.9	45.8	51.6
	overall t (%)	75.4	75.4	74.6	74.	67.8	64.7	73.2	69.6

time when model O has converged after having started from the same set of initial connection weights and then having been trained with the same set of pattern points. In this connection, it is worth mentioning that the recognition score depends on the terminating point of the algorithm as well as on the heuristic used for learning rate variation. However, when a neural algorithm converges to a reasonably good solution over a smaller number of sweeps through the training set, this is indicative of its efficiency. Hence the comparison provided with the other algorithms in Tables 4.4-4.6 (using the same terminating point and uniform input representation), with different heuristics for adapting the learning rates, helps to bring out the utility of the scheme for model O (as explained in Chapter 2). Further, it is to be noted that model S involves a larger number of connection weights (second order connections) as compared to all other models used here. Therefore, a certain number of sweeps in case of model S entails a much larger number of weight updates (increased computational overhead) as compared to the other models. Note that investigations regarding model R have been reported in detail with respect to *Pattern Set A* only (in Table 4.4 and Fig. 4.5) as this uses a constant learning rate that is very much problem dependent.

On the whole the performance of model O over *Pattern Set C* has been observed to be poorer than that of its performance over the other two pattern sets. This is perhaps an indicator to the *more difficult* nature of the problem in case of *Pattern Set C*. Comparing the results from Tables 4.4-4.6 it is seen that model O gives consistently

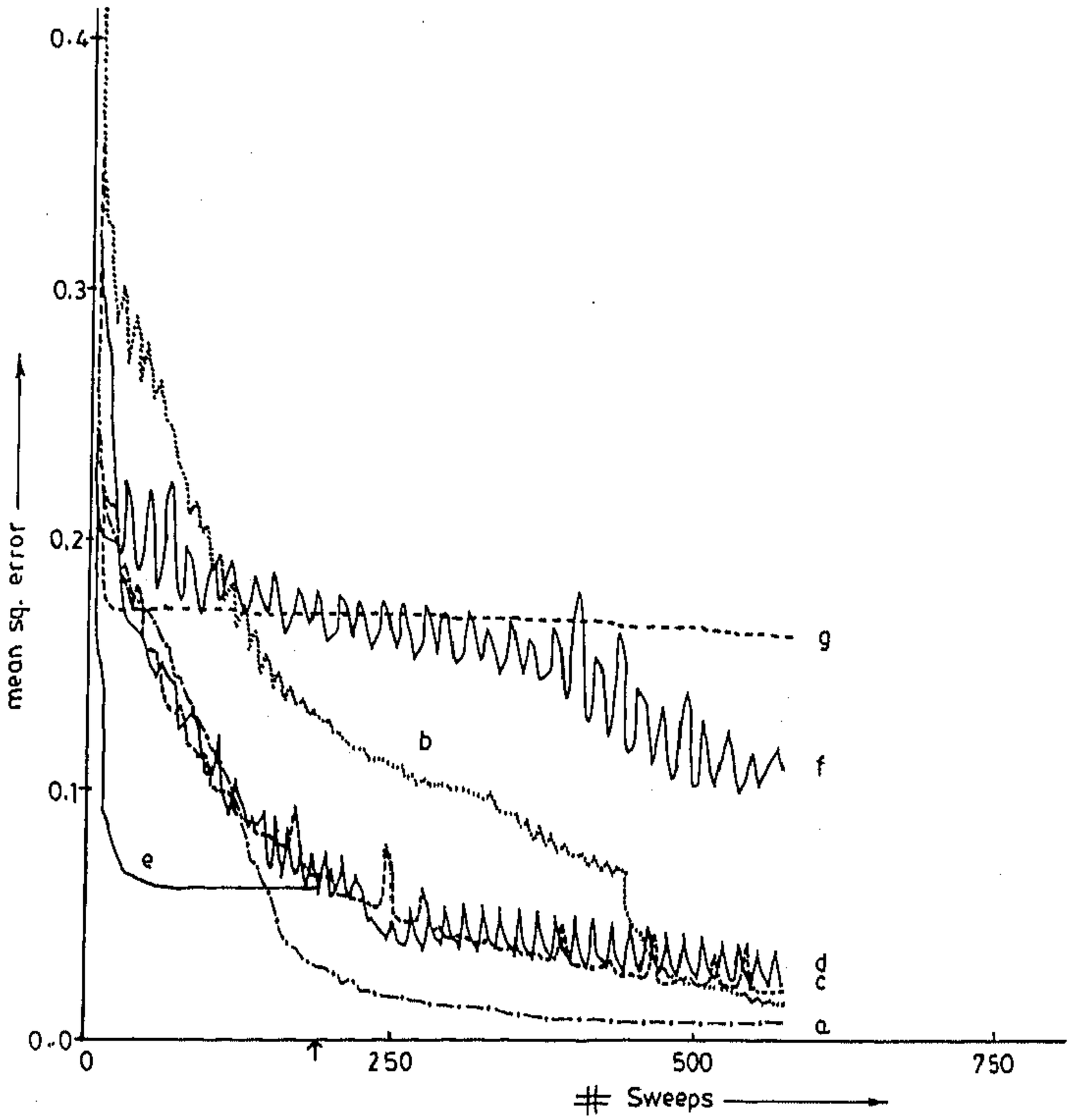

Figure 4.4: Variation of mean square error with number of sweeps for layered neural net models (M, C, P, S, A, F and O) over Pattern Set A

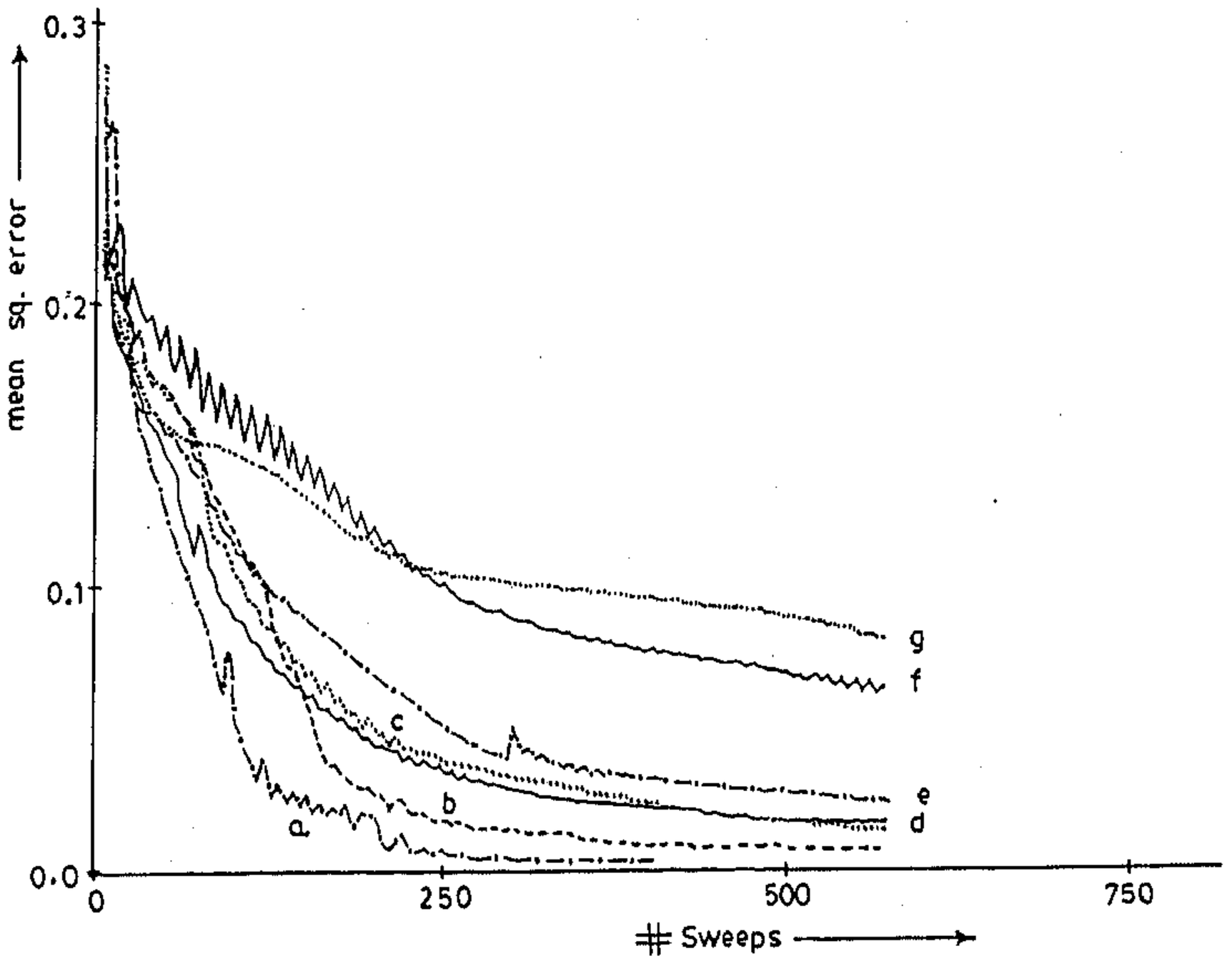

Figure 4.5: Variation of mean square error with number of sweeps over Pattern Set A using models R and O

good performances over all three of the given pattern sets. On the other hand, the behaviour of the other models is seen to vary over the same three cases.

Figs. 4.4-4.7 illustrate the variation of the mean square error of the various layered neural net models during training with the number of sweeps, using $perc = 10$, over the three pattern sets (A, B, C). All models (except S) are assessed by their status upto the sweep number when the corresponding three-layered model O has converged. Fig. 4.4 shows the results of using variations of three-layered nets with $m = 17$ over *Pattern Set A*. We observe that in the case of model S (e) the mse decreases very rapidly in the initial stages but stabilises to 0.06 after around 60 sweeps. On the other hand models O (a), C (c) and P (d) exhibit similar behaviour in the early stages with the mse of model O falling more rapidly after about 125 sweeps to an ultimate low value of 0.006. Model S terminates at around 190 sweeps, as indicated by the arrow along the abscissa. Note that this corresponds to a larger number of weight updates as

compared to the corresponding stage in the other models. The *mse* for model M (b) decreases rapidly from an initial high value to a satisfactory final value that was lower than those obtained by models C and P, which incidentally behaves better initially. Oscillations are evident for models P and A (f) in the process of convergence. Models A (f) and F (g) are seen to behave rather poorly.

Fig. 4.5 demonstrates the results of using three-layered models R with $\epsilon = 2$ (f), 1 (c), 0.5 (d), 0.3 (e), 0.1 (g) and model O (b) with $m = 17$ nodes on *Pattern Set A*. It is seen that model O (b) had the best overall behaviour, although (c), (d), (e) also exhibit satisfactory performance. However, a four-layered version of model O (a) with $m = 19$ converges to a lower final value of *mse* over a fewer number of sweeps through the training set. But simultaneously this also entails a larger number of weight updates. The results of Figs. 4.4-4.5 may be compared with those of Table 4.4 for a better understanding of the behaviour of the various neural models. Note that as reported in [163], the choice of the appropriate value of ϵ is very much problem dependent. Here lies the utility of choosing adaptive algorithms for varying the learning rate.

In Fig. 4.6 we depict the results of using variations of the layered neural network models on *Pattern Set B*. The four-layered version of model O (a) with $m = 14$ did not give superior results as compared to its three-layered counterpart (c) with $m = 11$. This may also be verified from Table 4.2. All other models use three layers with $m = 11$ hidden nodes. The *mse* for model S (b) initially decreases rather rapidly and finally stabilises to a low value at around 100 sweeps. The arrow along the abscissa marks the point of termination of this algorithm. Model P (d) has a satisfactory overall performance. However, models C (f) and M (e) result in a final larger value of *mse* at termination. Oscillations are evident for model A (g) while model F (h) fares the worst. These results may be compared with those of Table 4.5.

Fig. 4.7 illustrates the results of using different layered neural network variations on *Pattern Set C*. The four-layered version of model O (a) with $m = 12$ gives the lowest final value of *mse*. This can be verified also from Table 4.3. All other models use three layers with $m = 13$ hidden nodes. The three-layered version of model O (b) has a good overall performance. Models P (e), C (d) and M (c) have similar final *mse* values (at termination), that are higher than that of model O. Models M (c) and A (f) exhibit a lot of oscillations in the process of convergence. The *mse* value for model M is initially rather high but finally decreases somewhat after around 350

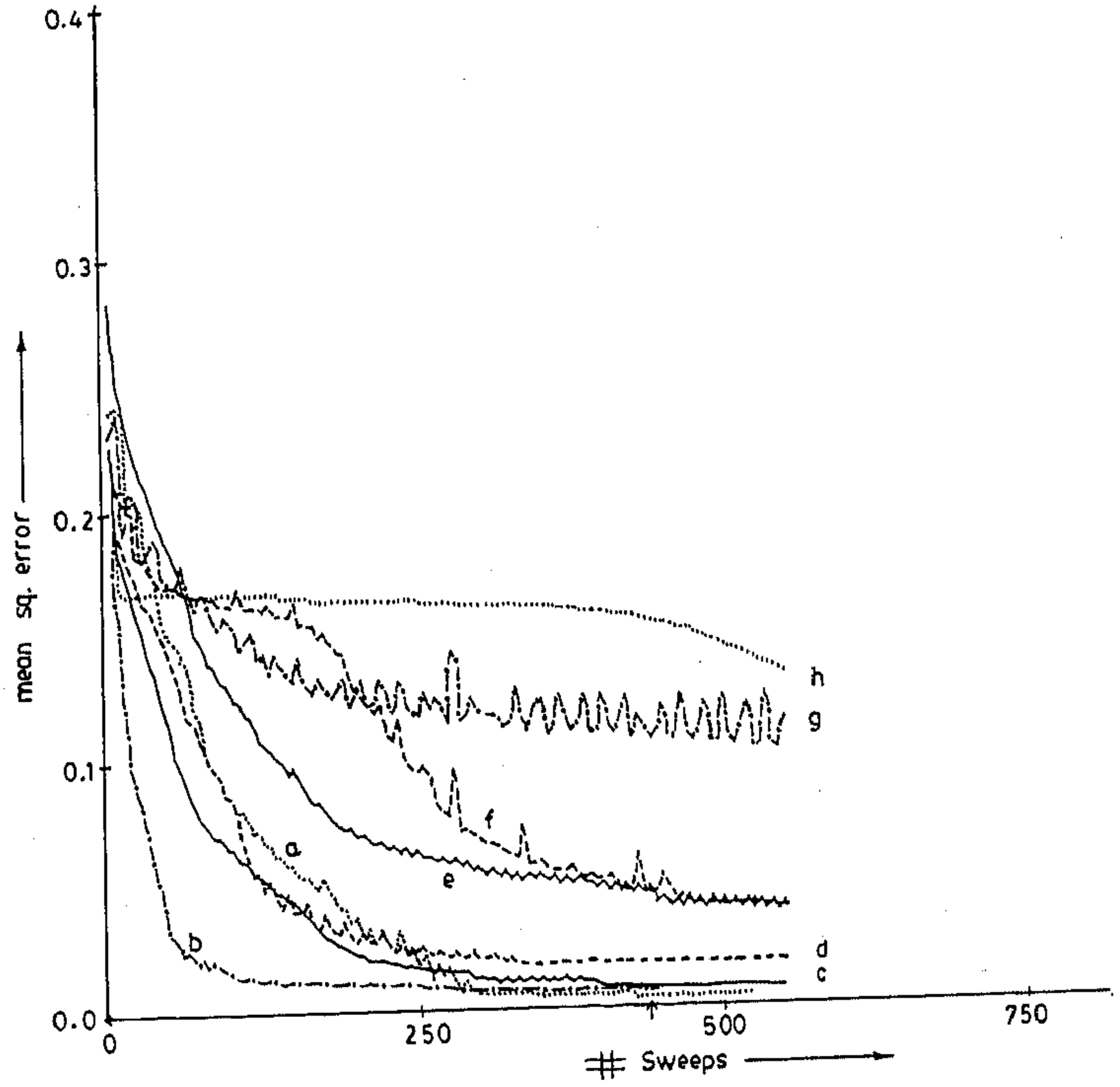

Figure 4.6: Variation of mean square error with number of sweeps for layered neural net models over Pattern Set B

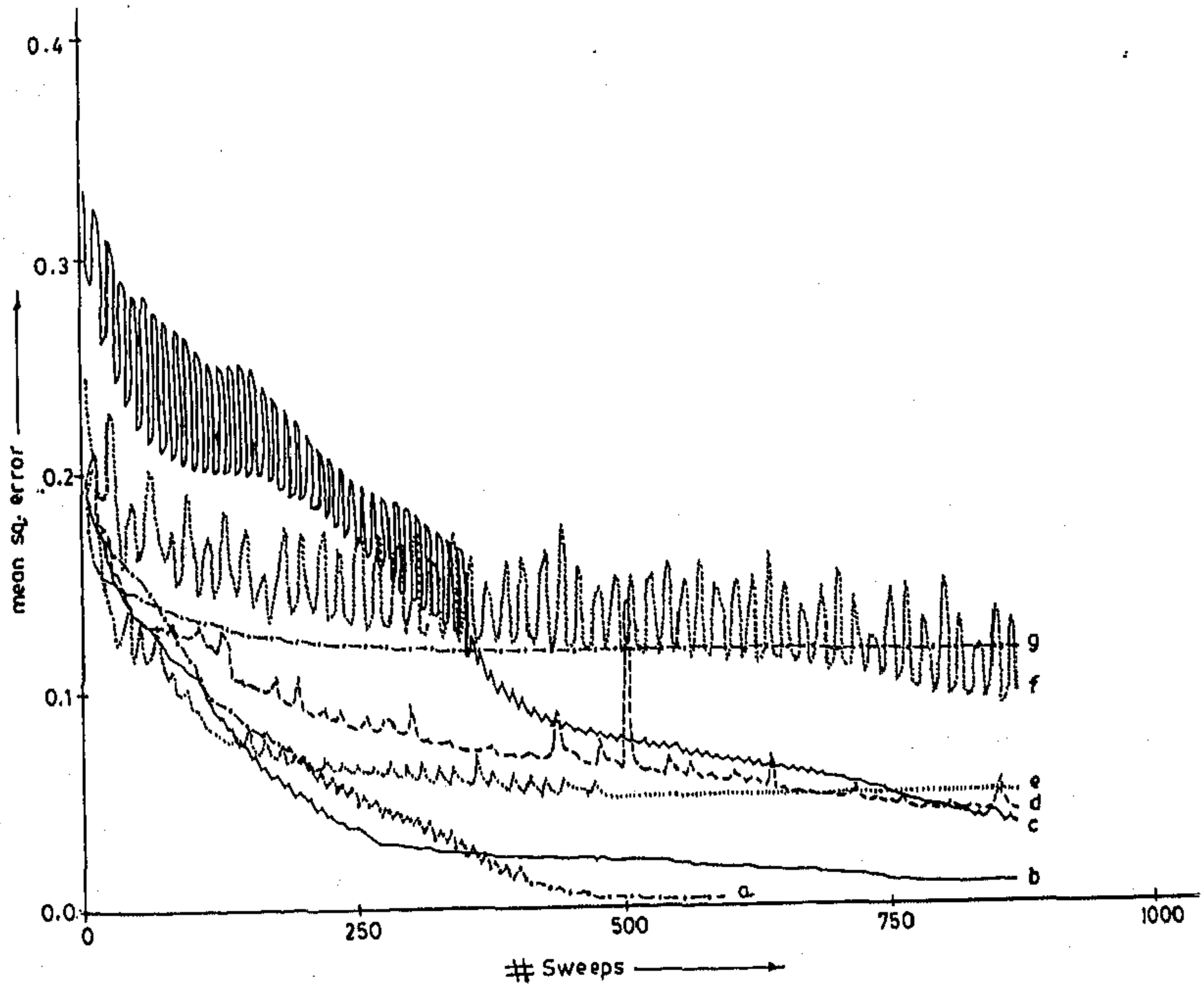

Figure 4.7: Variation of mean square error with number of sweeps for layered neural net models over Pattern Set C

sweeps. Both models A (f) and F (g) behave rather poorly. All results of this figure may be compared with those of Table 4.6.

It is therefore seen that the fuzzy MLP (model O) is better both from the point of overall recognition score as well as the number of sweeps required for convergence.

4.3.3 Using fuzzy Kohonen's model

Table 4.7: Recognition scores for fuzzy Kohonen's net and its nonfuzzy version, on Pattern Set A

Model	Fuzzy						Nonfuzzy	
Size	14×14		16×16		18×18		14×14	
perc	50	10	50	10	50	10	50	10
1	77.4	33.3	65.6	55.8	56.5	55.3	96.9	93.2
2	67.3	60.2	69.3	68.2	63.3	87.5	0.0	0.0
none	55.5	68.3	53.7	52.2	49.4	57.7	4.3	98.6
Overall	68.3	49.2	61.2	55.9	54.6	59.8	52.1	49.2

Tables 4.7-4.9 are used to compare the performance on test set (both classwise and overall) of different sizes of the fuzzy Kohonen's net, trained (self-organized) using *perc* % training samples, for the three pattern sets A, B, C respectively. In all these cases, we have used $s = 0.2$ in eqn. (3.5).

Comparison is made with the performance of the nonfuzzy version of the network. The fuzzy model is found to result in better performances in all the three cases, considering individual class-wise behaviour. Note that for the nonfuzzy case we have provided the results for one network size only, corresponding to which the fuzzy version gave the best result. For the other network sizes the nonfuzzy version resulted in poorer performance.

In addition, Table 4.10 shows a comparison of the classification performance of a 14×14 fuzzy Kohonen's net, trained on *Pattern Set A* using $perc = 50\%$ of the samples, for different values of the scale factor s of eqn. (3.5). Here $s = 0.2$ is found to yield the best results. It may be noted that large values of s result in a greater dependency on the contextual class information part of the input vector during self organization.

Table 4.8: Recognition scores for fuzzy Kohonen's net and its nonfuzzy version, on Pattern Set *B*

Model	Fuzzy						Nonfuzzy	
Size	14 × 14		16 × 16		18 × 18		16 × 16	
perc	50	10	50	10	50	10	50	10
1	63.4	58.7	83.0	64.1	51.7	54.2	10.7	0.0
2	50.5	30.8	55.6	30.2	40.2	42.3	6.1	0.0
none	50.4	78.1	55.6	63.5	56.0	62.4	94.4	100.
Overall	53.7	62.8	62.6	56.4	51.5	55.9	53.7	52.6

Table 4.9: Recognition scores for fuzzy Kohonen's net and its nonfuzzy version, on Pattern Set *C*

Model	Fuzzy						Nonfuzzy	
Size	14 × 14		16 × 16		18 × 18		16 × 16	
perc	50	10	50	10	50	10	50	10
1	77.3	50.6	82.7	51.7	72.7	26.1	53.4	2.8
2	53.8	58.7	19.2	84.7	65.3	32.6	0.0	0.0
none	44.5	51.6	55.4	35.8	53.5	73.1	72.9	96.4
Overall	64.4	51.5	69.4	48.0	65.5	43.0	57.1	35.6

Hence during testing (when $s = 0$) the input vector becomes less *informative*, thereby leading to a poorer recognition score on the test set consisting of the input feature information part only. It may be noted that $0 < s < 0.5$ appears suitable in this context.

Table 4.10: Effect of varying s on the recognition score of a fuzzy Kohonen's net for Pattern Set A

Class	Scale factor s				
	0.1	0.2	0.3	0.5	1.0
1	70.0	77.4	66.0	39.1	40.0
2	69.3	67.3	73.4	73.4	93.8
none	58.0	55.5	62.3	51.2	44.4
Overall	65.5	68.3	65.5	47.4	47.6

The fuzzy partitioning of the pattern classes along with the hard partitioning of the output space is also depicted for each set of patterns. Figs. 4.8-4.10 illustrate the output maps generated for the three pattern sets A, B, C respectively, using 50% of the samples from each class during self-organization and $s = 0.2$ in eqn. (3.5). In each case, parts (a) and (b) show the fuzzy partitioning for classes 1 and 2 separately (for the sake of clarity) while part (c) gives the hard partitioning of the output space (considering all classes). Note that along the horizontal axes each pair of digits denotes the location of a single neuron while along the vertical axes one digit corresponds to a single neuronal location. This is done to compensate for the difference in resolution along the two perpendicular axes. A comparison of these output maps with the corresponding original pattern sets in Figs. 4.1-4.3 brings forth the utility of the fuzzy Kohonen's net in modeling the given pattern sets.

4.3.4 Using k-nearest neighbors classifier

The k-nearest neighbors (k-NN) classifier [2], with $k = 1, 3, 5, 7$, has also been implemented on the three given pattern sets of Figs. 4.1-4.3. The k-NN classifier is reputed to be able to generate piecewise linear decision boundaries and, thereby, is quite efficient in handling concave and linearly nonseparable pattern classes. Therefore, a

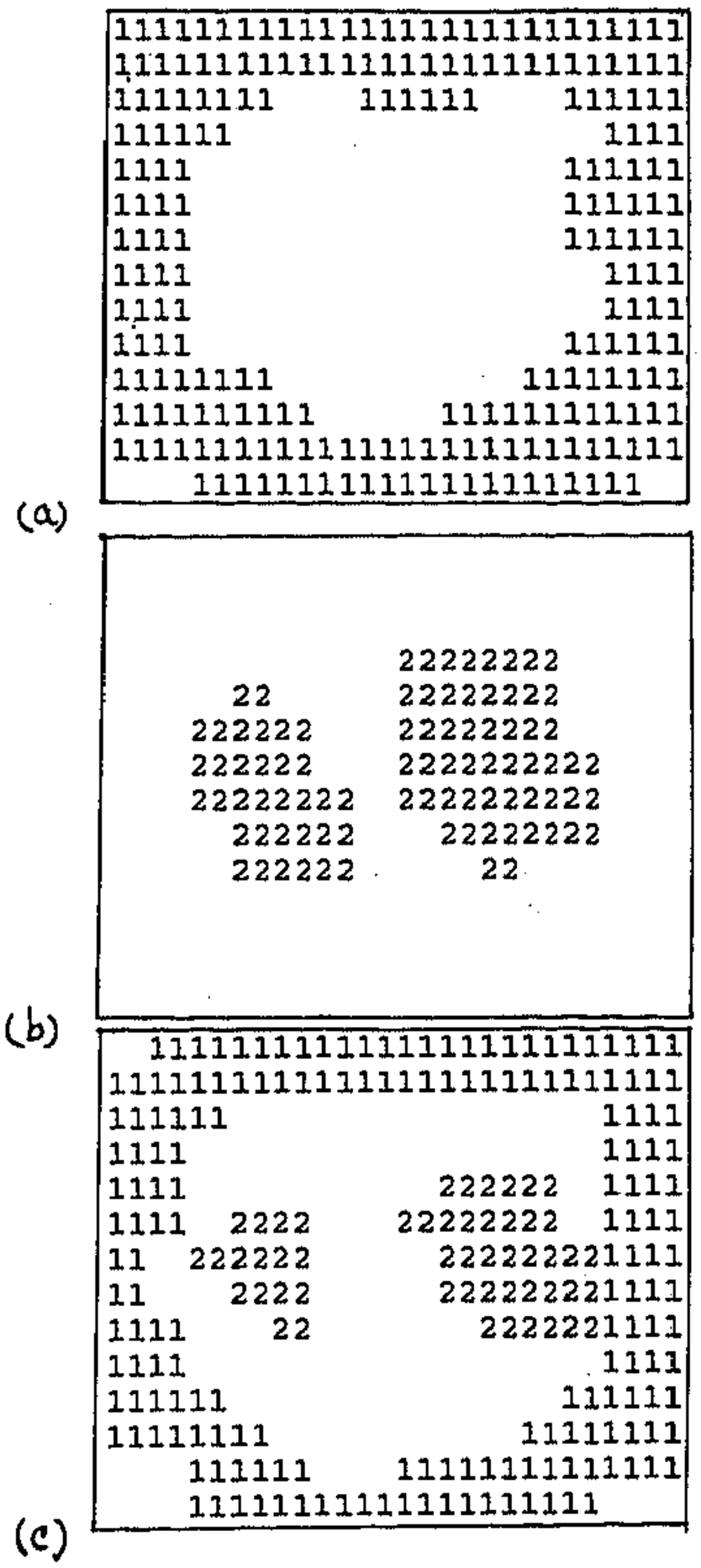

Figure 4.8: Output map generated by 14×14 fuzzy Kohonen's net for Pattern Set A; (a)-(b)Fuzzy and (c)Hard partitioning

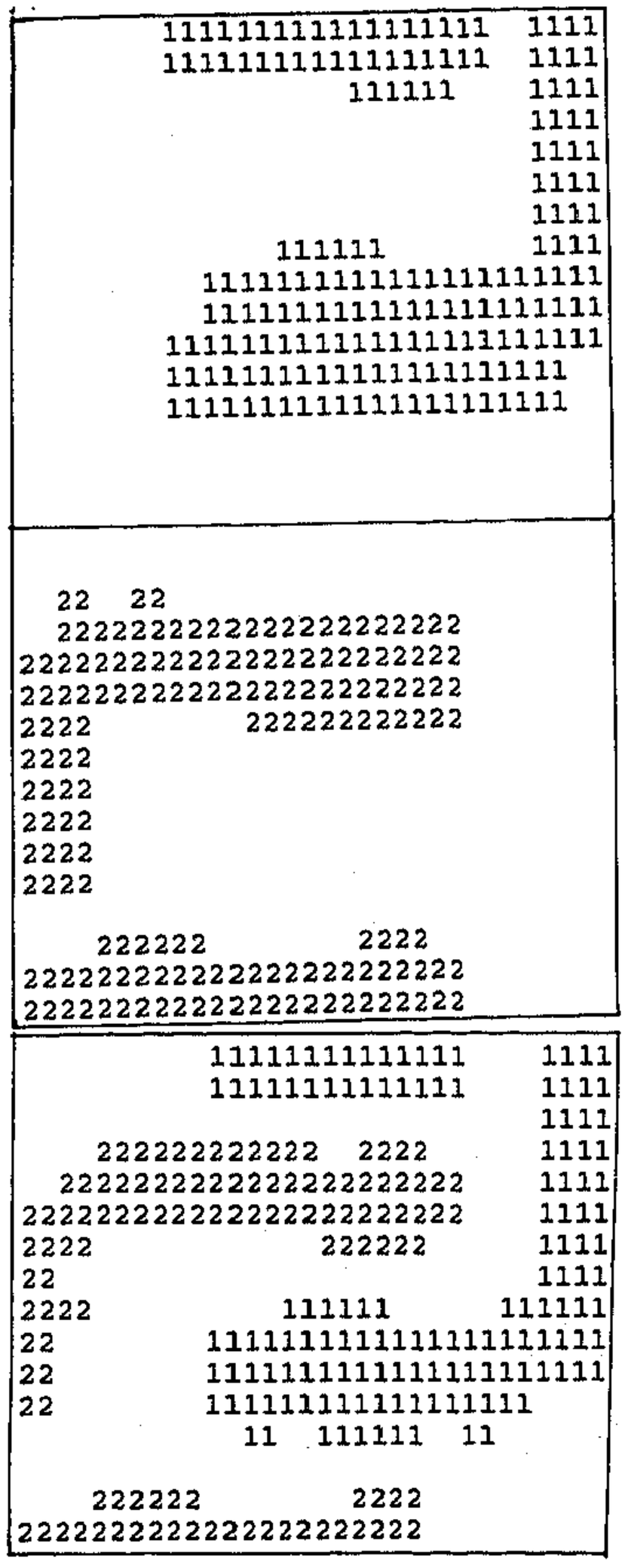

Figure 4.9: Output map generated by 16×16 fuzzy Kohonen's net for Pattern Set B; (a)-(b)Fuzzy and (c)Hard partitioning

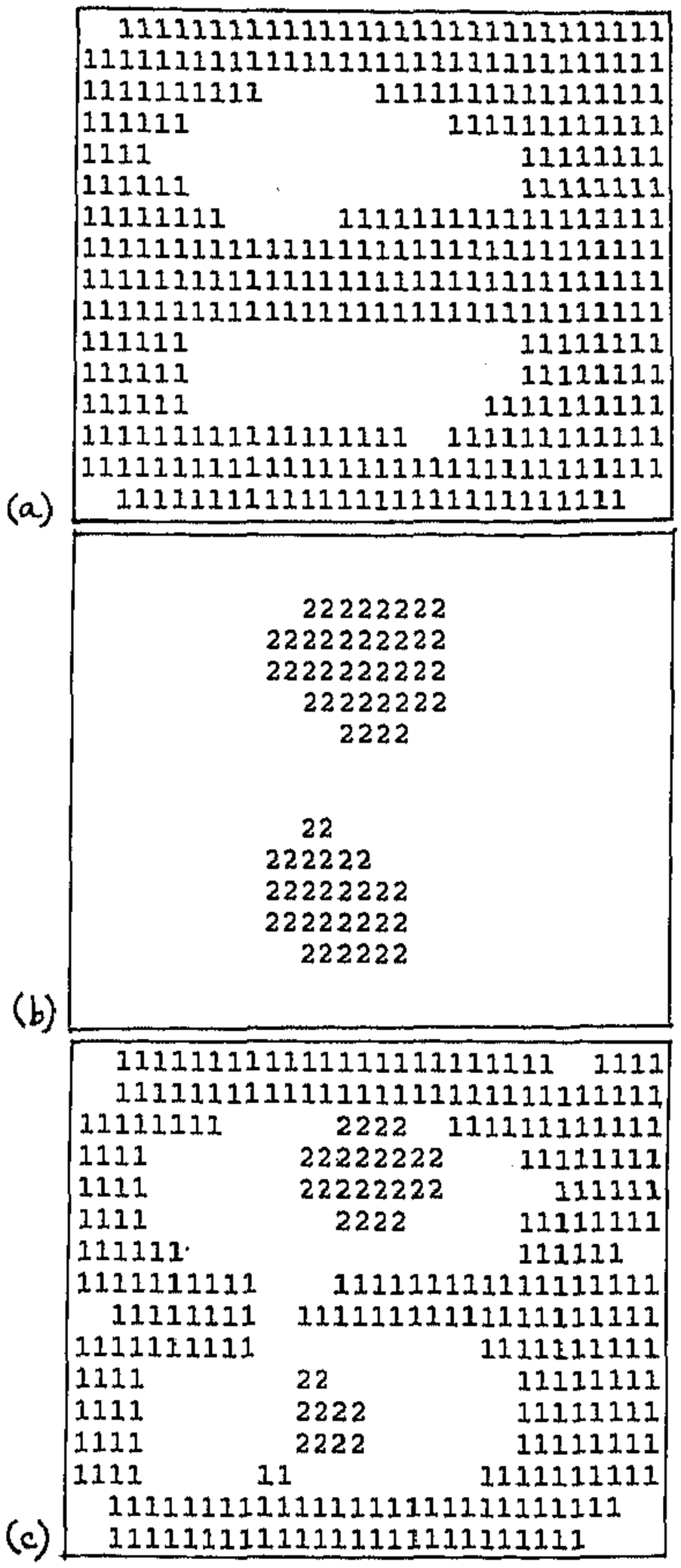

Figure 4.10: Output map generated by 16 × 16 fuzzy Kohonen's net for Pattern Set C; (a)-(b)Fuzzy and (c)Hard partitioning

comparison of the performance of the fuzzy neural models with that of the k-NN classifier on handling the complex decision regions of the aforesaid three synthetic pattern sets is highly desirable.

