FEATURE EVALUATION, CLASSIFICATION AND RULE GENERATION USING FUZZY SETS AND NEURAL NETWORKS

Rajat Kumar De

Machine Intelligence Unit

Indian Statistical Institute

203, B. T. Road

Calcutta 700035

India.

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

Doctor of Philosophy

1999

To my Parents

ACKNOWLEDGEMENTS

Words seem insufficient to express my gratitude and indebtedness to *Prof. Sankar K. Pal*, who has not only supervised my dissertation work, but also acted as an elder brother during the tenure of my research. I owe a lot to him for providing me constant encouragement and affection throughout the last few years. I also express my sincere gratitude to *Dr. Samar SenSarma*, Department of Computer Science, Calcutta University, Calcutta, India, for helping me a lot in various ways.

Heartfelt thanks are due to *Dr. Jayanta Basak*, *Prof. Nikhil R. Pal* and *Dr. Sushmita Mitra*. I consider this thesis as a kind of joint venture between myself, *Dr. Jayanta Basak*, *Prof. Nikhil R. Pal*, *Dr. Sushmita Mitra* and *Prof. Sankar K. Pal*. Thanks are due to them for their kind permission to include the joint research work in this thesis.

Special note of thanks to Dr. C. A. Murthy, Dr. M. K. Kundu, Mr. B. UmaShankar, Dr. D. P. Mandal, Mr. S. N. Biswas, Dr. A. Ghosh. I also thank Mr. Suman K. Mitra, Mr. Sitabhra Sinha, Ms. Sanghamitra Bandyopadhyay, Mr. Pabitra Mitra, Mr. Partha S. Bhattacharya, Dr. Suptendra N. Sarbadhikari, Ms. Susmita Ghosh (nee De), Ms. Mousumi Acharyya, Ms. Arunima Banerjee, Dr. Kuhu Pal, Ms. Swati Chowdhury. I acknowledge the office staff of Machine Intelligence Unit for providing a friendly environment, the workers in the reprography unit for careful photocopying, CSSC for providing computing facilities, Mr. S. Chakraborty for drawing some of the diagrams and the authorities of ISI for extending various facilities.

I take this opportunity to express heartfelt gratitude to my teachers and friends at the Department of Computer Science, Calcutta University and Jadavpur University, Calcutta, India. Thanks are due to all my friends for constant encouragement.

I shall forever remain indebted to my parents, wife, brother and of course not the least to my newly born son. Heartfelt gratitude are also due to my in-laws and relatives. It is their constant encouragement, enthusiasm and support that has helped me throughout my academic career and specially during the research work.

Finally, I acknowledge the Department of Atomic Energy and Council of Scientific and Industrial Research, Government of India, for providing me financial support as the forms of a Dr. K. S. Krishnan Senior Research Fellowship and Research Associateship respectively.

ISI, Calcutta
March 1999.

Rajat Kumar De
(Rajat K. De)

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

Pattern recognition and machine learning form a major area of research and development activity that encompasses the processing of pictorial and other non-numerical information obtained from the interaction between science, technology and society. A motivation for the spurt of activity in this field is the need for people to communicate with the computing machines in their natural mode of communication. Another important motivation is that the scientists are also concerned with the idea of designing and making intelligent machines that can carry out certain tasks that we human beings do. The most salient outcome of these is the concept of future generation computing systems.

Machine recognition 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 that a new sample is a possible member of the class by noting that it has properties common to those of the set of samples. The task of pattern recognition by a computer can be described as a transformation from the measurement space $\mathcal{M}$ to the feature space $\mathcal{F}$ and finally to the decision space $\mathcal{D}$, i.e.,

$$\mathcal{M} \rightarrow \mathcal{F} \rightarrow \mathcal{D}.$$

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

When the input pattern is an image, some processing tasks such as enhancement, filtering, noise reduction, contour extraction and skeleton extraction are performed in the measurement space, in order to extract salient features from the image pattern. This is what is basically known as image processing [189, 70]. The ultimate aim is to make its understanding, recognition and interpretation from the processed information available from the image pattern. Such a complete image recognition/interpretation scheme is called a vision system [5] which may be viewed as consisting of three levels, *viz.*, low level, mid level and high level corresponding to $\mathcal{M}$, $\mathcal{F}$ and $\mathcal{D}$ with an extent of overlapping among them.

The theory of fuzzy set has been introduced in 1965 by Zadeh [226] as a new way of representing vagueness in everyday life. This theory [228, 232, 100, 156, 109, 110, 222, 216, 178] 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 behavior, 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 judgment, 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 to pattern recognition and image processing problems has been adequately reported in literature [20, 22, 99, 75, 51, 103, 227, 196, 166]. 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 the uncertainty in a recognition system.

Artificial neural networks (ANN) [193, 127, 112, 111, 72, 37, 205, 81, 97, 138] are signal processing systems that try to emulate the behavior 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 (called neurons) 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 [78], where a local learning rule was proposed. This rule assumes that correlations between the states of two neurons determine the strength of the coupling between them. Thereby, a synaptic connection that is very active grows 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 simplification 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 [193, 112, 72, 168] to pattern recognition are attempts to achieve these goals. In short, neural networks are natural classifiers having resistance to noise, tolerance to distorted patterns/images (ability to generalize), superior ability to recognize partially occluded or degraded images/overlapping pattern classes or classes with highly nonlinear boundaries, and potential for parallel processing. 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 [63]. They use the set of connection weights of a *trained* neural net for encoding the knowledge base and generating rules 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 decision making problems. For the last ten years or so, there have been several attempts [22, 168, 62, 96] 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, both theoretical and ex-

perimental, demonstrating the effectiveness of fuzzy set theoretic, connectionist and neuro-fuzzy approaches to certain tasks of pattern recognition, namely, feature evaluation (selection and extraction), classification and rule generation. Some new connectionist models have been developed for performing these tasks. The task of feature evaluation is performed under both supervised and unsupervised learning. For the purpose of classification, neuro-fuzzy case-based and knowledge-based systems have been designed. The knowledge-based network is also used to extract linguistic rules in *If-Then* form. The effectiveness of these methodologies along with extensive comparisons with related algorithms has been demonstrated on various synthetic as well as real life recognition problems. Before we describe the scope of the thesis, we provide a brief review on feature evaluation, classification and rule generation in the aforesaid framework.

Section 1.2 presents a description of the basic concept, features and techniques of pattern recognition briefly along with some applications. Fuzzy sets and their relevance to pattern recognition are described in Section 1.3. A review on fuzzy set theoretic approaches to pattern recognition is also provided in this chapter. Section 1.4 describes the basic principles and features of artificial neural networks, and its relevance to pattern recognition along with a review on its various approaches. Some of the attempts using knowledge-based networks are also mentioned here. An overview of neuro-fuzzy methodologies along with neuro-fuzzy knowledge-based systems for pattern recognition is described in Section 1.5. A brief account of case-based reasoning is provided in Section 1.6. Finally, Section 1.7 discusses the scope of the thesis.

1.2 Overview of Pattern Recognition

A typical pattern recognition system consists of three phases as shown in Fig. 1.1. These are *data acquisition, feature selection/extraction* and *classification/clustering*. In the data acquisition phase, depending on the environment within which the objects are to be classified/clustered, data are gathered using a set of sensors. These are then passed on to the feature selection/extraction phase, where the dimensionality of the data is reduced by retaining/measuring only some characteristic features or properties. In a broader perspective, this stage significantly influences

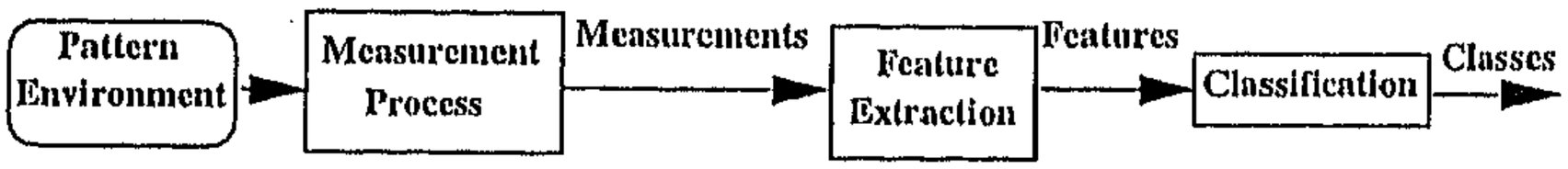

Figure 1.1: A typical pattern recognition system.

the entire recognition process. Finally, in the classification/clustering phase, the selected/extracted features are passed on to the classifying/clustering system that evaluates the incoming information and makes a final decision. This phase basically establishes a transformation between the features and the classes/clusters. Different forms of transformation can be a Bayesian rule of computing *a posterior* class probabilities, nearest neighbor rule, linear discriminant functions, perceptron rule, nearest prototype rule etc. [52, 49, 56, 59, 214, 69, 172]

1.2.1 Data acquisition

Pattern recognition techniques are applicable in a wide domain, where the data may be qualitative, quantitative, or both; they may be numerical, linguistic, pictorial, or any combination thereof. The collection of data constitutes data acquisition phase. Generally, the data structures that are used in pattern recognition systems are of two types : *object data vectors* and *relational data*. Object data, set of numerical vectors, are represented in the sequel as $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_s\}$, a set of s feature vectors in the n-dimensional measurement space Ω_Y. A pth object, $p = 1, 2, \ldots, s$, observed in the process has vector $\mathbf{y}_p$ as its numerical representation; y_{pi} is an ith, $(i = 1, 2, \ldots, n)$ feature value associated with the pth object. Relational data is a set of s^2 numerical relationships, say $\{r_{pq}\}$, between pairs of objects. In other words, r_{pq} represents the extent to which pth and qth objects are related in the sense of some binary relationship ρ. If the objects that are pairwise related by ρ are called $O = \{o_1, o_2, \ldots, o_s\}$, then $\rho : O \times O \rightarrow I\!R$.

1.2.2 Feature selection/extraction

Feature selection/extraction is a process of selecting a map of the form $X = f(Y)$, by which a sample $\mathbf{y}$ $(=[y_1, y_2, \ldots, y_n])$ in an n-dimensional measurement space Ω_Y is transformed into a point $\mathbf{x}$ $(=[x_1, x_2, \ldots, x_{n'}])$ in an n'-dimensional feature space Ω_X, where $n' < n$. The main objective of this task or feature evaluation [15, 49], is to retain/develop the optimum salient characteristics necessary for the recognition process and to reduce the dimensionality of the measurement space Ω_Y so that effective and easily computable algorithms can be devised for efficient classification. The problem of feature selection/extraction has two aspects – formulation of a suitable criterion to evaluate the goodness of a feature, and deciding on its retention in the optimal set. In general, those features are considered to have optimal saliencies for which interclass/intraclass distances are maximized/minimized. The criterion of a good feature is that it should be unchanging with any other possible variation within a class, while emphasizing differences that are important in discriminating between patterns of different types.

The major mathematical measures so far devised for the estimation of feature quality are mostly statistical in nature, and can be broadly classified into two categories – *feature selection in the measurement space* and *feature selection in a transformed space*. The techniques in the first category generally reduce the dimensionality of the measurement space by discarding redundant or least information carrying features. On the other hand, those in the second category utilize all the information contained in the measurement space to obtain a new transformed space; thereby mapping a higher dimensional pattern to a lower dimensional one. This is referred to as feature extraction.

The pioneering research on feature selection mostly deals with statistical tools. Later, the thrust of the research shifted to the development of various other approaches to feature selection, including fuzzy and neural approaches [156, 152, 191]. Each approach has its own advantages and drawbacks. In Appendix A, we describe an algorithm [49] based on statistical approach for feature selection which has been used for comparing the performance of one of our feature selection algorithms. Some of the attempts made using fuzzy set theoretic and connectionist approaches are provided in Sections 1.3.2 and 1.4.2 respectively.

1.2.3 Classification

The problem of classification is basically one of partitioning the feature space into regions, one region for each category of input. Thus it attempts to assign every data point in the entire feature space to one of the possible (say, M) classes. Pattern classification, by its nature, admits many approaches, sometimes complementary, sometimes competing, to the approximate solution of a given problem. These include decision theoretic approach (both deterministic and probabilistic), syntactic approach, connectionist approach, and fuzzy set theoretic approach.

In the decision theoretic approach, once a pattern is transformed, through feature evaluation, 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 [52, 214, 49]. In deterministic classification approach, it is assumed that there exists only one unambiguous pattern class corresponding to each of the unknown pattern vectors. *Nearest neighbor classifier (NN rule)* [52, 59, 214] is an example of this category.

In most of the practical problems, the features are usually noisy and the classes in the feature space are overlapping. In order to model such systems, the features $x_1, x_2, \ldots, x_i, \ldots, x_n$ are considered as random variables in the probabilistic approach. The most commonly used classifier in such probabilistic systems is the *Bayes maximum likelihood classifier* [214]. Both k-NN and Bayes maximum likelihood classifiers have been used in the present investigation for comparison.

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 [56] which deal with the representation of structures via sentences, grammars and automata. In the syntactic method [56, 214], the ability of selecting and classifying the simple pattern primitives and their relationships represented by the composition operations is the vital criterion of making a system effective. Since the techniques of composition of primitives into patterns are usually governed by the formal language theory, the approach is often referred to as linguistic approach. An introduction to a variety of approaches based on this idea can be found in [56].

A good pattern recognition system should possess several characteristics. These

are on-line adaptation (to cope with the changes in the environment), nonlinear separability (to tackle real life problems), handling of overlapping classes/clusters (for discriminating almost similar but different objects), real-time processing (for making a decision in a reasonable time), generation of soft and hard decisions (to make the system flexible), verification and validation mechanisms (for evaluating its performance), and minimizing the number of parameters in the system that have to be tuned (for reducing the cost and complexity). Moreover, the system should be made artificially intelligent in order to emulate some aspects of the human processing system. Connectionist approaches (or artificial neural network based approaches) [193, 127, 168, 81, 37] to pattern recognition are attempts to achieve these goals, and have drawn the attention of researchers because of its major characteristics like adaptivity, robustness/ruggedness, speed and optimality. Since this thesis involves the design of connectionist models for various tasks of pattern recognition, a review on this aspect is provided in Section 1.4.

All these approaches to pattern recognition can again be fuzzy set theoretic [226, 20, 229, 166, 232, 100] in order to handle uncertainties, arising from vague, incomplete, linguistic, overlapping patterns etc., at various stages of pattern recognition systems. Fuzzy set theoretic classification approach is developed based on the realization that a pattern may belong to more than one class, with varying degree of class membership. Accordingly, fuzzy decision theoretic, fuzzy syntactic, fuzzy neural approaches are developed [20, 22, 99, 100, 166, 173]. We have included a detailed review on this approach in Section 1.3 along with its relevance.

There have been several attempts over the last decade to evolve new approaches to pattern recognition and deriving their hybrids by combining the merits of several techniques. An integration of neural network and fuzzy set theories, commonly known as the neuro-fuzzy approach, is one such hybrid paradigm [144, 74, 176, 96, 62] in a soft computing framework. Since this thesis concerns mostly on the neuro-fuzzy approach to certain tasks of pattern recognition, a detailed review on this is provided in Section 1.5.

Application of rule based systems in pattern recognition has also gained popularity in the recent past. By modeling the rules and facts in terms of fuzzy sets, it is possible to make interfaces using the concept of approximate reasoning [147]. An approach to classifying unknown patterns by combining the k-NN rule with Dempster-Shafer

theory of evidence [202] has been formulated in [48].

Recently, some investigations have been made in the area of pattern recognition using genetic algorithms [163, 6]. Like neural networks, genetic algorithms (GAs) [36, 68] 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. Some GA based attempts in the area of feature evaluation and classification can be found in [163, 6, 203, 164, 4], where the efficient and robust searching capabilities of GAs are being exploited in determining various parameters optimally. ♣

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 selector/extractor or classifying/clustering system. Under such circumstances, it is assumed that these samples are representative of the classes. 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 or clustering 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 learning*; otherwise we call it *unsupervised learning*. Supervised learning is used for classifying different objects, while clustering is performed through unsupervised learning.

1.2.4 Applications of pattern recognition

Pattern recognition research is mostly driven by the need to process data and information obtained from the interaction between human society, science and technology. As already mentioned, in order to make machines more intelligent and human like, it must possess automatic pattern recognition capability. Some of the application areas of pattern recognition are :

1. *medicine :* medical diagnosis, image analysis, disease classification;

2. *natural resource study and estimation :* agriculture, forestry, geology, environment;

3. *man-machine communication :* automatic speech recognition and understanding, natural language processing, image processing, script recognition;

4. *vehicular :* automobile, airplane, train, boat controllers;

5. *defense :* automatic target recognition, guidance, control;

6. *police and detective :* crime and criminal detection from analyses of speech, handwriting, fingerprint, photograph;

7. *remote sensing :* detection of man-made objects and estimation of natural resources;

8. *industry :* CAD, CAM, product testing and assembly, inspection, quality control;

9. *domestic systems :* appliances;

1.3 Overview of Fuzzy Models for Pattern Recognition

A fuzzy set A in a set of points $R = \{r\}$ is a class of events with a continuum of grades of membership. It is characterized by a membership function $\mu_A(r)$ which associates a real number $\mu_A(r) \in [0, 1]$ with each element of R. $\mu_A(r)$ at r represents 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 $\mu_i = \mu_A(r_i)$, 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. 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 [229]. 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 [227] 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 [73].

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 [166]. 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.

1.3.1 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 [52, 214, 59], 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 or clusters. 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 or cluster. 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 [131] described in the 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 the sample points. The size of the extended regions should also decrease with the increase in the number of sample points. This leads one to define a multivalued or fuzzy (with continuum grade of belonging) 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 aforementioned examples we see that the concept of fuzzy sets can be used at both the *feature level* and *classification level*. In *feature level*, it is used in representing input data as an array of membership values denoting the degree of possession of certain linguistic properties and in weakening the strong commitments for extracting ill-defined image regions, properties, primitives, and relations among them. At the *classification level*, the said concept is used for representing class membership of objects, and for providing an estimate (or a representation) of missing information in terms of membership values.

1.3.2 Various approaches

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 twofold: (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 evaluation and classification procedures. Bezdek and Pal [22] 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 [173].

Research on the application of fuzzy set theory to supervised pattern recognition has been started in 1966 in the seminal note of Bellman, Kalaba and Zadeh [17] 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. Pal and Dutta Majumder [157] have outlined 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.

Although the task of feature selection plays an important role in designing a pattern recognition system, the research done in this area using fuzzy set theory is not significant. Bezdek and Castalez [22] have shown an application of the fuzzy c-means clustering algorithm to select an optimum feature subset from the set of available features so that there is no appreciable loss of classifier performance with the reduced set of features. Pal and Chakraborty have explained in [156] 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 by Pal [152] to evaluate the importance of any subset of features using an average quantitative index of goodness. In Appendix B, we describe this method as it is used for comparison with our supervised feature selection methods, stated in the thesis.

There have been several attempts showing the application of fuzzy set theoretic approaches to real life recognition (classification) problems. Some of these can be found in literature [166, 22, 46] for recognizing speech patterns which are biological in origin and the patterns manifest a considerable amount of fuzziness (vagueness). A fuzzy version of the well known *k-nearest neighbor (k-NN)* classifier has been provided by Keller *et al.* [104]. An extensive discussion on the application of fuzzy k-NN classifier for diagnosing gastric cancer has been provided in [218]. Defining degrees of membership to correspond to the severity of *metastases*, the authors have designed a fuzzy pattern recognition system with multiple class membership values.

Pathak and Pal [170, 171] have 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 [22].

A multivalued recognition system has been developed by Pal and Mandal [159, 132], based on the concept of 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 [159] and geometrical structures [132] of training samples. The performance of the algorithms has been demonstrated on speech recognition problem and analysis of satellite imagery for detecting man-made objects [134]. 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 [133]. 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.* [131], in order to avoid committing oneself to a specific determination of boundary.

Another idea in pattern recognition is the partitioning of the initial feature space into regions and the application of different classification rules to them [92]. A partition may be based on the geometric properties of the classes detected by a preliminary clustering method. In the fuzzy classification rule described in [92], 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.* has extended this work in [93] 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.

Besides the abovementioned supervised classification methods, fuzzy set theory has been extensively used in clustering problems where the task is to provide cluster 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 [194]. A new direction in this line has been initiated by Bezdek and Dunn in their work on fuzzy ISODATA and the fuzzy c-means algorithms [20]. Another important branch called fuzzy image processing has also grown 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 this area are not mentioned here to a greater details, as it is beyond the scope of this thesis. Readers may consult [160] for further details.

1.4 Overview of Connectionist Models for Pattern Recognition

During the mid 1950's and early 1960's a class of machines called perceptrons, proposed by Rosenblatt [188], 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 [139] 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 [192] 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 has rejuvenated a stagnating area of research and led to a veritable surge of interest in neural network models.

Artificial neural networks (ANNs) [193, 127, 112, 82, 97] 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 follow-

ing 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.

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.

1.4.1 Relevance of ANN

As mentioned before for any pattern recognition system, it is always desirable for a model to possess characteristics such as adaptivity, speed, robustness, ruggedness

and optimality. Since 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.

1.4.2 Various approaches

Some of the commonly known neural networks generally used for pattern recognition and optimization tasks include the single-layer perceptron [188, 64], Boltzmann machine [3, 1], multi-layer perceptron using back propagation of errors [192], Hopfield nets [86, 85], Hamming nets [127], Grossberg/Carpenter nets (ART) [29], Kohonen's self-organizing feature maps [112], Counterpropagation networks [79, 80] and Neocognitron [60, 61]. The Hopfield and Hamming nets are primarily used with fixed weights for optimization tasks. The single and multi-layer perceptrons undergo supervised learning, basically for classifier design. On the other hand, the Gross-

berg/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 [95] for pattern recognition. In Appendix C, we describe, in detail, the multilayer perceptron (MLP), as it is used in some of our investigations presented in the thesis.

Applications of the various neural network models have been made in diverse spheres of pattern recognition. A good survey can be found in [28, 126, 221]. Some of the notable areas encompass

- feature selection/extraction [98, 135, 182, 217],
- invariant recognition schemes [180, 58],
- partially occluded object recognition [14, 13],
- applications to text recognition [212, 24],
- character recognition [55],
- speech recognition [124, 26],
- recognition of hand gesture [125],
- sonar target recognition [186, 89],
- knowledge representation [169, 71], and
- image processing and object recognition [27, 231, 65, 67].

Since, the present thesis deals with feature evaluation and classification in connectionist framework, we shall describe some of the relevant attempts concerning these areas.

Feature selection/extraction

Some of the recent attempts made for feature evaluation (selection/extraction) in connectionist framework are mainly based on multilayer feedforward networks

[191, 116, 19, 129, 135, 182, 199, 200] and self-organizing networks [135, 119, 117]. The methods based on multilayer feedforward networks include, among others, determination of saliency (usefulness) of input features [191, 182], development of Sammon's nonlinear discriminant analysis (NDA) network, and linear discriminant analysis (LDA) network [135]. On the other hand, those based on self-organizing networks include development of nonlinear projection (NP-SOM) based Kohonen's self-organizing feature map [135], distortion tolerant Gabor transformations followed by minimum distortion clustering by multilayer self-organizing maps [119], and a non-linear projection method based on Kohonen's topology preserving maps [117].

The method of Ruck et al. [191] uses a multilayer perceptron for ranking the input features. The rate of change of output of the network with respect to an input feature is used as an index, called *saliency*, for measuring the degree of importance of the feature. Individual features are ranked based on their *saliency*-values. Appendix D describes the method in detail, as we have used it for comparing one of our supervised feature selection methods.

Demartines et al. [47] have described a new strategy called "curvilinear component analysis (CCA)" for dimensionality reduction and representation of multidimensional data sets. The principle of CCA is implemented in a self-organized neural network performing two tasks: *vector quantization* of the submanifold in the data set (input space) and *non-linear projection* of these quantized vectors toward an output space, providing a revealing unfolding of the submanifold. After learning, the network has the ability to continuously map any new point from one space into another.

The decision boundary feature extraction method, proposed by Lee et al. [121, 122], is based on the fact that all the necessary features for classification can be extracted from the decision boundary between a pair of pattern classes. The term "decision boundary" has been defined by them. The algorithm can take advantage of characteristics of neural networks which can solve complex problems with arbitrary decision boundaries without assuming underlying probability distribution functions of the data.

Setino et al. [201] have demonstrated how a three layer feedforward neural network can be used to select the input attributes that are most important for discriminating classes in a given set of input patterns. The algorithm is based on network pruning.

By adding a penalty term to the error function of the network, redundant network connections can be distinguished from those relevant ones by their small weights when the network training process has been completed. A simple criterion to remove an attribute, based on the accuracy rate of the network, is developed. The network is trained after removal of an attribute, and the selection process is repeated until no attribute meets the criterion for removal.

Chatterjee et al. [33] have described various self-organized learning algorithms and associated neural networks to extract features that are effective for preserving *class separability*. An adaptive algorithm for the computation of $Q^{-1/2}$ (where Q is the correlation or covariance matrix of a random vector sequence) is described. Convergence of this algorithm with probability one is established by using stochastic approximation theory. A single layer linear network, called $Q^{-1/2}$ network, for this algorithm is described. Networks with different architectures are designed for extracting features for different cases.

Battiti [15] has investigated the application of *mutual information* criterion to evaluate a set of candidate features and to select an informative subset to be used as input data for a neural network classifier. Mutual information, being a measure of arbitrary dependencies between random variables, is used for assessing the "information content" of features in complex classification tasks. The fact that the mutual information is independent of the coordinates chosen permits a robust estimation. In addition, the use of the mutual information for tasks characterized by high input dimensionality requires suitable approximations because of the prohibited demands on computation and samples. An algorithm is described based on a "greedy" selection of the features and it takes both the mutual information with respect to the output class and the already-selected features into account.

Principal component analysis network of Rubner and Tavan [190] performs the task of feature extraction through the well known principal component analysis. The network consists of two layers, *viz.*, input and output. The weights of the network are adjusted through local learning rules. The description of the network along with its learning algorithm is provided in Appendix E, as we have compared our feature extraction method with this network.

Hornik et al. [87] have demonstrated the asymptotic behavior of a general class

of on-line principal component analysis (PCA) learning networks which are based strictly on local learning rules [190]. It is established that the behavior of the algorithms is intimately related to an ordinary differential equation which is obtained by suitable averaging over the training patterns. They have studied the equilibria of these equations and their local stability properties. It has been shown that local PCA algorithms should always incorporate hierarchical rather than more competitive, symmetric decorrelation, for providing their superior performance.

Certain issues related to classification

Several investigations on the comparison of the abilities of neural nets with conventional pattern recognition techniques have been provided by Barnard and Casasent [9], Bounds *et al.* [25], and Chen [34]. Efforts at improving the speed of convergence and/or efficiency of the back propagation algorithm was reported Tollenaere [213]. Some other recent approaches in this direction include the works of Bello [18] (using nonlinear least squares and quasi-Newton optimization techniques), and Wessels and Barnard [220] (by properly initializing connection weights).

Automated generation of appropriate/optimal neural network topology has been investigated by Fahlman *et al.* [53] (employing the cascade-correlation learning algorithm), Sietsma and Dow [204] (by pruning and/or adding units), Bauer and Pawelzik [16] (utilizing a topographic product to indicate perfect neighborhood preservation of the optimal self-organizing feature maps) and Smyth [208]. 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.

Salzberg's Nested Generalized Exemplar (NGE) theory [195] creates hyperrectangles (exemplars) to handle the training examples. If the class of the nearest exemplar matches, the nearest exemplar will be expanded to include the training example, otherwise the second nearest exemplar will be tried (this is the second match heuristic).

If the classes of the first and second nearest exemplars are both different from the class of training example, this training example will be stored as a new exemplar.

Abou-Nasr *et al.* [2] have designed a prototype-based neural network classifier (NNC). It automatically chooses the initial firing threshold conditions of its prototype nodes. When a new prototype node is created, it will try to set it with the maximum initial firing threshold value such that it does not overlap prototypes of other classes and does not violate any given constraints. The firing threshold values of the neurons are adjusted during learning, and the neurons belonging to different classes are not allowed to overlap.

Connectionist models can estimate Bayesian a posterior probabilities. It can be shown that, for one-from-many classification problems, attainment of the minimum value of a variety of cost functions with respect to the weights yields an estimate of the posterior (class conditional) probabilities required for the implementation of a Bayesian classifier [185]. Thus, given sufficient data, computational resources and time, it is possible to estimate the Bayes optimal classifier to any desired degree of accuracy, directly and with no prior assumptions on the probabilistic structure of the data.

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 [88, 91, 191, 197, 8]. Some attempts on the hardware implementation of the various neural network models can also be found in [137, 183, 224]. Parallel versions of the different algorithms involved have been implemented in [150, 140].

1.4.3 Knowledge-based networks

Generally ANNs consider a fixed topology of neurons connected by links in a predefined manner. These connection weights are usually initialized by small random values. Knowledge-based networks [57, 215] constitute a special class of ANN that consider crude domain knowledge to generate the initial network architecture which is later refined in the presence of training data. This process helps in reducing the searching space and time while the network traces the optimal solution. Growing/pruning of link/node is also made in order to generate the optimal network

architecture.

Such a model has the capability of outperforming a standard network. 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.

A brief review on the knowledge-based networks for classification and rule generation is now provided in the following paragraphs. The model by Gallant [63], 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*. Dependency information regarding the variables, in the form of an adjacency matrix, is provided by the expert. 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 [225] have 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.

Fu [57] has 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 knowledge of the trained network is then used to extract the revised rules for the problem domain. Its application to a control problem is also demonstrated.

Towell and Shavlik [215] have mapped problem-specific "domain theories", represented in propositional logic, into layered neural networks and then refined this reformulated knowledge using backpropagation. The hybrid learning system is found to generalize better and is evaluated on problems from molecular biology. Disjunctive rules are rewritten as multiple conjunctive rules while building the network structure. Nodes and links are incorporated, upon instructions from the user, to augment the knowledge-based module.

1.5 Overview of Neuro-fuzzy Approach

So far we have described various fuzzy set theoretic and connectionist approaches for pattern recognition. For the last few years, researchers all over the world [22, 168, 96, 62, 21, 115, 31, 30, 158, 161, 230] have been trying to combine the merits of these approaches under the heading *neuro-fuzzy computing* for building more intelligent decision making systems in soft computing framework. 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 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 [158]:

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 [105];

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 [90, 23, 108, 210, 211, 94, 206, 207];

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 [118, 106, 174, 176];

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 [123, 223]; 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 [66].

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 [105]. 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). The investigations of Mitra and Pal [144] also fall under this category, and are described below.

Pal and Mitra have designed several neuro-fuzzy models for classification, inferencing and rule generation. These include fuzzy versions of the multilayer perceptron

(fuzzy MLP) [167] and self-organizing Kohonen's model [145]. The methods describe a way of integrating the uncertainty handling capability of fuzzy set theory with the ability of neural networks in generating highly nonlinear decision boundaries to design intelligent systems for pattern recognition. The systems accept input features in linguistic form (low, medium and high). The output vector of the fuzzy version of the MLP model, corresponding to a sample point coming from overlapping regions, is given unconstrained memberships to more than one class. This helps in reducing oscillations of the decision boundaries caused by patterns from overlapping regions. In the fuzzy version of the Kohonen's model, the input vector consists of membership values for *linguistic* properties (low, medium and high) along with some *contextual class membership* information which is used during self organization to permit efficient modeling of *fuzzy* (ambiguous) patterns. In Appendix F we describe the fuzzy MLP model [167] which is used in our investigation both for developing new systems and for comparison.

Mitra and Pal have used these models for inferencing and generating rules in *If-Then* form [146]. Based on the output class membership values 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 it is required. Justification for an inferred decision may be produced in rule form, when so desired by the user. Appendix F describes the method, as we have used this for comparison.

Fuzzy min-max neural network of Simpson [206] is another example of category 1. The network is used for supervised classification where pattern classes are modeled as fuzzy sets. Each fuzzy set is an aggregate (union) of fuzzy set hyperboxes. A fuzzy set hyperbox is an n-dimensional box (for an n-dimensional feature space) defined by min and max points with the corresponding membership function. The min-max points are determined using the fuzzy min-max learning algorithm, an expansion-contraction process that can learn nonlinear class boundaries in a single pass through the data. Appendix G describes the network in detail, as we have used it for comparison.

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 [90] are the first to modify Kohonen's network for generating the fuzzy self-organizing feature map. Fuzziness has also been 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 are reported by Bezdek *et al.* [23]. A relationship between the fuzzy version of Kohonen's algorithm and the fuzzy c-means algorithm [20] is also established. A neural network architecture that can be used for fuzzy clustering and classification has been suggested in [149]. The system uses a control structure similar to that found in adaptive resonance theory of Carpenter and Grossberg [72]; and employs 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) is introduced in [206]. 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.

Pedrycz [177] has used radial basis function (RBF) neural networks aimed at an approximation of nonlinear mapping from R^n to R. The study is devoted to the design of these networks, especially their layer composed of RBF's, using the techniques of fuzzy clustering, and falls under category 2. The main objective of this idea is to develop clusters (receptive fields) preserving homogeneity of the clustered patterns with regard to their similarity in the input space as well as their respective values assumed in the output space.

Fusion made in category 3 replaces the integration/transformation operation at each node with the fuzzy aggregation operation (*i.e.*, fuzzy union, intersection, etc.). Pedrycz [174] has used logical operators *max* and *min*, with the *crisp* implication operator, for designing two-layered neural nets capable of solving two-class problems. This has been extended in 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, is introduced in [176, 175] 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.* [219] have 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 [118]. They used fuzzy aggregation connectives with compensatory behavior (lying in the range between the two extremes, *viz.*, *min* and *max*) as the activation functions of the neurons. A modified version of the backpropagation algorithm 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 are implemented by the layered networks. The model of Zimmermann and Zysno [233] 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. Hirota and Pedrycz [84] have 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.

A fuzzy layered neural network for classification and rule generation using *logical neurons* [143, 144] has been developed by Mitra and Pal. The input and output representations are the same as that of the fuzzy MLP [167]. Instead of the standard weighted sum and sigmoid functions, the system uses the logical *And* and *Or* operators, and the methodology falls under category 3. 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.*, *min-max* and *product-probabilistic sum*, are utilized 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. 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.

Chandrasekaran and Liu [32] have presented a novel topology constraint free neural network architecture using a generalized fuzzy gated neuron model for pattern recognition task. This attempt also falls under category 3. The main feature is that the network does not require weight adaptation at its input and the weights are initialized directly from the training set. This facilitates quick network set up times. The performance of the network is found to be functionally equivalent to spatio-temporal feature maps under a mild technical condition

The idea of making the individual neuron fuzzy falls under category 4, and is first promulgated in 1975 by Lee and Lee [123]. 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 jth 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 investigation of Ghosh *et al.* [66] falls under category 5, where an attempt has been made to incorporate various fuzziness measures in a multilayer 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 [156]) 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 and noise-insensitive segmentation algorithm. Similar investigations using fuzziness measures as error criteria are available in [155, 198].

Neuro-fuzzy hybridization is done broadly in two ways [161]: a neural network equipped with the capability of handling fuzzy information (termed fuzzy-neural network FNN), and a fuzzy system augmented by neural networks to enhance some of its characteristics like flexibility, speed and adaptibility (termed neural-fuzzy system NFS).

In an FNN either the input signals and/or connection weights and/or the outputs are fuzzy subsets or membership values to some fuzzy sets [123, 167]. Usually linguistic values like *low*, *medium* and *high*, or fuzzy numbers/intervals are used to model them. Neural networks with fuzzy neurons are also termed FNN as they are capable of processing fuzzy information.

A neural-fuzzy system (NFS), on the other hand, is designed to realize the process of fuzzy reasoning, where the connection weights of the network correspond to the parameters of fuzzy reasoning [107, 108]. Using the backpropagation type learning algorithms, the NFS can identify fuzzy rules and learn membership functions of the fuzzy reasoning. Usually for an NFS, it is easy to establish a one-to-one correspondence between the network and the fuzzy system. In other words, the NFS architecture has distinct nodes for antecedent clauses, conjunction operators and consequent clauses. There can be, of course, another black-box type NFS where a multilayer network is used to learn the input-output relation represented by a fuzzy system. For such a system the network structure has no such relation to the architecture of the fuzzy reasoning system.

1.5.2 Neuro-fuzzy knowledge-based systems

Several attempts on neuro-fuzzy approaches to the design of knowledge-based systems have also been reported. Masuoka *et al.* [136] have 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.

Fuzzy signed digraph with feedback, termed as fuzzy cognitive map, is used by Kosko [114] 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 [130] have 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 [120]. 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 [179] have 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 [83] has incorporated the use of fuzzy clustering for developing the geometric constructs leading to the design of knowledge-based networks. Its applications to classification problems are also described.

1.6 Use of Case-based Reasoning

Case-based reasoning [7, 181, 113] may be defined as a model of reasoning that incorporates problem solving, understanding and learning, and integrates all of them with memory processes. These tasks are performed using some typical situations, called *cases*, already experienced by the system. There are widespread applications of the concept of case-based reasoning in various decision making processes where the knowledge available is usually incomplete and/or evidence is sparse. Case-based reasoning integrates reasoning and learning and it is not enough for a case-based reasoner to stop reasoning after it derives a solution. Rather, it must continue by collecting feedback about its solution and evaluating that feedback. Without evaluation processes based on feedback, learning could not take place reliably, and case-based reasoning itself would too unreliable to depend on.

The quality of a case-based reasoner's reasoning depends on the experiences it has had, its ability to understand new situations in terms of those of old experiences, its adeptness at adaptation, and its adeptness at evaluation and repair. The major processes employed by a case-based reasoner are *case storage and retrieval*, *adaptation*, and *criticism*.

Case-based reasoning is applicable to a wide range of real-world situations, ranging from knowledge-rich situations in which construction of solutions is complex to knowledge-poor situations in which cases provide the only available knowledge. Since the problems in pattern recognition may involve both knowledge-rich and knowledge-poor situations, case-based reasoning has a great applicability in this area. Case-based reasoning has many advantages, allowing a reasoner to propose solutions to problems quickly, to reason in domains that are not well understood, to evaluate solutions when algorithmic methods are not available, to avoid previous problems, and to focus on important parts of a situation. Its major disadvantages are all linked to using cases poorly to reason, for example, relying too heavily on their proposals without evaluating them. Recently, attempts are being made for solving cased-based reasoning problems in the framework of fuzzy sets, neural networks and neuro-fuzzy hybridization.

Application of case-based reasoning to pattern recognition problems has not yet been significant. Here we mention briefly some of the related attempts. PROTOS [113,

181] implements both case-based classification and case-based knowledge acquisition in the domain of audiological (hearing) disorders. Given a description of a disorder, it classifies the type. When it misclassifies a situation, its expert consultant steps in and informs PROTOS of its mistake and what knowledge it needs to classify the situation correctly. Narazaki and Watanabe [148] have proposed a case-based approach for modeling nonlinear systems. Given an input to the system, the method predicts the output by interpolating the outputs of the "similar" cases whose inputs are close to the given input. Ding et al. [50] have designed a neural network structure for describing relations of fuzzy rules and modified the weights of the neural network to realize learning from cases (examples). Here, the concept of *approximate case-based reasoning* and its neural network implementation have been developed. Recently, Liu et al. [128] have designed a fuzzy neural network for diagnosis of symptoms in electronic system using the concept of case-based reasoning.

1.7 Scope of the Thesis

The objective of this thesis is to present some results of investigations, both theoretical and experimental, that demonstrate the effectiveness of the fuzzy set theoretic and connectionist approaches, particularly in integrated manner, by developing some new methodologies for feature evaluation, pattern classification and rule generation. Various fuzzy neural networks are developed along with learning algorithms. Feature selection/extraction is done under both supervised and unsupervised modes of learning. A theoretical analysis is also provided for the supervised method. For the purpose of classification, neuro-fuzzy case-based and knowledge-based systems have been designed. The knowledge-based network has been used to extract linguistic rules in *If-Then* form.

The effectiveness of these models is demonstrated on some artificially generated pattern sets, as well as some real life problems e.g., speech recognition, classification of Iris flowers and mango leaves, and medical diagnosis, with the number of dimensions ranging from three to eighteen. The superiority of these models over other related ones is also established. The results of the investigations are summarized below under different chapter headings.

1.7.1 Feature selection using neural networks and fuzzy set theory [43]

Chapter 2 describes two techniques for feature selection. One of them is based on fuzzy set theory, while the other uses a multilayer perceptron (MLP). The former one is a modification of the method of Pal [152], and involves minimization of a fuzzy feature evaluation index. The index is defined in terms of entropy of fuzzy sets by taking care of interset and intraset distances. The connectionist approach, on the other hand, involves a feature quality index which is defined in terms of outputs of an MLP in the presence and absence of features. The basic idea behind this scheme is as follows. After the MLP has learnt the data set, the absence of a more (less) important feature is likely to influence the output much (less). The importance of features both individually and in a group is determined. An extensive comparison of the performance of these algorithms is made with those of a statistical [49], fuzzy set theoretic [152] and a connectionist [191] method for a three dimensional speech, four dimensional Iris, nine dimensional medical and an eighteen dimensional mango-leaf data. Note that the methodologies discussed above consider the theories of fuzzy sets and artificial neural networks individually. The problem of integrating them for performing the task of feature evaluation in different modes of learning is addressed in the next three chapters.

1.7.2 Neuro-fuzzy supervised feature selection [154, 153, 10, 38]

Chapter 3 provides a neuro-fuzzy system for feature selection along with its theoretical and experimental performance. First of all, a fuzzy set theoretic feature evaluation index is defined in terms of individual class membership. Its relation with Mahalanobis distance and divergence measure is demonstrated. Then we have provided a new connectionist model to perform the task of minimizing the aforesaid fuzzy evaluation index which incorporates weighted distance for computing class membership values. This minimization process results in a set of weighting coefficients representing the importance of the individual features. These weighting coefficients lead to a transformation of the feature space for better modeling the class structures. Finally,

the performance of the system is theoretically analyzed. This includes derivation of upper and lower bounds of the evaluation index, and determining its relation with interclass distance and weighting coefficient. In another part of the investigation, the aforesaid fuzzy evaluation index is used alone to find the best subset of features. This is done by computing the evaluation index (with weighting coefficients being unity) on different subsets of features and then ordering them accordingly.

The effectiveness of these algorithms is adequately demonstrated on the aforesaid four real life data sets. The experimental results are validated independently with both scatter plots and k-NN classifier for different values of k.

1.7.3 Neuro-fuzzy unsupervised feature selection [11, 12, 165]

The method described in Chapter 3 for feature selection is supervised. In Chapter 4, we have described an unsupervised neuro-fuzzy approach for feature selection. A fuzzy feature evaluation index for a set of features is defined in terms of membership values denoting the degree of similarity between two patterns with respect to their belongingness to a cluster. The similarity between two patterns is measured with an weighted distance between them. The weighting coefficients are used to denote the degree of importance of the individual features in characterizing/discriminating different clusters and to provide flexibility in modeling various clusters. The evaluation index is such that, for a set of features, the lower is its value, the higher is the importance of that set in characterizing/discriminating various clusters. A layered network, different from that used in Chapter 3, is formulated for performing the tasks of computation and minimization of the evaluation index through unsupervised learning process; thereby determining the optimum weighting coefficients reflecting the individual feature importance. Like Chapter 3, the aforesaid fuzzy evaluation index is also used alone, in a part of the investigation, to find the best subset of features. The effectiveness of the algorithms has been extensively demonstrated on the same data sets, used in Chapters 2 and 3. This also includes a validation of the results using k-NN classifier.

1.7.4 Neuro-fuzzy unsupervised feature extraction [39, 40, 165]

The feature selection method described in Chapter 4 is extended for the task of feature extraction in Chapter 5. For this purpose, a set of different linear transformation functions is applied on the original feature space and the computation of the aforesaid evaluation index has been made on the transformed spaces. The similarity between two patterns in the transformed space is computed, as in the case of feature selection method of Chapter 4, using a set of weighting coefficients. A layered network is designed where the transformation functions are embedded. An optimum transformed space along with the degrees of individual importance of the transformed (extracted) features is obtained through connectionist minimization. Like feature selection methods, all these operations are performed in a single network, but with different architecture. Here the network is such that the number of nodes in its second hidden layer determines the desired number of extracted features. The method is compared with a principal component analysis network [190] using k-NN classifier, scatter plots and fuzzy c-means clustering algorithm, when the same data sets used in Chapters 2–4 are considered as input. The next two chapters address the problems of classification and rule generation in neuro-fuzzy paradigm.

1.7.5 Fuzzy case-based classification in connectionist framework [44, 45]

Chapter 6 deals with the development of a case-based pattern classification system using fuzzy set theory in a connectionist framework. *Cases* are typically labeled patterns which represent different regions of the classes. Incorporation of fuzzy set theory helps in selecting the *cases* from ambiguous/overlapping regions. The concept of fuzzy similarity, in terms of distance measure, is used to determine the degree of resemblance between two patterns. These tasks (e.g., computation of the degree of similarity, selection of *cases* etc.) are performed with a layered network whose architecture is determined adaptively through *growing* and *pruning* of nodes under supervised mode of training. Growing and pruning correspond to addition and deletion of cases respectively. The superiority in performance of the system over

those of Bayes maximum likelihood and k-NN classifiers is established with different bandwidths of the membership function for an artificially generated data, as well as the same speech (vowel) and medical data.

1.7.6 Knowledge-based fuzzy MLP for classification and rule generation [41, 42, 142]

Chapter 7 provides the design of a knowledge-based system in neuro-fuzzy framework for both classification and rule generation. The input is modeled in terms of linguistic features (*low*, *medium* and *high*) while the output consists of class membership values. The *a priori* class information and the distribution of pattern points in the feature space are taken into account while encoding the crude domain knowledge from the data set among the connection weights. An accurate estimation of the links connecting the output and hidden layers (in terms of the preceding layer link weights and node activations) is also provided.

The network is next trained to refine its architecture, and node growing or link pruning is incorporated (if necessary) to generate the optimal topology. An exhaustive set of linguistic rules is generated in *If-Then* form (i) through forward pass using both linguistic (natural) and numerical inputs for the antecedent clauses and the computed network output for the consequent part, and (ii) by backtracking along the maximum weighted paths using the trained connection weights along with the input and output activations of the network.

Note that the model is capable of handling input in numerical, linguistic and set forms and can tackle uncertainty due to overlapping classes. Negative rules, indicative of cases where a pattern does not belong to a class, can also be generated. This is specially suitable in the ambiguous cases where positive rules (dealing with the belongingness of a pattern to a particular class) cannot be obtained. As in the previous chapter, the same artificially generated pattern set, as well as real life vowel and medical data have been considered for demonstrating the effectiveness of the system. The results are also compared with those of conventional and fuzzy versions [167] of MLP, and fuzzy min-max neural network [206].

1.7.7 Conclusions and scope for further research

The concluding remarks along with the scope for further research are presented in Chapter 8.

Chapter 2

Feature Selection Using Neural Networks and Fuzzy Set Theory

2.1 Introduction

As mentioned in Chapter 1, feature selection is a process of selecting some important features from the measurement space to constitute what is called the feature space. The purpose of this task is to retain the optimum salient characteristics necessary for the recognition process; thereby reducing the cost and complexity of the problem. In this chapter we describe two methods [43] for feature selection. One of them is based on a modification of the fuzzy set theoretic method (Appendix B) of Pal [152], while the other approach uses a multilayer perceptron (MLP). The MLP-based method involves, first of all, learning a set of labeled data using an MLP with an adequate architecture. Then, for each training data point we set a feature value to zero (one may call such a vector as corrupted data point) and use it as an input for classification. The deviation of the output vector thus produced from the output generated by the corresponding uncorrupted data point is noted. A feature is considered more important if the average deviation over the entire data set for that feature is more. The basic idea behind this scheme is as follows. After the MLP has learnt the data set, the absence of an important feature is likely to influence the output significantly. On the other hand, for a less important feature, the output is not expected to change much with variation of the value of that feature. The performance of these methods has been compared with some existing techniques [49, 152, 191] (Appendices A, B and D), using scatter plots on both artificial and real life data sets. The real life data includes 4-dimensional 3-class Iris [54], 3-dimensional 6-class vowel [166], 9-dimensional 4-class medical [77] and 18-dimensional 3-class mango-leaf [152] data.

Section 2.2 describes the said fuzzy set theoretic and MLP-based methods of feature selection. Experimental results are analyzed in Section 2.3. The chapter is concluded with Section 2.4.

2.2 Feature Selection Using Fuzziness and Neural Networks

In this section we describe two schemes for feature selection. The first one is based on fuzzy set theoretic concepts. It is an extension of an earlier work of Pal [152].

The other scheme is based on multilayer perceptron.

2.2.1 Feature evaluation using fuzziness

Let $X = \{x_1, x_2, ..., x_s\}$ be a universe of discourse and a fuzzy set $\mathcal{A} = \{\mu_{\mathcal{A}}(x_p)/x_p | x_p \in X; p = 1, 2, ..., s; \mu_{\mathcal{A}} \in [0, 1]\}$ be defined on X where $\mu_{\mathcal{A}}(x_p)$ denotes the membership of x_p to $\mathcal{A}$. For measuring the fuzziness of $\mathcal{A}$, which is rewritten [151] here (for the convenience of readers) as

$$H(\mathcal{A}) = \frac{1}{s \ln 2} \sum_{p=1}^{s} \{-\mu_{\mathcal{A}}(x_p) \ln(\mu_{\mathcal{A}}(x_p)) - (1 - \mu_{\mathcal{A}}(x_p)) \ln(1 - \mu_{\mathcal{A}}(x_p))\}. \quad (2.1)$$

$H(\mathcal{A})$ is called entropy [151] of the fuzzy set $\mathcal{A}$. $H(\mathcal{A})$ attains the maximum value when $\mathcal{A}$ is most fuzzy, *i.e.*, when $\mu_{\mathcal{A}}(x_p) = 0.5$, $\forall p$, and it attains the minimum value when $\mu_{\mathcal{A}}(x_p) = 0$ *or* 1, $\forall\ p$.

For computing μ-values we use S-type [166] membership function, which is

$$\begin{aligned} \mu_{\mathcal{A}}(x_p; a, b, c) &= 0, & x_p \leq a \\ &= 2[\tfrac{x_p - a}{c - a}]^2, & a \leq x_p \leq b \\ &= 1 - 2[\tfrac{x_p - c}{c - a}]^2, & b \leq x_p \leq c \\ &= 1, & x_p \geq c \end{aligned} \quad (2.2)$$

in the interval $[a, c]$ with $b = (a + c)/2$. The parameter b is known as the crossover point for which $\mu_{\mathcal{A}}(b; a, b, c) = 0.5$.

In order to compute H of class C_k along ith feature, the parameters of the S-function can be computed following Pal [152] as

$$b = (x_{ik})_{av}, \quad (2.3)$$

$$c = b + \max\{|(x_{ik})_{av} - (x_{ik})_{max}|, |(x_{ik})_{av} - (x_{ik})_{min}|\}\ , \quad (2.4)$$

and

$$a = 2b - c. \quad (2.5)$$

Here av, max and min are used to denote the average, maximum and the minimum value of x_{ik}, respectively. If each x_{ik} is equal to b, H will be maximum and equal to 1. H tends to zero as x_{ik} moves away from b towards either c or a. The higher

the value of H, the greater would be the number of samples having $\mu(x) \approx 0.5$ and hence greater would be the tendency of the samples to cluster around its mean value, resulting in less (internal) scatter within the class. If we pool together classes C_k and $C_{k'}$ and compute the average, maximum and minimum values of the ith feature over all $(s_k + s_{k'})$ samples where $s_r(r = k, k')$ is the number of samples of class C_r, H for the pooled samples would decrease as the goodness of feature increases. This is because, for a good feature, the samples from both classes should be away from the overall mean, *i.e.*, most of the points will have $\mu(x) \approx$ 0 *or* 1.

Now the overall feature evaluation index for ith feature ($OFEI_i$) can be defined as

$$OFEI_i = \frac{\sum_{k \neq k'}^{M} H_{ikk'}}{\sum_{k=1}^{M} H_{ik}}. \tag{2.6}$$

H_{ik} is obtained by Eqn. 2.1 for class C_k along the feature x_i while $H_{ikk'}$ is computed (using the same equation) by pooling the classes C_k and $C_{k'}$ together. The average, maximum and minimum values of the ith feature over all $(s_k + s_{k'})$ samples, where $s_r(r = k, k')$ is the number of samples of class C_r, are computed from this pooled data.

An ith feature will be good if it can discriminate every pair of the M classes. Therefore, the goodness of a feature x_i increases as $H_{ikk'}$ $(k, k' = 1, 2, \ldots, M$ and $k \neq k')$ decreases and H_{ik} $(k = 1, 2, \ldots, M)$ increases, *i.e.*, $\sum_{k \neq k'} H_{ikk'}$ decreases and $\sum_{k=1}^{M} H_{ik}$ increases; thereby $OFEI$ decreases.

The lower the value of $OFEI$, the better will be the performance of the feature with respect to discriminating all the classes. In Eqn. (2.6), it may happen that $H_{ikk'} < H_{jkk'}$ but $H_{ill'} > H_{jll'}$, *i.e.*, ith feature is more important to discriminate classes C_k and $C_{k'}$ than jth feature but the converse is true for classes C_l and $C_{l'}$. Since Eqn. (2.6) considers all possible pairs of classes, $OFEI_i$ will reflect the overall (average) discriminating power of the ith feature.

The index $OFEI$ has an advantage over $(FEI)^{av}$ (Eqn. (B.7)) of Pal [152]. Unlike $(FEI)^{av}$, $OFEI_i$ does not depend on the size of a class. In $(FEI)^{av}$ (Eqn. (B.7)), the weights are nothing but the *a priori* probabilities of different classes. Hence, $(FEI)^{av}$ depends on the number of points in a class, which may not be desirable.

Moreover, in Eqn. (B.7), even when $s_k + s_{k'} = \psi$ (a constant) for two different pairs of classes, $W_k W_{k'}$ could be different for the two pairs. $W_k W_{k'}$ attains the maximum value when $s_k = s_{k'} = \frac{\psi}{2}$. Hence, $(FEI)^{av}$ is biased towards equiprobable classes, which is not desirable.

2.2.2 Feature selection using an MLP

The weights of the links of an MLP (Appendix C), after it successfully learns a data set, are so adjusted that the value of a redundant (less important) feature does not influence the output vector much. The lower the importance of a feature in discriminating between classes, the lower would be the influence of its value on the output of the network. The MLP based feature selection algorithm that will be described here is based on this concept.

For every ith feature we compute a feature quality index, FQI_i and then rank the features accordingly. To compute FQI_i we proceed as follows: For each training data point $\mathbf{x}_p$, $p = 1, 2, \ldots, s$, we set x_{pi} to zero. Let this modified data point be denoted by $\mathbf{x}_p^{(i)}$; *i.e.*, $x_{pj}^{(i)} = x_{pj}$, $\forall\ j \neq i$ and $x_{pi}^{(i)} = 0$. Setting ith component to zero is equivalent to delinking the ith input node and hence delinking all connections associated directly from the ith input node. Thus, the impact of the ith feature will not reach any node of the network. Let the output vectors obtained by $\mathbf{x}_p$ and $\mathbf{x}_p^{(i)}$ be $\mathbf{o}_p$ and $\mathbf{o}_p^{(i)}$ respectively. Note that $\mathbf{o}_p$ is *not* the target output corresponding to $\mathbf{x}_p$, but the *actual* output that is obtained for $\mathbf{x}_p$ from the trained net. For a less important feature, the output vectors $\mathbf{o}_p$ and $\mathbf{o}_p^{(i)}$ are not expected to differ much. Any function of $\mathbf{o}_p$ and $\mathbf{o}_p^{(i)}$ that can measure this variation between the two can be used as an index for feature ranking. A very simple choice would be to define

$$FQI_i = \frac{1}{s} \sum_{p=1}^{s} \|\mathbf{o}_p - \mathbf{o}_p^{(i)}\|^2 \ . \tag{2.7}$$

After computing FQI_is for all the n features, they can be ranked according to their importance as $x_1, x_2, \ldots, x_n$ when $FQI_1 \geq FQI_2 \geq \ldots \geq FQI_n$.

Instead of ordering individual features, if the problem is to select n' ($n' \leq n$) best features (feature selection), best from the point of view of discrimination between classes, the feature set $\{x_1, x_2, \ldots, x_{n'}\}$ may not be the optimal set. But $\{x_1, x_2, \ldots, x_{n'}\}$ will definitely give a very good subset of features. To get the best n' features we proceed

as follows: There are $\binom{n}{n'}$ possible subsets of features. Let the lth subset be denoted by $\mathcal{S}_l$. Now we define $FQI_l^{(n')}$ as

$$FQI_l^{(n')} = \frac{1}{s} \sum_{p=1}^{s} ||\mathbf{o}_p - \mathbf{o}_p^l||^2 , \tag{2.8}$$

where $\mathbf{o}_p^l$ is the output from the net with $\mathbf{x}_p^l$ as input. Note that $\mathbf{x}_p^l$ is derived from $\mathbf{x}_p$ as follows:

$$\begin{aligned} {x_{pi}}^l &= 0, \quad x_i \in \mathcal{S}_l \\ &= x_{pi}, \quad otherwise. \end{aligned} \tag{2.9}$$

We choose $\mathcal{S}_j$ as the optimal subset of n' features, when $FQI_j^{(n')} \geq FQI_l^{(n')}$, $\forall l; l \neq j$.

Instead of Eqns. (2.7)-(2.8), one can also compute *feature devaluation indices* ($FDIs$) as

$$FDI_i = \frac{1}{s} \sum_{p=1}^{s} \frac{{\mathbf{o}'}_p \mathbf{o}_p^{(i)}}{||\mathbf{o}_p||\ ||\mathbf{o}_p^{(i)}||} \tag{2.10}$$

and

$$FDI_l^{(n')} = \frac{1}{s} \sum_{p=1}^{s} \frac{{\mathbf{o}'}_p \mathbf{o}_p^l}{||\mathbf{o}_p||\ ||\mathbf{o}_p^l||}, \tag{2.11}$$

where ${\mathbf{o}'}_p$ is the transpose of $\mathbf{o}_p$. Here, lower the value of FDI, the more is the importance of the feature or the feature subset.

2.2.3 Comparison with the scheme of Ruck *et al.* [191]

We now compare our neural network-based algorithm with that of Ruck *et al* (Appendix D). Both their method and our scheme are in a sense based on the same concept – sensitivity of network output with respect to its input. In our approach we find the output of the net after removing a feature and then measure the deviation of this output from the learnt output, but not from the target output. We have not considered the target output, because the network might not have been able to learn the target output to a desirable level. It is more logical to consider the sensitivity with respect to what has been learnt by the network. Moreover, in our approach, setting a feature value to zero is equivalent to assuming the absence of that feature. Thus, it is a conservative approach. On the other hand, Ruck *et al.* calculated the rate of change of network output with respect to the input.

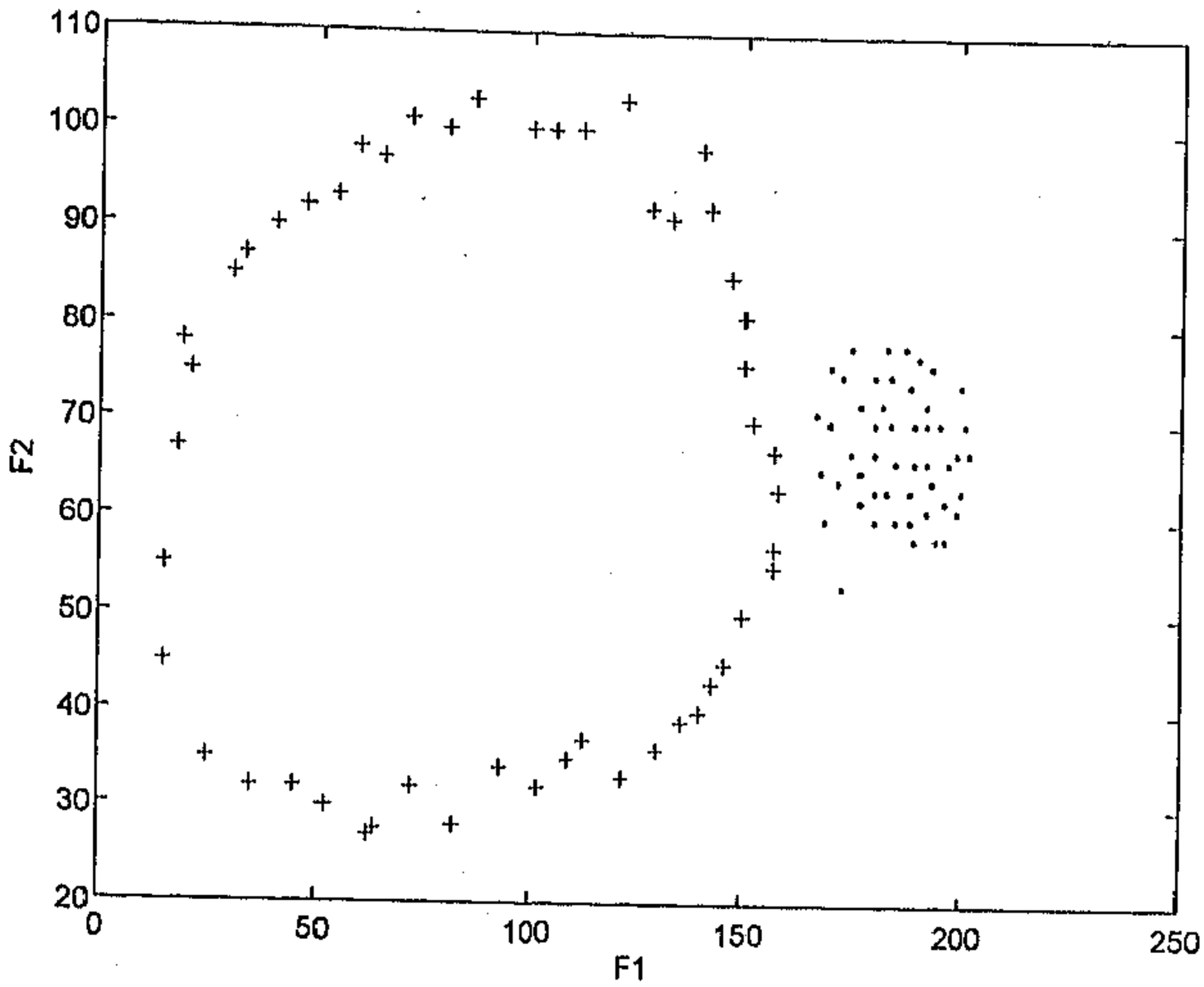

Figure 2.1: Scatter plot $F_1 - F_2$ of a two dimensional synthetic data Pat1. '+' indicates a pattern belonging to class 1 and '.' indicates a pattern belonging to class 2.

Let us assume that a system has n features for characterizing two classes. Among these features, we consider two features x_1 and x_2. Also assume that $\frac{\partial o_k}{\partial x_1} > \frac{\partial o_k}{\partial x_2}$, but the variance of x_1 is less than that of x_2. In this case the algorithm of Ruck *et al.* will usually show that x_1 is more important than x_2. Since the values of x_2 are more disperse compared to x_1, two patterns from different classes may have well apart x_2 values but close x_1 values. If we sample the domains of x_1 and x_2 uniformly, *i.e.*, into the same number of intervals, then Δx_2 will be greater than Δx_1. Δx_i is the separation between successive x_i values for points considered to compute the Λ_i (Eqn. (D.1)). It may then happen that the product $\frac{\partial o_k}{\partial x_2} \times \Delta x_2 > \frac{\partial o_k}{\partial x_1} \times \Delta x_1$, *i.e.*, x_2 is effectively more sensitive than x_1. The algorithm of Ruck *et al.* may fail here.

The method of sampling data points in [191] has another drawback. Let us take a pattern set (Pat1) in two dimension as in Fig. 2.1. The pattern set has two classes *viz.* class 1 and class 2. Consider a pattern vector $\mathbf{F}$ in the training set from class 1. If the value of feature F_1 is kept fixed and that of F_2 is varied over its domain, some of the points may be generated outside both classes 1 and 2. The network is neither trained with these pattern points nor do these points belong to any of the two

classes. Therefore, incorporation of these points in calculating the feature saliency may mislead the process of feature selection.

We illustrate the above two observations with an example. Table 2.1 depicts the ranking of the two features of Pat1 data given in Fig. 2.1. It is found that according to the index FQI (Eqn. (2.7)), the feature F_1 is more important than F_2 which is also desirable as the feature F_1 alone can separate the two pattern classes, whereas F_2 cannot do the same. On the other hand, the measure Λ (Eqn. (D.1)) of Ruck *et al.* [191] strongly recommends that the feature F_2 is more important than F_1.

Table 2.1: Values of FQI and Λ for the features of Pat1 data.

Feature used	FQI (Eqn. (2.7))	Rank	Λ (Eqn. (D.1))	Rank
1	0.678548	1	855.036690	2
2	0.676471	2	2329.231636	1

Finally, the algorithm of Ruck *et al.* ranks individual features but cannot select the best subset of $n' < n$ features. However, our algorithm ranks the features individually as being able to select the best subset of $n' < n$ features.

2.3 Experimental Results

The effectiveness of the algorithms, along with extensive comparisons, is demonstrated on four real life data sets, namely, 4-dimensional 3-class Iris [54], 3-dimensional 6-class vowel [166], 9-dimensional 4-class medical [77] and 18-dimensional 3-class mango-leaf [152] data. Anderson's Iris data [54] set contains three classes, *i.e.*, three varieties of Iris flowers, namely, Iris Setosa (IS), Iris Versicolor (IV) and Iris Virginica (IVir) consisting of 50 samples each. Each sample has four features, namely, Sepal Length (SL), Sepal Width (SW), Petal Length (PL) and Petal Width (PW). Iris data has been used in many research investigations related to pattern recognition and has become a sort of benchmark data.

The vowel data consists of a set of 871 Indian Telugu vowel sounds. These were

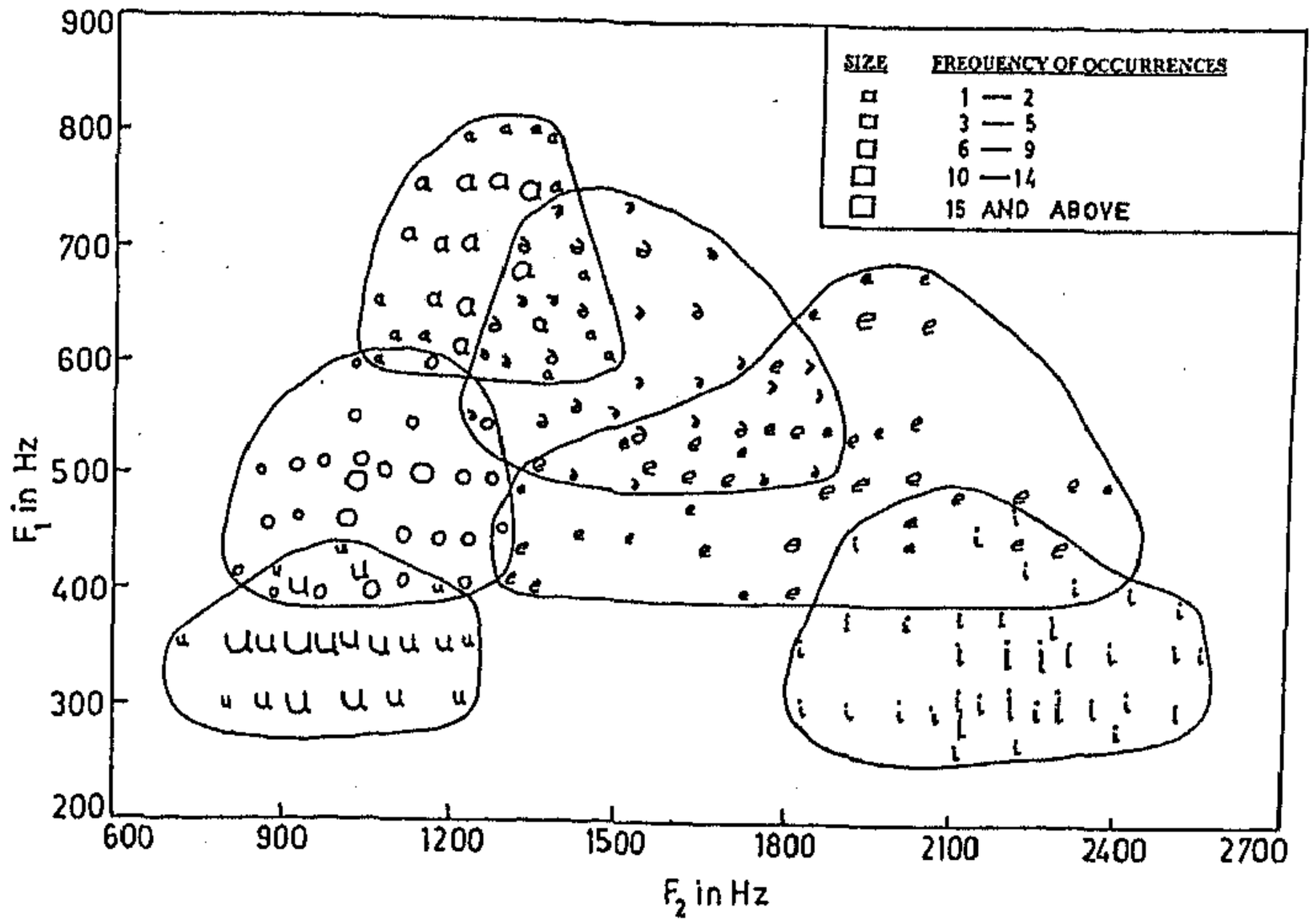

Figure 2.2: Two dimensional ($F_1 - F_2$) plot of the vowel data.

uttered in a consonant-vowel-consonant context by three male speakers in the age group of 30 to 35 years. 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. Fig. 2.2 shows the overlapping nature of the six vowel classes (*viz.*, ∂, a, i, u, e, o) in the $F_1 - F_2$ plane (for ease of depiction). The details of the data and its extraction procedure are available in [166]. This vowel data is being extensively used for two decades in the area of pattern recognition.

The medical data consisting of nine input features and four pattern classes, deals with various *Hepatobiliary disorders* [77] of 536 patient cases. The input features are 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 Hemoglobin (MCH, pg), Total Bilirubin (TBil, mg/dl) and Creatinine (CRTNN, mg/dl). The hepatobiliary disorders Alcoholic Liver Damage (ALD), Primary Hepatoma (PH), Liver Cirrhosis (LC) and Cholelithiasis (C), constitute the four classes.

Table 2.2: Values of different indices obtained by MLP for Iris data.

Feature made zero	FQI (Eqn. (2.7))	Rank	FDI (Eqn. (2.10))	Rank	Λ (Eqn. (D.1))	Rank	Misclassification
SL	0.596329	3	0.683017	4	1663.891834	4	50
SW	0.804521	2	0.531856	2	2498.589697	3	98
PL	0.926212	1	0.335803	1	4041.534737	1	100
PW	0.495294	4	0.675122	3	3580.965868	2	50

Mango-leaf data [152], on the other hand, is a set of different kinds of 166 mango-leaves with eighteen features each. It has three classes representing three kinds of mango. The feature set consists of measurements like Z-value (Z), area (A), perimeter (Pe), maximum length (L), maximum breadth (B), petiole (P), K-value (K), S-value (S), shape index (SI), L+P, L/P, L/B, (L+P)/B, A/L, A/B, A/Pe, upper midrib/lower midrib (UM/LM) and perimeter upper half/perimeter lower half (UPe/LPe). The terms 'upper' and 'lower' are used with respect to maximum breadth position.

Since each feature has different domain of values, *i.e.*, some have quite large values while others have very low even fractional values, all features are normalized to the same scale so that the differences in their domains are reduced. In other words, we apply the following transformation on each feature x',

$$x = \frac{x' - k_1}{k_2 - k_1} \tag{2.12}$$

where $k_1 = \min_i\{\min_j\{x'_{ij}\}\}$ and $k_2 = \max_i\{\max_j\{x'_{ij}\}\}$. Note that this transformation does not change the structure of the classes as it is only a change of scale and origin of the entire data.

For Iris data, the ranking of features obtained by the MLP-based scheme is shown in Table 2.2. In this investigation we have considered different network architectures and different initializations. Table 2.2 presents a typical result. We have obtained mostly the same relative ranking of the features (as shown in the Table 2.2), although the absolute values of the indices were different in different runs. An ith entry in the *misclassification* column indicates the number of data points that are wrongly

Table 2.3: Values of J and different entropy based measures using one of the features of Iris data.

Feature used	$OFEI$ (Eqn. (2.6))	Rank	$(FEI)^{av}$ (Eqn. (B.7))	Rank	J (Eqn. (A.1))	Rank
SL	0.998300	3	0.166672	3	1.622646	3
SW	1.018069	4	0.169757	4	0.668844	4
PL	0.656602	2	0.109842	2	16.056615	1
PW	0.634167	1	0.106668	1	13.061322	2

Table 2.4: Values of FQI and J associated with pairs of features for Pat2 data. (Features mentioned in column 1 are made zero for FQI and while they are used for J.)

Features made zero/ used	FQI (Eqn. (2.7))	Rank	J (Eqn. (A.1))	Rank	Misclassification
$\{x_1, x_2\}$	0.268825	1	0.354866	2	4
$\{x_3, x_4\}$	0.191828	2	1.119447	1	3

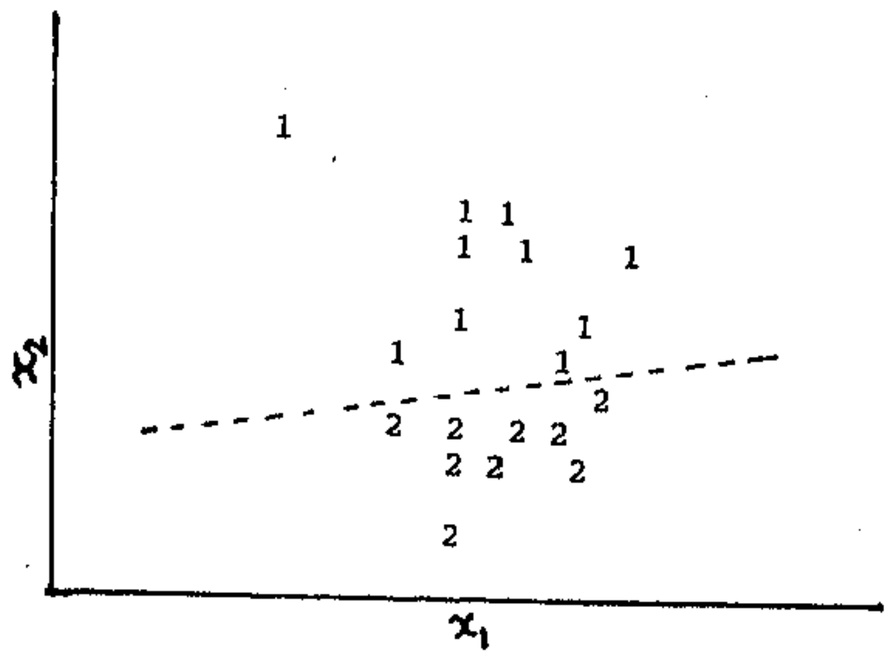

Figure 2.3: Scatter plot $x_1 - x_2$ of Pat2 data. Here, '1' and '2' indicate patterns belonging to classes 1 and 2 respectively.

classified after the ith feature is made zero. The other columns are self explanatory. The ranks obtained by FQI and FDI are different, although the first and second most important features are the same. Here feature PL is found to be the most important, while feature SW is the next most important one. On the other hand, the indices Λ (Eqn. (D.1)), $OFEI$ (Eqn. (2.6)), $(FEI)^{av}$ (Eqn. (B.7)) and J (Eqn. (A.1)) indicate features PL and PW as the most important (Tables 2.2 and 2.3). Several authors [209] also believe that features PL and PW are more important for Iris data.

Why does our MLP-based method show a different result? To get an answer to this, let us consider a four dimensional synthetic data set (Pat2). The data set has twenty data points, ten points for each of two classes. The scatter plot of the first two components (x_1 and x_2) is shown in Fig. 2.3, while Fig. 2.4 shows the scatter plot of the third and fourth components (x_3 and x_4). In the scatter plots, class 1 is indicated by '1' and class 2 is represented by '2'. Clearly, scatter plot of features x_1 and x_2 (Fig. 2.3) can be easily separated by a straight line to discriminate the classes, but their centroids are very close. While, for Fig. 2.4 although the centroids are widely separated, it requires a combination of lines to separate the two classes. Thus, for an MLP based method features x_1 and x_2 may turn out to be more important (Table 2.4) (of course, depending on the initial condition, it may not necessarily be true) than features x_3 and x_4. However, for the method based on J features x_3 and x_4 will be important. Table 2.4 shows that it is indeed the case.

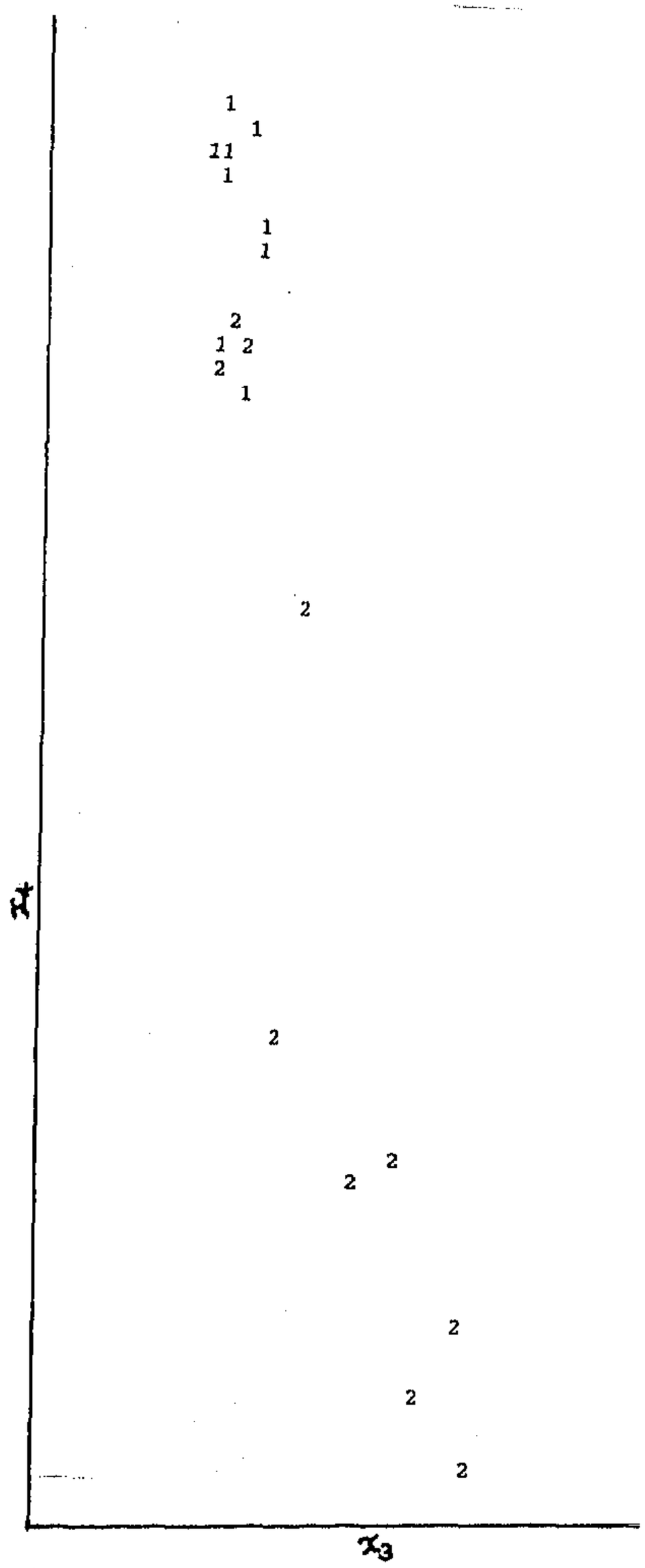

Figure 2.4: Scatter plot $x_3 - x_4$ of Pat2 data. Here, '1' and '2' indicate patterns belonging to classes 1 and 2 respectively.

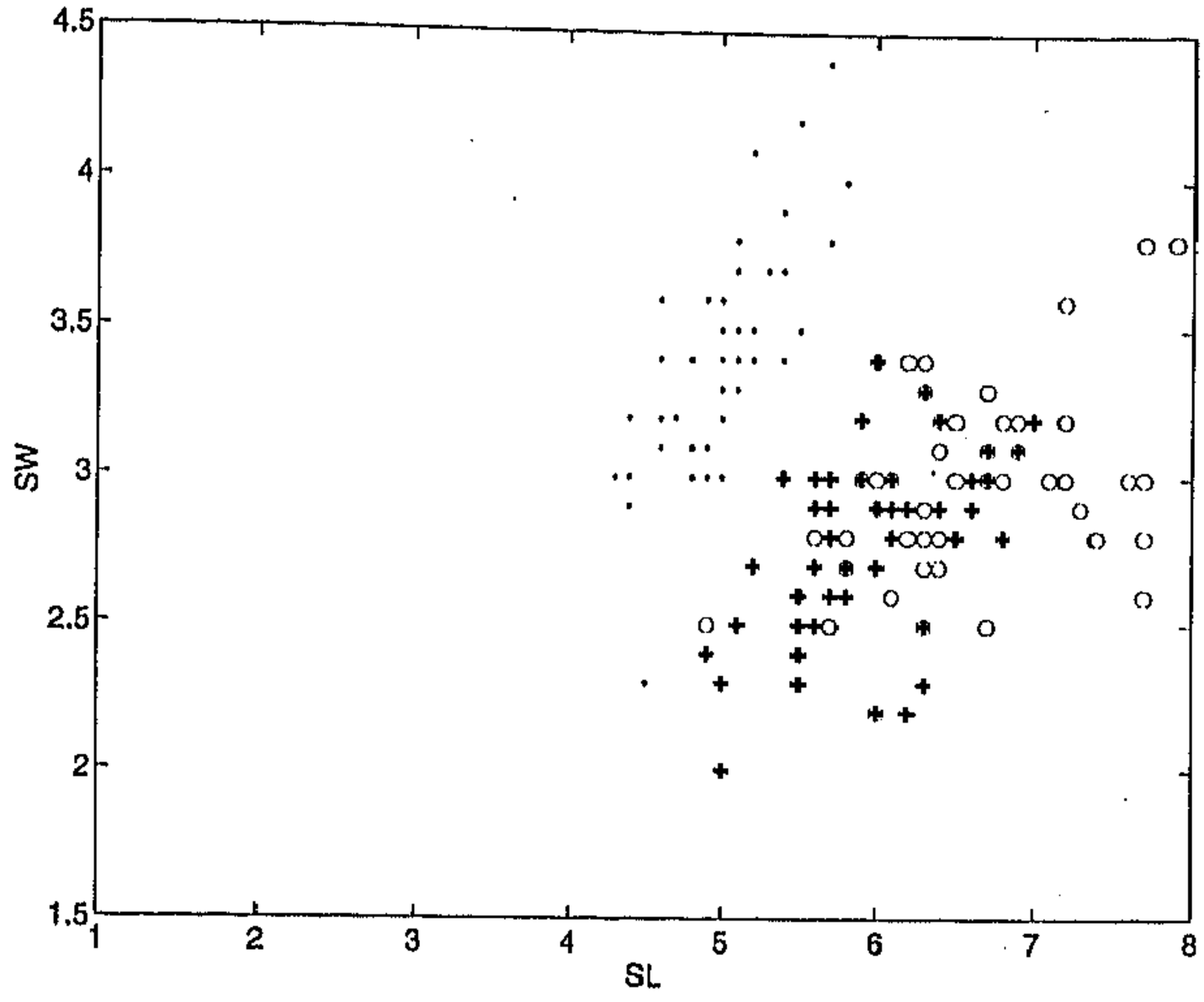

Figure 2.5: Scatter plot $SL - SW$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

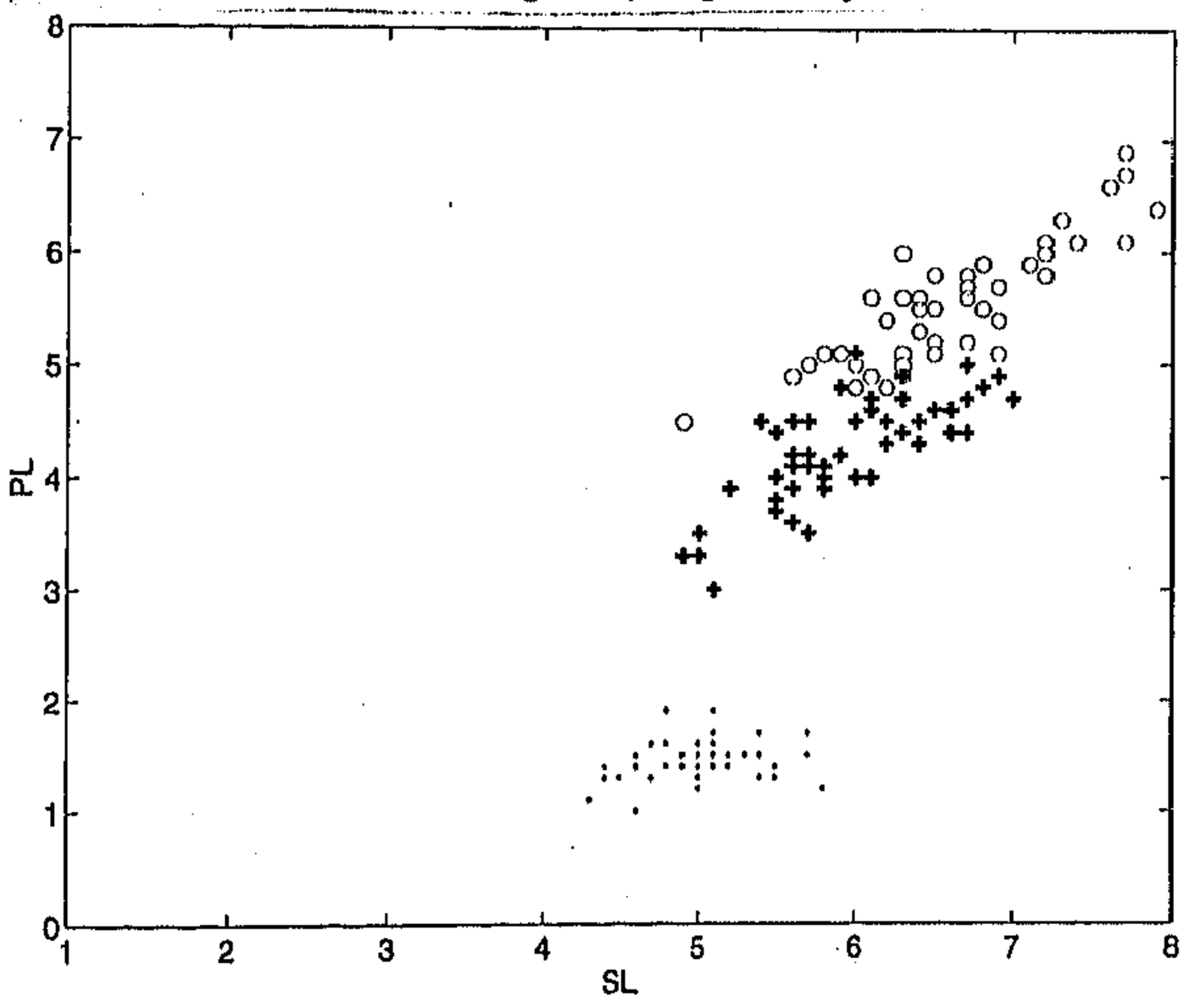

Figure 2.6: Scatter plot $SL - PL$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

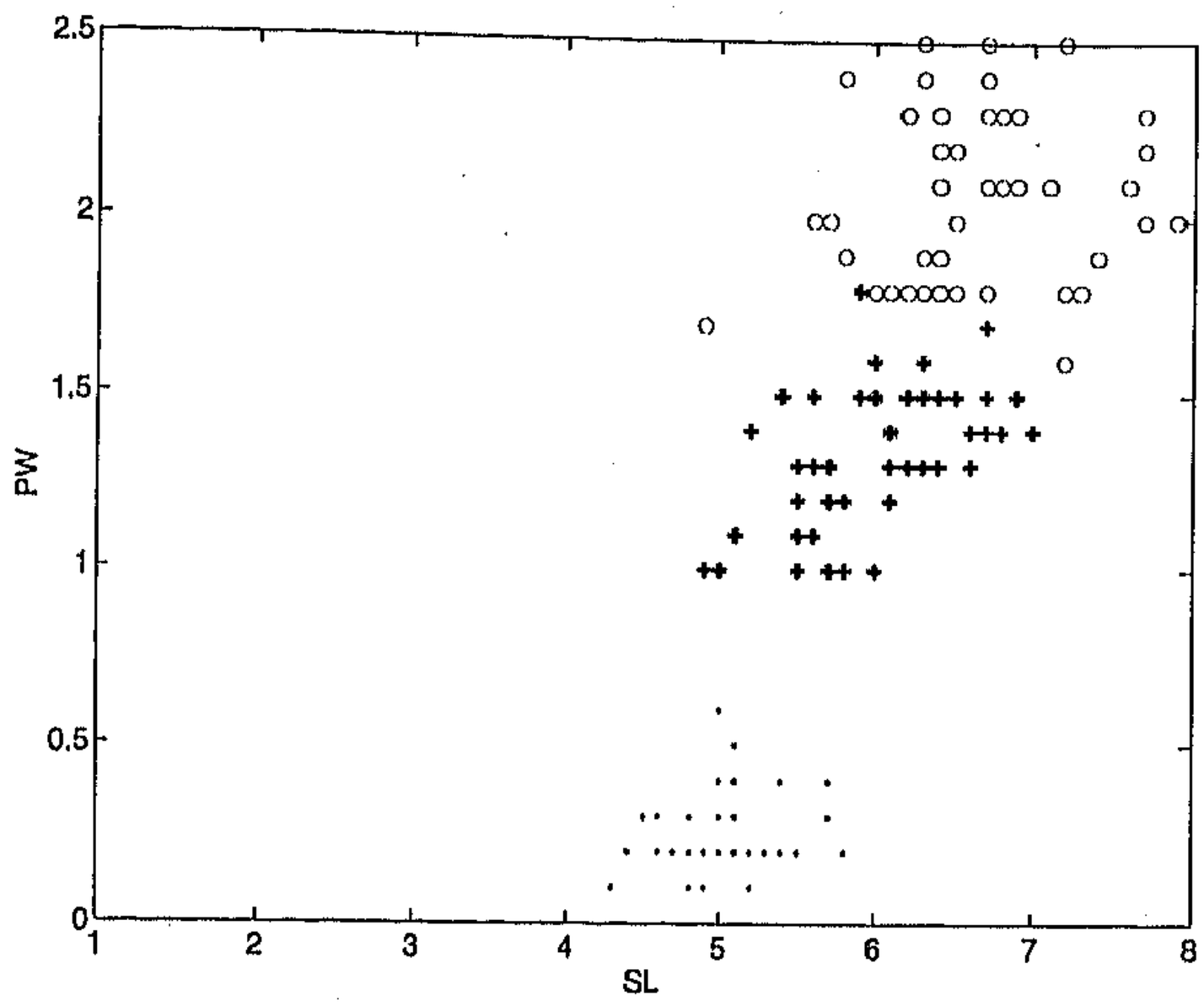

Figure 2.7: Scatter plot $SL - PW$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

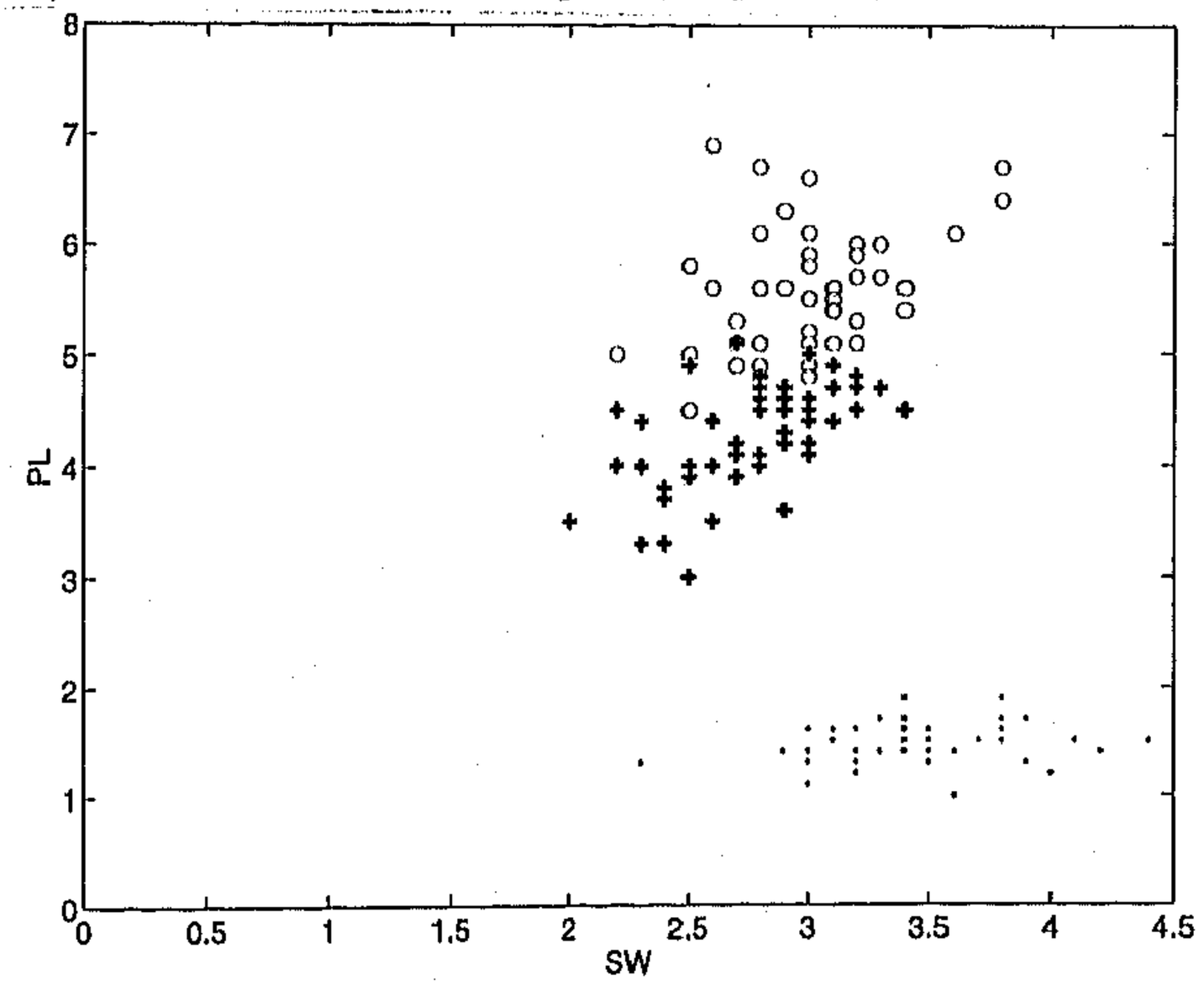

Figure 2.8: Scatter plot $SW - PL$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

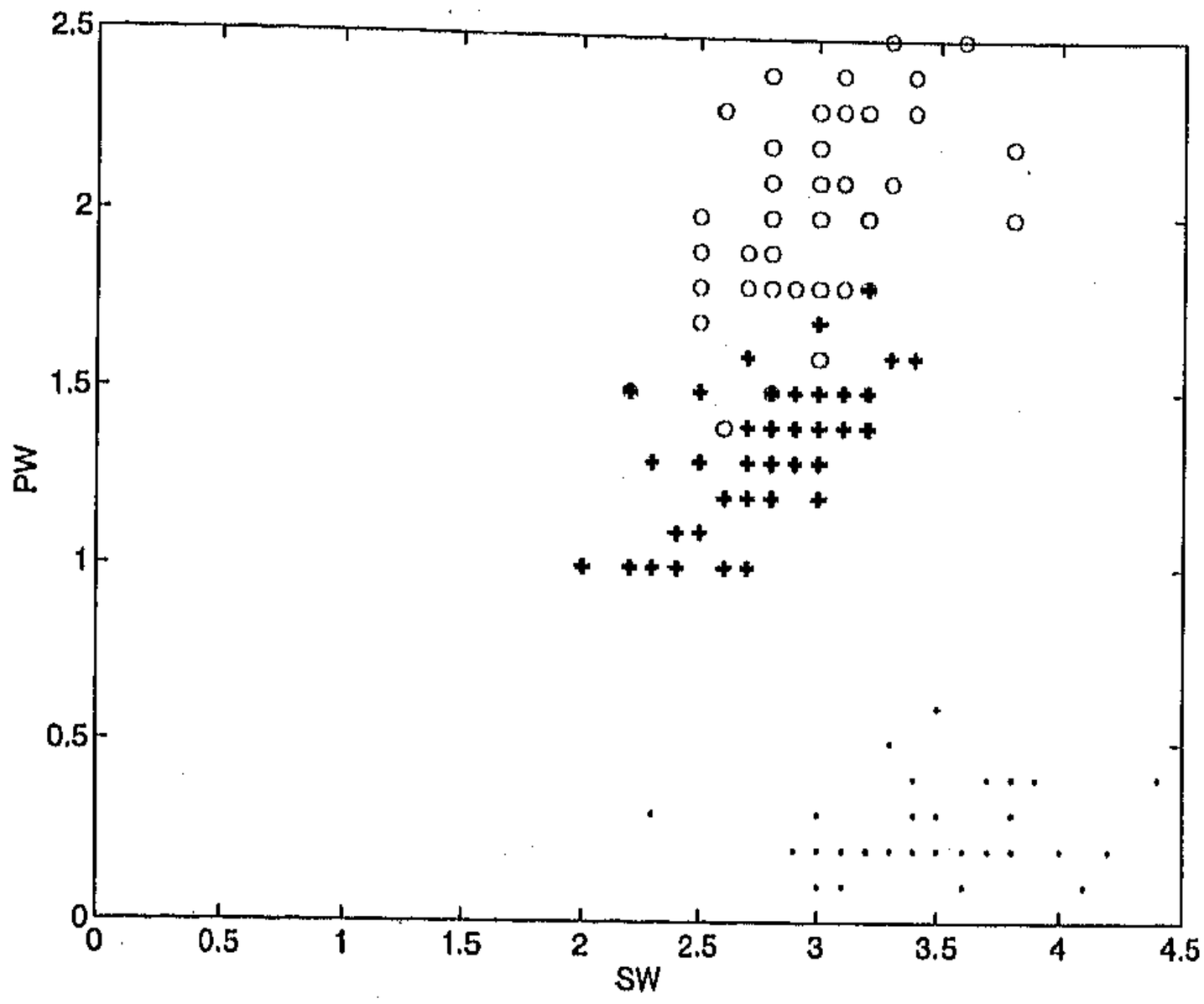

Figure 2.9: Scatter plot $SW - PW$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

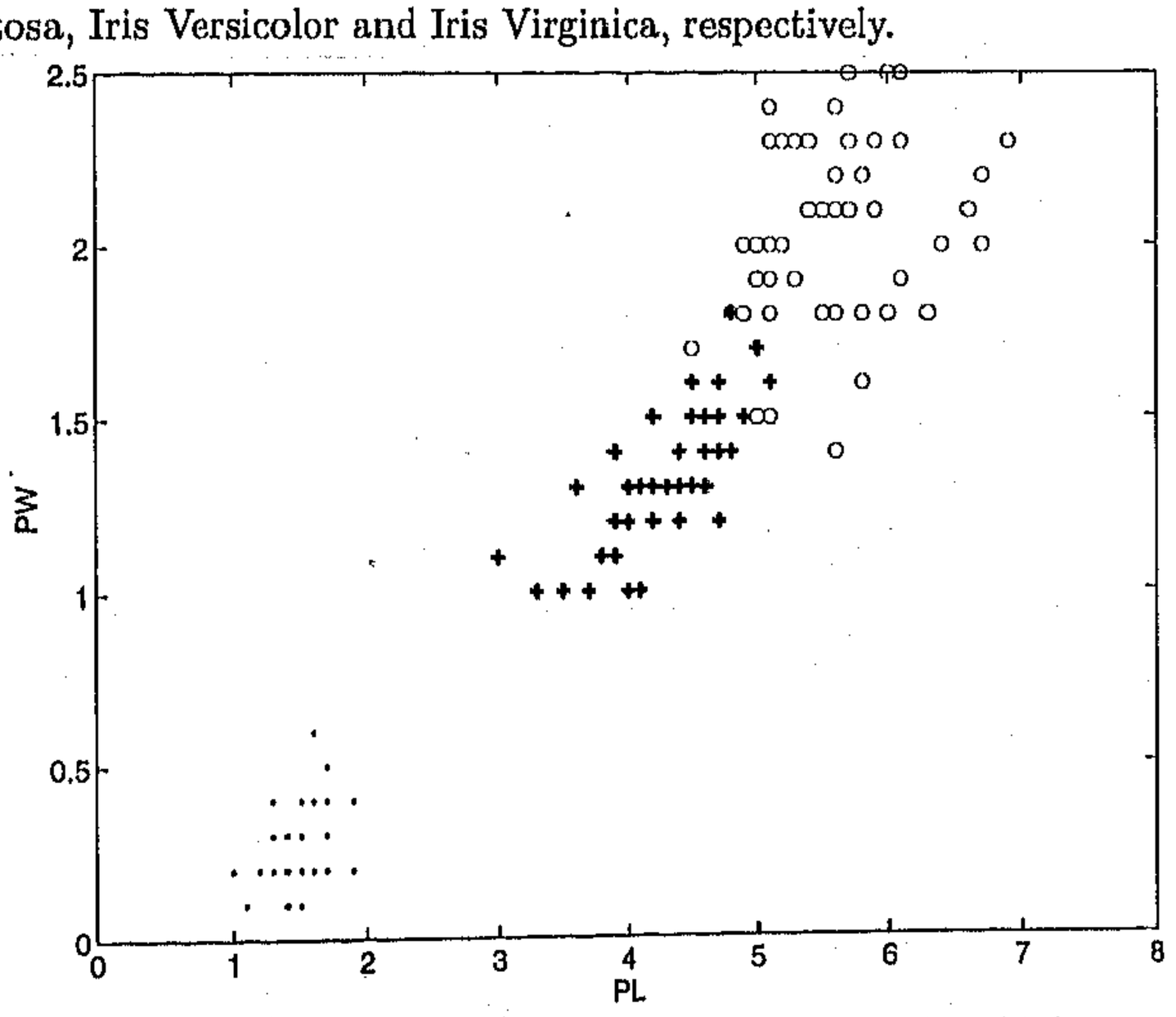

Figure 2.10: Scatter plot $PL - PW$ of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

Let us now look at the scatter plots (Figs. 2.5–2.10) of Iris data. Among these, Figs. 2.8 (SW–PL) and 2.10 (PL–PW) are found to be the two best plots. In Figs. 2.8 and 2.10 different classes are separated by dotted lines. It is seen from both these figures that classes can be separated by two lines with only 2-3 errors, but the centroids of the classes as represented by features PL and PW are much more separated than those represented by features SW and PL. This explains why our MLP based method shows features SW and PL as more important while others indicate features PL and PW.

Table 2.5: Values of different indices obtained by MLP when a pair of features is set to zero for Iris data.

Features made zero	FQI (Eqn. (2.8))	Rank	FDI (Eqn. (2.11))	Rank	Misclassification
{SL,SW}	0.903465	3	0.351944	3	100
{SL,PL}	0.756021	4	0.335055	1	100
{SL,PW}	0.564501	6	0.684518	5	51
{SW,PL}	1.035633	1	0.452740	4	100
{SW,PW}	0.593168	5	0.806045	6	61
{PL,PW}	0.923468	2	0.343355	2	100

For feature ranking we have considered the effect of only one feature at a time on the performance of the network. Whereas, for the feature selection problem (when we want to select the most important, say, n' features) we need to consider the combined effect of feature subsets. Thus, the set of features with rank $\leq n'$ may not necessarily be the optimal set of n' features. Table 2.5 depicts the results obtained by setting two of the features to zero, *i.e.*, the combined effect of two features on the performance of the network. Table 2.5 reveals that for Iris data, feature pair {SW,PL} is the most important. Based on individual rank also these two features PL and SW are found to be the most important.

Tables 2.6 and 2.7 depict the ranking of the individual features obtained by various indices for the vowel data. In all the cases, except for Λ (Eqn. (D.1)), the ranking is found to be $F_2 > F_1 > F_3$. Note that here $x > y$ means feature x is more important than y. The relative importance of the feature pairs obtained by FQI and FDI,

Table 2.6: Values of different indices obtained by MLP for vowel data.

Feature made zero	FQI (Eqn. (2.7))	Rank	FDI (Eqn. (2.10))	Rank	Λ (Eqn. (D.1))	Rank	Misclassification
F_1	0.924619	2	0.392495	2	90793.883605	1	548
F_2	0.991573	1	0.349314	1	25834.668102	2	673
F_3	0.681320	3	0.566575	3	11340.452341	3	545

Table 2.7: Values of J and different entropy based measures using one of the features of vowel data.

Feature used	$OFEI$ (Eqn. (2.6))	Rank	$(FEI)^{av}$ (Eqn. (B.7))	Rank	J (Eqn. (A.1))	Rank
F_1	0.641289	2	0.116589	2	15.071362	2
F_2	0.620437	1	0.106668	1	17.874560	1
F_3	1.043592	3	0.158311	3	10.156732	3

Table 2.8: Values of different indices obtained by MLP when pair of features is set to zero for vowel data.

Features made zero	FQI (Eqn. (2.8))	Rank	FDI (Eqn. (2.11))	Rank	Misclassification
$\{F_1, F_2\}$	1.072675	1	0.180790	1	782
$\{F_1, F_3\}$	0.906033	3	0.224710	3	699
$\{F_2, F_3\}$	1.054548	2	0.222150	2	720

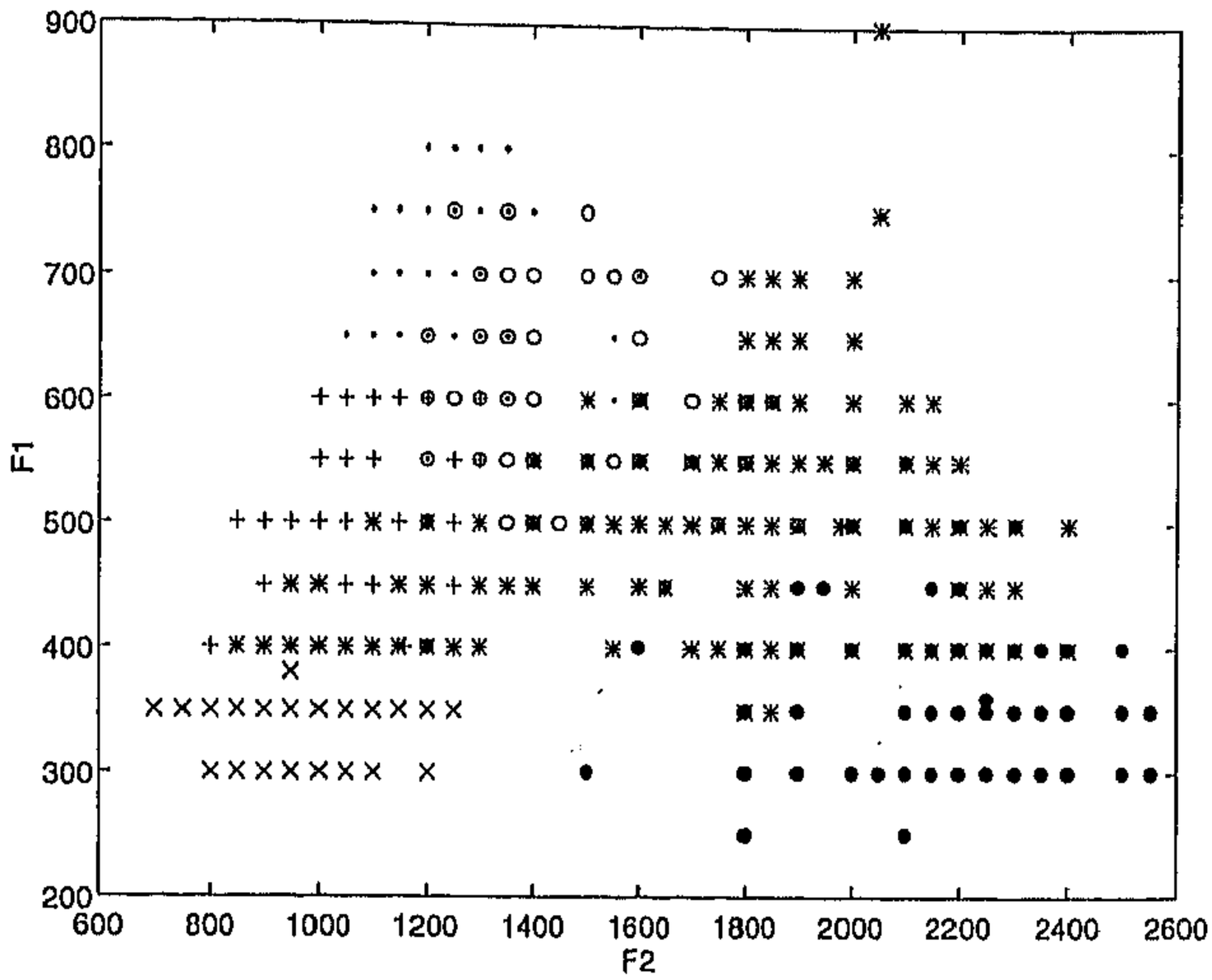

Figure 2.11: Scatter plot $F_2 - F_1$ of vowel data. Here 'o', '.', '•', 'x', '*' and '+' represent classes ∂, a, i, u, e and o, respectively. This figure is the same as Fig. 2.2. The only difference is that here approximate boundaries are not drawn.

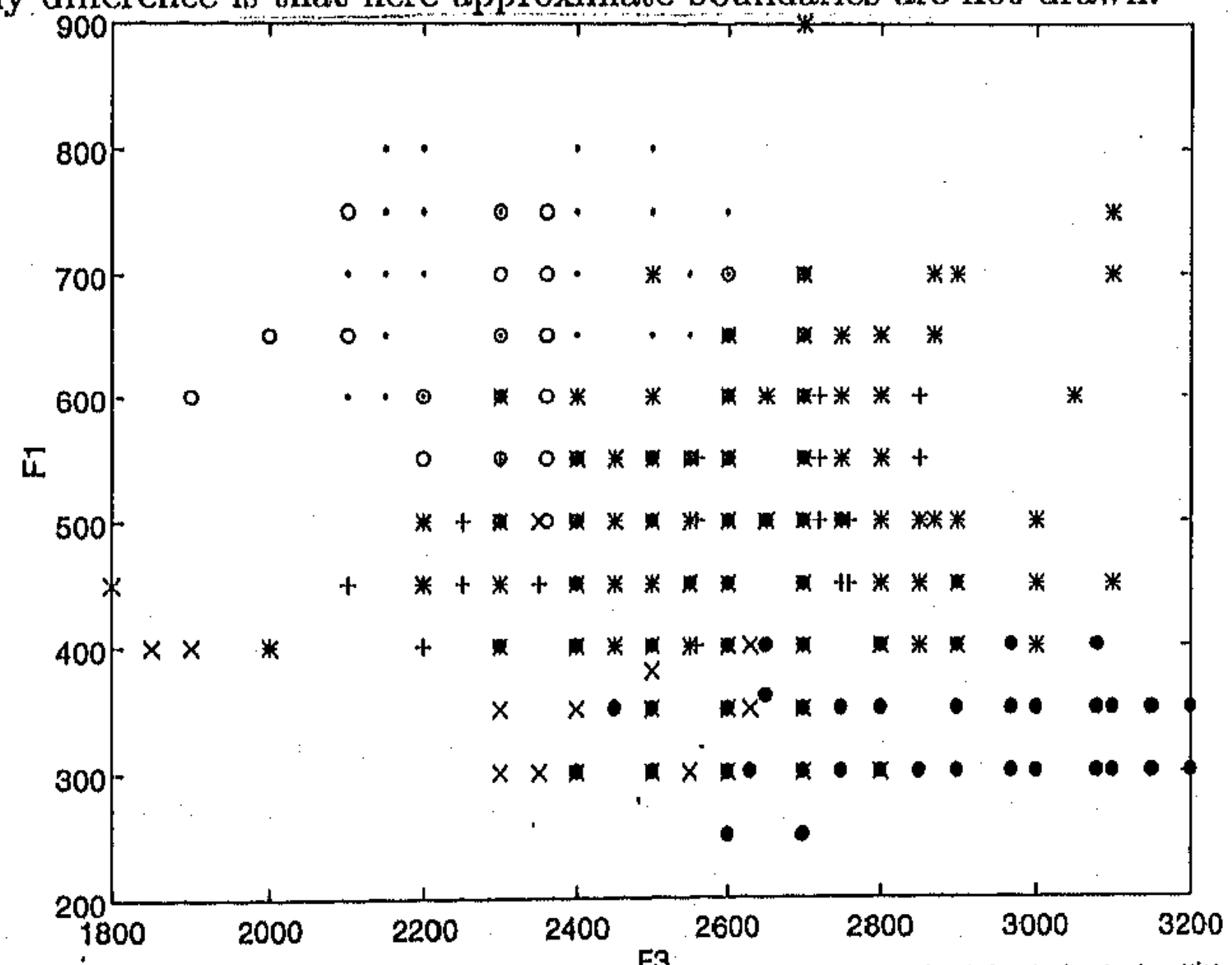

Figure 2.12: Scatter plot $F_3 - F_1$ of vowel data. Here 'o', '.', '•', 'x', '*' and '+' represent classes ∂, a, i, u, e and o, respectively.

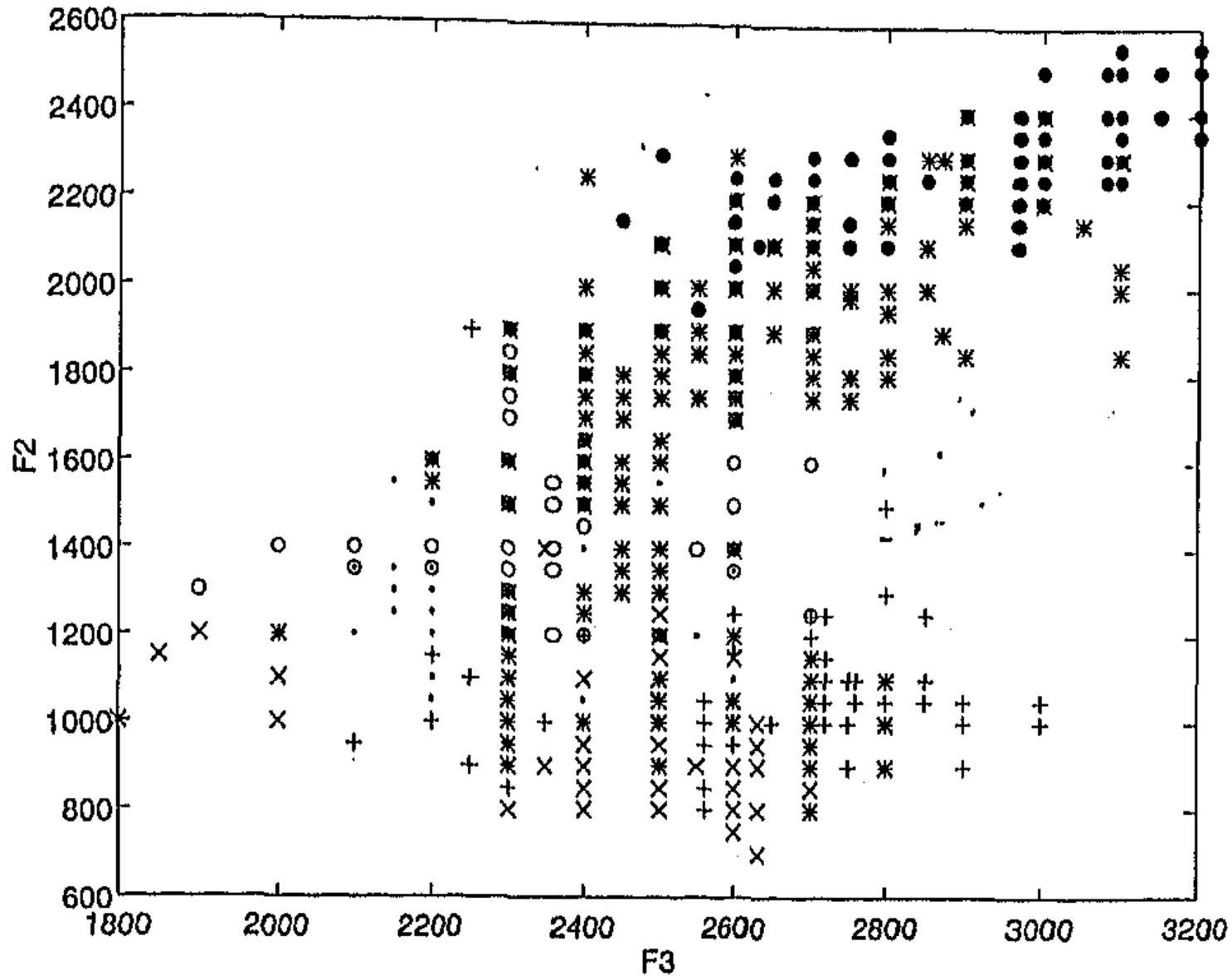

Figure 2.13: Scatter plot $F_3 - F_2$ of vowel data. Here 'o', '.', '•', 'x', '*' and '+' represent classes ∂, a, i, u, e and o, respectively.

for the said data, is shown in Table 2.8. It is found that $\{F_1, F_2\}$ is the best feature obtained by these two indices, which is in agreement with the results on number of misclassifications. This fact is also supported by the scatter plots in Figs. 2.11–2.13, where the one corresponding to $\{F_1, F_2\}$ shows better discrimination ability. is seen to be the best of all with respect to class structures. Note that F_2, being the best individual feature (Table 2.6), is present in both the best and the second best feature pairs.

To establish further the effectiveness of our scheme, let us now consider the medical and mango-leaf data. For mango-leaf data, the results on ranking obtained by FQI and FDI, as shown in Table 2.9, are not exactly identical, but almost the same. However, the ranking obtained by Λ is found to differ from those obtained by FQI and FDI. The difference in the ranking by Λ with that by FQI or FDI may be due to the complex class structure of the data set. In order to assess the validity of the ranks (based on FQI or FDI) we have trained the network with (a) top three features, (b) top five features, (c) top six features, (d) the features with ranks 4,5,6,7,8 and 9, and (e) all eighteen features. In each case, the network was trained for 60,000 epochs. The results (number of misclassifications) are presented in Table

Table 2.9: Values of different indices obtained by MLP for mango-leaf data.

Feature made zero	*FQI* (Eqn. (2.7))	Rank	*FDI* (Eqn. (2.10))	Rank	Λ (Eqn. (D.1))	Rank	Misclassification
Z	0.125161	9	0.970644	9	10259.124586	14	25
Λ	0.526458	5	0.737481	4	13607.626761	12	66
Pe	0.353275	6	0.883209	6	2685.724258	16	44
L	0.981794	1	0.406196	1	43191.505706	4	130
B	0.089942	13	0.990154	12	19161.967354	10	23
P	0.335267	7	0.889027	7	82910.026417	1	46
K	0.000717	17	0.999999	17	830.137520	17	18
S	0.000160	18	1.000000	18	531.974334	18	18
SI	0.888633	2	0.723810	3	61851.395600	2	66
L+P	0.553753	4	0.770063	5	23367.307882	8	59
L/P	0.181140	8	0.956033	8	28306.887051	7	30
L/B	0.108732	10	0.988997	11	21500.352227	9	24
(L+P)/B	0.092504	11	0.991579	13	10867.592324	13	23
Λ/L	0.047141	15	0.997323	15	14402.327757	11	20
Λ/B	0.683353	3	0.678076	2	42520.772601	5	82
Λ/Pe	0.012551	16	0.999815	16	7677.518133	15	19
UM/LM	0.090810	12	0.986581	10	57658.592076	3	23
UPe/LPe	0.058416	14	0.995781	14	35721.791323	6	18

Table 2.10: Misclassifications obtained by a trained MLP (for 60,000 epochs) with different feature subsets of mango-leaf data.

Cases	Features used	Misclassification
(a)	{L,SI,A/B}	49
(b)	{L,SI,A/B,L+P,A}	25
(c)	{L,SI,A/B,L+P,A,Pe}	40
(d)	{L+P,A,Pe,P,L/P,Z}	41
(e)	All	31

Table 2.11: Values of J and different entropy based measures using one of the features of mango-leaf data.

Feature used	$OFEI$ (Eqn. (2.6))	Rank	$(FEI)^{av}$ (Eqn. (B.7))	Rank	J (Eqn. (A.1))	Rank
Z	1.053797	7	0.146951	7	0.116022	11
A	1.082116	9	0.152861	10	0.424932	5
Pe	1.133981	16	0.155923	16	0.097274	13
L	1.018112	1	0.144551	3	0.351634	6
B	1.026228	2	0.142962	1	0.458305	3
P	1.092866	13	0.154936	13	0.023515	16
K	1.036342	5	0.146140	6	0.009207	18
S	1.035661	4	0.146018	5	0.014633	17
SI	1.092510	12	0.151614	9	0.793117	1
L+P	1.088604	11	0.155043	12	0.242699	8
L/P	1.203172	17	0.166411	18	0.042566	14
L/B	1.130467	15	0.155414	14	0.188824	9
(L+P)/B	1.169015	18	0.160234	17	0.034397	15
A/L	1.051984	6	0.146503	4	0.482680	2
A/B	1.033192	3	0.143206	2	0.342572	7
A/Pe	1.082951	10	0.152190	11	0.433072	4
UM/LM	1.123898	14	0.154561	15	0.107068	12
UPe/LPe	1.059308	8	0.148626	8	0.173945	10

2.10. Table 2.10 clearly reveals the effectiveness of the ranks obtained by FQI or FDI. Here, case (a) just signifies the inadequacy of the features. The top six features (case (c)) clearly show better discriminating ability than the six other features with rank 4, 5, 6, 7, 8 and 9 (case (d)). Table 2.10 also shows that the top five features together (case (b)) are more effective than taking the top six or all the eighteen features together. This may be due to the redundancy of the system while taking the top six or all the features together. The results (ranking) obtained by $OFEI$, $(FEI)^{av}$ and J for the data set are found to be quite different from that based on FQI and FDI (Tables 2.9 and 2.11). Comparing the features with ranks ≤ 5 we find that the ranks obtained by the entropy-based methods are closer to those based on FQI or FDI.

Table 2.12: Values of different indices obtained by MLP for medical data.

Feature made zero	FQI (Eqn. (2.7))	Rank	FDI (Eqn. (2.10))	Rank	Λ (Eqn. (D.1))	Rank	Misclassification
GOT	0.467849	4	0.781531	4	28554.541437	2	229
GPT	0.201160	9	0.925929	9	24145.169070	4	150
LDH	0.248336	8	0.912192	8	9715.267869	8	158
GGT	0.515683	2	0.736569	3	27262.535413	3	228
BUN	0.443288	5	0.801517	5	18196.655595	5	211
MCV	0.502787	3	0.732989	2	5495.854577	9	237
MCH	0.750239	1	0.642894	1	11740.906010	7	287
TBil	0.249424	7	0.911978	7	35745.375484	1	174
CRTNN	0.358155	6	0.862072	6	12708.130732	6	187

In the case of medical data, the results obtained by FQI and FDI, as in the case of mango-leaf data, are almost identical, but differ from that obtained by Λ (Table 2.12). Although the results on ranking obtained by FQI and FDI does not exactly get reflected by the number of misclassifications, their difference is less as compared to that obtained when Λ is considered. As in the case of mango-leaf data, we have trained the network with (a) top three features, (b) top five features, (c) top six features, (d) the features with ranks 4,5,6,7,8 and 9, and (e) all nine features. In each case, the network is trained for 100,000 epochs. Table 2.13 shows that the

Table 2.13: Misclassifications obtained by a trained MLP (for 100,000 epochs) with different feature subsets of the medical data.

Cases	Features used	Misclassification
(a)	{GGT,MCV,MCH}	242
(b)	{GOT,GGT,BUN,MCV,MCH}	172
(c)	{GOT,GGT,BUN,MCV,MCH,CRTNN}	159
(d)	{GOT,GPT,LDH,BUN,TBil,CRTNN}	209
(e)	All	111

Table 2.14: Values of J and different entropy based measures using one of the features of the medical data.

Feature used	$OFEI$ (Eqn. (2.6))	Rank	$(FEI)^{av}$ (Eqn. (B.7))	Rank	J (Eqn. (A.1))	Rank
GOT	1.240253	8	0.291024	8	0.819359	1
GPT	1.241028	9	0.298179	9	0.808419	3
LDH	1.235413	5	0.271894	5	0.799125	5
GGT	1.239180	7	0.289741	7	0.775119	8
BUN	1.230112	4	0.258881	3	0.781582	6
MCV	1.218782	2	0.260192	4	0.771485	9
MCH	1.228197	3	0.258190	2	0.815416	2
TBil	1.213694	1	0.241289	1	0.807519	4
CRTNN	1.237824	6	0.289124	6	0.778349	7

performance of case (d) is not only worse than case (c), but also than case (b); thereby signifying the effectiveness of FQI and FDI. However, unlike mango-leaf data, consideration of all the features (case (e)) leads to the best performance. Note that the ranks obtained by $OFEI$, $(FEI)^{av}$ and J for the data set are found to be quite different from that based on FQI and FDI (Tables 2.12 and 2.14).

Table 2.15: Misclassifications obtained by MLP (after trained for 60,000 epochs) with top 5 features of mango-leaf data obtained by different ranking schemes described.

Ranking based on	Features used	Misclassification
FQI/FDI	{L,SI,A/B,L+P,A}	25
Entropy	{L,B,A/B,S,K}	32
J	{SI,A/L,B,A/Pe,A}	33
Λ	{P,SI,UM/LM,L,A/B}	43

Table 2.16: Misclassifications obtained by MLP (after trained for 100,000 epochs) with top 5 features of the medical data obtained by different ranking schemes described.

Ranking based on	Features used	Misclassification
FQI/FDI	{GOT,GGT,BUN,MCV,MCH}	172
Entropy	{LDH,BUN,MCV,MCH,TBil}	190
J	{GOT,GPT,LDH,MCH,TBil}	196
Λ	{GOT,GPT,GGT,BUN,TBil}	201

In order to establish the superiority of the MLP-based scheme (using the indices FQI and FDI) over the others, we have trained the same network (*i.e.*, with the same architecture and initialization) separately with a few good features obtained by (i) FQI or FDI, (ii) fuzzy entropies, (iii) J and (iv) Λ, for both mango-leaf and medical data. Table 2.15 reports the results for mango-leaf data with the top five features when the networks are trained for 60,000 epochs. The top five features selected by FQI or FDI are found to produce the least number of *misclassification* (Table 2.15). Similar experiment is performed for the medical data, where we have

trained the networks for 100,000 epochs using the top five features. Here also our MLP based scheme (FQI or FDI) outperforms the others (Table 2.16).

2.4 Conclusions and Discussion

We have described a scheme for both *feature ranking* and *selection* based on a multilayer perceptron network. The scheme is based on the idea that the effect of a missing feature (setting the feature value to zero) on the output of a trained network will depend heavily on the importance of the feature. In fact, the more important a feature is, the more will be its impact on the output of the network. We have also provided a scheme for selecting a feature subset based on the same idea. A feature subset may be regarded as good if the network outputs are heavily affected by assuming the absence of these features (in the subset), *i.e.*, setting the values of the features to zero. In addition to this, we have modified an existing fuzzy entropy-based method. The superiority of the MLP-based scheme has been established empirically with several synthetic and real life data sets. The novelty of the MLP-based scheme and its difference to the method of Ruck *et al.*, which is also based on a similar concept, have been analyzed.

Chapter 3

Neuro-fuzzy Supervised Feature Selection

3.1 Introduction

In the previous chapter we have described two methods for feature selection based on the theories of fuzzy sets and neural networks individually. This chapter is an attempt in integrating these two theories for performing the said task [154, 153, 10, 38]. The methodology involves connectionist minimization of a fuzzy feature evaluation index. Its effectiveness is demonstrated both theoretically and experimentally.

First of all, a fuzzy set theoretic evaluation index is defined in terms of individual class membership. Its performance is compared with that of Eqn. B.7 [152] for ranking the features (or subsets of features). Its relation with Mahalanobis distance and divergence measure is demonstrated. Then we provide a new connectionist model to perform the task of optimizing the aforesaid fuzzy evaluation index by incorporating weighted distance for computing class membership values. This optimization process results in a set of weighting coefficients representing the importance of the individual features. These weighting coefficients lead to a transformation of the feature space for better modeling the class structures. The performance of the system is then theoretically analyzed. This includes derivation of upper and lower bounds of the evaluation index, and determining its relation with interclass distances and weighting coefficients. The effectiveness of the algorithms is demonstrated on the same set of four different data, described in Chapter 2, namely, vowel, Iris, medical and mango-leaf data. The results are verified, along with comparisons, independently with scatter plots and k-NN classifier for different values of k.

Section 3.2 defines a fuzzy feature evaluation index and weighted membership function. In Section 3.3, the neural network model is described. The theoretical analysis of the neuro-fuzzy system is provided in Section 3.4. Experimental results are analyzed in Section 3.5. The chapter is concluded with Section 3.6.

3.2 Fuzzy Feature Evaluation Index and Weighted Membership Function

Let us consider an n-dimensional feature space Ω containing $x_1, x_2, x_3, \ldots, x_i, \ldots, x_n$ features (components). Let there be M classes $C_1, C_2, C_3, \ldots, C_k, \ldots, C_M$. The fea-

ture evaluation index for a subset ($\Omega_{\mathbf{x}}$) containing few of these n features is defined as

$$E = \sum_{k} \sum_{\mathbf{x} \in C_k} \frac{H_k(\mathbf{x})}{\sum_{k' \neq k} H_{kk'}(\mathbf{x})} \times \alpha_k, \tag{3.1}$$

where $\mathbf{x}$ is constituted by the features of $\Omega_{\mathbf{x}}$ only.

$$H_k(\mathbf{x}) = \mu_{C_k}(\mathbf{x}) \times (1 - \mu_{C_k}(\mathbf{x})) \tag{3.2}$$

and

$$H_{kk'}(\mathbf{x}) = \frac{1}{2}[\mu_{C_k}(\mathbf{x}) \times (1 - \mu_{C_{k'}}(\mathbf{x}))] + \frac{1}{2}[\mu_{C_{k'}}(\mathbf{x}) \times (1 - \mu_{C_k}(\mathbf{x}))]. \tag{3.3}$$

$\mu_{C_k}(\mathbf{x})$ and $\mu_{C_{k'}}(\mathbf{x})$ are the membership values of the pattern $\mathbf{x}$ in classes C_k and $C_{k'}$ respectively. α_k is the normalizing constant for class C_k which takes care of the effect of relative sizes of the classes.

Note that, H_k is zero (minimum) if $\mu_{C_k} = 1$ *or* 0, and is 0.25 (maximum) if $\mu_{C_k} = 0.5$. On the other hand, $H_{kk'}$ is zero (minimum) when $\mu_{C_k} = \mu_{C_{k'}} = 1$ *or* 0, and is 0.5 (maximum) for $\mu_{C_k} = 1$, $\mu_{C_{k'}} = 0$ or *vice-versa*.

Therefore, the term $\frac{H_k}{\sum_{k' \neq k} H_{kk'}}$ is minimum if $\mu_{C_k} = 1$ and $\mu_{C_{k'}} = 0$ for all $k' \neq k$ *i.e.*, if the ambiguity in the belongingness of a pattern $\mathbf{x}$ to classes C_k and $C_{k'}$ $\forall k' \neq k$ is minimum (the pattern belongs to only one class). It is maximum when $\mu_{C_k} = 0.5$ for all k. In other words, the value of E decreases as the belongingness of the patterns increases for only one class (*i.e.*, compactness of individual classes increases) and at the same time decreases for other classes (*i.e.*, separation between classes increases). E increases when the patterns tend to lie at the boundaries between classes (*i.e.*, $\mu \rightarrow 0.5$). Our objective is, therefore, to select those features for which the value of E is minimum. E is computed over all the samples in the feature space irrespective of the size of the classes. Therefore, it is expected that the contribution of a class of bigger size (*i.e.* with larger number of samples) will be more in the computation of E. As a result, the index value will be more biased by the bigger classes; which might affect the process of feature selection. In order to overcome this *i.e.*, to normalize this effect of the size of the classes, a factor α_k, corresponding to the class C_k, is introduced. In the present investigation, we have chosen $\alpha_k = 1 - P_k$, where P_k is *a priori* probability for class C_k. However, other expressions like $\alpha_k = \frac{1}{|C_k|}$ or $\alpha_k = \frac{1}{P_k}$ could also have been used.