Table 4.11: Performance of the k-NN classifier

P	*perc*	50			10		
A	$k =$	1	3	5	1	3	5
T	1 (%)	93.5	96.1	95.7	89.6	88.4	83.1
T	2 (%)	79.6	73.4	69.4	62.5	58.0	59.1
	none (%)	81.5	86.4	84.6	72.9	67.7	62.5
A	*Net* (%)	87.5	90.0	88.7	80.5	77.4	72.9
P	*perc*	50			10		
A	$k =$	1	3	5	1	3	5
T	1 (%)	83.0	83.9	80.4	74.6	63.7	58.7
T	2 (%)	83.5	86.6	84.5	72.0	50.3	49.7
	none (%)	84.9	86.6	83.6	79.4	71.0	65.2
B	*Net* (%)	84.1	85.9	83.0	76.5	64.6	60.2
P	*perc*	50			10		
A	$k =$	1	3	5	1	3	5
T	1 (%)	89.6	94.2	97.3	88.5	89.1	86.3
T	2 (%)	88.5	88.5	80.8	63.0	50.0	47.8
	none (%)	81.9	87.7	80.6	67.7	57.3	52.7
C	*Net* (%)	86.8	91.6	90.5	79.7	75.7	72.3

Table 4.11 depicts the recognition scores for $k = 1, 3$ and 5. (Results for $k = 7$ were found to be poorer and are therefore not included here). It is seen from the table that for training set size of *perc* = 50, best results are obtained with $k = 3$, while *perc* = 10 generates good performance with $k = 1$. The k-NN classifier is found to perform better than the conventional MLP, and both the fuzzy and conventional Kohonen's nets. However, the fully supervised fuzzy MLP (model *O*) results in the best performance (both classwise and overall) for all the three data sets.

4.3.5 Fuzzification at the input

The effect of fuzzification at the input on the recognition score vector has been investigated for both the fuzzy MLP and the fuzzy Kohonen's net. The input feature information is given in the $3n$-dimensional space of eqn. (2.10) in terms of the linguistic property sets *low, medium* and *high*. The π-functions representing these properties are defined by the radius λ and centre c values given by eqns. (2.12)-(2.14).

Table 4.12: Effect of varying fuzziness at input for the three-layered fuzzy MLP

$fnos =$		0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
best b (%)		94.3	98.9	100.	100.	100.	100.	100.	100.	100.	100.
perfect p (%)		21.9	0.0	54.1	18.4	34.5	36.8	17.3	46.0	21.9	19.6
mse		.039	.019	.006	.008	.006	.006	.007	.005	.008	.012
T	mse_t	.113	.092	.068	.073	.076	.077	.099	.094	.128	.131
e	1 (%)	85.0	90.5	90.8	88.4	89.3	86.7	87.9	86.0	78.9	86.4
s	2 (%)	75.0	69.3	60.2	67.0	75.0	73.8	57.9	64.7	57.9	54.5
t	*none* (%)	69.7	75.9	89.0	86.6	84.5	91.0	83.8	81.7	78.3	69.4
	Overall t (%)	78.3	82.8	86.7	85.3	86.0	86.8	83.1	82.1	76.4	76.6

Varying λ_{medium} while keeping λ_{low} and λ_{high} of eqns. (2.13)-(2.14) fixed, one can alter the overlapping among the three π-functions. Let $\lambda_{medium} = fnos * \lambda_{medium}$ where $fnos = 1$ for the value of λ_{medium} given by eqn. (2.12). As we decrease $fnos$, the radius λ_{medium} decreases around c_{medium} such that ultimately there is insignificant overlapping between the π-functions *medium* and *low* or *medium* and *high*. This implies that certain regions along the feature axis F_j go under-represented such that $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ and $\mu_{high(F_{ij})}(\vec{F}_i)$ attain small values. Note that the particular choice of the values of the λ's and c's by eqns. (2.12)-(2.14) ensure that for any pattern point $\vec{F}_i$ along the j^{th} axis, at least one of $\mu_{low(F_{ij})}(\vec{F}_i)$, $\mu_{medium(F_{ij})}(\vec{F}_i)$ and $\mu_{high(F_{ij})}(\vec{F}_i)$ should be greater than 0.5 (as explained in Chapter 2). On the other hand, as we increase $fnos$ the radius λ_{medium} increases around c_{medium} such that the amount of overlapping between the π-functions increases.

Tables 4.12-4.13 demonstrate the performance of the fuzzy MLP and the fuzzy Kohonen's net respectively with different values of $fnos$ on *Pattern Set A*. The fuzzy

Table 4.13: Effect of varying fuzziness at input for the fuzzy Kohonen's model

$fnos =$	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3
1	67.8	70.0	67.8	80.8	77.4	60.0	58.7	43.9
2	55.1	53.0	59.1	77.5	67.3	63.2	81.6	91.8
none	45.6	42.6	53.7	53.7	55.5	58.0	53.7	58.0
Overall t	58.2	58.0	61.6	70.5	68.3	59.6	59.4	54.4

MLP uses one hidden layer having 17 nodes with $perc = 10$ while the fuzzy Kohonen's net has been implemented on a 14×14 array using $perc = 50$. This has been done to keep uniformity with the results of Tables 4.1 and 4.7 respectively, where these network configurations yielded good recognition scores. It is observed that the MLP generates good performance for $0.7 < fnos < 1.2$ while for the Kohonen's model $0.8 < fnos < 1.2$ gives good results. This implies that the amount of overlapping signified by eqns. (2.12)-(2.14) indicates the most suitable choice of the values of the λ's and c's. Very large or very small amounts of overlapping among the linguistic properties of the input feature are found to be undesirable.

4.3.6 Contribution of apriori class information for backpropagation

It may be noted from Figs. 4.1 and 4.3 that the *a priori* probability of class 2 for *Pattern Sets A* and *C* is very low as compared to that of the other two classes. Therefore the contribution of class 2 pattern vectors towards weight correction of the MLP (positioning of the decision surface) by eqns. (2.5)-(2.8) is much smaller relative to the contribution of the other cases. This makes the nonfuzzy model O', with its n-dimensional input space, unable to recognize test patterns from class 2 in case of *Pattern Sets A* and *C* (as observed from Tables 4.4 and 4.6). This effect of low *a priori* probabilities has also been experimentally studied by Barnard and Botha [187].

An appropriate basis for studying this issue is provided by the theory of Bayes' classifiers [1]. Consider a two-class problem with *a priori* probabilities p_1 and p_2, and class-conditional probability densities $P_1(x)$ and $P_2(x)$ respectively. Then minimal

error is obtained by classifying a sample x_p into class 1, if $P_1(x_p) > \frac{P_2(x_p)p_2}{p_1}$, and into class 2, otherwise. One immediate implication of this formulation is that it is indeed possible for optimal decision boundaries to completely exclude a class if its *a priori* probability is too low, even if that is not exactly zero.

In order to take into account this fact an error correction term is introduced, such that the modified form of eqn. (2.7) becomes

$$\frac{\partial E}{\partial y_j} = \left(y_j^H - d_j\right) * l\,(1 - p_j) \quad \text{for } h = H \tag{4.9}$$

where $p_j = \frac{|C_j|}{N}$ is the *a priori* probability of class C_j, $|C_j|$ denotes the number of samples in class C_j and l indicates the number of pattern classes. The correction term ensures that the lower the value of p_j, the higher is its contribution in positioning the decision surface.

Table 4.14 demonstrates the effect of eqn. (4.9) to the backpropagation procedure of the MLP in improving the performance of the model in case of *Pattern Sets A* and *C*. Note that this modification is effective in case of pattern classes having widely varying *a priori* probabilities. The use of the multiplicative factor serves to counteract the insignificant contribution (to weight correction and hence to the positioning of the decision surface) of a pattern class having very few samples.

Models O'_p and O_p refer respectively to the variations of the three-layered models O' (nonfuzzy) and O (fuzzy) using the contribution of the *a priori* probabilities. Each network uses m hidden nodes with $perc = 10$ training samples chosen from each pattern class. Note that the recognition score of class 2 patterns improves radically for both *Pattern Sets A* and *C* in case of model O'_p. The performance on the whole is superior for both the models O'_p and O_p (relative to models O' and O respectively) particularly for *Pattern Set C* (involving more complicated decision regions). The relative improvement is always more noticeable in case of model O'_p (the nonfuzzy version).

In this connection, it is worth mentioning that Qi *et al.* [297] have also proposed a technique for modeling *multi-center pattern classes*, with grossly unequal pattern points, by computing output membership around cluster centroids with a factor accounting for the number of data in each cluster. However all such information, like the number of vectors (samples) in each cluster, the number of clusters in each class, the centroid vector, etc., need to be obtained on *a posteriori* basis.

Table 4.14: Contribution of a priori probability on output performance of MLP

Pattern Set		A				C			
Nodes m		17				13			
Model		O'	O'_p	O	O_p	O'	O'_p	O	O_p
best b (%)		73.6	72.5	100.	100.	77.1	86.2	100.	100.
perfect p (%)		3.5	0.0	34.5	1.2	8.1	5.8	32.2	41.4
mse		.096	.094	.006	.008	.104	.06	.008	.006
T	mse_t	.119	.125	.076	.08	.16	.145	.143	.135
e	1 (%)	97.8	95.9	89.3	87.4	87.1	89.1	83.9	80.9
s	2 (%)	0.0	45.4	75.0	73.8	0.0	40.9	84.8	76.0
t	*none* (%)	72.1	57.0	84.5	88.3	51.6	59.8	59.5	69.9
	Overall t (%)	77.5	76.0	86.0	86.2	69.6	73.0	75.4	76.8

4.4 Conclusions and discussion

The effectiveness of using the fuzzy MLP and the fuzzy Kohonen's net in classifying certain linearly nonseparable pattern classes with nonconvex decision regions (Figs. 4.1-4.3) has been demonstrated. The use of the overlapping partitions of linguistic properties *low, medium* and *high* at the inputs enables an n-dimensional feature space to be decomposed into 3^n overlapping sub-regions such that more local information of the feature space can be used. It has been found that this form of representation at the input is effective in modeling overlapping classes (for the vowel data of Chapters 2-3) as well as for concave and disjoint classes (as demonstrated in this chapter). The effect of fuzzification at the input, by varying the radius of the π-function corresponding to the linguistic set *medium*, has been demonstrated for both the models. The contribution of the *a priori* probabilities of the pattern classes in the backpropagation procedure for weight updating is seen to be effective in classifying patterns from classes with low *a priori* probabilities.

Performance of the fuzzy MLP has been compared with those of the conventional MLP and seven other layered neural algorithms. As the self-organizing fuzzy Kohonen's model is used as a partially supervised classifier, its performance has been compared

only with its conventional version (used as a classifier). It has been observed that the fuzzy MLP was on the whole better than the other layered neural algorithms compared, both from the aspect of recognition score as well as the number of sweeps required for convergence. The performance of the k-NN classifier has also been investigated. The k-NN classifier has been found to be better than the conventional MLP, and both the fuzzy and conventional Kohonen's nets. However, the fully supervised fuzzy MLP has resulted in the best overall performance over all three pattern sets.

Like the results of vowel recognition (in Chapters 2 and 3), the fully supervised fuzzy MLP also resulted in better performance over the three sets of linearly nonseparable synthetic patterns as compared to the partially supervised fuzzy Kohonen's net. This may be verified by comparing Tables 4.1-4.3 (fuzzy MLP) with Tables 4.7-4.9 (fuzzy self organizing network). It may be mentioned, in this connection, that the performance of the fuzzy MLP and fuzzy Kohonen's net have been compared with the Bayes' classifier in Chapters 2 and 3 for classifying vowel data (which follows normal distribution).

Chapter 5

Rule Generation and Inferencing using Fuzzy MLP and Fuzzy Kohonen's Models

5.1 Introduction

In Chapters 2-4 we have described fuzzy versions of MLP and Kohonen's net for pattern classification and have demonstrated their performance on speech and synthetic data, along with comparison with Bayes' classifier, k-NN classifier and several other neural algorithms. In the present chapter we develop methodologies for demonstrating how these models can be utilised for inferencing and rule generation problems [276, 279, 280]. This also leads to the design of classification type connectionist expert systems.

In both models, the input can be in quantitative, linguistic, or set forms, or a combination of these, or it can also be missing. In the learning phase the training samples are presented to the network in cycles until it finally converges to a minimum error solution. The connection weights thus obtained constitute the knowledge base for the particular classification problem under consideration. In the testing phase the network infers the decision in terms of class membership values for unknown test samples. When partial information about a test pattern is presented at the input, the model either infers its category or queries the user for *relevant* information in the order of their relative importance (decided from the *learned* connection weights). A measure of confidence (certainty) expressing belief in the decision is also defined.

If asked by the user, the model is capable of justifying its decision in rule form with the antecedent and consequent parts produced in linguistic and *natural* terms. The antecedent clauses are derived from the trained network by backtracking along the *maximum-weighted* paths, whereas the consequent part is generated using a certainty measure. This algorithm enables the model to generate a number of such rules in *If-Then* form. Based on the node excitations, corresponding to a test pattern, the appropriate *If-Then* parts of a rule (justifying the inferred decision) are obtained.

The effectiveness of the algorithms is demonstrated on the speech recognition problem and on the three sets of artificially generated nonconvex (linearly nonseparable) pattern classes. First we present the inferencing and rule generation aspects for the fuzzy MLP, followed by a similar presentation for the fuzzy Kohonen's net. Finally, the results are provided.

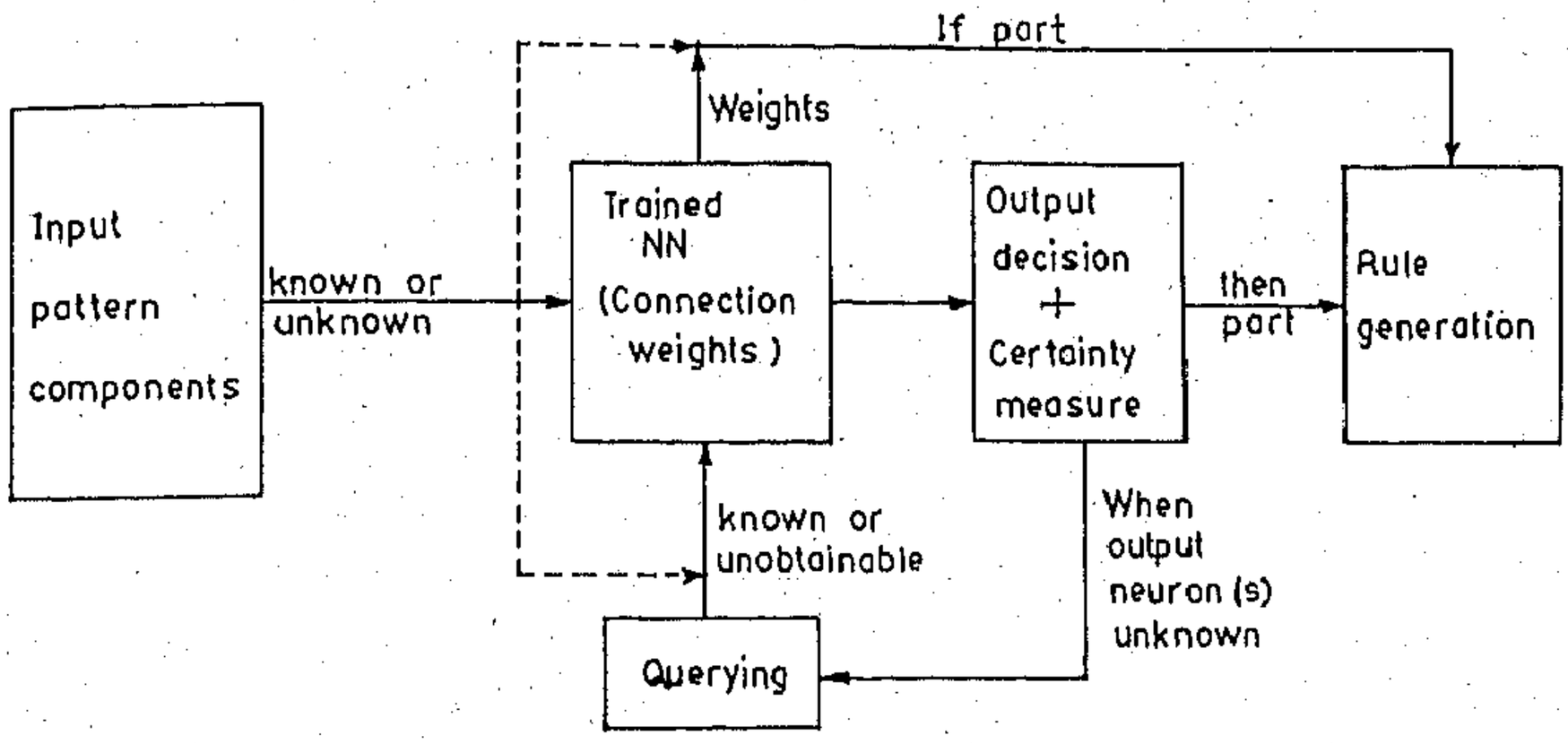

Figure 5.1: Block diagram of inferencing and rule generation phases of the model

5.2 Inferencing in fuzzy MLP

In this section we consider an $(H+1)$-layered fuzzy MLP (as depicted in Fig. 2.1) with $3n$ neurons in the input layer and l neurons in the output layer, such that there are $H-1$ hidden layers. The input vector with components x_j^0 represented as $\vec{F}$ by eqn. (2.10) is clamped at the input layer while the desired l-dimensional output vector with components d_j by eqn. (2.19) is clamped during training at the output layer. At the end of the training phase the model is supposed to have encoded the input-output information distributed among its connection weights. This constitutes the *knowledge base* of the desired decision making system. Handling of imprecise inputs is possible and natural decision is obtained associated with a certainty measure denoting the confidence in the decision. The model is capable of

- inferencing based on complete and/or partial information
- querying the user for unknown input variables that are key to reaching a decision
- producing justification for inferences in the form of *If-Then* rules.

Fig. 5.1 gives an overall view of the various stages involved in the process of inferencing and rule generation.

5.2.1 Input representation

The input can be in quantitative, linguistic or set forms or a combination of these. It is represented as a combination of memberships to the three primary linguistic properties *low, medium* and *high*, modeled as π-functions.

Quantitative form

When the information is in exact numerical form like F_j *is* r_1, say, we determine the membership values for the different linguistic feature properties *low, medium* and *high* in the corresponding 3-dimensional space of eqn. (2.10) by the π-function using eqns. (2.9) and (2.12)-(2.14).

Linguistic form

When the input is given as F_j *is prop* (say), where *prop* stands for any of the primary linguistic properties *low, medium* or *high*, the membership values in the 3-dimensional space of eqn. (2.10) are assigned using the π-sets of eqn. (2.11).

The model can also handle the linguistic hedges [84, 298] *very, more or less* and *not*. Consider a fuzzy set A with the *Concentration (Con)* and *Dilation (Dil)* operators [298] defined as

$$\begin{aligned} \mu_{Con(A)}(z) &= (\mu_A(z))^2 \\ \mu_{Dil(A)}(z) &= (\mu_A(z))^{\frac{1}{2}} \end{aligned} \tag{5.1}$$

Using eqns. (2.11) and (5.1) we define the hedges *very* and *more or less (Mol)* as

$$\begin{aligned} very\ low &\equiv \{Con(L), Con(M), Con(H)\} \\ very\ medium &\equiv \{Con(L), Dil(M), Con(H)\} \\ very\ high &\equiv \{Con(L), Con(M), Con(H)\} \end{aligned} \tag{5.2}$$

and

$$\begin{aligned} more\ or\ less\ low &\equiv \{Con(L), Dil(M), Dil(H)\} \\ more\ or\ less\ medium &\equiv \{Dil(L), Con(M), Dil(H)\} \\ more\ or\ less\ high &\equiv \{Dil(L), Dil(M), Con(H)\} \end{aligned} \tag{5.3}$$

The sets *very low* and *low* or, say, *very high* and *high* are considered to be pairs of different but overlapping sets [84], such that the *minimum (maximum)* feature value has a higher membership to the set *very low (very high)* as compared to that in the set *low (high)*. Hence π-functions are found to be appropriate for modeling these linguistic sets.

The hedge *not* is defined as

$$\mu_{Not(A)}(x) = 1 - \mu_A(x) \tag{5.4}$$

Set form

Here the input is a mixture of linguistic hedges and quantitative terms. Since the linguistic term increases the impreciseness in the information, the membership value of a quantitative term should be lower when modified by a hedge [84]. The modifiers used are *about, less than, greater than* and *between.*

For an input F_j *is about* r_1, we use

$$\mu\,(about\ \ r_1) = \{\mu\,(r_1)\}^{1.25} \tag{5.5}$$

where $F_j = r_1$ is the quantitative input. Note that $\mu(r_1)$ is computed in the corresponding 3-dimensional space of eqn. (2.10) by using eqns. (2.9) and (2.12)-(2.14).

When the input under consideration is F_j *is less than* r_1, the expression becomes

$$\mu\,(less\ \ than\ \ r_1) = \begin{cases} \{\mu\,(r_1)\}^{\frac{1}{2}} & \text{if } r_1 \geq c_{prop} \\ \{\mu\,(r_1)\}^2 & \text{otherwise} \end{cases} \tag{5.6}$$

where c_{prop} denotes c_{low}, c_{medium} and c_{high} given by eqns. (2.12)-(2.14) respectively, for each of the corresponding three overlapping partitions as in eqn. (2.10), and $\mu\,(r_1)$ is computed as explained above.

Similarly, for an input F_j *is greater than* r_1, the expression becomes

$$\mu\,(greater\ \ than\ \ r_1) = \begin{cases} \{\mu\,(r_1)\}^{\frac{1}{2}} & \text{if } r_1 \leq c_{prop} \\ \{\mu\,(r_1)\}^2 & \text{otherwise} \end{cases} \tag{5.7}$$

This also holds for the modifier *more than.*

Input information of the form F_j *is between* r_1 *and* r_2 or F_j *is less than* r_2 *and/but greater than* r_1 may be modeled as the geometric mean of the two component membership values as

$$\mu\,(between\ \ r_1\ \ and\ \ r_2) = \{\mu\,(less\ \ than\ \ r_2) * \mu\,(greater\ \ than\ \ r_1)\}^{\frac{1}{2}} \qquad (5.8)$$

If any input feature F_j is *not available* or *missing*, we clamp the three corresponding neurons $x_k^0 = x_{k+1}^0 = x_{k+2}^0 = 0.5$, such that $k = (j-1) * 3 + 1$. Here $1 \leq k \leq 3n$ and $1 \leq j \leq n$, where n is the dimension of the input pattern vector. We use

$$no\ \ information \equiv \left\{\frac{0.5}{L}, \frac{0.5}{M}, \frac{0.5}{H}\right\} \qquad (5.9)$$

as 0.5 represents the *most ambiguous* value in the fuzzy membership concept. We also tag these input neurons with $noinf_k^0 = noinf_{k+1}^0 = noinf_{k+2}^0 = 1$. Note that in all other cases the variable $noinf_k^0$ is tagged with 0 for the corresponding input neuron k, indicating absence of ambiguity in its input information.

The appropriate input membership values obtained by eqns. (2.9)-(2.10) and (5.1)-(5.9), with/ without the hedges or modifiers, are clamped at the corresponding input neurons.

5.2.2 Forward pass

The l-dimensional output vector with components y_j^H is computed using eqns. (2.1)-(2.2) in a single forward pass. This output vector, with components in the range [0,1], gives the inferred membership values of the test pattern to the l output classes. Associated with each neuron j in layer $h+1$ are also

- its confidence estimation factor $conf_j^{h+1}$
- a variable $unknown_j^{h+1}$ providing the sum of the weighted information from the preceding *ambiguous* neurons i in layer h having $noinf_i^h = 1$
- a variable $known_j^{h+1}$ giving the sum of the weighted information from the (remaining) *non-ambiguous* preceding neurons with $noinf_i^h = 0$.

Note that for a neuron j in layer $h+1$ with no preceding neurons i tagged with $noinf_i^h = 1$, we have $unknown_j^{h+1} = 0$. For neuron j we define

$$\begin{aligned} unknown_j^{h+1} &= \sum_i w_{ji}^h y_i^h \\ unden_j^{h+1} &= \sum_i |w_{ji}^h| \end{aligned} \tag{5.10}$$

for all i having $noinf_i^h = 1$ and

$$known_j^{h+1} = \sum_i w_{ji}^h y_i^h \tag{5.11}$$

for all i with $noinf_i^h = 0$, where for neurons in layer $h > 0$ we have

$$noinf_j^h = \begin{cases} 1 & \text{if } |known_j^h| \leq |unknown_j^h| \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

For neuron j in the input layer ($h = 0$), the value of $noinf_j^0$ is assigned as explained earlier with reference to eqn. (5.9). Neuron j with $noinf_j^h = 1$ signifies the lack of meaningful information. For an input neuron this implies missing input information while for other neurons ($h > 0$) this is an indicator to the transmission of a larger proportion of weighted *ambiguous information* as compared to *more certain information* from the input layer.

Using eqns. (2.1)-(2.2) and (5.10)-(5.12), we define

$$conf_j^h = \begin{cases} \left| \frac{\sum_i y_i^{h-1} w_{ji}^{h-1}}{unden_j^h} \right| & \text{if } noinf_j^h = 1 \text{ and } h > 0 \\ y_j^h & \text{otherwise} \end{cases} \tag{5.13}$$

Note that $conf_j^h$ is comparable either among the set of neurons having $noinf_j^h = 1$, or among those with $noinf_j^h = 0$, but not between the neurons belonging to these two different sets. In the output layer ($h = H$) if $noinf_j^H = 0$ then $conf_j^H$ is higher for neurons having larger y_j^H, implying a greater belongingness to output class j. Hence this is a measure of the confidence in the decision. However if $noinf_j^H = 1$ then $conf_j^H$ gives a measure of the confidence of the *ambiguous* neuron output. This is because as $unden_j^h$ by eqn. (5.10) (absolute sum of connection weights from *ambiguous* preceding layer neurons) increases, the confidence $conf_j^h$ decreases and vice versa.

If there is no output neuron j with $noinf_j^H = 1$, then the system finalises the decision inferred irrespective of whether the input information is complete or partial.

In case of partial inputs, this implies presence of all the *necessary* features required for taking the decision. It may be mentioned that the weights (learned during training), that constitute the *knowledge-base*, play an important part in determining whether a missing input feature information is *essential* to the final inferred decision or not. This is because these weights are used in computing the $noinf_j^h$'s for the neurons by eqns. (5.10)-(5.12) and these in turn determine whether the inferred decision may be taken.

It is to be noted that the difficulty in arriving at a particular decision in favour of class j is dependent not only on the membership value y_j^H but also on its differences with other class membership values y_i^H, where $i \neq j$. To take this factor into account, a certainty measure (for each output neuron) is defined as

$$bel_j^H = y_j^H - \sum_{i \neq j} y_i^H \tag{5.14}$$

where $bel_j^H \leq 1$. The higher the value of $bel_j^H (> 0)$, the lower is the difficulty in deciding an output class j and hence the greater is the degree of certainty of the output decision.

5.2.3 Querying

If the system has not yet reached a conclusion at the output layer (as explained in Section 5.2.2) to complete the session, it must find an input neuron with *unknown* activation and ask the user for its value. If there is any neuron j in the output layer H with $noinf_j^H = 1$ by eqn. (5.12), we begin the querying phase.

We select the *unknown* output neuron j_1 from among the neurons with $noinf_j^H = 1$ such that $confj_{j_1}^H$ by eqn. (5.13) (among them) is maximum. This enables starting the process at an output neuron that is *most certain* among the *ambiguous* neurons. We pursue the path from neuron j_1 in layer H, in a *top-down* manner, to find the *ambiguous* neuron i_1 in the preceding layer ($h = H - 1$) with the greatest absolute influence on neuron j_1. For this, we select $i = i_1$ such that

$$|w_{j_1 i_1}^h * y_{i_1}^h| = \max_i |w_{j_1 i}^h * y_i^h| \text{ where } noinf_i^h = 1 \tag{5.15}$$

This process is repeated until the input layer ($h = 0$) is reached. Then the model

queries the user for the value of the corresponding input feature u_1 such that

$$u_1 = (i_1 - 1) \bmod 3 + 1 \tag{5.16}$$

where $1 \leq i_1 \leq 3n$, $1 \leq u_1 \leq n$ and n is the dimension of the input pattern vector.

When the user is asked for the value of a *missing* variable, s/he can respond in any of the forms stated in Section 5.2.1. However if a *missing* input variable of eqn. (5.9) is found to be missing once again, we now tag it as *unobtainable*. This implies that the value of this variable will not be available for the remainder of this session. The inferencing mechanism treats such variables as *known* with values $x^0_{k_1} = x^0_{k_1+1} = x^0_{k_1+2} = 0.5$ but with $noinf^0_{k_1} = noinf^0_{k_1+1} = noinf^0_{k_1+2} = 0$ such that $k_1 = (u_1 - 1) * 3 + 1$. We now have

$$information \equiv \left\{\frac{0.5}{L}, \frac{0.5}{M}, \frac{0.5}{H}\right\} \tag{5.17}$$

Note the difference from eqn. (5.9) in the value of $noinf^0_k$ and its effect in the confidence estimation by eqns. (5.10)-(5.13). The response from an *unobtainable* input variable might allow the neuron activations in the following layers to be inferred, unlike that of a *missing* variable. Besides, a *missing* variable has a temporary value of 0.5 that may be changed later in the session, whereas an *unobtainable* variable has a *known final* value of 0.5.

Once the requested input variable is supplied by the user, the procedure in Section 5.2.2 is followed either to infer a decision or to again continue with further querying. On completion of this phase all neurons in the output layer have $noinf^H_j = 0$ by eqn. (5.12).

5.2.4 Justification

The user can ask the system why it inferred a particular conclusion. The system answers with an *If - Then* rule applicable to the case at hand. It is to be noted that these *If - Then* rules are not represented explicitly in the knowledge base; they are generated by the *inferencing system* from the connection weights as needed for explanations. As the model has already inferred a conclusion (in this stage), we take a subset of the currently known information to justify this decision. A particular conclusion regarding output j is inferred depending upon the certainty measure bel^H_j. It is ensured that output nodes j with $bel^H_j > 0$ (or, large y^H_j values) are chosen for

obtaining the justification. This is because explanation becomes feasible only when the decision is not uncertain.

Output layer

Let the user ask for the justification about a conclusion regarding class j. Starting from the output layer $h = H$, the process continues in a *top-down* manner until the input layer ($h = 0$) is reached. In the first step, for layer H, we select those neurons i in the preceding layer that have a positive impact on the conclusion at output neuron j. Hence we choose neuron i of layer $H-1$ if $w_{ji}^{H-1} > 0$. Let the set of m_{H-1} neurons of layer $H-1$, so selected, be $\left\{a_1^{H-1}, a_2^{H-1}, \ldots, a_{m_{H-1}}^{H-1}\right\}$ and let their connection weights to neuron j in layer H be given as $\left\{wet_{a_1^{H-1}} = w_{ja_1}^{H-1}, \ldots, wet_{a_{m_{H-1}}^{H-1}} = w_{ja_{m_{H-1}}}^{H-1}\right\}$. For the remaining layers we obtain the *maximum weighted* paths through these neurons down to the input layer.

Intermediate layers

We select neuron i in layer $0 < h < H-1$ if

$$y_i^h > 0.5, \text{ and}$$

$$wet_{ih} = \max_{a_k^{h+1}} \left[wet_{a_k^{h+1}} + w_{a_k i}^h\right] \tag{5.18}$$

such that $wet_{ih} > 0$. Let the set of m_h neurons so chosen be $\left\{a_1^h, a_2^h, \ldots, a_{m_h}^h\right\}$ and their cumulative link weights to neuron j in layer H be $\left\{wet_{a_1^h}, wet_{a_2^h}, \ldots, wet_{a_{m_h}^h}\right\}$ respectively, by eqn. (5.18). Note that this heuristic ensures that each of the selected m_h neurons have a significant output response y_i^h. This implies choosing a path with neurons that are currently active for deciding the conclusion that is being justified. It also enables each neuron i to lie along one of the *maximum weighted* paths from the input layer ($h = 0$) to the output node j in $h = H$, by choosing only one of the m_{h+1} previously selected paths that provides the largest net weight wet_{ih}.

Input layer

Let the process of eqn. (5.18) result in m_0 chosen neurons(paths) in(from) the input layer ($h = 0$). These neurons indicate inputs that are *known* and have contributed to

the ultimate positivity of the conclusion at neuron j in the output layer H. It may happen that $m_0 = 0$, such that no clear justification may be provided for a particular input-output case. This implies that no suitable path can be selected by eqn. (5.18) and the process terminates.

Let the set of the selected m_0 input neurons be $\left\{a_1^0, a_2^0, \ldots, a_{m_0}^0\right\}$ and their corresponding path weights to neuron j in layer H be $\left\{wet_{a_1^0}, wet_{a_2^0}, \ldots, wet_{a_{m_0}^0}\right\}$. We arrange these neurons in the decreasing order of their *net impacts*, where we define the net impact for neuron i as

$$net\ \ impact_i = y_i^0 * wet_{i^0}$$

Then we generate clauses for an *If-Then* rule from this ordered list until

$$\sum_{i_s} wet_{i_s^0} > 2 \sum_{i_n} wet_{i_n^0} \tag{5.19}$$

where i_s indicates the input neurons selected for the clauses and i_n denotes the input neurons remaining from the set $\left\{a_1^0, a_2^0, \ldots, a_{m_0}^0\right\}$ such that

$$|i_s| + |i_n| = m_0$$

and $|i_s|$, $|i_n|$ refer respectively to the number of neurons selected and remaining from the said set. This heuristic allows selection of those *currently active* input neurons contributing *the most* to the final conclusion (among those lying along the maximum weighted paths to the output node j) as the clauses of the antecedent part of a rule. Hence, it enables the *currently active* test pattern inputs (current evidence) to influence the generated *knowledge base* (connection weights learned during training) in producing a rule to justify the *current* inference.