The membership ($\mu_{C_k}(\mathbf{x})$) of a pattern $\mathbf{x}$ to a class C_k is defined with a multi-dimensional π-function [162] which is given by,

$$\begin{aligned} \mu_{C_k}(\mathbf{x}) &= 1 - 2d_k^2(\mathbf{x}) \quad 0 \leq d_k(\mathbf{x}) < \tfrac{1}{2}, \\ &= 2[1 - d_k(\mathbf{x})]^2 \quad \tfrac{1}{2} \leq d_k(\mathbf{x}) < 1, \\ &= 0 \quad \textit{otherwise.} \end{aligned} \tag{3.4}$$

$d_k(\mathbf{x})$ is the distance of the pattern $\mathbf{x}$ from $\mathbf{m}_k$ (the center of class C_k). It can be defined as,

$$d_k(\mathbf{x}) = \left[\sum_i \left(\frac{x_i - m_{ki}}{\lambda_{ki}}\right)^{r_k}\right]^{\frac{1}{r_k}}, \quad r_k > 0, \tag{3.5}$$

where

$$\lambda_{ki} = 2\max_{\mathbf{x} \in C_k}[|x_i - m_{ki}|], \tag{3.6}$$

and

$$m_{ki} = \frac{\sum_{\mathbf{x} \in C_k} x_i}{|C_k|}. \tag{3.7}$$

Eqns. (3.4)-(3.7) are such that the membership $\mu_{C_k}(\mathbf{x})$ of a pattern $\mathbf{x}$ is 1 if it is located at the mean of C_k, and 0.5 if it is at the boundary (*i.e.*, ambiguous region) for a symmetric class structure.

In practice, the class structure may not be symmetric. In that case, the membership values of some patterns at the boundary of the class will be greater than 0.5. Also, some patterns of other classes may have membership values greater than 0.5 for the class under consideration. For handling this undesirable situation, the membership function corresponding to a class needs to be transformed so that it can model the real life class structures appropriately. For this purpose, we have incorporated a weighting factor corresponding to a feature which transforms the feature space in such a way that the transformed membership functions model the class structures appropriately. Note that, this incorporation of weighting factors makes the method of modeling the class structures more generalized; a symmetric class structure being a special case.

For this purpose, we define weighted distance from Eqn. (3.5) as,

$$d_k(\mathbf{x}) = \left[\sum_i w_i^{r_k} \left(\frac{x_i - m_{ki}}{\lambda_{ki}}\right)^{r_k}\right]^{\frac{1}{r_k}}, \quad w_i \in [0, 1]. \tag{3.8}$$

The membership values (μ) of the sample points of a class become dependent on w_i. The values of w_i (< 1) make the function of Eqn. (3.4) flattened along the axis of x_i. The lower the value of w_i, the higher is the extent of flattening. In the extreme case, when $w_i = 0$, $d_k = 0$ and $\mu_{C_k} = 1$ for all the patterns.

In pattern recognition literature, the weight w_i (Eqn. (3.8)) can be viewed to reflect the relative importance of the feature x_i in measuring the similarity (in terms of distance) of a pattern to a class. It is such that the higher the value of w_i, the more is the importance of x_i in characterizing/discriminating a class/between classes. $w_i = 1$ (0) indicates most (least) importance of x_i.

Therefore, the compactness of the individual classes and the separation between the classes as measured by E (Eqn. (3.1)) is now essentially a function of w ($= [w_1, w_2, \ldots w_n]$), if we consider all the n features together. The problem of feature selection/ranking thus reduces to finding a set of w_is for which E becomes minimum; w_is indicate the relative importance of x_is in characterizing/discriminating classes. The task of minimization may be performed with various techniques [36]. Here, we have adopted gradient descent technique in a connectionist framework (because of its massive parallelism, fault tolerance etc.) for minimizing E. A new connectionist model is developed for this purpose. This is described in the next section.

3.3 Neural Network Model for Supervised Feature Selection

The network (Fig. 3.1) consists of two layers, namely, input and output. The input layer represents the set of all n features and the output layer corresponds to the pattern classes. Input nodes accept activations corresponding to the feature values of the input patterns. The output nodes produce the membership values of the input patterns corresponding to the respective pattern classes. With each output node, an auxiliary node is connected which controls the activation of the output node through modulatory links. An output node can be activated from the input layer only when the corresponding auxiliary node remains active. Input nodes are connected to the auxiliary nodes through feedback links. The weight of the feedback link from the auxiliary node, connected to the kth output node (corresponding to the class C_k), to

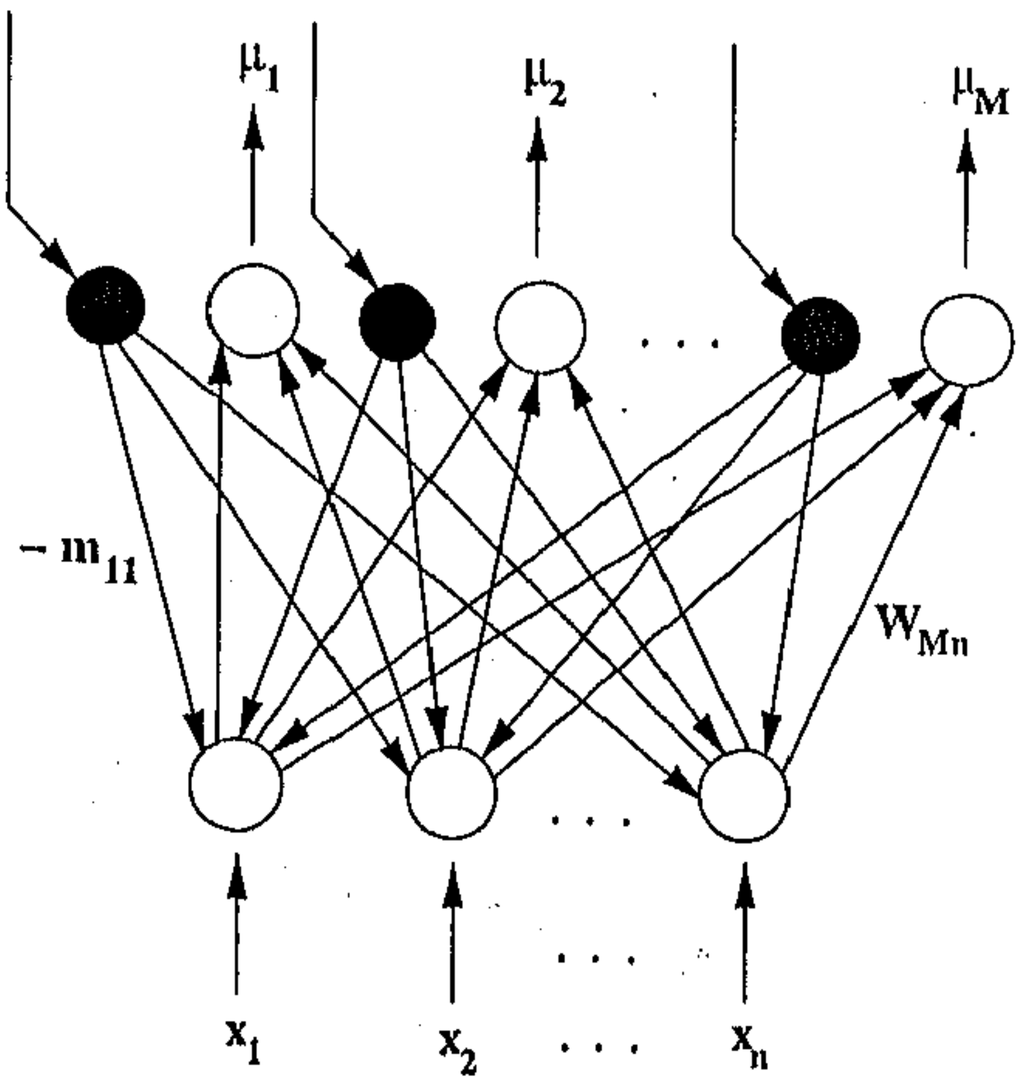

Figure 3.1: A schematic diagram of the neural network model for supervised feature selection. Black circles represent the auxiliary nodes, and white circles represent input and output nodes. Small triangles attached to the output nodes represent the modulatory connections from the respective auxiliary nodes.

the ith input node (corresponding to the feature x_i) is equated to $-m_{ki}$. The weight of the feedforward link from the ith input node to the kth output node provides the degree of importance of the feature x_i, and is given by,

$$W_{ki} = (\frac{w_i}{\lambda_{ki}})^{r_k}. \tag{3.9}$$

During training, the patterns are presented at the input layer and the membership values are computed at the output layer. The feature evaluation index for these membership values is computed (Eqn. (3.14)) and the values of w_is are updated in order to minimize this index. Note that, λ_{ki}s and m_{ki}s are directly computed from the training set and kept fixed during updating of w_is. The auxiliary nodes are activated (*i.e.* activation values are equated to unity) one at a time while the others are made inactive (*i.e.*, the activation values are fixed at 0). Thus, during training, at a time only one output node is allowed to get activated.

When the kth auxiliary node is activated, input node i has an activation value as,

$$u_{ik} = (I_{ik})^{r_k}, \tag{3.10}$$

where I_{ik} is the total activation received by the ith input node for the pattern $\mathbf{x}$, when the auxiliary node k is active. I_{ik} is given by,

$$I_{ik} = x_i - m_{ki}. \tag{3.11}$$

x_i is the external input (value of the ith feature for the pattern $\mathbf{x}$) and $-m_{ki}$ is the feedback activation from the kth auxiliary node to the ith input node. The activation value of the kth output node is given by,

$$v_k = g(y_k), \tag{3.12}$$

where $g(.)$, the activation function of each output node, is a π-function as given in Eqn. (3.4). y_k, the total activation received by the kth output node for the pattern $\mathbf{x}$, is given by,

$$y_k = \left(\sum_i u_{ik} \times (\frac{w_i}{\lambda_{ki}})^2 \right)^{\frac{1}{r_k}}. \tag{3.13}$$

Note that, y_k is the same as d_k (Eqn. (3.8)) for the given input pattern $\mathbf{x}$, and v_k is equal to the membership value of the input pattern $\mathbf{x}$ in the class C_k.

The expression for $E(\mathbf{w})$ (from Eqn. (3.1)), in terms of the output node activations, is given by,

$$E(\mathbf{w}) = \sum_k \sum_{\mathbf{x} \in C_k} \frac{v_k(1-v_k)}{\sum_{k' \neq k} \frac{1}{2} \left[v_k(1-v_{k'}) + v_{k'}(1-v_k) \right]} \times \alpha_k. \tag{3.14}$$

The training phase of the network takes care of the task of minimization of $E(\mathbf{w})$ (Eqn. (3.14)) with respect to $\mathbf{w}$ which is performed using simple gradient-descent technique. The change in w_i (Δw_i) is computed as,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i, \tag{3.15}$$

where η is the learning rate.

For the computation of $\frac{\partial E}{\partial w_i}$, the following expressions are used.

$$\frac{\partial H_{kk'}(\mathbf{x})}{\partial w_i} = \frac{1}{2} \left[[1-2v_{k'}] \frac{\partial v_k}{\partial w_i} + [1-2v_k] \frac{\partial v_{k'}}{\partial w_i} \right], \tag{3.16}$$

$$\frac{\partial H_k(\mathbf{x})}{\partial w_i} = [1 - 2v_k]\frac{\partial v_k}{\partial w_i}, \tag{3.17}$$

$$\begin{aligned} \frac{\partial v_k}{\partial w_i} &= -4d_k(\mathbf{x})\frac{\partial d_k(\mathbf{x})}{\partial w_i}, && 0 \leq d_k(\mathbf{x}) < \tfrac{1}{2} \\ &= -4[1 - d_k(\mathbf{x})]\frac{\partial d_k(\mathbf{x})}{\partial w_i}, && \tfrac{1}{2} \leq d_k(\mathbf{x}) < 1 \\ &= 0, && otherwise \end{aligned} \tag{3.18}$$

and

$$\frac{\partial d_k(\mathbf{x})}{\partial w_i} = \left(\frac{w_i}{d_k(\mathbf{x})}\right)^{r_k - 1}\left(\frac{x_i - m_{ki}}{\lambda_{ki}}\right)^{r_k}. \tag{3.19}$$

Alternately, we can also express E as a function of W_{ki}, where $W_{ki} = (\frac{w_i}{\lambda_{ki}})^{r_k}$, and then minimize E with respect to W_{ki}. In this case, during training phase, the values of W_{ki}s can be updated using the same gradient-descent technique. After training, the degree of importance of ith feature can be computed as $w_i = W_{ki}^{\frac{1}{r_k}} \times \lambda_{ki}$.

Algorithm for learning w

- Calculate the mean vectors ($\mathbf{m}_k$) of all the classes from the data set. Set the weight of the feedback link from the auxiliary node corresponding to the class C_k to the input node i as $-m_{ki}$ (for all i and k).

- Get the values of r_ks (in Eqn. (3.8)) and λ_{ki}s (in Eqn. (3.6)) from the data set, and initialize the weight of the feedforward link from ith input node to kth output (for all values of i and k) node.

- For each input pattern :

  - Present the pattern vector to the input layer of the network.
  - Activate only one auxiliary node at a time.

    Whenever an auxiliary node is activated, it sends the feedback to the input layer. The input nodes, in turn, send the resultant activations to the output nodes. The activation of the output node (connected to the active auxiliary node) provides the membership value of the input pattern to the corresponding class. Thus, the membership values of the input pattern corresponding to all the classes are computed by sequentially activating the auxiliary nodes one at a time.
  - Compute the desired change in weights of the feedforward links to be made using the updating rule given in Eqn. (3.15).

- Compute total change in w_i for each i, over the entire set of patterns. Update w_i (for all i) with the average value of Δw_i.

- Repeat the whole process until convergence, *i.e.*, the change in E becomes less than certain predefined small quantity.

After convergence, $E(\mathbf{w})$ attains a local minimum. In that case, the weights of the feedforward links indicate the order of importance of the features. Note that, the method considers the effect of inter-dependencies of the features unlike those in Chapter 2. In the following section, the convergence of E is theoretically established, and the validity of the ordering of features in terms of network parameters is demonstrated for some well defined class structures.

3.4 Theoretical Analysis

Here, we analyze mathematically the characteristics of the feature evaluation index (E) and the significance of weighting coefficients (w_i). For this purpose, we proceed as follows.

- A fixed upper bound and a varying lower bound of E (Eqn. (3.1)) are derived. The variation of E with respect to the lower bound is studied.

- A relation between E, w_i and interclass distance is derived.

3.4.1 Upper bound and lower bound of E

We can write E (Eqn. (3.1)) as,

$$E = \sum_k \sum_{\mathbf{x} \in C_k} \frac{\mu_k \times (1 - \mu_k)\alpha_k}{\frac{1}{2} \sum_{k' \neq k} [\mu_k \times (1 - \mu_{k'}) + \mu_{k'} \times (1 - \mu_k)]} \tag{3.20}$$

where $\mu_k = \mu_{C_k}(\mathbf{x})$ and $\mu_{k'} = \mu_{C_{k'}}(\mathbf{x})$. Let, $E = \sum_k E_k = \sum_k \sum_{\mathbf{x} \in C_k} E_k(\mathbf{x}|\mathbf{x} \in C_k)$ where

$$E_k = \sum_{\mathbf{x} \in C_k} \frac{\mu_k \times (1 - \mu_k)\alpha_k}{\frac{1}{2} \sum_{k' \neq k} [\mu_k \times (1 - \mu_{k'}) + \mu_{k'} \times (1 - \mu_k)]} \tag{3.21}$$

and

$$E_k(\mathbf{x}|\mathbf{x} \in C_k) = \frac{\mu_k \times (1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k' \neq k}[\mu_k \times (1-\mu_{k'}) + \mu_{k'} \times (1-\mu_k)]}. \tag{3.22}$$

That is, E_k is the value of the evaluation index corresponding to a class C_k, and $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ is contribution of a pattern $\mathbf{x}$ in class C_k, to E_k.

For a pattern $\mathbf{x}$ in class C_k,

$$\frac{1}{2}\sum_{k' \neq k}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)] = \frac{1}{2}\sum_{k' \neq k}\left[\mu_k(1-\mu_k) + (\mu_k - \mu_{k'})^2 + \mu_{k'}(1-\mu_{k'})\right].$$

Since $[(\mu_k - \mu_{k'})^2 + \mu_{k'}(1-\mu_{k'})] \geq 0$,

$$\frac{1}{2}\sum_{k' \neq k}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)] \geq \frac{M-1}{2}\mu_k(1-\mu_k),$$

where M is the number of classes. Since, $0 < \alpha_k < 1$, we can write,

$$E_k(\mathbf{x}|\mathbf{x} \in C_k) \leq \frac{2}{(M-1)}. \tag{3.23}$$

Therefore,

$$\mathcal{E}(E) \leq \frac{2M}{M-1}, \tag{3.24}$$

where $\mathcal{E}$ denotes the 'mathematical expectation' operator.

♣

Again, for a pattern $\mathbf{x}$ in class C_k, $\mu_k, \mu_{k'} \in [0,1]$, we can write,

$$\frac{1}{2}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)] \leq \frac{1}{2},$$

$$\sum_{k' \neq k}\frac{1}{2}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)] \leq \frac{1}{2}(M-1),$$

$$\frac{1}{\sum_{k' \neq k}\frac{1}{2}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)]} \geq \frac{2}{(M-1)},$$

$$\sum_k \frac{\mu_k(1-\mu_k)\alpha_k}{\sum_{k' \neq k}\frac{1}{2}[\mu_k(1-\mu_{k'}) + \mu_{k'}(1-\mu_k)]} \geq \frac{2}{(M-1)}\sum_k \mu_k(1-\mu_k)\alpha_k.$$

Thus,

$$E_k(\mathbf{x}|\mathbf{x} \in C_k) \geq \frac{2}{(M-1)}\mu_k(1-\mu_k)\alpha_k.$$

That is,

$$\mathcal{E}(E) \geq \frac{2}{(M-1)}\mathcal{E}(\sum_k \mu_k(1-\mu_k)\alpha_k). \tag{3.25}$$

Therefore,

$$\tfrac{2}{(M-1)}\mathcal{E}(\sum_k \mu_k(1-\mu_k)\alpha_k) \;\leq\; \mathcal{E}(E) \;\leq\; \tfrac{2M}{(M-1)}. \tag{3.26}$$

Note that, the upper bound of $\mathcal{E}(E)$ is fixed, whereas the lower bound is varying with $\frac{2}{(M-1)}\mathcal{E}(\sum_k \mu_k(1-\mu_k)\alpha_k)$.

♣

Let us now analyze the behavior of $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ with respect to $\mu_k(1-\mu_k)$. For this purpose, we substitute $\mu_k(1-\mu_k)$ by h_k in Eqn. (3.22). In that case,

$$\begin{aligned}
\frac{dE_k(\mathbf{x}|\mathbf{x}\in C_k)}{dh_k} &= \frac{\alpha_k\left[\sum_{k'\neq k}[\mu_{k'}(1-\mu_k)+\mu_k(1-\mu_{k'})](1-2\mu_k) - \mu_k(1-\mu_k)\sum_{k'\neq k}(1-2\mu_{k'})\right]}{\frac{1}{2}\left[\sum_{k'\neq k}[\mu_{k'}(1-\mu_k)+\mu_k(1-\mu_{k'})]\right]^2 (1-2\mu_k)} \\
&= \nu_k\alpha_k/\tfrac{1}{2}\left[\sum_{k'\neq k}[\mu_{k'}(1-\mu_k)+\mu_k(1-\mu_{k'})]\right]^2,
\end{aligned} \tag{3.27}$$

where

$$\nu_k = \frac{\sum_{k'\neq k}[\mu_{k'}(1-\mu_k)+\mu_k(1-\mu_{k'})](1-2\mu_k) - \mu_k(1-\mu_k)\sum_{k'\neq k}(1-2\mu_{k'})}{(1-2\mu_k)}. \tag{3.28}$$

It is clear from Eqn. (3.27) that $\frac{dE_k(\mathbf{x}|\mathbf{x}\in C_k)}{dh_k}$ is positive/negative if ν_k is positive/negative. In other words, $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ increases/decreases monotonically with $\mu_k(1-\mu_k)$ if ν_k is positive/negative. Simplifying the expression on the right hand side of Eqn. (3.28) we get,

$$\nu_k = \sum_{k'\neq k}\mu_{k'} - \frac{\mu_k^2\sum_{k'\neq k}(1-2\mu_{k'})}{(1-2\mu_k)}. \tag{3.29}$$

In order to show that $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ monotonically increases with $\mu_k(1-\mu_k)$ for both *non-overlapping* and *overlapping* class structures, we consider the following cases.

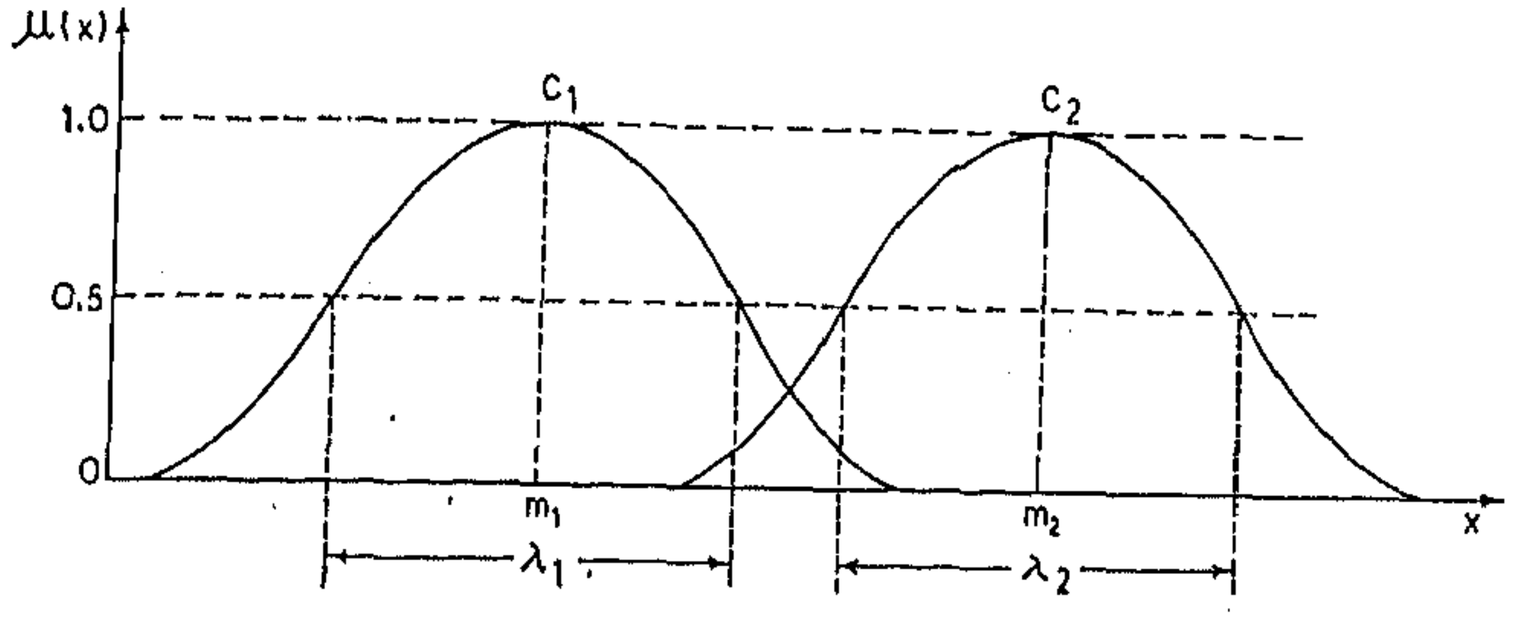

Figure 3.2: Non-overlapping pattern classes modeled with π-function.

Case 1 : *Non-overlapping* (Fig. 3.2)

Here, for a pattern $\mathbf{x}$, if $|x_i - m_{ki}| \leq \frac{\lambda_{ki}}{2}$ holds for all values of i, $\mu_k \geq 0.5$ and $\mu_{k'} < 0.5$, $\forall k' \neq k$. Therefore, $\nu_k > 0$ (Eqn. (3.29)), and as a result $\frac{dE_k(\mathbf{x}|\mathbf{x}\in C_k)}{dh_k} > 0$. This indicates $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ is monotonically increasing with $\mu_k(1-\mu_k)$.

Case 2 : *Overlapping* (Fig. 3.3)

In this case, for a pattern $\mathbf{x}$, if $|x_i - m_{ki}| \leq \frac{\lambda_{ki}}{2}$ holds for all values of i, $\mu_k \geq 0.5$ and $\mu_{k'} \gtrsim 0.5$, $\forall k' \neq k$. Since the classes are overlapped, we consider two different possibilities: $\mathbf{x}$ lying outside the overlapping zone (*i.e.*, $|x_i - m_{ki}| \leq \frac{\lambda_{ki}}{2}$, $\forall i$ and $|x_i - m_{k'i}| > \frac{\lambda_{k'i}}{2}$) and $\mathbf{x}$ lying within the overlapping zone (*i.e.*, $|x_i - m_{ki}| < \frac{\lambda_{ki}}{2}$, $\forall i$ and $|x_i - m_{k'i}| < \frac{\lambda_{k'i}}{2}$, $\forall i$).

If the pattern $\mathbf{x}$ lies outside the overlapping zone, then $\mu_{k'} < 0.5$ and thereby $\nu_k > 0$ (Eqn. (3.29)). This indicates $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ monotonically increases with $\mu_k(1-\mu_k)$.

If $\mathbf{x}$ lies within the overlapping zone, both $\mu_k, \mu_{k'} > 0.5$. Then we have three possibilities: (a) $\mu_k > \mu_{k'}$, (b) $\mu_k \approx \mu_{k'}$ and (c) $\mu_k < \mu_{k'}$.

(a) $\mu_k > \mu_{k'}$

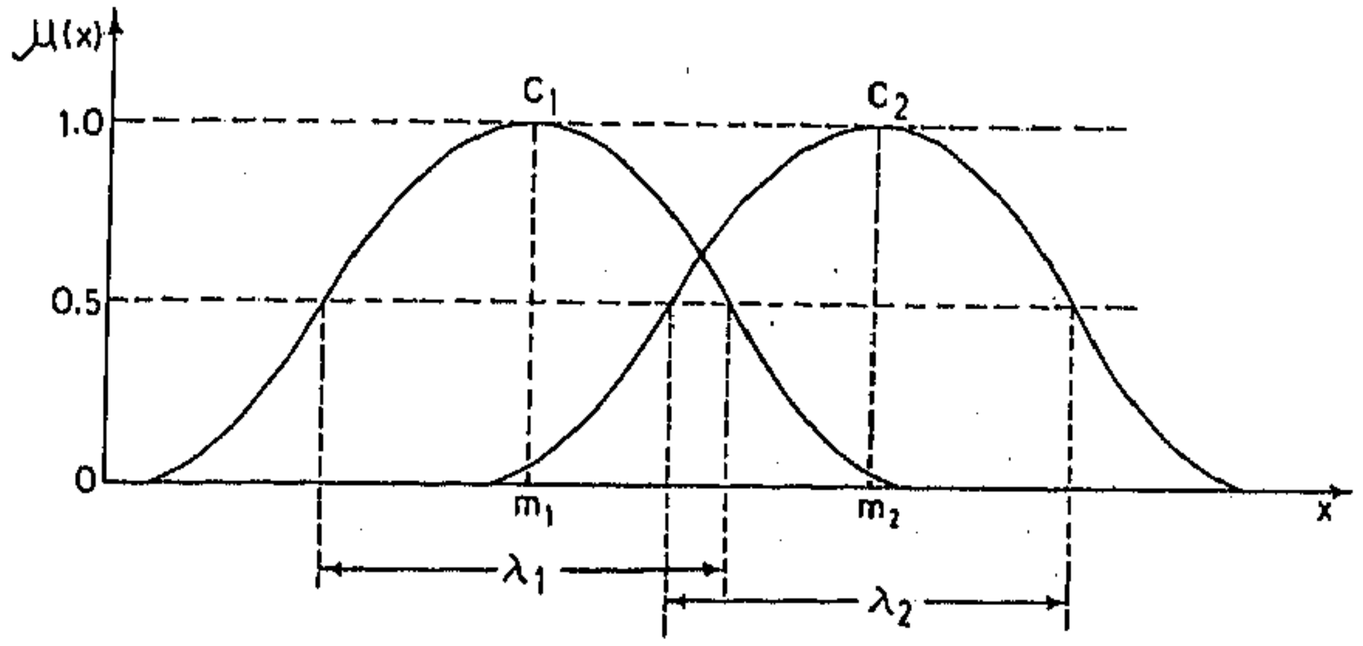

Figure 3.3: Overlapping pattern classes modeled with π-function.

Let $\mu_{k'} = \mu_k - \epsilon_{kk'}$, where $\epsilon_{kk'} > 0$. Therefore, from Eqn. (3.29) we get,

$$\nu_k = \sum_{k' \neq k} (\mu_k - \epsilon_{kk'}) - \frac{\mu_k^2 \sum_{k' \neq k} (1 - 2\mu_k + 2\epsilon_{kk'})}{(1 - 2\mu_k)}, \tag{3.30}$$

i.e.,

$$\nu_k = (M-1)\mu_k - \sum_{k' \neq k} \epsilon_{kk'} - \frac{2\mu_k^2 \sum_{k' \neq k} \epsilon_{kk'} - \mu_k^2(2\mu_k - 1)(M-1)}{1 - 2\mu_k}. \tag{3.31}$$

Thus, $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ increases monotonically with $\mu_k(1-\mu_k)$ if

$$(M-1)\mu_k - \sum_{k' \neq k} \epsilon_{kk'} - \frac{2\mu_k^2 \sum \epsilon_{kk'} - \mu_k^2(2\mu_k - 1)(M-1)}{1 - 2\mu_k} > 0. \tag{3.32}$$

i.e., if

$$\frac{1}{M-1} \sum_{k' \neq k} \epsilon_{kk'} > -\frac{\mu_k(1-\mu_k)(2\mu_k - 1)}{(1-\mu_k)^2 + \mu_k^2}. \tag{3.33}$$

Since, $\epsilon_{kk'} > 0$, the above inequality always holds, and therefore, in such cases, $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ always increases monotonically with $\mu_k(1-\mu_k)$.

(b) $\mu_k \approx \mu_{k'}$

In this case, $\epsilon_{kk'} \approx 0$, and therefore, the inequality (3.33) always holds. Thus, in this case also, we get a monotonic increasing nature of $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ with respect to $\mu_k(1-\mu_k)$.

(c) $\mu_k < \mu_{k'}$

In this case, $\epsilon_{kk'} < 0$. Let us replace $\epsilon_{kk'}$ by $-\varepsilon_{kk'}$, *i.e.*, $\mu_{k'} = \mu_k + \varepsilon_{kk'}$. Then, the condition for $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ being monotonically increasing function with respect to $\mu_k(1-\mu_k)$ becomes,

$$\frac{1}{M-1}\sum_{k' \neq k} \epsilon_{kk'} < \frac{\mu_k(1-\mu_k)(2\mu_k-1)}{(1-\mu_k)^2+\mu_k^2}. \tag{3.34}$$

This condition provides an upper bound on the average value of $\varepsilon_{kk'}$ (hence on the average value of $\mu_{k'}$) that can be allowed in order to get a monotonic increasing behavior of $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ with respect to $\mu_k(1-\mu_k)$.

First of all, the chance of $\mu_k < \mu_{k'}$ is low for a pattern in class C_k. Even if this happens (say, for overlapping case), the chance of happening $\frac{1}{M-1}\sum_{k' \neq k} \epsilon_{kk'} > \frac{\mu_k(1-\mu_k)(2\mu_k-1)}{(1-\mu_k)^2+\mu_k^2}$ is very low (as illustrated in the following two examples). Therefore, $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ is most likely monotonically increasing with $\mu_k(1-\mu_k)$.

Example 1

Let, $\mu_1 = 0.6$ for a pattern $\mathbf{x}$ lying within the region $\|\mathbf{x}-\mathbf{m}_1\| < \frac{\lambda_1}{2}$ in class C_1. Then, the condition (3.34) becomes,

$$\frac{1}{M-1}\sum_{k' \neq k} \epsilon_{kk'} < 0.1.$$

In order to violet this condition, the average membership value of $\mathbf{x}$ (say, μ_2) to classes other than C_1 should be at least 0.7. It can also be seen that whatever be the value of μ_1 (> 0.5), the value of μ_2 should be greater than μ_1. This is unusual. Thus, we can say that in this case the above inequality (3.34) will be satisfied and thereby, we can expect a monotonic increasing behavior of $E_1(\mathbf{x}|\mathbf{x} \in C_1)$ with respect to $\mu_1(1-\mu_1)$.

Example 2

Let, $\mu_1 = 0.5$. In that case, the condition (3.34) becomes

$$\frac{1}{M-1}\sum_{k' \neq k} \varepsilon_{kk'} < 0.$$

That is, the average membership value of $\mathbf{x}$ to classes other than C_1 should be greater than or equal to 0.5. This situation occurs when the classes are highly overlapped. In other words, if there is high amount of overlap, the behavior of $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ becomes unpredictable for ambiguous patterns. ♣

Thus, we can say that almost in all the cases, $E_k(\mathbf{x}|\mathbf{x} \in C_k)$ is monotonically increasing with $\mu_k(1-\mu_k)$. Therefore, we can expect that E_k $(= \sum_{\mathbf{x} \in C_k} E_k(\mathbf{x}|\mathbf{x} \in C_k))$ increases monotonically with $\sum_k \mu_k(1-\mu_k)$. In other words, almost in all the cases $\mathcal{E}(E)$ is a monotonically increasing function of $\mathcal{E}(\sum_k (\mu_k(1-\mu_k)\alpha_k)$, as α_ks are positive constants.

3.4.2 Relation between E, interclass distance and w_i

Let us now derive a relation of the lower bound of $\mathcal{E}(E)$ with interclass distance and weighting coefficients for some well defined class structures.

- Let us assume that the classes $C_1, C_2, \ldots, C_k, \ldots, C_M$ have independent, identical Gaussian distributions with respective means $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k, \ldots, \mathbf{m}_M$ and with the same variance σ^2. Let $\wp(\mathbf{x}|C_k)$ be the class-conditional probability density function for class C_k. Then

$$\wp(\mathbf{x}|C_k) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2}\right)} \tag{3.35}$$

- Let the membership of a pattern $\mathbf{x}$ in a class C_k be given by,

$$\mu_k = \mu_k(\mathbf{x}) = e^{\left(-\sum_i \frac{(x_i - m_{ki})^2 w_i^2}{2\lambda^2}\right)} \tag{3.36}$$

where λ is the bandwidth of the class C_k, and is the same for all the classes.

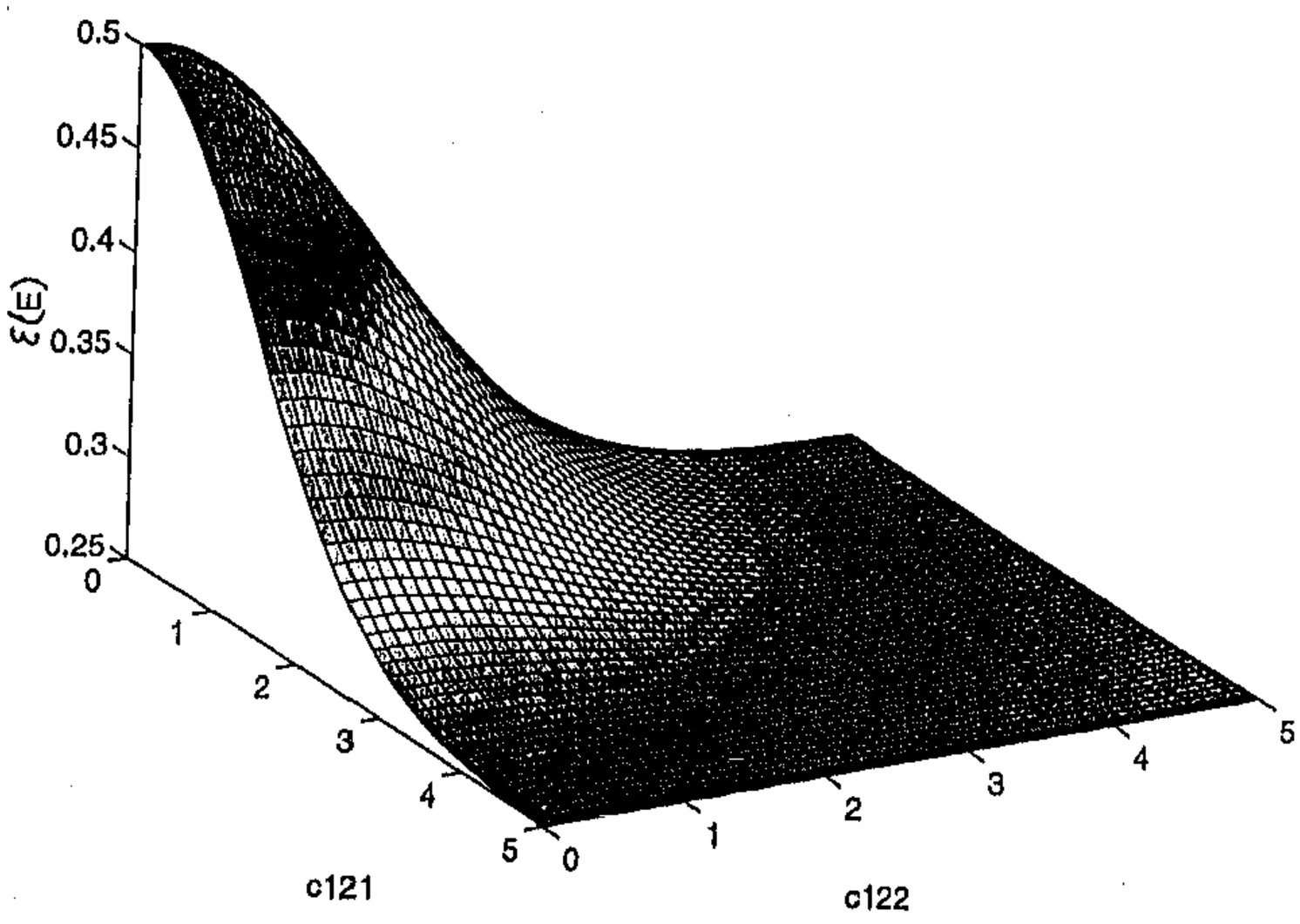

Figure 3.4: Graphical representation of $\mathcal{E}(E)$ with respect to c_{121} and c_{122} with $w_1 = w_2 = 1.0$.

$\mathcal{E}(E)$ is given by,

$$\mathcal{E}(E) = \int_{\mathbf{x}} E\wp(\mathbf{x})d\mathbf{x}, \tag{3.37}$$

where

$$\wp(\mathbf{x}) = \sum_k P_k \wp(\mathbf{x}|C_k); \tag{3.38}$$

P_k being *a priori* probability of class C_k. Evaluating the right hand side of Eqn. (3.37) (see Appendix H), we have

$$\mathcal{E}(E) \approx \sum_k \frac{\alpha_k P_k}{M-1} \frac{\sum_i w_i^2}{2\rho^2} \left(1 + \sum_{k' \neq k} e^{-\sum_i \frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}} \right). \tag{3.39}$$

where $\rho = \frac{\lambda}{\sigma}$ and $c_{kk'i} = m_{ki} - m_{k'i}$ is a measure of interclass distance between the classes C_k and $C_{k'}$ along the feature axis x_i.

Let us consider two classes C_1 and C_2, with two features x_1 and x_2. Let, C_1 and C_2 have unit normal distribution, *i.e.*, $\sigma = 1.0$. Let, $\lambda = 1.0$ and $P_k = \alpha_k = 0.5$ $(\forall k)$. c_{121} and c_{122} are the interclass distances between class C_1 and class C_2 along the

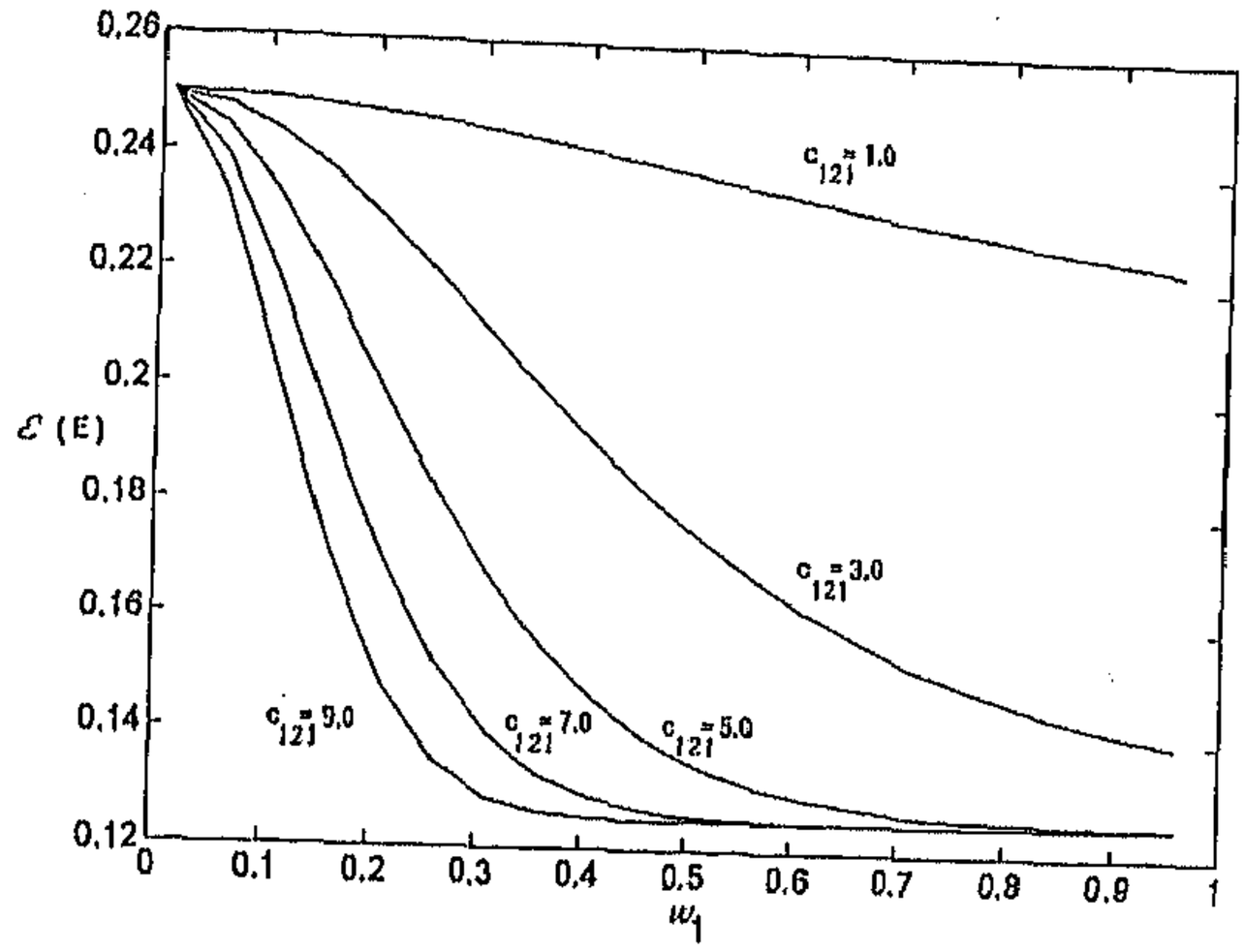

Figure 3.5: Graphical representation of $\mathcal{E}(E)$ with respect to w_1 for different values of c_{121}, with $c_{122} = 0$ and $\sum_{i=1}^{2} w_i^2 = 1$.

feature axes x_1 and x_2 respectively. We now demonstrate graphically the variation of $\mathcal{E}(E)$ with respect to c_{121} and c_{122}, and w_1 and w_2.

Fig. 3.4 shows the variation of $\mathcal{E}(E)$ with respect to c_{121} and c_{122} with $w_1 = w_2 = 1$. $\mathcal{E}(E)$ is maximum when $c_{121} = c_{122} = 0$, *i.e*, when the two classes completely overlap. $\mathcal{E}(E)$ decreases with the increase in c_{121} and c_{122}. This variation is symmetric with respect to both c_{121} and c_{122}. The rate of decrease in $\mathcal{E}(E)$ also decreases as c_{121} (and c_{122}) increases. Finally, after a certain value of c_{121} (and c_{122}) the rate of decrease in $\mathcal{E}(\sum_k \mu_k(1-\mu_k)\alpha_k)$ becomes infinitesimally small. This is also evident from the way of computing μ-value where μ_2 of a pattern x with fixed μ_1 decreases with increase in interclass distance. If the interclass distance exceeds a certain value, μ_2 becomes very small. Thus, the contribution of the pattern to the evaluation index does not get affected further by the extent of the class separation.

Figs. 3.5–3.6 show the variation of $\mathcal{E}(E)$ with respect to w_1 and w_2 for different interclass distances when $\sum_{i=1}^{2} w_i^2 = 1$. Here we have considered $c_{122} = 0$ throughout,

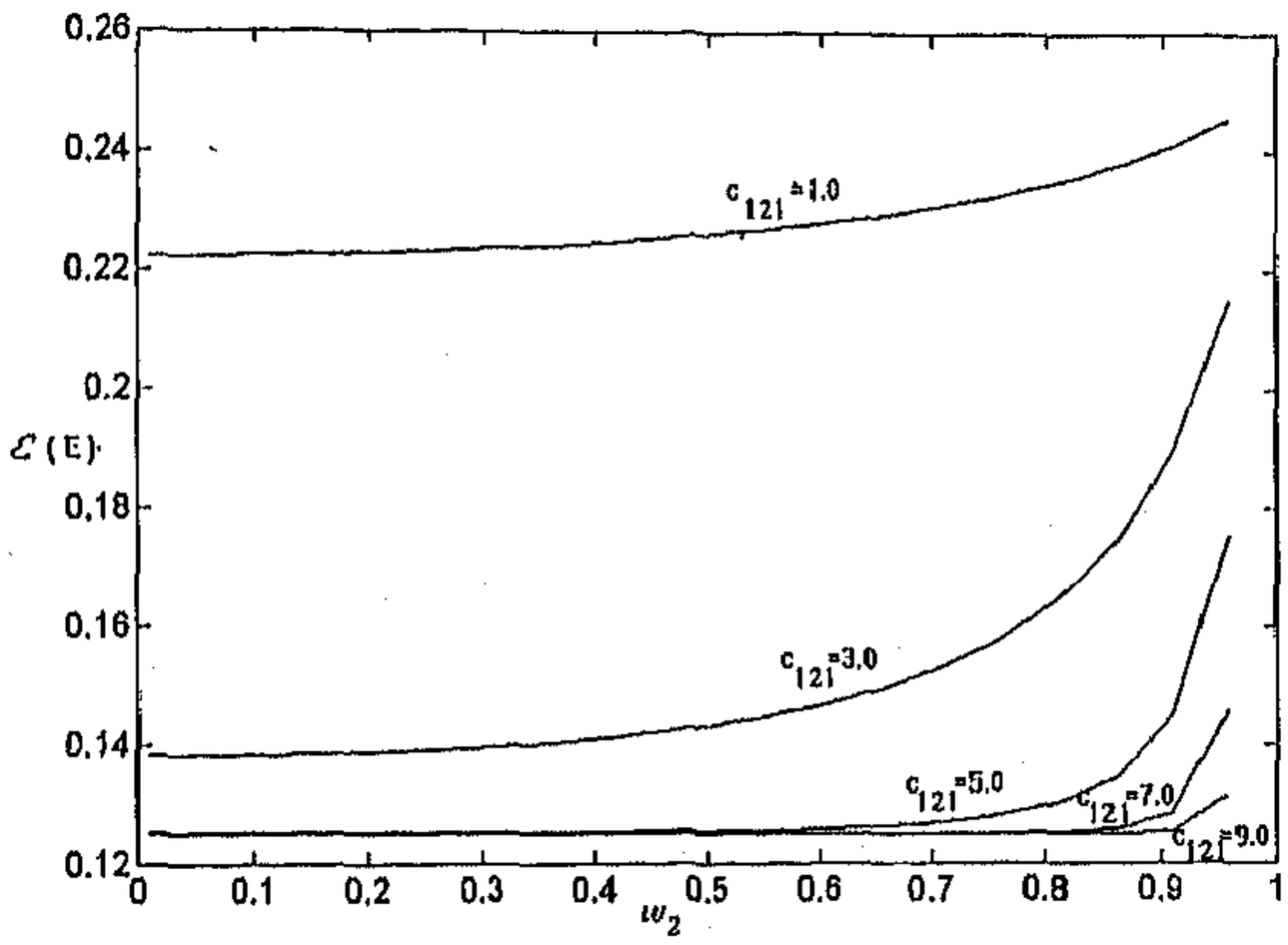

Figure 3.6: Graphical representation of $\mathcal{E}(E)$ with respect to w_2 for different values of c_{121}, with $c_{122} = 0$ and $\sum_{i=1}^{2} w_i^2 = 1$.

whereas c_{121} is considered to be 1.0, 3.0, 5.0, 7.0 and 9.0 respectively. It is seen from the figures that E decreases with w_1 (or increases with w_2) and attains a maximum (or minimum) when $w_1 = 0$ (or when $w_2 = 0$). This is due to the fact that the feature x_2 has no discriminating power as $c_{122} = 0$. On the other hand, the feature x_1 is necessary for classification as there is a separation ($c_{121} \neq 0$) between the classes along its axis. Note also from Figs. 3.5–3.6 that for higher values of c_{121}, the decrease (or increase) of E is more sharp. This indicates that the rate of convergence of the network to a local minimum, as expected, increases with the decrease in overlap between the classes.

3.5 Results

Here we demonstrate the effectiveness of the neuro-fuzzy approach on four data sets – vowel, Iris, medical and mango-leaf (described in Chapter 2). Besides this, we also provide results on a method of feature selection based on E (Eqn. (3.1)) *alone*. First of all, we describe the results of feature selection using E in Section 3.5.1. Then the results of neuro-fuzzy method are provided in Section 3.5.2.

In the following experiments the values of r_k in Eqns. (3.5) and (3.8) are so chosen that the membership values of all the patterns of a class are at least 0.5 for that class. For 6-class vowel data the values of r_k are found to be 28.8, 78.5, 21.4, 74.0, 20.4 and 47.8 corresponding to its different classes. Similarly, these values are 71.7, 241.3 and 193.9 for 3-class Iris data; 65.0, 38.5, 12.8 and 163.2 for 4-class medical data; and 133.8, 71.2 and 225.2 for 3-class mango-leaf data.

3.5.1 Using feature evaluation indices

From Section 3.2 we see that if a particular subset (Ω_1) of features is more important than another subset (Ω_2) then E computed over Ω_1 will be less than that computed over Ω_2. Therefore, the task of feature subset selection boils down to selecting the subset Ω from a given set of n features for which E is minimum. This is done by computing the E values for all possible $(2^n - 1)$ subsets of features using Eqns. (3.1)–(3.7), and ranking them accordingly. The order of importance of these subsets is

compared with that obtained by the feature evaluation index ($FEI^{(av)}$) of Pal [152].

Table 3.1: Importance of different feature subsets. $X > Y$ means X is more important than Y. Since the number of subsets for medical and mango-leaf data is large, only first fifteen are shown.

Data sets	Order of importance using	
	E (Eqn. (3.1))	$FEI^{(av)}$ of Pal [152]
Vowel	$\{F_2\} > \{F_1\} > \{F_1, F_2\} >$ $\{F_2, F_3\} > \{F_1, F_2, F_3\} >$ $\{F_1, F_3\} > \{F_3\}$	$\{F_1, F_2\} > \{F_2\} > \{F_1\} >$ $\{F_2, F_3\} > \{F_1, F_2, F_3\} >$ $\{F_3\} > \{F_1, F_1, F_3\}$
Iris	$\{PW\} > \{PL\} > \{PL, PW\} >$ $\{SW, PW\} > \{SW, PL\} > \{SL, PL\} >$ $\{SL, PW\} > \{SW, PL, PW\} > \{SL\} >$ $\{SL, SW, PW\} > \{SL, SW, PL\} > \{SL, PL, PW\}$ $\{SL, SW, PL, PW\} > \{SL, SW\} > \{SW\}$	$\{PL\} > \{SW, PL\} > \{PL, PW\} >$ $\{PW\} > \{SW, PL, PW\} > \{SL, SW, PL, PW\} >$ $\{SW, PW\} > \{SL, PL\} > \{SL, PL, PW\} >$ $\{SL, SW, PL\} > \{SL, SW, PW\} > \{SL, PW\} >$ $\{SW\} > \{SL\} > \{SL, SW\}$
Medical	$\{MCV\} > \{LDH, MCV\} > \{MCH\} >$ $\{MCV, MCH\} > \{MCV, TBil\} > \{LDH, MCV, TBil\} >$ $\{LDH, MCV, MCH\} > \{LDH, MCH\} > \{BUN, MCV\} >$ $\{LDH\} > \{MCH, TBil\} > \{LDH, BUN, MCV\} >$ $\{BUN, MCV, MCH\} > \{BUN, MCV, TBil\} > \{LDH, BUN, MCV, MCH\} > \ldots$	$\{MCV, MCH, TBil\} > \{TBil\} > \{MCV, TBil\} >$ $\{MCH\} > \{BUN, MCV, MCH\} > \{BUN, MCV\} >$ $\{MCH, TBil\} > \{BUN, MCV, TBil\} > \{BUN, MCV, MCH, TBil\} >$ $\{BUN, MCH\} > \{MCV, MCH\} > \{BUN, TBil\} >$ $\{BUN\} > \{BUN, MCH, TBil\} > \{MCV\} > \ldots$
Mango-leaf	$\{L/B\} > \{L/B, UPe/LPe\} > \{SI, L/B\} >$ $\{SI\} > \{SI, L/B, UPe/LPe\} > \{SI, UPe/LPe\} >$ $\{SI, L/B, (L+P)/B\} > \{B, L/B\} > \{B, SI, L/B\} >$ $\{SI, (L+P)/B\} > \{B, L/B, (L+P)/B\} > \{B, SI\} >$ $\{L/B, (L+P)/B, UPe/LPe\} > \{(L+P)/B, UPe/LPe\} > \ldots$	$\{B\} > \{L/B\} > \{B, UPe/LPe\} >$ $\{Pe\} > \{(L+P)/B\} > \{A/L\} >$ $\{B, L/B\} > \{B, L/B, UPe/LPe\} > \{P\} >$ $\{A\} > \{L+P\} > \{S\} >$ $\{SI, L/B\} > \{SI, L/B, UPe/LPe\} > \{L/B, (L+P)/B, UPe/LPe\} > \ldots$

Table 3.2: Recognition score with k-NN classifier for individual and pairwise features of Iris data.

Feature Subset	% classification				
	$k=1$	$k=2$	$k=3$	$k=5$	$k=9$
$\{SL\}$	48.67	64.00	66.67	67.33	66.67
$\{SW\}$	55.33	55.33	52.67	52.67	54.67
$\{PL\}$	93.33	89.33	95.33	95.33	95.33
$\{PW\}$	89.33	89.33	96.00	96.00	94.67
$\{SL, SW\}$	74.67	76.67	76.67	76.00	78.00
$\{SL, PL\}$	95.33	92.00	93.33	95.33	96.00
$\{SL, PW\}$	94.67	94.67	94.00	94.00	91.33
$\{SW, PL\}$	94.67	90.67	92.00	93.33	95.33
$\{SW, PW\}$	90.67	92.67	94.00	94.67	94.00
$\{PL, PW\}$	93.33	94.00	96.00	96.00	96.67

Table 3.3: Recognition score with k-NN classifier for individual and pairwise features of vowel data.

Feature Subset	% classification				
	$k=1$	$k=2$	$k=3$	$k=5$	$k=21$
$\{F_1\}$	26.52	18.25	27.21	27.21	31.92
$\{F_2\}$	38.58	36.28	38.23	47.76	60.28
$\{F_3\}$	26.06	26.41	33.41	33.87	26.75
$\{F_1, F_2\}$	56.37	55.68	68.20	76.35	77.73
$\{F_1, F_3\}$	44.32	45.58	46.84	55.80	54.65
$\{F_2, F_3\}$	58.21	56.14	63.03	63.95	65.10

In the case of vowel data, the order of importance of the subsets of features is

$$\{F_2\} > \{F_1\} > \{F_1, F_2\} > \{F_2, F_3\} > \{F_1, F_2, F_3\} > \{F_1, F_3\} > \{F_3\}$$

according to E of Eqn. (3.1), and

$$\{F_1, F_2\} > \{F_2\} > \{F_1\} > \{F_2, F_3\} > \{F_1, F_2, F_3\} > \{F_3\} > \{F_1, F_3\}$$

according to the $FEI^{(av)}$ of Pal [152]. Here $x > y$ indicates that the importance of feature x is greater than that of feature y. For both the methods, three best subsets are found to be the same. Note that the relative ranks of the individual features and feature pairs are in conformity to those obtained by the indices FQI and FDI of Chapter 2. For Iris data (Table 3.1), the subsets $\{PW\}$, $\{PL\}$ and $\{SW, PW\}$ are found to be the first, second and third best subsets by E (Eqn. (3.1)), whereas the corresponding subsets are $\{PL\}$, $\{SW, PL\}$ and $\{PL, PW\}$ by the index of Pal *et al.* [152]. Note that, SL has not come out as a member of these subsets by either method.

In the case of medical data, since the number of features is nine, we have computed the evaluation indices for individual features (*i.e.*, for the nine subsets), and for all the subsets containing elements of the best four individual features obtained by the respective indices. Note that, these four features are found to be MCV, MCH, LDH and $TBil$ by Eqn. (3.1), and $TBil$, MCH, BUN and MCV by $FEI^{(av)}$ of [152]. Therefore we consider five features LDH, BUN, MCV, MCH and $TBil$ to

constitute these subsets. The total number of subsets thus considered including the nine individual features becomes 35. Among all these, the order of importance of the best five subsets, as seen from Table 3.1, is

$$\{MCV\} > \{LDH, MCV\} > \{MCH\} > \{MCV, MCH\} > \{MCV, TBil\}$$

according to E of Eqn. (3.1), and

$$\{MCV, MCH, TBil\} > \{TBil\} > \{MCV, TBil\} > \{MCH\} > \{BUN, MCV, MCH\}$$

according to the $FEI^{(av)}$ of Pal [152]. Note that, the features MCV and/or MCH are present in all these subsets obtained by E (Eqn. (3.1)), whereas it is MCH and/or TBil which are present in all the best five subsets obtained by the index of Pal [152]. This conforms to the ranking order obtained for individual features where MCV and MCH are found to be the best two features using Eqn. (3.1), and TBil and MCH are those as obtained by the algorithm in [152].

Similarly, in the case of mango-leaf data, since the number of features is eighteen, we have computed the evaluation indices for individual features (*i.e.*, for the eighteen subsets), and for all the subsets containing elements of the best four individual features obtained by the respective indices. Here, the best four features obtained by these two indices are found to be the elements of $\{Pe, B, SI, L/B, (L + P)/B, UPe/LPe\}$; thereby making a total of 75 subsets. Among them, the best five subsets as obtained with E (Eqn. (3.1)) and the $FEI^{(av)}$ of Pal [152] are (Table 3.1)

$$\{L/B\} > \{L/B, UPe/LPe\} > \{SI, L/B\} > \{SI\} > \{Pe, L/B\}$$

and

$$\{B\} > \{L/B\} > \{B, UPe/LPe\} > \{Pe\} > \{(L + P)/B\}$$

respectively. Note that the features L/B and/or SI are present in all these five subsets obtained by E (Eqn. (3.1)). This conforms to the ranking order obtained for individual feature where L/B and SI are found to be the best two features using Eqn. (3.1). On the other hand, for $FEI^{(av)}$ [152] the best two individual features, e.g., B and L/B are seen to be present only in the first three subsets.

In order to show the validity of these orders of importance, we consider both scatter plots and k-NN classifier for $k = 1, 2, 3, 5$ and $\sqrt{s}$; s being the number of samples

in the training set. The results are shown only for Iris and vowel data. In the case of Iris data, it is seen from Figs. 2.5–2.10 that the order of importance (in terms of class structures) of the feature pairs conforms to those (Table 3.1) obtained by the evaluation index E (Eqn. (3.1)). Among all the feature pairs, $\{PL, PW\}$ is the best. In other words, the result obtained by $FEI^{(av)}$ of Pal [152] that the subset $\{SW, PL\}$ is more important than $\{PL, PW\}$, does not get reflected by the scatter plots. Although, the order of importance of PW and PL, individually, is found to be different for E and $FEI^{(av)}$, according to Fig. 2.10, they are seen to have more or less the same importance.

From the results of k-NN classifier (Table 3.2), PW is seen to be better than PL for most of the values of k, although the difference is not significant. In fact, the ranking $PW > PL > SL > SW$ as obtained by E for individual features is seen to be exactly reflected in Table 3.2. As in the case of scatter plots, $\{PL, PW\}$ is seen here to be the best of all such pairs. In other words, the order obtained by $FEI^{(av)}$ of Pal [152] that $\{SW, PL\} > \{PL, PW\}$ does not get supported by the k-NN classifier. The subset $\{SW, PW\}$ is also found to be more important (in terms of classification performance) than $\{SW, PL\}$ for all the cases except $k = 9$. These signify the superiority of the measure E over $FEI^{(av)}$ considering the ranking within both individual features and pairwise features. ♣

In the case of overlapping vowel data, it is seen from Figs. 2.11–2.13 that $\{F_1, F_2\}$ is the best feature pair, and this conforms to that obtained by both the indices. The order of importance of the feature pairs, $\{F_1, F_2\} > \{F_2, F_3\} > \{F_1, F_3\}$, as obtained by both the indices, is also in conformity to the results obtained by k-NN classifier. However, unlike E, the relative importance of the best three subsets obtained by $FEI^{(av)}$ is seen to be maintained in the results of k-NN classifier. ♣

Finally, the relation of feature evaluation index, E (Eqn. (3.1)) with Mahalanobis distance and divergence measure is graphically depicted in Figs. 3.7 and 3.8 (for vowel data), in Figs. 3.9 and 3.10 (for Iris data), in Figs. 3.11 and 3.12 (for the medical data) and in Figs. 3.13 and 3.14 (for mango-leaf data). They are computed over every pair of classes. As expected, Figs. 3.7–3.14 show a decrease in feature evaluation index with increase in Mahalanobis distance and divergence measure between the classes.

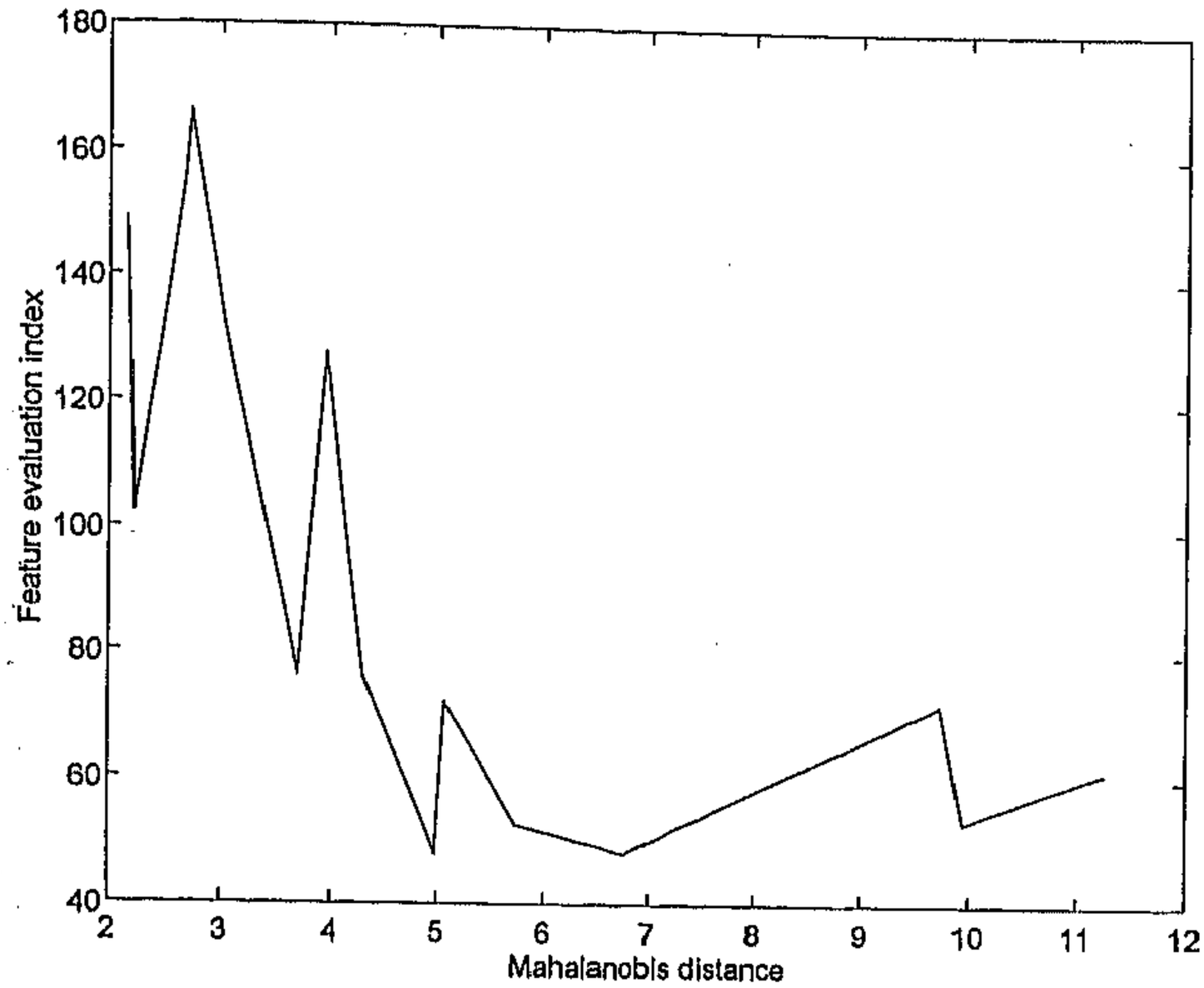

Figure 3.7: Graphical representation of the relationship between feature evaluation index and Mahalanobis distance for the vowel data.

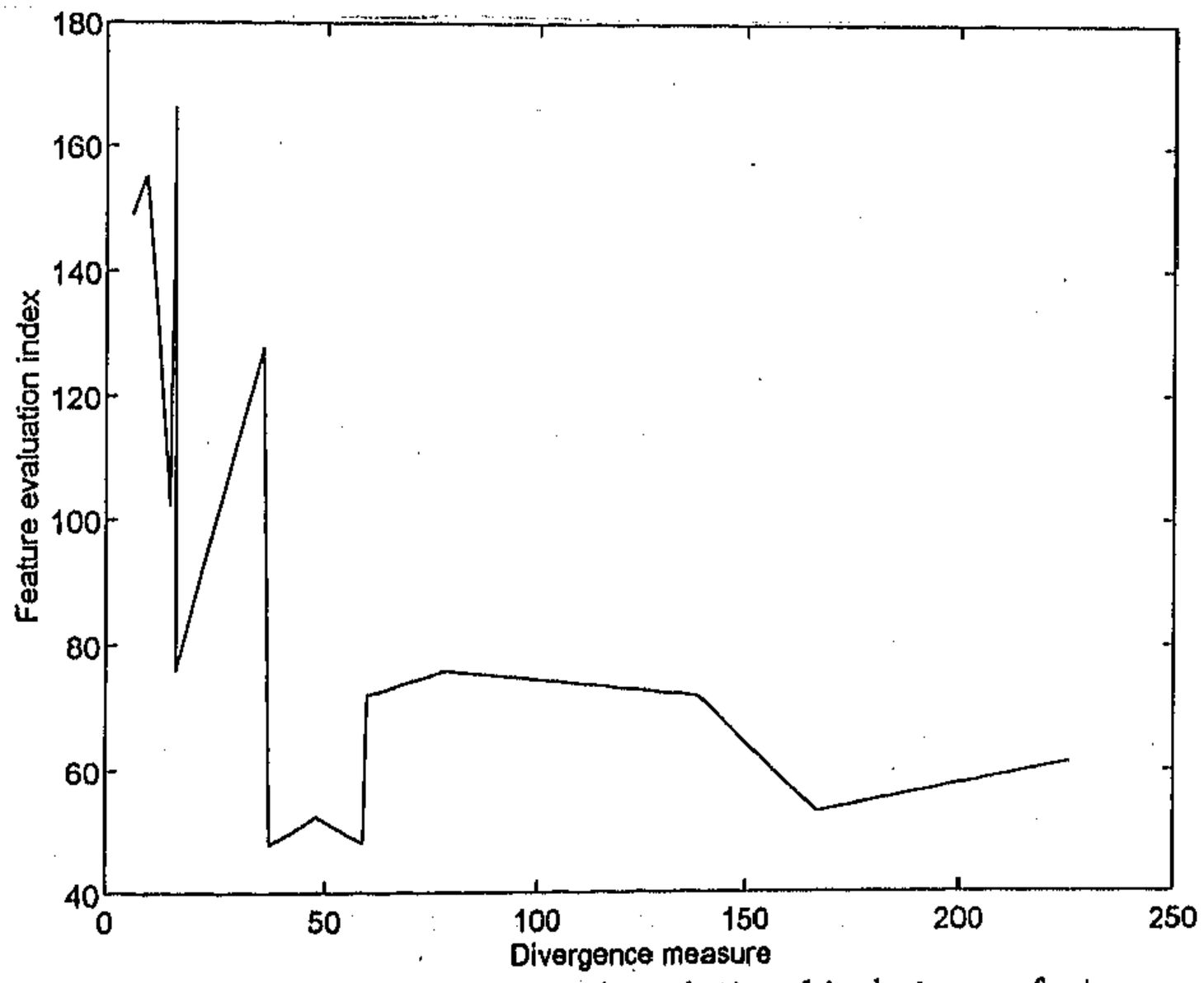

Figure 3.8: Graphical representation of the relationship between feature evaluation index and divergence measure for the vowel data.

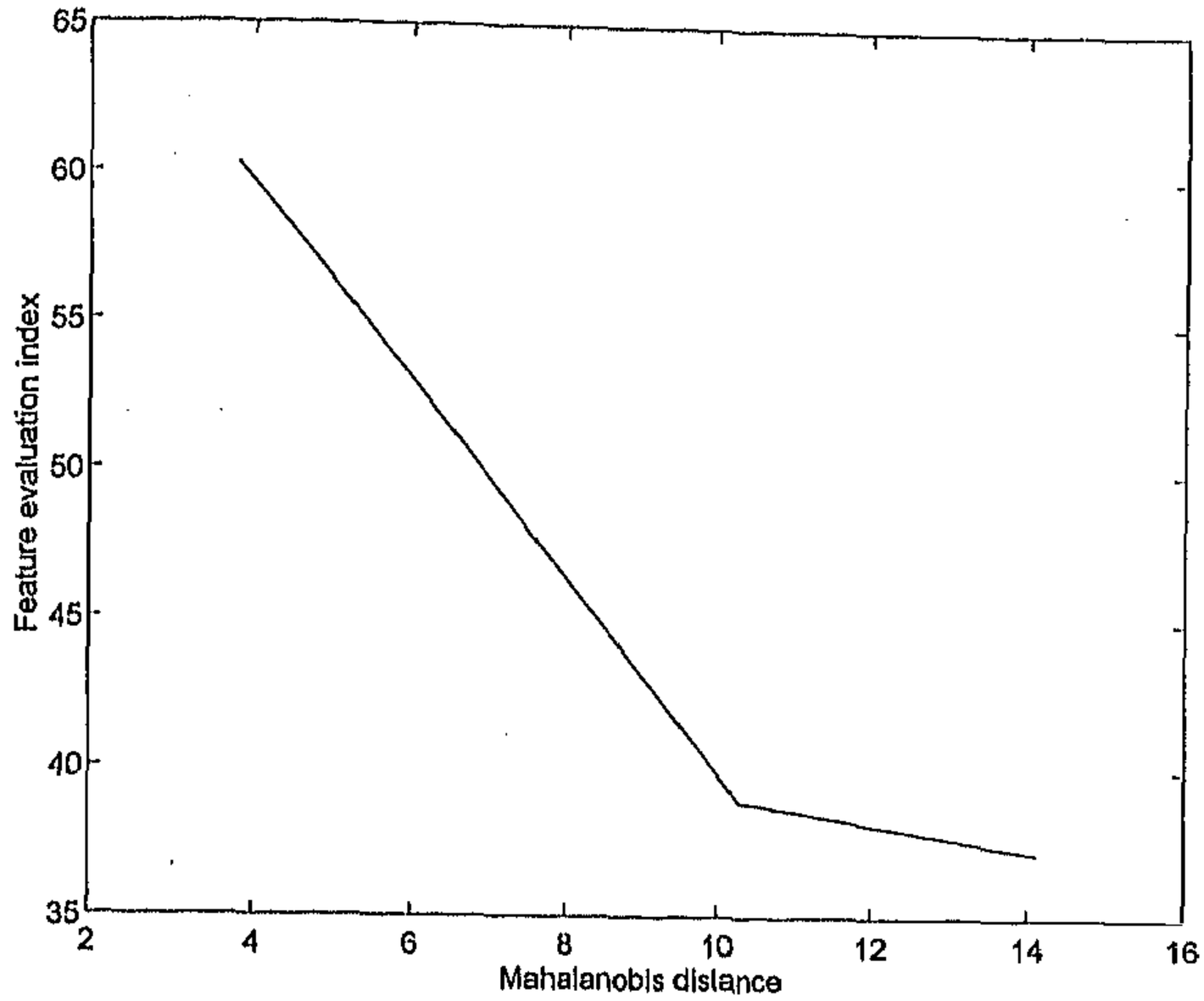

Figure 3.9: Graphical representation of the relationship between feature evaluation index and Mahalanobis distance for Iris data.

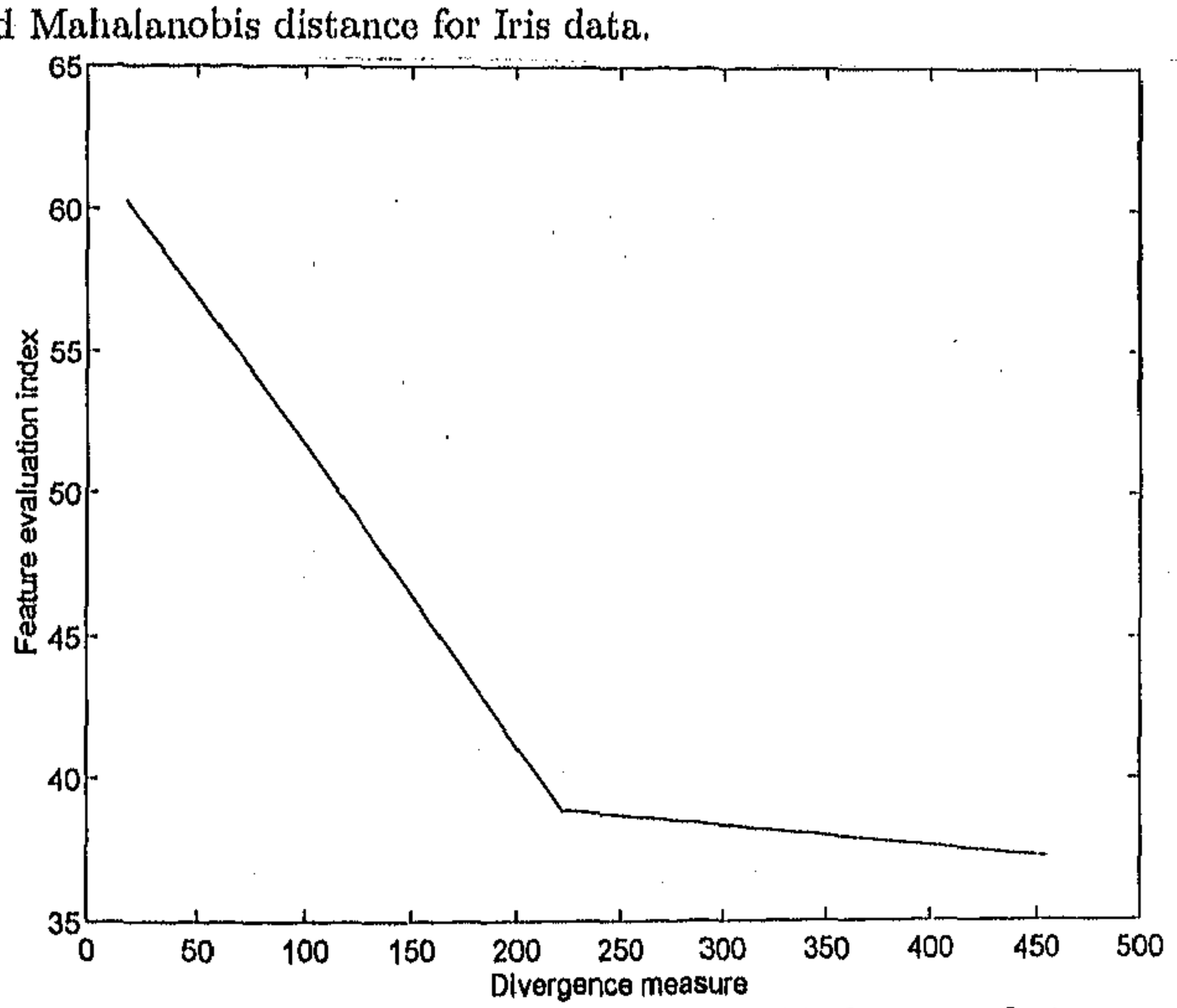

Figure 3.10: Graphical representation of the relationship between feature evaluation index and divergence measure for Iris data.

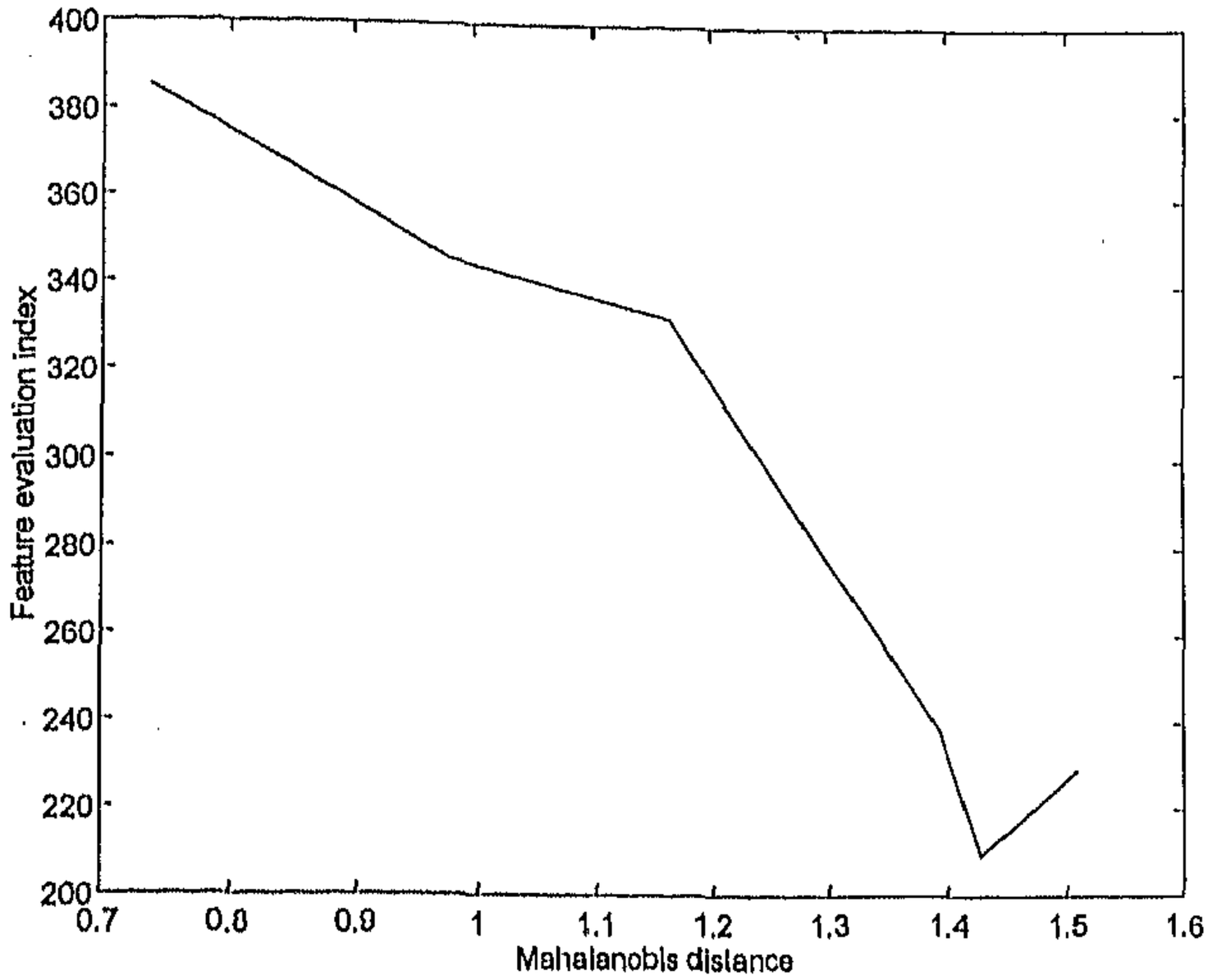

Figure 3.11: Graphical representation of the relationship between feature evaluation index and Mahalanobis distance for the medical data.

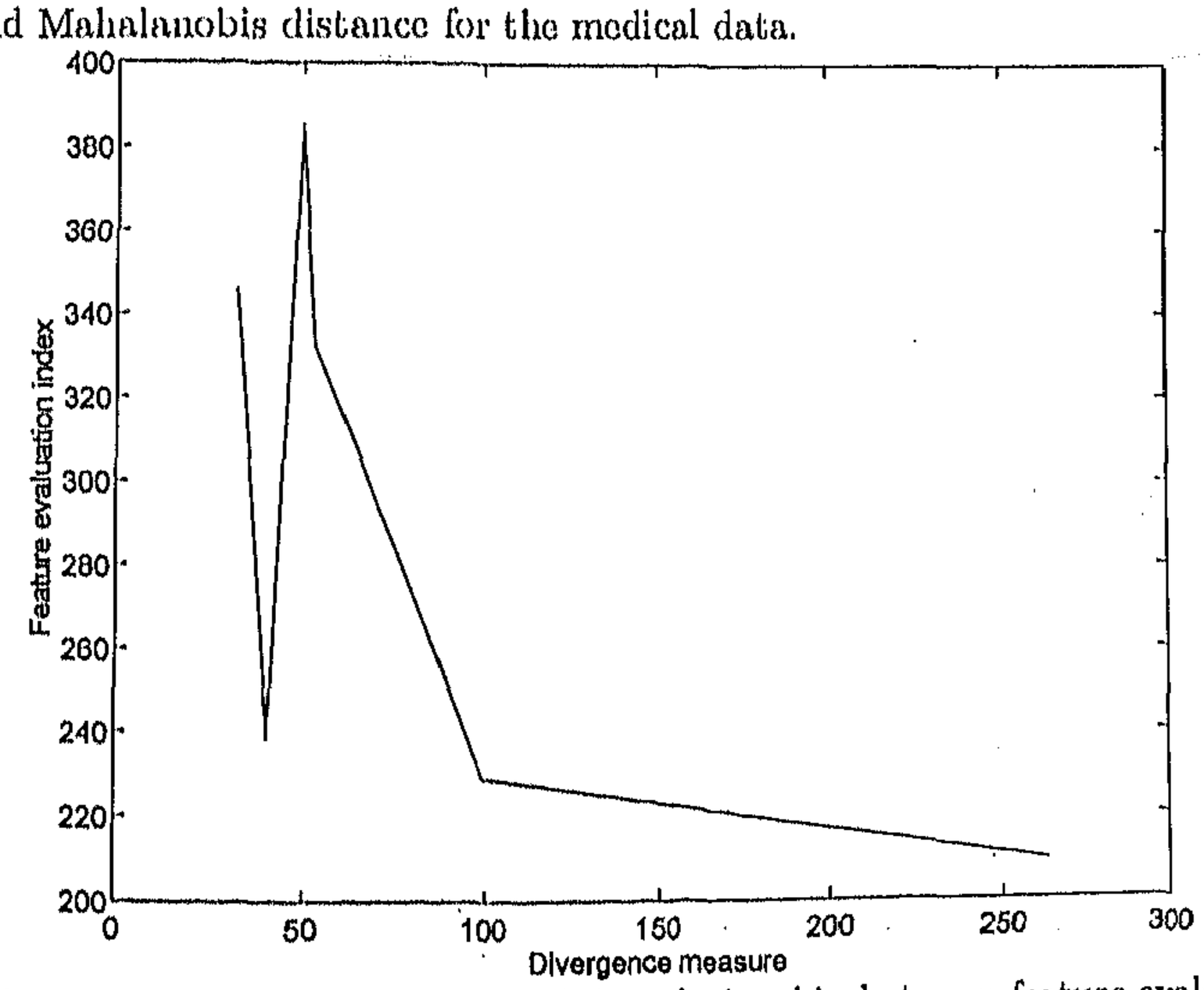

Figure 3.12: Graphical representation of the relationship between feature evaluation index and divergence measure for the medical data.

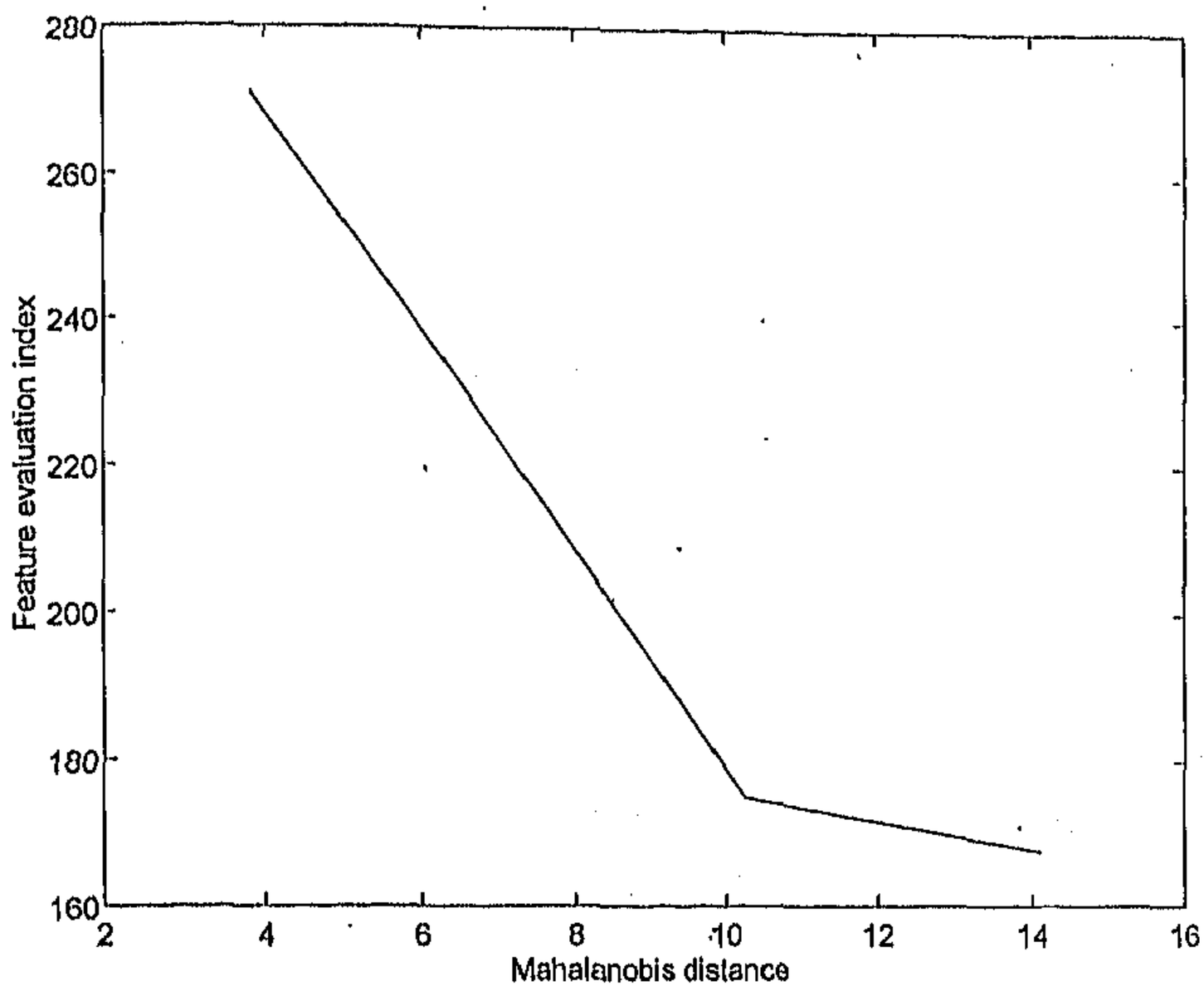

Figure 3.13: Graphical representation of the relationship between feature evaluation index and Mahalanobis distance for mango-leaf data.

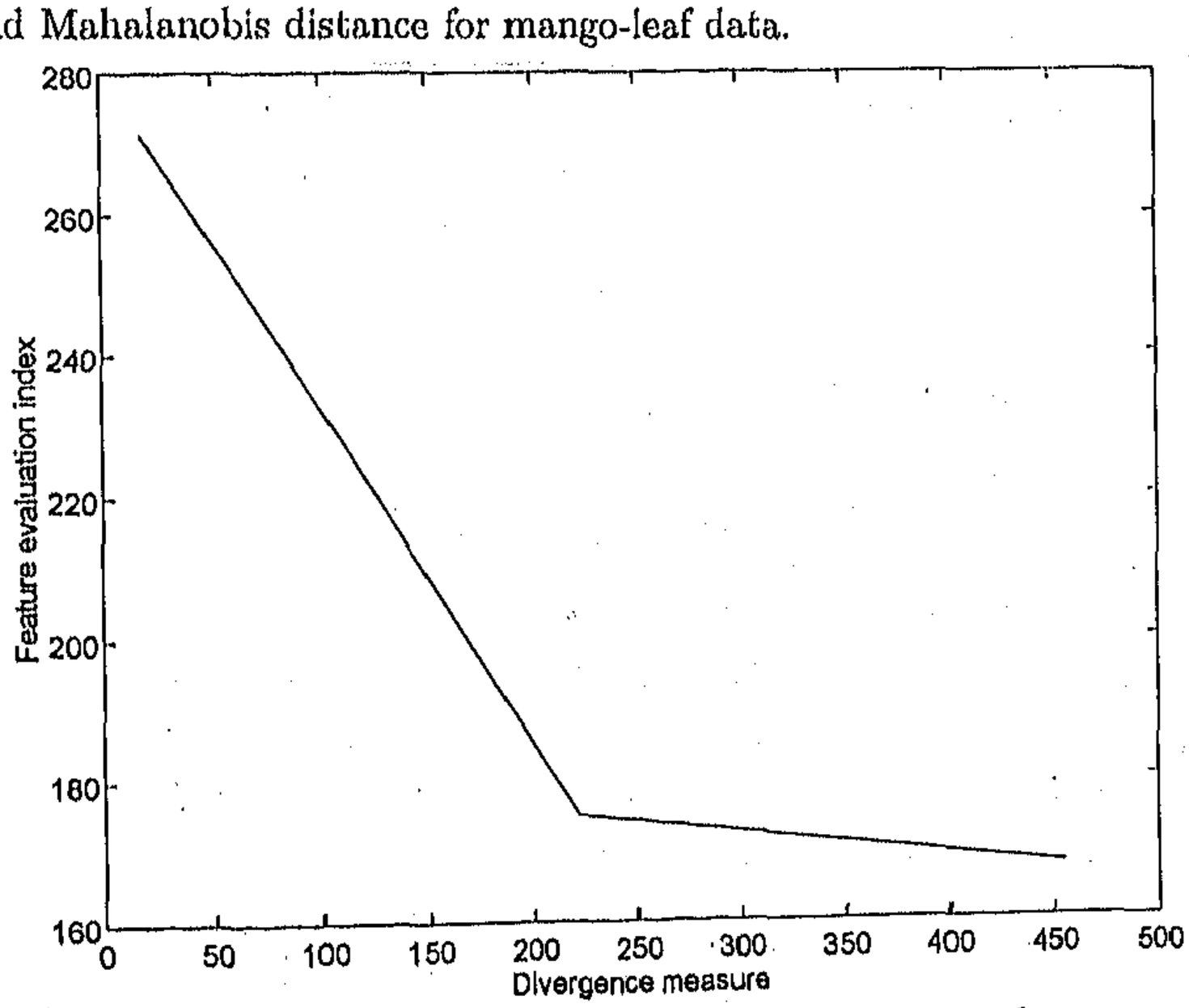

Figure 3.14: Graphical representation of the relationship between feature evaluation index and divergence measure for mango-leaf data.

3.5.2 Using the neuro-fuzzy method

Tables 3.4–3.7 provide the degrees of importance (w) of individual features, obtained by the neural network-based method (Section 3.3), corresponding to the vowel, Iris, medical and mango-leaf data. Three different initializations of w were used in order to train the network. These are:
(i) $w_i = 1$, for all i, *i.e.*, all the features are considered to be equally most important,
(ii) $w_i \in [0, 1]$, for all i, *i.e.*, the network starts searching for a sub-optimal set of weights from an arbitrary point in the search space, and
(iii) $w_i = 0.5 \pm \epsilon$, for all i, $\epsilon \in [0, 0.01]$. In this case the features are considered to be almost equally but not fully important. Note that, $w_i = 1$ means the feature x_i is most important. That is, its presence is a must for characterizing the pattern classes. Similarly, $w_i = 0$ means x_i has no importance and therefore, its presence in the feature vector is not required. $w_i = 0.5$ indicates an ambiguous situation about such presence of x_i. ϵ adds a small perturbation to the degree of presence/importance.

It is found from Table 3.4 that the order of importance of individual features for the vowel data, under all initializations of w, is $F_2 > F_1 > F_3$ which is the same as obtained by both E (Eqn. (3.1)), $FEI^{(av)}$ [152], FQI and FDI (Chapter 2). For Iris data (Table 3.5), like E (Eqn. (3.1)), $FEI^{(av)}$ [152] and $OFEI$ (Chapter 2), PL and PW are found to be the best two features. Note that this result is different from that obtained by the MLP-based method (Chapter 2), where the best two features are found to be PL and SW. As established in Section 3.5.1 by the scatter plots (Figs. 2.5–2.10) and the results of k-NN classifier (Table 3.2), $\{PL, PW\}$ is the best feature pair. Within them it is hard to find the edge of one over the other. This justifies the interchangeable order as obtained by E (Eqn. (3.1)) and $FEI^{(av)}$ [152] between PW and PL.

In the case of medical data (Table 3.6), the order of the best four features as obtained by the neuro-fuzzy approach is $MCV > GOT > GPT > LDH$, whereas this is $MCV > MCH > LDH > TBil$ by Eqn. (3.1). Note that, MCV has come out as the best individual feature in both the cases. Table 3.8 shows that the results of k-NN classifier using these feature sets. Here, the neuro-fuzzy method is seen to perform better than E (Eqn. (3.1)) (with respect to classification performance) for all values of k. On the other hand, for mango-leaf data, the set of best four features

Table 3.4: Importance of different features of vowel data.

Feature	Initial w					
	$= 1.0$		in $[0,1]$		$= 0.5 \pm \epsilon$	
	w	Rank	w	Rank	w	Rank
F_1	0.640382	2	0.257358	2	0.213647	2
F_2	0.759389	1	0.437536	1	0.342621	1
F_3	0.435496	3	0.154319	3	0.123651	3

Table 3.5: Importance of different features of Iris data.

Feature	Initial w					
	$= 1.0$		in $[0,1]$		$= 0.5 \pm \epsilon$	
	w	Rank	w	Rank	w	Rank
SL	0.480797	4	0.203230	4	0.229066	4
SW	0.572347	3	0.302529	3	0.374984	3
PL	0.617570	1	0.422186	1	0.420367	1
PW	0.617173	2	0.402027	2	0.402833	2

Table 3.6: Importance of different features of the medical data.

Feature	Initial w					
	$= 1.0$		in $[0, 1]$		$= 0.5 \pm \epsilon$	
	w	Rank	w	Rank	w	Rank
GOT	0.576090	2	0.601643	2	0.613058	2
GPT	0.300417	3	0.529896	3	0.534147	3
LDH	0.181370	4	0.341677	4	0.322765	4
GGT	0.133649	5	0.300638	5	0.235711	6
BUN	0.070480	9	0.142536	8	0.123007	9
MCV	0.735713	1	0.748205	1	0.747224	1
MCH	0.128931	6	0.101046	7	0.300428	5
TBil	0.123402	7	0.204479	6	0.201762	7
CRTNN	0.103465	8	0.125008	9	0.149290	8

obtained by the neuro-fuzzy approach (Table 3.7) is found to perform poorer (Table 3.9). In this connection we mention here that the neuro-fuzzy method considers interdependence among the features, whereas the other method assumes features to be independent of each other.

As mentioned in Section 3.2, the transformed feature space is obtained by multiplying the original feature values with their respective (optimum) weighting coefficients as obtained by the ANN model. As typical illustrations, Figs. 3.15–3.17 depict three scatter plots in the 2-dimensional transformed spaces for Iris data. Note that, the scales along both the transformed axes are kept identical to those of the original ones, for the sake of comparison. From Figs. 2.5–2.10 and 3.15–3.17 it is seen that the classes in the transformed feature spaces are more compact than those in the original spaces; thereby validating one of the objectives of the algorithm. In order to support this finding, k-NN classifier was also used on the transformed spaces. It was found, for example, for the pair $\{PL, PW\}$ that k-NN classifier results in 94%, 94%, 96%, 96.67% and 97.33% in the transformed space as compared to 93.33%, 94%, 96%, 96% and 96.67% in the original one for $k = 1, 2, 3, 5$ and 9 respectively. Similarly, for overlapping vowel classes, the classification performance is seen to improve in the transformed space for lower values of k. For example, for the feature pairs $\{F_1, F_2\}$,

Table 3.7: Importance of different features of mango-leaf data.

Feature	Initial w					
	$= 1.0$		in $[0, 1]$		$= 0.5 \pm \epsilon$	
	w	Rank	w	Rank	w	Rank
Z	0.398839	13	0.096816	10	0.007504	17
A	0.509456	9	0.080296	12	0.121824	13
Pe	0.451312	12	0.080145	13	0.209411	11
L	0.507300	11	0.070094	14	0.007141	18
B	0.598589	5	0.426404	4	0.445410	5
P	0.273254	17	0.012582	15	0.300251	9
K	0.600539	4	0.411154	5	0.457997	4
S	0.535693	7	0.186507	9	0.328927	6
SI	0.313462	15	0.008756	16	0.201877	12
L+P	0.508099	10	0.300547	7	0.233489	10
L/P	0.191838	18	0.096777	11	0.111012	15
L/B	0.588887	6	0.213001	8	0.310926	7
(L+P)/B	0.293149	16	0.007061	18	0.116798	14
A/L	0.625549	3	0.500711	3	0.529431	3
A/B	0.523274	8	0.401327	6	0.309092	8
A/Pe	0.643935	2	0.600085	2	0.714805	2
UM/LM	0.322303	14	0.007913	17	0.095220	16
UPe/LPe	1.0	1	0.768731	1	0.720648	1

Table 3.8: Recognition score for medical data with k-NN classifier corresponding to four best individual features, obtained by the neuro-fuzzy method and E.

Feature Subset	% classification									
	$k = 1$	Rank	$k = 2$	Rank	$k = 3$	Rank	$k = 5$	Rank	$k = 16$	Rank
$\{GOT, GPT, LDH, MCV\}$	44.40	1	45.90	1	48.51	1	47.76	1	48.88	1
$\{LDH, MCV, MCH, TBil\}$	43.66	2	38.06	2	40.67	2	45.90	2	45.15	2

Table 3.9: Recognition score for mango-leaf data with k-NN classifier corresponding to four best individual features, obtained by the neuro-fuzzy method and E.

Feature Subset	% classification									
	$k=1$	Rank	$k=2$	Rank	$k=3$	Rank	$k=5$	Rank	$k=9$	Rank
$\{K, A/L, A/Pe, UPe/LPe\}$	61.90	2	67.86	2	67.86	2	64.29	2	70.24	2
$\{B, SI, L/B, UPe/LPe\}$	76.19	1	80.95	1	78.57	1	77.38	1	77.38	1

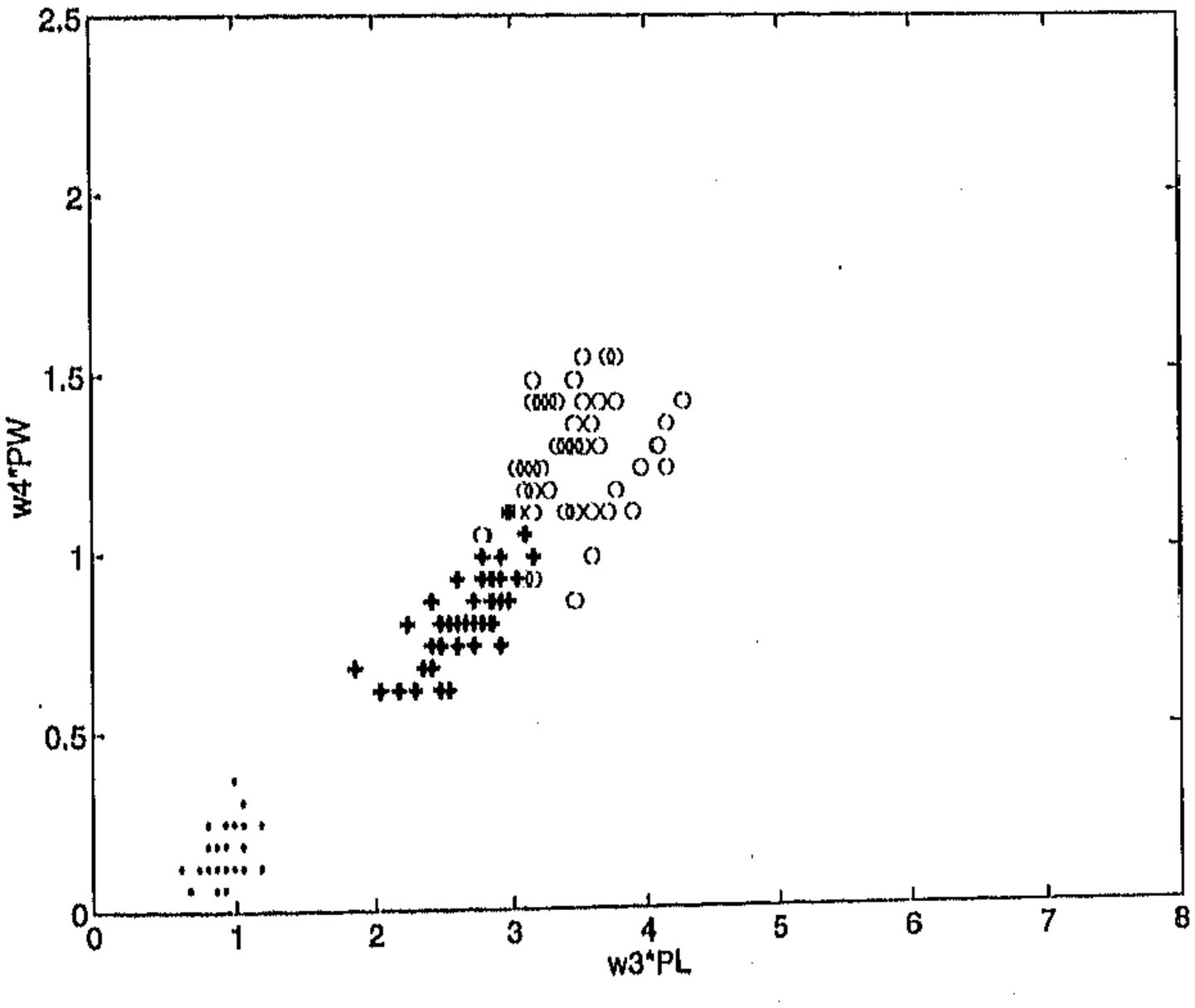

Figure 3.15: Scatter plot $PL - PW$, in the transformed space, of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

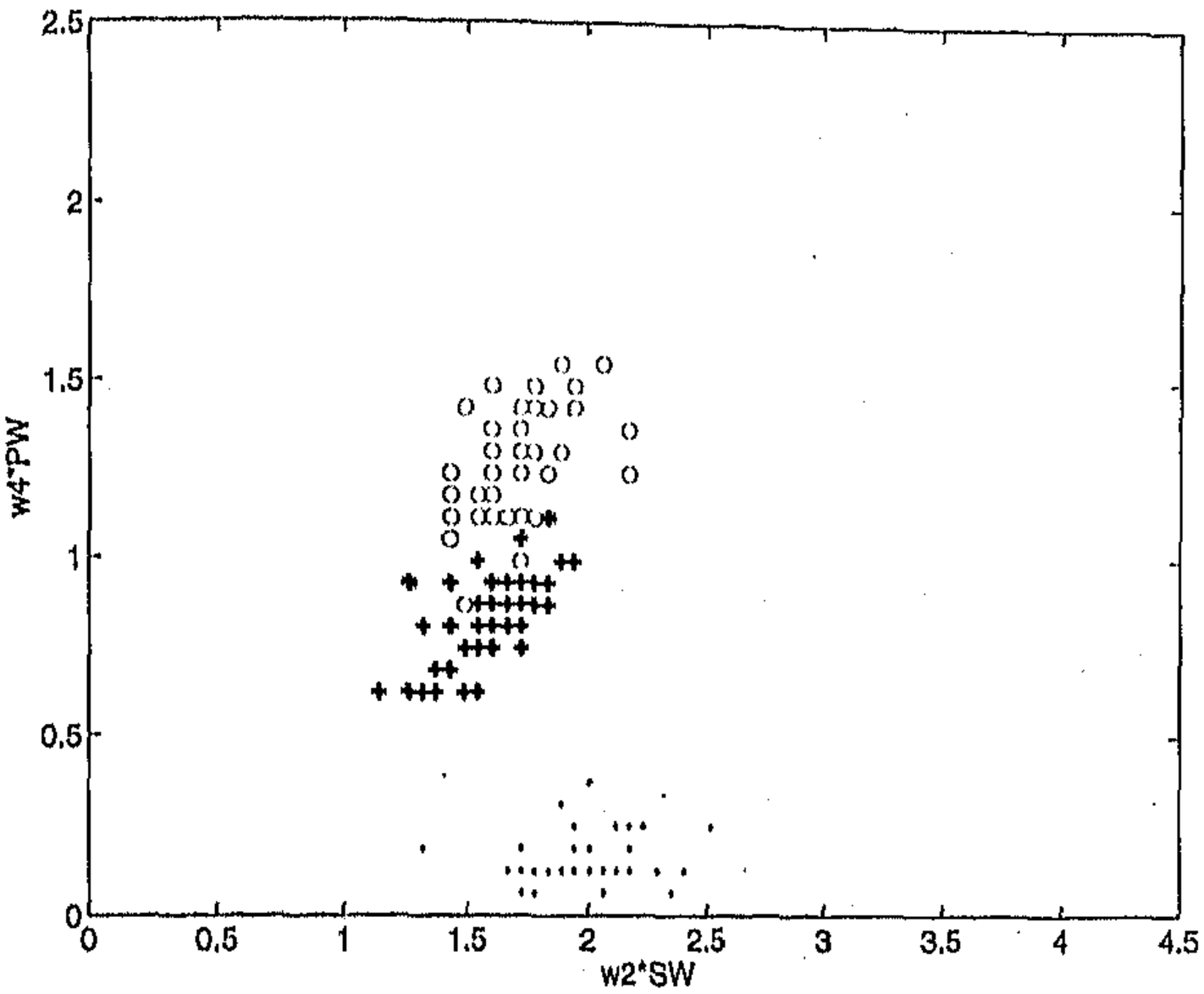

Figure 3.16: Scatter plot $SW - PW$, in the transformed space, of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

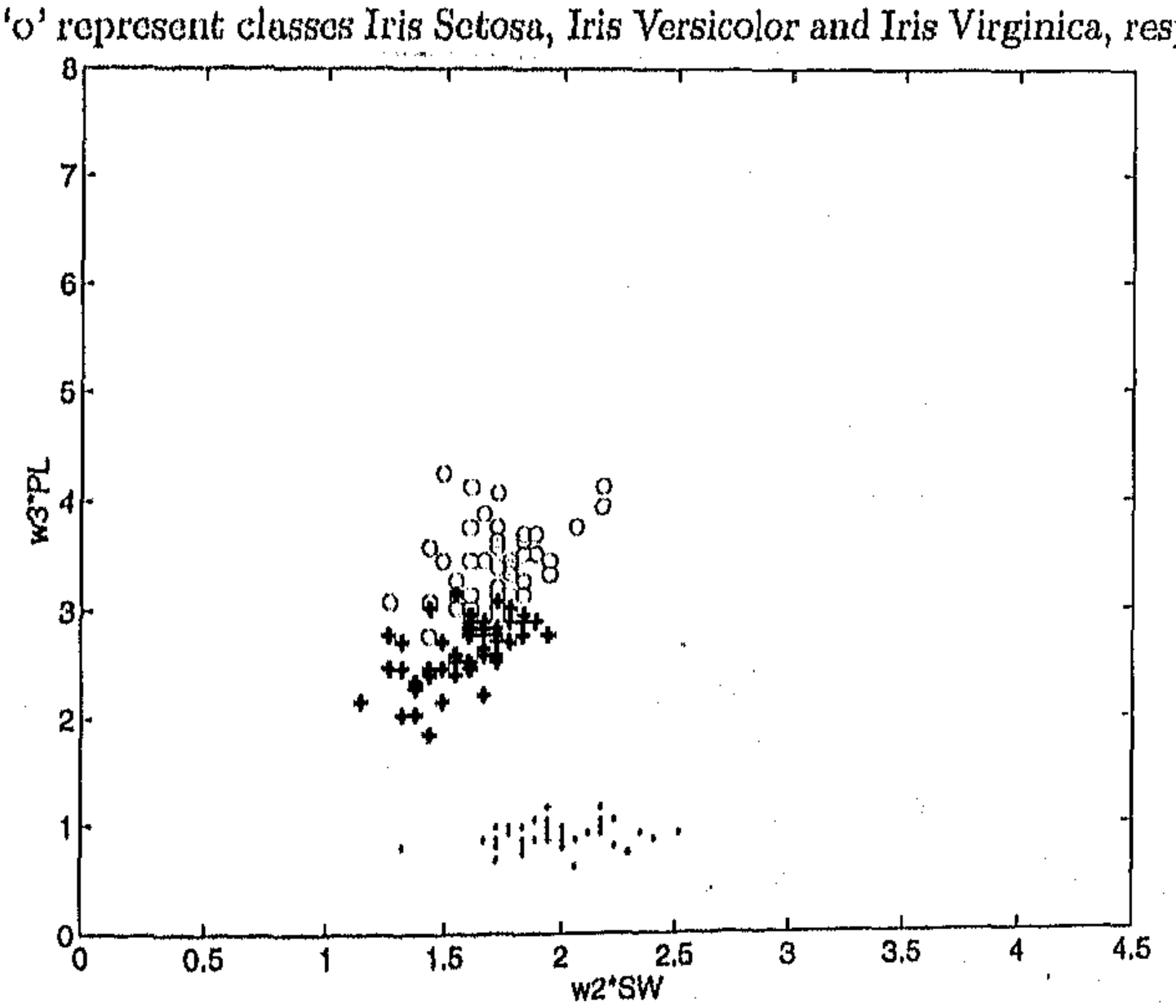

Figure 3.17: Scatter plot $SW - PL$, in the transformed space, of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

$\{F_1, F_3\}$ and $\{F_2, F_3\}$ in the transformed space, k-NN classifier results in 59.01%, 55.34% and 62.80% for $k = 1$, and 57.98%, 52.81% and 60.05% for $k = 2$. In contrast to that the figures are (Table 3.3) 56.37%, 44.32% and 58.21% for $k = 1$, and 55.68%, 45.58% and 56.14% for $k = 2$ in the original space.

It has been observed experimentally that the network converges much slower with the initialization $w_i = 1$, for all i, as compared to the other values. For example, the number of iterations required to converge the network corresponding to the initializations 1, [0, 1] and $0.5 \pm \epsilon$ are 17300, 10000 and 11500 for vowel data, 9400, 7000 and 5600 for the Iris data, 4700, 3000 and 1900 for medical data, and 1700, 1200 and 900 for mango-leaf data.

3.6 Conclusions and Discussion

In this chapter we have presented a neuro-fuzzy model for feature evaluation along with its theoretical analysis and experimental performance on speech (vowel) data, Iris data, medical data and mango-leaf data (having dimension three, four, nine and eighteen respectively). The investigation demonstrates a way how the concept of neuro-fuzzy integration can be exploited for developing a methodology for feature selection. First, a feature evaluation index is defined based on the aggregated measure of compactness of the individual classes and the separation between the classes in terms of class membership functions. The index value decreases with the increase in both the compactness of individual classes and the separation between the classes. Using this index, the best subset from a given set of features can be selected. As Mahalanobis distance and divergence between the classes increase, the feature evaluation index decreases.

Weighting factors representing feature importance are then introduced into membership functions. Incorporation of these weighting factors into membership function gives rise to a transformation of the feature space which provides a generalized framework for modeling class structures. A new connectionist model is designed in order to perform the task of minimizing this index. Note that, this neural network based minimization procedure considers all the features simultaneously, in order to find the relative importance of the features. In other words, the interdependencies of the

features have been taken into account unlike that in Section 3.5.1.

It is shown theoretically that the evaluation index has a fixed upper bound and a varying lower bound. The monotonic increasing behavior of the evaluation index with respect to the lower bound is established for different cases. A relation of the evaluation index, interclass distance and weighting coefficients is derived. It is also shown that the higher the interclass distances, the greater is the chance of the network in getting converged into local minima.

Results obtained by the feature evaluation index E of Eqn. (3.1) is seen to be superior to that of $FEI^{(av)}$ of Pal [152]. This is validated by both scatter plots (*i.e.*, in terms of class structures) and k-NN classifier (*i.e.*, in terms of classification performance). Moreover, in the index $FEI^{(av)}$, the separation between two classes is measured by pooling the classes together, and modeling them with a single membership function. Therefore, for an M-class problem, the number of membership functions required is $M + \begin{pmatrix} M \\ 2 \end{pmatrix}$; where the first term and the second term correspond to individual class and pairwise class membership functions respectively. In other words, one needs $M(M+1)$ parameters for computing the $FEI^{(av)}$. On the other hand, for computing the evaluation index E, we need to compute only M individual class membership functions *i.e.*, $2M$ parameters.

Individual ranking, as obtained by the neuro-fuzzy method, conforms well to those obtained by E (Eqn. (3.1)) for both vowel and Iris data. For medical data the former method is seen to perform better as per the k-NN classifier is concerned, whereas it is the reverse for the mango-leaf data. Although the ranking obtained by the neuro-fuzzy method, in the case of vowel data, has been found to be identical to that of the MLP-based method (Chapter 2), for Iris data the former method has provided better result than the latter. This may be one of the justifications of considering the notion of fuzzy sets in connectionist framework.

Chapter 4

Neuro-fuzzy Unsupervised Feature Selection

4.1 Introduction

Connectionist feature selection algorithms described in Chapters 2 and 3 are based on supervised training. The present chapter deals with an unsupervised method for neuro-fuzzy feature selection [11, 12, 165]. The methodology involves formulation of a layered network for minimization of a fuzzy feature evaluation index. The fuzzy feature evaluation index for a set of features is defined in terms of membership values denoting the degree of similarity between two patterns with respect to belonging to a class both in the original and the transformed spaces. Unlike Chapter 3, this does not need to know the information on class labels of the samples. The evaluation index is such that, for a set of features, the lower is its value, the higher is the importance of that set in characterizing/discriminating various clusters.

As in Chapter 3, a set of weighting coefficients is used to denote the degree of importance of the individual features in characterizing/discriminating different clusters and to provide flexibility in modeling various clusters. The similarity between two patterns in the transformed space, which is obtained by incorporating these weighting coefficients in the original feature space, is measured by an weighted distance between them. Minimization of the evaluation index through unsupervised learning of the network determines the optimum weighting coefficients providing an ordering of the importance of features individually. Note that all these operations, as in Chapter 3, are embedded in a single layered network, and the algorithm considers interdependence of the original features. It is also to be noted that the number of clusters present in the feature space, unlike Chapter 3, does not need to be assumed.

The effectiveness of the algorithm is demonstrated on the same Iris, vowel, medical and mango-leaf data described in Chapter 2. The validity of the results is investigated using scatter plots and k-NN classifier for different values of k.

The chapter is organized as follows. The fuzzy feature evaluation index is defined in Section 4.2. Section 4.3 describes the neural network model for unsupervised feature selection. Section 4.4 provides the analysis of experimental results. Conclusions and Discussion are mentioned in Section 4.5

4.2 Fuzzy Feature Evaluation Index and Weighted Membership Function

In this section we first of all provide a definition of the fuzzy feature evaluation index. The membership function for its realization is then defined in terms of distance measure and weighting coefficients.

Let, μ_{pq}^O be the degree that both the pth and qth patterns belong to the same cluster in the n-dimensional original feature space, and μ_{pq}^T be that in the n'-dimensional ($n' \leq n$) transformed feature space. μ values determine how similar a pair of patterns are in the respective features spaces. That is, μ may be interpreted as the membership value of a pair of patterns belonging to the fuzzy set "similar". Let, s be the number of samples on which the feature evaluation index is computed.

The feature evaluation index for a set (Ω) of transformed features is defined as

$$E = \frac{2}{s(s-1)} \sum_{p} \sum_{q \neq p} \frac{1}{2} [\mu_{pq}^T (1 - \mu_{pq}^O) + \mu_{pq}^O (1 - \mu_{pq}^T)]. \tag{4.1}$$

It has the following characteristics.
(i) If $\mu_{pq}^O = \mu_{pq}^T = 0$ or 1, the contribution of the pair of patterns to the evaluation index E is zero (minimum).
(ii) If $\mu_{pq}^O = \mu_{pq}^T = 0.5$, the contribution of the pair of patterns to E becomes 0.25 (maximum).
(iii) For $\mu_{pq}^O < 0.5$ as $\mu_{pq}^T \rightarrow 0$, E decreases.
For $\mu_{pq}^O > 0.5$ as $\mu_{pq}^T \rightarrow 1$, E decreases.

Therefore, the feature evaluation index decreases as the membership value representing the degree of belonging of pth and qth patterns to the same cluster in the transformed feature space tends to either 0 (when $\mu^O < 0.5$) or 1 (when $\mu^O > 0.5$). In other words, the feature evaluation index decreases as the decision on the similarity between a pair of patterns (*i.e.*, whether they lie in the same cluster or not) becomes more and more crisp. This means, if the intercluster/intracluster distances in the transformed space increase/decrease, the feature evaluation index of the corresponding set of features decreases. Therefore, our objective is to select/extract those features for which the evaluation index becomes minimum; thereby optimizing the decision on the similarity of a pair of patterns with respect to their belonging to a

cluster.

The membership function μ_{pq} in a feature space, satisfying the characteristics of E (Eqn. (4.1)), may be defined as

$$\begin{aligned} \mu_{pq} &= 1 - \tfrac{d_{pq}}{D} \quad if\ d_{pq} \leq D, \\ &= 0, \qquad\quad otherwise. \end{aligned} \tag{4.2}$$

d_{pq} is a distance measure which provides similarity (in terms of proximity) between the pth and qth patterns in the feature space. Note that, the higher the value of d_{pq}, the lower is the similarity between pth and qth patterns, and *vice versa*. D is a parameter which reflects the minimum separation between a pair of patterns belonging to two different clusters. When $d_{pq} = 0$ and $d_{pq} = D$, we have $\mu_{pq} = 1$ and 0, respectively. If $d_{pq} = \frac{D}{2}$, $\mu_{pq} = 0.5$. That is, when the distance between the patterns is just half the value of D, the difficulty in making a decision, whether both the patterns are in the same cluster or not, becomes maximum; thereby making the situation most ambiguous.

The term D (in Eqn. (4.2)) may be expressed as

$$D = \beta d_{max}, \tag{4.3}$$

where d_{max} is the maximum separation between a pair of patterns in the entire feature space, and $0 < \beta \leq 1$ is a user defined constant. β determines the degree of flattening of the membership function (Eqn. (4.2)). The higher the value of β, more will be the degree, and *vice-versa*.

The distance d_{pq} (Eqn. (4.2)) can be defined in many ways. Considering Euclidian distance, we have

$$d_{pq} = [\sum_i (x_{pi} - x_{qi})^2]^{\frac{1}{2}}, \tag{4.4}$$

where x_{pi} and x_{qi} are values of ith feature (in the corresponding feature space) of pth and qth patterns, respectively. d_{max} is defined as

$$d_{max} = [\sum_i (x_{maxi} - x_{mini})^2]^{\frac{1}{2}}, \tag{4.5}$$

where x_{maxi} and x_{mini} are the maximum and minimum values of the ith feature in the corresponding feature space.

Incorporating weighting coefficients

In the above discussion, we have measured the similarity between two patterns in terms of proximity, as conveyed by the expression for d_{pq} (Eqn. (4.4)). Since, d_{pq} is an Euclidian distance, the methodology implicitly assumes that the clusters are hyperspherical. But in practice, this may not necessarily be the case. To model the practical situation we have introduced the concept of weighted distance such that

$$\begin{aligned} d_{pq} &= [\sum_i w_i^2 (x_{pi} - x_{qi})^2]^{\frac{1}{2}}, \\ &= [\sum_i w_i^2 \chi_i^2]^{\frac{1}{2}}, \quad \chi_i = (x_{pi} - x_{qi}), \end{aligned} \tag{4.6}$$

where $w_i \in [0, 1]$ represents weighting coefficient corresponding to ith feature.

The membership value μ_{pq} is now obtained by Eqns. (4.2), (4.3), (4.5) and (4.6), and becomes dependent on w_i. The values of w_i (< 1) make the μ_{pq} function of Eqn. (4.2) flattened along the axis of d_{pq}. The lower the value of w_i, the higher is extent of flattening. In the extreme case, when $w_i = 0$, $\forall i$, $d_{pq} = 0$ and $\mu_{pq} = 1$ for all pair of patterns, *i.e.*, all the patterns lie on the same point making them indiscriminable.

The weight w_i (in Eqn. (4.6)) reflects the relative importance of the feature x_i in measuring the similarity (in terms of distance) of a pair of patterns. The higher the value of w_i, the more is the importance of x_i in characterizing a cluster or discriminating various clusters. $w_i = 1$ (0) indicates most (least) importance of x_i.

Note that, one may define μ_{pq} in a different way satisfying the above mentioned characteristics. The computation of μ_{pq} in Eqn. (4.2) does not require the information on class label of the patterns.

4.3 Neural Network Model for Unsupervised Feature Selection

As mentioned in Section 4.2, our objective is to minimize the evaluation index E (Eqn. (4.1)) which involves the terms μ^O and μ^T. Note that the n-dimensional transformed space is obtained by introducing $\mathbf{w}$ ($= [w_1, w_2, \ldots w_n]$) on the n-dimensional original space. The computation of μ^O requires Eqns. (4.2)–(4.5), while μ^T needs Eqns. (4.2), (4.3), (4.5) and (4.6). Therefore, the evaluation index E (Eqn. (4.1)) becomes a function of $\mathbf{w}$, if we consider ranking of n features in a set.

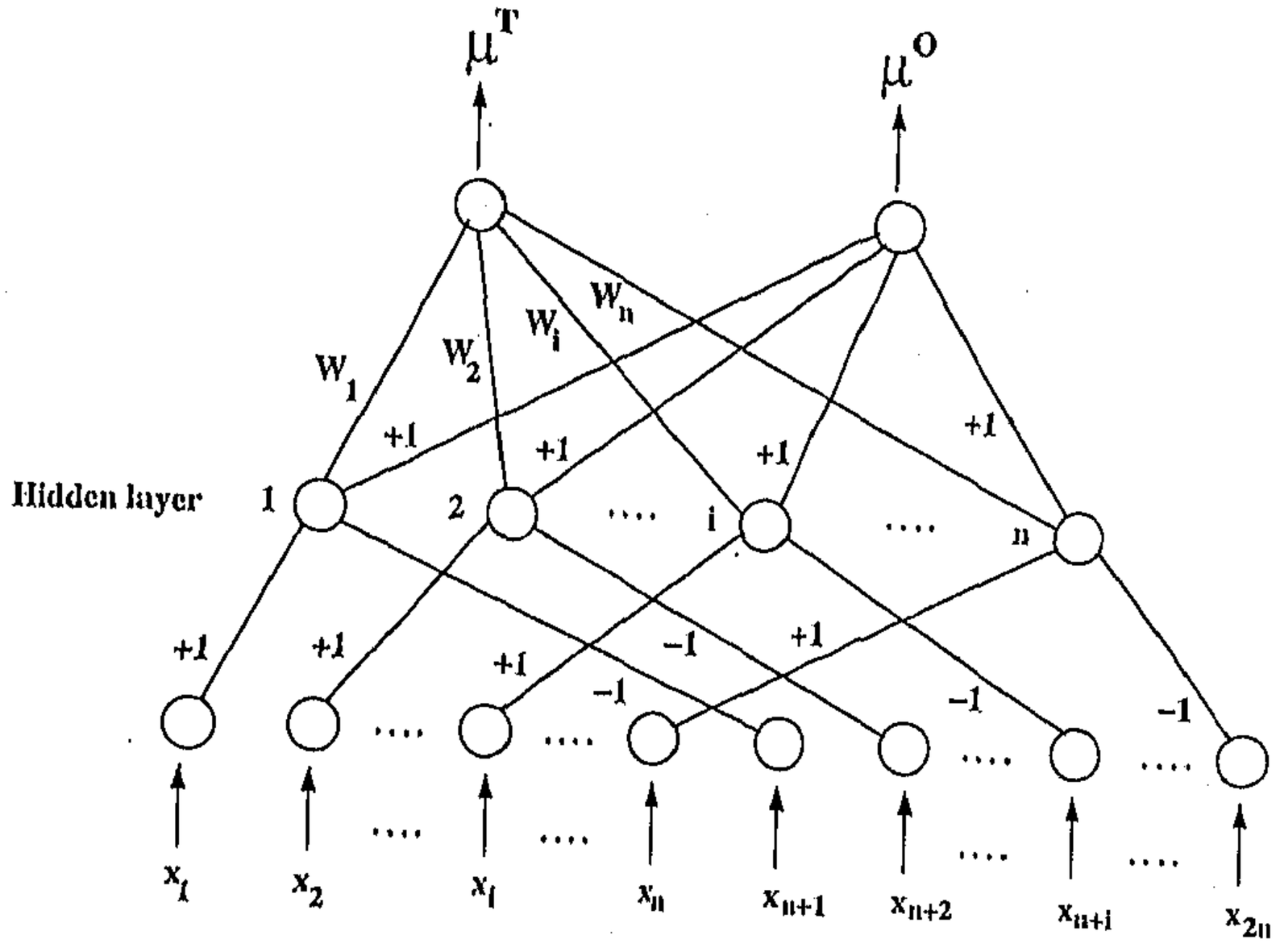

Figure 4.1: A schematic diagram of the neural network model for unsupervised feature selection.

The problem of feature selection/ranking thus reduces to finding a set of w_is for which E becomes minimum; w_is indicating the relative importance of x_is. The task of minimization is performed using gradient-descent technique under unsupervised mode. The network designed for performing all these operations (*i.e.*, computation of μ^O and μ^T, and minimization) is described below.

Connectionist model

The network (Fig. 4.1) consists of an input, a hidden and an output layer. The input layer consists of a pair of nodes corresponding to each feature, *i.e.*, the number of nodes in the input layer is $2n$, for n-dimensional (original) feature space. The hidden layer consists of n number of nodes which compute the part χ_i^2 of Eqn. (4.6) for each pair of patterns. The output layer consists of two nodes. One of them computes μ^O, and the other μ^T. The feature evaluation index E (Eqn. (4.15)) is computed from these μ-values off the network.

Input nodes receive activations corresponding to feature values of each pair of patterns. A jth node in the hidden layer is connected only to an ith and $(i+n)$th

input nodes via connection weights +1 and −1, respectively, where $j, i = 1, 2, \ldots, n$ and $j = i$. The output node computing μ^T-values is connected to a jth node in the hidden layer via connection weight W_j $(= w_j^2)$, whereas that computing μ^O-values is connected to all the nodes in the hidden layer via connection weights +1 each.

During learning, each pair of patterns are presented at the input layer and the evaluation index is computed. The weights W_js are updated using gradient-descent technique in order to minimize the index E. Note that, d_{max} is directly computed from the unlabeled training set. The values of d_{max} and β are stored in both the output nodes for the computation of D. When pth and qth patterns are presented to the input layer, the activation produced by ith $(1 \leq i \leq 2n)$ input node is

$$v_i^{(0)} = u_i^{(0)} \tag{4.7}$$

where

$$\begin{aligned} u_i^{(0)} &= x_{pi}, \quad for\ 1 \leq i \leq n \ \ and \\ u_{i+n}^{(0)} &= x_{qi}, \quad for\ 1 \leq i \leq n, \end{aligned} \tag{4.8}$$

the total activations received by ith and $(i+n)$th $(1 \leq i \leq n)$ input node, respectively. The total activation received by jth hidden node (connecting ith and $(i+n)$th, $1 \leq i \leq n$, input nodes) is given by

$$u_j^{(1)} = 1 \times v_i^{(0)} + (-1) \times v_{i+n}^{(0)}, \quad for\ 1 \leq i \leq n, \tag{4.9}$$

and the activation produced by it is

$$v_j^{(1)} = (u_j^{(1)})^2. \tag{4.10}$$

The total activation received by the output node which computes μ^T-values, is

$$u_T^{(2)} = \sum_j W_j v_j^{(1)}, \tag{4.11}$$

and that received by the other output node which computes μ^O-values, is

$$u_O^{(2)} = \sum_j v_j^{(1)}. \tag{4.12}$$

Therefore, $u_T^{(2)}$ and $u_O^{(2)}$ represent d_{pq}^2 as given by Eqns. (4.6) and (4.4), respectively. The activations, $v_T^{(2)}$ and $v_O^{(2)}$, of the output nodes represent μ_{pq}^T and μ_{pq}^O for pth and qth pattern pair, respectively. Thus,

$$\begin{aligned} v_T^{(2)} &= 1 - \frac{(u_T^{(2)})^{\frac{1}{2}}}{D}, \quad if\ (u_T^{(2)})^{\frac{1}{2}} \leq D, \\ &= 0, \qquad\qquad otherwise, \end{aligned} \tag{4.13}$$

and

$$\begin{aligned} v_O^{(2)} &= 1 - \frac{(u_O^{(2)})^{\frac{1}{2}}}{D}, \quad if\ (u_O^{(2)})^{\frac{1}{2}} \leq D, \\ &= 0, \qquad\qquad otherwise. \end{aligned} \tag{4.14}$$

The evaluation index (which is computed off the network), in terms of these activations, is then written (from Eqn. (4.1)) as

$$E(\mathbf{W}) = \frac{2}{s(s-1)} \sum_p \sum_{q \neq p} \frac{1}{2} [v_T^{(2)}(1 - v_O^{(2)}) + v_O^{(2)}(1 - v_T^{(2)})]. \tag{4.15}$$

As mentioned before, the task of minimization of $E(\mathbf{W})$ (Eqn. (4.15)) with respect to $\mathbf{W}$ is performed using gradient-descent technique, where the change in W_j ($\triangle W_j$) is computed as

$$\triangle W_j = -\eta \frac{\partial E}{\partial W_j}, \forall j, \tag{4.16}$$

where η is the learning rate.

For computation of $\frac{\partial E}{\partial W_j}$ corresponding to a pair of patterns, the following expressions are used.

$$\frac{\partial E(\mathbf{W})}{\partial W_j} = \frac{1}{2} \left[1 - 2v_O^{(2)}\right] \frac{\partial v_T^{(2)}}{\partial W_j}, \tag{4.17}$$

$$\begin{aligned} \frac{\partial v_T^{(2)}}{\partial W_j} &= -\frac{\frac{1}{2}(u_T^{(2)})^{-\frac{1}{2}} \frac{\partial u_T^{(2)}}{\partial W_j}}{D}, \quad if\ (u_T^{(2)})^{\frac{1}{2}} \leq D \\ &= 0, \qquad\qquad otherwise, \end{aligned} \tag{4.18}$$

and

$$\frac{\partial u_T^{(2)}}{\partial W_j} = v_j^{(1)}. \tag{4.19}$$

Algorithm for learning W

- Calculate d_{max} from the unlabeled training set and store it in both the output nodes. Store β (user specified) in both the output nodes.
- Initialize W_j with small random values in $[0, 1]$.
- Repeat until convergence, *i.e.*, until the value of E becomes less than or equal to certain predefined small quantity, or number of iterations attains certain predefined number of iterations:
  - For each pair of patterns:

* Present the pattern pair to the input layer.
* Compute ΔW_j for each j using the updating rule in Eqn. (4.16).