Clause generation

For a neuron i_{s_1} in the input layer ($h = 0$), selected for clause generation, the corresponding input feature u_{s_1} is obtained as in eqn. (5.16). The antecedent of the rule is given in linguistic form with the linguistic property being determined as

$$prop = \begin{cases} low & \text{if } i_{s_1} - 3(u_{s_1} - 1) = 1 \\ medium & \text{if } i_{s_1} - 3(u_{s_1} - 1) = 2 \\ high & \text{otherwise} \end{cases} \tag{5.20}$$

Here, the 3-dimensional components for the input feature u_{s_1} correspond to the appropriate part of the test pattern vector (given in quantitative, linguistic or set form and converted to the respective 3-dimensional space of eqn. (2.10)). Suppose that the relevant input feature had been initially supplied in linguistic form as *medium* with the individual components given by eqn. (2.11). The neuron i_{s_1} selected for clause generation by eqns. (5.18)-(5.19) can, however, result in feature u_{s_1} corresponding to any of the three properties *low, medium* or *high* by eqn. (5.20). This is because the path generated during backtracking is primarily determined by the connection weight magnitudes encoded during training. However, the test pattern component magnitudes at the input also play a part in determining whether the input neuron i_{s_1} can be selected or not. In the example under consideration, the input feature components being {0.7, 0.95, 0.7}, the linguistic property *prop* can be either *low* or *medium* or *high* and is not constrained to be *medium* only. Therefore, feature properties highlighted for the input pattern may not necessarily be reflected in a similar manner while selecting the value of *prop* in feature u_{s_1} for a clause of the rule. In fact, such an input feature component may also not be selected at all as an antecedent clause.

A linguistic hedge *very, more or less* or *not* may be attached to the linguistic property in the antecedent part, if necessary. We use the mean square distance $d(u_{s_1}, pr_m)$ between the 3-dimensional input feature values corresponding to feature u_{s_1} and the linguistic property *prop* of eqn. (5.20), with or without modifiers, represented as pr_m. The corresponding 3-dimensional values for pr_m corresponding to *prop* are given by eqn. (2.11) (with no modifiers) and by eqns. (5.2)-(5.4) with the modifiers *very, more or less* and *not* respectively. The pr_m for which $d(u_{s_1}, pr_m)$ is the *minimum* is selected as the antecedent clause corresponding to feature u_{s_1} (or neuron i_{s_1}) for the rule justifying the conclusion regarding output neuron j.

This procedure is repeated for all the $|i_s|$ neurons selected by eqn. (5.19) to generate a set of conjunctive antecedent clauses for the rule regarding inference at output node j. All input features (of the test pattern) need not necessarily be selected for antecedent clause generation.

Consequent deduction

The consequent part of the rule can be stated in quantitative form as membership value y_j^H to class j. However a more *natural* form of decision can also be provided for the class j, having significant membership value y_j^H, considering the value of bel_j^H of eqn. (5.14). For the linguistic output form, we use

1. *very likely* for $0.8 \leq bel_j^H \leq 1$
2. *likely* for $0.6 \leq bel_j^H < 0.8$
3. *more or less likely* for $0.4 \leq bel_j^H < 0.6$
4. *not unlikely* for $0.1 \leq bel_j^H < 0.4$
5. *unable to recognize* for $bel_j^H < 0.1$.

In principle it should be possible to examine a connectionist network and produce every such *If-Then* rule. These rules can also be used to form the knowledge base of a traditional expert system.

An example

Consider the simple 3-layered network given in Fig. 5.2 demonstrating a simple rule generation instance regarding class 1. Let the paths be generated by eqn. (5.18). A sample set of connection weights w_{ji}^h, input activation y_i^0 and the corresponding linguistic labels are depicted in the figure. The solid and dotted-dashed paths (that have been selected) terminate at input neurons i_s and i_n respectively, as determined by eqn. (5.19). The dashed lines indicate the paths not selected by eqn. (5.18), using the w_{ji}^h and y_i^h values in the process. Let the certainty measure for the output neuron under consideration be 0.7. Then the rule generated by the model in this case to justify its conclusion regarding class 1 would be

If F_1 is *very medium AND* F_2 is *high*
then *likely* class 1.

In this case, the *net path weights* by eqn. (5.19) at the end of the clause selection process are found to be 2.7 (= 1.6 + 1.1) and 1.05 for the *selected* i_s and *not selected*

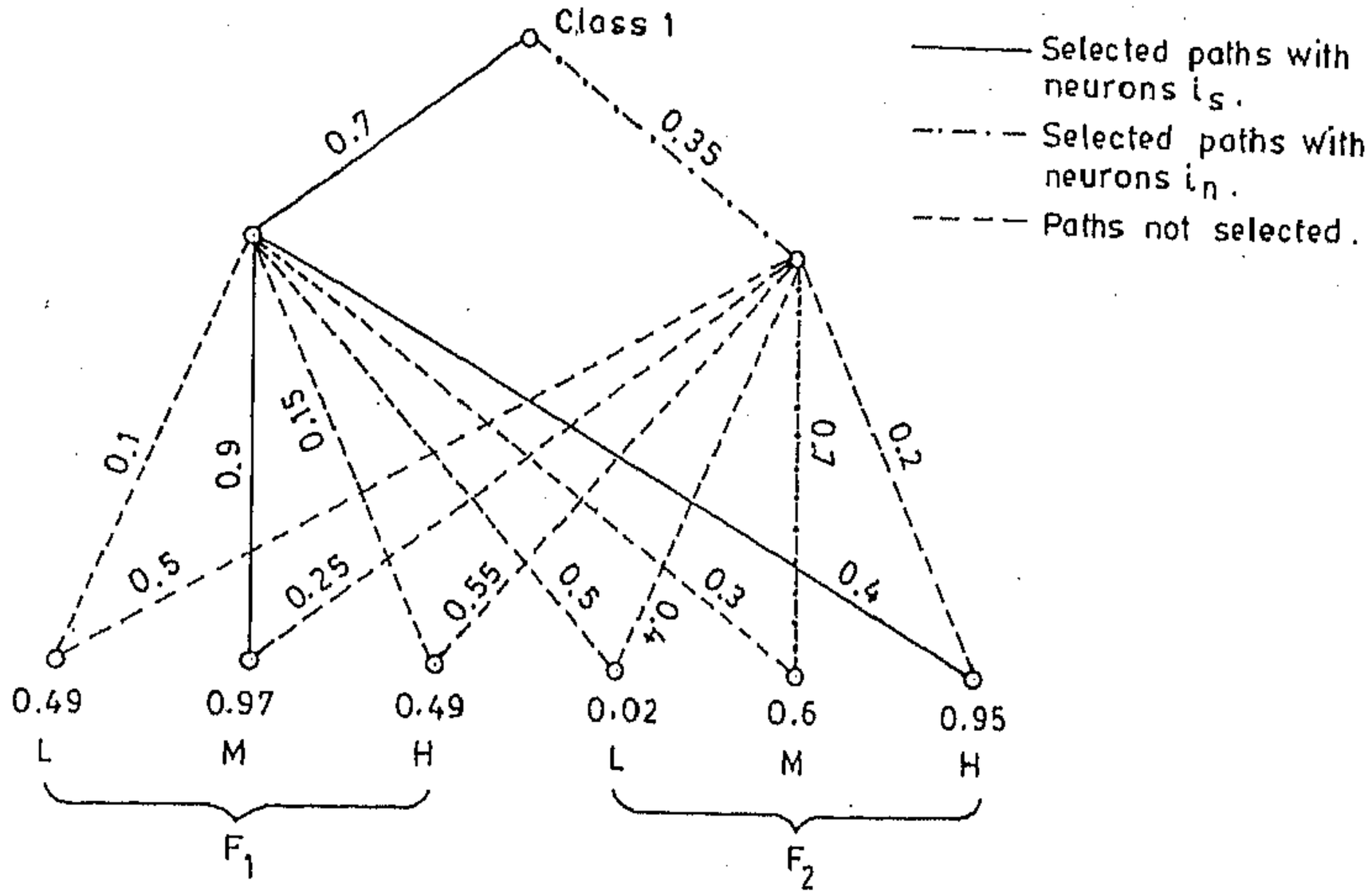

Figure 5.2: Example to demonstrate rule generation scheme by backtracking

i_n neurons respectively such that $2.7 > 2 * 1.05$. The modifier *very* is obtained by applying appropriate operators [84], and this is found to result in the *minimum* value for $d(u_{s_1}, pr_m)$.

To demonstrate querying, let us consider F_1 to be initially *unknown*. Then $y_1^0 = y_2^0 = y_3^0 = 0.5$, with the other values corresponding to those given in Fig. 5.2. From eqns. (5.10)-(5.12), we have $known_1^1 = 0.57$, $known_2^1 = 0.618$, $unknown_1^1 = 0.575$, $unknown_2^1 = 0.65$, and therefore $noinf_1^1 = noinf_2^1 = noinf_1^2 = 1$. As the system cannot reach any conclusion in this state, the querying phase is started. In this case, the only *unknown* input feature is F_1 and it can be supplied in any of the forms mentioned in Section 5.2.1.

5.3 Inferencing in fuzzy Kohonen's net

The algorithm described in this section is somewhat analogous to that described in Section 5.2 for the fuzzy MLP based model. Here we consider the inferencing ability of the fuzzy Kohonen's network of Chapter 3. To prevent repetitions, only those

portions of this partially supervised fuzzy Kohonen's net-based model that are specifically different from the fully supervised fuzzy MLP will be discussed. Besides, it is worth noting that the fuzzy Kohonen's model uses a $(3n + l)$-dimensional input space consisting of the contextual class information along with the linguistic features.

The input feature F_j may be provided in quantitative, linguistic and/or set forms, as described in Section 5.2.1. However, if any input feature F_j is *not available* or *missing*, we clamp the three corresponding input vector components of eqn. (5.9) (incident on the neurons) $x_k = x_{k+1} = x_{k+2} = 0.5$, such that $k = (j-1)*3+1$. Here $1 \leq k \leq 3n+l$ and $1 \leq j \leq n$, where n is the dimension of the input pattern vector. Note that here we are dealing with the $3n$-dimensional x' constituent (feature information) of the input vector x of eqn. (3.4). We also tag these vector components with values $ininf_k = ininf_{k+1} = ininf_{k+2} = 1$. This is a tag used for determining whether the corresponding neuron is *known* or *unknown* by eqns. (5.21)-(5.22). It is to be mentioned that for the remaining $\{3(n - |j|) + l\}$ input vector components of x of eqn. (3.4), the corresponding variables $ininf_k$ are tagged with 0 (where $|j|$ denotes the number of *missing* input features). This indicates absence of ambiguity in the corresponding input information components.

5.3.1 Forward pass

Initially the system prompts the user for the input feature information that may be provided in any of the forms listed earlier. The components of the l-dimensional contextual class information part x'' of the input vector x (of eqn. (3.4)) along with the corresponding $ininf_k$'s are kept clamped at 0. The N^2 neuron outputs η_i are computed using eqns. (3.1) and (3.9). Analogous to the procedure discussed in Section 5.2.2, here each neuron i is associated with

- its confidence estimation factor $conf_i$
- a variable $unknown_i$ providing the sum of the weighted information from the input components x_k having $ininf_k = 1$
- a variable $known_i$ giving the sum of the weighted information from the (remaining) *non-ambiguous* input constituents with $ininf_k = 0$.

Note that when there are no input components x_k tagged with $ininf_k = 1$, we have $unknown_i = 0$ for all the N^2 neurons. The contextual class information part x'' of x is kept clamped at zero and therefore produces no contribution in this stage. For neuron i we define

$$\begin{aligned} unknown_i &= \sum_k m_{ik} x_k \\ unden_i &= \sum_k |m_{ik}| \end{aligned} \tag{5.21}$$

for all k having $ininf_k = 1$ and

$$known_i = \sum_k m_{ik} x_k \tag{5.22}$$

for all k with $ininf_k = 0$. For the N^2 neurons, we have

$$noinf_i = \begin{cases} 1 & \text{if } |known_i| \leq |unknown_i| \\ 0 & \text{otherwise} \end{cases} \tag{5.23}$$

where $noinf_i$ for the i^{th} neuron is a tag analogous to $ininf_k$ for the k^{th} input component.

Using eqns. (3.1),(3.9) and (5.21)-(5.23), we define

$$conf_i = \begin{cases} |\frac{\eta_i}{unden_i}| & \text{if } noinf_i = 1 \\ \eta_i & \text{otherwise} \end{cases} \tag{5.24}$$

Let neurons $p1$ and $p2$ generate the highest and second highest output responses η_{f_p} and η_{s_p} respectively for pattern p with input vector x. If neither $noinf_{p1} = 1$ nor $noinf_{p2} = 1$, then the system finalises the decision inferred irrespective of whether the input information is complete or partial. In case of partial inputs, this implies presence of the *necessary* features required for taking the decision. The weights (learned during training) are used in computing the $noinf_i$'s for the neurons by eqns. (5.21)-(5.23) and they in turn determine whether the inferred decision may be taken.

We decide that η_{f_p} and η_{s_p} are in favour of classes C_{k1} and C_{k2} respectively when

$$\begin{aligned} m_{(p1)(3n+k1)} &= \max_k \left[m_{(p1)(3n+k)} \right] \\ m_{(p2)(3n+k2)} &= \max_k \left[m_{(p2)(3n+k)} \right] \end{aligned} \tag{5.25}$$

where $k = 1, \ldots, l$. Note that the $(3n + l)$-dimensional connection weights m_i's are *learned* during self-organization and constitute the *knowledge base* for the problem after appropriate labeling during calibration.

The inferred highest and second highest output memberships $\mu_{k_1}(\eta)$ and $\mu_{k_2}(\eta)$ to classes C_{k_1} and C_{k_2} respectively are given as

$$\begin{aligned} \mu_{k_1}(\eta) &= \frac{m_{(p1)(3n+k1)}}{s} \\ \mu_{k_2}(\eta) &= \frac{m_{(p2)(3n+k2)}}{s} * \frac{\eta_{s_p}}{\eta_{f_p}} \end{aligned} \tag{5.26}$$

with $p_1 = p1$, $p_2 = p2$, $k_1 = k1$, $k_2 = k2$, if $m_{(p1)(3n+k1)} \geq m_{(p2)(3n+k2)} * \frac{\eta_{s_p}}{\eta_{f_p}}$. Otherwise,

$$\begin{aligned} \mu_{k_1}(\eta) &= \frac{m_{(p2)(3n+k2)}}{s} * \frac{\eta_{s_p}}{\eta_{f_p}} \\ \mu_{k_2}(\eta) &= \frac{m_{(p1)(3n+k1)}}{s} \end{aligned} \tag{5.27}$$

where $p_1 = p2$, $p_2 = p1$, $k_1 = k2$ and $k_2 = k1$. Here p_1 and p_2 refer to the neurons inferred to be generating the highest and second highest membership values to classes C_{k_1} and C_{k_2} respectively and s is the scaling factor from eqn. (3.5).

A certainty measure for the neuron p_1 is defined as

$$bel_{p_1}^{k_1} = \frac{1}{s} * \left[m_{(p_1)(3n+k_1)} - \sum_{k=1}^{l} m_{(p_1)(3n+k)} \right] \tag{5.28}$$

where $k \neq k_1$ and $bel_{p_1}^{k_1} \leq 1$. Here k_1 and p_1 are obtained from eqns. (5.26)-(5.27).

5.3.2 Querying

If either $noinf_{p_1} = 1$ or $noinf_{p_2} = 1$ by eqn. (5.23), where p_1 and p_2 are obtained from eqns. (5.26)-(5.27), we begin the querying phase. We select the *unknown* output neuron i_1 from among the p_1^{th} and/or p_2^{th} neuron(s) with $noinf_{p_1} = 1$ and/or $noinf_{p_2} = 1$ such that $conf_{i_1}$ by eqn. (5.24) (among them) is maximum.

We pursue the path from neuron i_1 to find the *ambiguous* input feature vector component x_{k_1} with the greatest absolute influence on neuron i_1. For this, we select $k = k_1$ such that

$$|m_{i_1 k_1} * x_{k_1}| = \max_k |m_{i_1 k} * x_k| \text{ where } ininf_k = 1 \tag{5.29}$$

Here $ininf_k$ is obtained as explained in page 147 and $1 \leq k_1 \leq 3n + l$. The model queries the user for the value of the corresponding input feature j_1 such that

$$j_1 = (k_1 - 1) \bmod 3 + 1 \tag{5.30}$$

where $1 \leq j_1 \leq n$ and n is the dimension of the input pattern vector. Note that here only the $3n$-dimensional input feature information vector x' of eqn. (3.4) is under consideration.

If a *missing* input variable (as represented by eqn. (5.9)) is found to be missing once again, we now tag it as *unobtainable*. The inferencing mechanism treats such variables as *known* with values $x_{k_1} = x_{k_1+1} = x_{k_1+2} = 0.5$ but with $ininf_{k_1} = ininf_{k_1+1} = ininf_{k_1+2} = 0$. On completion of this phase we have $noinf_{p_1} = noinf_{p_2} = 0$ by eqn. (5.23).

5.3.3 Justification

Let the user ask justification for a conclusion regarding class C_{k_1} at neuron p_1. Starting from neuron p_1, the process reasons *backwards* to the input vector along the *maximum weighted* paths. We select those input feature vector components x_k that have a significant positive impact on the conclusion reached at neuron p_1.

We choose input feature vector component x_k if

$$x_k > 0.5 \tag{5.31}$$

where $0 \leq k \leq 3n$. Let the set of h components so chosen be $\{x_{k_1}, x_{k_2}, \ldots, x_{k_h}\}$ and their corresponding link weights to neuron p_1 be $\{m_{p_1 k_1}, m_{p_1 k_2}, \ldots, m_{p_1 k_h}\}$ respectively.

We arrange the chosen input components in the decreasing order of their *net impacts*, where we define the net impact for x_k as

$$net\ \ impact_k = x_k * m_{p_1 k}$$

Then we generate clauses for an *If-Then* rule from this ordered list (by a procedure analogous to that described in Section 5.2.4 for the fuzzy MLP) until

$$\sum_{k_s} m_{p_1 k_s} > 2 \sum_{k_n} m_{p_1 k_n} \tag{5.32}$$

where k_s indicates the input components selected for the clauses and k_n denotes the input components remaining from the set $\{x_{k_1}, x_{k_2}, \ldots, x_{k_h}\}$ such that

$$|k_s| + |k_n| = h$$

and $|k_s|$, $|k_n|$ refer respectively to the number of components selected and remaining from the said set.

For an input component $x_{k_{s_1}}$, selected for clause generation, the corresponding input feature j_{s_1} is obtained as in eqn. (5.30), such that $1 \leq j_{s_1} \leq n$ and $1 \leq k_{s_1} \leq 3n$. The antecedent of the rule is given in linguistic form with the linguistic property being determined from eqn. (5.20) (as described for the fuzzy MLP).

The consequent part of the rule can be stated in *quantitative form* as membership value $\mu_{k_1}(\eta)$ to class C_{k_1} by eqns. (5.26)-(5.27). However a more *natural* form of decision (analogous to that discussed for the fuzzy MLP) can also be provided, considering the value of $bel_{p_1}^{k_1}$ of eqn. (5.28).

5.4 Implementation and results

The above-mentioned algorithms have first been tested on the set of 871 vowel sounds, depicted in Fig. 2.6. The training data has the complete set of input features in the 9-dimensional form for the MLP while the desired output gives the membership to the six vowel classes. The test set uses complete/ partial sets of inputs and the appropriate classification is inferred by the trained neural model. The dimension of the input vector in case of the Kohonen's net, on the other hand, is 15 (including the contextual class information).

Then, both the models have been used on the three sets (A, B, C respectively) of artificially generated nonconvex (linearly nonseparable) patterns, each set consisting of 880 pattern points. These are depicted in Figs. 4.1-4.3. The training set consists of the complete pattern vectors in the 6-dimensional form of eqn. (2.10) in case of the fuzzy MLP while the Kohonen's net uses pattern vectors in the 9-dimensional form.

5.4.1 Vowel data

Fuzzy MLP

Here we demonstrate a sample of the inferencing ability of the *trained fuzzy MLP* (with five layers having 10 nodes per hidden layer) that functions as a *knowledge base* for the vowel recognition problem. The training phase uses 50% samples from each class.

Note that this corresponds to the network configuration used in Table 2.3. The results are demonstrated in Tables 5.1-5.3.

Table 5.1 illustrates the inferred output responses of the model on a set of partial and complete input feature vectors. It is observed that often the two features F_1 and F_2 are sufficient for reaching a conclusion. This may easily be verified from the 2D representation of the vowel data in Fig. 2.6. In the table, the 2^{nd} entry corresponds to no particular vowel class and hence the certainty measure is appreciably low with both classes *e* and *i* registering *ambiguous* output membership values slightly less than 0.5. Note that F_3 = *unobtainable* is generated by eqn. (5.17). The $4a^{th}$ entry has only one accurate input value corresponding to F_1. Hence this maps to a line parallel to the F_2 axis at $F_1 = 700$ in Fig. 2.6. Note that both classes *a* and *∂* register positive belongingness, although class *a* is the *more likely* winner. On the other hand the 3^{rd} entry, with a complete feature vector, specifies a more certain decision in favour of class *a*. In entry $4b$, with a *certain* value for F_2, the decision shifts in favour of class *e*. The 5^{th} entry also possesses finite possibility of belongingness to classes *e* and *i*, as verified from the vowel diagram. However the certainty measure is indicative of the *uncertainty* in the decision. The *ambiguity* of the 6^{th} and 7^{th} entries are evident both from Fig. 2.6 as well as the two highest output membership values and the certainty measures. The $11a^{th}$ entry corresponds to a horizontal band across Fig. 2.6 around $F_1 = 350$. The classes *e* and *i*, having the two highest horizontal coverages in this region, correspond to the significant responses obtained. This may be contrasted with entry $4a$ where at least F_1 has a *definite* value 700. On the other hand, entry $11a$ correponds to a pattern point having relatively more uncertainty at all three frequency values. This results in the difficulty of decision as is evident from the value of the certainty measure. Besides, pattern class *u* (with a lower horizontal coverage *around* the *broader* band *about* 350) also does not figure among the top two significant responses. In entry $11b$, as F_2 becomes set at *high*, the response in favour of class *i* increases. However, the *ambiguity* in the decision is still evident.

In Table 5.2 we demonstrate a sample of the partial input feature combinations that are insufficient for inferring any particular decision. The *more essential* of the feature value(s) is queried for by eqns. (5.15)-(5.16). The 3^{rd} and 5^{th} entries are seen to lack *essential* information inspite of having specific values corresponding to two features. This can be explained from the ambiguity of decision (with respect to a class) observed

Table 5.1: Inferred output responses with fuzzy MLP for vowel data

Sr.	Input features			Highest output		Significant 2^{nd} choice		Certainty
No.	F_1	F_2	F_3	Class j	Membership y_j^H	Class	Membership	bel_j^H
1	300	900	*missg.*	*u*	0.89	-	-	0.88
2	*250*	1550	*unobl.*	*e*	*0.49*	*i*	*0.47*	*0.02*
3	700	1000	2600	*a*	0.89	-	-	0.89
4a	700	*missg.*	*missg.*	*a*	0.85	*ə*	0.14	0.71
4b	700	2300	*missg.*	*e*	0.77	-	-	0.66
5	450	2400	*missg.*	*e*	0.70	*i*	0.11	0.47
6	600	1200	*missg.*	*ə*	0.71	*o*	0.27	0.39
7	*low*	*very low*	*missg.*	*u*	0.48	*o*	0.35	0.10
8	*high*	*Mol low*	*missg.*	*a*	0.91	-	-	0.91
9	*between* 500 & 600	1600	*missg.*	*e*	0.75	-	-	0.72
10	*greater than* 650	*high*	*missg.*	*e*	0.75	-	-	0.60
11a	*about* 350	*missg.*	*missg.*	*e*	0.70	*i*	0.10	0.50
11b	*about* 350	*high*	*missg.*	*e*	0.65	*i*	0.34	0.31

at these pattern points in Fig. 2.6.

Table 5.3 shows the rules generated from the *knowledge base* by presenting a sample set of test patterns. The antecedent parts are obtained using eqns. (5.18)-(5.20) while the consequent parts are deduced from the values of the certainty measure bel_j^H. The rules obtained may be verified by comparing with Fig. 2.6. Note that the 5^{th}, 6^{th} and 9^{th} entries generate no justification as explained with reference to eqn. (5.18).

Fuzzy Kohonen's net

Here we demonstrate a sample of the inferencing ability of a *trained* 10×10 *fuzzy Kohonen's net* (with *cdenom* = 100) that functions as a *knowledge base* for the vowel recognition problem. Here 10% of the samples have been used during training (self organization). Note that this corresponds to the network configuration used in Table 3.7. The results are provided in Tables 5.4-5.6. Table 5.4 illustrates the inferred output response of the model on a set of partial and complete input feature vectors. Let us consider the first three entries in the table. When only F_1 is specified (entry 3), we have a horizontal line across the vowel diagram of Fig. 2.6 at this F_1 value. For this ambiguous case both classes *a* and *ə* register positive belongingness, although

Table 5.2: Querying with fuzzy MLP for vowel data

Serial No.	Input features			Query for
	F_1	F_2	F_3	
1a	700	*missing*	*missing*	F_2
1b	700	2300	*missing*	-
2a	*about* 350	*missing*	*missing*	F_2
2b	*about* 350	*high*	*missing*	-
3	400	800	*missing*	F_3
4	400	*missing*	*missing*	F_3
5	250	1550	*missing*	F_3

Table 5.3: Rules generated with fuzzy MLP for vowel data

Serial No.	Input features			Justification/ Rule generation	
	F_1	F_2	F_3	*If* clause	*Then* conclusion
1	300	900	*missing*	F_2 is *very low* and F_1 is *very low*	*very likely* class *u*
2	250	1550	*unobtainable*	F_1 is *very low* and F_2 is *Mol low*	*unable to recognize*
3	700	1000	2600	F_2 is *very low* and F_1 is *Mol high* and F_3 is *Mol high*	*very likely* class *a*
4	700	*unobtainable*	*missing*	F_1 is *Mol high*	*likely* class *a*
5	450	2400	*missing*	*no explanation*	-
6	700	2300	*missing*	*no explanation*	-
7	*high*	*Mol low*	*missing*	F_1 is *high* and F_2 is *Mol low*	*very likely* class *a*
8	*between* 500 & 600	1600	*missing*	F_2 is *very medium* and F_1 is *very medium*	*likely* class *e*
9	*greater than* 650	*high*	*missing*	*no explanation*	-
10	*about* 350	*high*	*missing*	F_2 is *high* and F_1 is *very low*	*not unlikely* class *e*

class a is the *more likely* winner. The specification of F_2 and F_3 is seen to further consolidate this decision. The 4^{th} entry corresponds to pattern class e, owing to its highest horizontal coverage along this F_1 value, although the ambiguity in the decision is evident from the value of the certainty measure. The 8^{th} entry is in favour of class a owing to its proximity to this class. The 10^{th} entry corresponds to a strip of area at $F_2 = 1600$ between $F_1 = 500$ and 600 in Fig. 2.6. This region corresponds to both classes ∂ and e and the ambiguity of the point is evident in the value of the certainty measure.

Table 5.4: Inferred output responses with fuzzy Kohonen's net for vowel data

Serial No.	Input features			First choice		Second choice		Certainty
	F_1	F_2	F_3	C_{k_1}	$\mu_{k_1}(\eta)$	C_{k_2}	$\mu_{k_2}(\eta)$	$bel_{p_1}^{k_1}$
1	700	1000	*missing*	a	0.84	a	0.81	0.80
2	700	1000	2600	a	0.83	a	0.67	0.45
3	700	*missing*	*unobtainable*	a	0.66	∂	0.38	-0.55
4	400	*unobtainable*	*missing*	e	0.59	e	0.48	0.30
5	300	900	*missing*	u	0.87	u	0.76	0.65
6	450	2400	*missing*	e	0.92	e	0.91	0.85
7	700	2300	*missing*	e	0.89	e	0.75	0.65
8	900	1400	*missing*	a	0.78	a	0.67	0.45
9	*high*	*Mol low*	*missing*	a	0.80	a	0.67	0.45
10	*between* 500 & 600	1600	*missing*	∂	0.62	e	0.48	0.30
11	*greater than* 650	*high*	*missing*	e	0.75	∂	0.37	0.65

In Table 5.5 we demonstrate a sample of the partial input feature combinations that are insufficient for inferring any particular decision. The *more essential* of the feature value(s) is queried for by eqns. (5.29)-(5.30). Table 5.6 shows the rules generated from the *knowledge base* by presenting a sample set of test patterns. The antecedent parts are obtained using eqns. (5.31)-(5.32) and (5.20) while the consequent parts are deduced from the values of the certainty measure $bel_{p_1}^{k_1}$. The rules obtained may be verified by comparing with the vowel diagram of Fig. 2.6. Note that the entries 3 and 4 generate slightly different consequent parts for a rule with the same antecedent clauses. This is because different pattern points are used to obtain the two justifications. Entries 5 and 6 are found to generate the same rules from numeric and linguistic input specifications respectively.

Table 5.5: *Querying with fuzzy Kohonen's net for vowel data*

Serial No.	Input features			Query for
	F_1	F_2	F_3	
1	700	*missing*	*missing*	F_3
2	*about* 350	*missing*	*missing*	F_2
3	400	*missing*	*missing*	F_2
4	*missing*	1000	*missing*	F_3

Table 5.6: Rules generated with fuzzy Kohonen's net for vowel data

Sr. No.	Input features			Justification/ Rule generation	
	F_1	F_2	F_3	*If* clause	*Then* conclusion
1	300	900	*missing*	F_2 is *very low*	*likely* class *u*
2	700	1000	2600	F_2 is *very low* and F_1 is *Mol high*	*more or less likely* class *a*
3	700	2300	*missing*	F_2 is *very high* and F_1 is *very medium*	*likely* class *e*
4	450	2400	*missing*	F_2 is *very high* and F_1 is *very medium*	*very likely* class *e*
5	900	1400	*missing*	F_2 is *Mol low* and F_1 is *very high*	*more or less likely* class *a*
6	*high*	*Mol low*	*missing*	F_2 is *Mol low* and F_1 is *high*	*more or less likely* class *a*
7	*between* 500 & 600	1600	*missing*	F_2 is *very medium* and F_1 is *very medium*	*not unlikely* class *o*
8	*greater than* 650	*high*	*missing*	F_1 is *very medium* and F_2 is *high*	*likely* class *e*

5.4.2 Artificially generated data

Next, the fuzzy MLP and the fuzzy Kohonen's net were trained on the three sets of nonconvex patterns (Sets A, B and C), depicted in Figs. 4.1-4.3 respectively, in succession. Two nonseparable pattern classes 1 and 2 were considered in each case. The region of *no pattern points* was modeled as class *none (no class)*.

Fuzzy MLP

In Tables 5.7, 5.10 and 5.13 we demonstrate the inferred output responses of a five-layered model (with 10 nodes per hidden layer and trained with $perc = 50$) on some partial and complete input feature vectors for the three pattern sets. The classification performance of the network on the three pattern sets has been provided in Tables 4.1-4.3. We choose these networks to demonstrate the effectiveness of our algorithm even for the complex five-layered configurations. The querying phase is demonstrated by Tables 5.8, 5.11 and 5.14 while Tables 5.9, 5.12 and 5.15 illustrate the generation of a few rules from the above-mentioned three *knowledge bases*. Verification regarding these tables may be made by examining the original patterns given in Figs. 4.1-4.3. The *disjunctive (Or)* terms in the antecedent parts are obtained by combining the various conjunctive clauses generated for the *same* feature corresponding to a single rule (produced to justify a single inferred decision). These disjunctive clauses result due to the concave and/or disjoint nature of the pattern class(es).

The 1^{st}, 5^{th}, 6^{th} and 7^{th} entries in Table 5.7 correspond to horizontal bands across Fig. 4.1 around the regions of the given F_1 values. Class 1, having the highest horizontal coverage at $F_1 = low$ in entry 1, produces a *significant* response. As F_1 changes from *medium* (entry 5) to *very medium* (entry 6), the response for class 2 increases and the corresponding ambiguity in the certainty factor decreases. With $F_1 = Mol\ medium$ (in entry 7) class *none* produces a relatively larger response, although the result is *more ambiguous* as observed from the certainty measure. Comparing the 2^{nd} or 8^{th} entries with the 3^{rd} entry we note that as F_1 changes from *very low* or *not medium* to *Mol low*, the response for class 1 decreases in favour of that of class *none*. All results of Tables 5.7-5.9 may be verified by comparing with Fig. 4.1 that depicts Pattern Set A. We note from Table 5.9 that entries 1, 5 and 8 generate *no justification* as explained earlier with reference to eqn. (5.18).