– Update W_j for each j with ΔW_j averaged over all the patterns.

After convergence, $E(\mathbf{W})$ attains a local minimum. Then the weights ($W_j = w_j^2$) of the links connecting hidden nodes and the output node computing μ^T-values, indicate the order of importance of the features. Note that this unsupervised method performs the task of feature selection without clustering the feature space explicitly and *does not* need to know the number of clusters present in the feature space.

4.4 Results

Here we demonstrate the effectiveness of the above mentioned algorithm on Iris, vowel, medical and *mango-leaf data*. All the data sets are described in Chapter 2. As in Chapter 3, the results on a method where the feature evaluation index E (Eqn. 4.1)) is used *alone* for ordering various subsets of features, are also provided in a part of the investigation. The results of feature selection using E (Eqn. 4.1)) *alone* are shown in Section 4.4.1, followed by that using neuro-fuzzy method in Section 4.4.2.

4.4.1 Using the evaluation index E (Eqn. (4.1))

Table 4.1 shows these ordering for the four types data. Note that, for vowel and Iris data we have computed E-value for all possible subsets, including the individual features, (*i.e.*, seven for vowel and fifteen for Iris data) and ranked them accordingly. For medical and *mango-leaf data*, since the number of features is large, we have, first of all, computed the E-value for the individual features. A few bests of them (e.g., GOT, LDH, GPT, GGT for medical data, and Pe, (L+P)/B, UM/LM, L/B, Z for mango-leaf data) are selected *after ranking*. Then we have computed E-value for different subsets containing only these selected features. As a result, we have 20 subsets for medical data and 44 subsets for mango-leaf data. (However, for mango-leaf data, we have shown in Table 4.1 the ordering of first twenty subsets only, for brevity.)

Table 4.1: Importance of different feature subsets. ($X > Y$ means X is more important than Y.)

Data sets	Order of importance
Vowel	$\{F_1, F_2\} > \{F_2\} > \{F_1, F_2, F_3\} > \{F_2, F_3\} > \{F_1, F_3\} > \{F_3\} > \{F_1\}$
Iris	$\{PL\} > \{PL, PW\} > \{SW, PL\} > \{SL, PL\} > \{SW, PL, PW\} > \{PW\} > \{SL, PL, PW\} > \{SL, SW, PL\} > \{SL, SW, PL, PW\} > \{SL, PW\} > \{SL\} > \{SL, SW, PW\} > \{SW, PW\} > \{SL, SW\} > \{SW\}$
Medical	$\{GOT\} > \{GOT, GPT\} > \{LDH\} > \{GPT, LDH\} > \{GOT, LDH\} > \{GOT, GPT, LDH\} > \{GOT, GGT\} > \{GOT, GPT, GGT\} > \{LDH, GGT\} > \{GPT\} > \{GPT, LDH, GGT\} > \{GOT, LDH, GGT\} > \{GOT, GPT, LDH, GGT\} > \{GGT\} > \{GPT, GGT\} > \{CRTNN\} > \{TBil\} > \{BUN\} > \{MCV\} > \{MCH\}$
Mango-leaf	$\{Pe\} > \{Pe, UM/LM\} > \{Pe, L/B\} > \{Pe, L/B, UM/LM\} > \{Pe, (L+P)/B\} > \{Pe, (L+P)/B, UM/LM\} > \{Pe, L/B, (L+P)/B\} > \{Pe, L/B, (L+P)/B, UM/LM\} > \{Z, Pe, UM/LM\} > \{Z, Pe, L/B\} > \{Z, Pe, L/B, UM/LM\} > \{Z, Pe, (L+P)/B\} > \{Z, Pe, (L+P)/B, UM/LM\} > \{Z, Pe, L/B, (L+P)/B\} > \{Z, Pe, L/B, (L+P)/B, UM/LM\} > \{(L+P)/B\} > \{(L+P)/B, UM/LM\} > \{L/B, (L+P)/B\} > \{L/B, (L+P)/B, UM/LM\} > \{UM/LM\} > \ldots$

It is seen from Table 4.1 that a subset of higher cardinality may not necessarily be more important than ones of lower cardinality. For vowel, F_2 being the best individual feature is seen to be a member of the best four subsets. This conforms to an earlier investigation [152], and in Chapters 2 and 3. Similarly, for Iris data, it is PL which has become a member of the first five best subsets. For medical data, the best 15 subsets contain at least one of the four best individual features (GOT, LDH, GPT and GGT). Similarly, for mango-leaf data, it is the first 10 subsets in which at least one of the five best individual features (Pe, (L+P)/B, UM/LM, L/B, Z) became a member.

We have used k-nn classifier to study the importance of these selected features in classifying the data set under supervised mode. For this purpose, we have used only vowel data with 50% training set and $k = 3$. It is found that the order of importance of the individual features as obtained by k-nn classifier is $F_2 > F_3 > F_1$, which is the same as that obtained in Table 4.1. For the pairwise features also, k-nn classifier and the evaluation index E produce the same ordering *i.e.*, $\{F_1, F_2\} > \{F_2, F_3\} > \{F_1, F_3\}$. It may be noted that k-nn classifier provides the best classification performance with F_1, F_2 and F_3 taken together, although this subset ranks third using E.

4.4.2 Using the neuro-fuzzy method

Tables 4.2–4.5 provide the degrees of importance (w-value) of different features corresponding to these data sets obtained by the neuro-fuzzy approach. Note that, their initial values were considered to be random numbers in $[0, 1]$ while training the network.

The order of importance of the features for the vowel data is found, from Table 4.3, to be $F_2 > F_1 > F_3$, where $x > y$ means feature x is more important than y. This conforms to those obtained in several other investigations based on both feature evaluation and classification under supervised mode, described in Chapters 2 and 3, and in [166, 156, 152]. For Iris data, the best two features are found to be PL and PW (Table 4.2). This is also in agreement with those obtained using a supervised neural [209] and neuro-fuzzy (in Chapter 3) methods.

In the case of medical data, our unsupervised method results in the order of im-

Table 4.2: w-values for Iris data.

Feature	w	Rank
SL	0.058414	4
SW	0.194421	3
PL	0.965575	1
PW	0.603508	2

Table 4.3: w-values for vowel data.

Feature	w	Rank
F_1	0.590065	2
F_2	0.896044	1
F_3	0.120944	3

portance of the nine features as $GOT > LDH > CRTNN > MCH > TBil > BUN > MCV > GPT > GGT$ (Table 4.4), whereas it is $MCV > GOT > GPT > LDH > GGT > MCH > TBil > CRTNN > BUN$, obtained by the neuro-fuzzy supervised feature selection algorithm (in Chapter 3). From these results, it is interesting to note that the relative importance of five features, e.g., $GOT > LDH > MCH > TBil > BUN$, remains the same in both the approaches, although their individual ranks are different. Further, the features GOT and LDH have come out as members of the sets of best four features by both the methods. Similarly for mango-leaf data, such common features are found to be K and A/Pe out of best four by these methods.

One may note that the recognition scores obtained by k-NN classifier using $\{GOT, LDH, MCH, CRTNN\}$ are 44.22, 41.98 and 47.57 for $k = 1, 3$ and 5 respectively, whereas the corresponding figures are 44.40, 48.51 and 47.76 for the set $\{GOT, GPT, LDH, MCV\}$. On the other hand, for the mango-leaf data, these results are 77.71, 69.88 and 69.88 using the set $\{Pe, K, S, A/Pe\}$, and 61.90, 67.86 and 64.29 by $\{K, A/L, A/Pe, UPe/LPe\}$. These demonstrate that our algorithm, although being unsupervised, performs comparable/superior to the method under

supervised learning (Chapter 3).

Table 4.4: w-values for the medical data.

Feature	w	Rank
GOT	0.851015	1
GPT	0.665853	8
LDH	0.733647	2
GGT	0.055946	9
BUN	0.704469	6
MCV	0.704249	7
MCH	0.706765	4
TBil	0.706562	5
CRTNN	0.707109	3

In order to show the validity of these orders of importance, we consider both scatter plots and k-NN classifier for $k = 1, 3$ and 5. The results are shown only for Iris data. From the results of k-NN classifier (Table 4.6), PL and PW are found, as in Table 4.2, to be the best two individual features. PL is seen to be better than PW for $k = 1$, and it is the reverse for $k = 3$ and 5, although the difference is not significant. From Tables 4.2 and 4.6 it is seen that both w-values and recognition scores corresponding to features PL and PW are much higher than those of SL and SW. This signifies the larger difference in importance of PL, PW over SL, SW. The scatter plots, in Figs. 2.5-2.10, also reflect the similar observation that the feature pair $\{PL, PW\}$ is the best of all. From Fig. 2.10 too, it is hard to discriminate the relative importance of PL and PW.

Note that, there is no unsupervised connectionist feature selection approach available in literature, to our knowledge. For this reason, we have referred only to those of the related supervised methods for the sake of comparison.

Table 4.5: w-values for mango-leaf data.

Feature	w	Rank
Z	0.021467	17
A	0.0	18
Pe	1.0	1
L	0.657581	14
B	0.704104	8
P	0.700014	10
K	0.707102	2
S	0.707097	3
SI	0.656674	15
L+P	0.621514	16
L/P	0.674567	12
L/B	0.703800	9
(L+P)/B	0.673539	13
A/L	0.705429	6
A/B	0.680706	11
A/Pe	0.706753	4
UM/LM	0.704375	7
UPe/LPe	0.706594	5

Table 4.6: Recognition score with k-NN classifier for individual features of Iris data.

Feature	% classification		
	$k = 1$	$k = 3$	$k = 5$
SL	48.67	66.67	67.33
SW	55.33	52.67	52.67
PL	93.33	95.33	95.33
PW	89.33	96.00	96.00

4.5 Conclusions and Discussion

In this chapter we have demonstrated how the concept of neuro-fuzzy computing can be exploited for developing a methodology for feature selection under unsupervised mode. The methodology developed involves connectionist optimization of a fuzzy feature evaluation index; thereby determining the ranking of various features. The algorithm considers interdependence of the original features. Unlike the method based on fuzzy c-means algorithm [22], the algorithm provides ranking of individual features without clustering the feature space explicitly. The algorithm also does not need to assume the number of clusters present in the feature space. The effectiveness of the method is demonstrated extensively on a 3-d speech (vowel) data, 4-d Iris data, 9-d medical data and a 18-d mango-leaf data.

Besides the neuro-fuzzy method, we have developed another unsupervised feature selection algorithm where the aforesaid fuzzy evaluation index is used *alone* to find the best subset of features from a given set. Here the algorithm assumes, unlike the neuro-fuzzy methods, independence of the original features. Experimental results on the ordering of original features by both the algorithms conform well to those obtained using other methods [152, 209]. Although a network is used for minimization of the evaluation index, one may consider other optimization techniques for this task.

Chapter 5

Neuro-fuzzy Unsupervised Feature Extraction

5.1 Introduction

In Chapters 2-4, we have discussed various feature selection methods. The present chapter deals with the task of feature extraction under unsupervised training [39, 40, 165]. The methodology involves minimization of the same fuzzy feature evaluation index (Eqn. (4.1)) defined in Chapter 4, but in a different connectionist framework. As mentioned in Chapter 4, the index is defined based on the membership functions denoting the degrees of similarity between two patterns in both the original and transformed feature spaces. The lower the value of the index, the higher is the importance of the transformed features in characterizing/discriminating various clusters. The transformed space is obtained through a set of linear transformations. Computation of membership values in the transformed space involves a set of weighting coefficients. These coefficients provide flexibility in modeling various clusters and reflect the degree of individual importance of the transformed features. A layered network is designed for performing the task of minimization of the said index through unsupervised learning process; thereby extracting the optimum transformed space along with the weighting coefficients. This is described in Fig. 5.1. The algorithm considers interdependence of the original features. The architecture of the network is such that the number of nodes in its second hidden layer determines the desired number of extracted features.

The effectiveness of the algorithm is demonstrated on Iris, vowel, medical and mango-leaf data, described in Chapter 2. The superior discrimination ability of the extracted features over the original ones is established using k-NN classifier for different values of k. The method is also compared with the principal component analysis network (PCAN) of Rubner and Tavan [190] using scatter plots and fuzzy c-means clustering algorithm.

The unsupervised fuzzy feature evaluation index (Eqn. (4.1)), defined in Chapter 4, is mentioned in Section 5.2 briefly for convenience. Section 5.3 describes the feature extraction method. The analysis of the experimental results is provided in Section 5.4. Section 5.5 concludes the chapter.

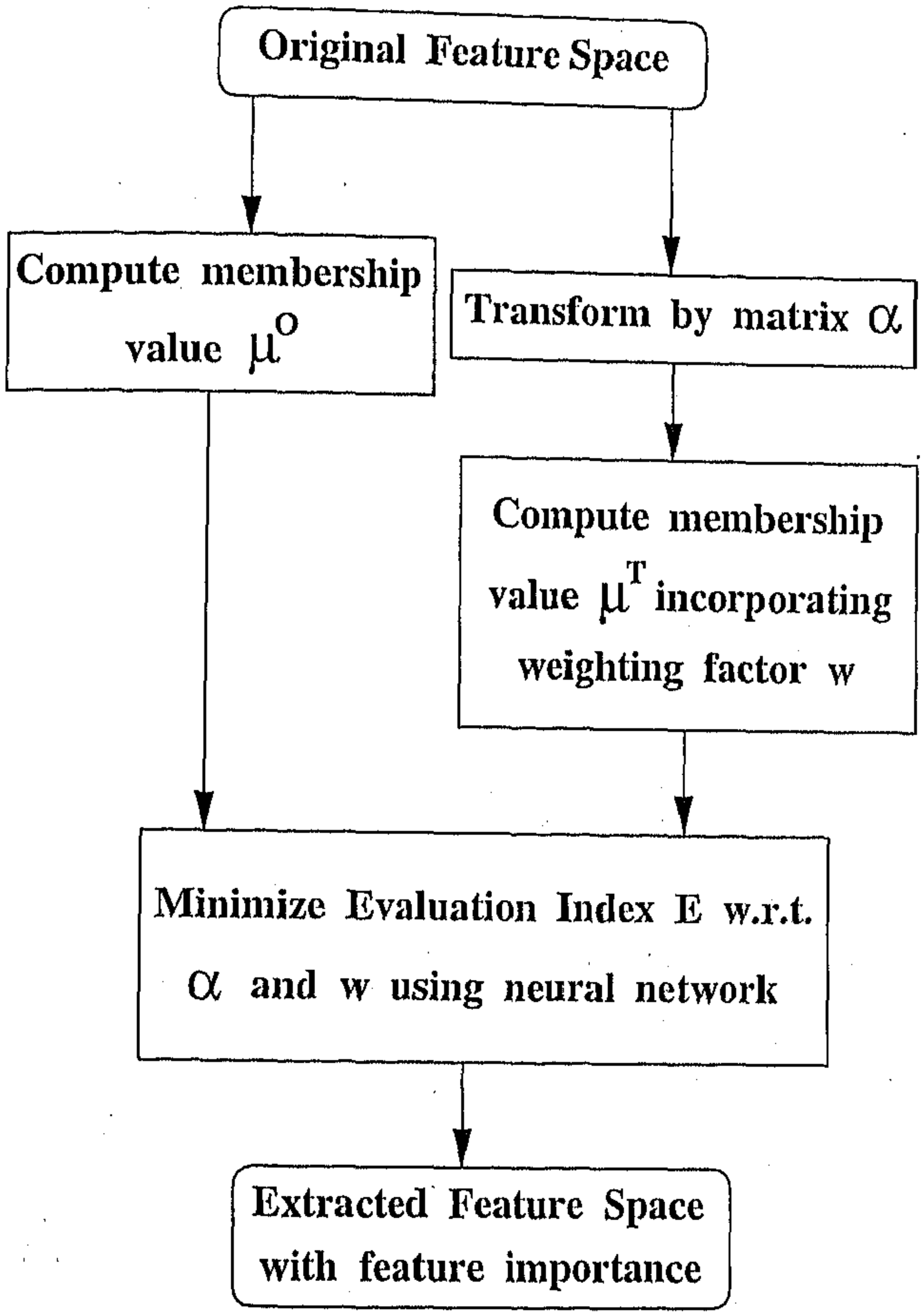

Figure 5.1: A schematic description of the neuro-fuzzy method for feature extraction.

5.2 Fuzzy Feature Evaluation Index and Weighted Membership Function

In this section the definitions of the fuzzy feature evaluation index (Eqn. (4.1)), membership function (Eqn. (4.2)) and distance (Eqn. (4.4)) between a pair of patterns, are mentioned again for the sake of convenience of readers.

The feature evaluation index (Eqn. (4.1)) for a set (Ω) of features is

$$E = \frac{2}{s(s-1)} \sum_{p} \sum_{q \neq p} \frac{1}{2} [\mu_{pq}^{T}(1-\mu_{pq}^{O}) + \mu_{pq}^{O}(1-\mu_{pq}^{T})]. \tag{5.1}$$

As mentioned in Chapter 4, the value of E decreases as the membership value representing the degree of belonging of pth and qth patterns to the same cluster in the transformed feature space tends to either 0 (when $\mu^O < 0.5$) or 1 (when $\mu^O > 0.5$). In other words, the feature evaluation index decreases as the decision on the similarity between a pair of patterns (*i.e.*, whether they lie in the same cluster or not) becomes more and more crisp. This means, if the intercluster/intracluster distances in the transformed space increase/decrease, the feature evaluation index of the corresponding set of features decreases. Therefore, our objective is to extract those features for which the evaluation index becomes minimum; thereby optimizing the decision on the similarity of a pair of patterns with respect to their belonging to a cluster.

The membership function (μ) for implementing Eqn. 5.1 is

$$\begin{aligned} \mu_{pq} &= 1 - \frac{d_{pq}}{D} \quad if\ d_{pq} \leq D, \\ &= 0, \qquad otherwise, \end{aligned} \tag{5.2}$$

where the distance d_{pq}, in the original feature space, is

$$d_{pq} = [\sum_{i} (x_{pi} - x_{qi})^2]^{\frac{1}{2}} \tag{5.3}$$

with

$$D = \beta d_{max},\ 0 < \beta \leq 1. \tag{5.4}$$

Here,

$$d_{max} = [\sum_{i} (x_{maxi} - x_{mini})^2]^{\frac{1}{2}}, \tag{5.5}$$

for the original space. The expressions for d_{pq} and d_{max}, in the transformed feature space, are provided in the next section.

It may be worth mentioning, as in Chapter 4, that μ_{pq} could be defined in a different way satisfying the above mentioned characteristics of E (Eqn. (5.1)). The computation of μ_{pq} does not require class information of the patterns.

5.3 Neural Network Model for Unsupervised Feature Extraction

In the process of feature extraction, the input feature space ($\mathbf{x}$) is transformed to $\mathbf{x}'$ by a matrix α ($= [\alpha_{ji}]_{n' \times n}$), *i.e.*,

$$\mathbf{x} \xrightarrow{\alpha} \mathbf{x}'.$$

The jth transformed feature is therefore,

$$x'_j = \sum_i \alpha_{ji} x_i, \tag{5.6}$$

where α_{ji} ($j = 1, 2, \ldots n'$, $i = 1, 2, \ldots n$ and $n > n'$) is a set of coefficients. The membership values (μ) are computed using Eqn. (4.2) based on the derived feature values. The distance d_{pq} between pth and qth patterns in the transformed space is, therefore,

$$\begin{aligned} d_{pq} &= \left[\sum_j w_j^2 \left(\sum_i \alpha_{ji}(x_{pi} - x_{qi})\right)^2\right]^{\frac{1}{2}}, \\ &= \left[\sum_j w_j^2 \left(\sum_i \alpha_{ji}\chi_i\right)^2\right]^{\frac{1}{2}}, \qquad \chi_i = x_{pi} - x_{qi}, \\ &= \left[\sum_j w_j^2 \psi_j^2\right]^{\frac{1}{2}}, \qquad \psi_j = \sum_i \alpha_{ji}(x_{pi} - x_{qi}), \end{aligned} \tag{5.7}$$

and the maximum distance d_{max} as

$$\begin{aligned} d_{max} &= \left[\sum_j \left(\sum_i |\alpha_{ji}|(x_{maxi} - x_{mini})\right)^2\right]^{\frac{1}{2}}, \\ &= \left[\sum_j \phi_j^2\right]^{\frac{1}{2}}, \qquad \phi_j = \sum_i |\alpha_{ji}|(x_{maxi} - x_{mini}). \end{aligned} \tag{5.8}$$

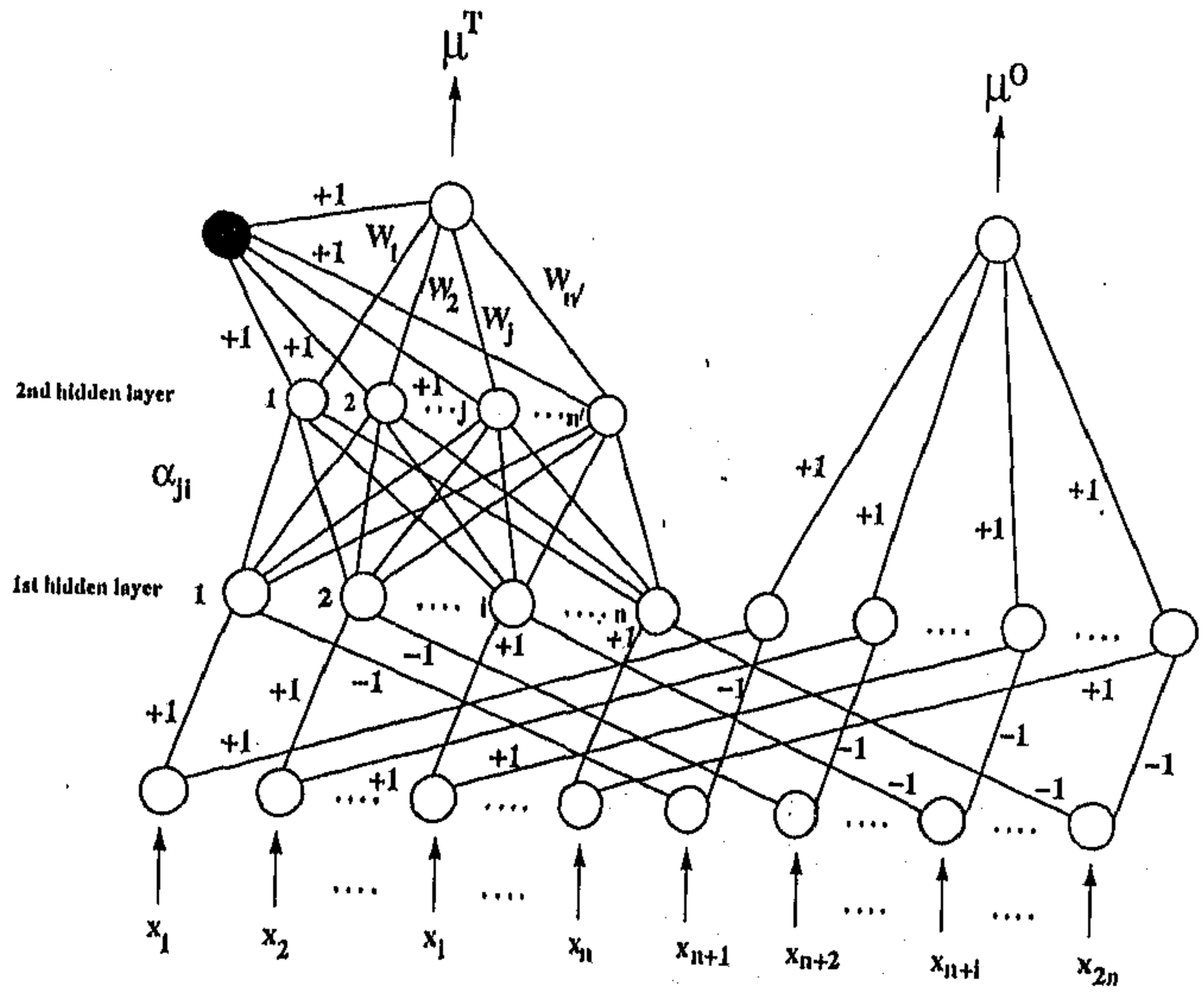

Figure 5.2: A schematic diagram of the neural network model for unsupervised feature extraction.

Weighting coefficients (w_j) representing the importance of the transformed features, make the shape of clusters in the transformed space hyperellipsoidal instead of hyperspherical.

The membership μ^T is computed using d_{pq} and d_{max} (Eqns. (4.2), (5.7) and (5.8)), while μ^O is done by Eqns. (4.2)–(5.5). The problem of feature extraction therefore reduces to finding a set of α_{ji} and w_j for which E (Eqn. (4.1)) becomes a minimum. This is schematically explained in Fig. 5.1. The task of minimization has been performed under unsupervised mode by gradient-descent technique in a connectionist framework. This is described below.

Connectionist model

The network (Fig. 5.2 consists of an input, two hidden and an output layers. The input layer consists of a pair of nodes corresponding to each feature. The first hidden layer consists of $2n$ (for n-dimensional original feature space) number of nodes. Each of the first n nodes computes the part χ_i of Eqn. (5.7) and the rest compute χ_i^2. The value of $(x_{maxi} - x_{mini})$ is stored in each of the first n nodes. The number of nodes

in the second hidden layer is taken as n', in order to extract n' number of features. Each of these nodes has two parts; one of which computing ψ_j^2 of Eqn. (5.7) and the other ϕ_j^2 of Eqn. (5.8). The output layer consists of two nodes which compute μ^T and μ^O values. There is a node (represented by black circle) in between the output node computing μ^T-values and the second hidden layer. This node computes d_{max} (Eqn. (5.8)) in the transformed feature space and sends it to the output node for computing μ^T. The value of β is stored in both the output nodes. The feature evaluation index E (Eqn. (5.19)) is computed from these μ-values off the network.

Input nodes receive activations corresponding to feature values of each pair of patterns. A j_1th node in the first hidden layer is connected to an ith $(1 \leq i \leq n)$ input node via connection weight $+1$, and to the $(i+n)$th $(1 \leq i \leq n)$ input node via connection weight -1. A j_2th node in the second hidden layer is connected to a j_1th node in the first hidden layer via connection weight $\alpha_{j_2 j_1}$. The output node computing μ^T-values is connected to a j_2th node in the second hidden layer via connection weight W_{j_2} $(= w_{j_2}^2)$, and that computing μ^O-values is connected to a j_1th $(n+1 \leq j_1 \leq 2n)$ node in the first hidden layer via connection weights $+1$ each. The node represented by the black circle is connected via weights $+1$ with the second hidden layer and also with the output node computing μ^T-values.

During training, each pair of patterns are presented to the input layer and the evaluation index is computed. The weights $\alpha_{j_2 j_1}$ and W_{j_2}s are updated using gradient-descent technique in order to minimize the index E. When pth and qth patterns are presented to the input layer, the activation produced by ith $(1 \leq i \leq 2n)$ input node is

$$v_i^{(0)} = u_i^{(0)} \tag{5.9}$$

where

$$\begin{aligned} u_i^{(0)} &= x_{pi}, \quad for\ 1 \leq i \leq n\ and \\ u_{(i+n)}^{(0)} &= x_{qi}, \quad for\ 1 \leq i \leq n. \end{aligned} \tag{5.10}$$

$u_i^{(0)}$ $(1 \leq i \leq 2n)$ is the total activation received by an ith input node. The total activation received by j_1th node in the first hidden layer (connecting ith and $(i+n)$th input nodes) is given by

$$u_{j_1}^{(1)} = 1 \times v_i^{(0)} + (-1) \times v_{i+n}^{(0)}, \quad for 1 \leq i \leq n. \tag{5.11}$$

and the activation produced by it is

$$\begin{aligned} v_{j_1}^{(1)} &= (u_{j_1}^{(1)}), \quad for\ 1 \leq j_1 \leq n, \\ &= (u_{j_1}^{(1)})^2, \quad for\ n+1 \leq j_1 \leq 2n. \end{aligned} \tag{5.12}$$

The total activation received by j_2th node in the second hidden layer is given by

$$u_{j_2}^{(2)} = \sum_{j_1} \alpha_{j_2 j_1} v_{j_1}^{(1)} \tag{5.13}$$

The activation produced by j_2th node in the second hidden layer is given by

$$v_{j_2}^{(2)} = (u_{j_2}^{(2)})^2. \tag{5.14}$$

The total activation received by the output node which computes μ^T-values is

$$u_T^{(3)} = \sum_{j_2} W_{j_2} v_{j_2}^{(2)}, \tag{5.15}$$

and that received by the other output node computing μ^O-values is

$$u_O^{(3)} = \sum_{j_2} v_{j_1}^{(2)}. \tag{5.16}$$

Therefore, $u_T^{(3)}$ and $u_O^{(3)}$ represent d_{pq}^2 as given by Eqns. (5.7) and (5.3), respectively. The activations, $v_T^{(3)}$ and $v_O^{(3)}$, of the output nodes represent μ_{pq}^T and μ_{pq}^O for pth and qth pattern pair, respectively. Thus,

$$\begin{aligned} v_T^{(3)} &= 1 - \frac{(u_T^{(3)})^{\frac{1}{2}}}{D}, \quad if\ (u_T^{(3)})^{\frac{1}{2}} \leq D \\ &= 0, \quad otherwise, \end{aligned} \tag{5.17}$$

and

$$\begin{aligned} v_O^{(3)} &= 1 - \frac{(u_O^{(3)})^{\frac{1}{2}}}{D}, \quad if\ (u_O^{(3)})^{\frac{1}{2}} \leq D \\ &= 0, \quad otherwise. \end{aligned} \tag{5.18}$$

The evaluation index, in terms of these activations, can then be expressed as (from Eqn. (4.1)) as

$$E(\alpha, \mathbf{W}) = \frac{2}{s(s-1)} \sum_p \sum_{q \neq p} \frac{1}{2} [v_T^{(3)}(1 - v_O^{(3)}) + v_O^{(3)}(1 - v_T^{(3)})]. \tag{5.19}$$

The task of minimization of $E(\alpha, \mathbf{W})$ (Eqn. (5.19)) with respect to $\alpha_{j_2 j_1}$ and W_{j_2}, for all j_1 and j_2 is performed using simple gradient-descent technique where the changes in $\alpha_{j_2 j_1}$ ($\Delta\alpha_{j_2 j_1}$) and W_{j_2} (ΔW_{j_2}) are computed as

$$\Delta\alpha_{j_2 j_1} = -\eta_1 \frac{\partial E}{\partial \alpha_{j_2 j_1}}, \forall j_1, j_2 \tag{5.20}$$

and

$$\Delta W_{j_2} = -\eta_2 \frac{\partial E}{\partial W_{j_2}}, \forall j_2, \tag{5.21}$$

where η_1 and η_2 are the learning rates.

For computation of $\frac{\partial E}{\partial \alpha_{j_2 j_1}}$ and $\frac{\partial E}{\partial w_{j_2}}$, the following expressions are used.

$$\frac{\partial E}{\partial \alpha_{j_2 j_1}} = \frac{2}{s(s-1)} \sum_p \sum_{q \neq p} \frac{1}{2} \left[1 - 2v_O^{(3)}\right] \frac{\partial v_T^{(3)}}{\partial \alpha_{j_2 j_1}}, \tag{5.22}$$

$$\begin{aligned} \frac{\partial v_T^{(3)}}{\partial \alpha_{j_2 j_1}} &= -\frac{D^{\frac{1}{2}} (u_T^{(3)})^{-\frac{1}{2}} \frac{\partial u_T^{(3)}}{\partial \alpha_{j_2 j_1}} - (u_T^{(3)})^{\frac{1}{2}} \frac{\partial D}{\partial \alpha_{j_2 j_1}}}{D^2}, \quad if\ (u_T^{(3)})^{\frac{1}{2}} \leq D \\ &= 0, \qquad\qquad otherwise, \end{aligned} \tag{5.23}$$

$$\frac{\partial u_T^{(3)}}{\partial \alpha_{j_2 j_1}} = W_{j_2} \frac{\partial v_{j_2}^{(2)}}{\partial \alpha_{j_2 j_1}}, \tag{5.24}$$

$$\frac{\partial v_{j_2}^{(2)}}{\partial \alpha_{j_2 j_1}} = 2u_{j_2}^{(2)} \frac{\partial u_{j_2}^{(2)}}{\partial \alpha_{j_2 j_1}}, \tag{5.25}$$

$$\frac{\partial u_{j_2}^{(2)}}{\partial \alpha_{j_2 j_1}} = v_{j_1}^{(1)}, \tag{5.26}$$

$$\frac{\partial D}{\partial \alpha_{j_2 j_1}} = \frac{\beta}{d_{max}} \left(\sum_i |\alpha_{j_2 i}| (x_{max i} - x_{min i})\right) (x_{max j_1} - x_{min j_1}), \tag{5.27}$$

$$\frac{\partial E}{\partial W_{j_2}} = \frac{2}{s(s-1)} \sum_p \sum_{q \neq p} \frac{1}{2} \left[1 - 2v_O^{(3)}\right] \frac{\partial v_T^{(3)}}{\partial W_{j_2}}, \tag{5.28}$$

$$\begin{aligned} \frac{\partial v_T^{(3)}}{\partial W_{j_2}} &= -\frac{\frac{1}{2} (u_T^{(3)})^{-\frac{1}{2}} \frac{\partial u_T^{(3)}}{\partial W_{j_2}}}{D}, \quad if\ (u_T^{(3)})^{\frac{1}{2}} \leq D, \\ &= 0, \qquad\qquad otherwise, \end{aligned} \tag{5.29}$$

and

$$\frac{\partial u_T^{(3)}}{\partial W_{j_2}} = v_{j_2}. \tag{5.30}$$

Algorithm for learning α and W

- Calculate d_{max} (Eqn. (5.5)) from the unlabeled training set and store it in the output node computing μ^O values. Store β (user specified) in both the output nodes.

- Initialize $\alpha_{j_2 j_1}$ and W_{j_2} with small random values in $[0, 1]$.
- Repeat until convergence, *i.e.*, until the value of E becomes less than or equal to certain predefined small quantity, or number of iterations attains certain predefined number of iterations:
  - For each pair of patterns:
    * Present the pattern pair to the input layer.
    * Compute $\Delta\alpha_{j_2 j_1}$ and ΔW_{j_2} for each j_1 & j_2, using the updating rules in Eqns. (5.20) and (5.21).
  - Update $\alpha_{j_2 j_1}$ and W_{j_2} for each j_1 & j_2 with the average values of $\Delta\alpha_{j_2 j_1}$ and ΔW_{j_2}.

After convergence, $E(\alpha, \mathbf{W})$ attains a local minimum. Then the extracted features are obtained by Eqn. (5.6) using the optimum α-values. The weights of the links connecting the output node, computing μ^T-values, to the nodes in the second hidden layer indicate the order of importance of the extracted features. Note that, the algorithm, as in Chapter 4, does not provide clustering of the feature space explicitly for extracting features.

5.4 Results

Here we demonstrate the effectiveness of the above mentioned algorithm, as in Chapters 2–4, on Iris, vowel, medical and mango-leaf data sets. As mentioned in Section 5.3, the number of nodes in the second hidden layer determines the desired number of extracted features. That is, in order to extract n' number of features, one needs to employ exactly n' nodes in the second hidden layer. For each data set, we have performed experiments for different number of nodes in the second hidden layer for finding different sets of extracted features. The particular set for which E-value is minimum in a fixed number of iterations is considered to be the best set of extracted features.

Let us consider the case of Iris data. Table 5.1 shows the values of α_{ji} (in Eqn. (5.6)) for different sets of extracted features along with their E-values. The extracted

Table 5.1: α-values corresponding to different sets of extracted features with their E-values for Iris data.

Extracted feature set containing	Coefficients (α) of				E (Eqn. (4.1))
	SL	SW	PL	PW	
one feature	0.071854	-0.028614	0.195049	0.139982	0.102437
two features	0.040649	-0.000405	0.168035	0.164546	0.099286
	-0.118670	-0.000103	-0.012020	-0.123748	
three features	-0.017140	0.005148	-0.123089	-0.152892	0.104762
	-0.003976	-0.024542	-0.005904	-0.084350	
	0.023984	-0.004368	0.237469	0.199510	

features are obtained by Eqn. (5.6). Note that, the set containing two extracted features results in minimum E-value, and therefore, is considered to be the best of all. The expressions for these two extracted features are then written, from Eqn. (5.6), as

$$I_1 = 0.040649 * SL - 0.000405 * SW + 0.168035 * PL + 0.164546 * PW$$

and

$$I_2 = -0.118670 * SL - 0.000103 * SW - 0.012020 * PL - 0.123748 * PW$$

w-values representing the importance of the features I_1 and I_2 are found to be 0.992983 and 0.744317 respectively.

Table 5.2: α-values corresponding to the best set of extracted features with their w-values for vowel data.

Extracted Features	Coefficients (α) of			w	Rank
	F_1	F_2	F_3		
V_1	-0.005676	0.050687	0.000573	0.710050	2
V_2	0.000755	-0.159839	0.000934	0.737597	1

Similarly, the dimension of the best extracted feature space is found to be 2 for vowel data, and 8 for both medical and mango-leaf data. Tables 5.2–5.4 show α and w-values for the best extracted feature sets corresponding to vowel, medical and mango-leaf data. Note that, in these experiments the values of β are found to be 0.33, 0.16, 0.25 and 0.33 for Iris, vowel, medical and mango-leaf data respectively.

Table 5.3: α-values corresponding to the best set of extracted features with their w-values for medical data.

Extracted Features	Coefficients (α) of									w	Rank
	GOT	GPT	LDH	GGT	BUN	MCV	MCH	TBil	CRTNN		
H_1	-0.193	-0.020	-0.155	-0.059	-0.081	0.096	0.135	0.193	-0.096	0.705	4
H_2	-0.046	0.097	0.035	-0.045	0.070	0.082	0.042	0.088	-0.136	0.711	1
H_3	-0.163	0.102	-0.122	-0.124	-0.155	-0.106	-0.110	0.101	-0.004	0.703	6
H_4	0.123	-0.170	-0.028	-0.107	0.142	0.043	-0.194	0.162	0.035	0.706	3
H_5	0.142	0.173	0.132	0.073	-0.045	-0.177	-0.188	-0.032	-0.030	0.705	4
H_6	-0.208	-0.003	0.083	0.102	0.013	-0.030	0.132	0.032	-0.081	0.707	2
H_7	-0.160	0.116	-0.163	0.082	-0.146	0.094	0.052	-0.142	-0.078	0.704	5
H_8	0.137	0.002	0.125	0.047	-0.078	-0.047	-0.164	0.125	0.053	0.707	2

In order to demonstrate the effectiveness of the feature extraction method, we have compared the discriminating capability of the extracted features with that of the original ones, using k-NN classifier for $k = 1, 3$ and 5. For Iris and vowel data, Tables 5.5 and 5.6 demonstrate the percentage classification using the extracted feature set and all possible subsets of the original feature set. In the case of Iris data, the recognition score using the extracted feature set is found to be greater than or equal to that obtained using any set of the original features, except for one case (e.g., the set $\{SL, SW, PL, PW\}$ with $k = 5$). Similar is the case with the vowel data, where the extracted feature pair performs better than any other set of original features, except the set $\{F_1, F_2, F_3\}$.

For medical and mango-leaf data, comparison is made only between the extracted feature set and the entire original feature set (Tables 5.7–5.8). Table 5.8 shows that the classification performance in the 8-dimensional extracted feature space of mango-leaf data is much better than that of its 18-dimensional original feature space for all values of k. Similar finding is obtained in the case of medical data, except for $k = 1$ (Table 5.7).

In a part of the experiment, the neuro-fuzzy method for feature extraction is com-

Table 5.4: α-values corresponding to the best set of extracted features with their w-values for mango-leaf data.

Extracted Features	Coefficients (α) of																		w	Rank
	Z	A	Pe	L	B	P	K	S	SI	(L+P)	L/P	L/B	(L+P)/B	A/L	A/B	A/Pe	UM/LM	UPe/LPe		
M_1	-0.171	-0.001	-0.168	-0.075	-0.079	0.095	0.135	0.192	-0.094	-0.039	0.010	0.001	0.001	0.003	0.047	-0.001	-0.003	0.002	0.710	5
M_2	-0.147	0.085	-0.133	-0.168	-0.155	-0.109	-0.111	0.100	-0.005	0.102	-0.097	-0.001	-0.070	0.024	0.004	-0.134	0.027	-0.002	0.713	4
M_3	0.126	0.141	0.126	0.100	-0.047	-0.177	-0.188	-0.031	-0.031	-0.200	0.001	0.002	0.040	0.001	0.001	0.069	-0.002	-0.001	0.708	6
M_4	-0.127	0.099	-0.200	0.098	-0.148	0.093	0.052	-0.142	-0.078	0.124	-0.001	0.021	0.001	0.003	-0.002	-0.086	-0.004	-0.002	0.716	1
M_5	-0.158	-0.065	-0.070	0.073	-0.011	-0.126	0.185	-0.170	-0.021	-0.081	-0.114	-0.001	-0.138	-0.030	0.112	-0.000	0.002	0.048	0.708	6
M_6	-0.047	0.106	0.119	0.110	-0.153	-0.164	-0.122	0.142	0.179	0.129	0.001	-0.002	-0.020	0.001	-0.127	0.042	-0.003	-0.003	0.707	7
M_7	0.084	0.116	-0.223	-0.079	0.024	-0.001	0.145	0.103	-0.096	0.096	0.002	-0.030	0.001	-0.002	0.079	-0.025	-0.003	0.003	0.714	3
M_8	-0.149	0.085	0.234	-0.089	0.166	-0.067	0.174	-0.068	0.031	-0.151	-0.020	-0.027	0.024	-0.001	0.057	-0.035	0.003	0.003	0.715	2

Table 5.5: Recognition score with k-NN classifier for different feature sets of Iris data.

Data set	Feature set	% classification		
		$k=1$	$k=3$	$k=5$
Original	$\{SL\}$	48.67	66.67	67.33
	$\{SW\}$	55.33	52.67	52.67
	$\{PL\}$	93.33	95.33	95.33
	$\{PW\}$	89.33	96.00	96.00
	$\{SL, SW\}$	74.67	76.67	76.00
	$\{SL, PL\}$	95.33	93.33	95.33
	$\{SL, PW\}$	94.67	94.00	94.00
	$\{SW, PL\}$	94.67	92.00	93.33
	$\{SW, PW\}$	90.67	94.00	94.67
	$\{PL, PW\}$	93.33	96.00	96.00
	$\{SL, SW, PL\}$	94.00	94.00	94.00
	$\{SL, SW, PW\}$	93.33	93.33	92.00
	$\{SL, PL, PW\}$	96.00	96.67	96.00
	$\{SW, PL, PW\}$	94.00	96.67	95.33
	$\{SL, SW, PL, PW\}$	95.33	96.00	96.67
Extracted	$\{I_1, I_2\}$	96.00	96.67	96.00

Table 5.6: Recognition score with k-NN classifier for different feature sets of vowel data.

Data set	Feature set	% classification		
		$k=1$	$k=3$	$k=5$
Original	$\{F_1\}$	26.52	27.21	27.21
	$\{F_2\}$	38.58	38.23	47.76
	$\{F_3\}$	26.06	33.41	33.87
	$\{F_1, F_2\}$	56.37	68.20	76.35
	$\{F_1, F_3\}$	44.32	46.84	55.80
	$\{F_2, F_3\}$	58.21	63.03	63.95
	$\{F_1, F_2, F_3\}$	78.42	81.29	82.43
Extracted	$\{V_1, V_2\}$	74.63	75.78	76.35

Table 5.7: Recognition score with k-NN classifier for extracted (obtained by the neuro-fuzzy feature extraction) and original feature sets of medical data.

Feature set	% classification		
	$k=1$	$k=3$	$k=5$
Extracted	53.92	56.34	59.89
Original	55.22	56.16	59.14

Table 5.8: Recognition score with k-NN classifier for extracted (obtained by the neuro-fuzzy feature extraction) and original feature sets of mango-leaf data.

Feature set	% classification		
	$k=1$	$k=3$	$k=5$
Extracted	85.71	88.10	92.86
Original	71.69	68.67	70.48

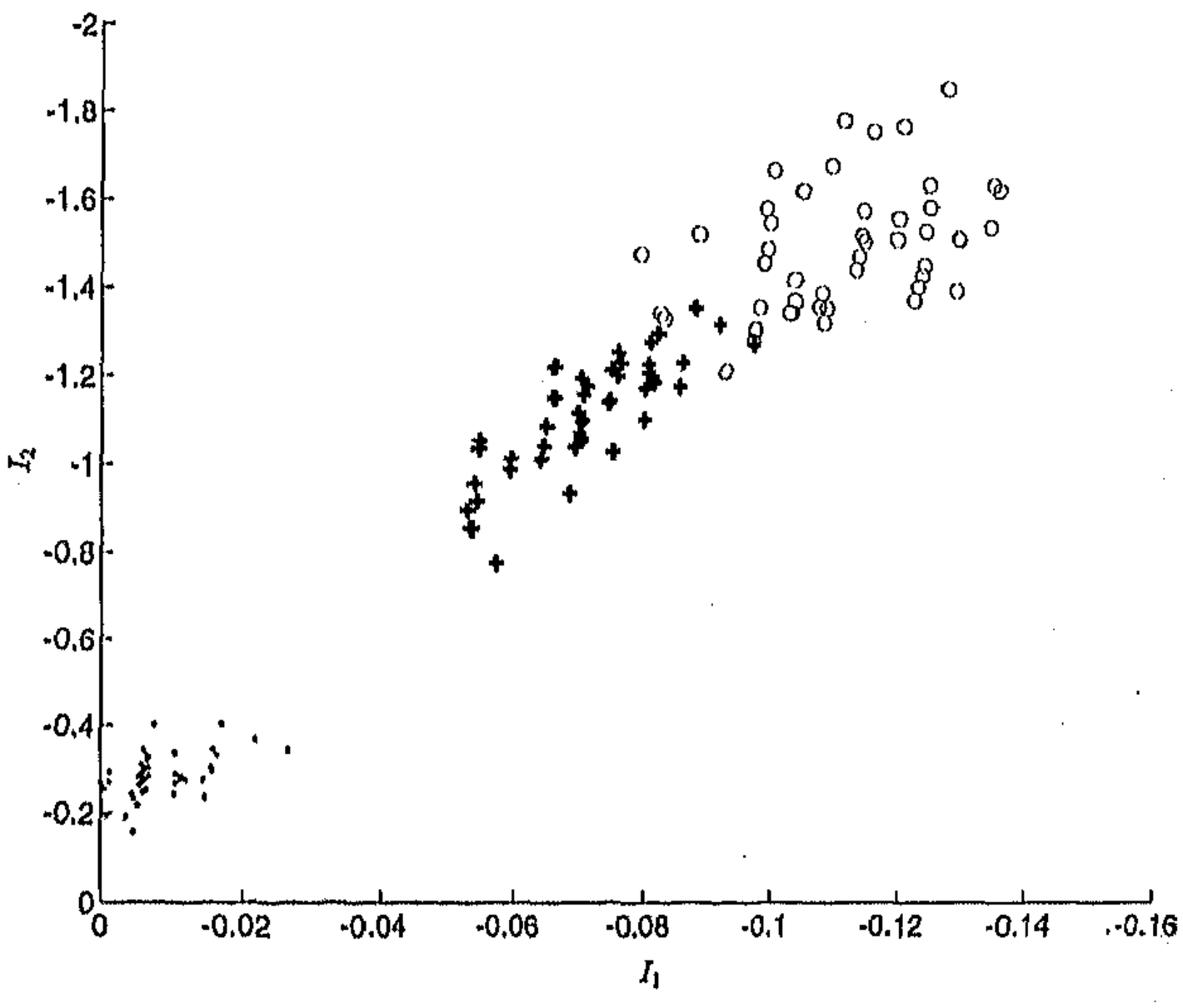

Figure 5.3: Scatter plot $I_1 - I_2$, in the extracted plane obtained by the neuro-fuzzy method, of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

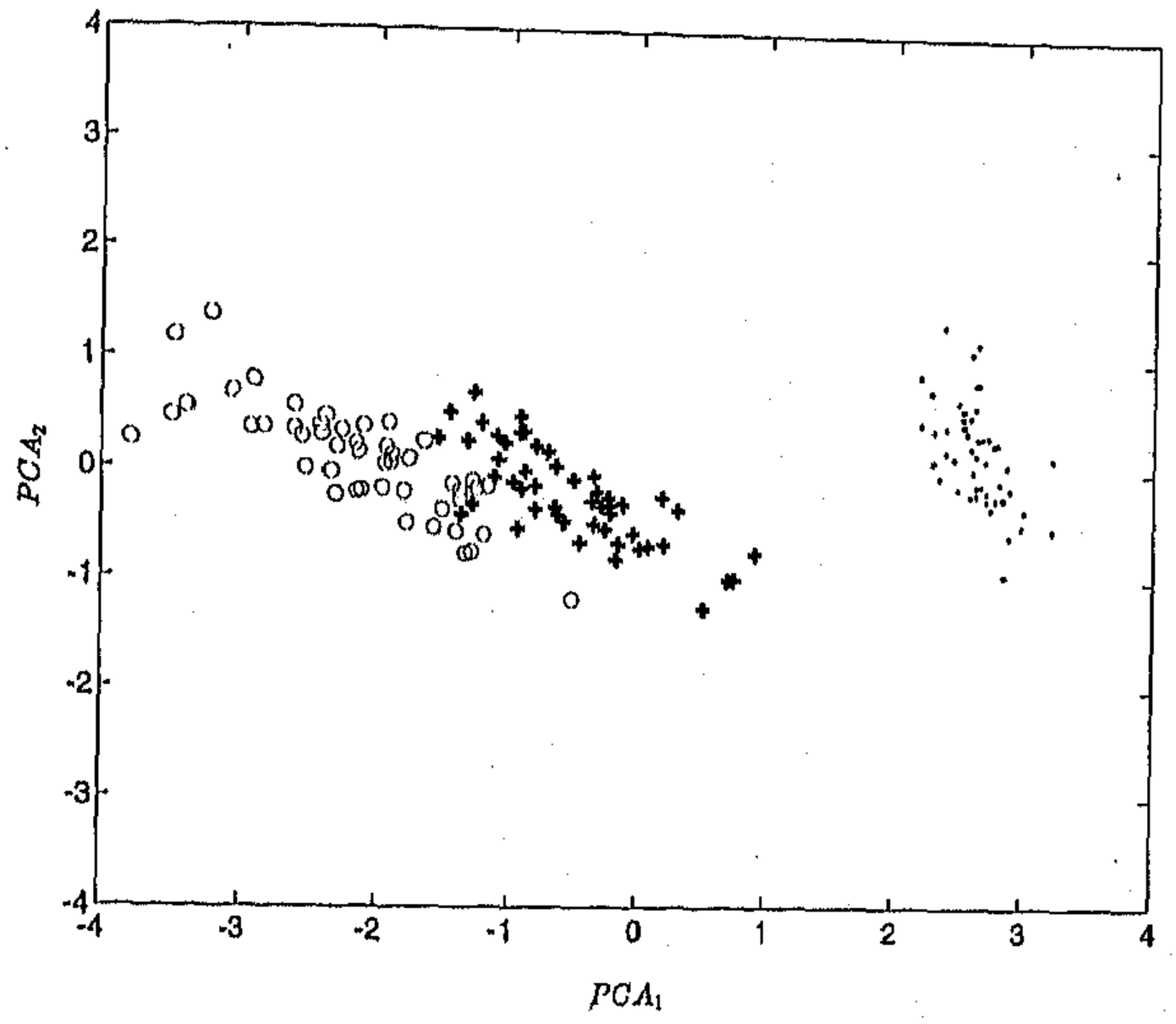

Figure 5.4: Scatter plot $PCA_1 - PCA_2$, in the extracted plane obtained by PCAN, of Iris data. Here '.', '+' and 'o' represent classes Iris Setosa, Iris Versicolor and Iris Virginica, respectively.

pared with the well-known principal component analysis in a connectionist framework, called principal component analysis network (PCAN) [190] (Appendix E). Here, we provide the comparison for Iris data only. Scatter plots in Figs. 5.3 and 5.4 show the class structures in the 2-dimensional extracted planes obtained by the neuro-fuzzy method and the PCAN respectively. The number of samples lying in the overlapping region is seen to be more for the latter case. This is also verified from the results of fuzzy c-means clustering algorithm (for $c = 3$), where the number misclassified samples (lying in other regions) is 14 for the former case, as compared to 17 in the latter case.

In order to compare the said class structures of the extracted planes (Figs. 5.3–5.4) with that of the original feature space, we provide the scatter plot for $PL - PW$ in Fig. 2.10. (Note that $\{PL, PW\}$ is known to be the best feature pair [209, 11] for Iris data.) The extracted feature plane $I_1 - I_2$ (Fig. 5.3) is seen to have more resemblance with that in Fig. 2.10, as compared to Fig. 5.4.

5.5 Conclusions and Discussion

In this chapter we have demonstrated how the concept of neuro-fuzzy computing can be exploited for developing a methodology for feature extraction under unsupervised mode. The methodology developed involves connectionist minimization of a fuzzy feature evaluation index; thereby extracting an optimum transformed feature space along with the importance of various features. The algorithm considers interdependence of the original features. The number of nodes in the second hidden layer determines the number of extracted features.

Although, the method is unsupervised, the extracted feature space has been able to provide better classification performance than the original ones, in most of the cases, for all the data sets. The extent of overlapping region in the extracted plane of the neuro-fuzzy method is less (as found by the scatter plots and fuzzy c-means algorithm) than that of the PCAN. Moreover, the neuro-fuzzy method preserves the data structure, cluster shape and inter pattern distances better than PCAN. Here we mention that the scatter plots obtained by PCAN and Sammon's nonlinear discriminant analysis (NDA) network [135] are alike.

Note that, both neuro-fuzzy method and PCAN extract features without clustering the feature space explicitly, and do not require to assume the number of clusters. It is also to be noted that the task of feature extraction by the neuro-fuzzy method involves projection of an n-dimensional original space directly to an n'-dimensional transformed space. On the other hand, in the case of PCAN, this task involves projection of the n-dimensional original space to an n-dimensional transformed space, followed by selection of the best n' number of transformed components. ♣

In Chapters 2–5 we have discussed the issue of feature evaluation. Chapter 6 provides a methodology for the design of a fuzzy case-based system in connectionist framework, for the task of classification, while Chapter 7 deals with a knowledge-based system in neuro-fuzzy paradigm, for classification and rule generation.

Chapter 6

Fuzzy Case-based Classification in Connectionist Framework

6.1 Introduction

A *case-based* system adapts old solutions to meet new demands, explains and critiques new situations using old instances (called *cases*), and performs reasoning from precedents to interpret new problems [113]. A *case* may be defined as a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the system. The system learns as a byproduct of its reasoning activity. It becomes more efficient and more competent as a result of storing the experience of the system and referring to them in later reasoning. *Case-based* system, in contrast to the traditional *knowledge-based* system, operates through a process of remembering one or a small set of concrete instances or *cases* and basing decisions on comparisons between the new situation and the old one.

The present chapter is an attempt in designing a case-based pattern classification system [44, 45] using fuzzy set theory in a connectionist framework. *Cases* are typically labeled patterns which represent different regions of the classes. Incorporation of fuzzy set theory helps in selecting the *cases* from ambiguous/overlapping regions. The concept of fuzzy similarity is used, in terms of distance measure, to determine the degree of similarity between two patterns. These tasks (e.g., computation of degree of similarity, selection of *cases* etc.) are performed in a connectionist framework whose architecture is determined adaptively through *growing* and *pruning* of nodes under supervised mode of training. Results of the algorithm, along with comparison, are effectively demonstrated on an artificial as well as the same vowel and medical data (Chapter 2).

Section 6.2 describes the methodology of the *case-based* classification. The connectionist realization of the same is discussed in Section 6.3. Experimental results are analyzed in Section 6.4. Section 6.5 describes the concluding remarks.

6.2 Case-based Classification

The methodology of our case-based pattern recognition system involves the task of *selection of few samples from each class as cases*, followed by *classification of an unknown sample* based on the *cases*. (For the sake of convenience, the

samples which are not selected as *cases*, are referred to as patterns in the subsequent discussion.) For performing these tasks, let us define a concept of similarity between a pattern and a *case* using the notion of fuzzy sets in representing a region around a *case*. The membership function characterizing this fuzzy set is such that the higher its value, the higher is the degree of similarity between a pattern and a *case*.

Let $\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ be a pattern vector of known classification in an n-dimensional feature space containing M classes. $\xi_{l_k} = [\xi_{l_k1}, \xi_{l_k2}, \ldots, \xi_{l_ki}, \ldots, \xi_{l_kn}]$ denotes l_kth *case* from kth class C_k. $\mu_{l_k}(\mathbf{x})$ is the degree of belongingness of $\mathbf{x}$ to the fuzzy set R_{l_k} representing a region with ξ_{l_k} as its center. $d_{l_k}(\mathbf{x})$ stands for the distance between $\mathbf{x}$ and ξ_{l_k}.

- Membership function denoting the degree of similarity between a pattern $\mathbf{x}$ and a *case* ξ_{l_k} is defined as

$$\begin{aligned} \mu_{l_k}(\mathbf{x}) &= 1 - 2(\tfrac{d_{l_k}(\mathbf{x})}{\lambda})^2 \quad 0 \le d_{l_k}(\mathbf{x}) < \tfrac{\lambda}{2}, \\ &= 2[1 - \tfrac{d_{l_k}(\mathbf{x})}{\lambda}]^2 \quad \tfrac{\lambda}{2} \le d_{l_k}(\mathbf{x}) < \lambda, \\ &= 0 \quad \textit{otherwise}, \end{aligned} \tag{6.1}$$

where λ is the bandwidth of the membership function, *i.e.*, the separation between its two cross-over points where $\mu_{l_k} = 0.5$. The distance $d_{l_k}(\mathbf{x})$ may be expressed in many ways. Considering Euclidian norm, we have

$$d_{l_k}(\mathbf{x}) = [\sum_{i=1}^{n}(x_i - \xi_{l_ki})^2]^{\frac{1}{2}}. \tag{6.2}$$

It is clear from Eqn. (6.1) that $\mu_{l_k}(\mathbf{x})$ decreases with the increase in $d_{l_k}(\mathbf{x})$ and *vice-versa*. It is maximum (= 1.0), if $d_{l_k}(\mathbf{x})$ is zero (*i.e.*, if a pattern $\mathbf{x}$ and the l_kth *case* are identical). The value of $\mu_{l_k}(\mathbf{x})$ is minimum (= 0.0), if $d_{l_k}(\mathbf{x}) \ge \lambda$. When $d_{l_k}(\mathbf{x}) = \frac{\lambda}{2}$, $\mu_{l_k}(\mathbf{x})$ is 0.5, *i.e.*, an ambiguous situation arises. Note that, one may define $\mu_{l_k}(\mathbf{x})$ in a different way satisfying the above mentioned characteristics.

6.2.1 Selection of *cases* and class representation

First of all, a pattern $\mathbf{x}$ is selected randomly from any class C_k. It is considered as the first *case* if the *case-base* B_k corresponding to class C_k is empty. Otherwise, its membership values $\mu_{l_k}(\mathbf{x})$ (Eqn. (6.1)) corresponding to the fuzzy sets around the

cases ξ_{l_k} in the *case-base* B_k, are computed. $\mathbf{x}$ is selected as a new *case*, if

$$\mu_{l_k}(\mathbf{x}) \leq 0.5 \; \forall \; l_k,$$

i.e., $\mathbf{x}$ does not fall within 0.5-cut of μ_{l_k}. When a *case* is selected, it is *inserted* into the *case-base*. After repeating this process over all the training patterns, a set of *cases* constituting the *case-base* for each class is obtained. The *case-base* B for the entire training set is the union of all B_ks, *i.e.*, $B = \cup_{k=1}^{M} B_k$.

After the formation of this *case-base* B, a *case* ξ_{l_k} for which $\mu_{l_k}(\mathbf{x}) \leq 0.5$ is minimum, is *deleted* from B, if the number of patterns with $\mu_{l_k}(\mathbf{x}) > 0.5$ (or with $d_{l_k}(\mathbf{x}) < \frac{\lambda}{2}$), *i.e.*, the number of patterns within 0.5-cut is less than some pre-specified number. The processes of *insertion* and *deletion* are repeated until the *case-base* becomes stable, *i.e.*, the set of *cases* does not change further. This *deletion* process reduces the possibility of a spurious pattern being considered as a *case*.

Therefore, the class C_k can be viewed as a union of all fuzzy sets R_{l_k} representing the regions around its different *cases*, *i.e.*,

$$C_k = \cup_{l_k=1}^{s_k} R_{l_k},$$

where s_k is the number of *cases* in class C_k. Note that as the value of λ increases, the extent of R_{l_k}s representing different regions around ξ_{l_k}s increases, and therefore, the number of *cases* s_k decreases.

Effect of λ

As λ increases, the extent of the region around a *case* increases, and therefore the number of *cases* required for representing a class decreases. This implies that the generalization capability of an individual *case* increases with increase in λ. Initially, although the number of *cases* decreases with the increase in λ, the generalization capability of individual *cases* dominates. For further increase in λ, the number of *cases* becomes so low that the generalization capability of the individual *cases* may not cope with the proper representation of the class structures. As a result, the recognition score decreases. In other words, the ratio between the number of correct and incorrect samples falling within the classes decreases with further increase in λ.

6.2.2 Classification

As described in Section 6.2.1, each class is modeled with a few membership functions defined around the *cases* selected from that class. The degree of belongingness of a pattern $\mathbf{x}$ to class C_k may be defined as

$$g_k(\mathbf{x}) = \max_{l_k}\{\mu_{l_k}(\mathbf{x})\} \tag{6.3}$$

or

$$g_k(\mathbf{x}) = \min_{l_k}\{\mu_{l_k}(\mathbf{x})\}, \tag{6.4}$$

using connective property, or

$$g_k(\mathbf{x}) = \frac{1}{s_k}\sum_{l_k=1}^{s_k}\mu_{l_k}(\mathbf{x}) \tag{6.5}$$

using collective property, depending on the problem.

Therefore, the decision rule is decide $\mathbf{x} \in C_k$ if

$$g_k(\mathbf{x}) = \max_t\{g_t(\mathbf{x})\},\ k,t = 1,2,\ldots M.$$

6.3 Connectionist Realization

The methodology described in Section 6.2 is implemented in a layered network whose architecture is determined adaptively through *growing* and *pruning* of hidden nodes. Note that these *growing* and *pruning* phenomena correspond to the tasks of *insertion* and *deletion* of *cases*. These are described below.

6.3.1 Architecture

The connectionist model (Fig. 6.1) consists of three layers, namely, input, hidden and output. The input layer represents the set of input features, *i.e.*, for each feature there is a node (called input node) in the input layer. Similarly, for each *case* there is a node in the hidden layer. For each hidden node, there is an auxiliary node which makes the hidden node ON or OFF. An auxiliary node corresponding to a hidden node sends back signal to the input layer only when it sends a signal to the hidden

node for making it ON. The hidden nodes are made ON one at a time keeping the remaining OFF. Finally, each node in the output layer functions as a Winner-Take-All network and represents a class.

The input nodes are connected to the hidden and auxiliary nodes by feedforward and feedback links respectively. The weight of a feedforward link connecting ith input node and l_kth hidden node is

$$w_{l_k i}^{(0)} = 1, \ \forall l_k, i. \tag{6.6}$$

The weight $w_{l_k i}^{(fb)}$ of a feedback link connecting the auxiliary node corresponding to l_kth hidden node and ith input node is the same as the ith feature value of the l_kth case ($\xi_{l_k i}$). That is,

$$w_{l_k i}^{(fb)} = \xi_{l_k i} \tag{6.7}$$

The hidden layer is connected to the output layer via feedforward links. The weight ($w_{k l_k}^{(1)}$) of the link connecting l_kth hidden node and kth output node is 1, iff the *case* corresponding to the hidden node belongs to class C_k. Otherwise, there is no such links between hidden nodes and output nodes. That is,

$$\begin{aligned} w_{k l_k}^{(1)} &= 1, \ \ if \ \xi_{l_k} \in C_k, \\ &= 0, \ \ otherwise. \end{aligned} \tag{6.8}$$

At the beginning, since the *case-base* is empty, there is no hidden node. Hence, the connectivity between the layers is not established. When there is at least one hidden node, a pattern $\mathbf{x}$ is presented to the input layer of the network. The activation of ith input node when l_kth hidden node is ON, is given by

$$v_{l_k i}^{(0)} = (u_{l_k i}^{(0)})^2. \tag{6.9}$$

$u_{l_k i}^{(0)}$ is the total input received by the ith input node when the l_kth hidden node is ON, and is given by

$$u_{l_k i}^{(0)} = x_{pi} - u_{l_k i}^{(fb)} \tag{6.10}$$

where $u_{l_k i}^{(fb)} = (-1) * w_{l_k i}^{(fb)}$ (-1 being the feedback activation value of the auxiliary node corresponding to the l_kth hidden node) is the feedback input received by the input node. The total input received by the l_kth hidden node when it is made ON, is

$$u_{l_k}^{(1)} = \sum_i v_{l_k i}^{(0)} * w_{l_k i}^{(0)}. \tag{6.11}$$

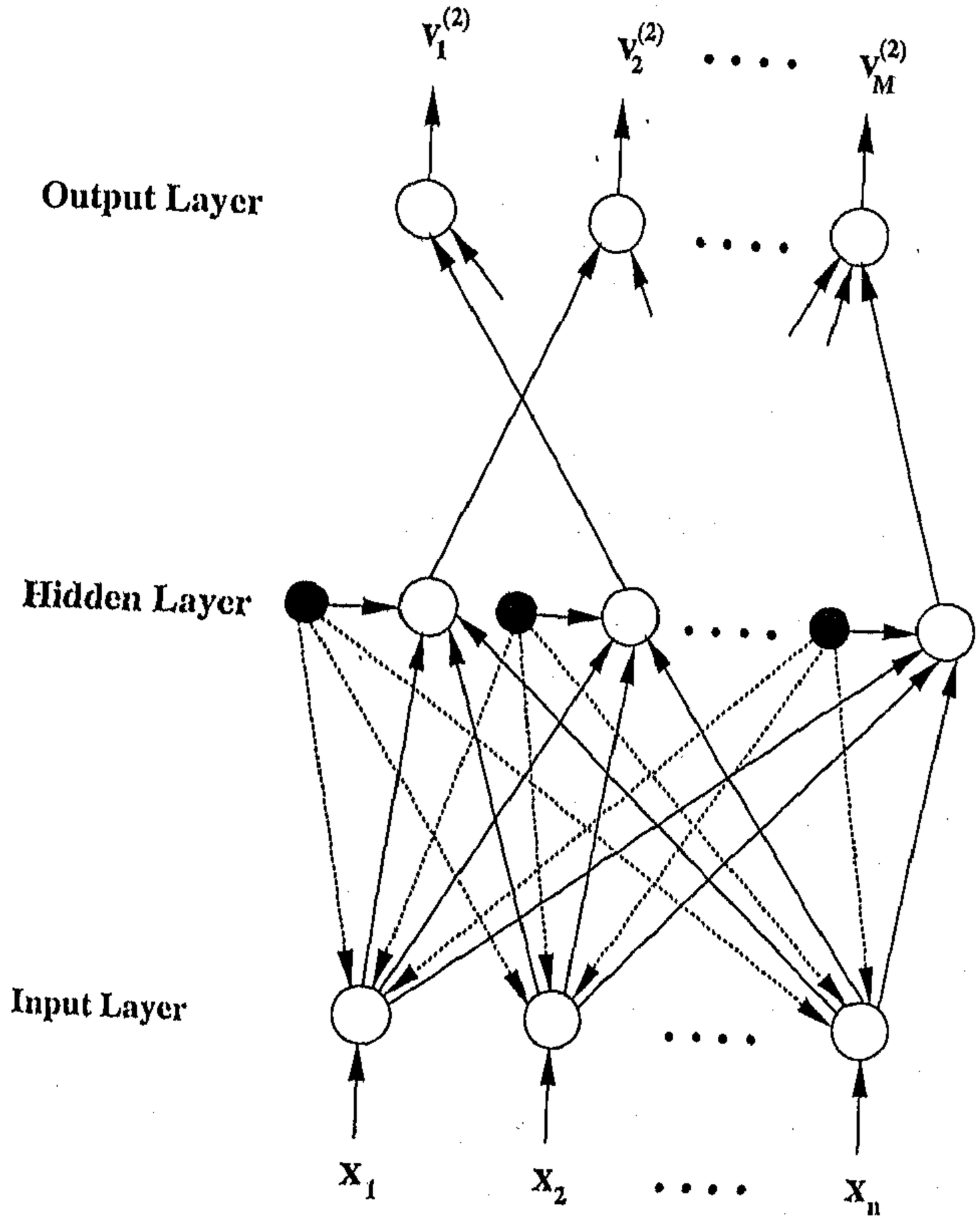

Figure 6.1: A schematic diagram of the neural network model for case-based classification. Black circles represent the auxiliary nodes, and white circles represent input, hidden and output nodes.