Table 5.7: Inferred output responses for Pattern Set A with fuzzy MLP

Serial No.	Input features		Highest output		Significant 2^{nd} choice		Certainty
	F_1	F_2	Class j	Membership y_j^H	Class	Membership	bel_j^H
1	*low*	*missing*	1	1.0	-	-	0.99
2	*very low*	*low*	1	1.0	-	-	1.0
3	*Mol low*	*low*	*none*	1.0	2	0.02	0.99
4	*not low*	*missing*	*none*	1.0	-	-	-
	not low	*low*	1	0.97	2	0.02	0.95
5	*medium*	*missing*	2	0.83	*none*	0.16	0.66
6	*very medium*	*missing*	2	0.96	*none*	0.01	0.95
7	*Mol medium*	*missing*	*none*	0.68	2	0.41	0.27
8	*not medium*	*low*	1	1.0	-	-	1.0
9	*low*	*high*	1	1.0	2	0.02	0.97
10	*medium*	*medium*	*none*	0.93	2	0.13	0.73
11	*high*	*high*	1	1.0	2	0.03	0.97

Table 5.8: Querying for Pattern Set A with fuzzy MLP

Serial No.	Input features		Query for
	F_1	F_2	
1	*low*	*missing*	-
2	*very low*	*missing*	F_2
3	*Mol low*	*missing*	F_2
4	*not low*	*missing*	F_2
5	*medium*	*missing*	-
6	*very medium*	*missing*	-
7	*Mol medium*	*missing*	-
8	*not medium*	*missing*	F_2
9	*high*	*missing*	-
10	*very high*	*missing*	F_2
11	*Mol high*	*missing*	F_2

Table 5.9: Rules generated for Pattern Set A with fuzzy MLP

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*missing*	*no explanation*	-
2	*Mol low*	*low*	F_1 is *very medium* and F_2 is *low* or *very medium*	*very likely no class*
3	*medium*	*missing*	F_1 is *medium*	*likely* class 2
4	*very medium*	*missing*	F_1 is *very medium*	*very likely* class 2
5	*not medium*	*low*	*no explanation*	-
6	*Mol medium*	*low*	F_1 is *Mol medium* or *Mol high* and F_2 is *very medium*	*very likely no class*
7	*Mol high*	*low*	F_1 is *very medium* or *Mol high* and F_2 is *very medium*	*very likely no class*
8	*low*	*high*	*no explanation*	-
9	*medium*	*medium*	F_1 is *medium* and F_2 *is medium*	*likely no class*
10	*high*	*medium*	F_2 is *medium* and F_1 is *high* or *very medium*	*very likely no class*
11	*Mol medium*	*missing*	F_1 *is Mol medium* or *not high*	*not unlikely no class*

Table 5.10: Inferred output responses for Pattern Set B with fuzzy MLP

Serial No.	Input features		Highest output		Significant 2^{nd} choice		Certainty
	F_1	F_2	Class j	Membership y_j^H	Class	Membership	bel_j^H
1	*low*	*missing*	*none*	1.0	-	-	1.0
2	*very low*	*missing*	1	0.60	*none*	0.41	0.19
	very low	*low*	*none*	1.0	-	-	1.0
3	*Mol low*	*low*	2	1.0	-	-	1.0
4	*medium*	*missing*	1	0.67	*none*	0.34	0.32
5	*Mol medium*	*missing*	*none*	0.72	1	0.32	0.4
6	*not medium*	*missing*	2	0.73	*none*	0.27	0.47
	not medium	*low*	2	1.0	-	-	1.0
7	*Mol high*	*missing*	1	1.0	-	-	1.0
8	*not high*	*missing*	*none*	0.8	1	0.2	0.62
	not high	*low*	*none*	1.0	2	0.01	0.99
9	*low*	*high*	*none*	1.0	-	-	1.0
10	*medium*	*medium*	*none*	1.0	-	-	1.0

In Table 5.10, the 1^{st}, 4^{th}, 5^{th} and 7^{th} entries correspond to horizontal bands across Fig. 4.2 showing Pattern Set B. Class *none*, having the largest horizontal coverage at $F_1 = low$ in entry 1, produces a *significant* response. Note that entry 4 (with $F_1 = medium$ and inferring class 1) and entry 5 (with $F_1 = Mol\ medium$ and inferring class *none*) denote *ambiguous* decisions as observed from the certainty measure. However

Table 5.11: Querying for Pattern Set B with fuzzy MLP

Serial No.	Input features		Query for
	F_1	F_2	
1	*very low*	*missing*	F_2
2	*Mol low*	*missing*	F_2
3	*medium*	*missing*	-
4	*Mol medium*	*missing*	-
5	*not medium*	*missing*	F_2
6	*very high*	*missing*	F_2
7	*Mol high*	*missing*	-
8	*not high*	*missing*	F_2

entry 7 with $F_1 = Mol\ high$ produces a *more definite* response in favour of class 1. As F_2 becomes *known* as *low* in entry 2, the response changes from class 1 to class *none*. This is because of the fact that along the horizontal band at $F_1 = very\ low$, class 1 has the largest horizontal coverage. However when the smaller region of interest is specified at $F_2 = low$, the decision shifts in favour of class *none* and the *ambiguity* in decision decreases drastically as the certainty increases ($bel_j^H = 1$ here). In case of entries 6 and 8 the corresponding responses in favour of classes 2 and *none* become *more certain* as F_2 becomes specified. All results of Tables 5.10-5.12 may be verified by comparing with Fig. 4.2. Note that in Table 5.12, entries 2 and 4 generate *no justification.*

In Table 5.13, entries 1, 2 and 5 correspond to horizontal bands across Fig. 4.3 showing Pattern Set C. The 1^{st} and 5^{th} entries, for $F_1 = not\ low$ and *very high* respectively, generate comparatively *less certain* decisions in favour of class 1. Entry 2 with $F_1 = medium$ produces a *decisive* response in favour of class 2. As F_2 becomes *known* as *low* in entry 3, the response changes from class *none* to class 2 as the region

Table 5.12: Rules generated for Pattern Set B with fuzzy MLP

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*Mol high*	*missing*	F_1 is *Mol high* or *very medium*	*very likely* class 1
2	*medium*	*missing*	*no explanation*	-
3	*medium*	*low*	F_1 is *medium* or *Mol high* and F_2 is *very medium*	*Mol likely* class 1
4	*Mol medium*	*missing*	*no explanation*	-
5	*medium*	*high*	F_1 is *medium* and F_2 is *high* or *very medium*	*very likely no class*
6	*high*	*medium*	F_2 is *medium* or *Mol high* and F_1 is *high*	*Mol likely no class*

Table 5.13: Inferred output responses for Pattern Set C with fuzzy MLP

Serial No.	Input features		Highest output		Significant 2^{nd} choice		Certainty
	F_1	F_2	Class j	Membership y_j^H	Class	Membership	bel_j^H
1	*not low*	*missing*	1	0.81	*none*	0.19	0.61
2	*medium*	*missing*	2	0.92	*none*	0.07	0.93
3	*Mol medium*	*missing*	*none*	0.97	2	0.07	0.90
	Mol medium	*low*	2	0.88	*none*	0.11	0.77
4	*not medium*	*low*	1	0.82	*none*	0.18	0.63
5	*very high*	*missing*	1	0.82	*none*	0.17	0.64
6	*medium*	*high*	2	1.0	-	-	1.0

Table 5.14: Querying for Pattern Set C with fuzzy MLP

Serial No.	Input features		Query for
	F_1	F_2	
1	*low*	*missing*	F_2
2	*very low*	*missing*	F_2
3	*not low*	*missing*	-
4	*medium*	*missing*	-
5	*Mol medium*	*missing*	F_2
6	*not medium*	*missing*	F_2
7	*high*	*missing*	F_2
8	*very high*	*missing*	-
9	*not high*	*missing*	F_2

of interest becomes more localised. But the *ambiguity* in decision is observed to be more in the case of the complete input specification. All results of Tables 5.13-5.15 may be verified by comparing with Fig. 4.3.

Fuzzy Kohonen's net

While a neural net array of size 14×14 has provided the best performance on Pattern Set A, the patterns B and C have been best classified by using neural arrays of size 16×16 (with 50% of the data used as training set in each case). Note that these correspond to the results of Tables 4.7-4.9.

In Tables 5.16, 5.19 and 5.22 we demonstrate the inferred output responses of the fuzzy Kohonen's net on some partial and complete input feature vectors for the three pattern sets. Tables 5.17, 5.20 and 5.23 show the querying phase, where in some cases the missing feature information is *necessary* for inferring a decision and hence queried for. Tables 5.18, 5.21 and 5.24 illustrate the generation of a few rules from the three *knowledge bases*. Verification regarding these tables may be made by examining the original patterns given in Figs. 4.1-4.3.

In Table 5.16 entries 2, 3 and 5 correspond to horizontal bands across Fig. 4.1,

Table 5.15: Rules generated for Pattern Set C with fuzzy MLP

Serial	Input features		Justification/ Rule generation	
No.	F_1	F_2	*If* clause	*Then* conclusion
1	*very low*	*low*	F_1 is *very low* and F_2 is *low* or *very medium*	*very likely no class*
2	*not low*	*missing*	F_1 is *very high*	*likely* class 1
3	*medium*	*missing*	F_1 is *medium* or *Mol low*	*very likely* class 2
4	*Mol medium*	*low*	F_1 is *Mol medium* or *Mol low* and F_2 is *low*	*likely* class 2
5	*not medium*	*low*	F_2 is *low* or *very medium*	*likely* class 1
6	*high*	*low*	F_1 is *high* and F_2 is *low*	*very likely no class*
7	*medium*	*high*	F_1 is *medium* or *Mol low* and F_2 is *high*	*very likely* class 2

Table 5.16: Inferred output responses for Pattern Set A with fuzzy Kohonen's net

Serial	Input features		First choice		Second choice		Certainty
No.	F_1	F_2	C_{k_1}	$\mu_{k_1}(\eta)$	C_{k_2}	$\mu_{k_2}(\eta)$	$bel_{p_1}^{k_1}$
1	*missing*	*low*	2	0.90	2	0.77	0.79
2	*not low*	*missing*	*none*	0.99	*none*	0.99	0.99
3	*Mol medium*	*missing*	*none*	0.93	*none*	0.90	0.79
4	*not medium*	*low*	1	0.90	1	0.59	0.17
5	*very high*	*missing*	*none*	0.99	*none*	0.99	0.99
6	*low*	*low*	*none*	0.94	*none*	0.82	0.88
7	*medium*	*low*	2	0.90	2	0.77	0.79
8	*medium*	*medium*	*none*	0.93	*none*	0.90	0.79
9	*medium*	*high*	2	0.96	2	0.82	0.91

Table 5.17: Querying for Pattern Set A with fuzzy Kohonen's net

Serial No.	Input features		Query for
	F_1	F_2	
1	*not low*	*missing*	-
2	*Mol medium*	*missing*	-
3	*not medium*	*missing*	F_2
4	*high*	*missing*	-

depicting Pattern Set A, around the given F_1 values. Here entry 3 produces a relatively *less certain* decision, as compared to the remaining entries, in favour of class *none*. Entry 1 maps to a vertical band around $F_2 = low$ and infers belongingness to class 2. Comparing entries 4, 6 and 7 we observe that $F_1 = not\ medium$ (entry 4) produces an *ambiguous* decision in favour of class 1. However, $F_1 = low$ (entry 6) and $F_1 = medium$ (entry 7) generate *more certain* decisions in favour of classes *none* and 2 respectively. All results of Tables 5.16-5.18 may be verified from Fig. 4.1. Note that in Table 5.18 the

Table 5.18: Rules generated for Pattern Set A with fuzzy Kohonen's net

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*very low*	*missing*	F_1 is *very low*	*very likely no class*
2	*missing*	*low*	F_2 is *low*	*likely* class 2
3	*not low*	*missing*	F_1 is *very high*	*very likely no class*
4	*Mol medium*	*missing*	F_1 is *Mol medium*	*likely no class*
5	*not medium*	*low*	F_2 is *low*	*not unlikely* class 1
6	*very high*	*missing*	F_1 is *very high*	*very likely no class*
7	*medium*	*low*	F_1 is *medium* and F_2 is *low*	*likely* class 2

entries 2 and 5 correspond to the same antecedent clause generating two consequent parts as justifications to two separate inferences, obtained from entries 1 and 4 in

Table 5.16. Both entries have the same F_2 value and hence can be used to generate separate but individually valid rules, although with different certainty measures.

Table 5.19: Inferred output responses for Pattern Set B with fuzzy Kohonen's net

Serial No.	Input features		First choice		Second choice		Certainty
	F_1	F_2	C_{k_1}	$\mu_{k_1}(\eta)$	C_{k_2}	$\mu_{k_2}(\eta)$	$bel_{p_1}^{k_1}$ or $bel_{p_1}^{k_2}$
1	*Mol low*	*missing*	*none*	0.81	2	0.65	(0.30)
2	*medium*	*missing*	*none*	0.78	*none*	0.67	0.56
3	*Mol high*	*missing*	1	0.99	1	0.99	0.99
4	*low*	*medium*	2	0.86	2	0.82	0.73
5	*medium*	*medium*	*none*	0.78	*none*	0.67	0.34
6	*high*	*high*	1	0.89	*none*	0.77	0.54

Entries 1, 2 and 3 in Table 5.19 correspond to horizontal bands across Fig. 4.2, illustrating Pattern Set B, around the given F_1 values. The certainty value in brackets (entry 1) indicates belief more in favour of the decision of second choice for class C_{k_2} as compared to class C_{k_1}. This is obtained from the connection weight values as given by eqn. (5.28). In the 1^{st} entry the decision is *ambiguous* and in favour of class 2, as observed from the certainty measure (although class *none* generates the highest response followed by class 2). As F_1 changes to *medium* (in entry 2), the decision becomes *more certain* in favour of class *none* while for F_1 = *Mol high* (entry 3), the decision is *certainly* in favour of class 1. Entry 5, with F_1 = *medium*, is *less certain* in inferring class *none* as compared to entry 4, with F_1 = *low* (both with F_2 = *medium*). The latter yields a *less ambiguous* decision (as observed from the certainty measure) in favour of class 2. Note that the value of the certainty measures, and not the output membership values, determine the *ambiguity* in a decision. Entry 6 gives a *more or less ambiguous* decision in favour of class 1 while also producing a significant response for class *none*. All results of Tables 5.19-5.21 may be verified from Fig. 4.2.

Entries 1, 2, 3, 4, 6 and 7 in Table 5.22 correspond to horizontal bands across Fig. 4.3, illustrating Pattern Set C, around the given F_1 values. Among these, entries 1, 3, 6 and 7 (with F_1 = *low, not low, high* and *not high* respectively) generate *certain* decisions in favour of class 1. However entries 4 and 2 (with F_1 = *medium*

Table 5.20: Querying for Pattern Set B with fuzzy Kohonen's net

Serial No.	Input features		Query for
	F_1	F_2	
1	*medium*	*missing*	-
2	*not medium*	*missing*	F_2
3	*Mol high*	*missing*	-
4	*not high*	*missing*	-

Table 5.21: Rules generated for Pattern Set B with fuzzy Kohonen's net

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*Mol low*	*missing*	F_1 is *very medium*	*not unlikely* class 2
2	*medium*	*missing*	F_1 is *medium*	*not unlikely no class*
3	*not medium*	*low*	F_2 is *low*	*not unlikely no class*
4	*Mol high*	*missing*	F_1 is *Mol high*	*very likely* class 1
5	*low*	*medium*	F_2 is *medium* and F_1 is *low*	*likely* class 2
6	*medium*	*low*	F_1 is *medium* and F_2 is *low*	*very likely no class*
7	*medium*	*medium*	F_1 is *medium* and F_2 is *medium*	*not unlikely no class*
8	*high*	*high*	F_2 is *high* and F_1 is *high*	*Mol likely* class 1

Table 5.22: Inferred output responses for Pattern Set C with fuzzy Kohonen's net

Serial No.	Input features		First choice		Second choice		Certainty
	F_1	F_2	C_{k_1}	$\mu_{k_1}(\eta)$	C_{k_2}	$\mu_{k_2}(\eta)$	$bel_{p_1}^{k_1}$
1	*low*	*missing*	1	0.93	1	0.59	0.87
2	*Mol low*	*missing*	*none*	0.67	*none*	0.54	0.07
3	*not low*	*missing*	1	0.99	1	0.98	0.97
4	*medium*	*missing*	1	0.60	1	0.56	0.11
5	*not medium*	*low*	1	1.0	1	0.99	1.0
6	*high*	*missing*	1	0.99	1	0.98	0.97
7	*not high*	*missing*	1	0.94	1	0.58	0.87
8	*low*	*low*	*none*	0.97	*none*	0.76	0.95
9	*low*	*medium*	1	0.94	*none*	0.53	0.87
10	*medium*	*medium*	1	0.95	1	0.60	0.20
11	*high*	*low*	*none*	0.98	*none*	0.95	0.97
12	*high*	*medium*	1	0.99	1	0.98	0.97

and *Mol low* respectively) produce *rather ambiguous* decisions (low values of certainty measure) in favour of classes 1 and *none* respectively. Comparing entries 5, 8 and 11, we find that entries 8 and 11 (with F_1 = *low* and *high* respectively) generate decisions in favour of class *none* while entry 5 (with F_1 = *not medium*) produces a decision in favour of class 1. From the 9^{th}, 10^{th} and 12^{th} entries we observe that entry 12 (F_1 = *high*) produces the *most certain* decision in favour of class 1 with entry 9 (F_1 = *low*) following close behind. On the other hand, entry 10 (with F_1 = *medium*) produces a *more ambiguous (less certain)* decision for class 1. All results of Tables 5.22-5.24

Table 5.23: Querying for Pattern Set C with fuzzy Kohonen's net

Serial No.	Input features		Query for
	F_1	F_2	
1	*low*	*missing*	-
2	*Mol low*	*missing*	-
3	*medium*	*missing*	-
4	*not medium*	*missing*	F_2
5	*high*	*missing*	-
6	*not high*	*missing*	-

may be verified from Fig. 4.3. In Table 5.24, entry 2 (corresponding to the 2^{nd} entry in Table 5.22) is unable to infer any positive decision (*unable to recognize*) due to the extremely low certainty measure generated in this case.

5.5 Conclusions and discussion

In this chapter we have developed methodologies, based on fuzzy MLP and fuzzy Kohonen's net, to demonstrate the utility of these models for inferencing and rule generation. The *trained* connection weights are found to constitute the *knowledge base* for the application domain under consideration. The system is capable of handling uncertainty and/or impreciseness in the input representation provided in quantitative, linguistic and/or set forms. The output decision is inferred in terms of membership values to one or more output categories. In case of partial inputs, the system can query

Table 5.24: Rules generated for Pattern Set C' with fuzzy Kohonen's net

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*missing*	F_1 is *low*	*very likely* class 1
2	*Mol low*	*missing*	F_1 is *Mol low*	*unable to recognize*
3	*not low*	*missing*	F_1 is *very high*	*very likely* class 1
4	*medium*	*missing*	F_1 is *medium*	*Mol likely* class 1
5	*not medium*	*low*	F_2 is *low*	*very likely* class 1
6	*high*	*missing*	F_1 is *high*	*very likely* class 1
7	*not high*	*missing*	F_1 is *very low*	*very likely* class 1
8	*low*	*low*	F_1 is *low* and F_2 is *low*	*very likely no class*
9	*low*	*medium*	F_2 is *medium* and F_1 is *low*	*very likely* class 1
10	*low*	*high*	F_1 is *low* and F_2 is *high*	*very likely no class*
11	*medium*	*low*	F_1 is *medium* and F_2 is *low*	*likely no class*
12	*medium*	*medium*	F_2 is *medium* and F_1 is *medium*	*not unlikely* class 1
13	*high*	*low*	F_1 is *high* and F_2 is *low*	*very likely no class*
14	*high*	*medium*	F_2 is *medium* and F_1 is *high*	*very likely* class 1

the user for the *more essential* feature information. Justification for a decision reached is provided to the user/ expert in terms of rules whose antecedent and consequent parts are in linguistic and *natural* forms. The magnitudes of the connection weights of the *trained* net are used in every stage of the inferencing procedure. The aforesaid characteristics of the system have been demonstrated on both the vowel data and the artificially generated nonconvex pattern sets. A rigorous application of the fuzzy MLP-based model for medical diagnosis has been provided in Chapter 7.

The antecedent parts of the rules are generated by backtracking along the maximum-weighted connection paths of the trained network. The consequent part is determined from the certainty measure which expresses the confidence (belief) of an output decision. The node excitations corresponding to a test pattern determine the appropriate *If-Then* parts of a rule generated to justify an inferred decision. Note that this investigation provides a basic module for designing a classification type connectionist expert system. The rules thus obtained can also constitute the knowledge base of a traditional expert system in the same application domain.

The rules generated for the vowel data have been observed to be more complete and/or accurate as compared to that for the three sets of synthetic data. This may be verified by comparing the rules in each case with the corresponding figures. However, it is to be noted that the nonconvex pattern sets under consideration are quite complex because of the *difficult* nature of the problem of class separability in these cases. This accounts for the relatively better performance of both the models on the vowel data.

As in Chapters 2-4, the performance (inferencing and rule generation capability) of the fully supervised fuzzy MLP has been observed here to be better than that of the partially supervised fuzzy Kohonen's net. In the following chapters our investigation will be concerned only with the fuzzy MLP and its variations.

Chapter 6

Use of Logical Neuron in Fuzzy MLP for Rule Generation and Inferencing

6.1 Introduction

The fuzzy MLP, described in Chapter 2 for pattern classification and in Chapter 5 for rule generation and inferencing, employed fuzziness only at the input and output stages of the network. In this chapter we consider a different version of this model that incorporates fuzziness also at the neuronal level [277, 281, 282]. The backpropagation algorithm is appropriately modified to model the logical operators *And* and *Or* used in place of the conventional weighted sum and sigmoidal function. The construction, classificatory behaviour, inferencing and rule generation aspects of the model are all presented in succession.

The model broadly performs two main tasks. First we construct the three-layered fuzzy logical network for classifying multi-class patterns. Next, the trained network is used to *generate* rules. The model is now capable of inferring the output decision for complete and/or partial inputs along with a certainty measure, querying the user for the *more essential* missing input information and providing justification for any conclusion. If asked by the user, the network is capable of justifying its decision in rule form (in terms of the salient features) with the antecedent and consequent parts produced in linguistic and *natural* terms. Various fuzzy implication operators are used to introduce different amounts of interaction during error backpropagation. The incorporation of fuzzy logic at the input, output and neuronal levels enables the model to handle uncertainty at the various stages. The relational structures, realized by *max-min* or *min-max* and *product-probabilistic sum* operators, introduced into the fuzzy model help in classifying patterns that exhibit a logical structure.

As mentioned in Section 1.5.1 one may note that the two-layered logical model of Pedrycz [222] used a *crisp* implication operator and dealt with a two-class problem. The work was extended in [216, 217, 223] to handle multi-class problems involving relational equations, using a different performance index, reference neurons, and the *Lukasiewicz* implication operator. We, on the other hand, study the behaviour of the three-layered fuzzy MLP with several fuzzy implication operators (for error propagation), using alternate *And-Or* operations, to solve multiclass problems. Our input and output vector representations are fuzzy and the mode of varying the learning rate is also different. The more interactive operators like the product and probabilistic sum are also studied. Another approach using logic neurons by Watanabe *et al.* [224] also

considered *min-max* operations but with two kinds of weight vectors and a different scheme of backpropagation for a three-layered network.

The effectiveness of the network, described here, is demonstrated on the speech data and on some artificially generated patterns. A comparison is made with the performance of the fuzzy MLP (having no logical operators) of Chapters 2 and 5 and its conventional nonfuzzy counterpart. Effects of fuzzification at the input as well as the output of the fuzzy logical model are also investigated.

6.2 Fuzzy version of the MLP using logical operators

Let us now describe the fuzzy logical model which has been used for both classification and rule generation. The input and output vector representations being analogous to those of the fuzzy MLP, already described in Chapter 2, will not be discussed here again. Hence only the portion of the algorithm dealing with the incorporation of fuzziness at the neuronal level will be explained in this chapter. The model consists of logical neurons employing conjugate pairs of *t-norms* T and *t-conorms* S in place of the *weighted sum* and *sigmoid* functions of the conventional MLP. The *back propagation* algorithm is modified to incorporate the logical operations in the error derivative term.

6.2.1 T-norm and T-conorm

We consider two special cases of the conjugate pair of *t-norm* T and *t-conorm* S [299], *viz., min max* and *product probabilistic sum* operators to represent the *And* and *Or* nodes at the hidden and output layers respectively. At the *And* nodes in the hidden layer we use $T(a,b) = a \wedge b$ while at the *Or* nodes in the output layer we have $S(a,b) = a \vee b$. The *min* T^m and *max* S^m operators are defined as

$$T^m(a,b) = \min(a,b)$$
$$S^m(a,b) = \max(a,b) \tag{6.1}$$

while the *product* T^p and *probabilistic sum* S^p operators are given as

$$T^p(a,b) = ab$$

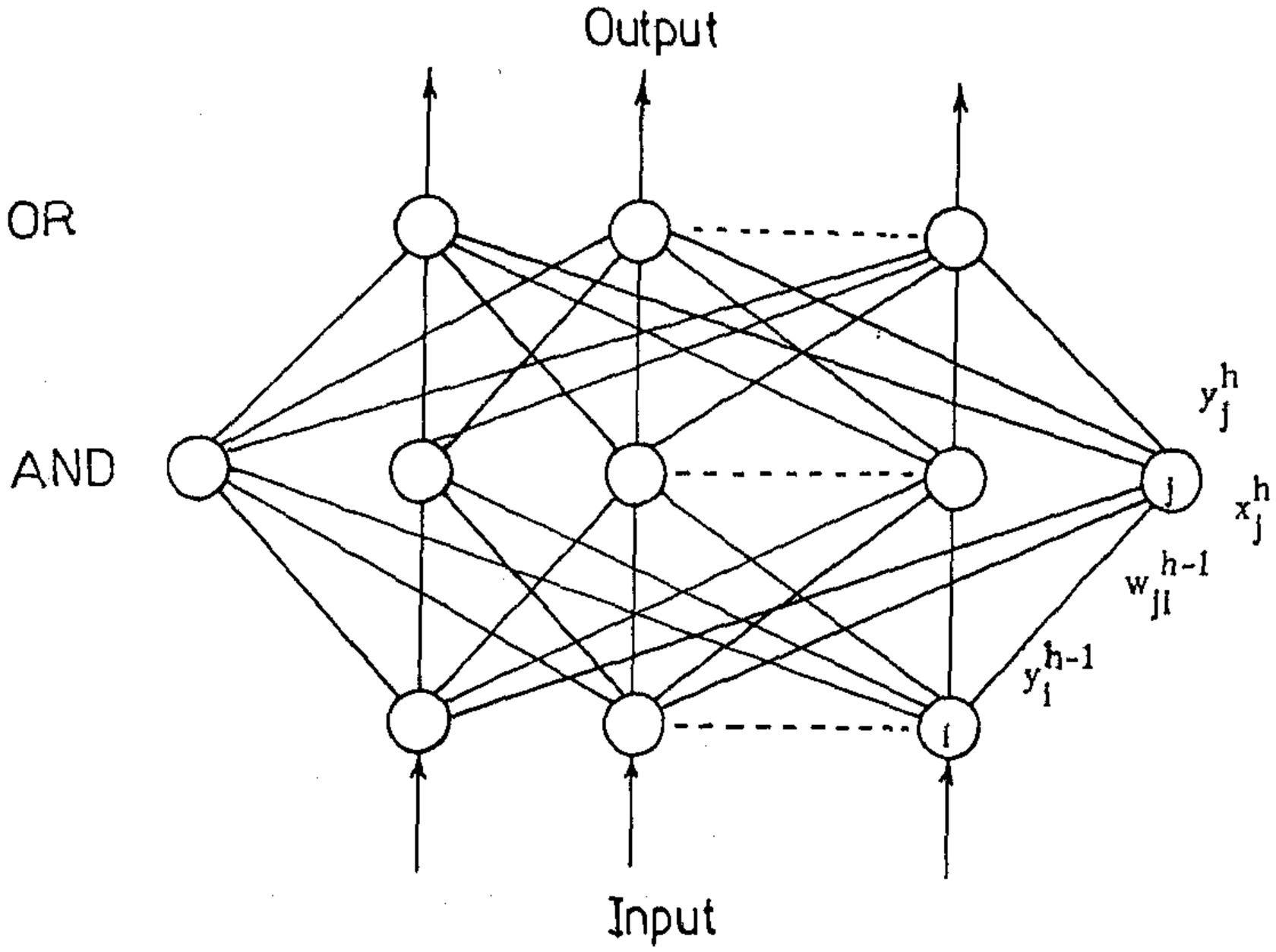

Figure 6.1: Three-layered logical MLP

$$S^p(a,b) = a + b - ab \tag{6.2}$$

such that $T^p(a,b) \leq T^m(a,b) < S^m(a,b) \leq S^p(a,b)$ and (T^p, S^p), (T^m, S^m) are the two special cases of the conjugate pair (T,S).

Let us consider the three-layered model as shown in Fig. 6.1 with $H = 2$, such that there are n_1 and n_2 neurons in the layers $h = 0$ and 1 respectively. The output y_j^1 of the j^{th} neuron in the first layer is defined as

$$y_j^1 = T\left[S\left(y_1^0, w_{j1}^0\right), \ldots, S\left(y_{n_1}^0, w_{jn_1}^0\right)\right] \tag{6.3}$$

Here the input vector is in the $3n$-dimensional space of eqn. (2.10), y_i^0 is the output of neuron i in the input layer as defined by eqn. (2.3), w_{ji}^0 denotes the corresponding connection weight and $i = 1, \ldots, n_1$. The T operation is performed over all n_1 S operation outputs corresponding to the neurons in the input layer.

Analogously, the output y_k^2 in the second layer is given as

$$y_k^2 = S\left[T\left(y_1^1, w_{k1}^1\right), \ldots, T\left(y_{n_2}^1, w_{kn_2}^1\right)\right] \tag{6.4}$$

where y_j^1 (for $j = 1, \ldots, n_2$) is given by eqn. (6.3) and the S operation is performed over all n_2 T operation outputs corresponding to the neurons in the first layer.

Note that for the *probabilistic sum* operator S^p, the output is computed iteratively as

$$y_k^2 = S_{n-1}^p$$

such that

$$S_{j+1}^p = a_{j+2} + S_j^p - a_{j+2} S_j^p \tag{6.5}$$

for $j = 1, 2, \ldots, n-2$, where

$$\begin{aligned} S_1^p &= a_1 + a_2 - a_1 a_2 \\ a_j &= y_j^1 w_{kj}^1 \end{aligned} \tag{6.6}$$

6.2.2 Fuzzy implication

It is to be mentioned that the pair T^p and S^p of eqn. (6.2) are interactive and their results depend on the values of both the arguments. However the lattice operations T^m and S^m of eqn. (6.1) are completely noninteractive. Hence we use various implication operators to introduce different amounts of interaction during backpropagation of errors with the T^m and S^m operations.

For two variables X and Y, we have the *crisp* implication operator defined as

$$X \rightarrow Y = \|X \subset Y\| = \begin{cases} 1 & \text{if } X \leq Y \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

This is observed to cause either *activity* or *disactivity* depending upon the individual conditions. To alleviate this problem, we introduce some interaction between the two variables using fuzzy concepts. A few fuzzy implication operators [21] used to incorporate a degree of the amount of inclusion or containment are given below.

- Lukasiewicz

$$\|X \subset Y\| = T^m(1, 1 - X + Y) \tag{6.8}$$

- Kleene-Dienes-Lukasiewicz

$$\|X \subset Y\| = 1 - X + XY \tag{6.9}$$

- Kleene-Dienes

$$\|X \subset Y\| = S^m(1 - X, Y) \tag{6.10}$$

- Early Zadeh

$$\|X \subset Y\| = S^m(T^m(X,Y),(1-X)) \tag{6.11}$$

where the S^m and T^m operators are as defined in eqn. (6.1). It is to be noted that these operators introduce various degrees of mutual interaction between the two variables and hence their choice may be application dependent.

6.2.3 The backpropagation algorithm based on logical operations

Use of logical operators in place of the more conventional *weighted sum* and *sigmoid* functions of the MLP necessitates modification of the derivatives involved in the traditional backpropagation scheme. The LMS error E of eqn. (2.4) is minimized by the gradient-descent technique of eqn. (2.5) with the restriction that $0 \leq w_{ji}^h \leq 1$ for $0 \leq h \leq 2$. This is done by first truncating each weight update $\Delta w_{ji}^h(t)$ so that $0 \leq |\Delta w_{ji}^h(t)| \leq 0.1$ and then truncating the corresponding w_{ji}^h values. The error derivative $\frac{\partial E}{\partial w_{ji}}$ is computed as

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j^h}{\partial w_{ji}} \tag{6.12}$$

For the output layer ($h = H$) we substitute

$$\frac{\partial E}{\partial y_j} = y_j^H - d_j \tag{6.13}$$

In case of the hidden layer we use

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k^{h+1}}{\partial y_j} = \sum_k \left(y_k^H - d_k\right)\frac{\partial y_k^{h+1}}{\partial y_j} \tag{6.14}$$

where neurons j and k lie in layers h and $h+1$ respectively.

In order to evaluate the derivative $\frac{\partial y_j^h}{\partial w_{ji}}$ of eqn. (6.12), for $h > 0$, let us define

$$sm^h = T\left[S\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, S\left(y_{i-1}^{h-1}, w_{j(i-1)}^{h-1}\right), S\left(y_{i+1}^{h-1}, w_{j(i+1)}^{h-1}\right), \ldots, S\left(y_{n_1}^{h-1}, w_{jn_1}^{h-1}\right)\right]$$

if $h = 1$, and

$$sm^h = S\left[T\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, T\left(y_{i-1}^{h-1}, w_{j(i-1)}^{h-1}\right), T\left(y_{i+1}^{h-1}, w_{j(i+1)}^{h-1}\right), \ldots, T\left(y_{n_2}^{h-1}, w_{jn_2}^{h-1}\right)\right] \tag{6.15}$$

otherwise, where the $T(S)$ operation at layer h is performed over all $l = 1, \ldots, i-1, i+1, \ldots, n_1(n_2)$ $S(T)$ operation outputs from the neurons in the preceding layer $h-1$, provided $l \neq i$, for $h = 1(2)$ respectively; also let

$$sm_i^h = \begin{cases} S\left(y_i^{h-1}, w_{ji}^{h-1}\right) & \text{if } h = 1 \\ T\left(y_i^{h-1}, w_{ji}^{h-1}\right) & \text{otherwise} \end{cases} \tag{6.16}$$

Using eqns. (6.3)-(6.4) and (6.15)-(6.16), we have

$$\frac{\partial y_j^h}{\partial w_{ji}} = \begin{cases} \frac{\partial}{\partial w_{ji}}\left[T\left(sm^h, sm_i^h\right)\right] & \text{if } h = 1 \\ \frac{\partial}{\partial w_{ji}}\left[S\left(sm^h, sm_i^h\right)\right] & \text{otherwise} \end{cases} \tag{6.17}$$

where the *t-norm* T and *t-conorm* S are given by either of eqns. (6.1) or (6.2) in order to model the logical operators *And* and *Or*.

The max and min operators

Using any of the implication operators from eqns. (6.7)-(6.11) and eqns. (6.15)-(6.16) in eqn. (6.17), we compute the derivative $\frac{\partial y_j^h}{\partial w_{ji}}$ for the conjugate pair (T^m, S^m) as

$$\frac{\partial y_j^h}{\partial w_{ji}} = \begin{cases} \|w_{ji}^{h-1} \subset y_i^{h-1}\| * \|sm^h \subset sm_i^h\| & \text{if } h = 2 \\ \|y_i^{h-1} \subset w_{ji}^{h-1}\| * \|sm_i^h \subset sm^h\| & \text{otherwise} \end{cases} \tag{6.18}$$

where $h > 0$. Similarly, the sensitivity measure $\frac{\partial y_k^h}{\partial y_j}$ from the h^{th} layer by eqn. (6.14), for $h = 2$, is evaluated as

$$\frac{\partial y_k^h}{\partial y_j} = \|y_k^{h-1} \subset w_{jk}^{h-1}\| * \|sm^h \subset sm_k^h\| \tag{6.19}$$

where sm^h and sm_k^h are given by eqns. (6.15)-(6.16) with k substituted for i.

The product and probabilistic sum operators

On the other hand, for the conjugate pair (T^p, S^p) using eqns. (6.2), (6.5)-(6.6) and (6.15)-(6.17), we have

$$\frac{\partial y_j^h}{\partial w_{ji}} = \begin{cases} (1 - sm^h) y_i^{h-1} & \text{if } h = 2 \\ sm^h (1 - y_i^{h-1}) & \text{otherwise} \end{cases} \tag{6.20}$$

Analogously, we compute the sensitivity measure as

$$\frac{\partial y_k^h}{\partial y_j} = (1 - sm^h) w_{jk}^{h-1} \tag{6.21}$$

Substituting the values of $\frac{\partial y_j^h}{\partial w_{ji}}$ and $\frac{\partial y_k^h}{\partial y_j}$ from eqns. (6.18)-(6.19) or eqns. (6.20)-(6.21), as the case may be, into eqns. (6.12) and (6.14) enables one to evaluate the error derivative $\frac{\partial E}{\partial w_{ji}}$ of eqn. (2.5) and thereby update the connection weights during training. This constitutes the *back propagation* algorithm for a network incorporating logical nodes.