The activation function of an l_kth hidden node is the same as $\mu_{l_k}(\mathbf{x})$ (Eqn. (6.1)). Thus, the activation $(v_{l_k}^{(1)})$ of l_kth hidden node is given by

$$\begin{aligned} v_{l_k}^{(1)} &= 1 - 2(\frac{(u_{l_k}^{(1)})^{\frac{1}{2}}}{\lambda^2}), \quad 0 \leq (u_{l_k}^{(1)})^{\frac{1}{2}} < \frac{\lambda}{2}, \\ &= 2[1 - \frac{(u_{l_k}^{(1)})^{\frac{1}{2}}}{\lambda}]^2, \quad \frac{\lambda}{2} \leq (u_{l_k}^{(1)})^{\frac{1}{2}} < \lambda, \\ &= 0, \qquad\qquad \textit{otherwise.} \end{aligned} \tag{6.12}$$

Here the value of λ is stored in all the hidden nodes. A kth output node receives activations only from the hidden nodes corresponding to the *cases* in C_k. That is, the activation received by the kth output node from the l_kth hidden node is

$$u_{kl_k}^{(2)} = v_{l_k}^{(1)} * w_{kl_k}^{(1)}. \tag{6.13}$$

The activation function of kth output node has the same form as in Eqn. (6.3). That is,

$$v_k^{(2)} = \max_{l_k}\{u_{kl_k}^{(2)}\}. \tag{6.14}$$

Note that here we have considered maximum of $u_{kl_k}^{(2)}$s as the activation value of the output node. However, minimum (Eqn. (6.4)) or average (Eqn. (6.5)) value could have also been used for this purpose depending upon the problem.

6.3.2 Training and formation of the network

The network whose architecture is described in Section 6.3.1, is formed through *growing* and *pruning* of the hidden nodes during the training phase under supervised mode. Initially there is only input and output layers. The patterns are presented in a random sequence to the input layer of the network. The first pattern that is presented to the network, is considered as a *case*. A hidden node along with its auxiliary node representing this *case* is added to the network. The connections of these auxiliary and hidden nodes with the input and output layers are established as described by Eqns. (6.6)–(6.8).

For the remaining patterns, their degrees of similarity with the *cases* represented by existing hidden nodes are computed, and if they are decided to be new *cases* (Section 6.2.1), hidden nodes are added through *growing* operation. After the process of addition is over, it is checked if there is any redundant hidden node. This is done

through *pruning* operation depending on the criterion mentioned in Section 6.2.1. In this connection, one may note the as λ increases, the number of *cases* and hence the number of hidden nodes decreases. These two operations, which together constitute a single iteration, are described below. These iterations are continued until the structure of the network becomes stable, *i.e.*, till the set of hidden nodes representing the *cases* does not change.

Growing of hidden nodes :

For a pattern $\mathbf{x} \in C_k$, if $v_{l_k}^{(1)} \leq 0.5$ and $\mathbf{w}_{l_k}^{(lb)} = \xi_{l_k} \in C_k$ for all the hidden nodes, $\mathbf{x}$ is selected as a *case*. A hidden node along with its auxiliary node is added to the network for representing this *case* and the links are established accordingly, using Eqns. (6.6)–(6.8). This process is called *growing of hidden nodes*. Note that the task 'insertion' of *cases* described in Section 6.2.1, is performed through this process.

Pruning of hidden nodes :

An l_kth hidden node is deleted, if

$$v_{l_k}^{(1)} = \min_{\xi_{l_k}=\mathbf{w}_{l_k}^{(lb)} \in C_k} v_{l_k}^{(1)} \leq 0.5,$$

and the number of training samples for which $v_{l_k}^{(1)} > 0.5$ is less than a pre-defined value. In this way, the network is pruned. Note that the task 'deletion' of *cases* described in Section 6.2.1, is performed through this process. ♣

During testing, an unknown pattern $\mathbf{x}$ is said to be in class C_k if the value of $v_k^{(2)}$ is the maximum of all such activation values of the output nodes.

6.4 Experimental Results

In this section we demonstrate the effectiveness of the *case-based* classification system, along with comparison, on an artificially generated data Pat3, and the same vowel and medical data (Chapter 2). Pat3 (Fig. 6.2) has two input features and two classes, and consists of 557 pattern points. In all the cases, the data set has been divided into two subsets – *training* and *testing*. While *perc*% samples are considered during training, the remaining (100 – *perc*)% is used for testing.

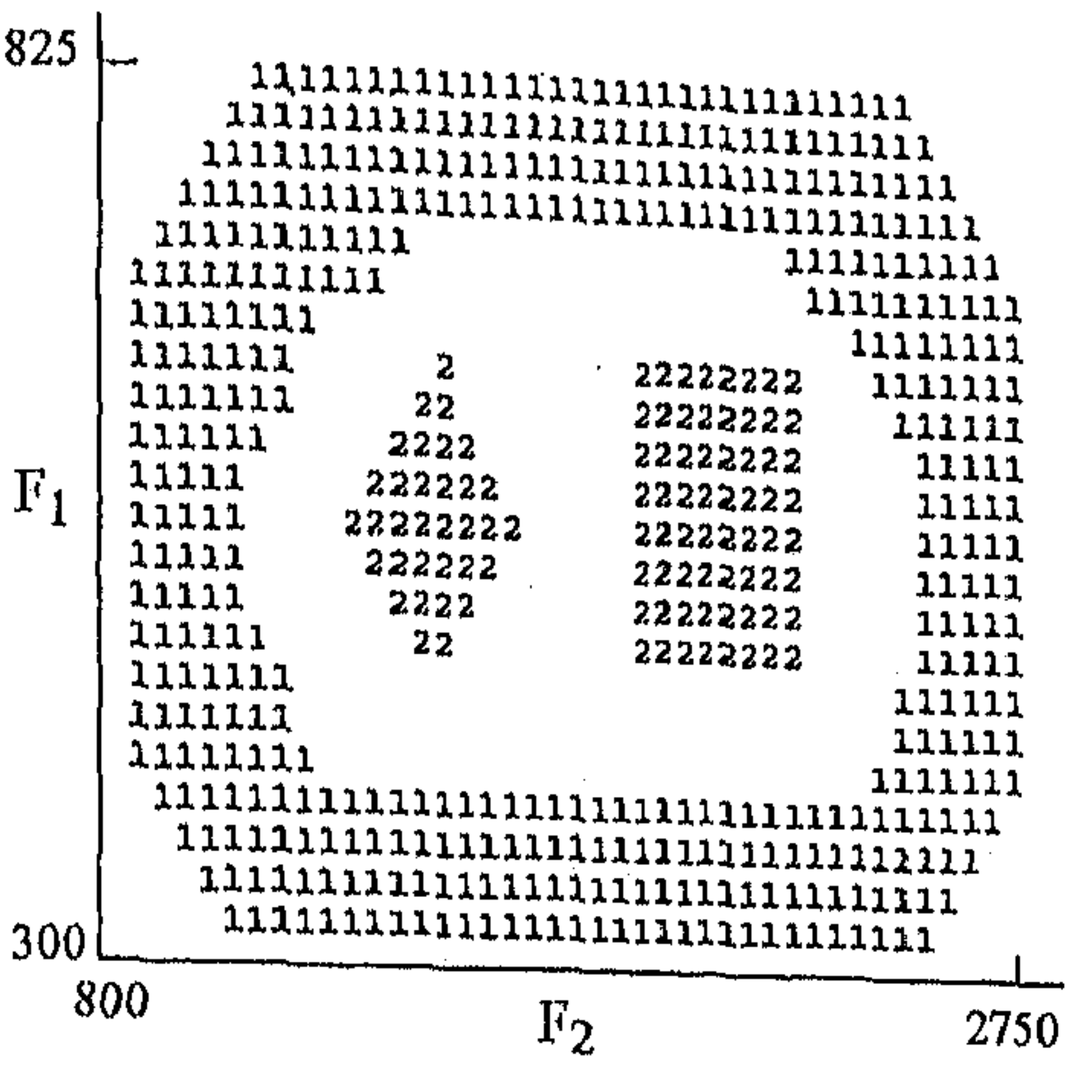

Figure 6.2: Scatter plot of the artificially generated Pat3 data. Here '1' and '2' represent patterns in classes 1 and 2 respectively.

Tables 6.1–6.3 depict some of the results obtained with the above data sets for different values of λ when $perc = 30$ is considered. The first column of these tables indicates the number of iteration(s) required by the network until it stabilizes during training. It is found from these tables that the recognition scores on the training set, as expected, are higher than those on the test set. The recognition score during training decreases with the increase in the value of λ. On the other hand, for the test data, the recognition score increases with λ up to a certain value, beyond which it decreases. This can be explained as follows.

During training, the recognition score increases with decrease in λ due to better abstraction capability. While for the test data, as λ decreases, the modeling of class structures improves because of the increase in the number of *cases*, and therefore, the recognition score increases upto a certain value of λ. Beyond that, as mentioned in Section 6.2.1, the number of *cases* with poor generalization capability (*i.e.*, membership functions with very small bandwidth) increases. As a result, the recognition score decreases due to *overlearning*.

As mentioned in Section 6.3.2, the number of hidden nodes of the network decreases

Table 6.1: Classification performance for different λ using Pat3 for $perc = 30$.

Number of iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
1	150.0	1	54	100.0	100.0
		2	12	100.0	100.0
		Overall	66	100.0	100.0
1	200.0	1	45	100.0	100.0
		2	11	100.0	100.0
		Overall	56	100.0	100.0
1	250.0	1	29	100.0	100.0
		2	7	100.0	100.0
		Overall	36	100.0	100.0
1	300.0	1	23	100.0	99.69
		2	6	100.0	94.12
		Overall	29	100.0	98.72
1	350.0	1	16	98.91	97.55
		2	5	89.47	87.18
		Overall	21	97.30	95.74

Table 6.2: Classification performance for different λ using vowel data for $perc = 30$.

Number of Iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
3	100.0	ɲ	21	95.24	41.18
		a	22	100.0	84.13
		i	42	98.04	72.73
		u	31	97.78	81.13
		e	53	98.39	67.59
		o	38	94.44	80.68
		Overall	207	97.30	75.0
3	150.0	ɲ	18	95.24	64.71
		a	13	96.15	93.65
		i	23	96.08	86.78
		u	20	88.89	86.79
		e	37	96.77	80.0
		o	26	92.59	85.71
		Overall	137	94.21	83.82
3	200.0	ɲ	16	80.95	64.71
		a	13	92.31	90.48
		i	21	98.04	87.60
		u	10	91.11	85.85
		e	36	93.55	81.38
		o	25	90.74	86.51
		Overall	130	92.28	83.09
1	250.0	ɲ	12	71.43	58.82
		a	9	88.46	80.95
		i	11	92.16	85.95
		u	9	84.44	72.64
		e	20	91.04	80.69
		o	14	81.48	74.60
		Overall	75	86.49	77.29
1	300.0	ɲ	10	57.14	52.94
		a	8	92.31	80.95
		i	10	92.16	86.78
		u	8	97.78	83.96
		e	20	88.71	80.69
		o	11	64.81	59.52
		Overall	67	83.78	75.82
3	350.0	ɲ	8	52.38	52.94
		a	7	92.31	95.24
		i	9	94.12	90.08
		u	8	97.78	89.62
		e	13	70.97	66.21
		o	8	46.30	42.86
		Overall	53	75.68	72.06
1	400.0	ɲ	8	57.14	56.86
		a	7	96.15	95.24
		i	7	88.24	86.78
		u	6	97.78	84.91
		e	10	60.35	65.52
		o	8	72.22	64.29
		Overall	46	80.31	75.16
1	450.0	ɲ	7	71.43	70.59
		a	5	84.62	68.25
		i	5	58.82	61.16
		u	6	93.33	83.02
		e	9	83.87	76.55
		o	6	68.52	67.46
		Overall	38	76.45	71.41

Table 6.3: Classification performance for different λ using the medical data for $perc = 30$.

Number of Iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
1	150.0	ALD	17	61.76	32.93
		PH	30	81.13	48.80
		LC	19	91.89	73.56
		C	13	42.86	27.71
		Overall	79	71.07	46.42
1	160.0	ALD	20	71.43	56.79
		PH	35	70.37	29.84
		LC	17	78.95	66.28
		C	8	58.33	57.32
		Overall	80	69.94	50.13
1	170.0	ALD	20	71.43	58.02
		PH	34	68.52	29.03
		LC	17	78.95	66.28
		C	8	55.56	54.88
		Overall	79	68.71	49.60
1	180.0	ALD	20	68.57	56.79
		PH	34	72.22	31.45
		LC	16	71.05	61.63
		C	8	55.56	54.88
		Overall	78	67.48	49.06
1	190.0	ALD	19	80.00	61.73
		PH	33	77.78	36.29
		LC	14	76.32	68.60
		C	6	8.33	12.20
		Overall	72	62.58	43.97
7	200.0	ALD	15	76.47	58.54
		PH	25	71.70	45.60
		LC	14	72.97	68.97
		C	11	25.71	9.64
		Overall	137	62.89	45.89

Table 6.4: Classification performance for different λ and *perc* on vowel data.

λ	Recognition score (%)						
	$perc = 10$	$perc = 20$	$perc = 30$	$perc = 40$	$perc = 50$	$perc = 60$	$perc = 70$
150.0	70.99	81.55	83.82	80.42	84.67	86.29	87.01
200.0	75.70	82.12	83.99	83.27	84.67	85.71	86.20
250.0	75.06	80.11	77.29	80.23	82.38	83.43	84.03

with the increase in λ, for all the cases (Tables 6.1–6.3). Since class 1 of Pat3 is more sparse than class 2 (Fig. 6.2), it needs more *cases* (and hence more hidden nodes) for its representation. This is reflected in Table 6.1. Similar observations hold good for vowel and medical data where class 'e' for vowel and PH for medical data, being most sparse, have the maximum number of hidden nodes. Note from Tables 6.1–6.3 that, for all the data sets, the stability of the architecture of the networks is achieved with a very few iterations.

In order to demonstrate the effect of the size of a training set on the performance of the network, we have considered only the vowel data. Different values of *perc* considered are 10, 20, 30, 40, 50, 60 and 70 with $\lambda = 150.0$, 200.0 and 250.0. (Note that the network achieves the best generalization capability for $\lambda = 200.0$ (Table 6.2).) Table 6.4 shows that the recognition score on the test set, as expected, increases in general with the size of the training set.

In a part of the experiment, the performance of the *case-based* system (CBNN) is compared with those of k-NN and Bayes maximum likelihood classifiers. The results are shown for $perc = 30$. k-NN algorithm is executed taking k equal to $\sqrt{s}$, where s is the number of training samples. It is known that as s goes to infinity, if the values of k and k/s can be made to approach infinity and zero respectively, then the k-NN classifier approaches the optimal Bayes classifier [35, 59]. One such value of k for which the limiting conditions are satisfied is $\sqrt{s}$. For the Bayes maximum likelihood classifier, we assume a multivariate normal distribution of the samples with different dispersion matrices and *a priori* probabilities ($= \frac{s_i}{s}$, for s_i patterns from class i) for different classes.

Table 6.5: *Comparative recognition score of various classifiers on different data sets.*

Data set	Class	Recognition score (%)					
		CBNN		Bayes		k-NN	
		Training	Testing	Training	Testing	Training	Testing
Pat3	1	100.0	100.0	100.0	100.0	100.0	99.38
	2	100.0	100.0	34.48	19.12	96.55	100.0
	Overall	100.0	100.0	88.62	85.90	99.40	99.49
vowel	∂	80.95	64.71	38.10	43.14	23.81	33.33
	a	92.31	90.48	88.46	85.71	80.77	85.71
	i	98.04	87.60	90.20	85.12	88.24	85.12
	u	91.11	85.85	91.11	90.57	86.67	76.42
	e	93.55	81.38	75.81	80.69	75.81	77.93
	o	90.74	86.51	92.59	85.71	92.59	88.89
	Overall	92.28	83.99	83.01	81.70	79.92	78.43
medical	ALD	71.43	56.79	61.76	50.00	52.94	46.34
	PH	70.37	29.84	54.72	64.80	69.81	77.60
	LC	78.95	66.28	51.35	36.78	21.62	29.89
	C	58.33	57.32	91.43	75.90	54.29	61.45
	Overall	69.94	50.13	63.52	57.56	51.57	56.23

Table 6.5 depicts that CBNN performs better than both k-NN and Bayes maximum likelihood classifiers for Pat3 and vowel data. In the case of medical data, while the performance of CBNN on the training set is better than those obtained by the other two classifiers, the reverse is true on the test samples.

6.5 Conclusions and Discussion

We have described a fuzzy case-based system for pattern classification in connectionist framework. Different *cases* have been selected during training of the network, and are stored as its parameters. The architecture of the network is determined adaptively through *growing* and *pruning* of hidden nodes.

The number of hidden nodes increases with the decrease in extent λ (Eqn. 6.1) of the fuzzy region around a *case*. As λ decreases, the performance during training increases because of the higher number of representative *cases*. On the other hand, during testing, it increases with the decrease in λ upto a certain value, beyond which the performance deteriorates because of *overlearning* (poor generalization capability of the *cases*).

It has been found that the *case-based system* performs better than both k-NN and Bayes maximum likelihood classifiers for Pat3 and vowel data. However, for medical data, the generalization capability of CBNN is seen to be poorer than k-NN and Bayes classifiers.

Chapter 7

Knowledge-Based Fuzzy MLP for Classification and Rule Generation

7.1 Introduction

A knowledge-based system considers problem-specific domain knowledge for determining optimal solution to the problem. The domain knowledge is extracted from the given data set/objects and is injected to the system. The system then finds the solution to the problem as well as refines the knowledge injected to it during the course of attaining the solution. This process helps in reducing the searching space and time while the system traces for the optimal solution.

In this chapter we consider a new idea of knowledge encoding among the connection weights of a fuzzy MLP [167] [41, 42, 142]. The methodology involves development of a technique for generating an appropriate architecture of the fuzzy MLP [167] in terms of hidden nodes and links. To demonstrate its significance an application to pattern classification and rule generation has been provided. The model is capable of generating both *positive* (indicating the belongingness of a pattern to a class) and *negative* rules (indicating *not* belongingness of a pattern to a class) in linguistic form to justify any decision reached. This is found to be useful for inferencing in ambiguous cases. Note that the rule generation procedures described in this chapter are different from that reported in [146]. The model is capable of handling input in numerical, linguistic and set forms, and can tackle uncertainty due to overlapping classes. The knowledge encoding procedure, unlike most other methods [57, 215], involves a non-binary weighting mechanism.

It is found that the classification performance improves appreciably with the encoding of the initial knowledge in the network architecture. The system converges much earlier and hence more meaningful rules are generated at this stage as compared to the other models.

The effectiveness of the knowledge-based model is demonstrated on the same artificially generated Pat3 data (Chapter 6), as well as the real life vowel and medical data (Chapter 2). The models are compared with conventional and fuzzy multilayer perceptrons (Appendices C and F) [167] and fuzzy min-max network (Appendix G) [206].

Section 7.2 describes the method of knowledge encoding into a fuzzy MLP. The tasks of link pruning/node growing are also described in this regard. The techniques for

rule generation are provided in Section 7.3. Experimental results are analyzed in Section 7.4. Concluding remarks are mentioned in Section 7.5.

7.2 Knowledge-based Classification

In this section, we formulate a methodology for encoding *a priori* initial knowledge in the *fuzzy* MLP [167]. Our concept is based on the fact that if a classifier is initially provided with some knowledge from the data set, the resulting searching space is reduced; thereby leading to a more efficient learning. The architecture of the network may become simpler due to the inherent reduction of the redundancy among the connection weights. The network topology is then refined using the training data. Scope for growing hidden nodes and pruning links, when necessary (as determined by the network performance), enables the generation of a near optimal network architecture with improved classification performance. We now describe the input and output representations of the system followed by the knowledge encoding procedure.

7.2.1 Input representation

An n-dimensional pattern $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$, as in the case of fuzzy MLP (Appendix F) [167], is represented as a $3n$-dimensional input vector [159] which is given by

$$\left[u_1^{(0)}, u_2^{(0)}, \ldots, u_{3n}^{(0)}\right] = \left[\mu_{low(\mathrm{x}_1)}(\mathbf{x}), \mu_{medium(\mathrm{x}_1)}(\mathbf{x}), \mu_{high(\mathrm{x}_1)}(\mathbf{x}), \ldots, \mu_{high(\mathrm{x}_n)}(\mathbf{x})\right], \quad (7.1)$$

where μ-values indicate the degree of belongingness of $\mathbf{x}$ to the corresponding linguistic π-sets *low*, *medium* or *high*, along each feature axis. $u_i^{(0)}$ represents the input received by ith input node of the knowledge-based system.

As in the case of fuzzy MLP, we use one dimensional π-type membership function for numerical input feature, *i. e.*,

$$\pi(x_i; c, \lambda) = \begin{cases} 2\left(1 - \frac{\|x_i - c\|}{\lambda}\right)^2, & \text{for } \frac{\lambda}{2} \leq \|x_i - c\| \leq \lambda \\ 1 - 2\left(\frac{\|x_i - c\|}{\lambda}\right)^2, & \text{for } 0 \leq \|x_i - c\| \leq \frac{\lambda}{2} \\ 0, & \text{otherwise} \end{cases} \quad (7.2)$$

where $\lambda > 0$ is the radius of the π-function with c as the central point.

The central points c (*i.e.*, c_{i_l}, c_{i_m} and c_{i_h} corresponding to feature x_i) and the radii λ (*i.e.*, λ_{i_l}, λ_{i_m} and λ_{i_h} corresponding to feature x_i) of the π-functions corresponding to the fuzzy sets *low*, *medium* and *high*, as in Appendix F, are computed from the training set. Let x_{imax} and x_{imin} denote the upper and lower bounds of numeric feature x_i for all the pattern points (in the training set). If m_i, m_{i_l} and m_{i_h} represent mean values of ith component of the patterns for which $x_i \in [x_{imin}, x_{imax}]$, $x_i \in [x_{imin}, m_i)$ and $x_i \in (m_i, x_{imax}]$ respectively, then

$$\begin{aligned} c_{i_l} &= m_{i_l} \\ c_{i_m} &= m_i \\ c_{i_h} &= m_{i_h} \end{aligned} \tag{7.3}$$

and

$$\begin{aligned} \lambda_{i_l} &= 2\,(c_{i_m} - c_{i_l}) \\ \lambda_{i_h} &= 2\,(c_{i_h} - c_{i_m}) \\ \lambda_{i_m} &= fnos * \frac{\lambda_{i_l}\,(x_{imax} - c_{i_m}) + \lambda_{i_h}\,(c_{i_m} - x_{imin})}{x_{imax} - x_{imin}}. \end{aligned} \tag{7.4}$$

fnos is a multiplicative parameter controlling the extent of the overlapping.

7.2.2 Output representation

The output of a node in any layer $(h+1)$ other than the input layer, as in the case of the fuzzy MLP [167], is given by

$$v_j^{(h+1)} = \frac{1}{1 + exp\left(-\sum_i v_i^{(h)} w_{ji}^{(h)}\right)}, \tag{7.5}$$

where $v_i^{(h)}$ is the activation of ith node in the preceding hth layer and $w_{ji}^{(h)}$ is the weight of the connection from ith node in layer h to jth node in layer $(h+1)$. For nodes in the input layer, $v_i^{(0)}$ corresponds to ith component of the input vector.

The degree of belongingness of $\mathbf{x}$ to the various M classes is

$$\mu_k(\mathbf{x}) = \frac{1}{1 + \left(\frac{z_k}{f_d}\right)^{f_e}}, \tag{7.6}$$

where

$$z_k = \sqrt{\sum_{i=1}^{n} \left[\frac{x_i - o_{ki}}{v_{ki}}\right]^2} \text{ for } k = 1, \ldots, M, \tag{7.7}$$

with positive constants f_d and f_e being the denominational and exponential fuzzy generators controlling the amount of fuzziness. $\mathbf{o_k} = [o_{k1} \ldots o_{kn}]$ and $\mathbf{v_k} = [v_{k1} \ldots v_{kn}]$ denote the mean and standard deviation respectively, of the numerical training data in kth class C_k. Note that $\mu_k(\mathbf{x})$ is taken as the desired output of kth output node during its backpropagation training.

7.2.3 Knowledge encoding

Let an interval $[x_{i_1}, x_{i_2}]$ denote the range of feature x_i covered by class C_k. Then we denote the membership value of the interval as $\mu([x_{i_1}, x_{i_2}])$ $(= \mu(between\ x_{i_1}\ and\ x_{i_2}))$ and compute it as [159]

$$\mu(between\ x_{i_1}\ and\ x_{i_2}) = \{\mu(greater\ than\ x_{i_1}) * \mu(less\ than\ x_{i_2})\}^{\frac{1}{2}} \tag{7.8}$$

where

$$\begin{aligned} \mu(greater\ than\ x_{i_1}) &= \{\mu(x_{i_1})\}^{\frac{1}{2}} \quad if\ x_{i_1} \leq c_{prop} \\ &= \{\mu(x_{i_1})\}^2 \quad otherwise \end{aligned} \tag{7.9}$$

and

$$\begin{aligned} \mu(less\ than\ x_{i_2}) &= \{\mu(x_{i_2})\}^{\frac{1}{2}} \quad if\ x_{i_2} \geq c_{prop} \\ &= \{\mu(x_{i_2})\}^2 \quad otherwise. \end{aligned} \tag{7.10}$$

Here c_{prop} denotes c_{i_l}, c_{i_m} and c_{i_h} for each of the corresponding three overlapping fuzzy sets *low*, *medium* and *high* of Eqn. (7.4). The output membership for the corresponding class C_k is found using Eqns. (7.6) and (7.7), where x_i of Eqn. (7.7) is replaced by the mean of the interval $[x_{i_1}, x_{i_2}]$ of ith feature.

We have also considered the intervals in which a class is *not* included. The complement of the interval $[x_{i_1}, x_{i_2}]$ of feature x_i is the region where class C_k does not lie and is defined as $[x_{i_1}, x_{i_2}]^c$ (where S^c denotes the complement of S). The linguistic membership values for $[x_{i_1}, x_{i_2}]^c$ is denoted by $\mu([x_{i_1}, x_{i_2}]^c)$ $(= \mu(not\ between\ x_{i_1}\ and\ x_{i_2}))$ and is calculated as

$$\mu(not\ between\ x_{i_1}\ and\ x_{i_2}) = max\{\mu(less\ than\ x_{i_1}), \mu(greater\ than\ x_{i_2})\}, \tag{7.11}$$

since $not\ between\ x_{i_1}\ and\ x_{i_2} \equiv less\ than\ x_{i_1}\ OR\ greater\ than\ x_{i_2}$.

Let the linguistic membership values for class C_k in the interval $[x_{i_1}, x_{i_2}]$, as calculated by Eqns. (7.8)-(7.10), be $\{\mu_L([x_{i_1}, x_{i_2}]), \mu_M([x_{i_1}, x_{i_2}]), \mu_H([x_{i_1}, x_{i_2}])\}$. Similarly for the complement of the interval, using Eqn. (7.11), we have

$$\{\mu_L([x_{i_1}, x_{i_2}]^c), \mu_M([x_{i_1}, x_{i_2}]^c), \mu_H([x_{i_1}, x_{i_2}]^c)\}.$$

A fuzzy MLP (Appendix F) [167] with only one hidden layer is considered, taking two hidden nodes corresponding to $[x_{i_1}, x_{i_2}]$ and its complement respectively. Links are introduced between the input nodes and the corresponding nodes in the hidden layer

$$iff \;\; \mu_A([x_{i_1}, x_{i_2}]) \; or \; \mu_A([x_{i_1}, x_{i_2}]^c) \geq 0.5, \; \forall i,$$

where $A \in \{L, M, H\}$. The weight $w^{(0)}_{k_{\alpha_p} i_m}$ between k_{α_p}th hidden node (corresponding to the interval $[x_{i_1}, x_{i_2}]$ for class C_k) and i_m ($m \in \{first(L), second(M), third(H)\})th$ node of the input layer corresponding to feature x_i is set by,

$$w^{(0)}_{k_{\alpha_p} i_m} = P_k + \epsilon, \tag{7.12}$$

where P_k is the *a priori* probability of class C_k and ϵ is a small random number. This hidden node is designated as *positive* hidden node. A second hidden node k_{α_n} is considered for the complement case and is termed as *negative* hidden node. Its connection weights are initialized as

$$w^{(0)}_{k_{\alpha_n} i_m} = (1 - P_k) + \epsilon \, . \tag{7.13}$$

Note that the small random number ϵ is considered to destroy any symmetry among the weights. Thus for an M-class problem domain we have $2M$ nodes in the hidden layer. In our algorithm we have considered the following two cases:

- *All* connections between these $2M$ hidden nodes and all nodes in the input layer are possible. The other weights are *initially* set as *small random numbers*.
- Only those *selected* connection weights initialized by Eqns. (7.12) and (7.13) are allowed..

It is to be mentioned that the method described above can suitably handle convex pattern classes only. In case of concave classes we consider multiple intervals for a

feature x_i corresponding to the various convex partitions that may be generated to approximate the given concave decision region. This also holds for the complement of the region in x_i in which a particular class C_k is not included. Hence, in such cases we introduce hidden nodes, *positive* and *negative*, for each of the intervals with connections being established by Eqns. (7.12) and (7.13) for the cases of a class belonging and not belonging to a region respectively, such that we get multiple hidden nodes for each of the two cases. In this connection it is to be noted that a concave class may also be subdivided into several convex regions as in [131].

Let there be $(k_{pos} + k_{neg})$ hidden nodes, where $k_{pos} = \sum_{\alpha_p} k_{\alpha_p}$ and $k_{neg} = \sum_{\alpha_n} k_{\alpha_n}$, generated for class C_k such that $k_{pos} \geq 1$ and $k_{neg} \geq 1$. Now connections are established between kth output node (for class C_k) and only the corresponding $(k_{pos} + k_{neg})$ hidden nodes. We assume that if any feature value (for class C_k) is outside some interval α, the total input received by the corresponding hidden node k_α is zero and this thereby produces an output $v_{k_\alpha}^{(1)} = 0.5$ due to the sigmoid nonlinearity of Eqn. (7.5). The connection weight $w_{kk_\alpha}^{(1)}$ between kth output node and $k_\alpha th$ hidden node is calculated from a series of equations generated as below. For an interval α as input for class C_k, the expression for output $v_k^{(2)}$ of kth output node is given by

$$v_k^{(2)} = f(v_{k_\alpha}^{(1)} w_{kk_\alpha}^{(1)} + \sum_{r \neq \alpha} 0.5 w_{kk_r}^{(1)}) \tag{7.14}$$

where $f(.)$ is the sigmoid function as in Eqn. (7.5) and the hidden nodes k_r correspond to the intervals not represented by the convex partition α. Thus for a particular class C_k we have as many equations as the number of intervals (including *not*) used for approximating any concave and/or convex decision region C_k. Thereby, we can uniquely compute each of the connection weights $w_{kk_\alpha}^{(1)}$ $\forall \alpha$ (corresponding to each hidden node k_α and class C_k pair).

The network architecture, so encoded, is then refined by training it on the pattern set supplied as input. In case of "all connections" between input and hidden layers, all the link weights are trained. In case of "selected connections" only the selected link weights are trained, while the other connections are kept clamped at zero. If the network achieves satisfactory performance, the classifier design is complete. Otherwise, we resort to node growing or link pruning.

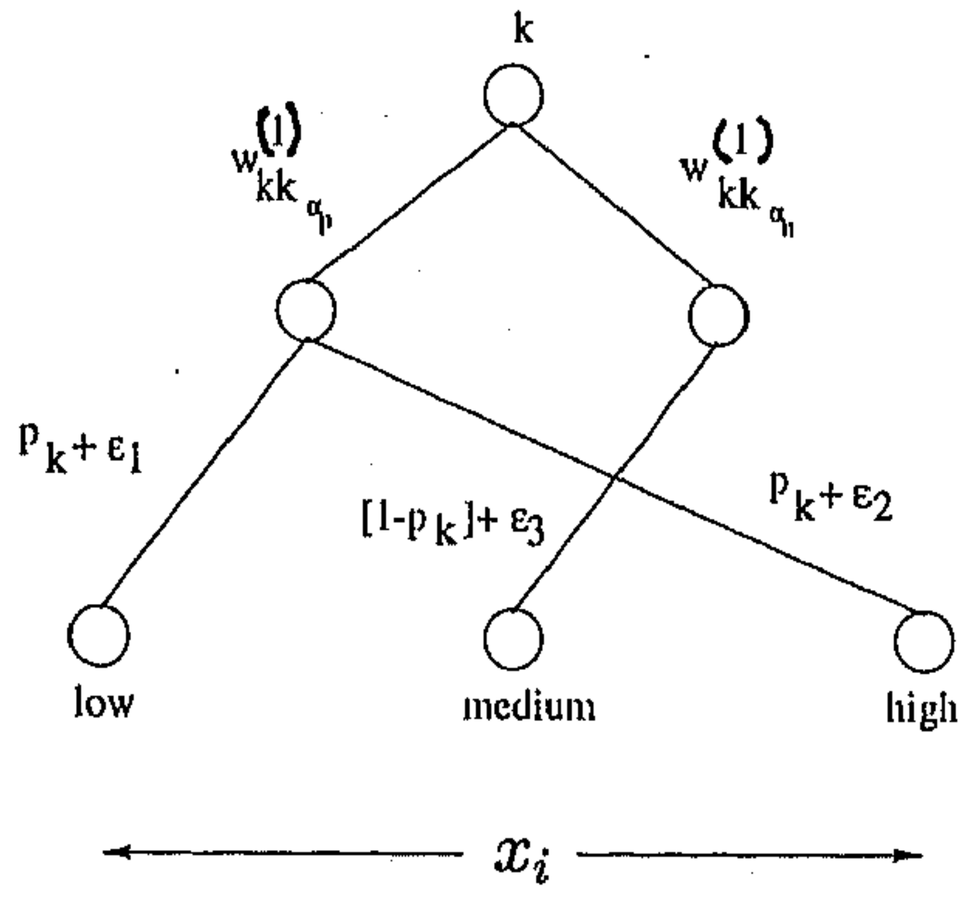

Figure 7.1: An example to demonstrate knowledge encoding.

Example

Consider the network depicted in Fig. 7.1. Let the output node k corresponding to a class C_k be connected to two hidden nodes k_{α_p} and k_{α_n} via connection weights $w^{(1)}_{kk_{\alpha p}}$ and $w^{(1)}_{kk_{\alpha n}}$. Let class C_k lie in the interval $[x_{i_1}, x_{i_2}]$ of input feature x_i. Then the weights between the input and the hidden layers are initially set from Eqns. (7.12) and (7.13) as

$$w^{(0)}_{k_{\alpha p} i_L} = P_k + \epsilon_1 \ ,$$

$$w^{(0)}_{k_{\alpha p} i_H} = P_k + \epsilon_2$$

and

$$w^{(0)}_{k_{\alpha n} i_M} = (1 - P_k) + \epsilon_3 \ .$$

In the case of the network with *all connections* the other weights between the input and the hidden layers (e.g., $w^{(0)}_{k_{\alpha p} i_M}$, $w^{(0)}_{k_{\alpha n} i_L}$, $w^{(0)}_{k_{\alpha n} i_H}$) are initialized by small random values. On the other hand, in the case of the network with *selected connections*, these are not considered at all.

Substituting Eqn. (7.14) in Eqn. (7.5), we have

$$v^{(2)}_k = \frac{1}{1 + exp\{-(v^{(1)}_{k_{\alpha p}} w^{(1)}_{kk_{\alpha p}} + 0.5 w^{(1)}_{kk_{\alpha n}})\}} \tag{7.15}$$

where $v_k^{(2)}$ is the output of kth output node and $v_{k_{ap}}^{(1)}$ is that of the hidden node, corresponding to the presented interval, connected to kth output node. From Eqn. 7.15 we have

$$v_{k_{ap}}^{(1)} w_{kk_{ap}}^{(1)} + 0.5 w_{kk_{an}}^{(1)} = \ln \frac{v_k^{(2)}}{1 - v_k^{(2)}} \tag{7.16}$$

Similarly, considering the complement-interval $[x_{i_1}, x_{i_2}]^c$ of feature x_i, we can write

$$1 - v_k^{(2)} = \frac{1}{1 + exp\{-(0.5 w_{kk_{ap}}^{(1)} + v_{k_{an}}^{(1)} w_{kk_{an}}^{(1)}\}}. \tag{7.17}$$

Therefore,

$$0.5 w_{kk_{ap}}^{(1)} + v_{k_{an}}^{(1)} w_{kk_{an}}^{(1)} = \ln \frac{1 - v_k^{(2)}}{v_k^{(2)}}. \tag{7.18}$$

The outputs $v_{k_{ap}}^{(1)}$ and $v_{k_{an}}^{(1)}$ are calculated using Eqn. (7.5) with appropriate input values. Then from Eqns. (7.16) and (7.18) we can evaluate $w_{kk_{ap}}^{(1)}$ and $w_{kk_{an}}^{(1)}$.

7.2.4 Pruning

A large number of connection weights in a network often results in redundancy, leading to the problem of just memorizing the patterns. In such cases pruning of less important links and/or hidden nodes is incorporated in order to get a near optimal network architecture and thereby enhance the generalization capability. There exist various algorithms for pruning [184, 102, 76, 101, 204] ANNs. Here we have incorporated link pruning of the knowledge-based network in a slightly different way.

A connection weight is pruned if its contribution toward the network output is the least significant during the presentation of the training set. Therefore, the link $w_{ji}^{(h)}$ in layer (h) is pruned if

$$\sum_p w_{ji}^{(h)} v_i^{(h)} = \min_{k,m} \left\{ \sum_p w_{km}^{(h)} v_m^{(h)} \right\}, \tag{7.19}$$

where the summation is taken over all the patterns p in the training set and the *minimum* is computed over the indices k, m.

When a network with large number of connection weights results in poor classification performance after a certain number of epochs, links between layers $(h+1)$ and (h), for each (h), need to be selected for pruning by Eqn. (7.19). The resulting network is

retrained for a few more epochs and this process is continued till we get a satisfactory recognition score.

Note that we do not resort to node pruning as the number of hidden nodes are initially encoded with the domain knowledge and are, therefore, not redundant. During refinement by training, it is the growth of extra links that leads to redundancy. Hence it is our objective to prune a few such redundant links to improve the generalization capability of the network.

7.2.5 Growing of hidden nodes

If after a certain number of epochs (experimentally determined) the classifier still does not recognize a certain class C_k well and the network size is not too large, we resort to adding a hidden node (instead of pruning) to our knowledge-based model. Connection weights are established between this new node and all the output nodes. Links are also introduced from all input nodes to this newly added node. Now training is allowed on these new connection weights for a few epochs (again empirically set) using only those samples which are in class C_k while keeping all the other links frozen. Then all the links are retrained with the entire training set, and the process of adding, freezing and retraining are continued, until all the classes are reasonably well recognized. Note that other approaches for growing of nodes in ANNs may be found in [187, 53].

7.3 Rule Generation

The trained knowledge-based network is used for rule generation in *If-Then* form in order to justify any decision reached. These rules describe the extent to which a test pattern belongs or does not belong to one of the classes in terms of antecedent and consequent clauses provided in natural form. We use two rule-generation strategies as described below. The algorithms are, however, different from that reported in [146].

- *Method (i)* – Treating the network as a black-box and using the training set input (in numeric and/or linguistic forms) and network output (with confidence

factor) to generate the antecedent and consequent parts respectively.

- *Method (ii)* – Backtracking along maximal weighted paths using the trained net and utilizing its input and output activations (with confidence factor) for obtaining the antecedent and consequent clauses respectively.

7.3.1 Using numeric and/or linguistic inputs–*Method (i)*

In this method we use an exhaustive set of numeric and/or linguistic inputs along with their hedges at the input for antecedent clauses (*If* parts). For this purpose we use the following equations for the linguistic inputs and their hedges.

When input feature x_i is linguistic, its membership values for the π-sets *low*, *medium* and *high* are quantified as [159]

$$\begin{aligned}
low &\equiv \left\{ \frac{0.95}{L}, \frac{\pi\left(x_i\left(\frac{0.95}{L}\right); c_{i_m}, \lambda_{i_m}\right)}{M}, \frac{\pi\left(x_i\left(\frac{0.95}{L}\right); c_{i_h}, \lambda_{i_h}\right)}{H} \right\} \\
medium &\equiv \left\{ \frac{\pi\left(x_i\left(\frac{0.95}{M}\right); c_{i_l}, \lambda_{i_l}\right)}{L}, \frac{0.95}{M}, \frac{\pi\left(x_i\left(\frac{0.95}{M}\right); c_{i_h}, \lambda_{i_h}\right)}{H} \right\} \\
high &\equiv \left\{ \frac{\pi\left(x_i\left(\frac{0.95}{H}\right); c_{i_l}, \lambda_{i_l}\right)}{L}, \frac{\pi\left(x_i\left(\frac{0.95}{H}\right); c_{i_m}, \lambda_{i_m}\right)}{M}, \frac{0.95}{H} \right\}
\end{aligned} \tag{7.20}$$

where $x_i\left(\frac{0.95}{L}\right)$, $x_i\left(\frac{0.95}{M}\right)$ and $x_i\left(\frac{0.95}{H}\right)$ denote the corresponding feature values x_i at which the three linguistic properties attain membership values of 0.95.

For the linguistic hedges *very*, *more or less (mol)* and *not* corresponding to *low*, *medium* and *high* [159] we use

$$\begin{aligned}
very\ low &\equiv \{Con(L), Con(M), Con(H)\} \\
very\ medium &\equiv \{Con(L), Dil(M), Con(H)\} \\
very\ high &\equiv \{Con(L), Con(M), Con(H)\},
\end{aligned} \tag{7.21}$$

$$\begin{aligned}
more\ or\ less\ low &\equiv \{Con(L), Dil(M), Dil(H)\} \\
more\ or\ less\ medium &\equiv \{Dil(L), Con(M), Dil(H)\} \\
more\ or\ less\ high &\equiv \{Dil(L), Dil(M), Con(H)\}
\end{aligned} \tag{7.22}$$

and

$$\begin{aligned}
not\ low &\equiv \left\{\frac{1.0-0.95}{L}, \frac{\pi\left(x_i\left(\frac{1.0-0.95}{L}\right); c_{i_m}, \lambda_{i_m}\right)}{M}, \frac{\pi\left(x_i\left(\frac{1.0-0.95}{L}\right); c_{i_h}, \lambda_{i_h}\right)}{H}\right\} \\
not\ medium &\equiv \left\{\frac{\pi\left(x_i\left(\frac{1.0-0.95}{M}\right); c_{i_l}, \lambda_{i_l}\right)}{L}, \frac{1.0-0.95}{M}, \frac{\pi\left(x_i\left(\frac{1.0-0.95}{M}\right); c_{i_h}, \lambda_{i_h}\right)}{H}\right\} \\
not\ high &\equiv \left\{\frac{\pi\left(x_i\left(\frac{1.0-0.95}{H}\right); c_{i_l}, \lambda_{i_l}\right)}{L}, \frac{\pi\left(x_i\left(\frac{1.0-0.95}{H}\right); c_{i_m}, \lambda_{i_m}\right)}{M}, \frac{1.0-0.95}{H}\right\}
\end{aligned} \tag{7.23}$$

where $\mu_{Con(A)}(x) = \{\mu_A(x)\}^2$ and $\mu_{Dil(A)}(x) = \{\mu_A(x)\}^{\frac{1}{2}}$.

Let us consider a data set with two features x_1 and x_2. The linguistic pattern corresponding to x_1 *low* and x_2 *high* is computed (using Eqn. (7.20)) as the vector

$$[0.95, \pi(x_{1_l}; c_{1_m}, \lambda_{1_m}), \pi(x_{1_l}; c_{1_h}, \lambda_{1_h}), \pi(x_{2_h}; c_{2_l}, \lambda_{2_l}), \pi(x_{2_h}; c_{2_m}, \lambda_{2_m}), 0.95],$$

where $\pi(x_{1_l}; c_{1_l}, \lambda_{1_l}) = 0.95$ and $\pi(x_{2_h}; c_{2_h}, \lambda_{2_h}) = 0.95$. Here x_{1_l} and x_{2_h} refer to the feature values for x_1 and x_2 at which $\mu_{low(x_1)} = 0.95$ and $\mu_{high(x_2)} = 0.95$ respectively in Eqn. (7.1). Note that a π-function has two symmetrical points – at left (where π-function is monotonic increasing) or at right (where π-function is monotonic decreasing) – corresponding to the value 0.95. By convention we have chosen the left one to compute (say) $\pi(x_{1_l}; c_{1_m}, \lambda_{1_m})$ corresponding to $\pi(x_{1_l}; c_{1_l}, \lambda_{1_l}) = 0.95$ or $\pi(x_{2_h}; c_{2_l}, \lambda_{2_l})$ corresponding to $\pi(x_{2_h}; c_{2_h}, \lambda_{2_h}) = 0.95$. All the linguistic patterns corresponding to the different hedges (as defined by Eqns. (7.21)-(7.23)) are also generated. Thus we have a total of 9^n such patterns (*i.e.*, corresponding to *very*, *more or less* and *not* for each of linguistic values *low*, *medium* and *high* of each of the features) for a data set with n features. These patterns constitute the antecedent part of the rules. In the case of *numeric patterns*, the distance between pth pattern and each of the linguistic pattern vectors are calculated. The linguistic pattern (with or without the hedges of Eqns. (7.20)-(7.23)) closest to pth pattern determines the antecedent part of the rule.

To generate the consequent part of the rule, we use a measure which reflects the amount of difficulty in arriving at a decision by minimizing the ambiguity in the computed output vector. A *confidence factor (CF)* is defined [159] as,

$$CF = \frac{1}{2}[\{v_{max}^{(2)}\}^{f_{max}} + \frac{1}{M-1}\sum_{k=1}^{M}\{v_{max}^{(2)} - v_k^{(2)}\}], \quad 0 \leq CF \leq 1, \tag{7.24}$$

where $v_{max}^{(2)} = \max_{k=1}^{M}\{v_k^{(2)}\}$, $v_k^{(2)}$ is the kth component in the output vector $\mathbf{v}^{(2)}$ (by Eqn. (7.5)) and f_{max} indicates the number of occurrences of $v_{max}^{(2)}$ in $\mathbf{v}^{(2)}$. Note that CF

takes care of the fact that the difficulty in assigning a particular pattern class depends not only on the highest entry $v_{max}^{(2)}$ in the output vector but also on its differences from the other entries $v_k^{(2)}$. It is seen that the higher the value of CF, the lower is the difficulty in deciding a class and hence greater is the degree of certainty of the output decision. Based on the value of CF, the system makes the following decisions while generating the consequent clause (*Then* part) of the rule. Let $v_k^{(2)} = v_{max}^{(2)}$ such that the pattern under consideration belongs to class C_k. We have

(i) if $(0.8 \leq CF_k \leq 1.0)$ then very likely class C_k, and there is no second choice,

(ii) if $(0.6 \leq CF_k < 0.8)$ then likely class C_k, and there is second choice,

(iii) if $(0.4 \leq CF_k < 0.6)$ then more or less likely class C_k, and there is second choice,

(iv) if $(0.1 \leq CF_k < 0.4)$ then not unlikely class C_k, and there is no second choice,

(v) if $(CF_k < 0.1)$ then unable to recognize class C_k and there is no second choice.

To obtain a second choice corresponding to a pattern class C_{k_2} (say), we find the *confidence factor* CF_{k_2} for the second highest entry $v_{k_2}^{(2)}$ in the output vector using Eqn. (7.24). There may be some cases where there are multiple entries with the highest value $v_{max}^{(2)}$ in the output vector. In that case, there will not be a second choice of pattern class. Instead, the form of the consequent will be "*likely* class C_k *or* $C_{k'}$" where the output values corresponding to classes C_k and $C_{k'}$ both have the highest value $v_{max}^{(2)}$. Note that identical rules, if any, are discarded from the generated rule set.

7.3.2 Backtracking along trained connection weights–*Method (ii)*

An input pattern $\mathbf{x}_p$ from the training set is presented to the input of the trained network and its output is computed. The consequent part of the corresponding *If-Then* rule is generated by Eqn. (7.24) as described in Section 7.3.1. To find the antecedent clauses of the rule, we backtrack from the output layer to the input through *positive* hidden nodes along the maximal weighted links. The path from node k in the output layer to node i_A in the input layer through node j in the hidden layer is maximal if

$$w_{kj}^{(1)} v_j^{(1)} + w_{ji_A}^{(0)} v_{i_A}^{(0)} = \max_m \{ w_{km}^{(1)} v_m^{(1)} + w_{mi_A}^{(0)} v_{i_A}^{(0)} \} \tag{7.25}$$

provided that $v_j^{(1)} \geq 0.5$ and $v_{i_A}^{(0)} > 0.5$, and the *maximum* is computed over the index m. Here the path length from node k in the output layer to node j in the hidden layer is $w_{kj}^{(1)} v_j^{(1)}$ and not $w_{kj}^{(1)}$ as defined by Mitra and Pal [146] in an earlier approach. Besides, the *confidence factor CF* of Eqn. (7.24) is also different and in certain ways better than the *belief* used there. We consider only one node i_A corresponding to the three linguistic values of each feature x_i so that

$$w_{ji_A}^{(0)} v_{i_A}^{(0)} = \max_{B \in \{L,M,H\}} w_{ji_B}^{(0)} v_{i_B}^{(0)} \tag{7.26}$$

where A and B correspond to *low*(L), *medium*(M) or *high*(H). For a linguistic feature x_{pA} (Eqn. (7.26)), the closest 3-dimensional linguistic pattern vector *low*, *medium* or *high* (Eqn. (7.20)) with or without hedges (Eqns. (7.21)-(7.23)), to the relevant 3-dimensional part of pattern $\mathbf{x}_p$, is selected as the antecedent clause [146]. This is done for all input features to which a path may be found by Eqn. (7.25). The complete *If* part of the rule is found by ANDing the clauses corresponding to each of the features, e.g.

If x_1 *is mol* A *and* x_2 *is not* A *and*...*and* x_n *is very* A.

Negative rules

It may sometimes happen that we are unable to classify a test pattern directly with the help of the *positive* rules (concerning its belonging to a class) derived by any of the above two methods. In such cases, we proceed by discarding some classes which are unlikely to contain the pattern, and thereby arrive at the class(es) to which the pattern possibly belongs. In other words, in the absence of positive information regarding the belonging of pattern $\mathbf{x}_p$ to class C_k, we use the complementary information about the pattern $\mathbf{x}_p$ not belonging to class $C_{k'}$. To handle such situations, we have generated *negative* rules with the consequent part of the form 'not in class $C_{k'}$' by backtracking from the output layer through the trained connection weights. Note that for *positive* rules we traverse the hidden node k_{α_p} while for *negative* rules we backtrack along the hidden node k_{α_n}.

Let an input pattern $\mathbf{x}_p$ from the training set be presented to the input layer of the trained network such that the output of the node in the output layer corresponding to class $C_{k'}$ is minimum, *i.e.*, $v_{k'}^{(2)} = \min_l \{v_l^{(2)}\}$. Therefore, we are certain that the

pattern is (possibly) not included in class $C_{k'}$. Hence, the consequent part of the corresponding rule becomes *not in class* $C_{k'}$. The antecedent part of the rule is obtained by backtracking from output node k' through *negative* hidden nodes along the maximal path of Eqn. (7.25) with the restrictions that now we consider the absolute values of the individual product terms. The corresponding rule, so obtained, is of the form

If x_1 *is mol A and*...*and* x_n *is very A then the pattern is not in class* $C_{k'}$.

Note that the approach in [146] did not consider such negative rules. ♣

It is worth mentioning that the above rule generation techniques can also handle the situations where the input is given in set form. In other words, the feature information x_i of a test pattern is neither linguistic nor numeric, but may be available as (a) $x_i \geq x_{i_1}$, some lower bound; (b) $x_i \leq x_{i_2}$, some upper bound; or (c) in some interval $[x_{i_1}, x_{i_2}]$ such that x_i lies between x_{i_1} and x_{i_2}. In these cases, the linguistic values *low, medium* and *high* corresponding to the input given in the set form and hedges are evaluated from Eqns. (7.8)-(7.10). Then the rule with this antecedent is picked up, and we take a decision on the basis of the corresponding consequent part regarding its class (as explained earlier).

Example

Consider a knowledge-based network (Model AN) given in Fig. 7.2 demonstrating the rule generation technique using Method (ii). We have considered feature x_i for the antecedent clause. Only classes C_k and $C_{k'}$, with maximum and minimum outputs respectively (on presenting a pattern at the input layer), have been considered for the generation of *positive* and *negative* rules. The hidden nodes 1 and 2 correspond to two convex segments of the region represented by class C_k, and 3 corresponds to the region complement to class C_k. Similarly, for class $C_{k'}$, we have considered the hidden nodes 4 corresponding to the region of class $C_{k'}$, and 5 and 6 correspond to the region other than class $C_{k'}$. Thus, the nodes 1 and 2, and 4 represent the *positive* nodes for classes C_k and $C_{k'}$ respectively. Similarly, the nodes 3, and 5 and 6 denote the *negative* nodes for classes C_k and $C_{k'}$ respectively.

A pattern with linguistic values $low(L) = 0.6$, $medium(M) = 0.8$ and $high(H) = 0.2$ of feature x_i is presented to the input layer of the network. Assume that the activations of the output nodes corresponding to classes C_k and $C_{k'}$ are 0.9 and 0.1

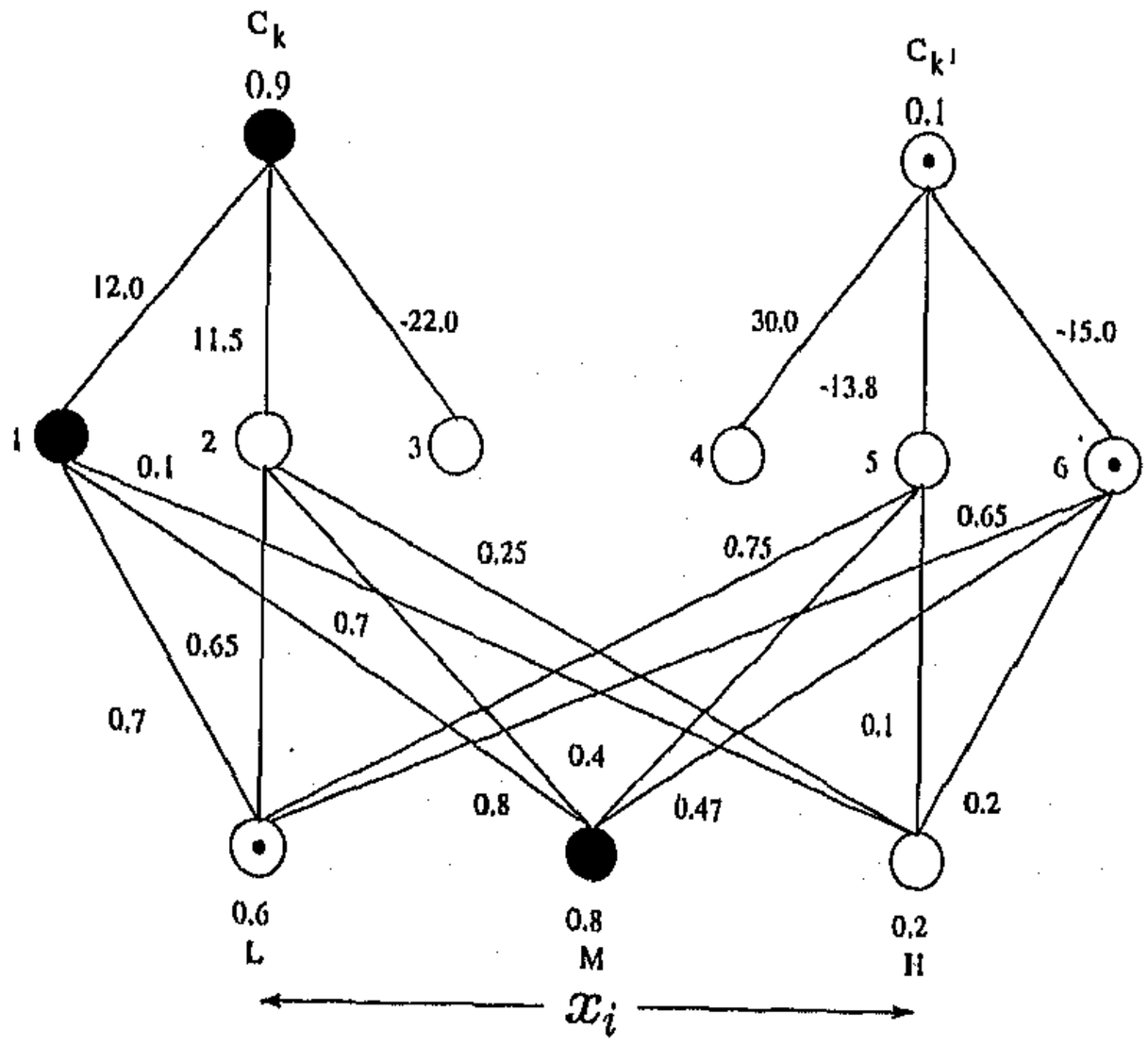

Figure 7.2: An example to demonstrate *positive* and *negative* rule generation using Method (ii).

respectively. Therefore, backtracking starts from the output node corresponding to class C_k and searches for the maximal path through the hidden nodes 1 and 2 only (if the activations of these nodes are at least 0.5) for *positive* rule generation. For the *negative* rule generation, it starts from the output node corresponding to class $C_{k'}$ and searches for the maximal path through the hidden nodes 5 and 6 only (if the outputs of these nodes are at least 0.5). The links with weights as shown in the figure are obtained during training. For clarity of the figure, we have not considered the links from hidden node 3 to L, M and H corresponding to feature x_i as we do not require these links for the generation of *positive* rule. Similarly, the links from the hidden node 4 to the input nodes L, M and H are not shown.

We denote the path length (as explained by Eqn. (7.25)) from the hidden node j to the input node i by $path_{ji}^{(0)}$, and that from the output node k to the hidden node j by $path_{kj}^{(1)}$. The path length from the output node k to the input node i via hidden node j is denoted by $path_{kji}$. Therefore, from Fig. 7.2 we find the following path lengths: $path_{1L}^{(0)} = 0.7*0.6 = 0.42$, $path_{1M}^{(0)} = 0.8*0.8 = 0.64$, $path_{2L}^{(0)} = 0.39$ and $path_{2M}^{(0)} = 0.56$. Note that $path_{1H}^{(0)}$ and $path_{2H}^{(0)}$ have not been considered as the activation of the input

node H is less than 0.5. The total inputs received by the hidden nodes 1 and 2 are found to be 1.08 and 1.0 respectively. Therefore, the activations of these nodes are $v_1^{(1)} = 0.75$ and $v_2^{(1)} = 0.73$ respectively by Eqn. (7.5). The path lengths between the nodes in the output layer and the hidden layer are $path_{k1}^{(1)} = 9.0$ and $path_{k2}^{(1)} = 8.4$. The total path lengths are found to be $path_{k1L} = 9.42$, $path_{k1M} = 9.64$, $path_{k2L} = 8.79$ and $path_{k2M} = 8.96$. Hence, the maximal path from the output node corresponding to class C_k is obtained as the path via the hidden node 1 to the input node M (the selected path consists of the links joining the nodes indicated by solid circles in Fig. 7.2). The antecedent clause corresponding to x_i is "x_i *is mol medium*", and is obtained by finding the closest match of the 3-dimensional vector corresponding to x_i, to the respective linguistic pattern (with/ without hedges) given by Eqns. (7.20)-(7.23). The consequent part of the rule is *very likely class* C_k, as obtained from Eqn. (7.24).

For the generation of *negative* rule, we compute the following path lengths: $path_{5L}^{(0)} = 0.45$, $path_{5M}^{(0)} = 0.32$, $path_{6L}^{(0)} = 0.39$ and $path_{6M}^{(0)} = 0.38$. The total inputs received by the hidden nodes 5 and 6 are 0.79 and 0.81 respectively. Therefore, the corresponding activations are $v_5^{(1)} = 0.69$ and $v_6^{(1)} = 0.69$. Thus, the path lengths between the nodes in the output layer and the hidden layer are $path_{k'5}^{(1)} = 9.52$ and $path_{k'6}^{(1)} = 10.35$. The total path lengths are $path_{k'5L} = 9.97$, $path_{k'5M} = 9.84$, $path_{k'6L} = 10.74$ and $path_{k'6M} = 10.73$. Hence, the maximal path from the output node corresponding to class $C_{k'}$ is obtained as the path via the hidden node 6 to the input node L (the path consisting of the links joining the nodes indicated by circles with dots inside in Fig. 7.2). Therefore, the corresponding antecedent clause, "x_i *is mol low*, of the *negative* rule is obtained as in the earlier case. Thus, the *negative* rule is: *If* x_i *is mol low then the pattern is not in class* $C_{k'}$.

7.4 Experimental Results

In this section we compare the classification and rule generation performance of the knowledge-based model with that of the conventional and fuzzy versions of the MLP (Appendices C and F) [167, 146], and the fuzzy min-max neural network (Appendix G) [206] both on the same artificially generated Pat3 (Chapter 6), and the real life vowel and medical data (Chapter 2). As in Chapter 6, in all the cases the data sets

have been divided into two subsets – *training* and *testing*. While *perc*% samples are considered during training, the remaining (100 – *perc*)% is used for testing.

It is found that the knowledge-based model converges to a good solution with a very small number of training epochs (iterations) for all the data sets. Note that we have used the following four neuro-fuzzy knowledge-based models designated as follows:

all connections with not – Model AN
all connections without not – Model A
selected connections with not – Model SN
selected connections without not – Model S.

Results are compared with those of the fuzzy MLP (Model F) [167], the conventional MLP (Model C) and the fuzzy min-max network (Model FMM) [206]. The number of links required in each case is appropriately indicated (in parenthesis after the name of the corresponding model) in the tables. The variables f_d and f_e of Eqn. (7.6) were set at 5.0 and 1.0 respectively for vowel and medical data. For Pat3 data we use $d_k \in \{0, 1\}$, d_k being the desired output of kth output node, and hence f_d and f_e are not required.

7.4.1 Classification

As the class structures of Pat3 data (Fig. 6.2) are concave, we have found approximately (by inspecting the feature space) the set of intervals of the features where each of the pattern classes lie (or do not lie). Table 7.1 depicts the results obtained with this data (for $perc = 10$). A total of six intervals (*i.e.*, six hidden nodes) for the two features are found to be sufficient to characterize the classes if we do not consider the intervals in which any of the classes is *not included* (*i.e.*, Eqns. (7.8)-(7.10) only). This is termed as the *without not* case. Otherwise, if we consider both the intervals of the features in which a class *is included* and *is not included*, we require a total of ten intervals (*i.e.*, ten hidden nodes). This is called the *with not* case. It is observed that models A and SN give 100% recognition score in just 600 epochs. The other models (AN and S) have not been able to recognize class 2 at this stage. In model AN, perhaps the large number of interconnections encode too much redundant information thereby not enabling the classifier to recognize class 2.

Table 7.1: Performance of different models on Pat3 data for $perc = 10$.

Model	Class	Score(%)	
		Training	Testing
AN (82)	1	100.0	100.0
	2	0.0	0.0
	Overall	83.64	82.47
A (47)	1	100.0	100.0
	2	100.0	100.0
	Overall	100.0	100.0
SN ($\leq$ 82)	1	100.0	100.0
	2	100.0	100.0
	Overall	100.0	100.0
S ($\leq$ 47)	1	100.0	100.0
	2	0.0	0.0
	Overall	83.64	82.47
FMM (84)	1	100.0	100.0
	2	100.0	84.09
	Overall	100.0	97.21
FMM (48)	1	100.0	100.0
	2	55.56	48.86
	Overall	92.73	91.04

On the other hand, model S provides poor result probably due to under-information. The performance of C and F is the same as that of AN and S. That is why we have not included the results for C and F in Table 7.1.

We have resorted to pruning of links in models AN, F and growing of hidden nodes in cases of models S, F, C. It is found that after only 100 epochs growing the model SN provides overall recognition score of 100% on the training set and 99.8% on the test set. This demonstrates a remarkable improvement in performance. Hidden nodes were also added to models C and F at the same stage but the performance is found to be poor (0% recognition for class 2) in case of Pat3 data. Pruning model AN results in 100% recognition scores for both the training and test sets. The links have been pruned from 600 epochs at intervals of ten epochs, upto 750 epochs, and then the network has been trained until 900 epochs. Although model F could now recognize around 20% patterns from class 2, this was considerably less than that by model AN.

Table 7.2 shows the results obtained with vowel data (for $perc = 10$). Since all the classes in the feature space are convex, we use two hidden nodes for each of the classes. Hence only *with not* models have been considered and we require a total of twelve hidden nodes for this data set. The results demonstrate that model AN gives acceptably good performance in just 200 epochs whereas model SN cannot do the same due to under-information. Note that the vowel classes are overlapping (fuzzy), thereby generating fuzzy output class membership values that require storage of more information than in case of crisp class membership values. Perhaps this accounts for the better performance of model AN (with more connections). Models C and F were unable to recognize classes 1, 2 and 4, and fared the worst (overall recognition score during training and testing being 42.35% and 39.19% for model C, and 55.29% and 52.93% for model F).

As model AN performs reasonably well for all the classes initially (before growing), the incorporation of additional hidden nodes does not improve the results in this case. However, when model SN is augmented for class '∂', it is found that after 350 epochs the model recognizes 14.29% of class '∂' during training and 1.54% during testing. The overall scores rose to 83.53% and 74.17% for the training and testing sets respectively. The results are depicted in Table 7.3.

Table 7.4 demonstrates the classification performance for medical data (for $perc = 30$)

Table 7.2: Performance of different models on vowel data for $perc = 10$.

Model	Class	Score(%)	
		Training	Testing
AN (138)	∂	42.86	27.69
	a	87.5	86.42
	i	94.12	87.74
	u	100.0	82.35
	e	90.0	69.52
	o	100.0	93.83
	Overall	90.59	78.63
SN ($\leq$ 138)	∂	0.0	0.0
	a	62.5	58.02
	i	94.12	87.74
	u	100.0	82.35
	e	85.0	68.45
	o	94.44	93.21
	Overall	82.35	73.79
FMM (504)	∂	71.43	30.77
	a	100.0	80.25
	i	100.0	91.61
	u	86.67	72.06
	e	95.0	62.57
	o	88.89	85.80
	Overall	91.76	73.92
FMM (198)	∂	71.43	75.38
	a	50.0	41.98
	i	76.47	74.19
	u	33.33	43.38
	e	65.0	68.98
	o	44.44	35.19
	Overall	56.47	56.36

Table 7.3: Effect of adding hidden node on the performance of the various knowledge-based models on vowel data for $perc = 10$.

Model	Class	Score(%)	
		Training	Testing
AN	∂	42.86	23.08
	a	87.5	88.89
	i	94.12	87.74
	u	100.0	82.35
	e	90.0	70.05
	o	100.0	93.83
	Overall	90.59	78.63
SN	∂	14.29	1.54
	a	62.5	58.02
	i	94.12	92.26
	u	100.0	84.56
	e	85.0	66.84
	o	94.44	93.83
	Overall	83.53	74.17

Table 7.4: Performance of different models on medical data for $perc = 30$.

Model	Class	Score(%)	
		Training	Testing
F (260)	*ALD*	41.18	37.80
	PH	90.57	89.60
	LC	2.70	9.20
	C	91.43	92.77
	Overall	59.75	60.48
AN (236)	*ALD*	61.76	52.44
	PH	84.91	77.60
	LC	59.46	43.68
	C	91.43	86.75
	Overall	75.47	66.31
SN ($\leq$ 236)	*ALD*	0.0	0.0
	PH	98.11	100.0
	LC	0.0	0.0
	C	0.0	0.0
	Overall	32.70	33.16
FMM (2068)	*ALD*	76.47	32.93
	PH	90.57	60.0
	LC	48.65	2.30
	C	82.86	55.42
	Overall	76.10	39.79
FMM (236)	*ALD*	8.82	0.0
	PH	100.0	100.0
	LC	18.92	0.0
	C	0.0	0.0
	Overall	39.62	33.16

where classes ALD, PH, LC and C correspond to the four disease classes described in Chapter 2. Note that here we do not know *a priori* the shape of the pattern classes in the nine dimensional feature space. We have assumed that the classes are convex, so that only eight hidden nodes are used corresponding to the four classes. As in the case of vowel data, only the knowledge-based models AN and SN have been used. Since we have approximated the structures of the classes as convex, model SN which uses only those links that are encoded with the initial knowledge performs rather poorly. Perhaps it would require more nodes and links than were available under our assumption. However, model AN, which is allowed to grow extra links, is found to have solved this problem. Its performance is considerably better than those of models SN and F in just 500 epochs (Table 7.4). Note that the model C, being unable to recognize classes ALD, LC and C, is not included in the table.

Tables 7.1, 7.2 and 7.4 also show the classification performance of the fuzzy min-max neural network (model FMM) [206] on the three data sets. In this model, the number of links can be varied by altering some parameters *viz.*, γ and θ (Appendix G). Here we show the results for two different configurations *viz.*, (i) providing more or less the same overall recognition score (on the training sets) as the knowledge-based model and (ii) providing more or less the same number of links as the knowledge-based model. Note that model A for Pat3 data (Table 7.1), and model AN for both vowel (Table 7.2) and medical (Table 7.4) data have been compared for this purpose, as they perform the best. It is clear that model FMM requires more links than the knowledge-based model to get more or less the same overall recognition score. Similarly, with more or less the same number of links model FMM performs poorer for all the data sets.

Figs. 7.3 and 7.4 depict the variation of *mean square error* with the number of iterations for Pat3 and vowel data respectively. In the case of Pat3 data we demonstrate the behavior of the two better knowledge-based models A and SN only, for ease of explanation. It is observed that model C has the worst performance. Model A (for Pat3 data) and model AN (for vowel data) behave the best. It is found that Model SN is better than Model F in the beginning and converges to a good solution very fast (in about 600 iterations in Fig. 7.3) for Pat3 data while Model F requires about twice to thrice this time to reach the same level of performance. In contrast, for vowel data, Model F surpasses model SN at around 500 iterations (as seen from

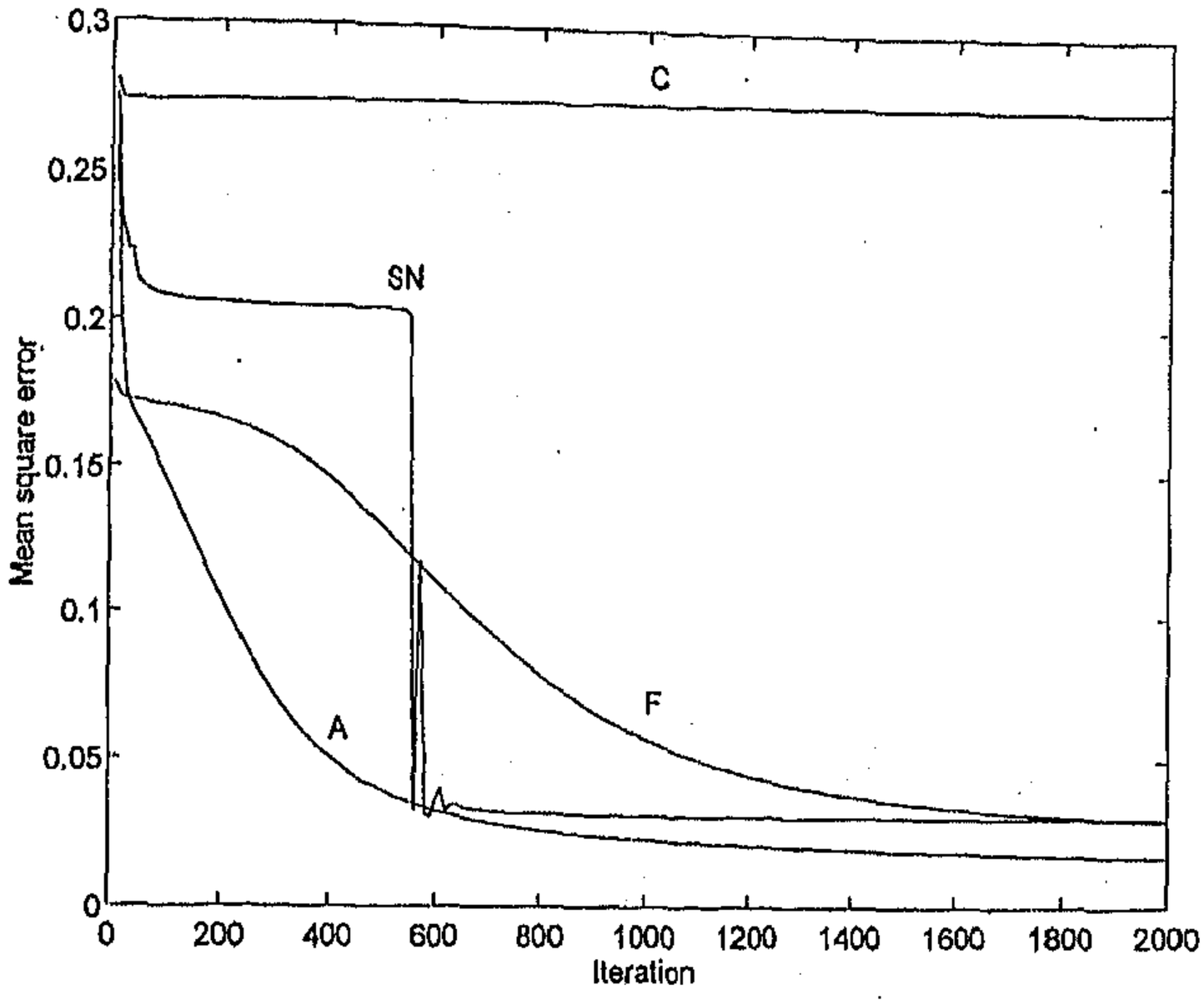

Figure 7.3: Variation of *mean square error* with *number of iterations* for Pat3 data.

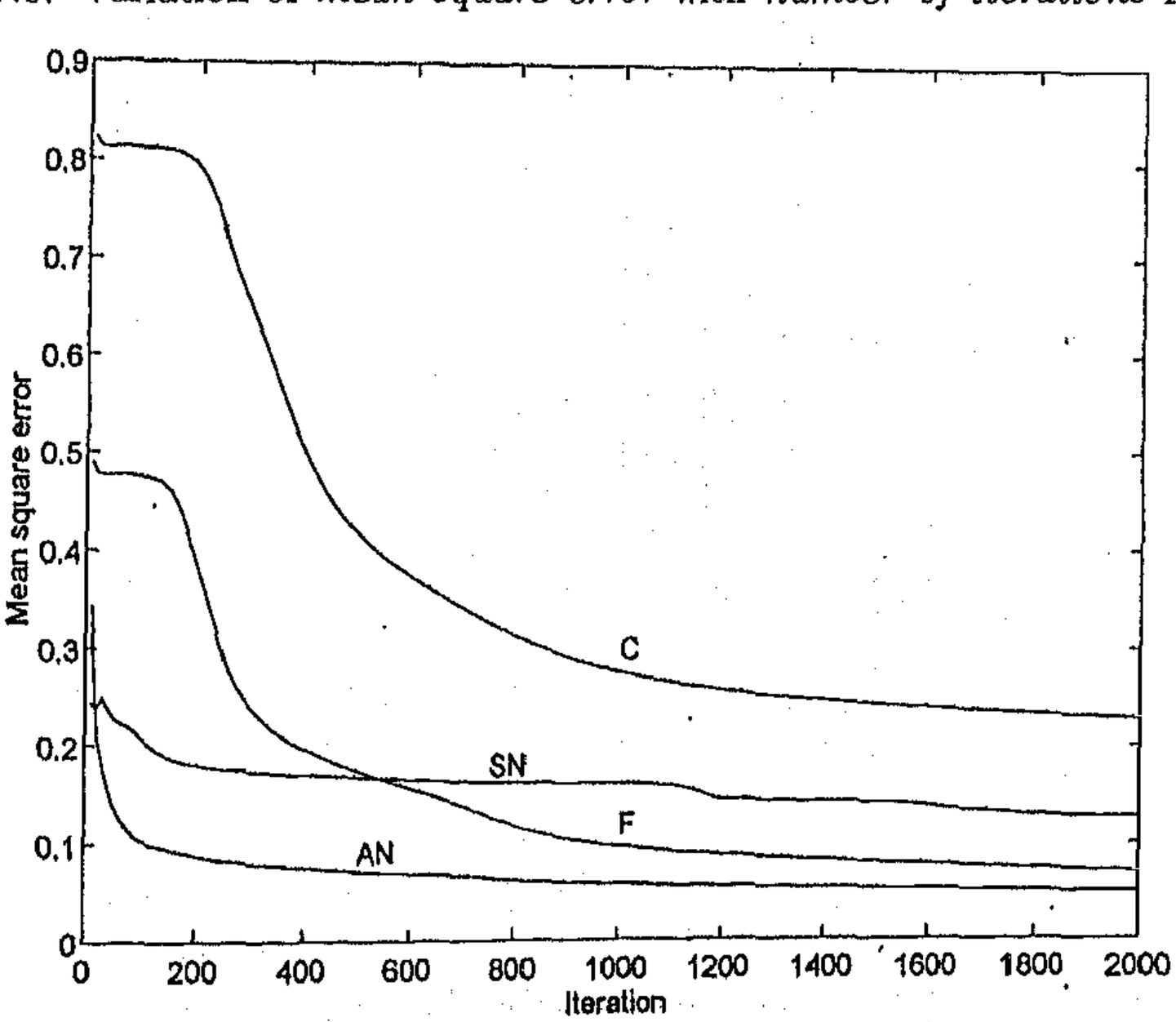

Figure 7.4: Variation of *mean square error* with *number of iterations* for vowel data.

Fig. 7.4). However, Model AN/A is always the best perhaps due to the presence of less redundancy (than Model F) along with more knowledge (than Model SN). Note that Tables 7.1 and 7.2 depict the performance of the knowledge-based models at 600 iterations respectively. This accounts for the relatively poor performance of Model F at this stage, whereas it fares better with longer training time (as is evident from the figures).

Table 7.5: Comparative recognition scores of neuro-fuzzy knowledge-based and case-based systems on different data sets.

Data set	Class	Recognition score (%)			
		KBNN		CBNN	
		Train	Test	Train	Test
Pat3	1	100.0	100.0	100.0	96.86
	2	100.0	100.0	100.0	87.50
	Overall	100.0	100.0	100.0	95.22
vowel	∂	42.86	27.69	100.0	52.31
	a	87.50	86.42	100.0	70.37
	i	94.12	87.74	100.0	65.16
	u	100.0	82.35	86.67	77.94
	e	90.00	69.52	100.0	81.28
	o	100.0	93.83	88.89	89.51
	Overall	90.59	78.63	95.29	75.70
medical	ALD	61.76	52.44	71.43	56.79
	PH	84.91	77.60	70.37	29.84
	LC	59.46	43.68	78.95	66.28
	C	91.43	86.75	58.33	57.32
	Overall	75.47	66.31	69.94	50.13

Table 7.5 compares the results obtained by the knowledge-based system with those of the case-based network CBNN (Chapter 6) for all these data sets. Among the different models (A, AN, S or SN) of knowledge-based network (KBNN), we have considered here only that model which has provided the best recognition score for a particular data. In the case of Pat3 and vowel data, the results correspond to

$perc = 10$, while for medical data, it is $perc = 30$.

The classification performance of KBNN and CBNN, on the training sets of Pat3 data, are identical. For vowel data, the recognition score obtained by CBNN, on the training data, is higher than that of KBNN, whereas the reverse is true in the case of medical data. However, for the test sets, KBNN performs better than CBNN. This signifies that the generalization capability of the former one is better. In all the cases, KBNN requires much less number of links (47, 138 and 236 for Pat3, vowel and medical data respectively) as compared to the CBNN (115, 455 and 1520 corresponding to Pat3, vowel and medical data). On the other hand, the number of iterations required to train a KBNN is larger than that of CBNN. For example, the number of iterations required to train KBNN is found to be 600, 200 and 500 for Pat3, vowel and medical data, whereas the corresponding figures are 1, 5 and 1 for CBNN.

7.4.2 Rule generation

Tables 7.6–7.8 compare the rules generated for Pat3, vowel and medical data respectively, by the methods described in Section 7.3 using the knowledge encoded networks and the fuzzy MLP [167]. The rules generated on the various models are not identical due to the different amounts of redundancy inherent in them and the difference in the encoding of their network architectures. Method (ii) often produces different results (as compared to method (i) where only the input and output of the neural net are considered) because here the trained connection weight magnitudes are utilized during the tracing of the maximal weighted paths, thereby using the encoded and refined domain knowledge along with the test case feature values.

Negative Rules

Let us consider the trained connection weights $w^{(1)}_{kk_\alpha}$ of the knowledge-based network in the case of Pat3 data to explain the generation of *negative* rules. It is interesting to note that the weights $w^{(1)}_{kk_{\alpha_n}}$ connecting k_{α_n} nodes in the hidden layer with the corresponding kth output node are found to be negative, whereas those connecting k_{α_p} nodes are positive for each of the classes C_k. Therefore, when a pattern belonging

Table 7.6: Rules obtained by different models for Pat3 data.

Model	Antecedent	Consequent	
		Method (i)	Method (ii)
A	F_1 low and F_2 high	likely class 1 but unable to recognize class 2	very likely class 1
	F_1 medium and F_2 very high	mol likely class 1 but unable to recognize class 2	likely class 1 but unable to recognize class 2
	F_1 high and F_2 mol medium	likely class 1 but unable to recognize class 2	very likely class 1
	F_1 mol medium and F_2 mol medium	mol likely class 2 but unable to recognize class 1	likely class 2 but unable to recognize class 1
SN	F_1 low and F_2 high	very likely class 1	very likely class 1
	F_1 medium and F_2 very high	mol likely class 1 but unable to recognize class 2	likely class 1 but unable to recognize class 2
	F_1 high and F_2 mol medium	likely class 1 but unable to recognize class 2	very likely class 1
	F_1 mol medium and F_2 mol medium	likely class 2 but unable to recognize class 1	very likely class 2
F	F_1 low and F_2 high	very likely class 1	very likely class 1
	F_1 medium and F_2 very high	mol likely class 1 but unable to recognize class 2	likely class 1 but unable to recognize class 2
	F_1 high and F_2 mol medium	likely class 1 but unable to recognize class 2	likely class 1 but unable to recognize class 2
	F_1 mol medium and F_2 mol medium	likely class 2 but unable to recognize class 1	likely class 2 but unable to recognize class 1

Table 7.7: Rules obtained by different models for vowel data.

Model	Antecedent	Consequent	
		Method (i)	*Method (ii)*
AN	F_1 medium, F_2 low and F_3 low	very likely class 'o'	very likely class 'u'
	F_1 low, F_2 high and F_3 mol medium	very likely class 'i'	very likely class 'i'
	F_1 low, F_2 low and F_3 high	not unlikely class 'o'	mol likely class 'o' but unable to recognize class 'u'
	F_1 high, F_2 medium and F_3 very low	mol likely class '∂' but unable to recognize class 'a'	not unlikely class '∂'
	F_1 very high, F_2 low and F_3 very medium	mol likely class 'a' but unable to recognize class 'e'	mol likely class 'a' but unable to recognize class 'e'
	F_1 high, F_2 high and F_3 medium	likely class 'e' but unable to recognize class 'i'	very likely class 'e'
F	F_1 medium, F_2 low and F_3 low	very likely class 'o'	very likely class 'u'
	F_1 low, F_2 high and F_3 mol medium	likely class 'i' but not unlikely class 'e'	likely class 'i' but unable to recognize class 'e'
	F_1 low, F_2 low and F_3 high	mol likely class 'o' but unable to recognize class 'u'	mol likely class 'o' but unable to recognize class 'u'
	F_1 high, F_2 medium and F_3 very low	likely class '∂' but unable to recognize class 'a'	likely class '∂' but unable to recognize class 'a'
	F_1 very high, F_2 low and F_3 very medium	mol likely class 'a' but unable to recognize class '∂'	mol likely class 'a' but not unlikely class '∂'
	F_1 high, F_2 high and F_3 medium	likely class 'e' but unable to recognize class 'i'	very likely class 'e'

Table 7.8: Rules obtained by different models for medical data.

Method	Antecedent	Consequent	
		AN	F
(i)	GOT mol low, GPT mol low, *LDH mol low, GGT very high,* BUN mol low, MCV high, *MCH high, TBil medium and* CRTNN mol medium	*likely class* ALD but unable to recognize class LC	likely class ALD but unable to recognize class C
	GOT high, GPT not low, LDH medium, GGT medium, BUN high, *MCV mol high,* MCH mol high, TBil not high and *CRTNN low*	very *likely class PH*	very likely class PH
	GOT mol low, GPT mol low, LDH medium, *GGT low,* BUN low, MCV mol high, MCH mol high, TBil mol low and CRTNN mol high	mol likely class *LC* but unable to recognize class ALD	*very likely class LC*
	GOT low, GPT low, LDH very low, GGT low, BUN medium, MCV medium, MCH medium, TBil mol low and CRTNN very low	not unlikely class *C*	not unlikely *class C*
(ii)	GOT low, GPT low, LDH low, GGT very high, BUN low, MCV high, MCH high, TBil very medium and CRTNN medium	likely class ALD but unable to recognize class LC	likely class ALD but unable to recognize class C
	GOT high, GPT mol medium, LDH medium, GGT very medium, BUN high, MCV mol medium, MCH high, TBil very low and CRTNN low	very likely class PH	very likely class PH
	GOT low, GPT low, LDH very medium, GGT low, BUN low, MCV very high, MCH high, TBil low and CRTNN high	mol likely class LC but unable to recognize class ALD	very likely class LC
	. GOT low, GPT low, LDH very medium, GGT low, BUN low, MCV medium, *MCH mol medium,* TBil low and CRTNN very medium	not unlikely class C	not unlikely class C

Table 7.9: Values of connection weights $w_{kk_a}^{(1)}$ for Pat3 data.

Hidden node	Class C_k	
k_α	$k=1$	$k=2$
1	8.241502	–
2	13.473857	–
3	13.286455	–
4	8.405158	–
5	-43.487919	–
6	–	16.488053
7	–	16.458573
8	–	-9.280226
9	–	-14.061259
10	–	-9.116443

to class $C_{k'}$ is presented to the input layer of the network, the output produced by k_{α_n} hidden nodes is greater than those by k_{α_p} hidden nodes (or sometimes comparable in magnitude when the weights $w_{kk_{\alpha p}}^{(1)}$ and $w_{kk_{\alpha n}}^{(1)}$ are also comparable). But in such cases, the output produced by k'_{α_p} nodes is always found to be greater than those by k'_{α_n} nodes. This can easily be verified from Table 7.9. The entries of the table are the values of $w_{kk_\alpha}^{(1)}$. The hidden nodes 1, 2, 3 and 4 correspond to the intervals to which class 1 belongs, while node 5 refers to the interval in which class 1 does not lie. Similarly, hidden nodes 6 and 7 correspond to the intervals to which class 2 belongs, while the hidden nodes 8, 9 and 10 are indicative of the region where class 2 is not included. Considering this, we backtrack along k'_{α_n} nodes while determining a rule about a pattern not belonging to class $C_{k'}$ and generate that path having the maximal value for the magnitude of the product term (as explained in Section 7.3.2 for *negative* rules).

Table 7.10 provides the *negative* rules obtained by method (ii) using the AN and F models for vowel data. Rules with the same antecedents are not always generated by the two models corresponding to the same test pattern due to the different amounts of redundancy inherent in them (as observed earlier). It is seen that the *negative*

Table 7.10: Negative rules obtained by different models for vowel data.

Antecedent		Consequent
Model AN	Model F	
F_1 low, F_2 low and F_3 high	F_1 mol medium, F_2 low and F_3 high	not in class '∂'
F_1 mol medium, F_2 low and F_3 mol low	F_1 very medium, F_2 high and F_3 mol medium	not in class 'a'
F_1 very high, F_2 mol medium and F_3 very medium	F_1 very high, F_2 mol medium and F_3 very medium	not in class 'i'
F_1 mol low, F_2 medium and F_3 very low	F_1 high, F_2 medium and F_3 high	not in class 'u'
F_1 low, F_2 low and F_3 very medium	F_1 mol medium, F_2 low and F_3 very medium	not in class 'e'
F_1 mol medium, F_2 very medium and F_3 low	F_1 mol medium, F_2 very medium and F_3 low	not in class 'o'

Table 7.11: Negative rules obtained by different models for medical data.

Antecedent		Consequent
Model AN	Model F	
GOT low, GPT low, LDH very medium, GGT low BUN low, MCV medium, MCH mol medium, TBil low and CRTNN very medium	GOT low, GPT low, LDH very medium, GGT low BUN mol medium, MCV medium, MCH mol medium, TBil mol low and CRTNN very medium	not in class ALD
GOT high, GPT high, LDH mol medium, GGT high, BUN modium, MCV high, MCH high, TBil very medium and CRTNN low	GOT low, GPT low, LDH low, GGT medium, BUN very low, MCV medium, MCH mol medium, TBil mol medium and CRTNN very medium	not in class PH
GOT low, GPT low, LDH low, GGT medium, BUN very low, MCV not low, MCH not low, TBil very low and CRTNN very medium	GOT mol low, GPT high, LDH mol medium, GGT high, BUN medium, MCV high, MCH mol low, TBil very medium and CRTNN low	not in class LC
GOT low, GPT low, LDH very medium, GGT low, BUN low, MCV not low, MCH high, TBil low and CRTNN high	GOT mol low, GPT low, LDH very medium, GGT low, BUN low, MCV very high, MCH mol high, TBil low and CRTNN low	not in class C

rules offer an useful solution in cases where no suitable *positive* rule can be found. Note that model F does not have k_{α_p} or k_{α_n} nodes encoded in its structure. We have provided the negative rules in this case by just backtracking along the maximal magnitude paths from the class producing the minimal output. The rules are provided as an extension to the approach of [146] while also enabling us to make a comparative study. In Table 7.11 we depict the *negative* rules generated by method (ii) with AN and F models for medical data.

7.5 Conclusions and Discussion

In this chapter a new methodology of knowledge encoding among the connection weights of a fuzzy MLP [167] is described. This enables the network to perform classification and rule generation more efficiently. It involves development of a technique for generating an appropriate architecture of the fuzzy MLP [167] in terms of hidden nodes and links. Node growing and link pruning are used to enhance performance. It is found that the knowledge-based classification leads to better result than those of the conventional and fuzzy versions of the MLP [167, 146], and the fuzzy min-max neural network [206].

During learning an MLP searches for the set of weights that corresponds to some local minima. There may be a large number of such minimum values corresponding to various *good* solutions. The knowledge-based network initially considers these weights so as to be near one such *good* solution. As a result, the searching space gets reduced and learning becomes faster. Note that unlike the other methods [57, 215], the knowledge encoding technique involves nonbinary weighting mechanism based on the domain knowledge of a data set. The incorporation of fuzziness at various levels also helps the model to efficiently handle uncertain and ambiguous information both at the input and the output.

Conventional and fuzzy versions of the MLP consider empirically determined fixed architecture, whereas the knowledge-based model automatically determines it. The fuzzy min-max network [206] generates hidden nodes from some empirically determined parameter values. It is observed that this network requires larger number of links than the knowledge-based model to generate more or less the same recognition

score.

The knowledge-based network is seen to exhibit better generalization capability than the case-based network. The number of connection weights in the knowledge-based network is much less than those in the case-based network. On the other hand, the number iterations required to train the former is much more than that is needed for the latter.

The system is capable of generating both *positive* and *negative* rules in linguistic form to justify any decision reached. These rules are found to be useful for inferencing in ambiguous cases. Note that the rule generation algorithms described in this chapter are different from that derived from fuzzy MLP in [146]. A comparative study with that algorithm has also been provided to support this. It is observed that the less redundant knowledge-based model yields better rules much earlier. The concept of negative rules has been introduced to handle situations where a pattern does not belong to a specific class with high certainty. In such ambiguous situations, the complementary case of a pattern certainly not belonging to a class is considered to provide an appropriate explanation.

Chapter 8

Conclusions and Scope for Further Research

8.1 Conclusions

The thesis dealt with fuzzy set theoretic and neural approaches, both in individual and integrated manner, to certain tasks of pattern recognition. The problems considered include feature selection and extraction, classification, and rule generation. Various new neural network models and algorithms, incorporating the concept of fuzzy sets, have been designed/formulated for these purposes. Feature evaluation is done under both supervised and unsupervised modes of learning. The problem of classification is dealt with both case-based and knowledge-based networks, while the issue of rule generation is considered with the latter. Different features of the methodologies, along with comparisons with those of the related ones, are demonstrated extensively on both artificially generated and real life data sets. These data have dimension ranging from two to eighteen, and class boundaries highly nonlinear to overlapping nature.

The fuzzy set theoretic approach in Chapter 2 involves formulation of a feature evaluation index based on the concept of fuzzy entropy. The index is such that its value decreases with the increase/decrease in interclass/intraclass distances measured by the fuzzy entropy. Therefore, a feature (or a feature subset) is said to be the best for which the evaluation index is minimum. In the MLP based approach (of Chapter 2), the underlying principle is that the higher the importance of a feature (or feature subset), the more is its impact on the output of the network. Accordingly, different indices have been defined in terms of outputs of a trained MLP in the presence and absence of features. A feature (or feature subset), for which the index value is optimum (maximum or minimum, as applicable), is considered to be the best.

The neuro-fuzzy methods for feature selection and extraction of Chapters 3–5 are based on the principle of minimizing certain fuzzy feature evaluation indices in connectionist framework, under both supervised and unsupervised learning. The index, used for the supervised feature selection, utilizes the information on the class labels and is defined based on the aggregated measure of intraclass and interclass distances in terms of class membership functions. On the other hand, for unsupervised feature selection, the definition of the index does not involve the information on the class labels. It is formulated based on the degree of similarity between a pair of patterns with respect to their belongingness to a cluster. A concept of flexible membership

function incorporating weighting coefficients is introduced. These coefficients make the modeling of class/cluster structures more appropriate. Two different layered networks have been designed for computation and minimization of the respective indices; thereby determining the optimum weighting coefficients representing the importance of different features.

The method of feature extraction (Chapter 5) uses the same fuzzy feature evaluation index as used for unsupervised feature selection, but in a different connectionist framework. The layered network designed for this purpose embeds the tasks of applying a set of linear transformation functions on the original feature space, computing the membership functions in both original and transformed spaces, and minimizing the said index through unsupervised learning. The number of nodes in the second hidden layer determines the number of extracted features. The system results in a set of optimum transformed (extracted) features along with their relative importance.

In case-based classification system (Chapter 6), various cases have been selected from the training set using the concept of fuzzy similarity during training of a layered network. The cases are stored as the parameters (weights or memory elements) of the network with adaptive architecture which is determined through *growing* and *pruning* of hidden nodes during its training.

The knowledge-based fuzzy MLP (Chapter 7) accepts input in linguistic terms *viz.*, low, medium, high, and provides output in terms of class membership values. The links between the input and hidden layers are estimated, in terms of *a priori* class information and pattern distribution in the feature space, during encoding of crude domain knowledge from the data set. The weights of the links between the hidden and output layers are estimated in terms of preceding layer link weights and the activations of various nodes. Concepts of *positive* and *negative* hidden nodes are introduced, which correspond to the belongingness of a pattern *to a class* and *not to a class*. The network is trained to refine its weights. Node growing/link pruning is incorporated (whenever necessary) to generate the optimal architecture of the network.

Linguistic rules have been generated in *If-Then* form (i) through forward pass using both linguistic (natural) and numerical inputs for the antecedent clauses and the network output for the consequent part, and (ii) by backtracking from the output layer

to the input layer along the maximum weighted paths through *positive* or *negative* hidden nodes. It is interesting to note that the model is capable of handling input in numerical, linguistic and set forms, and can tackle uncertainty due to overlapping classes. Negative rules, indicative of cases where a pattern does not belong to a class, can also be generated. This is specially suitable in the ambiguous cases where positive rules (dealing with the notion of belongingness of a pattern to a particular class) cannot be obtained.

The neuro-fuzzy systems, developed for feature evaluation (Chapters 3–5), may be included in category 5 of neuro-fuzzy integration described in Section 1.5 of Chapter 1. On the other hand, the case-based (Chapter 6) and knowledge-based systems (Chapter 7) belong to categories 2 and 1 respectively.

The MLP-based feature selection method, described in Chapter 2, has been found to provide better performance than the MLP-based method of Ruck *et al.* [191], in terms of both class structures (as demonstrated by scatter plots) and classification performance (as demonstrated by MLP classifier). Its superiority over a statistical method [49] and a fuzzy set theoretic approach [152] has also been established in the same line. Unlike $(FEI)^{av}$ of Pal [152], the fuzzy feature evaluation index ($OFEI$), described in Chapter 2, is not biased towards equiprobable classes.

The fuzzy feature evaluation index (E) defined in Chapter 3 for feature selection, decreases with the increase in Mahalanobis distance and divergence between the classes. It performs better than $(FEI)^{av}$ of Pal [152], when the scatter plots and k-NN classifier are used as reference. Moreover, the computation of $(FEI)^{av}$ needs $M(M + 1)$ membership functions and hence $M(M + 1)$ parameters for an M-class problem. On the other hand, this is only $2M$ for index E. Note that in the aforesaid MLP-based and fuzzy set theoretic methods based on $OFEI$ and E, the features are assumed to be independent.

The index E is theoretically shown to have a fixed upper bound and a varying lower bound. It has been established for different cases that E monotonically increases with the lower bound. It is also shown that the higher the interclass distances, the greater is the chance of the network in getting converged. The order of individual importance of the features, as obtained by the neuro-fuzzy method in Chapter 3, has conformed well to those obtained by E *alone* for both vowel and Iris data. For

medical data the former method performs better, whereas the reverse is true for the latter method, as indicated by k-NN classifier.

Although the feature selection method of Chapter 4 is unsupervised, it has resulted in the order of individual feature for vowel and Iris data in accordance with the outcome of the supervised methods described in Chapters 2 and 3, and in [152, 209]. This algorithm, unlike the one in [20], provides ranking of the individual features without clustering of the feature space explicitly and does not need to assume the number of clusters present in the feature space. Note that the connectionist approaches (described in Chapters 3 and 4) consider all the features together, in order to determine their relative importance. In other words, the interdependencies of the features have been taken into account.

As in Chapters 3 and 4, this neuro-fuzzy feature extraction method of Chapter 5 assumes interdependencies of the original features. The classification scores in the extracted feature space are seen to be always higher than those in the original one of the same dimension. Sometimes it is better even than those in the entire original space. It is shown using the scatter plots and fuzzy c-means algorithm [20] that the extent of overlapping region in the extracted plane is less than that of a principal component analysis network (PCAN) [190] which is also meant for unsupervised feature extraction. The neuro-fuzzy method is also found to preserve the data structure, cluster shape and inter pattern distances better than PCAN. (Note that the scatter plots obtained by the PCAN and Sammon's nonlinear discriminant analysis (NDA) network are alike [135].) The neuro-fuzzy feature extraction method directly maps an n-dimensional original feature space into an n'-dimensional transformed space, where n' is the number of nodes in the second hidden layer. On the other hand, for PCAN, this task involves projection of the n-dimensional original space to an n-dimensional transformed space followed by selection of the best n' number of transformed components. However, both the methods do not provide clustering of the feature space explicitly, and also need not require the information on the number of clusters present in the feature space.

For case-based method of classification (Chapter 6), the number of hidden nodes increases with the decrease in extent λ (Eqn. (6.1) of Chapter 6) of the fuzzy region around a case. As λ decreases, the performance of the network during training, as expected, increases because of the higher number of representative cases. The same

is also true, during testing, upto a certain value, beyond which the performance deteriorates because of *overlearning* (poor generalization capability of the cases). It is seen that the performance of the network is superior to those of k-NN and Bayes maximum likelihood classifiers for both the vowel and synthetic (with concave class structure) data. However, it is the reverse for medical data.

The recognition capability of the knowledge-based system (Chapter 7), is found to be improved appreciably with the encoding of the initial knowledge in the network architecture. Node growing and link pruning enhance the performance. It is found that the knowledge-based system with less number of links achieves better classification performance than the conventional and fuzzy versions [167] of MLP, and fuzzy min-max neural network [206] with more number of links.

During learning, an MLP searches for the set of weights that corresponds to some local minima. There may be a large number of such minimum values corresponding to various *good* solutions. The knowledge-based network initially considers these weights so as to be near one such *good* solution. As a result, the searching space gets reduced and learning becomes faster. Note that, unlike the other methods [57, 215], the knowledge encoding technique involves nonbinary weighting mechanism based on the domain knowledge of a data set. The incorporation of fuzziness at various levels also helps the model to efficiently handle uncertain and ambiguous information both at the input and the output.

Conventional and fuzzy versions of the MLP consider empirically determined fixed architecture, whereas the knowledge-based model automatically determines it. The fuzzy min-max network [206] generates hidden nodes from some empirically determined parameter values. It is observed that this network requires larger number of links than the knowledge-based model to generate more or less the same recognition score.

The knowledge-based system has been able to provide better generalization capability than the case-based network (Chapter 6). Although the number of epochs required to train a knowledge-based network is much higher than that for the case-based network, the number of connections (links) in the former one is less than that in the latter.

The model is capable of generating both *positive* and *negative* rules in linguistic form

to justify any decision reached. These rules are found to be useful for inferencing in ambiguous cases. Note that the rule generation algorithms described in Chapter 7 are different from that derived from fuzzy MLP in [146]. It is observed that the less redundant knowledge-based model yields better rules much earlier. The introduction of the concept of negative rules enables one to handle situations where a pattern does not belong to a specific class with high certainty. In such ambiguous situations, the complementary case of a pattern certainly not belonging to a class is considered to provide an appropriate explanation.

8.2 Scope for Further Research

The fuzzy feature evaluation indices, defined in Chapters 2 and 3, involve computation of class means and bandwidths which are computed from the training set of samples with known class information. A study may be made where these parameters are allowed to evolve adaptively; thereby making the algorithms unsupervised. The method of unsupervised feature selection in Chapter 4 has been extended in Chapter 5 for performing the task of feature extraction under unsupervised mode. Similar extension may be made from algorithm in Chapter 3 for performing the task of feature extraction under supervised mode. Neuro-fuzzy systems, described in Chapters 3 and 4, basically provide an ordering of the individual features according to their importance. There is a scope for further investigation to obtain an ordering of different feature subsets by introducing several hidden layers in the systems. The tasks of classification and clustering may also be performed using these neuro-fuzzy systems after appropriate modifications.

We have provided a theoretical analysis of the supervised feature selection method. Similar analysis may be done for unsupervised feature selection and extraction methodologies. Note that the connectionist models, described in Chapters 3–5, basically perform the optimization tasks. Some other optimization techniques, e.g., genetic algorithms, simulated annealing may also be used here to find their suitability.

Chapters 6 and 7 dealt with the case-based and knowledge-based systems for classification in neuro-fuzzy framework. An integration of the underlying concepts of these two systems in the said framework may be made for performing a more intelligent

decision making. Such a system would have all the merits of knowledge-based and case-based reasoning methods, and also the generic and application specific advantages of neural networks and fuzzy sets; thereby making it more efficient. The tasks that may be included in this investigation include, extraction/encoding of domain knowledge, selection/storage/retrieval of cases and design of training algorithms for the purpose of feature evaluation, classification and rule generation. Such a hybrid system would have an wide range of real life applications e.g., medical diagnosis, natural resource estimation from remote sensing images, finger print analysis, speech recognition, data mining.

In case-based classification (Chapter 6), different pattern points from various classes are selected and stored as cases. Representation of cases in terms of intervals of features may constitute another part of the investigation. Moreover, some methodologies for rule generation may be developed for this case-based system. The knowledge-based system (Chapter 7) considers various intervals of the feature values for encoding initial domain knowledge. Instead, various rules, in *If-Then* form, may be used. In that case the system should be able to refine the existing set of rules after its training. The degree of importance of the various input features may also be exploited in this regard. The design of both feature selection and extraction methodologies, using the concepts of case-based and knowledge-based reasoning, would be an area of further research utilizing the existing knowledge and the past experience from the cases.

The fuzzy membership functions considered in this thesis are suitable to represent convex classes. Formulation of new membership functions for representing concave (convex being a special case) class structures therefore constitutes a part of future study. Moreover, it would be interesting to see if the exponential time complexities of the algorithms in Chapter 2 can be reduced to polynomial time.

Appendix A

A Statistical Approach to Feature Selection [49]

Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ be a pattern vector in the n-dimensional measurement space. There are M classes $C_1, C_2, \ldots, C_M$ with *a priori* class probability P_k for class C_k. Let the number of samples in class C_k be s_k and the total number of samples be s. Let $\mathbf{y} = [y_1, y_2, \ldots, y_{n'}]$ be the corresponding pattern vector in an n'-dimensional feature space, obtained by some feature selection technique.

Let a criterion function for ranking the features be defined as [49],

$$J = \frac{tr(\mathbf{S}_b)}{tr(\mathbf{S}_w)}, \tag{A.1}$$

where

$$\mathbf{S}_w = \sum_{k=1}^{M} P_k \frac{1}{s_k} \sum_{p}^{s_k} (\mathbf{y}_{pk} - \mathbf{m}_k)(\mathbf{y}_{pk} - \mathbf{m}_k)^t \tag{A.2}$$

and

$$\mathbf{S}_b = \sum_{k=1}^{M} P_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^t. \tag{A.3}$$

The term $\mathbf{m}_k$, sample mean vector of kth class, is

$$\mathbf{m}_k = \frac{\sum_{p=1}^{s_k} \mathbf{y}_{pk}}{s_k}, \tag{A.4}$$

and the mixture sample mean, $\mathbf{m}$, is

$$\mathbf{m} = \sum_{k=1}^{M} P_k \mathbf{m}_k. \tag{A.5}$$

An estimate of *a priori* class probability P_k for class C_k may be taken as

$$P_k = \frac{s_k}{s}. \tag{A.6}$$

From Eqns. (A.1)–(A.6), it can be seen that the value of J increases with the increase in $tr(\mathbf{S}_b)$ (*i.e.*, with the increase in interclass distance) and decrease in $tr(\mathbf{S}_w)$ (*i.e.*, with the decrease in intraclass distance). Therefore, the objective is to select those features for which J is maximum.

Appendix B

Feature Ranking Using Fuzzy Entropy [152]

Let $X = \{x_1, x_2, ..., x_s\}$ be a universe of discourse and a fuzzy set $\mathcal{A} = \{\mu_{\mathcal{A}}(x_i)/x_i | x_i \in X; p = 1, 2, ..., s; \mu_{\mathcal{A}} \in [0, 1]\}$ be defined on X where $\mu_{\mathcal{A}}(x_i)$ denotes the membership of x_i to $\mathcal{A}$. A measure of fuzziness for $\mathcal{A}$ can be defined as [151]

$$H(\mathcal{A}) = \frac{1}{s \ln 2} \sum_{p=1}^{s} [-\mu_{\mathcal{A}}(x_p) \ln(\mu_{\mathcal{A}}(x_p)) - (1 - \mu_{\mathcal{A}}(x_p)) \ln(1 - \mu_{\mathcal{A}}(x_p))] \qquad \text{(B.1)}$$

$H(\mathcal{A})$ is called entropy of the fuzzy set $\mathcal{A}$. $H(\mathcal{A})$ attains the maximum value when $\mathcal{A}$ is most fuzzy, *i.e.*, when $\mu_{\mathcal{A}}(x_p) = 0.5, \ \forall p$, and it attains the minimum value when $\mu_{\mathcal{A}}(x_p) = 0$ *or* 1, $\forall\ p$.

Pal [152] used S-type and π-type [166] membership functions for modeling of μ. In order to explain the algorithm, let us consider the standard S-function, defined as [166],

$$\begin{aligned} \mu_{\mathcal{A}}(x_p; a, b, c) &= 0, && x_p \leq a \\ &= 2[\tfrac{x_p - a}{c - a}]^2, && a \leq x_p \leq b \\ &= 1 - 2[\tfrac{x_p - c}{c - a}]^2, && b \leq x_p \leq c \\ &= 1, && x_p \geq c \end{aligned} \qquad \text{(B.2)}$$

in the interval $[a, c]$ with $b = (a + c)/2$. The parameter b is known as the crossover point for which $\mu_{\mathcal{A}}(b; a, b, c) = 0.5$.

For computing H of class C_k along ith feature, the parameters of the S-function can be computed as [152]

$$b = (x_{ik})_{av}, \qquad \text{(B.3)}$$

$$c = b + \max\{|(x_{ik})_{av} - (x_{ik})_{max}|, |(x_{ik})_{av} - (x_{ik})_{min}|\}\ , \qquad \text{(B.4)}$$

and

$$a = 2b - c\ . \qquad \text{(B.5)}$$

Here av, max and min are used to denote the average, maximum and the minimum value of x_{ik} respectively. If each x_{ik} is equal to b, H will be maximum and equal to 1. H tends to zero as x_{ik} moves away from b towards either c or a. The higher the value of H, the greater would be the number of samples having $\mu(x) \approx 0.5$ and hence greater would be the tendency of the samples to cluster around its mean value, resulting in less (internal) scatter within the class. If we pool together classes C_k and $C_{k'}$ and compute the mean, maximum and minimum values of ith feature over all $(s_k + s_{k'})$ samples where $s_r (r = k, k')$ is the number of samples of class C_r, H for the

pooled sample would decrease as the goodness of feature increases. This is because, for a good feature, the samples from both classes should be away from the overall mean, *i.e.*, most of the points will have $\mu(x) \approx 0$ *or* 1. The feature evaluation index for feature i (FEI_i) corresponding to classes C_k and $C_{k'}$ is defined as [152]

$$FEI_i = \frac{H_{ikk'}}{H_{ik} + H_{ik'}} \tag{B.6}$$

where $H_{ikk'}$ is the value of the entropy for ith feature after pooling the classes $C_{k'}$ and C_k; H_{ik}, $H_{ik'}$ are those for the ith feature computed for C_k or $C_{k'}$ respectively. The lower the value of FEI_i, the higher is, therefore, the quality of ith feature in characterizing and discriminating classes C_k and $C_{k'}$. Instead of using only one ith feature, FEI can be calculated even for a set of features [152]. In this case, one needs to use the multidimensional membership function [162]. Considering all the classes, the average feature evaluation index for a set of features S may be defined as [152]

$$(FEI)_S^{av} = \sum_k \sum_{k'} W_k W_{k'} (FEI)_S^{(kk')} \tag{B.7}$$

where $W_k = \frac{s_k}{s}$, $W_{k'} = \frac{s_{k'}}{s}$, $s = \sum_k s_k$, $k, k' = 1, 2, ..., M; k \neq k'$, are weight factors.

Appendix C

Multilayer Perceptron (MLP) Model [192, 168]

An MLP (Fig. C.1) is a classifier network, capable of learning an input-output relation. MLP consists of several layers of processing elements called nodes or neurons. There is no connection between nodes within a layer but complete connections exist between nodes of successive layers. The layer of nodes which receives inputs from outside is called the input layer and the layer that produces output is called the output layer. In between input and output layers there are several layers called hidden layers. The number of nodes in the input layer is the same as the dimension of input data, and that in the output layer is the same as the number of pattern classes. The nodes in the hidden layers receive inputs from its preceding layer and produce outputs which become input to the nodes of the next layer. There is no computation in the input layer. Nodes in other layers receive inputs, which are functions of the outputs of the nodes in the previous layer and the connection weights between the two layers, and apply a nonlinear transformation (activation function) to produce the output.

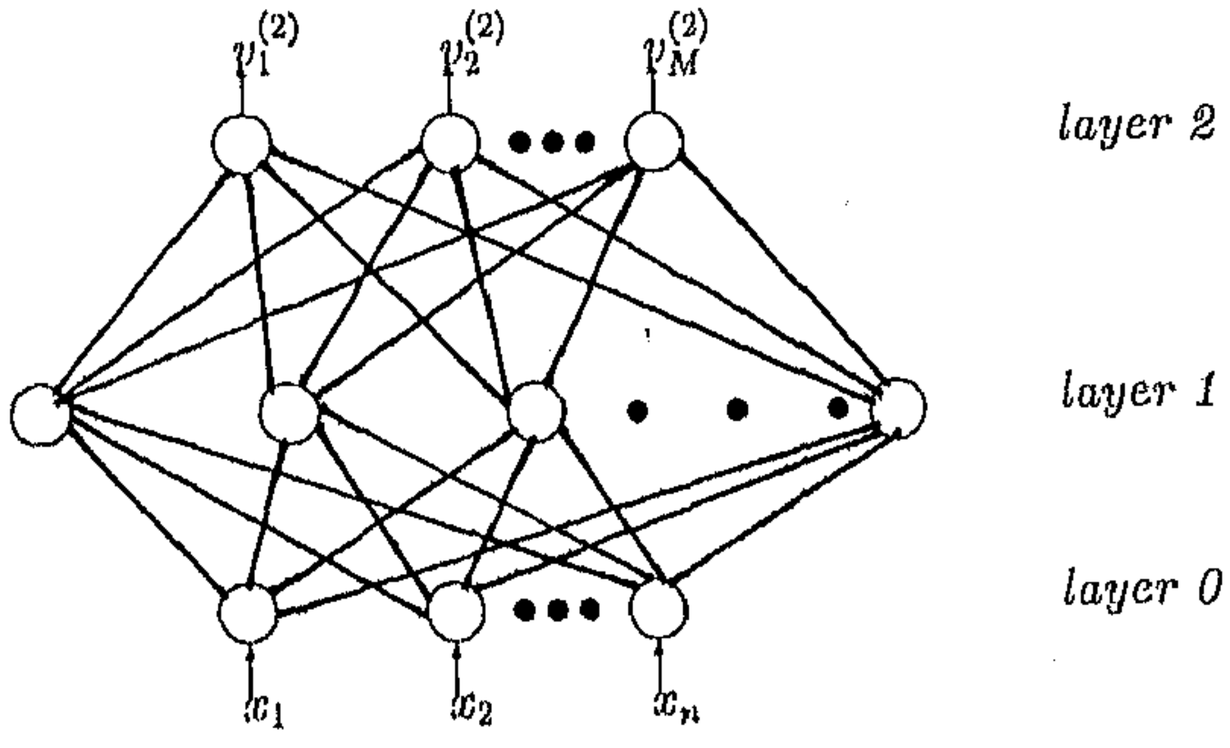

Figure C.1: A schematic diagram of a multilayer perceptron (MLP) model.

The total input received by jth node of layer $(h+1)$ $(h \geq 0)$ is

$$u_j^{(h+1)} = \sum_i w_{ij}^{(h)} v_i^{(h)}. \tag{C.1}$$

Here $v_i^{(h)}$ is the output of ith node in layer h ($h = 0$ corresponds to the input layer) and $w_{ji}^{(h)}$ is the connection weight between jth node in layer $(h+1)$ and ith node of layer h. The output of ith node in layer $h > 0$ is $v_i^{(h)} = g(u_i^{(h)})$, g is the activation function. Mostly sigmoidal activation functions are used where

$$g(y) = \frac{1}{1 + exp(-y)}; \tag{C.2}$$

y being the total input received by a node.

In the learning phase (training) of such a network, we present a pattern $\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ as input and adjust the set of weights in the connecting links such that the desired output $\mathbf{t} = [t_1, t_2, \ldots, t_k, \ldots, t_M]$ is obtained at the output nodes. The process is repeated until the weights stabilize.

In general, for the output layer (Hth layer) the actual output of kth node $o_k = v_k^{(H)}$ is not the same as the corresponding target or desired value t_k. Thus, for a pattern vector, the error is

$$e_{\mathbf{x}} = \sum_k (t_k - o_k)^2 \tag{C.3}$$

and the total error is

$$E = \sum_{\mathbf{x}} e_{\mathbf{x}}. \tag{C.4}$$

An MLP attempts to minimize E by moving in the direction of negative gradient of the instantaneous error $e_{\mathbf{x}}$. In other words, the incremental change $\Delta w_{kj}^{(h)}$ is taken as proportional to $(-\frac{\partial e_{\mathbf{x}}}{\partial w_{kj}^{(h)}})$ for a particular pattern $\mathbf{x}$, *i.e.*, $\Delta w_{kj}^{(h)} = -\eta \frac{\partial e_{\mathbf{x}}}{\partial w_{kj}^{(h)}}$, where η is the learning rate. After some algebraic manipulation the learning rule becomes:

$$\begin{aligned} \Delta w_{kj}^{(h)} &= -\eta (\tfrac{\partial e_{\mathbf{x}}}{\partial v_k^{(h+1)}}) g'(u_k^{(h+1)}) v_j^{(h)} && \textit{if layer } (h+1) \textit{ is output layer}, \\ &= -\eta (\sum_i \frac{\partial e_{\mathbf{x}}}{\partial u_i^{(H)}} w_{ik}^{(h)}) g'(u_k^{(h+1)}) v_j^{(h)} && \textit{for other layer}. \end{aligned} \tag{C.5}$$

The incremental changes $\Delta w_{ji}^{(h)}$ may be summed up for all the patterns in the training set and then the weights $w_{ji}^{(h)}$ are updated with the resulting increments. This process is called *batch mode* training. In this strategy, the learning process remains independent of the sequence in which the training data are fed. Normally, a complete pass through the training data is known as 'epoch' or 'iteration'. On the other hand, in *on-line* training weights are updated with each pattern. In this case, learning depends upon the sequence of data feeding. We have adopted the batch mode learning

in our experiment. Thus, the expression for the updated weight after t epochs is given by

$$w_{ji}^{(h)}(t+1) = w_{ji}^{(h)}(t) + \sum \Delta w_{ji}^{(h)}. \tag{C.6}$$

Appendix D

Feature Selection Method of Ruck et al. [191]

Ruck *et al.* [191] developed an algorithm for feature ranking using an MLP. The sensitivity of output of the network to its input is used to rank the input features. An expression for feature *saliency* (as proposed by them), used for feature ranking, is defined as

$$\Lambda_i = \sum_{\mathbf{x}\in\mathcal{S}} \sum_{k} \sum_{x_i\in D_i} \left|\frac{\partial o_k(\mathbf{x}, \mathbf{W})}{\partial x_i}\right| \tag{D.1}$$

where D_i is the domain of ith feature and $\mathcal{S}$ is the training set. The matrix $\mathbf{W}$ is an array of all connection weights in the network arranged in some suitable form. Ruck et al. have used the derivative as a sensitivity indicator of the network output with respect to the input feature. Therefore, $\Lambda_i > \Lambda_j$ is assumed to indicate that the importance of ith feature is higher than that of jth feature.

For evaluating $\frac{\partial o_k(\mathbf{x},\mathbf{W})}{\partial x_i}$ in Eqn. (D.1) one may use the chain rule for an MLP with one hidden layer (Appendix C)

$$\begin{aligned}
\frac{\partial o_k}{\partial x_i} &= o_k(1-o_k)\frac{\partial}{\partial x_i}\left(\sum_j w_{kj}^{(1)} v_j^{(1)} + \theta_k\right) \\
&= o_k(1-o_k)\sum_j w_{kj}^{(1)}\frac{\partial v_j}{\partial x_i} \\
&= o_k(1-o_k)\sum_j w_{kj}^{(1)} v_j(1-v_j) w_{ji}^{(0)}.
\end{aligned} \tag{D.2}$$

Here θ_k is the threshold for the sigmoidal activation function g. From Eqn. (D.2) we find that the derivative depends on the current input to the network as well as its weights. To calculate Λ_i, ideally, each input should be independently sampled over its expected range of values. For example, if R points are used for each input feature, the total number of points that the derivatives have to be evaluated at would be R^n, where n is the total number of features. Therefore, the problem is of exponential time complexity.

In order to reduce the computational load, Ruck *et al.* suggested to sample it at the most important points. The points of the greatest importance in the input space are those in which training data exist; hence, the training vectors are used as starting points to sample the input space. For every training vector, each feature is sampled over its range. Thus for s training vectors, the number of derivative evaluations is snR.

Appendix E

Principal Component Analysis Network of Rubner and Tavan [190]

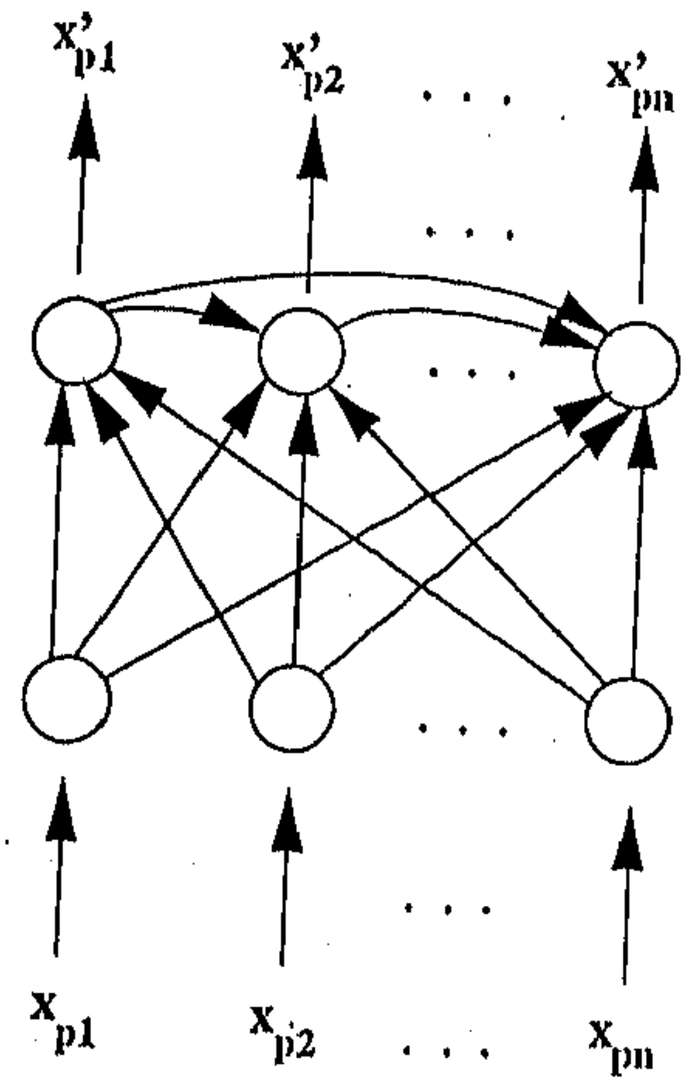

Figure E.1: A schematic diagram of the PCAN.

Principal component analysis is a well-known statistical method for feature extraction. It involves a linear orthogonal transformation from an n-dimensional feature space to an n'-dimensional space, $n' \leq n$, such that the features in the new n'-dimensional space are uncorrelated and maximal amount of variance of the original data is preserved by only a small number of features.

The principal component analysis network (PCAN) of Rubner and Tavan [190] performs principal component analysis in a connectionist framework. It consists of n input and output nodes. An ith input node is connected to a jth output node with connection weight w_{ij} (Fig. E.1). All the output nodes are hierarchically organized in such a way that an lth output node is connected to a jth output node via connection weight $w_{lj}^{(lat)}$ if and only if $l < j$. The training algorithm of the network is summarized below.

- Initialize all connection weights to small random values and choose the values of learning parameters.
- Repeat the following steps until all the lateral weights are sufficiently small for a given number of presentations (*i.e.*, until their absolute values are below some threshold).

– Randomly select an n-dimensional pattern $\mathbf{x}_p$ and present it to the input layer of the network. Compute the output $(\mathbf{x}'_p)$ of the network, using the equation

$$x'_{pj} = \mathbf{w}_j.\mathbf{x}_p + \sum_{l<j} w_{lj}^{(lat)} x'_{pl}, \quad j = 1, 2, \ldots, n. \tag{E.1}$$

– Update w_{ij}, $\forall i, j$ following the Hebbian rule,

$$\Delta w_{ij} = \eta_1 x_{pi} x'_{pj}, \tag{E.2}$$

where $\eta_1 > 0$ is the learning rate.

– Normalize w_{ij} in such a way that $||\mathbf{w}_j|| = 1$.

– Update $w_{lj}^{(lat)}$ by the anti-Hebbian rule,

$$\Delta w_{lj}^{(lat)} = -\eta_2 x'_{pl} x'_{pj}, \tag{E.3}$$

where η_2 is a positive learning parameter.

After convergence, the activations of the first n' $(\leq n)$ output nodes constitute pattern vectors in the n'-dimensional transformed space. It is expected that the features in the new n'-dimensional space are uncorrelated and maximal amount of variance of the original data is preserved by only n' number of these transformed features.

Appendix F

Fuzzy Multilayer Perceptron Model (MLP) for Classification and Rule Generation [167, 146]

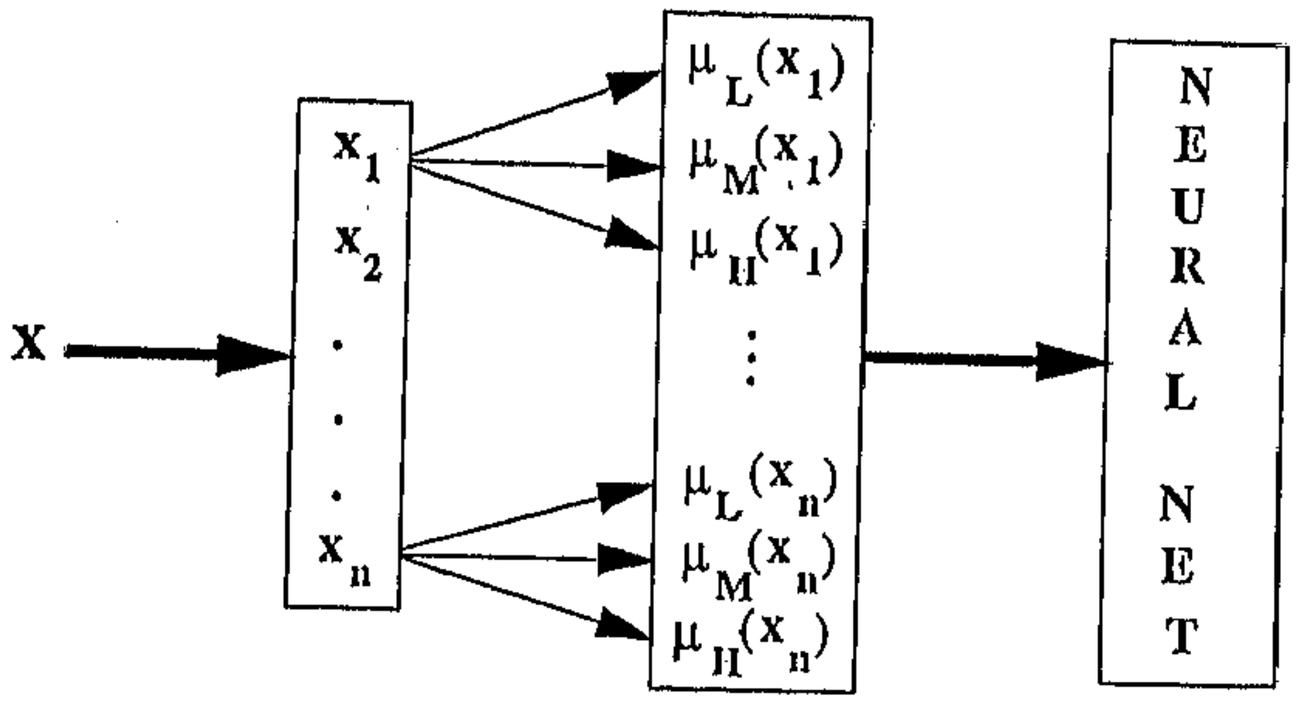

Figure F.1: A schematic diagram of the fuzzy multilayer perceptron model.

Fuzzy MLP (Fig. F.1) of Pal and Mitra [167] consists of several layers of processing elements called nodes, as in the case of conventional MLP (described in Appendix C). There is no connection between nodes within a layer but complete connections exist between nodes of successive layers. The layer of nodes which receives inputs from outside is called the input layer and the layer that produces output is called the output layer. The nodes in the input layer receive feature values in linguistic form of a pattern as input, while those in the output layer provide the degree of belongingness of the pattern to the various classes.

An n-dimensional pattern $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$ is represented as a $3n$-dimensional vector [159]

$$\left[u_1^{(0)}, u_2^{(0)}, \ldots, u_{3n}^{(0)}\right] = \left[\mu_{low(x_1)}(\mathbf{x}), \mu_{medium(x_1)}(\mathbf{x}), \mu_{high(x_1)}(\mathbf{x}), \ldots, \mu_{high(x_n)}(\mathbf{x})\right] \quad \text{(F.1)}$$

where the μ values indicate the membership functions of the corresponding linguistic π-sets along each feature axis.

When the input feature is numerical, the π-fuzzy sets (in one dimensional form) are

used, with range [0,1], represented as

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

where $\lambda > 0$ is the radius of the π-function with c as the central point.

The central points (c_{i_l}, c_{i_m} and c_{i_h}) and the radii (λ_{i_l}, λ_{i_m} and λ_{i_h}) of the π-functions corresponding to the fuzzy sets *low*, *medium* and *high* are computed from the training set. Let $x_{i_{max}}$ and $x_{i_{min}}$ denote the upper and lower bounds of the dynamic range of feature x_i for all the pattern points (in the training set) considering numerical values only. Let m_i be the mean of the pattern points along the ith axis. Then m_{i_l} and m_{i_h} are defined as the mean (along the ith axis) of the pattern points having co-ordinate values in the range $[x_{i_{min}}, m_i)$ and $(m_i, x_{i_{max}}]$ respectively. For the three linguistic property *low*, *medium* and *high* one may define the centers as

$$\begin{aligned} c_{i_l} &= m_{i_l} \\ c_{i_m} &= m_i \\ c_{i_h} &= m_{i_h} \end{aligned} \tag{F.3}$$

and the corresponding radii as

$$\begin{aligned} \lambda_{i_l} &= 2\,(c_{i_m} - c_{i_l}) \\ \lambda_{i_h} &= 2\,(c_{i_h} - c_{i_m}) \\ \lambda_{i_m} &= fnos * \frac{\lambda_{i_l}\,(x_{i_{max}} - c_{i_m}) + \lambda_{i_h}\,(c_{i_m} - x_{i_{min}})}{x_{i_{max}} - x_{i_{min}}} \end{aligned} \tag{F.4}$$

where *fnos* is a multiplicative parameter controlling the extent of the overlapping. Here the distribution of the pattern points along each feature axis is taken into account while choosing the corresponding centers and radii of the linguistic properties. This has been found to be quite efficient in modeling skewed data distributions [141]. It is also ensured that any feature value along ith axis for pattern x is assigned membership value combinations in the corresponding 3-dimensional linguistic space of Eqn. (F.1) in such a way that at least one of $\mu_{low(x_i)}(x_i)$, $\mu_{medium(x_i)}(x_i)$ or $\mu_{high(x_i)}(x_i)$ is greater than 0.5 in the interval $\left[c_{i_l} - \frac{\lambda_{i_l}}{2}, c_{i_h} + \frac{\lambda_{i_h}}{2}\right]$ which represents the relevant region of the feature space *sans* outliers.