6.3 Rule generation from the trained logical neural net

After the design and training of the network is complete, it is expected to be able to infer the correct classification for the test data. Handling of imprecise inputs is possible and natural decision is obtained associated with a certainty measure denoting the confidence in the decision. The procedure for rule generation and inferencing in the fuzzy MLP (involving no logical operators) has been discussed in Chapter 5. Here we describe this aspect for the logical model. The basic procedure is the same as depicted in the block diagram of Fig. 5.1. The input for a test pattern can be in quantitative, linguistic or set forms or a combination of these. This is similar to the description provided in Section 5.2.1. It is represented in the form of membership values to the three primary linguistic properties *low*, *medium* and *high* as in eqn. (2.10), modeled as π-functions.

6.3.1 Forward pass

Associated with each neuron j in layer $h+1$ are the variables $conf_j^{h+1}$, $unknown_j^{h+1}$ and $known_j^{h+1}$. For neuron j in layer $h > 0$ we define

$$unknown_j^h = \begin{cases} T\left[S\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, S\left(y_{n_1}^{h-1}, w_{jn_1}^{h-1}\right)\right] & \text{for } h = 1 \\ S\left[T\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, T\left(y_{n_2}^{h-1}, w_{jn_2}^{h-1}\right)\right] & \text{otherwise} \end{cases} \tag{6.22}$$

and

$$unden_j^h = \sum_i w_{ji}^{h-1} \tag{6.23}$$

for all i having $noinf_i^{h-1} = 1$, where $i = 1, \ldots, n_1(n_2)$ for $h = 1(2)$, and

$$known_j^h = \begin{cases} T\left[S\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, S\left(y_{n_1}^{h-1}, w_{jn_1}^{h-1}\right)\right] & \text{for } h = 1 \\ S\left[T\left(y_1^{h-1}, w_{j1}^{h-1}\right), \ldots, T\left(y_{n_2}^{h-1}, w_{jn_2}^{h-1}\right)\right] & \text{otherwise} \end{cases} \tag{6.24}$$

for all i with $noinf_i^{h-1} = 0$, where T and S stand for the conjugate pair of *t-norm* and *t-conorm* defined in eqns. (6.1)-(6.2). Here the $T(S)$ operation at layer h is performed over all $n_1(n_2)$ $S(T)$ operation outputs from the neurons in the preceding layer $h-1$ for $h = 1(2)$ respectively. This is done keeping in mind the *And-Or* structure of the nodes in the hidden and output layers as given by eqns. (6.3)-(6.4). For $h > 0$ we have

$$noinf_j^h = \begin{cases} 1 & \text{if } known_j^h > unknown_j^h \quad \text{for } h = 1 \\ 1 & \text{if } known_j^h < unknown_j^h \quad \text{for } h = 2 \\ 0 & \text{otherwise} \end{cases} \tag{6.25}$$

Note that the relations in eqn. (6.25) are opposite for $h = 1$ and 2. This is because we apply the T and S operators in opposite sequences for the two layers in eqns. (6.22) and (6.24) while deriving $unknown_j^h$ and $known_j^h$ respectively.

Using eqns. (6.3)-(6.4) and (6.22)-(6.25), we define

$$conf_j^h = \begin{cases} \frac{y_j^h}{unden_j^h} & \text{if } noinf_j^h = 1 \text{ and } h > 0 \\ y_j^h & \text{otherwise} \end{cases} \tag{6.26}$$

A certainty measure (for each output neuron) is defined as

$$cert_j^H = \frac{y_j^H}{\sum_i y_i^H} \tag{6.27}$$

where $0 \leq cert_j^H \leq 1$. Depending on the value of $cert_j^H$, the final inferred output may be given in natural form irrespective of whether the input is fuzzy/ deterministic and complete/ partial.

6.3.2 Querying

If there is any neuron j in the output layer H with $noinf_j^H = 1$ by eqn. (6.25), we begin the querying phase. We select the *unknown* output neuron j_1 from among the neurons with $noinf_j^H = 1$ such that $conf_{j_1}^H$ by eqn. (6.26) (among them) is maximum.

We select $i = i_1$ such that with $noinf_i^h = 1$, for $0 \leq h < H-1$, we have

$$\begin{aligned} S\left(w_{j_1 i_1}^h, y_{i_1}^h\right) &= \min_i \left[S\left(w_{j_1 i}^h, y_i^h\right)\right] \quad \text{for } h = 0 \\ T\left(w_{j_1 i_1}^h, y_{i_1}^h\right) &= \max_i \left[T\left(w_{j_1 i}^h, y_i^h\right)\right] \quad \text{otherwise} \end{aligned} \tag{6.28}$$

For node i_1 in the input layer ($h = 0$), the model queries the user for the value of the corresponding input feature u_1.

6.3.3 Justification

The antecedent and consequent parts of the justificatory rules are obtained by an algorithm analogous to that described earlier in Section 5.2.4 for the fuzzy MLP (with no logical operators). The clauses are conjunctive when the corresponding generated paths bifurcate at layer 1 nodes and are disjunctive when the said bifurcation occurs at layer 2 nodes. This follows from the *And-Or* structure of the network given by eqns. (6.3)-(6.4). The selected clauses correspond to the *salient* input features (determined both by the connection weight magnitudes learned during training as well as the input feature components of the test pattern under consideration).

6.4 Implementation and results

Three-layered fuzzy logical MLP has been used on the vowel data and on four sets (H, X, $X1$, L) of artificially generated patterns. Note that the logical model, based on *And* and *Or* operations, has been developed using a single hidden layer. Therefore, the implementation reported here refers to three-layered models only. The vowel data consists of a set of the 871 patterns as depicted in Fig. 2.6. Each of the four sets (H, X, $X1$, L) of artificially generated pattern classes consist of 880 pattern points in the feature space $F_1 - F_2$. These are depicted in Figs. 6.2-6.5. Note that the region of *no pattern points* has been modeled as the class *none (no class)*. The pattern set H consists of 6 classes, the sets X and $X1$ possess 3 classes while set L is made up of 2 classes. The input is represented in the $3n$-dimensional linguistic form of eqn. (2.10). The network has been trained using $perc\%$ samples from each pattern class of the data set.

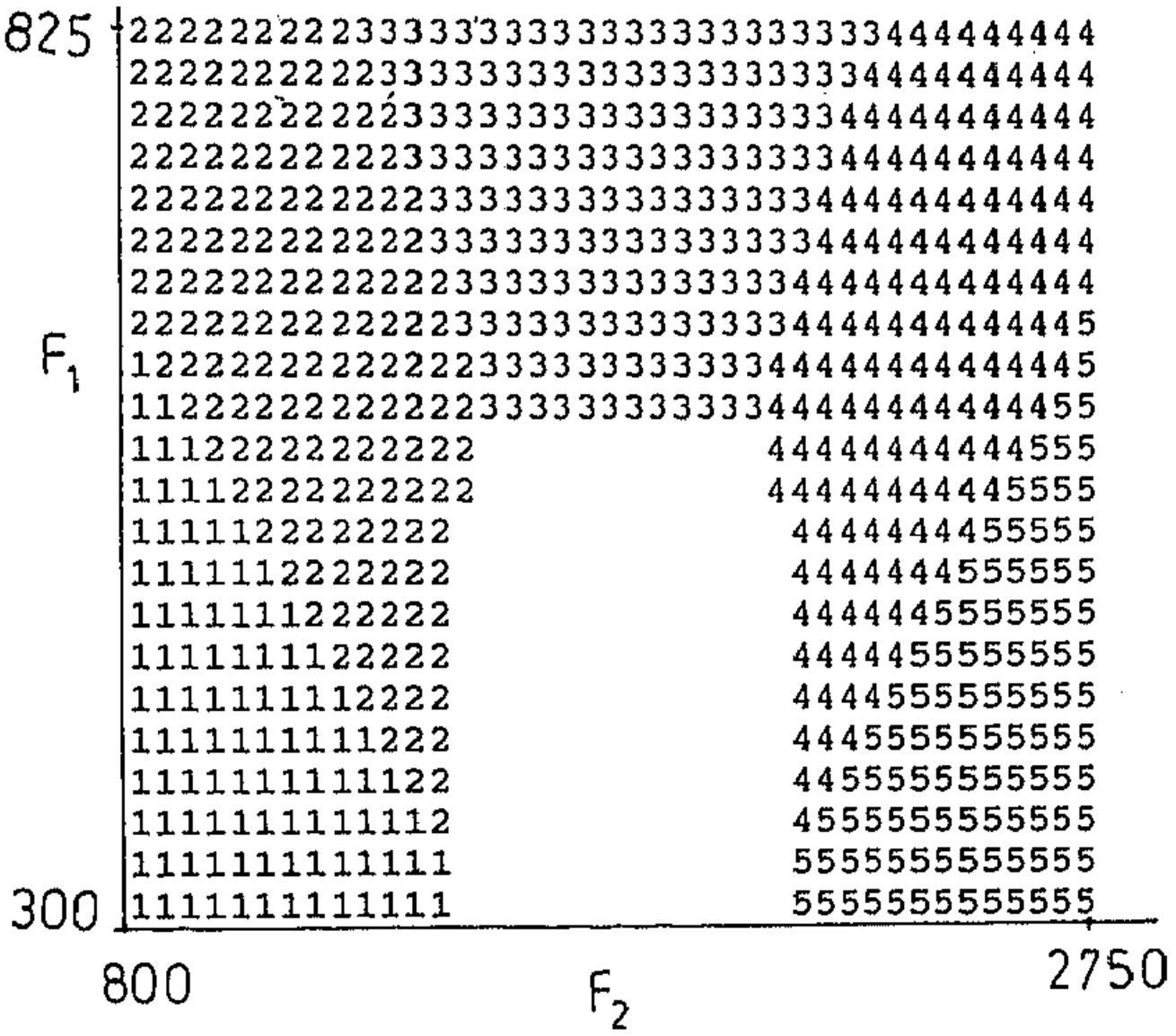

Figure 6.2: Pattern Set H in the $F_1 - F_2$ plane

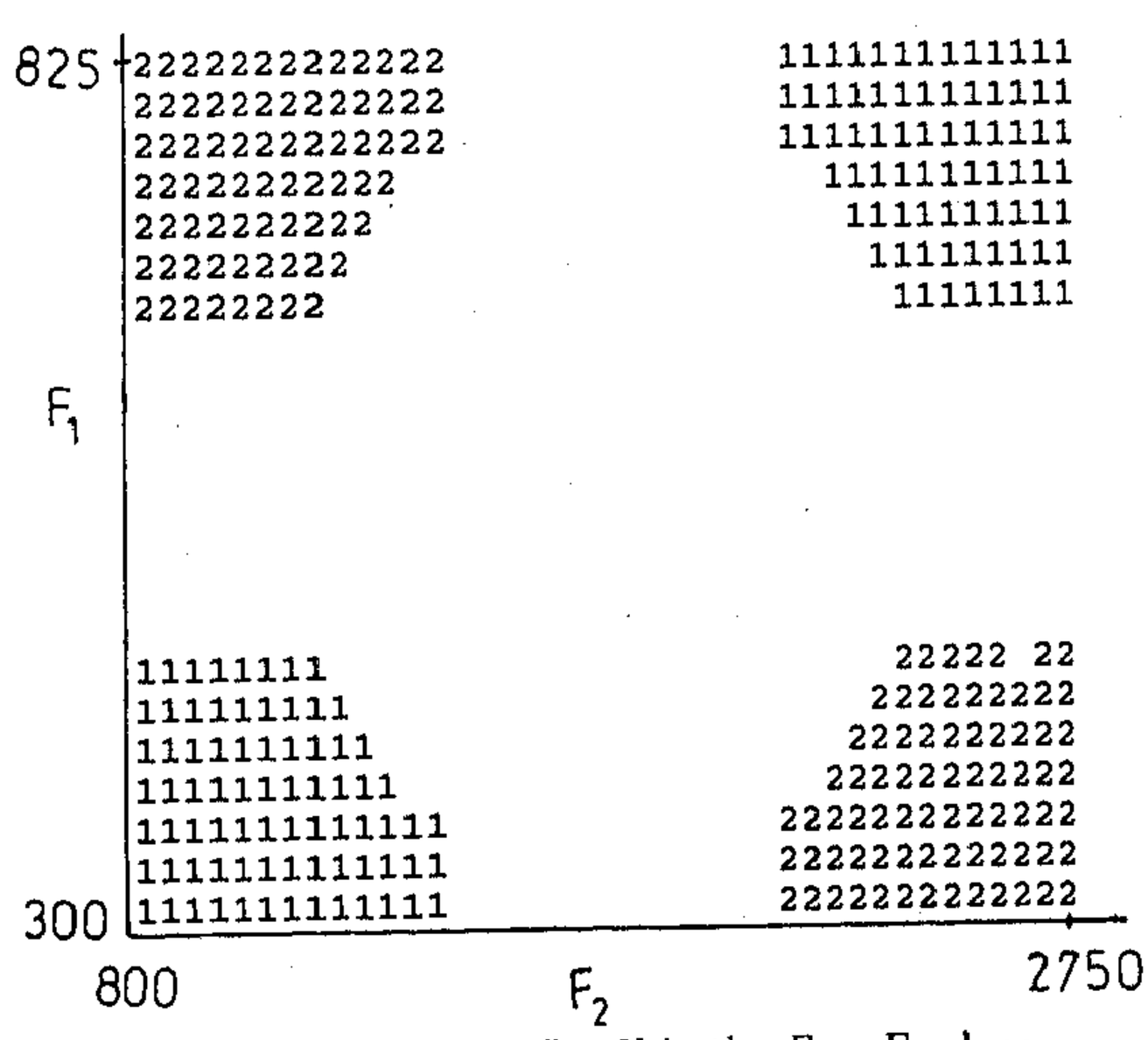

Figure 6.3: Pattern Set X in the $F_1 - F_2$ plane

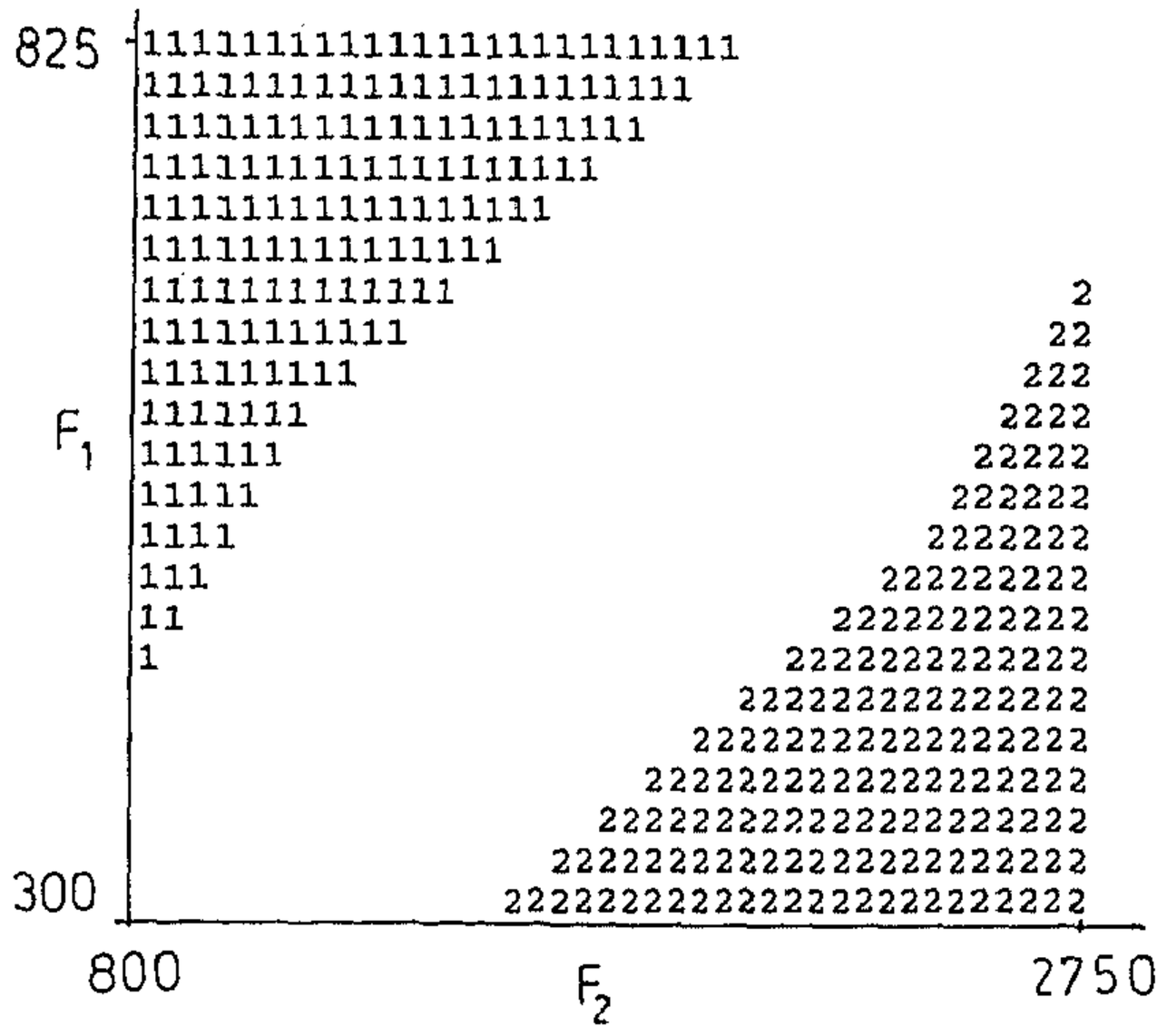

Figure 6.4: Pattern Set $X1$ in the $F_1 - F_2$ plane

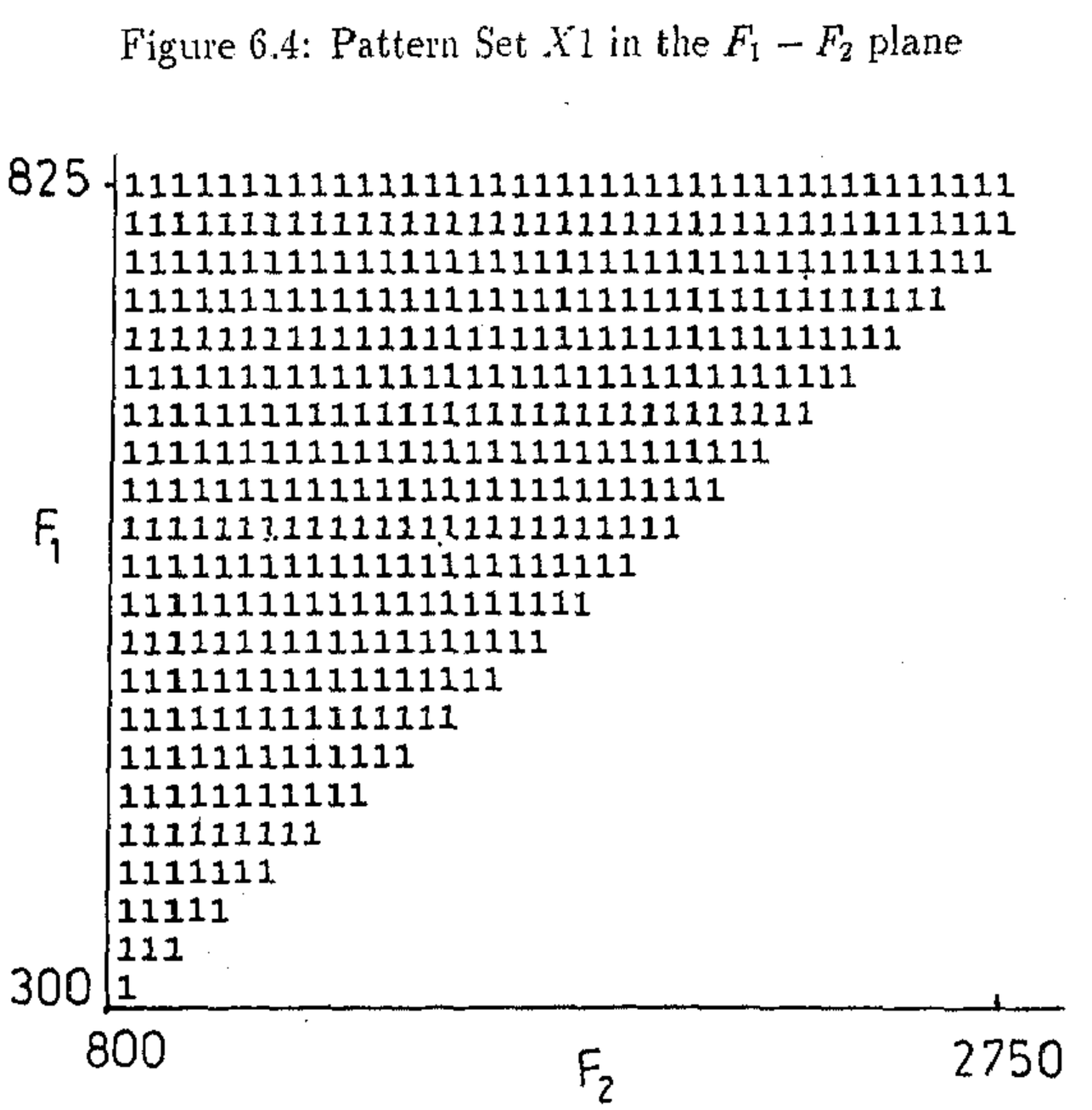

Figure 6.5: Pattern Set L in the $F_1 - F_2$ plane

6.4.1 Classification

Here *mse*, *perfect match p* and *best match b* refer to the training set while mse_t and *overall best score t* are indicative of the test set (remaining $(100 - perc)\%$ samples). Besides the criterion for b is stricter than that for t in the sense that t denotes simply the best match while b indicates the best match provided that $y_j^H > 0.5$ when $d_j > 0.5$. Hence sometimes we observe t to be greater than b. Note that H, L, KDL, KD and EZ refer to the *Crisp, Lukasiewicz, Kleene-Dienes-Lukasiewicz, Kleene-Dienes* and *Early Zadeh* operators respectively by eqns. (6.7)-(6.11) (used during backpropagation of error) for the conjugate pair (T^m, S^m) of eqn. (6.1) while P refers to the logical operators (T^p, S^p) of eqn. (6.2). The performance of the fuzzy MLP (with no logical operators) of Chapter 2 is given under *fuzzy (O)* while the conventional MLP (with inputs in the range [0,1] and *crisp* output decision) is denoted as *hard (O')*. The effect of fuzzification (both at the input and the output) has been investigated in detail (using *fully fuzzified* output representation) for the vowel data only, as the synthetic data has crisp output values.

Vowel data

Table 6.1 compares the performance of the logical model and the fuzzy version (with no logical operators) on the vowel data using one hidden layer with $m = 20, 22$ hidden nodes (corresponding to three-layered configurations yielding best results) and $perc = 10$. We have used $f_d = 5$ and $f_e = 1$ in eqn.(2.16) for output membership computation. The fuzzy model O gives good overall performance. Note that the traditional hard version of the MLP provided very poor results and hence the details are not included here. This is perhaps because the three-layered *hard* MLP is unable to classify the given vowel data having fuzzy class separation. The use of linguistic input features (incorporating more local information of the feature space) and fuzzy output membership values alleviates this problem in the other models.

Effect of fuzzification

Table 6.2 demonstrates the effect on the performance of the networks using a *fully fuzzified* output representation. Three-layered nets with m hidden nodes are used, with a training set size of $perc = 10$. In this case the j^{th} desired output d_j, as expressed by eqn. (2.19), consists of l nonzero components for an l-class problem domain. This

Table 6.1: Performance of models KDL, P and O on vowel data

Model used		Logical				Fuzzy	
		P		KDL		O	
Hidden nodes m		20	22	20	22	20	22
perfect p (%)		3.6	8.3	0.0	0.0	22.4	0.0
best b (%)		74.2	74.2	20.0	0.0	91.8	87.1
mse		.036	.031	.077	.084	.011	.027
T	∂ (%)	31.2	24.6	9.3	1.5	44.6	69.8
e	a (%)	76.5	93.8	97.5	96.3	65.4	72.8
s	i (%)	90.2	92.2	96.7	98.7	79.2	81.8
t	u (%)	80.0	93.5	85.9	100.	88.3	85.9
s	ϵ (%)	75.1	62.3	15.8	0.0	75.0	75.0
e	o (%)	86.5	77.9	3.0	0.0	85.7	87.2
t	Overall t (%)	77.8	77.1	48.8	47.3	76.8	80.1
	mse_t	.042	.047	.072	.081	.046	.04

is unlike the scheme of model O of Chapter 2 or Table 6.1, that involves one or more *nonzero* components corresponding to the training set samples. Note that now the desired fuzzy output vector d is dependent on the class means and hence on the training set selected. Therefore, a higher value of the *perfect match* p is more indicative of good performance as compared to the maximum-activation-related *best match* b which is more dependent on the choice of the training patterns. This fully fuzzified output representation is used here in Tables 6.2-6.4.

It is observed from Table 6.2 that the model KDL fares poorer as compared to the model P. This is because of the fact that the max and min operations of KDL cause the different *input-connection weight* combinations to be less interactive, with only the maximum or minimum term(s) controlling the neuron activation(s). The *product* and *probabilistic sum* operators, being more *co-operative*, yield better results. Because of the same reason, if we replace the logical operator by the original sigmoidal function the results (shown as model O) improve. The use of logical operators seem to cause deterioration in the classification efficiency of the neural model, perhaps due to the inherent loss of some information. It may also be noted that the generalization ability on the test set for the fully fuzzified version seems to be better in case of models O and P. The *perfect match* is also found to be significantly higher for the fully fuzzified case of model O.

In order to investigate the utility of the linguistic input and output class membership representations, experiments have also been done using the conventional MLP with one hidden layer, inputs in the range [0,1] and *crisp* output. The results were found to be very poor. The incorporation of logical operations (in the conventional model) have further worsened the performance. Incorporation of fuzziness in the input and output representations was seen to improve the performance of the conventional MLP.

Effects of fuzzification at the input and output have been investigated in detail using the three-layered logical model P (having fully fuzzified output representation) with $perc = 10$ and $m = 20$ hidden nodes (as this configuration yielded the best generalization performance in Table 6.2). The results are provided in Tables 6.3-6.4. The amount of overlapping between the linguistic properties *low*, *medium* and *high* have been varied by altering the radius of the π-function corresponding to the linguistic set *medium* (as explained earlier in Section 4.3.5 (using $fnos$) with reference to the fuzzy

Table 6.2: Performance of the fully fuzzified models on vowel data

Model		P				KDL				O			
$m =$		18	19	20	21	18	19	20	21	18	19	20	21
perf p		21.2	28.3	34.2	25.9	3.6	2.4	1.2	1.2	70.6	82.4	70.6	67.1
best b		76.5	84.7	78.9	75.3	16.5	28.3	18.9	28.3	77.7	78.9	82.4	84.7
mse		.01	.006	.007	.008	.033	.032	.039	.034	.002	.001	.002	.002
T	∂	60.0	66.1	76.8	32.3	10.9	46.3	100.	67.6	78.1	82.7	85.3	95.1
e	a	93.0	91.5	46.5	98.8	73.9	0.0	0.0	0.0	74.4	97.9	99.0	77.4
s	i	86.9	98.1	100.	89.9	56.5	83.5	99.2	55.9	92.5	100.	100.	98.4
t	u	89.3	99.2	65.9	92.1	64.7	87.9	69.1	45.4	95.5	88.2	99.4	95.9
s	e	92.5	81.0	85.3	89.9	40.4	60.3	40.9	74.2	93.2	95.3	92.1	91.9
e	o	72.6	92.0	91.3	99.3	47.2	10.3	0.0	48.9	96.5	92.8	100.	95.2
t	t	84.7	88.1	84.1	90.5	51.4	50.1	48.7	52.4	90.2	93.7	96.5	92.8
	mse_t	.01	.009	.009	.01	.034	.034	.037	.035	.004	.003	.004	.007

MLP with no logical operators). It may be seen from Table 6.3 that, in general, there is no pronounced change in the output performance with variation in input overlapping (determined by $fnos$). However, it is observed that very large amount of overlapping among the linguistic properties of the input feature lead to poorer recognition scores. It is also noticed that very small overlapping (for low values of the multiplicative factor $fnos$) sometimes results in poor performance. Note that in Tables 6.2-6.3 we have used $f_d = 2$ and $f_e = 2$ for computing the output class membership values by eqn. (2.16), while $fnos = 1$ in Tables 6.2 and 6.4.

Table 6.3: Effect of fuzzification at input with model P for vowel data

$fnos =$		0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3
perfect p (%)		10.6	15.3	18.9	22.4	30.6	34.2	34.2	22.4	25.9
best b (%)		71.8	71.8	75.3	80.0	77.7	78.9	77.7	74.2	75.3
mse		.011	.011	.009	.009	.008	.007	.008	.01	.009
T	∂ (%)	84.2	74.7	78.9	82.1	73.6	76.8	76.8	77.9	76.8
e	a (%)	34.8	39.5	37.2	55.8	46.5	46.5	55.8	16.2	16.2
s	i (%)	94.2	94.2	98.5	99.2	99.2	100.	100.	98.5	97.8
t	u (%)	69.1	69.1	65.9	65.9	65.9	65.9	82.9	82.9	84.0
s	e (%)	82.4	84.4	91.7	87.3	87.3	85.3	87.3	86.8	87.8
e	o (%)	89.9	86.1	91.3	92.3	91.3	91.3	91.3	90.9	90.9
t	Overall t (%)	82.5	81.1	85.2	85.8	84.1	84.1	87.1	84.6	84.7
	mse_t	.014	.013	.012	.01	.009	.009	.009	.01	.009

Table 6.4 demonstrates the variation in performance of the above-mentioned logical model P with different amounts of fuzziness at the output. Here f_d and f_e of eqn. (2.16) (corresponding to the output membership evaluation scheme which is similar to that discussed for the fuzzy MLP with no logical operators) are varied as shown. It is to be noted that higher values of f_e and lower values of f_d help in enhancing the contrast among the output membership values and generally lead to a higher recognition score. However too high contrast (very low f_d) results in poor performance. The logical model P of Table 6.4 (with fully fuzzified output representation) has good overall performance (better than model P with the same network architecture in Table 6.1) for $f_d = 2$ with $f_e = 2$ and 4, and $f_d = 3$ with $f_e = 4$ (that correspond to high

Table 6.4: Effect of fuzzification at output with model P for vowel data

$f_d =$		2			3		4		5		
$f_e =$		1	2	4	2	4	2	4	1	2	4
perfect p (%)		34.2	34.2	25.9	29.5	27.1	55.3	40.0	43.6	60.0	50.6
best b (%)		69.5	78.9	80.0	83.6	84.7	85.9	70.6	71.8	71.8	61.2
mse		.006	.007	.014	.008	.01	.005	.007	.004	.004	.006
T	∂ (%)	66.3	76.8	78.9	77.9	82.1	82.1	67.3	87.3	72.6	54.7
e	a (%)	51.1	46.5	51.1	0.0	37.2	2.3	72.1	0.0	53.4	65.1
s	i (%)	99.2	100.	100.	97.1	90.7	93.5	85.0	94.2	97.1	88.5
t	u (%)	48.9	65.9	82.9	53.2	63.8	52.1	24.4	0.0	1.0	0.0
s	e (%)	87.3	85.3	84.4	90.7	93.6	93.6	95.6	88.7	92.6	99.5
e	o (%)	77.0	91.3	90.4	91.3	89.4	93.3	94.7	92.8	93.3	76.5
t	Overall t (%)	77.6	84.1	86.1	81.0	83.9	82.1	80.2	75.2	78.1	72.2
	mse_t	.007	.009	.015	.009	.014	.005	.009	.004	.003	.006

contrast). It is revealed under investigation that this choice of parameter values enables the membership function curves of the various pattern classes to represent the dynamic range of the given pattern points more suitably. The classwise recognition score in most other cases of f_d-f_e combinations is seen to be rather poor, especially in the cases of classes a and u.

Artificially generated pattern sets

Four sets (H, X, $X1$, L) of artificially generated pattern classes have been used to evaluate the classificatory performance of the fuzzy logical model. We have used the three-layered logical version of the neural net with $perc = 10$ and m nodes in the hidden layer. Both the conjugate pairs (T^p, S^p) and (T^m, S^m) have been applied in the process. A comparison in performance has also been made with the fuzzy model O (with no logical operators) and its hard counterpart O' (conventional MLP). Note that only one network configuration, corresponding to the best overall performance of the fuzzy logical model, has been selected for models O and O'. The main objective of this study is to demonstrate the classificatory performance of the fuzzy logical model. It

Table 6.5: Performance of models P, O and O' on Pattern Set H

Model used		Logical				Fuzzy	Hard
		P				O	O'
Hidden nodes m		10	11	12	13	12	12
perfect p (%)		1.2	0.0	2.4	7.0	65.2	0.0
best b (%)		80.3	83.8	88.4	87.2	100.0	98.9
mse		.046	.042	.039	.038	.004	.017
T	1 (%)	72.3	84.0	85.1	76.6	76.6	95.7
e	2 (%)	91.0	81.4	81.4	84.6	89.1	94.2
s	3 (%)	81.5	84.2	83.5	86.9	96.5	86.3
t	4 (%)	83.8	81.9	83.2	83.2	92.9	80.0
s	5 (%)	83.1	83.1	81.0	83.1	86.3	100.0
e	6 (%)	85.8	84.4	91.9	87.8	91.2	91.2
t	Overall (%)	83.7	83.1	84.5	84.2	89.8	90.3
	mse_t	.059	.058	.057	.062	.026	.025

is observed that model O is always better, perhaps due to the use of the conventional sigmoidal nonlinearities at the neuronal level. Tables 6.5-6.8 depict the performance of the MLP on the four pattern sets. Note that the output vector in this case is nonfuzzy with components $d_j = \{0, 1\}$ in eqn. (2.19) (corresponding to the output representation of the fuzzy MLP with no logical operators).

It is observed from Tables 6.5-6.6 that the conjugate pair (T^p, S^p) (model P) is capable of modeling the non-linear decision surfaces of *Pattern Sets* H and X. However the more conventional fuzzy model O yields an overall better performance for the same number of nodes in the hidden layer. This is especially evident from the high values of *perfect match p* and low values of *mse*. The *hard* model O' also provides good recognition scores. Table 6.7 demonstrates that models O, O', P, KDL and L can handle *Pattern Set* $X1$, albeit with decreasing orders of efficiency. From Table 6.8 we find that the simple two-class *Pattern Set* L can be handled by the neural net models O, O', P, KDL, L, H, KD and EZ in decreasing orders of performance. Note that Tables 6.7-6.8 provide the results for $m = 10$ hidden nodes only, as this network configuration yields the best overall scores.