The output of a node (node) in any layer $(h+1)$ other than the input layer is given as

$$v_j^{(h+1)} = \frac{1}{1 + exp\left(-\sum_i v_i^{(h)} w_{ji}^{(h)}\right)} \tag{F.5}$$

where $v_i^{(h)}$ is the state of ith node in the preceding hth layer and $w_{ji}^{(h)}$ is the weight of the connection from ith node in layer h to jth node in layer $(h+1)$. For nodes in the input layer, $v_i^{(0)}$ corresponds to ith component of the input vector. The mean square error in output vectors is minimized by the backpropagation algorithm using a gradient descent with a gradual decrease of the gain factor.

For determining the degree of belongingness of $\mathbf{x}$ to the various M classes (by the network), there are M nodes in the output layer. Let the n-dimensional vectors $\mathbf{o}_k = [o_{k1} \ldots o_{kn}]$ and $\mathbf{v}_k = [v_{k1} \ldots v_{kn}]$ denote the mean and standard deviation respectively of the numerical training data for kth class C_k. The weighted distance of the training pattern $\mathbf{x}$ from kth class C_k is defined as

$$z_k = \sqrt{\sum_{i=1}^{n} \left[\frac{x_i - o_{ki}}{v_{ki}}\right]^2} \text{ for } k = 1, \ldots, M \tag{F.6}$$

where x_i is the value of ith component of the pattern point.

The membership of $\mathbf{x}$ pattern in class C_k, lying in the range $[0, 1]$, is defined as [166]

$$\mu_k(\mathbf{x}) = \frac{1}{1 + \left(\frac{z_k}{f_d}\right)^{f_e}} \tag{F.7}$$

where positive constants f_d and f_e are the denominational and exponential fuzzy generators controlling the amount of fuzziness in this class-membership set.

Then, for $\mathbf{x}$, the desired output of kth output node is

$$d_k = \mu_k(\mathbf{x}) \tag{F.8}$$

According to this definition, a pattern can simultaneously belong to more than one class, and this is determined basically from the training set used during the learning phase. However, it may be noted that in the crisp case a pattern can either belong or not belong to a class. Then we have $z_k \in \{0, \infty\}$ in Eqn. (F.6), such that the output membership value of Eqn. (F.7) reduces to $\mu_k(\mathbf{x}) \in \{1, 0\}$. The network is trained using backpropagation algorithm, described in Appendix C, under supervised mode.

Classification

After the training phase is over, the degree of belongingness of an unknown sample to a class C_k is determined by the output of kth output node considering the sample as the input. The sample may be said to be in class C_K, if $v_K^{(H)} = \max_k\{v_k^{(H)}\}$.

Rule generation

The network may be used to produce justifications for its inference in the form of *If-Then* rule. The method of rule generation [146] is described below.

Forward pass

The M-dimensional output vector with components $v_k^{(H)}$ is computed using Eqn.(F.5) 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 M output classes. Associated with each node j in layer $h+1$ are also

- its confidence estimation factor

$$\begin{aligned} conf_j^{(h+1)} &= \left|\frac{\sum_i v_i^{(h)} w_{ji}^{(h)}}{unden_j^{(h+1)}}\right|, \quad if\ noinf_j^{(h+1)} = 1\ and\ h+1 > 0, \\ &= v_j^{(h+1)}, \qquad\qquad otherwise, \end{aligned} \tag{F.9}$$

where

$$unden_j^{(h+1)} = \sum_i |w_{ji}^{(h)}|, \tag{F.10}$$

- a variable $unknown_j^{(h+1)}$ defined by

$$unknown_j^{(h+1)} = \sum_i w_{ji}^{(h)} v_i^{(h)}, \tag{F.11}$$

such that $noinf_i^{(h)} = 1$, $\forall i$, where

$$\begin{aligned} noinf_i^{(h)} &= 1, \quad if\ |known_i^{(h)}| \leq |unknown_i^{(h)}|, \\ &= 0, \quad otherwise, \end{aligned} \tag{F.12}$$

with

$$known_i^{(h)} = \sum_m w_{im}^{(h-1)} v_m^{(h-1)}, \tag{F.13}$$

and

- a variable $known_j^{(h+1)}$ (Eqn. F.13)) giving the sum of the weighted information from the (remaining) *non-ambiguous* preceding nodes with $noinf_i^{(h)} = 0$.

Note that, for a node j in layer $h+1$ with no preceding nodes i tagged with $noinf_i^{(h)} = 1$, the system considers $unknown_j^{(h+1)} = 0$.

$conf_j^{(h)}$ is comparable either among the set of nodes having $noinf_j^{(h)} = 1$, or among those with $noinf_j^{(h)} = 0$, but not between the nodes belonging to these two different sets. In the output layer ($h = H$) if $noinf_j^{(H)} = 0$ then $conf_j^{(h)}$ is higher for nodes having larger $v_j^{(H)}$, implying a greater belongingness to output class j. If $noinf_j^{(H)} = 1$, then the confidence $conf_j^{(h)}$ decreases with the increase in $unden_j^{(h)}$ (Eqns. (F.10)) (absolute sum of connection weights from *ambiguous* preceding layer nodes), and vice versa.

If there is no output node j with $noinf_j^{(H)} = 1$, then the system finalizes 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. One may note that the weights (learned during training), of the network play an important part in determining whether a missing input feature information is *essential* to the final inferred decision or not. The certainty factor corresponding to kth output node, reflecting the difficulty in arriving at a particular decision in favor of kth class, may be defined as

$$bel_k^{(H)} = v_k^{(H)} - \sum_{j \neq k} v_j^{(H)}, \tag{F.14}$$

where $bel_k^{(H)} \leq 1$.

Justification

The user can ask the system for the justification for its inferred decision. The system answers with an *If-Then* rule applicable to the case at hand. Note that these rules are generated by the *inferencing system*, by backtracking, from the connection weights as needed for explanations. As the model has already inferred a conclusion (at this stage), a subset of the currently known information is selected to justify this decision. It is ensured that output nodes k with $bel_k^{(H)} > 0$ (or, large $v_k^{(H)}$ values) are chosen for obtaining the justification.

- Output layer:

Let the user ask for the justification about a conclusion regarding class k. 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, those nodes j in the preceding layer that have a positive impact on the conclusion at output node k are selected, such that $w_{kj}^{H-1} > 0$. Let the set of m_{H-1} nodes so chosen, be $\left\{a_1^{(H-1)}, a_2^{(H-1)}, \ldots, a_{m_{H-1}}^{(H-1)}\right\}$ and let their connection weights to node k in layer H be given as $\left\{wet_{a_1^{(H-1)}} = w_{ka_1}^{(H-1)}, \ldots, wet_{a_{m_{H-1}}^{(H-1)}} = w_{ka_{m_{H-1}}}^{(H-1)}\right\}$. For the remaining layers one obtains the *maximum weighted* paths through these nodes down to the input layer.

• Intermediate layers:

Select node j in layer $0 < h < H - 1$ if

$$\begin{aligned} v_j^{(h)} &> 0.5, \text{ and} \\ wet_{j(h)} &= \max_{a_k^{(h+1)}} \left[wet_{a_k^{(h+1)}} + w_{a_k j}^{(h)}\right], \end{aligned} \tag{F.15}$$

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

• Input layer:

Let the process of Eqn. (F.15) result in m_0 chosen nodes (paths) in (from) the input layer ($h = 0$). These nodes indicate inputs that are *known* and have contributed to the ultimate positivity of the conclusion at node k 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. (F.15) and the process terminates.

Let the set of selected m_0 input nodes be $\left\{a_1^{(0)}, a_2^{(0)}, \ldots, a_{m_0}^{(0)}\right\}$ and their corresponding path weights to node k in layer H be $\left\{wet_{a_1^{(0)}}, wet_{a_2^{(0)}}, \ldots, wet_{a_{m_0}^{(0)}}\right\}$. These nodes are arranged in the decreasing order of their *net impacts*, where

$$net\ \ impact_i = v_i^{(0)} * wet_{i(0)}. \tag{F.16}$$

Then the clauses for an *If-Then* rule are generated from this ordered list until

$$\sum_{i_s} wet_{i_s^{(0)}} > 2 \sum_{i_n} wet_{i_n^{(0)}}, \tag{F.17}$$

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

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

$|i_s|$, $|i_n|$ refer respectively to the number of nodes selected and remaining from the said set. This heuristic allows the *currently active* test pattern inputs (current evidence) to influence the generated *knowledge base* (connection weights learned during training) in producing the antecedent part of a rule to justify the *current* inference.

• Antecedent clause generation:

For a node i_{s_1} in the input layer, selected for clause generation, the corresponding input feature u_{s_1} is obtained. The antecedent of the rule is given in linguistic form with

$$\begin{aligned} prop &= low \quad if\ i_{s_1} - 3(u_{s_1} - 1) = 1 \\ &= medium \quad if\ i_{s_1} - 3(u_{s_1} - 1) = 2 \\ &= high \quad otherwise. \end{aligned} \tag{F.18}$$

A linguistic hedge *very, more or less* or *not* may be attached to the linguistic property in the antecedent part, if necessary. For this purpose, the mean square distance $d(u_{s_1}, pr_m)$ between the three components of feature u_{s_1} and *prop* (of Eqn. (F.18)) with or without modifiers, represented as pr_m, is used. The corresponding 3-dimensional values for pr_m are given by Eqn. (7.20) (with no modifiers) and by Eqns. (7.21)-(7.23) 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} (node i_{s_1}) for the rule justifying the conclusion regarding output node k.

This procedure is repeated for all the $|i_s|$ nodes selected by Eqn. (F.17) to generate a set of conjunctive antecedent clauses for the rule. Note that 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 $v_k^{(H)}$ to class k. However a more *natural* form of decision can also be provided for the class k, considering the value of $bel_k^{(H)}$ of Eqn. (F.14). For the linguistic output form, one may use

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

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

Appendix G

Fuzzy Min-Max Neural Network Model [206]

Fuzzy min-max neural network model (Fig. C.1) of Simpson is used for supervised classification. The network has three layers *viz.*, F_A, F_B and F_C layers. The nodes in F_A layer, called input nodes, receive the feature values of pattern $\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ as inputs. The nodes in F_B layer are termed as *hyperbox* nodes, while those in F_C layer called class nodes and represent various classes.

Each F_B node (hyperbox node) represents a hyperbox fuzzy set representing a region of a class. F_A to F_B connections are the min-max points. F_B transfer function is the hyperbox membership function which may be defined as

$$\begin{aligned} b_j(\mathbf{x}) \;=\; \frac{1}{2n}\sum_{i=1}^{n}[&\max(0, 1 - \max(0, \gamma \min(1, x_i - w_{ji}))) \\ &+ \max(0, 1 - \max(0, \gamma \min(1, w_{ji} - x_i)))]. \end{aligned} \tag{C.1}$$

$V_j = [v_{j1}, v_{j2}, \ldots, v_{jn}]$ and $W_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]$ are the min and max points respectively for jth F_B node B_j, and γ is the *sensitivity parameter* that regulates how fast the membership values decreases as the distance between $\mathbf{x}$ and B_j increases.

The membership function $0 \leq b_j(\mathbf{x}) \leq 1$ is used to measure the degree to which input pattern $\mathbf{x}$ falls outside of hyperbox B_j. This can be considered as a measurement of how far each component of $\mathbf{x}$ is greater (less) than the max (min) point value along each dimension that falls outside the min-max bounds of the hyperbox. As $b_j(\mathbf{x})$ tends to 1, the point should be more contained by the hyperbox, with the value 1 representing complete hyperbox containment. The connections w_{ji} and w_{ji} are adjusted using the supervised learning algorithm described below.

The connection strength between jth F_B node and kth F_C node is binary valued as given by

$$\begin{aligned} u_{jk} \;&=\; 1, \quad \textit{if } B_j \textit{ corresponds to a hyperbox for class } C_k \\ &=\; 0, \quad \textit{otherwise.} \end{aligned}$$

The output of kth F_C node is

$$\mu_k = \max_{j=1}^{m}(b_j u_{jk}) \tag{C.2}$$

where is m is number of F_B nodes. μ_k denotes the degree of belongingness of $\mathbf{x}$ to kth class C_k.

Fuzzy min-max classification learning algorithm is a three-step process and can be stated as –

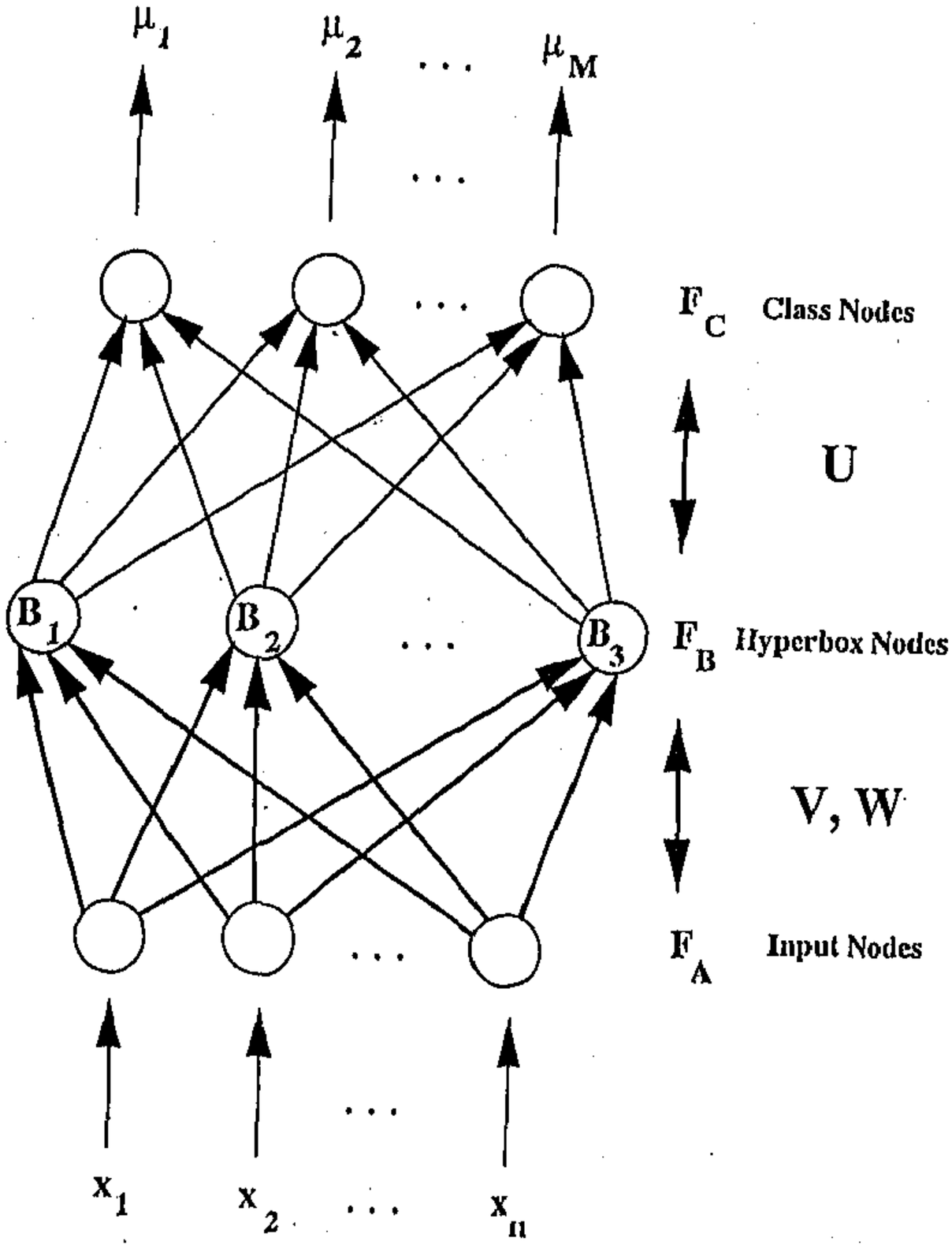

Figure G.1: A schematic diagram of the fuzzy min-max neural network model.

1) *Hyperbox expansion*: Identify the hyperbox that can expand and expand it. If an expandable hyperbox cannot be found, add a new hyperbox for that class.
2) *Hyperbox overlap test*: Determine if any overlap exists between hyperboxes from different classes.
3) *Hyperbox contraction*: If overlap between hyperboxes representing different classes does exist, eliminate the overlap by minimally adjusting each of the hyperboxes.
We now describe these operations.

Hyperbox expansion

A pattern $\mathbf{x}$ from class C_k is presented to F_A layer of the network. A hyperbox B_j that provides the highest $b_j(\mathbf{x})$ value, is allowed to expand if

$$n\theta \geq \sum_{i=1}^{n}(\max(w_{ji}, x_i) - \min(v_{ji}, x_i)),$$

where $0 \leq \theta \leq 1$ is the maximum size of a hyperbox; θ being a user defined value. The expansion *may* be allowed so that B_j represents the same class C_k. If the expansion criteria has been met for hyperbox B_j, one can adjust the min point of B_j using

$$v_{ji}^{new} = \min(v_{ji}^{old}, x_i), \ \forall i = 1, 2, \ldots, n, \tag{G.3}$$

and for the max point the updation rule is

$$w_{ji}^{new} = \max(w_{ji}^{old}, x_i), \ \forall i = 1, 2, \ldots, n. \tag{G.4}$$

Hyperbox overlap test

For each ith feature, if at least one of the following four cases is satisfied, overlap exists between two hyperboxes B_j and $B_{j'}$. While testing for the overlap, one can assume $\delta^{old} = 1$, initially.
Case 1: $v_{ji} < v_{j'i} < w_{ji} < w_{j'i}$,

$$\delta^{new} = \min(w_{ji} - v_{j'i}, \delta^{old}).$$

Case 2: $v_{j'i} < v_{ji} < w_{j'i} < w_{ji}$,

$$\delta^{new} = \min(w_{j'i} - v_{ji}, \delta^{old}).$$

Case 3: $v_{ji} < v_{j'i} < w_{ji} < w_{j'i}$

$$\delta^{new} = \min(\min(w_{j'i} - v_{ji}, w_{ji} - v_{j'i}), \delta^{old}).$$

Case 4: $v_{j'i} < v_{ji} < w_{j'i} < w_{ji}$

$$\delta^{new} = \min(\min(w_{ji} - v_{j'i}, w_{j'i} - v_{ji}), \delta^{old}).$$

If $\delta^{old} - \delta^{new} > 0$, then $\Delta = i$ and $\delta^{old} = \delta^{new}$, signifying that there is overlap for Δth dimension and overlap testing is repeated with the next dimension. Otherwise, the testing stops and Δ is set to -1.

Hyperbox contraction

If $\Delta > 0$, then Δth dimensions of two hyperboxes B_j and $B_{j'}$ are adjusted. Only one of the n dimensions is adjusted in each of the hyperboxes to keep the hyperboxes as large as possible. To determine the proper adjustment to make, one can examine the same four cases.

Case 1: $v_{j\Delta} < v_{j'\Delta} < w_{j\Delta} < w_{j'\Delta}$,

$$w_{j\Delta}^{new} = v_{j'\Delta}^{new} = \frac{w_{j\Delta}^{old} + v_{j'\Delta}^{old}}{2}.$$

Case 2: $v_{j'\Delta} < v_{j\Delta} < w_{j'\Delta} < w_{j\Delta}$,

$$w_{j'\Delta}^{new} = v_{j\Delta}^{new} = \frac{w_{j'\Delta}^{old} + v_{j'\Delta}^{old}}{2}.$$

Case 3a: $v_{j\Delta} < v_{j'\Delta} < w_{j'\Delta} < w_{j\Delta}$, and $(w_{j'\Delta} - v_{j\Delta}) \leq (w_{j\Delta} - v_{j'\Delta})$,

$$v_{j\Delta}^{new} = w_{j'\Delta}^{old}.$$

Case 3b: $v_{j\Delta} < v_{j'\Delta} < w_{j'\Delta} < w_{j\Delta}$, and $(w_{j'\Delta} - v_{j\Delta}) > (w_{j\Delta} - v_{j'\Delta})$,

$$w_{j\Delta}^{new} = v_{j'\Delta}^{old}.$$

Case 4a: $v_{j'\Delta} < v_{j\Delta} < w_{j\Delta} < w_{j'\Delta}$, and $(w_{j'\Delta} - v_{j\Delta}) < (w_{j\Delta} - v_{j'\Delta})$,

$$w_{j'\Delta}^{new} = v_{j\Delta}^{old}.$$

Case 4b: $v_{j'\Delta} < v_{j\Delta} < w_{j\Delta} < w_{j'\Delta}$, and $(w_{j'\Delta} - v_{j\Delta}) > (w_{j\Delta} - v_{j'\Delta})$,

$$v_{j'\Delta}^{new} = w_{j\Delta}^{old}.$$

Appendix H

Derivation of Eqn. (3.39)

For a pattern $\mathbf{x} \in C_k$,

$$\begin{aligned}
\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]} &= \frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k+\mu_{k'}-2\mu_k\mu_{k'}]} \\
&= \frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\mu_k\sum_{k'\neq k}[1-(2-\frac{1}{\mu_k})\mu_{k'}]} \\
&= \frac{(1-\mu_k)\alpha_k}{\frac{1}{2}[(M-1)-(2-\frac{1}{\mu_k})\sum_{k'\neq k}\mu_{k'}]} \\
&= \frac{(1-\mu_k)\alpha_k}{\frac{1}{2}(M-1)[1-(2-\frac{1}{\mu_k})\frac{\sum_{k'\neq k}\mu_{k'}}{M-1}]} \\
&\approx \frac{2(1-\mu_k)\alpha_k}{(M-1)}\left[1+(2-\frac{1}{\mu_k})\frac{\sum_{k'\neq k}\mu_{k'}}{M-1}\right],
\end{aligned}$$

as $(2-\frac{1}{\mu_k})(\frac{\sum_{k'\neq k}\mu_{k'}}{M-1}) < 1$. Thus,

$$\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]} \approx \frac{2\alpha_k}{M-1}\left(1-\mu_k+(3-2\mu_k-\frac{1}{\mu_k})\frac{\sum_{k'\neq k}\mu_{k'}}{M-1}\right)$$

Therefore, using Eqn. (3.38), $\mathcal{E}\left(\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]}\right)$ is given by,

$$\begin{aligned}
&\mathcal{E}\left(\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]}\right) \\
&= \int_{\mathbf{x}\in C_k}\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]}\wp(\mathbf{x})d\mathbf{x} \\
&\approx \int_{x_1=-\infty}^{\infty}\cdots\int_{x_n=-\infty}^{\infty}\frac{2\alpha_k}{M-1}\left(1-\mu_k+(3-2\mu_k-\frac{1}{\mu_k})\frac{\sum_{k'\neq k}\mu_{k'}}{M-1}\right)P_k\wp(\mathbf{x}|C_k)dx_1\ldots dx_n.
\end{aligned}$$

Let,

$$\begin{aligned} J_k &= \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} (1-\mu_k) P_k \tfrac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n \\ &= P_k - P_k \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} \mu_k \tfrac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n, \\ &= P_k - P_k J_{k1}, \end{aligned}$$

where

$$J_{k1} = \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} \mu_k \frac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n.$$

Also let,

$$\begin{aligned} J_{k2} &= \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} (3 - 2\mu_k - \tfrac{1}{\mu_k}) \sum_{k' \neq k} \mu_{k'} P_k \frac{1}{\sqrt{2\pi}\sigma} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n \\ &= P_k \sum_{k' \neq k} (3 J_{kk'1} - 2 J_{kk'2} - J_{kk'3}), \end{aligned}$$

where

$$\begin{aligned} J_{kk'1} &= \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} \tfrac{\mu_{k'}}{M-1} \tfrac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n, \\ J_{kk'2} &= \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} \mu_k \tfrac{\mu_{k'}}{M-1} \tfrac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n, \\ J_{kk'3} &= \int_{x_1=-\infty}^{\infty} \cdots \int_{x_n=-\infty}^{\infty} \tfrac{1}{\mu_k} \tfrac{\mu_{k'}}{M-1} \tfrac{1}{(\sqrt{2\pi}\sigma)^n} e \left(-\sum_i \frac{(x_i - m_{ki})^2}{2\sigma^2} \right) dx_1 \ldots dx_n. \end{aligned}$$

Therefore,

$$\mathcal{E} \left(\frac{\mu_k (1-\mu_k) \alpha_k}{\frac{1}{2} \sum_{k' \neq k} [\mu_k \times (1-\mu_{k'}) + \mu_{k'} \times (1-\mu_k)]} \right) = \frac{\alpha_k}{M-1} (J_k + J_{k2}). \qquad \text{(H.1)}$$

Let us also assume that

$$\begin{aligned}
J_{k1i} &= \int_{x_i=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(x_i-m_{ki})^2}{2\sigma^2}-\frac{(x_i-m_{ki})^2 w_i^2}{2\lambda^2}\right]} dx_i, \\
J_{kk'1i} &= \int_{x_i=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(x_i-m_{ki})^2}{2\sigma^2}-\frac{(x_i-m_{k'i})^2 w_i^2}{\lambda^2}\right]} dx_i, \\
J_{kk'2i} &= \int_{x_i=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(x_i-m_{ki})^2}{2\sigma^2}-\frac{(x_i-m_{k'i})^2 w_i^2}{2\lambda^2}-\frac{(x_i-m_{ki})^2 w_i^2}{2\lambda^2}\right]} dx_i, \\
J_{kk'3i} &= \int_{x_i=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(x_i-m_{ki})^2}{2\sigma^2}-\frac{(x_i-m_{k'i})^2 w_i^2}{\lambda^2}+\frac{(x_i-m_{ki})^2 w_i^2}{\lambda^2}\right]} dx_i,
\end{aligned}$$

so that

$$J_k = P_k(1-J_{k1}) = P_k\left(1-\prod_i J_{k1i}\right)$$

and

$$J_{k2} = P_k \sum_{k'\neq k}\left(3\prod_i J_{kk'1i} - 2\prod_i J_{kk'2i} - \prod_i J_{kk'3i}\right).$$

Therefore, from Eqn. (H.1) we have

$$\mathcal{E}\left(\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]}\right) = \left(\frac{\alpha_k}{M-1}\right)(J_k+\frac{J_{k2}}{M-1}). \qquad (II.2)$$

For evaluating the integrals $J_{k1i}, J_{kk'1i}$, $J_{kk'2i}$ and $J_{kk'3i}$ we use the result of the following integral

$$J = \int_{-\infty}^{\infty} e^{-(\alpha x^2+\beta x+\gamma)} dx.$$

Now,

$$\begin{aligned}
J &= \int_{-\infty}^{\infty} e^{-\alpha(x^2+2x\frac{\beta}{2\alpha}+\frac{\beta^2}{4\alpha^2})+(\frac{\beta^2}{4\alpha}-\gamma)} dx \\
&= e^{(\frac{\beta^2}{4\alpha}-\gamma)} \int_{-\infty}^{\infty} e^{-\alpha(x+\frac{\beta}{2\alpha})^2} dx \\
&= e^{(\frac{\beta^2}{4\alpha}-\gamma)} \int_{-\infty}^{\infty} e^{-\alpha y^2} dy
\end{aligned}$$

where

$$y = x+\frac{\beta}{2\alpha}.$$

Therefore,

$$\begin{aligned}
J &= 2e^{(\frac{\beta^2-4\alpha\gamma}{4\alpha})} \int_0^{\infty} e^{-\alpha y^2} dy \\
&= 2e^{(\frac{\beta^2-4\alpha\gamma}{4\alpha})} \int_0^{\infty} \frac{1}{2\sqrt{\alpha}} e^{-z} z^{-\frac{1}{2}} dz
\end{aligned}$$

where

$$z = \alpha y^2.$$

Hence,

$$J = \frac{e^{(\frac{\beta^2-4\alpha\gamma}{4\alpha})}\sqrt{\pi}}{\sqrt{\alpha}}. \tag{H.3}$$

We use the following transformation for evaluating J_{kli}, $J_{kk'1i}$, $J_{kk'2i}$ and $J_{kk'3i}$.

$$\begin{aligned} y_i &= \left(\frac{x_i - m_{kl}}{\sqrt{2}\lambda}\right) w_i, \\ dx_i &= \frac{\sqrt{2}\lambda}{w_i} dy_i. \end{aligned}$$

Then we can write

$$\begin{aligned}
\left[\frac{(x_i-m_{kl})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{ki})^2}{2\sigma^2}\right] &= y_i^2 + \frac{\rho^2}{w_i^2} y_i^2 \\
&= \left(1 + \frac{\rho^2}{w_i^2}\right) y_i^2, \\
\left[\frac{(x_i-m_{k'i})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{kl})^2}{2\sigma^2}\right] &= y_i^2 + \frac{\sqrt{2} w_i c_{kk'i}}{\lambda} y_i + \frac{c_{kk'i}^2 w_i^2}{2\lambda^2} \frac{\rho^2}{w_i^2} y_i^2 \\
&= \left(1 + \frac{\rho^2}{w_i^2}\right) y_i^2 + \frac{\sqrt{2} w_i c_{kk'i}}{\lambda} y_i + \frac{c_{kk'i}^2 w_i^2}{2\lambda^2}, \\
\left[\frac{(x_i-m_{kl})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{k'i})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{kl})^2}{2\sigma^2}\right] &= \left(2 + \frac{\rho^2}{w_i^2}\right) y_i^2 + \frac{\sqrt{2} w_i c_{kk'i}}{\lambda} y_i + \frac{c_{kk'i}^2 w_i^2}{2\lambda^2}, \\
\left[-\frac{(x_i-m_{kl})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{k'i})^2 w_i^2}{2\lambda^2} + \frac{(x_i-m_{kl})^2}{2\sigma^2}\right] &= \frac{\rho^2}{w_i^2} y_i^2 + \frac{\sqrt{2} w_i c_{kk'i}}{\lambda} y_i + \frac{c_{kk'i}^2 w_i^2}{2\lambda^2}.
\end{aligned}$$

Therefore, using the result of J (Eqn. (H.3)) we have

$$\begin{aligned}
J_{kli} &= \frac{1}{\sqrt{2\pi}\sigma} \frac{\sqrt{2}\lambda}{w_i} \frac{1}{(1+\frac{\rho^2}{w_i^2})^{\frac{1}{2}}} \sqrt{\pi} \\
&= \frac{\rho}{(w_i^2+\rho^2)^{\frac{1}{2}}},
\end{aligned}$$

where $\alpha = (1 + \frac{\rho^2}{w_i^2})$, $\beta = 0$ and $\gamma = 0$. Similarly, the expressions for $J_{kk'1i}$, $J_{kk'2i}$ and $J_{kk'3i}$ are obtained as follows.

$$\begin{aligned}
J_{kk'1i} &= \frac{1}{\sqrt{2\pi}\sigma} \frac{\sqrt{2}\lambda}{w_i} \frac{1}{(1+\frac{\rho^2}{w_i^2})^{\frac{1}{2}}} e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}} \sqrt{\pi} \\
&= \frac{\rho e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}}}{(\rho^2+w_i^2)^{\frac{1}{2}}}
\end{aligned}$$

where $\alpha = (1 + \frac{\rho^2}{w_i^2})$, $\beta = \frac{\sqrt{2} w_i c_{kk'i}}{\lambda}$ and $\gamma = \frac{c_{kk'i}^2 w_i^2}{2\lambda^2}$.

$$J_{kk'2i} = \frac{\rho e^{-\frac{c_{kk'i}^2(1+\frac{\rho^2}{w_i^2})w_i^2}{2\sigma^2(2+\frac{\rho^2}{w_i^2})}}}{(\rho^2+w_i^2)^{\frac{1}{2}}},$$

where $\alpha = (2 + \frac{\rho^2}{w_i^2})$, $\beta = \frac{\sqrt{2}w_i c_{kk'i}}{\lambda}$ and $\gamma = \frac{c_{kk'}^2 w_i^2}{2\lambda^2}$.

$$J_{kk'3i} = e^{-\frac{c_{kk'i}^2(1-\frac{w_i^2}{\rho^2})w_i^2}{2\sigma^2}}$$

where $\alpha = \frac{\rho^2}{w_i^2}$, $\beta = \frac{\sqrt{2}w_i c_{kk'i}}{\lambda}$ and $\gamma = \frac{c_{kk'}^2 w_i^2}{2\lambda^2}$.

Therefore, from Eqn. (H.2) we have

$$\begin{aligned}
&\mathcal{E}\left(\frac{\mu_k(1-\mu_k)\alpha_k}{\frac{1}{2}\sum_{k'\neq k}[\mu_k\times(1-\mu_{k'})+\mu_{k'}\times(1-\mu_k)]}\right) \\
&= \frac{\alpha_k P_k}{M-1}\left([1-\prod_i\frac{\rho}{(w_i^2+\rho^2)^{\frac{1}{2}}}]+\sum_{k'\neq k}[3\prod_i\frac{\rho e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}}}{(\rho^2+w_i^2)^{\frac{1}{2}}}-2\prod_i\frac{\rho e^{-\frac{c_{kk'i}^2(1+\frac{\rho^2}{w_i^2})w_i^2}{2\sigma^2(2+\frac{\rho^2}{w_i^2})}}}{(\rho^2+w_i^2)^{\frac{1}{2}}}-\prod_i e^{-\frac{c_{kk'i}^2(1-\frac{w_i^2}{\rho^2})w_i^2}{2\sigma^2}}]\right) \\
&\approx \frac{\alpha_k P_k}{M-1}\left([1-\prod_i\frac{\rho}{(w_i^2+\rho^2)^{\frac{1}{2}}}]+\sum_{k'\neq k}[3\prod_i\frac{\rho e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}}}{(\rho^2+w_i^2)^{\frac{1}{2}}}-2\prod_i\frac{\rho e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}}}{(\rho^2+2w_i^2)^{\frac{1}{2}}}-\prod_i e^{-\frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}}]\right)
\end{aligned} \tag{H.4}$$

Now,

$$\begin{aligned}
\prod_i\frac{\rho}{(w_i^2+\rho^2)^{\frac{1}{2}}} &= \prod_i\frac{\rho}{\rho(1+\frac{w_i^2}{\rho^2})^{\frac{1}{2}}} \\
&= \prod_i(1+\frac{w_i^2}{\rho^2})^{-\frac{1}{2}} \\
&= \prod_i(1-\frac{w_i^2}{2\rho^2}) \\
&= (1-\frac{\sum_i w_i^2}{2\rho^2}),
\end{aligned}$$

as $\frac{w_i}{\rho} \ll 1$. Similarly,

$$\begin{aligned}
\prod_i\frac{\rho}{(2w_i^2+\rho^2)^{\frac{1}{2}}} &= \prod_i\frac{\rho}{\rho(1+\frac{2w_i^2}{\rho^2})^{\frac{1}{2}}} \\
&= \prod_i(1+\frac{2w_i^2}{\rho^2})^{-\frac{1}{2}} \\
&= \prod_i(1-\frac{w_i^2}{\rho^2}) \\
&= (1-\frac{\sum_i w_i^2}{\rho^2}).
\end{aligned}$$

Therefore,

$$\mathcal{E}(E) \approx \sum_k \frac{\alpha_k P_k}{M-1} \frac{\sum_i w_i^2}{2\rho^2} \left(1 + \sum_{k' \neq k} e^{-\sum_i \frac{c_{kk'i}^2}{2\sigma^2(1+\frac{\rho^2}{w_i^2})}} \right). \tag{H.5}$$

Bibliography

[1] E. H. L. Aarts and J. H. M. Korst, *Boltzmann machines and their applications*, vol. 258 of *Lecture Notes on Computer Science*, pp. 34–50. Springer Verlag, 1987.

[2] M. A. Abou-Nasr and M. A. Sid-Ahmed, "Fast learning and efficient memory utilization with a prototype based neural classifier," *Pattern Recognition*, vol. 28, pp. 581–593, 1995.

[3] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machine," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[4] C. A. Ankenbrandt, B. P. Buckles, and F. E. Petry, "Scene recognition using genetic algorithms with semantic nets," *Pattern Recognition Letters*, vol. 11, pp. 285–293, 1990.

[5] D. H. Ballard and C. M. Brown, *Computer Vision.* Englewood Cliffs: Prentice-Hall, Inc., 1982.

[6] S. Bandyopadhyay, *Pattern Classification using Genetic Algorithms.* PhD thesis, Indian Statistical Institute, Calcutta, India, 1998.

[7] E. R. Bareiss, B. W. Porter, and C. C. Weir, "Protos: An exemplar-based learning apprentice," *International Journal of Man-Machine Studies*, vol. 29, pp. 549–561, 1988.

[8] E. Barnard and E. C. Botha, "Back-propagation uses prior information efficiently," *IEEE Trans. on Neural Networks*, vol. 4, pp. 794–802, 1993.

[9] E. Barnard and D. Casasent, "A comparison between criterion functions for linear classifiers, with an application to neural networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 19, pp. 1030–1041, 1989.

[10] J. Basak, R. K. De, and S. K. Pal, "Fuzzy feature evaluation and connectionist realization–II : Theoretical analysis," *Information Sciences*, vol. 111, pp. 1–17, 1998.

[11] J. Basak, R. K. De, and S. K. Pal, "Unsupervised feature selection using neuro-fuzzy approach," *Pattern Recognition Letters*, vol. 19, pp. 997–1006, 1998.

[12] J. Basak, R. K. De, and S. K. Pal, "Unsupervised neuro-fuzzy feature selection," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 18–23, 1998.

[13] J. Basak and S. K. Pal, "PsyCOP: A psychologically motivated connectionist system for object perception," *IEEE Trans. on Neural Networks*, vol. 6, pp. 1337–1354, 1995.

[14] J. Basak and S. K. Pal, "X-tron: An incremental connectionist model for category perception," *IEEE Trans. on Neural Networks*, vol. 6, pp. 1091–1108, 1995.

[15] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. on Neural Networks*, vol. 5, pp. 537–550, 1994.

[16] H.-U. Bauer and K. R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps," *IEEE Trans. on Neural Networks*, vol. 3, pp. 570–579, 1992.

[17] R. Bellman, R. Kalaba, and L. A. Zadeh, "Abstraction and pattern classification," *Journal of Mathematical Analysis Applications*, vol. 13, pp. 1–7, 1966.

[18] M. G. Bello, "Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks," *IEEE Trans. on Neural Networks*, vol. 3, pp. 864–875, 1992.

[19] L. M. Belue and K. W. Bauer, Jr., "Determining input features for multilayer perceptrons," *Neurocomputing*, vol. 7, pp. 111–121, 1995.

[20] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[21] 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.

[22] 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.

[23] 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.

[24] M. Blumenstein and B. Verma, "A segmentation algorithm used in conjunction with artificial neural networks for the recognition of real-world postal addresses," in *Proc. International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '97)*, (Gold Coast, Australia), pp. 155–160, 1997.

[25] 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.

[26] J. Braun and H. Levkowitz, "Automatic language identification with recurrent neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 2184–2189, May 1998.

[27] M. Brejl and M. Sonka, "Edge-based image segmentation: Machine learning from examples," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 814–819, May 1998.

[28] G. A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Networks*, vol. 2, pp. 243–257, 1989.

[29] 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.

[30] 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 Trans. on Neural Networks*, vol. 3, pp. 698–713, 1992.

[31] 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.

[32] V. Chandrasekaran and Z.-Q. Liu, "Topology constraint free fuzzy gated neural networks for pattern recognition," *IEEE Trans. on Neural Networks*, vol. 9, pp. 483–502, 1998.

[33] C. Chatterjee and V. P. Roychowdhury, "On self-organizing algorithms and networks for class-separability features," *IEEE Trans. on Neural Networks*, vol. 8, pp. 663–678, 1997.

[34] 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.

[35] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. on Information Theory*, vol. IT-13, pp. 21–27, 1967.

[36] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[37] J. E. Dayhoff, *Neural Network Architectures An Introduction*. New York: Van Nostrand Reinhold, 1990.

[38] R. K. De, J. Basak, and S. K. Pal, "Neuro-fuzzy feature evaluation with theoretical analysis," *Neural Networks*, (revised).

[39] R. K. De, J. Basak, and S. K. Pal, "Unsupervised feature extraction using neuro-fuzzy approach," *Fuzzy Sets and Systems*, (communicated).

[40] R. K. De, J. Basak, and S. K. Pal, "Unsupervised neuro-fuzzy feature extraction," in *Proc. 5th International Conference on Soft Computing IIZUKA'98*, (Iizuka, Fukuoka, Japan), pp. 577–580, 1998.

[41] R. K. De, S. Mitra, and S. K. Pal, "Knowledge-based fuzzy MLP for pattern classification," in *Proc. Sixth International Fuzzy Systems Association World Congress*, vol. II, (Sao Paulo, Brazil), pp. 237–240, 1995.

[42] R. K. De, S. Mitra, and S. K. Pal, "Neuro-fuzzy knowledge-based system for rule generation," in *Proc. Indian Conference on Pattern Recognition, Image Processing and Computer Vision (ICPIC'95)*, (I. I. T., Kharagpur, India), pp. 130–134, 1995.

[43] R. K. De, N. R. Pal, and S. K. Pal, "Feature analysis : Neural network and fuzzy set theoretic approaches," *Pattern Recognition*, vol. 30, pp. 1579–1590, 1997.

[44] R. K. De and S. K. Pal, "Fuzzy case-based classification in a connectionist framework," *Pattern Recognition Letters*, (communicated).

[45] R. K. De and S. K. Pal, "Case-based classification using fuzziness and neural networks," in *Proc. Colloquium on Knowledge Discovery and Data Mining*, (London, U. K.), 1998.

[46] R. De Mori and P. Laface, "Use of fuzzy algorithms for phonetic and phonemic labeling of continuous speech," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 136–148, 1980.

[47] P. Demartines and J. Herault, "Curvilinear component analysis : A self-organizing neural network for nonlinear mapping of data sets," *IEEE Trans. on Neural Network*, vol. 8, pp. 148–160, 1997.

[48] T. Denoeux, "A k-nearest neighbor classification rule based on Dempster-Shafer theory," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 804–813, 1995.

[49] P. A. Devijver and J. Kittler, *Pattern Recognition, A Statistical Approach*. Englewood Cliffs: Prentice–Hall, Inc., 1982.

[50] L. Ding and Z. Shen, "Neural network implementation of fuzzy inference for approximate case-based reasoning," in *Neural and Fuzzy Systems* (S. Mitra, M. M. Gupta, and W. F. Kraske, eds.), pp. 28–56, Washington: Spie Optical Engineering Press, 1994.

[51] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications.* Boston: Academic Press, Inc., 1980.

[52] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley, 1973.

[53] 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: Morgan-Kaufmann, 1990.

[54] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[55] H. Fu and Y. Y. Xu, "Recognition of handwritten similar Chinese characters by self-growing probabilistic decision-based neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 1754–1759, May 1998.

[56] K. S. Fu, *Syntactic Pattern Recognition and Applications.* Englewood Cliffs: Prentice–Hall, Inc., 1982.

[57] L. M. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, pp. 173–182, 1993.

[58] M. Fukumi, S. Omatu, F. Takeda, and T. Kosaka, "Rotation-invariant neural pattern recognition system with application to coin recognition," *IEEE Trans. on Neural Networks*, vol. 3, pp. 272–279, 1992.

[59] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic, 1972.

[60] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, 1988.

[61] K. Fukushima, "A neural network for visual pattern recognition," *IEEE Computer*, pp. 65–75, March 1988.

[62] *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, (Anchorage, USA), May 1998.

[63] S. I. Gallant, "Connectionist expert systems," *Communications of the Association for Computing Machinery*, vol. 31, pp. 152–169, 1988.

[64] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Trans. on Neural Networks*, vol. 1, pp. 179–191, 1990.

[65] A. Ghosh, N. R. Pal, and S. K. Pal, "Image segmentation using a neural network," *Biological Cybernetics*, vol. 66, pp. 151–158, 1991.

[66] A. Ghosh, N. R. Pal, and S. K. Pal, "Self-organization for object extraction using multilayer neural network and fuzziness measures," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 54–68, 1993.

[67] A. Ghosh and S. K. Pal, "Neural network, self-organization and object extraction," *Pattern Recognition Letters*, vol. 13, pp. 387–397, 1992.

[68] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading: Addison-Wesley, 1989.

[69] R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition : An Introduction.* Reading: Addison-Wesley, 1978.

[70] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Reading: Addison-Wesley, 1992.

[71] M. A. Grau and L. H. Molinero, "A fast method for rule extraction in neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 2334–2339, May 1998.

[72] S. Grossberg, ed., *Neural Networks and Natural Intelligence.* Cambridge: MIT Press, 1988.

[73] M. M. Gupta, A. Kandel, W. Bandler, and J. B. Kiszka, eds., *Approximate Reasoning in Expert Systems.* Amsterdam: North Holland, 1985.

[74] M. M. Gupta and D. H. Rao, "On the principles of fuzzy neural networks," *Fuzzy Sets and Systems*, vol. 61, pp. 1–18, 1994.

[75] M. M. Gupta and E. Sanchez, eds., *Fuzzy Information and Decision Processes*. Amsterdam: North Holland, 1982.

[76] M. Hagiwara, "A simple and effective method for removal of hidden units and weights," *Neurocomputing*, vol. 6, pp. 207–218, 1994.

[77] 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.

[78] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

[79] R. Hecht-Nielsen, "Counterpropagation networks," *Applied Optics*, vol. 26, pp. 4979–4984, 1987.

[80] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, pp. 131–139, 1988.

[81] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading: Addison-Wesley, 1994.

[82] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185–234, 1989.

[83] K. Hirota and W. Pedrycz, "Knowledge-based networks in classification problems," *Fuzzy Sets and Systems*, vol. 59, pp. 271–279, 1993.

[84] K. Hirota and W. Pedrycz, "Or/And neuron in modeling fuzzy set connectives," *IEEE Trans. on Fuzzy Systems*, vol. 2, pp. 151–161, 1994.

[85] 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.

[86] 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.

[87] K. Hornik and C.-M. Kuan, "Convergence analysis of local feature extraction algorithms," *Neural Networks*, vol. 5, pp. 229–240, 1992.

[88] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[89] A. Howard, C. Padgett, and C. C. Liebe, "A multistage neural network for automatic target detetion," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 231–236, May 1998.

[90] 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.

[91] D. R. Hush and B. G. Horne, "Progress in supervised neural networks: What's new since lippmann?," *IEEE Signal Processing Magazine*, pp. 8–39, January, 1993.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] *Proc. of Fifth International Conference on Soft Computing (IIZUKA'98)*, (Iizuka, Fukuoka, Japan), October 1998.

[97] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *IEEE Computer*, pp. 31–44, March, 1996.

[98] R. Kamimura, "Neural a-feature detector for feature detection and generalization," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 1845–1850, May 1998.

[99] A. Kandel, *Fuzzy Techniques in Pattern Recognition.* New York: Wiley Interscience, 1982.

[100] A. Kandel, *Fuzzy Mathematical Techniques with Applications.* New York: Addison-Wesley, Inc., 1986.

[101] N. B. Karayiannis, "ALADIN : Algorithms for learning and architecture determination," *IEEE Trans. on Circuits and Systems*, vol. 41, pp. 752–759, 1994.

[102] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. on Neural Networks*, vol. 1, pp. 239–242, 1990.

[103] A. Kaufmann and M. Gupta, *Introduction to Fuzzy Mathematics.* New York: Van Nostrand Reinhold, 1985.

[104] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 15, pp. 580–585, 1985.

[105] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 693–699, 1985.

[106] J. M. Keller, R. Krishnapuram, and F. C. -H. Rhee, "Evidence aggregation networks for fuzzy logic inference," *IEEE Trans. on Neural Networks*, vol. 3, pp. 761–769, 1992.

[107] 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.

[108] 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.

[109] G. J. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information.* Reading: Addison-Wesley, 1989.

[110] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Englewood Cliffs: Prentice Hall, Inc., 1995.

[111] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, pp. 3–16, 1988.

[112] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1989.

[113] J. L. Kolodner, *Case-Based Reasoning.* San Mateo: Morgan Kaufmann, 1993.

[114] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," *International Journal of Approximate Reasoning*, vol. 2, pp. 377–393, 1988.

[115] B. Kosko, *Neural Networks and Fuzzy Systems.* New Jersey: Prentice Hall, 1991.

[116] A. Kowalczyk and H. L. Ferra, "Developing higher-order neural networks with empirically selected units," *IEEE Trans. on Neural Networks*, vol. 5, pp. 698–711, 1994.

[117] M. A. Kraaijveld, J. Mao, and A. K. Jain, "A non-linear projection method based on Kohonen's topology preserving maps," *IEEE Trans. on Neural Networks*, vol. 6, pp. 548–559, 1995.

[118] R. Krishnapuram and J. Lee, "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks*, vol. 5, pp. 335–350, 1992.

[119] J. Lampinen and E. Oja, "Distortion tolerant pattern recognition based on self-organizing feature extraction," *IEEE Trans. on Neural Networks*, vol. 6, pp. 539–547, 1995.

[120] 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.

[121] C. Lee and D. A. Landgrebe, "Feature extraction based on decision boundaries," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 388–400, 1993.

[122] C. Lee and D. A. Landgrebe, "Decision boundary feature extraction for neural networks," *IEEE Trans. on Neural Network*, vol. 8, pp. 75–83, 1997.

[123] S. C. Lee and E. T. Lee, "Fuzzy neural networks," *Mathematical Biosciences*, vol. 23, pp. 151–177, 1975.

[124] J. Liaw and T. W. Berger, "Robust speech recognition with dynamic synapses," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 2175–2179, May 1998.

[125] D. Lin, "Spatio-temporal hand gesture recognition using neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 1794–1798, May 1998.

[126] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Communications Magazine*, pp. 47–64, 1989.

[127] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[128] Z.-Q. Liu and F. Yan, "Fuzzy neural network in case-based diagnostic system," *IEEE Trans. on Fuzzy Systems*, vol. 5, pp. 209–222, 1997.

[129] D. Lowe and A. R. Webb, "Optimized feature extraction and Bayes decision in feed-forward classifier networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 355–364, 1991.

[130] R. J. Machado and A. F. Rocha, "A hybrid architecture for connectionist expert systems," in *Intelligent Hybrid Systems* (A. Kandel and G. Langholz, eds.), Boca Raton: CRC Press, 1992.

[131] 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.

[132] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Formulation of a multi-valued recognition system," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 22, pp. 607–620, 1992.

[133] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Theoretical performance of a multivalued recognition system," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, pp. 1001–1021, 1994.

[134] 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.

[135] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Trans. on Neural Networks*, vol. 6, pp. 296–317, 1995.

[136] R. Masuoka, N. Watanabe, A. Kawamura, Y. Owada, and K. Asakawa, "Neurofuzzy system – fuzzy inference using a structured neural network," in *Proceedings of the 1990 International Conference on Fuzzy Logic and Neural Networks, Iizuka*, (Japan), pp. 173–177, 1990.

[137] C. Mead and M. Ismail, eds., *Analog VLSI Implementation of Neural Systems*. Boston: Kluwer Academic, 1989.

[138] D. A. Medler, "A brief history of connectionism," *Neural Computing Surveys*, vol. 1, pp. 61–101, 1998.

[139] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge: MIT Press, 1969.

[140] M. Misra, "Parallel environments for implementing neural networks," *Neural Computing Surveys*, vol. 1, pp. 48–60, 1997.

[141] S. Mitra, "Fuzzy MLP based expert system for medical diagnosis," *Fuzzy Sets and Systems*, vol. 65, pp. 285–296, 1994.

[142] S. Mitra, R. K. De, and S. K. Pal, "Knowledge-based fuzzy MLP for classification and rule generation," *IEEE Trans. on Neural Networks*, vol. 8, pp. 1338–1350, 1997.

[143] S. Mitra and S. K. Pal, "Logical operation based fuzzy MLP for classification and rule generation," *Neural Networks*, vol. 7, pp. 353–373, 1994.

[144] 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: Kluwer, 1994.

[145] S. Mitra and S. K. Pal, "Self-organizing neural network as a fuzzy classifier," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, pp. 385–399, 1994.

[146] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. on Neural Networks*, vol. 6, pp. 51–63, 1995.

[147] A. Nafarieh and J. Keller, "A fuzzy logic rule-based automatic target recognition," *Int. J. Intell. Syst.*, vol. 6, pp. 295–312, 1991.

[148] H. Narazaki and T. Watanabe, "A case-based approach for modeling nonlinear systems," *Fuzzy Sets and Systems*, vol. 77, pp. 77–86, 1996.

[149] S. C. Newton, S. Pemmaraju, and S. Mitra, "Adaptive fuzzy leader clustering of complex data sets in pattern recognition," *IEEE Trans. on Neural Networks*, vol. 3, pp. 794–800, 1992.

[150] 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.

[151] N. R. Pal and J. C. Bezdek, "Measuring fuzzy uncertainty," *IEEE Trans. on Fuzzy Systems*, vol. 2, pp. 107–118, 1994.

[152] S. K. Pal, "Fuzzy set theoretic measures for automatic feature evaluation: II," *Information Sciences*, vol. 64, pp. 165–179, 1992.

[153] S. K. Pal, J. Basak, and R. K. De, "Feature selection : A neuro-fuzzy approach," in *Proc. International Conference on Neural Networks (ICNN'96)*, (Washington DC, USA), pp. 1197–1202, 1996.

[154] S. K. Pal, J. Basak, and R. K. De, "Fuzzy feature evaluation index and connectionist realization," *Information Sciences*, vol. 105, pp. 173–188, 1998.

[155] S. K. Pal and D. Bhandari, "Genetic algorithms with fuzzy fitness function for object extraction using cellular neural networks," *Fuzzy Sets and Systems*, vol. 65, pp. 129–139, 1994.

[156] S. K. Pal and B. Chakraborty, "Fuzzy set theoretic measures for automatic feature evaluation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 16, pp. 754–760, 1986.

[157] S. K. Pal and D. Dutta Majumder, "Fuzzy sets and decision making approaches in vowel and speaker recognition," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 7, pp. 625–629, 1977.

[158] S. K. Pal and A. Ghosh, "Neuro-fuzzy computing for image processing and pattern recognition," *International Journal of System Sciences*, vol. 27, pp. 1179–1193, 1996.

[159] S. K. Pal and D. P. Mandal, "Linguistic recognition system based on approximate reasoning," *Information Sciences*, vol. 61, pp. 135–161, 1992.

[160] S. K. Pal and S. Mitra, "Fuzzy dynamic clustering algorithm," *Pattern Recognition Letters*, vol. 11, pp. 525–535, 1990.

[161] S. K. Pal and S. Mitra, *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing Paradigm*. New York: John Wiley, (to appear).

[162] S. K. Pal and P. K. Pramanik, "Fuzzy measures in determining seed points in clustering," *Pattern Recognition Letter*, vol. 4, pp. 159–164, 1986.

[163] S. K. Pal and P. P. Wang, eds., *Genetic Algorithms for Pattern Recognition*. Boca Raton: CRC Press, 1996.

[164] S. K. Pal, S. Bandyopadhyay, and C. A. Murthy, "Genetic algorithms for generation of class boundaries," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 28, pp. 816–828, 1998.

[165] S. K. Pal, R. K. De, and J. Basak, "Unsupervised feature evaluation : A neuro-fuzzy approach," *IEEE Trans. on Neural Networks*, (communicated).

[166] S. K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*. New York: John Wiley (Halsted Press), 1986.

[167] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. on Neural Network*, vol. 3, pp. 683–697, 1992.

[168] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. New York: Addison-Wesley, 1989.

[169] R. Parekh and V. Honavar, "Constructive theory refinement in knowledge based neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 2318–2323, May 1998.

[170] A. Pathak and S. K. Pal, "Fuzzy grammars in syntactic recognition of skeletal from x-rays," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 16, pp. 657–667, 1984.

[171] A. Pathak, S. K. Pal, and R. A. King, "Syntactic recognition of skeletal maturity," *Pattern Recognition Letters*, vol. 2, pp. 193–197, 1984.

[172] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

[173] W. Pedrycz, "Fuzzy sets in pattern recognition : Methodology and methods," *Pattern Recognition*, vol. 23, pp. 121–146, 1990.

[174] W. Pedrycz, "Neurocomputations in relational systems," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 289–297, 1991.

[175] W. Pedrycz, "A referential scheme of fuzzy decision making and its neural network structure," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, pp. 1593–1604, 1991.

[176] W. Pedrycz, "Fuzzy neural network with reference neurons as pattern classifiers," *IEEE Trans. on Neural Networks*, vol. 3, pp. 770–775, 1992.

[177] W. Pedrycz, "Conditional fuzzy clustering in the design of radial basis function neural networks," *IEEE Trans. on Neural Networks*, vol. 9, pp. 601–612, 1998.

[178] W. Pedrycz and F. Gomide, eds., *An Introduction to Fuzzy Sets Analysis and Design*. Cambridge: MIT Press, 1998.

[179] W. Pedrycz and A. F. Rocha, "Fuzzy-set based models of neurons and knowledge-based networks," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 254–266, 1993.

[180] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Trans. on Neural Networks*, vol. 3, pp. 241–251, 1992.

[181] B. W. Porter, E. R. Bareiss, and R. C. Holte, "Concept learning and heuristic classification in weak theory domains," *Artificial Intelligence*, vol. 45, pp. 229–263, 1990.

[182] K. L. Priddy, S. K. Rogers, D. W. Ruck, G. L. Tarr, and M. Kabrisky, "Bayesian selection of important features for feedforward neural networks," *Neurocomputing*, vol. 5, pp. 91–103, 1993.

[183] U. Ramacher and U. Ruckert, eds., *VLSI Design of Neural Networks*. Boston: Kluwer Academic Press, 1991.

[184] R. Reed, "Pruning algorithms - A survey," *IEEE Trans. on Neural Networks*, vol. 4, pp. 740–747, 1993.

[185] M. Richard and R. Lippman, "Neural network classifiers estimate Bayesian a posteriori probabilities," *Neural Computation*, vol. 3, pp. 461–483, 1991.

[186] S. K. Rogers, J. M. Colombi, C. E. Martin, J. C. Gainey, K. H. Fielding, T. J. Burns, D. W. Ruch, M. Kabrisky, and M. Oxley, "Neural networks for automatic target recognition," *Neural Networks*, vol. 8, pp. 1153–1184, 1995.

[187] S. G. Romaniuk and L. O. Hall, "Divide and conquer neural networks," *Neural Networks*, vol. 6, pp. 1105–1116, 1993.

[188] F. Rosenblatt, *Principles of Neurodynamics, Perceptrons and the Theory of Brain Mechanisms*. Washington: Spartan Books, 1961.

[189] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic Press, 1982.

[190] J. Rubner and P. Tavan, "A self-organizing network for principal component analysis," *Europhysics Letters*, vol. 10, pp. 693–698, 1989.

[191] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *Journal of Neural Network Computing*, pp. 40–48, Fall 1990.

[192] 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: MIT Press, 1986.

[193] D. E. Rumelhart, J. McClelland, and the PDP Research Group, eds., *Parallel Distributed Processing*, vol. 1:Foundations. Cambridge: MIT Press, 1986.

[194] E. H. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, pp. 22–32, 1969.

[195] S. Salzberg, "A nearest hyperrectangle learning method," *Machine Learning*, vol. 6, pp. 251–276, 1991.

[196] E. Sanchez and L. A. Zadeh, eds., *Approximate Reasoning in Intelligent Systems, Decision and Control*. Oxford: Pergamon Press, 1987.

[197] 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.

[198] M. Sarkar, B. Yegnanarayana, and D. Khemani, "Backpropagation learning algorithms for classification with fuzzy mean square error," *Pattern Recognition Letters*, vol. 19, pp. 43–51, 1998.

[199] E. Saund, "Dimensionality-reduction using connectionist networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 304–314, 1989.

[200] W. A. C. Schmidt and J. P. Davis, "Pattern recognition properties of various feature spaces for higher order neural networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 795–801, 1993.

[201] R. Setino and H. Liu, "Neural-network feature selector," *IEEE Trans. on Neural Networks*, vol. 8, pp. 654–662, 1997.

[202] G. Shafer, *A Mathematical Theory of Evidence*. Princeton: Princeton University Press, 1976.

[203] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recog. Lett.*, vol. 10, pp. 335–347, 1989.

[204] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, pp. 67–79, 1991.

[205] P. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations*. Elmsford, NY: Pergamon Press, 1990.

[206] P. K. Simpson, "Fuzzy min-max neural networks – Part 1: Classification," *IEEE Trans. on Neural Networks*, vol. 3, pp. 776–786, 1992.

[207] P. K. Simpson, "Min-max neural networks: 2.Clustering," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 32–45, 1993.

[208] S. G. Smyth, "Designing multilayer perceptrons from nearest-neighbor systems," *IEEE Trans. on Neural Networks*, vol. 3, pp. 329–333, 1992.

[209] J. M. Steppe and K. W. Bauer, Jr., "Improved feature screening in feedforward neural networks," *Neurocomputing*, vol. 13, pp. 47–58, 1996.

[210] H. Takagi and I. Hayashi, "Artificial neural network driven fuzzy reasoning," *International Journal of Approximate Reasoning*, vol. 5, pp. 191–212, 1991.

[211] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their applications," *IEEE Trans. on Neural Networks*, vol. 3, pp. 752–760, 1992.

[212] M. Tanaka, F. Takeda, K. Ohkouchi, and Y. Michiyuki, "Recognition of paper currencies by hybrid neural network," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 1748–1753, May 1998.

[213] T. Tollenaere, "Super SAB: fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, pp. 561–573, 1990.

[214] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading: Addison-Wesley, 1974.

[215] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, pp. 119–165, 1994.

[216] I. B. Turksen, "Four methods of approximate reasoning with interval-valued fuzzy sets," *International Journal of Approximate Reasoning*, vol. 3, pp. 121–142, 1989.

[217] F. Vacca and E. Chiarantoni, "Clustering noisy data by a principal feature extraction unsupervised neural networks," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 2361–2366, May 1998.

[218] 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.

[219] 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.

[220] L. F. A. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Trans. on Neural Networks*, vol. 3, pp. 899–905, 1992.

[221] B. Widrow and R. Winter, "Neural network for adaptive filtering and adaptive pattern recognition," *IEEE Computer*, pp. 25–39, March 1988.

[222] R. R. Yager and L. A. Zadeh, eds., *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Boston: Kluwer Academic Press, 1992.

[223] 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.

[224] T. Yamakawa, "Silicon implementation of a fuzzy neuron," *IEEE Trans. on Fuzzy Systems*, vol. 4, pp. 488–501, 1996.

[225] 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.

[226] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[227] L. A. Zadeh, "The concept of linguistic variable and its applications to approximate reasoning-ii," *Inform. Sci.*, vol. 8, pp. 301–357, 1975.

[228] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[229] 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.

[230] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Communications of the ACM*, vol. 37, pp. 77–84, 1994.

[231] B. Zhang, M. Fu, and H. Yan, "Application of neural 'gas' model in image compression," in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, USA), pp. 918–921, May 1998.

[232] H. -J. Zimmermann, *Fuzzy Set Theory and its Applications.* Boston: Kluwer, 1991.

[233] H. -J. Zimmermann and P. Zysno, "Decisions and evaluations by hierarchical aggregation of information," *Fuzzy Sets and Systems*, vol. 10, pp. 243–260, 1983.

LIST OF PUBLICATIONS OF THE AUTHOR

[1] R. K. De, N. R. Pal, and S. K. Pal, "Feature analysis : Neural network and fuzzy set theoretic approaches," *Pattern Recognition*, vol. 30, pp. 1579–1590, 1997.

[2] S. Mitra, R. K. De, and S. K. Pal, "Knowledge-based fuzzy MLP for classification and rule generation," *IEEE Trans. on Neural Networks*, vol. 8, pp. 1338–1350, 1997.

[3] S. K. Pal, J. Basak, and R. K. De, "Fuzzy feature evaluation index and connectionist realization," *Information Sciences*, vol. 105, pp. 173–188, 1998.

[4] J. Basak, R. K. De, and S. K. Pal, "Fuzzy feature evaluation index and connectionist realization–II : Theoretical analysis", *Information Sciences*, vol. 111, pp. 1–17, 1998.

[5] J. Basak, R. K. De, and S. K. Pal, "Unsupervised feature selection using neuro-fuzzy approach", *Pattern Recognition Letters*, vol. 19, pp. 997–1006, 1998.

[6] R. K. De, J. Basak, and S. K. Pal, "Neuro-fuzzy feature evaluation with theoretical analysis", *Neural Networks*, (revised).

[7] R. K. De, J. Basak, and S. K. Pal, "Unsupervised feature extraction using neuro-fuzzy approach", *Fuzzy Sets and Systems*, (communicated).

[8] S. K. Pal, R. K. De, and J. Basak, "Unsupervised feature evaluation : A neuro-fuzzy approach", *IEEE Trans. on Neural Networks*, (communicated).

[9] R. K. De and S. K. Pal, "Fuzzy case-based classification in a connectionist framework", *Pattern Recognition Letters*, (communicated).

[10] R. K. De, S. Mitra, and S. K. Pal, "Knowledge-based fuzzy MLP for pattern classification," in *Proc. Sixth International Fuzzy Systems Association World Congress*, (Sao Paulo, Brazil), vol. II, pp. 237–240, 1995.

[11] R. K. De, S. Mitra, and S. K. Pal, "Neuro-fuzzy knowledge-based system for rule generation," in *Proc. Indian Conference on Pattern Recognition, Image Processing and Computer Vision (ICPIC'95)*, (I. I. T., Kharagpur, India), pp. 130–134, 1995.

[12] S. K. Pal, J. Basak, and R. K. De, "Feature selection : A neuro-fuzzy approach,"

in *Proc. International Conference on Neural Networks (ICNN'96)*, (Washington DC, U. S. A.), pp. 1197–1202, 1996.

[13] J. Basak, R. K. Dè, and S. K. Pal, "Unsupervised neuro-fuzzy feature selection", in *Proc. 1998 IEEE International Joint Conference on Neural Networks IJCNN'98*, (Anchorage, U. S. A.), pp. 18–23, 1998.

[14] R. K. De, J. Basak, and S. K. Pal, "Unsupervised neuro-fuzzy feature extraction", in *Proc. 5th International Conference on Soft Computing IIZUKA'98*, (Iizuka, Fukuoka, Japan), October 16–20, pp. 577–580, 1998.

[15] R. K. De and S. K. Pal, "Case-based classification using fuzziness and neural networks", *Proc. Colloquium on Knowledge Discovery and Data Mining* (London, U. K.), May 07–08, 1998.