Table 6.6: Performance of models P, O and O' on Pattern Set X

Model used		Logical P										Fuzzy	Hard
		Prod-Sum					*Sum-Sum*					O	O'
Hidden nodes m		8	9	10	11	12	8	9	10	11	12	10	10
perfect p (%)		2.3	1.2	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	79.3	16.1
best b (%)		83.9	85.1	88.5	86.2	79.3	65.6	65.6	65.6	65.6	65.6	100.0	93.1
mse		.065	.058	.053	.059	.069	.095	.08	.097	.078	.089	.003	.043
T	1 (%)	83.4	84.1	89.9	89.9	97.1	33.1	68.3	31.6	76.2	73.3	96.4	87.7
e	2 (%)	86.2	86.9	87.7	93.4	86.2	60.8	63.7	18.8	76.8	66.6	94.2	86.2
s	*none* (%)	98.4	98.2	95.5	96.1	95.7	100.0	100.0	100.0	100.0	100.0	94.0	89.9
t	Overall (%)	93.7	93.8	93.2	94.5	94.3	81.4	88.1	73.9	91.8	89.5	94.4	88.9
	mse_t	.074	.074	.078	.073	.075	.11	.092	.112	.084	.09	.028	.062

Table 6.7: Performance of models KDL, L, P, O and O' on Pattern Set $X1$

Model used		Logical				Fuzzy	Hard
		(T^m, S^m)		(T^p, S^p)			
		KDL *Min-Max*	L *Min-Max*	P *Prod-Sum*	P *Sum-Sum*	O	O'
perfect p (%)		17.3	8.1	29.9	0.0	43.7	1.2
best b (%)		94.3	94.3	92.0	79.4	100.0	100.0
mse		.074	.076	.045	.097	.007	.011
T	1 (%)	84.6	84.6	84.6	74.8	85.2	84.0
e	2 (%)	85.9	85.9	84.6	43.5	95.7	93.2
s	*none* (%)	91.6	91.6	93.8	96.1	92.7	97.0
t	Overall (%)	89.0	89.0	90.0	80.9	91.8	93.5
	mse_t	.083	.085	.06	.115	.045	.028

We observe that the more interactive logical model P gives the best overall performance on the artificially generated pattern sets as compared to the logical models using the conjugate pair (T^m, S^m). Note that *Pattern Sets* X and H cannot be suitably modeled by the latter. However as the decision surface becomes simpler through *Pattern Sets* $X1$, L, the less interactive conjugate pair (T^m, S^m) is also able to model the pattern classes using the different implication operators (involving various degrees of interaction among the arguments during error propagation). The more conventional fuzzy version O (with the more general sigmoidal neurons) is found to yield a slightly superior performance, usually over the training sets. The hard model O' generally gives a lower value of p and higher value of mse as compared to model O. Otherwise the performance of O' is comparable and sometimes superior as compared to model O. This is perhaps because the given pattern sets (especially L, $X1$ and H) can be suitably modeled in the n-dimensional feature space of model O' and do not require the larger amount of local information available in the 3^n fuzzy subregions generated in the input space of model O.

Table 6.8: Performance of models L, H, KD, EZ, KDL, P, O, O' on Pattern Set L

Model used		Logical									Fuzzy	Hard
		(T^m, S^m)							(T^p, S^p)		O	O'
		KDL			L	H	KD	EZ	P			
		MinMax	*MaxMax*	*MinMin*	*Min-Max*				*PrdSum*	*SumSum*		
perfect p (%)		46.6	0.0	25.0	46.6	46.6	13.7	0.0	47.8	42.1	70.5	8.0
best b (%)		88.7	88.7	88.7	86.4	86.4	76.2	59.1	86.4	89.8	100.0	100.0
mse		.081	.098	.096	.08	.082	.091	.146	.062	.077	.011	.014
T	mse_t	.136	.16	.163	.13	.157	.14	.169	.086	.103	.024	.005
e	1 (%)	73.3	73.3	73.3	73.3	73.3	73.3	99.5	92.8	84.7	96.5	98.3
s	2 (%)	78.6	78.6	78.6	78.6	78.6	78.6	56.4	84.4	85.8	96.9	100.0
t	Net	75.6	75.7	75.7	75.6	75.6	75.6	79.8	88.9	85.2	96.6	99.1

It is to be mentioned that the *Sum-Sum* version for model P in Tables 6.6-6.8 and the *Max-Max* versions for model KDL in Table 6.8 refer to the cases where both layers 1 and 2 use the *Or* neurons while the *Min-Min* version for model KDL in Table 6.8 refers to the case involving only *And* neurons in both the layers. The other logical models use *And* neurons in layer 1 followed by *Or* neurons in layer 2, as given by eqns. (6.1)-(6.2). This *And-Or* combination of logical neurons is found to result in a better classificatory performance, compared to the *And-And* and *Or-Or* architectures. Note that the *And-*

Or and *Or-And* combination of neurons are equivalent in Boolean logic, resulting in expressions in *Sum-of-Products* and *Product-of-Sums* forms respectively.

6.4.2 Rule generation

In this phase, complete/partial sets of inputs are clamped at the input layer and the appropriate classification is inferred by the trained neural model. A measure of certainty is used and querying regarding unknown input feature values resorted to in case of some partial input sets. Justification in *If-Then* rule form, regarding a conclusion, is also obtained when desired.

The logical models lead to the generation of more appropriate rules in *And-Or* form (expressed as disjunction of conjunctive clauses) from the *learned knowledge base* embedded among the connection weights as compared to the more conventional fuzzy counterpart (O) (with no logical operators) of Chapter 5. This is because of the built-in *And-Or* structure incorporated into the network for the fuzzy logical model. On the other hand, the neurons of the fuzzy model O with sigmoidal nonlinearities approximate the *And* or *Or* functions in special cases depending on the values of the thresholds learned. Hence the clauses of the rules in *And-Or* forms cannot be generated with comparable efficiency. The details regarding the intermediate stages involving the inferred output response, querying and rule generation have been reported in detail for the vowel data only as an illustration. The comparison of the rules generated by the various models P, KDL and O are also provided.

Vowel data

Here we demonstrate the inferencing ability of a *trained* neural model (with one hidden layer having 20 nodes, corresponding to Table 6.1) that functions as a *knowledge base* for the vowel recognition problem. It was trained using 10% samples from each representative class. The results (using models P and O) are demonstrated in Tables 6.9-6.12 and may be verified from the vowel diagram of Fig. 2.6.

From Table 6.9 we observe a sample of the inferred output response obtained by the fuzzy logical model P (conjugate pair (T^p, S^p) of eqn. (6.2)) for the vowel data using partial and complete sets of input features. Tables 6.10 and 6.11 demonstrate the

Table 6.9: Output responses for vowel data with model P

Sr. No.	Input features			Highest output		Significant 2^{nd} choice		*Certainty* $cert_i^H$
	F_1	F_2	F_3	Class j	Membership y_i^H	Class	Membership	
1	700	1000	2600	*a*	0.96	-	-	0.99
2	400	800	*unobt*	*o*	0.27	-	-	0.89
3	400	*unobt*	*unobt*	*e*	0.17	*i*	0.17	0.4
4	700	1300	*unobt*	*a*	0.73	-	-	0.98
5	700	1000	*unobt*	*a*	0.84	-	-	1.0
6	450	2400	*unobt*	*i*	0.38	*e*	0.2	0.65
7	900	1400	*unobt*	*a*	0.44	-	-	1.0
8	600	1200	*unobt*	*a*	0.4	*o*	0.3	0.57
9	*between* 500 & 600	1600	*missing*	*e*	0.61	*o*	0.2	0.62
10	*high*	*Mol low*	*unobt*	*a*	0.75	-	-	1.0
11	*greater than* 650	*high*	*unobt*	*e*	0.37	-	-	1.0
12	*about* 350	*unobt*	*unobt*	*i*	0.16	-	-	0.81

Table 6.10: Querying by model P for vowel data

Serial No.	Input features			Query for
	F_1	F_2	F_3	
1	*about* 350	*missing*	*missing*	F_2
2	400	800	*missing*	F_3
3	700	*missing*	*missing*	F_2
4	450	2400	*mising*	F_3
5	900	1400	*missing*	F_3
6	600	1200	*missing*	F_3
7	*high*	*Mol low*	*missing*	F_3
8	*greater than* 650	*high*	*missing*	F_3
9	*between* 500 & 600	1600	*missing*	-

Table 6.11: Rules generated by model P for vowel data

Serial No.	Input features			Justification/ Rule generation	
	F_1	F_2	F_3	*If* clause	*Then* conclusion
1	300	900	*unobt*	F_1 is *very low* or F_2 is *very low*	*very likely* class *u*
2	700	1000	2600	F_3 is *very medium* and F_1 is *very medium* or F_1 is *Mol high* or F_2 is *very low*	*very likely* class *a*
3	700	*unobt*	*missing*	F_1 is *very medium* or *Mol high*	*likely* class *a*, but *not unlikely* class *e*
4	400	800	*unobt*	F_1 is *very low* and F_2 is *very low*	*very likely* class *o*, but *not unlikely* class *u*
5	400	*unobt*	*unobt*	F_1 is *very low*	*Mol likely* classes *e* or *i*, but *not unlikely* class *o*
6	700	1300	*unobt*	F_2 is *very medium* and F_1 is *very medium* or F_1 is *Mol high*	*very likely* class *a*
7	600	1200	*unobt*	F_1 is *very medium* and F_2 is *very medium* or F_2 is *low*	*Mol likely* classes *a* or *o*
8	450	2400	*unobt*	F_2 is *very high* and F_1 is *Mol low* or F_1 is *very medium*	*likely* class *i*, but *not unlikely* class *e*
9	900	1400	*unobt*	F_2 is *very medium* or F_1 is *very high*	*very likely* class *a*
10	*between* 500 & 600	1600	*missing*	F_2 is *very medium* or F_1 is *very medium* or *Mol low*	*likely* class e, but *not unlikely* class *o*
11	*greater than* 650	*high*	*missing*	F_2 is *high* and F_1 is *Mol high* or F_1 is *very medium*	*very likely* class *e*
12	*about* 350	*unobt*	*unobt*	F_1 is *very low*	*very likely* class *i*, but *not unlikely* class *e*

querying and rule generation phases of the said model (P) corresponding to a sample set of test input feature combinations. Table 6.12 depicts the rules generated by the fuzzy model O (with no logical operators) using the same network architecture and initial connection weights as model P. Note that the input feature values of a test pattern influence the generation of the rule from the trained set of connection weights. This helps extract rules relevant to that region of the feature space which is local to the area pointed to by the feature values of the test pattern.

Table 6.12: Rules generated by model O for vowel data

Serial No.	Input features			Justification/ Rule generation	
	F_1	F_2	F_3	*If* clause	*Then* conclusion
1	300	900	*unobt*	F_1 is *very low* and F_2 is *very low*	*very likely* class *u*, but *not unlikely* class *o*
2a	250	1550	*unobt*	F_1 is *very low* and F_2 is *Mol low*	*Mol likely* class *u*
2b	"	"	"	F_2 is *very medium* and F_1 is *very low*	*Mol likely* class *e*
3	700	1000	2600	F_3 is *very medium* and F_1 is *Mol high*	*very likely* class *a*
4a	700	*unobt*	*unobt*	F_1 is *very medium*	*not unlikely* class *∂*
4b	"	"	"	F_1 is *Mol high*	*not unlikely* class *a*
5	400	800	*missing*	F_2 is *very low* and F_1 is *very low*	*likely* class *o*, but *not unlikely* class *u*
6	700	2300	*missing*	F_2 is *very high* and F_1 is *very medium*	*very likely* class *e*
7	400	*unobt*	*unobt*	F_1 is *very low*	*not unlikely* classes *e* or *o* or *i* or *u* or *∂*
8	*unobt*	1000	*unobt*	F_2 is *very low*	*Mol likely* class *o*, but *not unlikely* classes *u* or *a*
9	700	1300	*unobt*	F_2 is *very medium* and F_1 is *Mol high*	*likely* class *a*
10	600	1200	*missing*	F_1 is *very medium*	*very likely* class *o*
11	*greater than 650*	*high*	*missing*	F_2 is *high* and F_1 is *very medium* or F_2 is *very medium*	*very likely* class *e*, but *not unlikely* class *∂*
12	*about* 350	*unobt*	*unobt*	F_1 is *very low*	*not unlikely* classes *u* or *e* or *i* or *o*

Artificially generated data

The rule generating ability of the three-layered *trained* neural networks with m

Table 6.13: Rules generated by model P for Pattern Set H

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*medium*	*low*	F_1 is *medium* to *Mol high* and F_2 is *very medium* or F_2 is *low*	*Mol likely* class 2, but *not unlikely* classes 1 or 3 or *no class*
2	*low*	*very high*	F_1 is *low* or *very medium* or F_2 is *very high*	*likely* class 5, and *Mol likely* classes 4 or *no class*
3	*low*	*Mol high*	F_2 is *very medium* or *Mol high* or F_1 is *low*	*Mol likely no class*, but *not unlikely* classes 5 or 4
4	*low*	*not high*	F_1 is *low* or *very medium* or F_2 is *very low*	*likely* class 1, but *not unlikely no class* or class 2
5	*medium*	*medium*	F_1 is *medium* to *Mol high* or F_2 is *medium* or *Mol low*	*not unlikely no class* or classes 3 or 2 or 4 or 1
6	*medium*	*high*	F_1 is *Mol low* to *medium* and F_2 is *high*	*Mol likely* class 4, but *not unlikely* classes 5 or *no class* or 3
7	*high*	*low*	F_1 is *high* and F_2 is *very medium* or F_2 *is low*	*Mol likely* classes 2 or 3
8	*high*	*high*	F_2 is *high* and F_1 is *very medium* to *high*	*likely* class 4, but *not unlikely* class 3

Table 6.14: Rules generated by model O forPattern Set H

Serial No.	*Input features*		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*Mol medium*	*low*	F_2 is *low* or *very medium* or F_1 is *Mol medium*	*very likely* class 2
2	*not low*	1600	F_2 is *very medium* or *Mol low*	*very likely* class 3
3	*low*	*very high*	F_1 is *low* or F_2 is *very high*	*likely* class 4, but *not unlikely* class 5
4	*low*	*not high*	F_2 is *very low* or F_1 is *low*	*likely* class 2, but *not unlikely* class 1
5	*low*	*medium*	F_2 is *medium* or *Mol high* or F_1 is *low* and F_2 is *Mol low*	*very likely no class*
6	*medium*	*high*	F_2 *is very medium* to *high* or F_1 is *Mol high*	*very likely* class 4
7	*high*	*low*	F_2 is *low* or *very medium* or F_1 is *high*	*Mol likely* classes 2 or 3
8	*high*	*medium*	F_2 is *Mol low* or *medium* or *Mol high*	*very likely* class 3
9	*high*	*high*	F_1 is *high* or F_2 is *very medium* to *high*	*very likely* class 4

hidden nodes (using 10% of the pattern points during training) are shown in Tables 6.13-6.22. We used $m = 12$ for *Pattern Set H* and $m = 10$ for the remaining

Table 6.15: Rules generated by model P for Pattern Set X

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_1 is *low* to *very medium* and F_2 is *low*	*likely no class*, but *not unlikely* class 1
2	*not low*	*low*	F_1 is *very high* and F_2 is *low* or F_2 is *very medium*	*Mol likely no class* or class 2
3	*medium*	*low*	F_1 is *Mol low* to *Mol high* or F_2 is *low*	*likely no class*, but *not unlikely* classes 2 or 1
4	*high*	*low*	F_1 is *high* and F_2 is *low* or F_1 is *very medium*	*likely no class*, but *not unlikely* class 2
5	*not high*	*low*	F_1 is *very low* and F_2 is *low* to *very medium*	*Mol likely no class*, but *not unlikely* class 1
6	*low*	*high*	F_1 is *low* to *very medium* and F_2 is *high*	*Mol likely no class*, but *not unlikely* class 2
7a	*medium*	*medium*	F_1 is *Mol low* to *Mol high* or F_2 is *medium*	*Mol likely no class*
7b	"	"	F_1 is *Mol low* to *medium* and F_2 is *Mol low* to *medium*	*not unlikely* class 1
7c	"	"	F_1 is *Mol low* to *medium* and F_2 is *medium* to *Mol high*	*not unlikely* class 2
8	*medium*	*high*	F_1 is *Mol low* to *medium* or F_1 is *Mol high* and F_2 is *high*	*Mol likely no class*, but *not unlikely* class 1
9	*high*	*medium*	F_2 is *Mol low* to *medium* or F_1 is *high* and F_2 is *Mol high*	*Mol likely no class*, but *not unlikely* class 1
10	*high*	*high*	F_1 is *high* and F_2 is *high* or F_1 is *very medium*	*Mol likely no class* or class 1

Pattern Sets (X, X1, L), to maintain uniformity with the results of Tables 6.5-6.8.

A sample of the rules generated from a set of test patterns using the logical model P for *Pattern Sets H, X, X1, L* are depicted in Tables 6.13, 6.15, 6.17 and 6.20 respectively. Tables 6.18 and 6.21 demonstrate the rules obtained using the corresponding logical model KDL (conjugate pair (T^m, S^m)) of eqns. (6.1) and (6.9) for *Pattern Sets* $X1$ and L. On the other hand, Tables 6.14, 6.16, 6.19 and 6.22 depict the case for the fuzzy model O (with no logical operators) with *Pattern Sets* H, X, $X1$ and L respectively. As model KDL could not result in good performance in cases of *Pattern Sets* H and X, we have decided to ignore its rule generating ability. Note that the fuzzy model O and the fuzzy logical model KDL use the same network architecture

Table 6.16: Rules generated by model O for Pattern Set X

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*not low*	*missing*	F_1 is *very high*	*likely no class*, but *not unlikely* class 1
2	*medium*	*missing*	F_1 is *medium* or *Mol high*	*very likely no class*
3	*Mol high*	*missing*	F_1 is *medium* or *Mol high*	*very likely no class*
4	*low*	*high*	F_2 is *very medium*	*likely no class*, but *not unlikely* class 2
5	*medium*	*medium*	F_1 is *medium* and F_2 is *medium* or F_1 is *Mol high*	*very likely no class*
6	*medium*	*high*	F_1 is *medium* and F_2 is *very medium*	*very likely no class*
7	*high*	*medium*	F_2 is *medium* and F_1 is *very medium* or F_1 is *high*	*very likely no class*

Table 6.17: Rules generated by model P for Pattern Set $X1$

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_1 is *low* and F_2 is *low* or F_1 is *very medium*	*very likely no class*
2	*low*	*medium*	F_2 is *Mol low* to *medium* and F_1 is *very medium* or F_1 is *low*	*Mol likely no class*, but *not unlikely* class 2
3	*low*	*high*	F_1 is *low* or *very medium* or F_2 is *high*	*likely* class 2, but *not unlikely no class*
4	*medium*	*low*	F_2 is *low* and F_1 is *Mol low* or F_1 is *medium* or *Mol high*	*Mol likely no class* or class 1
5	*high*	*low*	F_1 is *high* and F_2 is *low* or F_1 is *very medium*	*likely* class 1, but *not unlikely no class*

Table 6.18: Rules generated by model KDL forPattern Set $X1$

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_1 is *low* and F_2 is *low* or F_2 is *very medium*	*very likely no class*
2	*not low*	*low*	F_2 is *low* to *very medium*	*Mol likely* class 1, but *not unlikely no class*
3	*low*	*medium*	F_2 is *medium* to *Mol high* or F_1 is *low* and F_2 is *Mol low*	*Mol likely no class* or class 2
4	*medium*	*low*	F_1 is *medium* to *Mol high* or F_2 is *low* and F_1 is *Mol low*	*Mol likely no class* or class 1
5	*high*	*low*	F_2 is *low* or F_1 is *high* and F_2 is *very medium*	*Mol likely no class* or class 1
6	*high*	*medium*	F_2 is *medium* to *Mol high* and F_1 is *high* or F_2 is *Mol low*	*Mol likely no class* or class 1
7	*medium*	*Mol medium*	F_1 is *medium* and F_2 is *Mol medium* to *Mol high* or F_2 is *Mol low*	*not unlikely no class* or classes 1 or 2
8	*missing*	*high*	F_2 is *very medium* to *high*	*Mol likely* classes 2 or *no class*

Table 6.19: Rules generated by model O forPattern Set $X1$

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_1 is *low* and F_2 is *very medium* or F_2 is *low*	*very likely no class*
2	*low*	*high*	F_2 is *high* and F_1 is *low* or F_2 is *very medium*	*very likely* class 2
3	*medium*	*medium*	F_2 is *medium* to *Mol high* and F_1 is *Mol low* or F_1 is *medium*	*very likely no class*
4	*high*	*missing*	F_1 is *very medium* to *high*	*likely no class*, but *not unlikely* class 1
5	*high*	*low*	F_2 is *low* or *very medium* or F_1 is *very medium*	*likely* class 1, but *not unlikely no class*
6	*high*	*medium*	F_2 is *medium* to *Mol high* or F_1 is *high* or *very medium*	*very likely no class*, but *not unlikely* class 1

Table 6.20: Rules generated by model P for Pattern Set L

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_2 is *low* to *very medium* and F_1 is *low*	*likely* class 2, but *not unlikely* class 1
2	*low*	*medium*	F_2 is *Mol low* to *medium* and F_1 is *low* or F_2 is *Mol high*	*likely* class 2, but *not unlikely* class 1
3	*low*	*high*	F_2 is *high* or F_1 is *low* and F_2 is *very medium*	*very likely* class 2
4	*medium*	*low*	F_1 is *Mol low* to *medium* and F_2 is *low* or F_2 is *very medium*	*likely* class 1, but *not unlikely* class 2
5	*medium*	*high*	F_1 is *medium* and F_2 is *high* or F_1 is *Mol high* and F_2 is *very medium*	*likely* class 2, but *not unlikely* class 1
6	*high*	*low*	F_2 is *low* and F_1 is *very medium* or F_1 is *high*	*very likely* class 1
7	*high*	*medium*	F_2 is *medium* to *Mol high* and F_1 is *high* or F_2 is *Mol low*	*very likely* class 1, but *not unlikely* class 2
8	*high*	*high*	F_2 is *high* and F_1 is *high* or F_1 is *very medium*	*likely* class 1, but *not unlikely* class 2
9	*medium*	*medium*	F_1 is *Mol low* to *medium* and F_2 is *Mol low* or F_2 is *medium*	*Mol likely* classes 1 or 2

Table 6.21: Rules generated by model KDL for Pattern Set L

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*low*	F_2 is *low* to *very medium* and F_1 is *low*	*likely* class 2, but *not unlikely* class 1
2	*low*	*medium*	F_2 is *Mol low* to *medium* and F_1 is *low* or F_1 is *very medium*	*likely* class 2, but *not unlikely* class 1
3	*low*	*high*	F_2 is *high* or F_1 is *low* and F_2 is *very medium*	*very likely* class 2
4	*medium*	*low*	F_1 is *Mol low* to *medium* and F_2 is *low* or F_2 is *very medium*	*Mol likely* classes 1 or 2
5	*medium*	*high*	F_1 is *medium* or F_2 is *high* or F_1 is *Mol high* and F_2 is *very medium*	*Mol likely* classes 2 or 1
6	*high*	*low*	F_2 is *low* and F_1 is *very medium* or F_1 is *high*	*very likely* class 1
7	*high*	*medium*	F_2 is *medium* and F_1 is *high* or F_2 is *Mol high* and F_1 is *very medium*	*likely* class 1, but *not unlikely* class 2
8	*high*	*high*	F_2 is *Mol high* and F_1 is *very medium* F_2 is *high* and F_1 is *very medium* or F_1 is *high*	*likely* class 1, but *not unlikely* class 2

Table 6.22: Rules generated by model O for Pattern Set L

Serial No.	Input features		Justification/ Rule generation	
	F_1	F_2	*If* clause	*Then* conclusion
1	*low*	*medium*	F_2 is *Mol low* to *medium* and F_1 is *low*	*very likely* class 2
2	*high*	*low*	F_2 is *very medium* or F_1 is *high* or *very medium*	*likely* class 1, but *not unlikely* class 2
3	*medium*	*medium*	F_2 is *medium* to *Mol high* or F_1 is *medium* or *Mol high*	*likely* class 1, but *not unlikely* class 2

and initial connection weights as the fuzzy logical model P. It may be mentioned that the output responses and querying in each of the above cases are not demonstrated here as they are routine intermediate stages leading to the final rule generation phase and would have only served to make the results lengthier. All results may be verified by comparing with the pattern diagrams of Figs. 6.2-6.5.

6.5 Conclusions and discussion

A three-layered fuzzy neural network using logical operations, *viz.*, *t-norm* T and *t-conorm* S, for both classification and rule generation, has been presented in this chapter. The conventional back propagation algorithm has been modified incorporating the logical operations in place of weighted sum and sigmoid functions. The model can handle uncertainty and/or impreciseness in the input and output representations. Various fuzzy implication operators have been used to introduce different amounts of interaction during error propagation.

Like the models mentioned in Chapter 5, the fuzzy logical system can also query the user in cases of partial inputs and generate rules in *If-Then* form to justify an inferred decision. The magnitude of the connection weights and a certainty measure are used to determine the antecedent and consequent parts of these rules. The effectiveness of the model has been demonstrated on the speech data and on some synthetic data.

The effect of fuzzification has been studied, both at the input and the output, for the vowel data only. This is because the synthetic data used here has crisp output values. The logical model is seen to be robust with respect to variations in input

overlapping. It has been observed that those f_d-f_e values of eqn. (2.16) (for the fully fuzzified output representation of the fuzzy MLP) which correspond to higher contrast in membership values while also suitably representing the dynamic range of the given pattern points along the membership function curves, resulted in better performance.

During classification, the fuzzy MLP of Chapter 2 has been found to perform better than the logical models. Among the logical models used, the *product-probabilistic sum* operators have resulted in higher recognition scores as compared to the *max-min* operators. In this connection, it may be noted that the use of *And* and *Or* neurons in place of the *weighted sum* and *sigmoid* functions of the conventional MLP is expected to decrease the amount of computations required. The hardware implementation of logical neurons might also be easier. The involvement of logical neurons in the network structure has been found to help generate more appropriate rules (expressed in *And-Or* form as disjunction of conjunctive clauses) from the connection weights of the trained neural net (that constituted the embedded knowledge base for the problem at hand) as compared to that of the fuzzy MLP described in Chapter 5. This is due to the built-in *And-Or* structure of the fuzzy logical neural model.

However, it is worth mentioning that more meaningful rules can be generated by the fuzzy logical MLP in case of problem domains that can be represented in terms of *And-Or* combinations of the features. For complex feature spaces, like (say) real-life medical diagnosis, it will be shown in the next chapter that the fuzzy MLP (of Chapter 2) produces more accurate classification and better inferencing and/or rule generation.

Chapter 7

Application of the Fuzzy MLP to Medical Diagnosis

7.1 Introduction

Applications of the fuzzy MLP, for both classification and rule generation, have already been described in Chapters 2, 4 and 5 with reference to synthetic as well as real (*viz.*, speech) data. The present chapter demonstrates two more rigorous applications of the fuzzy MLP model in the medical domain, for diagnosing *hepatobiliary disorders* [283] and *kala-azar* [276]. However in order to handle the skewed distribution of the data set on hepatobiliary disorders, the input to this network is modified. The model is extended here to automate the generation of the input description of the patterns (along each feature axis) at the input layer from the training data. The centres and radii of the π-functions along each feature axis are determined automatically from the distribution of the training patterns. Initially it is used for classifying the multi-class patterns. Next, the trained network is used to generate rules (in If-Then form) in justification of the inferred decisions. A comparative study is made with that of the fuzzy neural expert system of Hayashi [241, 242] in the case of the hepatobiliary disorders data.

Medical data, or, rather more specifically, the results of biochemical tests [243] (on which we have implemented our model) involve imprecision, noise and individual difference. Often we cannot clearly distinguish the difference between normal and pathological values. Biochemical test results cannot be precisely evaluated by using crisp sets. Sometimes the patient can also be simultaneously diagnosed as suffering in different degrees from multiple diseases. Incorporation of fuzziness at the input and output levels of the neural network under consideration appears to be a good solution to such problems.

Before describing the results, we present the modifications incorporated into the fuzzy MLP for automating the generation of the input description of the patterns.

7.2 Modified input vector representation

When the input feature is numerical, we use the π-fuzzy sets of eqn. (2.9) (corresponding to the fuzzy MLP). However, when the input feature is linguistic, its membership

values for the π-sets low, medium and high are quantified as

$$\begin{aligned} low &\equiv \left\{ \frac{0.95}{L}, \frac{\pi\left(F_j\left(\frac{0.95}{L}\right); c_m, \lambda_m\right)}{M}, \frac{\pi\left(F_j\left(\frac{0.95}{L}\right); c_h, \lambda_h\right)}{H} \right\} \\ medium &\equiv \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_l, \lambda_l\right)}{L}, \frac{0.95}{M}, \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_h, \lambda_h\right)}{H} \right\} \\ high &\equiv \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{H}\right); c_l, \lambda_l\right)}{L}, \frac{\pi\left(F_j\left(\frac{0.95}{H}\right); c_m, \lambda_m\right)}{M}, \frac{0.95}{H} \right\} \end{aligned} \tag{7.1}$$

where c_l, λ_l, c_m, λ_m, c_h, λ_h refer to the centres and radii of the three linguistic properties, and $F_j\left(\frac{0.95}{L}\right)$, $F_j\left(\frac{0.95}{M}\right)$, $F_j\left(\frac{0.95}{H}\right)$ refer to the corresponding feature values F_j at which the three linguistic properties attain membership values of 0.95.

7.2.1 Choice of parameters

Let m_j be the mean of the pattern points along the j^{th} axis. Then m_{j_l} and m_{j_h} are defined as the mean (along the j^{th} axis) of the pattern points having co-ordinate values in the range $[F_{j_{min}}, m_j)$ and $(m_j, F_{j_{max}}]$ respectively, where $F_{j_{max}}$ and $F_{j_{min}}$ denote the upper and lower bounds of the dynamic range of feature F_j (for the training set) considering numerical values only. For the three linguistic property sets, we define the centres as

$$\begin{aligned} c_{medium(F_j)} &= m_j \\ c_{low(F_j)} &= m_{j_l} \\ c_{high(F_j)} &= m_{j_h} \end{aligned} \tag{7.2}$$

and the corresponding radii as

$$\begin{aligned} \lambda_{low(F_j)} &= 2\left(c_{medium(F_j)} - c_{low(F_j)}\right) \\ \lambda_{high(F_j)} &= 2\left(c_{high(F_j)} - c_{medium(F_j)}\right) \\ \lambda_{medium(F_j)} &= fno * \frac{\lambda_{low(F_j)} * \left(F_{j_{max}} - c_{medium(F_j)}\right) + \lambda_{high(F_j)} * \left(c_{medium(F_j)} - F_{j_{min}}\right)}{F_{j_{max}} - F_{j_{min}}} \end{aligned} \tag{7.3}$$

where fno is a multiplicative parameter controlling the extent of the overlapping. Note that this choice of parameters as well as the linguistic representation of features

are different from that described in Chapter 2. Here we take into account the distribution of the pattern points along each feature axis while choosing the corresponding centres and radii of the linguistic properties. This has been found to be more efficient in modeling skewed data distributions. Besides, the amount of overlap between the three linguistic properties can be different along the different axes, depending on the pattern set. In the process we are also able to eliminate the parameter $fdenom$, which required experimental evaluation, from the earlier input representation defined by eqns. (2.13)-(2.14).

7.3 Implementation and results

In this section we first present the results from the hepatobiliary disorders data, illustrating the classification and rule generation phases, followed by a similar application to kala-azar data.

7.3.1 Hepatobiliary disorders

Table 7.1: Upper and lower bounds and mean for data on hepatobiliary disorders

Feature	Unit	F_{jmin}	m_j	F_{jmax}
GOT	Karmen unit	8.0	113.0	4356.0
GPT	Karmen unit	3.0	54.5	1124.0
LDH	iu/l	179.0	476.3	6327.0
GGT	mu/ml	4.0	144.1	3075.0
BUN	mg/dl	3.3	17.2	91.0
MCV	fl	66.7	96.1	160.5
MCH	pg	20.3	32.1	52.5
TBil	mg/dl	0.1	3.2	37.0
CRTNN	mg/dl	0.4	1.1	4.3

The model has been implemented on a set of 536 patient cases of various hepatobiliary disorders. There are nine input features corresponding to the results of different biochemical tests, *viz.*, Glutamic Oxalacetic Transaminate (GOT, Karmen unit),

Glutamic Pyruvic Transaminase (GPT, Karmen Unit), Lactate Dehydrase (LDH, iu/l), Gamma Glutamyl Transpeptidase (GGT, mu/ml), Blood Urea Nitrogen (BUN, mg/dl), Mean Corpuscular Volume of red blood cell (MCV, fl), Mean Corpuscular Haemoglobin (MCH, pg), Total Bilirubin (TBil, mg/dl) and Creatinine (CRTNN, mg/dl). These have been represented in the $3n$-dimensional form of eqn. (2.10) (corresponding to the input representation of the fuzzy MLP). It is to be noted that the medical data under consideration has appreciably skewed distributions along most of the feature axes. The upper and lower bounds $F_{j_{max}}$, $F_{j_{min}}$ and the mean m_j along the j^{th} axis for each of the nine features are indicated in Table 7.1. This sort of data distribution is suitably handled by the choice of parameters given in eqns. (7.2)-(7.3) for the linguistic π-sets used. The 10^{th} feature corresponds to the sex of the patient and is represented in binary mode as (1,0) or (0,1). The hepatobiliary disorders used for the four output classes are Alcoholic Liver Damage (ALD), Primary Hepatoma (PH), Liver Cirrhosis (LC) and Cholelithiasis (C). The network has been trained using *perc* % samples from each pattern class of the data set. We have used $f_d = 5$, $f_e = 1$ in eqn. (2.16) (corresponding to the output membership evaluation for the fuzzy MLP) and $fno = 1$ in eqn. (7.3) after several experiments.

Table 7.2: Comparison of recognition scores for Hepatobiliary disorders data

Model	Hayashi's model with		Fuzzy MLP with	
	best choice	2^{nd} best choice	best choice	2^{nd} best choice
best match b	94.6	-	100.0	-
with 2^{nd} best match b_2	-	100.0	-	100.0
perfect match p	-	-	97.4	-
ALD	54.5	69.7	65.7	88.6
PH	66.6	82.3	87.0	90.7
LC	40.0	71.4	65.7	89.4
C	59.1	81.8	80.5	86.1
best net score t	56.4	-	76.0	-
with 2^{nd} best score t_2	-	77.3	-	88.9

The perfect match p, best match b and mean square error mse have been computed

for the training set while the individual classwise recognition scores, the overall score t and the mean square error mse_t have been computed for the test set (remaining $(100 - perc)$ % samples). The factors b_2 and t_2 correspond to the performance of the model when one also considers the second best choice for the training and test sets respectively. During the rule generation phase, complete/ partial sets of inputs are clamped at the input layer and the appropriate classification is inferred by the trained neural model. A measure of certainty has been used and querying regarding unknown feature values resorted to in case of some partial input sets. Justification in If-Then rule form, regarding a conclusion, can also be obtained when desired.

A comparison is provided with the results of Hayashi's model [241, 242]. There, the various cut-off levels for receptor responses of each feature were set manually after consultation with domain experts. Generally 50,000 iterations were required for convergence. In our approach, the choice of the parameters for the linguistic features along each feature axis is automated, depending on the pattern set distribution. Besides, convergence is also achieved generally within 500 sweeps. The use of the linear discriminant analysis on the same data have been reported by Hayashi *et al.* but the results were rather poor [241, 242].

Table 7.2 compares the classificatory performance of the fuzzy MLP (using three hidden layers having 40 nodes each) with that of Hayashi's method. Both models have used 70 % of the pattern set as training data. Note that the classwise recognition score, *using best choice, of our model* is comparable to the scores, *including the second best choice, of Hayashi's model.* The overall superiority of our model in classifying the pattern sets can be easily observed.

In Table 7.3 we provide a study of the effect on the recognition score (%) using different numbers of hidden layers, nodes and training set size *perc*. The number of hidden nodes in each case corresponds to the network configuration (found experimentally) providing the best results with the given combination of number of layers and training set size. It is observed that better results are obtained in cases representing large training set size coupled with large network configuration (in terms of hidden layers and nodes). Small training set sizes usually result in poor generalization capabilities on the test set.

Table 7.4 depicts the rule generation and querying phases of the fuzzy MLP (trained using $perc = 70\%$ with 40 nodes in each of the three hidden layers) for a sample set of

Table 7.3: Performance on Hepatobiliary disorders data using different network configurations

Layers	3			4			5		
perc	10	50	70	10	50	70	10	50	70
Nodes	20	30	10	15	15	20	15	25	40
b	98.1	94.8	79.0	100.	96.6	93.8	100.	99.2	100.
b_2	98.1	98.9	88.7	100.	98.5	98.1	100.	99.2	100.
p	56.9	23.9	9.2	98.1	26.9	54.7	96.1	90.3	97.4
mse	0.006	0.017	0.056	.0007	0.012	0.014	.0006	0.001	.0009
sweeps	350	280	200	180	370	470	210	430	400
ALD	31.4	53.4	42.8	62.8	56.9	62.8	44.7	60.3	65.7
PH	44.7	65.2	81.4	59.6	77.5	68.5	55.2	80.9	87.0
LC	56.2	58.1	55.2	33.0	56.4	68.4	46.4	59.7	65.7
C	80.3	79.7	83.3	65.4	89.8	83.3	68.2	84.7	80.5
t	52.3	64.2	67.4	55.4	70.9	70.5	53.8	72.4	76.0
t_2	71.3	83.2	80.9	72.1	83.2	82.2	75.2	84.3	88.9
$mse_t^{'}$	0.112	0.089	0.075	0.094	0.08	0.082	0.109	0.08	0.064

Table 7.4: Rule generation and querying phases for Hepatobiliary disorders data

Input features			Outp	Rule generated		Rule from
Initially supplied	Initially unknown	Queried for	memb val	If parts	Then part	initial features
$GOT > 100$ $GPT < 40$ $LDH > 700$ $GGT < 60$ $15 < BUN < 20$ $30 < MCH < 36$ *male*	MCV, TBil, CRTNN	-	0.77	GOT is medium GPT is low GGT is very low BUN is very medium	likely PH	ALD is completely false
$MCV > 100$ *male*	GOT, GPT, LDH, GGT, BUN, MCH, TBil, CRTNN	GOT is very low GPT is very low MCH is Mol medium BUN is very low LDH is very low	0.0 0.1 0.07 0.26 0.74 1.0	GOT is very low GPT is very low LDH is very low MCV is very high MCH is Mol medium MCH is Mol high	very likely LC	PH is completely false
$GOT < 40$ *female*	GPT, LDH, GGT, BUN, MCV, MCH, TBil, CRTNN	GPT is very low GGT is very low BUN is Mol medium LDH is low MCV is Mol medium	0.04 0.07 0.62 0.53 0.67 0.72	GOT is very low GPT is very low LDH is low GGT is very low BUN is Mol medium MCV is Mol medium	likely LC	PH is completely false
$GOT < 40$ $40 < GPT < 100$ $MCV < 90$ *female*	LDH, GGT, BUN, MCH, TBil CRTNN	MCH is Mol medium	0.75 0.8	GOT is very low MCV is very low MCH is Mol medium MCH is Mol low	likely C	PH is completely false
$LDH > 700$ $MCV < 90$ *male*	GOT, GPT, GGT, BUN, MCH, TBil, CRTNN	GOT is not high GPT is very medium MCH is Mol medium GGT is not medium BUN is very medium	0.87 0.91 0.94 0.94 0.87 0.84	GOT is low GPT is Mol low BUN is very medium MCV is very low MCH is Mol medium	very likely PH	LC is completely false
$BUN > 20$ $MCV > 100$ *male*	GOT, GPT, LDH, GGT, MCH, TBil, CRTNN	GOT is high GPT is high MCH is Mol medium	0.85 0.85 0.9 0.84	GOT is high BUN is Mol high MCV is high MCH is Mol medium MCH is Mol high	very likely PH	C is completely false
$GOT < 40$ $40 < GPT < 100$ *male*	LDH, GGT, BUN, MCV, MCH, TBil, CRTNN	MCH is high CRTNN is low GGT is very high BUN is very low MCV is very high	0.01 0.59 0.77 0.88 0.78 0.89	GOT is very low BUN is very low MCV is very high MCH is high CRTNN is low CRTNN very medium	very likely ALD	PH is completely false

partially known input features. Columns 3 and 4 refer respectively to the input feature supplied by the user after querying and the resulting output membership value of the neuron corresponding to the hepatobiliary disorder supported by the Then part of the generated rule in Column 6. The last column indicates the rules obtained from the initially supplied feature set in column 1 (from Hayashi, [300]). There were only two types of such rules in [300], viz., the ones excluding a disease and the ones confirming a disease. In our model we resort to querying and further updating to obtain rules that are more specifically indicative of a disease. It may be mentioned that the input feature values of the sample test pattern as well as the connection weights learned during training contribute to the rule generation process. Note that the certainty measure (and not the output membership value) is used in deciding whether querying should be resorted to at a particular stage and therefore querying is not required in all cases with partial set of input features (e.g., see row 1 of the table).

7.3.2 Kala-azar data

The next implementation is on *kala-azar* [301], a tropical disease, using a set of 68 patient cases. The input features are the symptoms while the output indicates the presence or absence of the disease. The symptoms are the measurements of *blood urea* (mg %), *serum creatinine* (mg %), *urinary creatinine* (mg %) and *creatinine clearance* (ml/min) indicated respectively as F_1, F_2, F_3 and F_4. These are represented in the linguistic form of eqn. (2.10). Due to limitations in the availability of sufficient real medical data, we have been forced to act in a constrained environment in this case.

The fuzzy MLP model has been trained on the *kala-azar* data using 30 (20 diseased and 10 control/ normal) cases. The test set consists of 38 samples constituting the responses of the above-mentioned 20 diseased patients (over the next 20 days) to the ongoing treatment [301]. Some of these patients were cured while the conditions of a few others worsened, sometimes ultimately culminating in death. The instances of patients cured constitute the output class *normal/ cured* while the remaining cases have been clubbed under the output class *diseased*. The classification performance of various sizes of the network on the *kala-azar* data using training set size $perc = 44.1(= 38/68)$ is depicted in Table 7.5. Note that mean square error mse, *perfect match* p and *best match* b refer to the training set while mean square error mse_t and *overall score* t are indicative of the test set (as mentioned earlier).

Table 7.5: Output Performance on Kala-azar data

Layers $H+1$	3		4
Nodes m	10	5	10
perfect p (%)	93.4	90.0	100.0
best b (%)	100.0	100.0	100.0
test t (%)	86.8	81.5	86.8
mse	0.002	0.004	0.001
mse_t	0.129	0.158	0.188

Table 7.6: Inferred output responses for Kala-azar data

Serial No.	Input features				Highest output		Significant 2^{nd} choice Membership	Certainty bel_j^H
	F_1	F_2	F_3	F_4	Class j	Membership y_j^H		
1	20.0	0.8	56.48	71.3	2	0.75	0.24	0.52
	22.5	0.87	61.21	60.5	2	0.91	-	0.83
2	26.0	0.9	51.45	76.6	1	0.50	0.49	0.01
	29.0	0.97	48.89	64.0	2	0.51	0.49	0.03
3	45.0	1.2	75.0	65.0	1	0.76	0.26	0.5
4	52.0	1.4	35.7	64.5	1	1.0	-	1.0
5	25.0	1.1	86.85	90.0	1	0.59	0.4	0.19
	27.0	1.3	117.27	89.3	1	1.0	-	1.0
6	18.0	0.83	78.8	65.5	2	0.75	0.25	0.5
	19.0	0.9	71.02	64.0	2	0.97	-	0.94
7	21.0	0.8	72.46	96.0	1	0.87	0.13	0.73
	30.0	1.1	96.4	85.0	1	1.0	-	1.0

Then the trained three-layered model with 10 hidden nodes has been used to demonstrate the inferencing ability (Tables 7.6-7.7) of the model on the *kala-azar* data. Table 7.6 shows the inferred output responses of the network for a sample set of test data. Here class 1 corresponds to *diseased* while class 2 refers to *cured*. The 1^{st} and 6^{th} entries correspond to patients experiencing speedy recovery during the course of treatment while the 2^{nd} entry refers to a patient who was gradually cured. The certainty measure and output membership values bear testimony to this. Note that the 1^{st} and 2^{nd} rows for each entry refer respectively to the status of the patient at the end of 10 and 20 days. The 3^{rd} and 4^{th} entries correspond to patients who expired after 10 days of treatment. The 5^{th} and 7^{th} entries refer to patients whose conditions deteriorated during treatment. All these cases may be verified from the patient records listed in [301].

Table 7.7: Rules generated for Kala-azar data

Serial No.	*If* clause	*Then* conclusion
1	F_3 is *very medium* and F_4 is *very low* and F_1 is *very low*	*more or less likely* cured
	F_3 is *very medium* and F_2 is *very medium* and F_1 is *Mol low*	*very likely* cured
2	F_4 is *very low* and F_2 is *very medium* and F_1 is *very medium* and F_3 is *low*	*unable to recognize*
	F_3 is *very medium* and F_2 is *Mol high*	*unable to recognize*
3	F_4 is *very low* and F_3 is *Mol high*	*more or less likely* diseased
4	F_4 is *very low* and F_3 is *very low*	*very likely* diseased
5	F_4 is *very medium* and F_1 is *very medium*	*not unlikely* diseased
	F_4 is *very medium* and F_1 is *Mol high*	*very likely* diseased
6	F_3 is *very medium* and F_2 is *very medium*	*more or less likely* cured
7	F_4 is *Mol high* and F_2 is *low*	*likely* diseased
	F_1 is *high* and F_4 is *Mol low*	*very likely* diseased

In Table 7.7 we illustrate a few of the rules generated from the knowledge base. The serial nos. refer to the corresponding test cases provided in Table 7.6. The antecedent and consequent parts are deduced as explained in Section 5.2.4.

7.4 Conclusions and discussion

An application of the fuzzy MLP in the medical domain has been described for the diagnosis of hepatobiliary disorders and kala-azar. The centres and radii of the π-sets used to model the linguistic properties at the input are automatically determined from the training patterns, in order to handle the skewed distribution of the data on hepatobiliary disorders. The *superiority of the model over Hayashi's network* [242, 300] has been established.

Medical information such as results from biochemical tests and/or the diagnosed disorder are often ambiguous and/or fuzzy [243]. Hence incorporation of fuzziness at the input and output levels becomes more effective in modeling such problems. The skewness of the data set under consideration is also found to be appropriately handled by the chosen input description that automatically determines the centres and radii of the linguistic π-sets.

Note that this medical diagnosis problem was also attempted using the fuzzy logical MLP (Chapter 6) and the fuzzy Kohonen's net, but the results were not satisfactory.

Chapter 8

Conclusions and Scope for Further Work

8.1 Conclusions

The present thesis has provided some results of investigation demonstrating the effectiveness of the neuro-fuzzy approach for pattern classification, rule generation and inferencing. Some new connectionist models have been designed, in which the concept of fuzzy sets is incorporated at the input level for handling impreciseness of information in terms of linguistic properties, at the output level for representing ambiguous decisions in terms of membership values, and also at the neuronal level by using *And-Or* operations. Methodologies have been developed for rule generation and querying in the process of *inferring a decision*. The basic models like the multilayer perceptron and the Kohonen's net have been used for the purpose. The effectiveness of these models has been demonstrated on speech, medical and synthetic data, and comparisons provided with other related algorithms.

A method of integration of the uncertainty handling capability of fuzzy set theory with the ability of the multilayer perceptron in generating highly nonlinear decision boundaries, has been described in Chapter 2 for pattern classification. The system accepts input features in linguistic form (*low*, *medium* and *high*), and provides soft (fuzzy) decisions for ambiguous patterns in terms of class membership values. The backpropagated error has inherently more weight in case of nodes with higher membership values and this helps to reduce oscillations of the *decision boundaries* caused by patterns from overlapping regions. During training, the *learning rate* is gradually decreased in discrete steps.

A fuzzy version of self-organizing Kohonen's model has been presented in Chapter 3. Like the fuzzy MLP of Chapter 2, the *input vector* consists of membership values for the *linguistic* properties. In addition, some *contextual class membership* information is used at the input during self organization to permit efficient modeling of *fuzzy* (ambiguous) patterns. A new definition of gain factor for weight updating is provided. An *index of disorder* involving mean square distance between the input and weight vectors is used to determine a measure of the *ordering* of the output space. This controls the number of sweeps required in the process. Incorporation of the concept of *fuzzy partitioning* allows natural self organization of the input data, especially when they have ill-defined boundaries. Note that the conventional Kohonen's net is generally used for clustering. Here, on the other hand, the network has been used as a partially

supervised classifier. The significance of this approach (as discussed in Section 3.1) is particularly evident when we consider the case of human language, where clustering or classifying solely on the basis of physical measurements is not useful. Some additional contextual factors, which cannot be directly elucidated from these physical measurements, become essential for the construction of meaningful clusters.

Although these fuzzy models have been designed for handling uncertainty arising from imprecision in input information and overlapping (*ambiguous*) nature of class boundaries, their capability of classifying linearly nonseparable, *nonconvex* (non overlapping) pattern sets has been demonstrated in Chapter 4. Here the output decision is crisp. An extensive comparison with several existing connectionist approaches has also been provided.

Some methodologies for inferencing and rule generation have then been developed in Chapter 5 using the aforesaid fuzzy models. Based on the output class membership value(s) of unknown patterns, the networks can generate some measure of certainty expressing confidence in the decision. In the case of partial inputs they are capable of querying the user for *important* input feature information. Justification for an inferred decision is produced in rule form, when so desired by the user /expert. The magnitudes of the connection weights and the certainty measure are utilised for generating the antecedent and consequent parts of these rules in *natural* forms. The node excitations, corresponding to a test pattern, help in extracting the appropriate *If-Then* parts of the rules.

Fuzziness has also been incorporated at the neuronal level of the fuzzy MLP by replacing the standard weighted sum and sigmoid functions with the logical *And* and *Or* operators (Chapter 6) for classification, rule generation and inferencing. (Note that the use of *And-Or* operators is expected to decrease the amount of computations required and to make the hardware implementation simpler). The conventional *back-propagation algorithm* is accordingly modified and two cases of the conjugate pairs of *t-norm* T and *t-conorm* S, *viz.*, *max-min* and *product-probabilistic sum*, are modeled. The hidden layer consists of *And* nodes while the output layer is made up of *Or* nodes. Fuzzy implication operators are employed to incorporate various amounts of mutual interaction during error propagation.

The merits of these fuzzy connectionist models have been demonstrated on speech data, certain linearly nonseparable nonconvex pattern sets and some medical data for

classification, inferencing and rule generation. Comparison with the respective conventional versions, the Bayes' classifier, the k-NN classifier and other neural algorithms (as applicable) have been provided. The representation of the input information in terms of low, medium and high enables the fuzzy models to utilize more local information about the feature space, and thereby classify better than their conventional counterparts. Even in cases where the results are comparable, the fuzzy version has the added advantage that it can handle imprecise and/or linguistic inputs. As expected, the fully supervised fuzzy MLP always performs better than the partially supervised fuzzy Kohonen's net. The fuzzy MLP is also found to be better than the Bayes' classifier for the vowel recognition problem, and over the k-NN classifier for the linearly non-separable nonconvex pattern sets. Incorporating *a priori* probabilities of the pattern classes during weight updating (using backpropagation) has resulted in improvement of the recognition score for the classes having relatively smaller number of samples. The effect of different network configurations, training set sizes, and fuzzification at the input and output (on the system performance) has also been investigated.

In the case of rule generation, better results have again been found with the fuzzy MLP compared to the fuzzy Kohonen's net. Relatively more complete and/or accurate rules have been generated by both the models for the vowel data, as compared to the afore-mentioned nonconvex pattern sets. This is because the nonconvex pattern sets are quite complex, considering the *difficult* nature of the problem of class separability in these cases, relative to the vowel data. Incorporation of the logical *And* and *Or* operators enables the generation of appropriate rules expressed as the disjunction of conjunctive clauses. This is specially true in case of problem domains that can be represented in terms of simple *And-Or* combinations of the features. However, it has been observed that the fuzzy MLP (using sigmoidal nonlinearities) usually performs better for classification. For the medical diagnosis of hepatobiliary disorders, the centres and radii of the linguistic π-sets at the input have been automatically determined from the distribution of the training set. It has been observed that in the case of more complex feature spaces (*e.g.*, medical diagnosis), it is appropriate to employ the fuzzy MLP (with no logical operators) for producing more accurate classification and better inferencing and/or rule generation. The results have also been found to be superior to that of Hayashi's network [242, 300].

The model based on fuzzy MLP for rule generation and inferencing can be consid-

ered to be a basic module for a rudimentary connectionist expert system. The rules generated by this model can also be used for constituting the knowledge base of a traditional classification-type expert system for the problems under consideration.

8.2 Scope for further work

All the models described in this thesis have considered a $3n$-dimensional input space in terms of the linguistic properties. This has enabled the networks to utilise more local information about the feature space, and thereby improved their performance. However, this entails an increase in the number of network components, thereby increasing the expense both computationally and from the hardware context. An investigation may be made for reducing this cost by selecting an optimal number of input features.

Determining the optimal architecture for a connectionist model is of prime importance for any application and is still an open problem. All the models described in this thesis have their network topologies (in terms of the number of layers and/or nodes) determined empirically. It would be interesting to investigate the determination of the optimal size of the network, as this is likely to help in improving the performance of the network both for classification and rule generation. Some mathematical analysis for the determination of the appropriate network configuration and the improvement of the speed of convergence needs to be made. Mathematical proof of the convergence to a minimum error solution can also be attempted.

Backpropagation, due to its gradient descent nature, often gets trapped in a local minimum of the error function and is inefficient in searching the global minimum of a function which is vast, multimodal and nondifferentiable. One way to overcome gradient descent search-based training algorithms' shortcomings, is to consider the training process as the evolution of connection weights towards an optimal (or near optimal) set defined by a fitness function, and the training task as the environment in which the evolution occurs. It would be interesting to use Genetic Algorithms (GAs) as function optimizers to maximize fitness functions (or minimize error functions), since GAs are good at dealing with large, complex, nondifferentiable and deceptive spaces. Network growing and pruning can also be investigated, using GAs, to generate the optimal set of interconnections. Both nodes and layers may be added and/or deleted in the process.

The efficiency of the training can also be improved by initially using GAs to quickly locate a good region, represented by a starting point in the weight space, and then employing a local search procedure (like backpropagation or simulated annealing) to find a near optimal solution in this region. This sort of hybrid approach can help avoid the problems associated with local minima of conventional gradient-descent algorithms while also utilising the fine-tuning capability of the latter. Moreover, as evolutionary training is usually computationally more intensive and slower than the faster variants of gradient descent-based techniques, such a hybrid approach could also speed up the resultant search procedure.

The fuzzy Kohonen's net used here has employed too many parameters that need to be defined by the user. This is definitely a drawback of the model and has to be looked into. Moreover, both the fuzzy models (described here) have used a number of parameters for evaluating the membership values at the input and the output. All these parameters have been computed empirically. Although a method to evaluate the parameters for the input linguistic features has been described in Chapter 7, yet some further investigation needs to be carried out in this direction. GA's could also be used for tuning these membership functions.

The connection weights of the trained neural models can be analysed for evaluating the impact of the various features on output decision. This has a useful application to the problem of feature evaluation and an associated dimensionality reduction. The effect of the magnitudes of the connection weights and their impact on positive and/or non-positive output decision may be investigated in order to prune redundant inter-connections and/or features. This can also lead to the generation of a more optimal network architecture, resulting in better performance of the model.

Rule generation and querying have been examined here considering the output nodes having high membership values only. These concepts need to be extended for low membership output nodes as well. This will enable the model to generate rules justifying decisions in the case of patterns *not belonging* to an output class (in addition to the case of justifying the *belongingness to a class* only, as described here).

Incorporation of fuzzy set-theoretic concepts has been made at input, output and neuronal (using *And-Or* operators) levels of the connectionist models. Some more investigations regarding the use of the various fuzzy hybrid and aggregate operators at the neuronal level and their effect on the rule generating ability of the network can

be made. The characteristic of the node transfer function can also be learned during training, thereby producing a more flexible connectionist system. Moreover, the sort of neuro-fuzzy integration studied here with reference to the MLP and the Kohonen's net can be extended to other well-known basic neural models, like (say) the Hopfield net.

The neuro-fuzzy diagnostic system for inferencing and rule generation, described here, can be considered to be a basic module of a connectionist expert system. This can be extended to larger domains to help design a more practical model of an expert system. In this thesis we have considered only patterns represented in the form of features. Extension to handle image data could be an interesting proposition. Some sort of preprocessing for extracting the relevant features could be made and then these might be used as the input to the designed system. Alternatively the network can be modified to accept the entire image array at the input, for different applications.

Finally, knowledge-based networks could be investigated (using the neuro-fuzzy models described in the thesis) to determine their effectiveness in generating powerful connectionist expert systems. Some knowledge could initially be elicited from one or more experts. Next, the nodes and/or layers could be grown and/or pruned to generate the optimal architecture. GAs and/or other traditional techniques could be employed for this purpose. The performance of the resultant model, for subsequent querying and rule generation, will certainly be better and holds promise for future investigations.

Bibliography

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley, 1973.

[2] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* London: Addison-Wesley, 1974.

[3] A. Rosenfeld and A. C. Kak, *Digital Picture Processing.* Vol. 1-2, New York: Academic Press, 1982.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Reading, MA: Addison-Wesley, 1992.

[5] D. Marr, *Vision.* New York: W. H. Freeman and Co., 1982.

[6] D. H. Ballard and C. M. Brown, *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

[7] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence.* Cambridge, MA: Addison-Wesley, 1985.

[8] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems.* London: Addison-Wesley, 1983.

[9] D. A. Waterman, *A Guide to Expert Systems.* London: Addison-Wesley, 1985.

[10] S. I. Gallant, "Connectionist expert systems," *Communications of the Association for Computing Machinery*, vol. 31, pp. 152–169, 1988.

[11] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[12] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning : Part 1, 2, and 3," *Information Sciences*, vol. 8, 8, 9, pp. 199–249, 301–357, 43–80, 1975.

[13] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[14] H. -J. Zimmermann, *Fuzzy Set Theory - and its Applications.* Boston: Kluwer, 1991.

[15] A. Kaufmann, *Introduction to the Theory of Fuzzy Subsets - Fundamental Theoretical Elements.* Vol. 1; New York: Academic Press, 1975.

[16] A. Kandel, *Fuzzy Mathematical Techniques with Applications.* Reading, MA: Addison-Wesley, 1986.

[17] S. K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition.* New York: John Wiley (Halsted Press), 1986.

[18] G. J. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information.* Reading, MA: Addison-Wesley, 1989.

[19] J. C. Bezdek, ed., *Analysis of Fuzzy Information.* Vol. I-III, Boca Raton: CRC Press, Inc., 1987.

[20] R. R. Yager and L. A. Zadeh, eds., *An Introduction to Fuzzy Logic Applications in Intelligent Systems.* Boston: Kluwer Academic Press, 1992.

[21] I. B. Turksen, "Four methods of approximate reasoning with interval-valued fuzzy sets," *International Journal of Approximate Reasoning*, vol. 3, pp. 121–142, 1989.

[22] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum Press, 1981.

[23] J. C. Bezdek and S. K. Pal, eds., *Fuzzy Models for Pattern Recognition : Methods that Search for Structures in Data.* New York: IEEE Press, 1992.

[24] A. Kandel, *Fuzzy Techniques in Pattern Recognition.* New York: Wiley Interscience, 1982.

[25] R. R. Yager, ed., *Fuzzy Set and Possibility Theory.* New York: Pergamon Press, 1982.

[26] M. M. Gupta and E. Sanchez, eds., *Fuzzy Information and Decision Processes.* Amsterdam: North Holland, 1982.

[27] J. M. Kickert, *Fuzzy Theories on Decision-making.* London: Martinus Nijhoff Social Sciences Division, 1978.

[28] D. Dubois and H. Prade, *Fuzzy Sets and Systems : Theory and Applications.* Boston: Academic Press, Inc., 1980.

[29] A. Kaufmann and M. Gupta, *Introduction to Fuzzy Mathematics.* New York: Van Nostrand Reinhold, 1985.

[30] L. A. Zadeh, K. S. Fu, K. Tanaka, and M. Shimura, eds., *Fuzzy Sets and their Applications to Cognitive and Decision Processes.* London: Academic Press, 1975.

[31] E. Sanchez and L. A. Zadeh, eds., *Approximate Reasoning in Intelligent Systems, Decision and Control.* Oxford: Pergamon Press, 1987.

[32] D. Dubois and H. Prade, *Possibility Theory.* New York: Plenum Press, 1988.

[33] H. -J. Zimmermann, *Fuzzy Sets, Decision Making and Expert Systems.* Boston, MA: Kluwer Academic Publishers, 1987.

[34] C. V. Negoita, ed., *Expert Systems and Fuzzy Systems.* California: The Benjamin Cummings Publishing Co. Inc., 1985.

[35] M. M. Gupta, A. Kandel, W. Bandler, and J. B. Kiszka, eds., *Approximate Reasoning in Expert Systems.* Amsterdam: North Holland, 1985.

[36] A. Hart, *Knowledge Acquisition for Expert Systems.* London: Kogan Page, 1986.

[37] L. A. Zadeh, "The role of fuzzy logic in the management of uncertainty in expert systems," *Fuzzy Sets and Systems*, vol. 11, pp. 199–227, 1983.

[38] A. Kandel, ed., *Fuzzy Expert Systems.* Boca Raton: CRC Press, Inc., 1991.

[39] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing : Explorations in the Microstructures of Cognition.* Vol. 1, Cambridge, MA: MIT Press, 1986.

[40] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoustics, Speech and Signal Processing Magazine*, vol. 4, pp. 4–22, 1987.

[41] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1989.

[42] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, pp. 3–16, 1988.

[43] S. Grossberg, ed., *Neural Networks and Natural Intelligence.* Cambridge, MA: MIT Press, 1988.

[44] J. Dayhoff, *Neural Network Architectures : An Introduction.* New York: Van Nostrand Reinhold, 1990.

[45] P. D. Wassermann, *Neural Computing : Theory and Practice.* New York: Van Nostrand Reinhold, 1990.

[46] P. Simpson, *Artificial Neural Systems : Foundations, Paradigms, Applications and Implementations.* Elmsford, NY: Pergamon Press, 1990.

[47] D. O. Hebb, *The Organization of Behaviour.* New York: Wiley, 1949.

[48] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks.* Reading, MA: Addison-Wesley, 1989.

[49] *Proc. of Second International Conference on Fuzzy Logic and Neural Networks (IIZUKA92)*, (Iizuka, Fukuoka, Japan), July 1992.

[50] *Proc. of First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, (San Diego, USA), March 1992.

[51] *Proc. of Second IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, (California, USA), March 1993.

[52] *Proc. of Third International Conference on Fuzzy Logic and Neural Networks (IIZUKA94)*, (Iizuka, Fukuoka, Japan), July 1994.

[53] *Proc. of Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, (Florida, USA), June 1994.

[54] P. W. Becker, *Recognition of Patterns.* Wien: Springer-Verlag, 1978.

[55] L. N. Kanal, *Pattern Recognition.* Washington DC: Thompson Books, 1968.

[56] G. S. Sebestyen, *Decision Making Processes in Pattern Recognition.* NY: The Macmillan Co., 1962.

[57] P. A. Devijver and J. Kittler, eds., *Pattern Recognition Theory and Applications.* Berlin: Springer-Verlag, 1987.

[58] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic Press, 1972.

[59] K. S. Fu, *Pattern Recognition and Machine Learning.* New York: Plenum Press, 1971.

[60] E. H. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, pp. 22–32, 1969.

[61] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in a plane," *IEEE Transactions on Information Theory*, vol. 29, pp. 551–559, 1983.

[62] C. A. Murthy, *On Consistent Estimation of Classes in R^2 in the Context of Cluster Analysis.* PhD thesis, Indian Statistical Institute, Calcutta, India, 1988.

[63] W. Pedrycz, "Fuzzy sets in pattern recognition : methodology and methods," *Pattern Recognition*, vol. 23, pp. 121–146, 1990.

[64] R. Bellman, R. Kalaba, and L. A. Zadeh, "Abstraction and pattern classification," *Journal of Mathematical Analysis Applications*, vol. 13, pp. 1–7, 1966.

[65] S. K. Pal and D. Dutta Majumder, "Fuzzy sets and decision making approaches in vowel and speaker recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 625–629, 1977.

[66] A. Pathak and S. K. Pal, "On the convergence of a self-supervised vowel recognition system," *Pattern Recognition*, vol. 20, pp. 237–244, 1987.

[67] A. K. Nath and T. T. Lee, "On the design of a classifier with linguistic variables as input," *Fuzzy Sets and Systems*, vol. 11, pp. 265–286, 1983.

[68] A. K. Nath, S. W. Liu, and T. T. Lee, "On some properties of linguistic classifier," *Fuzzy Sets and Systems*, vol. 17, pp. 297–311, 1985.

[69] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 580–585, 1985.

[70] B. B. Devi and V. V. S. Sarma, "A fuzzy approximation scheme for sequential learning in pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 668–679, 1986.

[71] J. C. Bezdek and P. F. Castelaz, "Prototype classification and feature selection with fuzzy sets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 87–92, 1977.

[72] S. K. Pal and B. Chakraborty, "Fuzzy set theoretic measure for automatic feature evaluation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 754–760, 1986.

[73] R. L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 28–35, 1977.

[74] E. T. Lee, "Fuzzy tree automata and syntactic pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 445–449, 1982.

[75] R. De Mori and P. Laface, "Use of fuzzy algorithms for phonetic and phonemic labeling of continuous speech," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 136–148, 1980.

[76] A. Pathak and S. K. Pal, "Fuzzy grammars in syntactic recognition of skeletal from x-rays," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 657–667, 1984.

[77] A. Pathak, S. K. Pal, and R. A. King, "Syntactic recognition of skeletal maturity," *Pattern Recognition Letters*, vol. 2, pp. 193–197, 1984.

[78] P. Siy and C. S. Chen, "Fuzzy logic for handwritten numeral character recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 4, pp. 570–575, 1974.

[79] A. Nafarieh and J. Keller, "A fuzzy logic rule-based automatic target recognition," *International Journal of General Systems*, vol. 6, pp. 295–312, 1991.

[80] H. F. Wang, C. W. Wu, C. H. Ho, and M. J. Hsieh, "Diagnosis of gastric cancer by fuzzy pattern recognition," *Journal of Systems Engineering*, vol. 2, pp. 151–163, 1993.

[81] K. Hirota and W. Pedrycz, "Geometric-logical pattern classification," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 675–678, 1992.

[82] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, pp. 21–32, 1992.

[83] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Efficient fuzzy partition of pattern space for classification problems," *Fuzzy Sets and Systems*, vol. 59, pp. 295–304, 1993.

[84] S. K. Pal and D. P. Mandal, "Linguistic recognition system based on approximate reasoning," *Information Sciences*, vol. 61, pp. 135–161, 1992.

[85] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Formulation of a multi-valued recognition system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 607–620, 1992.

[86] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Utility of multiple choices in detecting ill-defined roadlike structures," *Fuzzy Sets and Systems*, vol. 64, pp. 213–228, 1994.

[87] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Theoretical performance of a multivalued recognition system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, pp. 1001–1021, 1994.

[88] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Determining the shape of a pattern class from sampled points in R^2," *International Journal of General Systems*, vol. 20, pp. 307–339, 1992.

[89] J. C. Bezdek, "A physical interpretation of fuzzy ISODATA," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 387–389, 1976.

[90] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, pp. 32–57, 1973.

[91] M. Roubens, "Pattern classification problems and fuzzy sets," *Fuzzy Sets and Systems*, vol. 1, pp. 239–253, 1978.

[92] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 773–781, 1989.

[93] S. K. Pal and S. Mitra, "Fuzzy dynamic clustering algorithm," *Pattern Recognition Letters*, vol. 11, pp. 525–535, 1990.

[94] E. Rich and K. Knight, *Artficial Intelligence*. New York: McGraw Hill, Inc., 1991.

[95] M. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, (P. Winston, ed.), p. , New York: Mc-Graw Hill, 1975.

[96] R. Quillian, "Semantic memory," in *Semantic Information Processing*, (M. Minsky, ed.), Cambridge, MA: MIT Press, 1968.

[97] L. Shastri, "Default reasoning in semantic networks : a formalization of recognition and inheritance," *Artificial Intelligence*, vol. 39, pp. 283–355, 1989.

[98] J. Pearl, "Distributed revision of composite beliefs," *Artificial Intelligence*, vol. 33, pp. 173–215, 1987.

[99] K. Ng and B. Abramson, "Uncertainty management in expert systems," *IEEE Expert*, pp. 29–48, April 1990.

[100] B. G. Buchanan and E. H. Shortliffe, eds., *Rule-based Expert Systems : The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, MA: Addison-Wesley, 1984.

[101] G. Shafer, *A Mathematical Theory of Evidence.* Princeton, NJ: Princeton University Press, 1976.

[102] R. O. Duda, P. E. Hart, and N. J. Nilsson, "Subjective Bayesian methods for a rule-based inference system," in *Proceedings of the National Computer Conference*, (USA), pp. 1075–1082, 1976.

[103] E. H. Shortliffe, *Computer-based Medical Consultations : MYCIN.* New York: Elsevier North-Holland, 1976.

[104] I. B. Turksen and Z. Zhong, "An approximate analogical reasoning schema based on similarity measures and interval-valued fuzzy sets," *Fuzzy Sets and Systems*, vol. 34, pp. 323–346, 1990.

[105] M. Ishizuka, K. S. Fu, and J. T. P. Yao, "Inference procedures under uncertainty for the problem-reduction method," *Information Sciences*, vol. 28, pp. 179–206, 1982.

[106] A. O. Esogbue and R. C. Elder, "Fuzzy sets and the modelling of physician decision processes, Part I : The initial interview - information gathering session," *Fuzzy Sets and Systems*, vol. 2, pp. 279–291, 1979.

[107] E. Sanchez and R. Bartolin, "Fuzzy inference and medical diagnosis, a case study," *Biomedical Fuzzy Systems Bulletin*, vol. 1, pp. 4–21, 1990.

[108] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 1414–1427, 1992.

[109] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386-408, 1958.

[110] F. Rosenblatt, *Principles of Neurodynamics, Perceptrons and the Theory of Brain Mechanisms.* Washington: Spartan Books, 1961.

[111] M. Minsky and S. Papert, *Perceptrons : An Introduction to Computational Geometry.* Cambridge, MA: MIT Press, 1969.

[112] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing : Explorations in the Microstructures of Cognition*, (D. E. Rumelhart and J. L. McClelland, eds.), Cambridge, MA: MIT Press, 1986.

[113] J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing Foundations of Research.* Cambridge, MA: MIT, 1988.

[114] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185–234, 1989.

[115] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 1, pp. 179–191, 1990.

[116] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machine," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[117] E. H. L. Aarts and J. H. M. Korst, *Boltzmann machines and their applications*, pp. 34–50. Vol. 258 of *Lecture Notes on Computer Science*, Springer Verlag, 1987.

[118] D. Parker, "Learning logic," Tech. Rep. TR-87, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.

[119] Y. Le Cun, "Learning processes in an asymmetric threshold network," in *Disordered Systems and Biological Organization*, (E. Bienenstock, F. F. Souli, and G. Weisbuch, eds.), Berlin: Springer-Verlag, 1986.

[120] J. J. Hopfield, "Neurons with graded response have collective computational property like those of two-state neurons," *Proceedings of National Academy of Sciences, USA*, vol. 81, pp. 3088–3092, 1984.

[121] J. J. Hopfield, "Neural network and physical systems with emergent collective computational abilities," *Proceedings of National Academy of Sciences, USA*, vol. 79, pp. 2554–2558, 1982.

[122] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organising neural pattern recognition machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 54–115, 1987.

[123] G. A. Carpenter and S. Grossberg, "ART2 : Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, pp. 4919–4930, 1987.

[124] G. A. Carpenter and S. Grossberg, "ART 3 : Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, pp. 129–152, 1990.

[125] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks : Benchmarking studies," in *Proceedings of IEEE International Conference on Neural Networks*, pp. I:61–I:68, 1988.

[126] T. Kohonen, "Things you haven't heard about the self-organizing map," in *Proceedings of IEEE International Joint Conference on Neural Networks*, (San Francisco, USA), pp. 1147–1156, 1993.

[127] R. Hecht-Nielsen, "Counterpropagation networks," *Applied Optics*, vol. 26, pp. 4979–4984, 1987.

[128] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, pp. 131–139, 1988.

[129] K. Fukushima, "Neocognitron : A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, 1988.

[130] K. Fukushima, "A neural network for visual pattern recognition," *IEEE Computer*, pp. 65–75, March 1988.

[131] M. Okada and K. Fukushima, "Neocognitron learned by backpropagation," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 667–670, 1990.

[132] A. Iwata, K. Hotta, H. Matsuo, N. Suzumura, S. Matsuda, and M. Yoshida, "A large scale neural network "CombNET" on a neural network accelerator Neuro-

Turbo," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 329–333, 1990.

[133] G. A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Networks*, vol. 2, pp. 243–257, 1989.

[134] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Communications Magazine*, pp. 47–64, 1989.

[135] B. Widrow and R. Winter, "Neural network for adaptive filtering and adaptive pattern recognition," *IEEE Computer*, pp. 25–39, March 1988.

[136] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Transactions on Neural Networks*, vol. 3, pp. 241–251, 1992.

[137] M. Fukumi, S. Omatu, F. Takeda, and T. Kosaka, "Rotation-invariant neural pattern recognition system with application to coin recognition," *IEEE Transactions on Neural Networks*, vol. 3, pp. 272–279, 1992.

[138] P. W. M. Tsang and P. C. Yuen, "Recognition of partially occluded objects," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 228–236, 1993.

[139] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Systems*, vol. 1, pp. 145–168, 1987.

[140] D. J. Burr, "Experiments on neural net recognition of spoken and written text," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, pp. 1162–1168, 1988.

[141] K. Fukushima, "Character recognition with neural networks," *Neurocomputing*, vol. 4, pp. 221–233, 1992.

[142] S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," *IEEE Transactions on Neural Networks*, vol. 3, pp. 962–968, 1992.

[143] T. Kohonen, "The neural phonetic typewriter," *IEEE Computer*, pp. 11–22, March 1988.

[144] R. L. Watrous, "Context-modulated vowel discrimination using connectionist networks," *Computer Speech and Language*, vol. 5, pp. 341–362, 1991.

[145] B. R. Kammerer and W. A. Kupper, "Experiments for isolated-word recognition with single- and two-layered perceptrons," *Neural Networks*, vol. 3, pp. 693–706, 1990.

[146] S. S. Fels and G. E. Hinton, "Glove-talk : a neural network interface between a data-glove and a speech synthesizer," *IEEE Transactions on Neural Networks*, vol. 4, pp. 2–8, 1993.

[147] R. P. Gorman and T. J. Sejnowski, "Learned classification of sonar targets using a massively parallel network," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, pp. 1135–1140, 1988.

[148] Y. H. Pao and D. J. Sobajic, "Neural networks and knowledge engineering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 3, pp. 185–192, 1991.

[149] J. Basak, S. Chaudhury, S. K. Pal, and D. Dutta Majumder, "Matching of structural shape descriptions with Hopfield net," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, pp. 377–404, 1993.

[150] J. Basak and S. K. Pal, "Psycop : a psychologically motivated connectionist system for object perception," *IEEE Transactions on Neural Networks (to appear)*.

[151] J. Basak, B. Chanda, and D. Dutta Majumder, "On edge and line linking in graylevel images with connectionist models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, pp. 413–428, 1994.

[152] A. Ghosh, N. R. Pal, and S. K. Pal, "Image segmentation using a neural network," *Biological Cybernetics*, vol. 66, pp. 151–158, 1991.

[153] A. Ghosh and S. K. Pal, "Neural network, self-organization and object extraction," *Pattern Recognition Letters*, vol. 13, pp. 387–397, 1992.

[154] W. Y. Huang and R. P. Lippmann, "Neural net and traditional classifiers," in *Neural Information Processing Systems*, (D. Z. Anderson, ed.), pp. 387–396, New York: American Institute of Physics, 1988.

[155] E. Barnard and D. Casasent, "A comparison between criterion functions for linear classifiers, with an application to neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, pp. 1030–1041, 1989.

[156] D. G. Bounds, P. J. Lloyd, and B. G. Mathew, "A comparison of neural network and other pattern recognition approaches to the diagnosis of low back disorders," *Neural Networks*, vol. 3, pp. 583–591, 1990.

[157] C. H. Chen, "On the relationships between statistical pattern recognition and artificial neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, pp. 655–661, 1991.

[158] I. K. Sethi and A. K. Jain, eds., *Artificial Neural Networks and Statistical Pattern Recognition : Old and New Connections.* Amsterdam: North Holland, 1991.

[159] R. H. Silverman and A. S. Noetzel, "Image processing and pattern recognition in ultrasonograms by backpropagation," *Neural Networks*, vol. 3, pp. 593–603, 1990.

[160] M. A. Franzini, "Speech recognition with back propagation," in *Proceedings of 9th Annual Conference of the Engineering in Medicine and Biology Society*, pp. 1702–1703, IEEE, 1987.

[161] L. W. Chan and F. Fallside, "An adaptive training algorithm for back propagation networks," *Computer Speech and Language*, vol. 2, pp. 205–218, 1987.

[162] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[163] T. Tollenaere, "Super SAB : fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, pp. 561–573, 1990.

[164] M. G. Bello, "Enhanced training algorithms, and integrated training / architecture selection for multilayer perceptron networks," *IEEE Transactions on Neural Networks*, vol. 3, pp. 864–875, 1992.

[165] L. F. A. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Transactions on Neural Networks*, vol. 3, pp. 899–905, 1992.

[166] S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, and Y. Shimeki, "Reduction of required precision bits for backpropagation applied to pattern recognition," *IEEE Transactions on Neural Networks*, vol. 4, pp. 270–275, 1993.

[167] J. N. Hwang, J. J. Choi, S. Oh, and R. J. Marks II, "Query-based learning applied to partially trained multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 2, pp. 131–136, 1991.

[168] H. Drucker and Y. L. Cun, "Improving generalization performance using double backpropagation," *IEEE Transactions on Neural Networks*, vol. 3, pp. 991–997, 1992.

[169] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, (D. S. Touretzky, ed.), pp. 524–532, Los Altos, CA: Morgan-Kaufmann, 1990.

[170] J. Sietsma and R. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, pp. 67–79, 1991.

[171] S. Geva and J. Sitte, "A constructive method for multivariate function approximation by multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 3, pp. 621–624, 1992.

[172] H. -U. Bauer and K. R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 570–579, 1992.

[173] R. Zollner, H. J. Schmitz, and F. W, "Fast generating algorithm for a general three-layer perceptron," *Neural Networks*, vol. 5, pp. 771–777, 1992.

[174] S. G. Smyth, "Designing multilayer perceptrons from nearest-neighbor systems," *IEEE Transactions on Neural Networks*, vol. 3, pp. 329–333, 1992.

[175] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[176] K. Funahashi, "On the approximate realization of continuous mapping by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[177] A. Lapedes and R. Farber, "How neural nets work," in *Neural Information Processing Systems*, (D. Anderson, ed.), pp. 442–456, American Institute of Physics, 1988.

[178] E. D. Sontag, "Feedback stabilization using two-hidden-layer nets," *IEEE Transactions on Neural Networks*, vol. 3, pp. 981–990, 1992.

[179] D. R. Hush, B. Horne, and J. M. Salas, "Error surfaces for multilayer perceptrons," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 1152–1161, 1993.

[180] H. Ritter and K. Schulten, "Convergence properties of Kohonen's topology conserving maps : fluctuations, stability, and dimension selection," *Biological Cybernetics*, vol. 60, pp. 59–71, 1988.

[181] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Transactions on Neural Networks*, vol. 4, pp. 136–141, 1993.

[182] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Transactions on Neural Networks*, vol. 1, pp. 296–298, 1990.

[183] G. J. Gibson and C. F. N. Cowan, "On the decision regions of multilayer perceptrons," *Proceedings of the IEEE*, vol. 78, pp. 1590–1594, 1990.

[184] I. D. Longstaff and J. F. Cross, "A pattern recognition approach to understanding the multi-layer perceptron," *Pattern Recognition Letters*, vol. 5, pp. 315–319, 1987.

[185] T. D. Sanger, "Optimal hidden units for two-layer nonlinear feedforward neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, pp. 545–561, 1991.

[186] J. J. Shynk and N. J. Bershad, "Stationary points of a single-layer perceptron for nonseparable data models," *Neural Networks*, vol. 6, pp. 189–202, 1993.

[187] E. Barnard and E. C. Botha, "Back-propagation uses prior information efficiently," *IEEE Transactions on Neural Networks*, vol. 4, pp. 794–802, 1993.

[188] C. Mead and M. Ismail, eds., *Analog VLSI Implementation of Neural Systems*. Boston, MA: Kluwer Academic, 1989.

[189] U. Ramacher and U. Ruckert, eds., *VLSI Design of Neural Networks*. Boston: Kluwer Academic Press, 1991.

[190] D. Ruwisch, M. Bode, and H. G. Purwins, "Parallel hardware implementation of Kohonen's algorithm with an active medium," *Neural Networks*, vol. 6, pp. 1147–1157, 1993.

[191] "Special issue on neural network hardware design," *IEEE Transactions on Neural Networks*, vol. 3, 1992.

[192] E. J. H. Kerckhoffs, F. W. Wedman, and E. E. E. Frietman, "Speeding up back-propagation training on a hypercube computer," *Neurocomputing*, vol. 4, pp. 43–63, 1992.

[193] K. Obermayer, H. Ritter, and K. Schulten, "Large-scale simulations of self-organizing neural networks on parallel computers : applications to biological modelling," *Parallel Computing*, vol. 14, pp. 381–404, 1990.

[194] H. F. Yin and P. Liang, "A connectionist incremental expert system combining production systems and associative memory," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, pp. 523–544, 1991.

[195] K. Saito and R. Nakano, "Medical diagnostic expert system based on PDP model," in *Proceedings of IEEE International Conference on Neural Networks*, (San Diego, USA), pp. I.255–I.262, 1988.

[196] L. M. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 173–182, 1993.

[197] J. C. Bezdek, "A review of probabilistic, fuzzy, and neural models for pattern recognition," *Journal of Intelligent and Fuzzy Systems*, vol. 1, pp. 1–25, 1993.

[198] B. Kosko, *Neural Networks and Fuzzy Systems*. New Jersey: Prentice Hall, 1991.

[199] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART : fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.

[200] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP : a neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.

[201] S. K. Pal and A. Ghosh, "Neuro-fuzzy image processing : relevance and feasibility," in *Neural and Fuzzy Systems : The Emerging Science of Intelligence and Computing*, (S. Mitra, W. Kraske, and M. M. Gupta, eds.), New York: SPIE Press, 1993.

[202] J. K. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 693–699, 1985.

[203] B. R. Kammerer, "Incorporating uncertainty in neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 6, pp. 179–192, 1992.

[204] E. Sanchez, "Fuzzy connectionist expert systems," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 31–35, 1990.

[205] T. L. Huntsberger and P. Ajjimarangsee, "Parallel self-organizing feature maps for unsupervised pattern recognition," *International Journal of General Systems*, vol. 16, pp. 357–372, 1990.

[206] J. C. Bezdek, E. C. Tsao, and N. R. Pal, "Fuzzy Kohonen clustering networks," in *Proceedings of 1st IEEE International Conference on Fuzzy Systems*, (San Diego, USA), pp. 1035–1043, 1992.

[207] J. M. Keller, R. R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets and Systems*, vol. 45, pp. 1–12, 1992.

[208] H. Takagi and I. Hayashi, "Artificial neural network driven fuzzy reasoning," *International Journal of Approximate Reasoning*, vol. 5, pp. 191–212, 1991.

[209] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their applications," *IEEE Transactions on Neural Networks*, vol. 3, pp. 752–760, 1992.

[210] H. Ishibuchi and H. Tanaka, "Identification of real-valued and interval-valued membership functions by neural networks," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 179–182, 1990.

[211] S. C. Newton, S. Pemmaraju, and S. Mitra, "Adaptive fuzzy leader clustering of complex data sets in pattern recognition," *IEEE Transactions on Neural Networks*, vol. 3, pp. 794–800, 1992.

[212] P. K. Simpson, "Fuzzy min-max neural networks : 1.Classification," *IEEE Transactions on Neural Networks*, vol. 3, pp. 776–786, 1992.

[213] P. K. Simpson, "Min-max neural networks : 2.Clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, pp. 32–45, 1993.

[214] R. Krishnapuram and J. Lee, "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks*, vol. 5, pp. 335–350, 1992.

[215] J. M. Keller, R. Krishnapuram, and F. C. -H. Rhee, "Evidence aggregation networks for fuzzy logic inference," *IEEE Transactions on Neural Networks*, vol. 3, pp. 761–769, 1992.

[216] W. Pedrycz, "Neurocomputations in relational systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 289–297, 1991.

[217] W. Pedrycz, "Fuzzy neural networks with reference neurons as pattern classifiers," *IEEE Transactions on Neural Networks*, vol. 3, pp. 770–775, 1992.

[218] J. M. Keller, Y. Hayashi, and Z. Chen, "Interpretation of nodes in networks for fuzzy logic," in *Proceedings of 2nd IEEE International Conference on Fuzzy Systems*, (San Francisco, USA), pp. 1203–1207, 1993.

[219] S. C. Lee and E. T. Lee, "Fuzzy neural networks," *Mathematical Biosciences*, vol. 23, pp. 151–177, 1975.

[220] T. Yamakawa, E. Uchino, T. Miki, and H. Kusanagi, "A neo fuzzy neuron and its application to system identification and prediction of the system behaviour," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 477–483, 1992.

[221] A. Ghosh, N. R. Pal, and S. K. Pal, "Self-organization for object extraction using multilayer neural network and fuzziness measures," *IEEE Transactions on Fuzzy Systems*, vol. 1, pp. 54–68, 1993.

[222] W. Pedrycz, "Relational structures in fuzzy sets and neurocomputations," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 235–238, 1990.

[223] W. Pedrycz, "A referential scheme of fuzzy decision making and its neural network structure," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 1593–1604, 1991.

[224] T. Watanabe, M. Matsumoto, and T. Hasegawa, "A layered neural network model using logic neurons," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 675–678, 1990.

[225] H. -J. Zimmermann and P. Zysno, "Decisions and evaluations by hierarchical aggregation of information," *Fuzzy Sets and Systems*, vol. 10, pp. 243–260, 1983.

[226] K. Hirota and W. Pedrycz, "Or/And neuron in modeling fuzzy set connectives," *IEEE Transactions on Fuzzy Systems*, vol. 2, pp. 151–161, 1994.

[227] H. Takagi, "Fusion technology of fuzzy theory and neural network - Survey and future directions," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 13–26, 1990.

[228] J. M. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks," *International Journal of Approximate Reasoning*, vol. 6, pp. 221–240, 1992.

[229] H. Ishibuchi, H. Okada, and H. Tanaka, "Interpolation of fuzzy if-then rules by neural networks," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 337–340, 1992.

[230] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy If-Then rules," *IEEE Transactions on Fuzzy Systems*, vol. 1, pp. 85–97, 1993.

[231] H. -J. Zimmermann and P. Zysno, "Latent connectives in human decision making," *Fuzzy Sets and Systems*, vol. 4, pp. 37–51, 1980.

[232] S. Nakanishi and T. Takagi, "Pattern recognition by neural networks and fuzzy inference," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 183–186, 1990.

[233] W. P. Zhuang, W. Z. Qiao, and T. H. Heng, "The truth-valued flow inference network," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 267–281, 1990.

[234] J. Yen, "The role of fuzzy logic in the control of neural networks," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 771–774, 1990.

[235] R. R. Yager, "Implementing fuzzy logic controllers using a neural network framework," *Fuzzy Sets and Systems*, vol. 48, pp. 53–64, 1992.

[236] H. Isshiki and H. Endo, "Learning of expert's knowledge by neural network and deduction of fuzzy rules by Powell's method," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 95–98, 1992.

[237] A. F. Rocha, "K-Neural nets and expert reasoning," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 143–146, 1990.

[238] A. F. Rocha, M. Theoto, and P. Torasso, "Heuristic learning expert systems - general principles," in *Fuzzy Logic in Knowledge-Based Systems, Decision and Control*, (M. M. Gupta and T. Yamakawa, eds.), pp. 289–306, North-Holland: Elsevier Science Publishers B. V., 1988.

[239] Y. Hayashi, J. J. Buckley, and E. Czogala, "Approximation between fuzzy expert systems and neural networks," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 135–139, 1992.

[240] P. V. Krishnamraju, K. D. Reilly, and Y. Hayashi, "Neural and conventional expert systems : competitive and cooperative schemes," in *Proceedings of 2nd Workshop on Neural Networks : Academic / Industrial / NASA / Defense*, (Auburn, Alabama, USA), pp. 649–656, 1991.

[241] K. Yoshida, Y. Hayashi, A. Imura, and N. Shimada, "Fuzzy neural expert system for diagnosing hepatobiliary disorders," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 539–543, 1990.

[242] Y. Hayashi, "A neural expert system with automated extraction of fuzzy if-then rules and its application to medical diagnosis," in *Advances in Neural Information Processing Systems*, (R. P. Lippmann, J. E. Moody, and D. S. Touretzky, eds.), pp. 578–584, Los Altos: Morgan Kaufmann, 1991.

[243] Y. Hayashi, "Neural expert system using fuzzy teaching input and its application to medical diagnosis," *Information Sciences Applications*, vol. 1, pp. 47–58, 1994.

[244] D. L. Hudson, M. E. Cohen, and M. F. Anderson, "Use of neural network techniques in a medical expert system," *International Journal of Intelligent Systems*, vol. 6, pp. 213–223, 1991.

[245] D. L. Hudson and M. E. Cohen, "Combination of rule-based and connectionist expert systems," *International Journal of Microcomputer Applications*, vol. 10, pp. 36–41, 1991.

[246] M. E. Cohen, D. L. Hudson, and M. F. Anderson, "A neural network learning algorithm with medical applications," in *Computer Applications in Medical Care*, (L. C. Kingsland, ed.), pp. 307–311, IEEE Computer Society Press, 1989.

[247] S. G. Romaniuk and L. O. Hall, "Decision making on creditworthiness, using a fuzzy connectionist model," *Fuzzy Sets and Systems*, vol. 48, pp. 15–22, 1992.

[248] F. C. H. Rhee and R. Krishnapuram, "Fuzzy rule generation methods for high-level computer vision," *Fuzzy Sets and Systems*, vol. 60, pp. 245–258, 1993.

[249] R. Masuoka, N. Watanabe, A. Kawamura, Y. Owada, and K. Asakawa, "Neuro-fuzzy system – fuzzy inference using a structured neural network," in *Proceed-*

ings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka, (Japan), pp. 173–177, 1990.

[250] H. Okada, N. Watanabe, A. Kawamura, K. Asakawa, T. Taira, K. Ishida, T. Kaji, and M. Narita, "Knowledge implementation multilayer neural networks with fuzzy logic," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 99–102, 1992.

[251] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," *International Journal of Approximate Reasoning*, vol. 2, pp. 377–393, 1988.

[252] R. J. Machado and A. F. Rocha, "A hybrid architecture for connectionist expert systems," in *Intelligent Hybrid Systems*, (A. Kandel and G. Langholz, eds.), CRC Press, 1992.

[253] R. J. Machado and A. F. da Rocha, "Inference, inquiry and explanation in expert systems by means of fuzzy neural networks," in *Proceedings of 2nd IEEE International Conference on Fuzzy Systems*, (San Francisco, USA), pp. 351–356, 1993.

[254] B. F. Leao and A. F. Rocha, "Proposed methodology for knowledge acquisition : a study on congenital heart disease diagnosis," *Methods of Information in Medicine*, vol. 29, pp. 30–40, 1990.

[255] W. Pedrycz and A. F. Rocha, "Fuzzy-set based models of neurons and knowledge-based networks," *IEEE Transactions on Fuzzy Systems*, vol. 1, pp. 254–266, 1993.

[256] K. Hirota and W. Pedrycz, "Knowledge-based networks in classification problems," *Fuzzy Sets and Systems*, vol. 59, pp. 271–279, 1993.

[257] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[258] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[259] *Proc. of Fourth International Conference on Genetic Algorithms*, (University of California, San Diego, USA), 1991.

[260] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Transactions on Neural Networks*, vol. 3, pp. 324–328, 1992.

[261] H. Shehadeh and R. N. Lea, "A genetic algorithms approach for altering the membership functions in fuzzy logic controllers," in *Proceedings of the North American Fuzzy Logic Processing Society (NAFIPS '92)*, (Mexico), pp. 515–523, 1992.

[262] S. Bornholdt and D. Graudenz, "General asymmetric neural networks and structure design by genetic algorithms," *Neural Networks*, vol. 5, pp. 327–334, 1992.

[263] S. K. Sin and R. J. P. de Figueiredo, "An evolution-oriented learning algorithm for the optimal interpolative net," *IEEE Transactions on Neural Networks*, vol. 3, pp. 315–323, 1992.

[264] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks : optimizing connections and connectivity," *Parallel Computing*, vol. 14, pp. 347–361, 1990.

[265] H. Muhlenbein, "Limitations of multi-layer perceptron networks - step towards genetic neural networks," *Parallel Computing*, vol. 14, pp. 249–260, 1990.

[266] S. K. Pal and D. Bhandari, "Selection of optimum set of weights in a layered network using genetic algorithms," *Information Sciences (to appear)*.

[267] R. J. Machado and A. F. da Rocha, "Evolutive fuzzy neural networks," in *Proceedings of 1st IEEE International Conference on Fuzzy Systems*, (San Diego, USA), pp. 493–500, 1992.

[268] E. Sanchez, "Genetic algorithms, neural networks and fuzzy logic systems," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 17–19, 1992.

[269] K. S. Leung, Y. Leung, L. So, and K. F. Yam, "Rule learning in expert systems using genetic algorithm : 1, Concepts; 2, Empirical studies," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 201–208, 1992.

[270] S. K. Pal, D. Bhandari, P. Harish, and M. K. Kundu, "Cellular neural networks, genetic algorithms and object extraction," *Far East Journal of Mathematical Sciences*, vol. 1, pp. 139–155, 1993.

[271] S. K. Pal and D. Bhandari, "Genetic algorithms with fuzzy fitness function for object extraction using cellular neural networks," *Fuzzy Sets and Systems (to appear)*.

[272] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, pp. 539–567, 1993.

[273] S. K. Pal and S. Mitra, "Multi-layer perceptron, fuzzy sets and classification," *IEEE Transactions on Neural Networks*, vol. 3, pp. 683–697, 1992.

[274] S. Mitra and S. K. Pal, "Self-organizing neural network as a fuzzy classifier," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 3, pp. 385–399, 1994.

[275] S. K. Pal and S. Mitra, "Fuzzy versions of Kohonen's net and MLP-based classification : Performance evaluation for certain nonconvex decision regions," *Information Sciences*, vol. 76, pp. 297–337, 1994.

[276] S. Mitra and S. K. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, 1994 (to appear).

[277] S. Mitra and S. K. Pal, "Logical operation based fuzzy MLP for classification and rule generation," *Neural Networks*, vol. 7, pp. 353–373, 1994.

[278] S. Mitra and S. K. Pal, "Layered neural net as a fuzzy classifier," in *Proceedings of the 4th International Conference of AI and Expert Systems, Hawaii*, (USA), pp. 128–137, 1991.

[279] S. Mitra and S. K. Pal, "Rule generation and inferencing with a layered fuzzy neural network," in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 641–644, 1992.

[280] S. Mitra and S. K. Pal, "Fuzzy self organization, inferencing and rule generation," *IEEE Transactions on Systems, Man and Cybernetics*, (to appear).

[281] S. Mitra and S. K. Pal, "Fusion of fuzzy sets and layered neural networks at the input, output and neuronal levels," *Indian Journal of Pure and Applied Mathematics*, vol. 24, pp. 121–133, 1994.

[282] S. Mitra and S. K. Pal, "Neuro-fuzzy expert systems : overview with a case study," in *Fuzzy Reasoning in Information, Decision and Control Systems*, (S. G. Tzafestas and A. N. Venetsanopoulos, eds.), pp. 121–143, Boston/Dordrecht: Kluwer Academic Publishers, 1994.

[283] S. Mitra, "Fuzzy MLP based expert system for medical diagnosis," *Fuzzy Sets and Systems*, vol. 64, 1994 (to appear).

[284] G. E. Hinton, "Learning translation invariant recognition in a massively parallel network," in *PARLE : Parallel Architectures and Languages Europe*, (G. Goos and J. Hartmanis, eds.), pp. 1–13, Berlin: Springer-Verlag, 1987.

[285] S. K. Pal and P. K. Pramanik, "Fuzzy measures in determining seed points in clustering," *Pattern Recognition Letters*, vol. 4, pp. 159–164, 1986.

[286] S. Becker and Y. le Cun, "Improving the convergence of back-propagation learning with second order methods," Tech. Rep. CRG-TR-88-5, Connectionist Research Group, University of Toronto, Canada, 1988.

[287] G. E. Hinton, "Learning distributed representation of concepts," in *Proceedings of 8th Annual Conference of the Cognitive Science Society*, (Amherst, Mass., USA), pp. 1–12, 1986.

[288] D. C. Plaut and G. E. Hinton, "Learning sets of filters using backpropagation," *Computer Speech and Language*, vol. 2, pp. 35–61, 1987.

[289] S. K. Pal, *Studies on the application of fuzzy set-theoretic approach in some problems of pattern recognition and man-machine communication by voice.* PhD thesis, University of Calcutta, Calcutta, India, 1978.

[290] J. M. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Information Sciences*, vol. 62, pp. 205–221, 1992.

[291] W. Pedrycz and H. C. Card, "Linguistic interpretation of self-organizing maps," in *Proceedings of 1st IEEE International Conference on Fuzzy Systems*, (San Diego, USA), pp. 371–378, 1992.

[292] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biological Cybernetics*, vol. 61, pp. 241–254, 1989.

[293] H. Ichiki, M. Hagiwara, and M. Nakagawa, "Kohonen feature maps as a supervised learning machine," in *Proceedings of IEEE International Joint Conference on Neural Networks*, (San Francisco, USA), pp. 1944–1948, 1993.

[294] T. Fritsch, P. H. Kraus, H. Przuntek, and P. Tran-Gia, "Classification of Parkinson rating-scale-data using a self-organizing neural net," in *Proceedings of IEEE International Joint Conference on Neural Networks*, (San Francisco, USA), pp. 93–98, 1993.

[295] T. Kohonen, "Analysis of a simple self-organizing process," *Biological Cybernetics*, vol. 44, pp. 135–140, 1982.

[296] H. Ritter and K. Schulten, "On the stationary state of Kohonen's self-organizing sensory mapping," *Biological Cybernetics*, vol. 54, pp. 99–106, 1986.

[297] Y. Qi, B. R. Hunt, and N. Bi, "The use of fuzzy membership in network training for isolated word recognition," in *Proceedings of IEEE International Joint Conference on Neural Networks*, (San Francisco, USA), pp. 1823–1828, 1993.

[298] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 28–44, 1973.

[299] I. B. Turksen, "Interval valued fuzzy sets based on normal forms," *Fuzzy Sets and Systems*, vol. 20, pp. 191–210, 1986.

[300] Y. Hayashi, "Personal communication," 1993, on medical data.

[301] D. K. Biswas, *Study of Kala-azar with special reference to renal function.* Master's thesis, School of Tropical Medicine, University of Calcutta, Calcutta, India, 1989.

LIST OF PUBLICATIONS OF THE AUTHOR

1. "Graphics tool for the representation and analysis of Petri nets", in *Proceedings of Seminar on Parallel Processing Systems and their Applications*, (Calcutta, India), pp. 92-95, 1988. [Co-Author : B. Saha]

2. "Implementation of fault simulation and testing of combinational circuits", *Intl. Journal of Electronics*, Vol. 66, pp. 665-678, 1989. [Co-Author : B. Saha]

3. "Fuzzy dynamic clustering algorithm by optimisation of ambiguity", in *Proceedings of National Conference on Electronic Circuits and Systems*, (Roorkee, India), pp. 361-363, 1989. [Co-Author : S. K. Pal]

4. "Fuzzy dynamic clustering algorithm", *Pattern Recognition Letters*, Vol. 11, pp. 525-535, 1990. [Co-Author : S. K. Pal]

5. "Layered neural net as a fuzzy classifier", in *Proceedings of the 4th International Conference of Artificial Intelligence and Expert Systems*, (Hawaii, USA), pp. 128-137, 1991. [Co-Author : S. K. Pal]

6. "Multi-layer perceptron, fuzzy sets and classification", *IEEE Transactions on Neural Networks*, Vol. 3, pp. 683-697, 1992.[1] [Co-Author : S. K. Pal]

7. "Rule generation and inferencing with a layered fuzzy neural network", in *Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks*, (Iizuka, Japan), pp. 641-644, 1992. [Co-Author : S. K. Pal]

8. "Fusion of fuzzy sets and layered neural networks at the input, output and neuronal levels", *Indian Journal of Pure and Applied Mathematics*, Vol. 24, pp. 121-133, 1994. [Co-Author : S. K. Pal]

9. "Fuzzy versions of Kohonen's net and MLP-based classification : Performance evaluation for certain nonconvex decision regions", *Information Sciences*, Vol. 76, pp. 297-337, 1994. [Co-Author : S. K. Pal]

10. "Logical operation based fuzzy MLP for classification and rule generation", *Neural Networks*, Vol. 7, pp. 353-373, 1994. [Co-Author : S. K. Pal]

---

[1]Received the 1994 IEEE Trans. on Neural Networks Outstanding Paper Award

11. "Self-organizing neural network as a fuzzy classifier", *IEEE Transactions on Systems, Man and Cybern.*, Vol. 24, pp. 385-399, 1994. [Co-Author : S. K. Pal]

12. "Fuzzy MLP based expert system for medical diagnosis", *Fuzzy Sets and Systems*, Vol. 64, 1994 (to appear).

13. "Fuzzy multi-layer perceptron, inferencing and rule generation", *IEEE Transactions on Neural Networks*, Vol. 6, No. 5, 1994 (to appear). [Co-Author : S. K. Pal]

14. "Neuro-Fuzzy Expert Systems : Overview with a Case Study", in *Fuzzy Reasoning in Information, Decision and Control Systems*, (S. G. Tzafestas and A. N. Venetsanopoulos, eds.), Boston/ Dordrecht : Kluwer Academic Publishers, pp. 121-143, 1994. [Co-Author : S. K. Pal]

15. "A two-level classification scheme trained by a fuzzy neural network", in *Proceedings of 12th International Conference on Pattern Recognition*, (Jerusalem, Israel), pp. 467-469, 1994. [Co-Author : L. I. Kuncheva]

16. "Change-glasses pattern classification with a fuzzy neural network", in *Proceedings of 1st Workshop on Fuzzy Based Expert Systems*, (Sofia, Bulgaria), pp. 29-32, 1994. [Co-Author : L. I. Kuncheva]

17. "Improving classification performance using fuzzy MLP and two-level selective partitioning of the feature space", *Fuzzy Sets and Systems*, (to appear). [Co-Author : L. I. Kuncheva]

18. "Fingerprint classification using a fuzzy multilayer perceptron", *Neural Computing and Applications*, Vol. 4, 1994 (to appear). [Co-Authors : S. K. Pal and M. K. Kundu]

19. "Fuzzy self organization, inferencing and rule generation", *IEEE Transactions on Systems, Man and Cybernetics*, (to appear). [Co-Author : S. K. Pal]