ON SOME NEW METHODOLOGIES FOR PATTERN RECOGNITION AIDED BY SELF-ORGANIZING MAPS

Arijit Laha
Institute for Development and Research in Banking Technology
Hyderabad - 500057
India.

A thesis submitted to the *Indian Statistical Institute*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2005

ACKNOWLEDGEMENTS

This work is done under supervision of Prof. Nikhil R. Pal. I would like to express my most heartfelt gratitude to him for his constant guidance and painstaking care that enabled me carrying out quality research work. He has been a constant source of encouragement and inspiration while I went through several changes in my professional life during last few years which threatened discontinuation of the research work. He has been the teacher who taught me how to proceed systematically for solving research problems.

I thank Prof. Bhabatosh Chanda for his kind help during the research and imparting valuable knowledge in the field of image processing. I also thank Prof. Jyotirmoy Das for sharing his vast knowledge in the field of remote sensing as well as in capacity of the head of the unit allowing me to occasionally utilize computation and communication resources in ECSU. Along with Prof. Pal I thank Prof. Chanda and Prof. Das for allowing our joint works to be included in this thesis.

I would like to thank my friends in ISI Dr. Debrup Chakrabarty, Mr. Partha P. Mohanta and Mr. Subhasis Bhattacharya for their help. I also thank my former colleagues Dr. A. K. Laha, Mrs. S. Samaddar and Mr. N. Ghosh in NIMC as well as all my colleagues in IDRBT for their encouragement and support.

I would like to record my gratitude for my wife Susmita, my parents and my brother for their unstinting support and courage, but for which I could never have left a financially rewarding career in IT industry and follow my heart to an intellectually satisfying career in academics.

Last but never the least, I thank my little daughter Deya. In moments of crisis, her single smile could lift the cloud of despair hanging overhead and make me feel again the grace and beauty of life.

IDRBT, Hyderabad — Arijit Laha
2005

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

In this thesis we develop several techniques for performing different pattern recognition tasks. In particular, the pattern recognition tasks considered here are classification and vector quantization. We propose several methods for designing classifiers and address various issues involved in the task. For vector quantization, we develop a method for image compression with superior psychovisual reproduction quality. We also propose a method for fast codebook search in a vector quantizer. We exploit different properties of Self-organizing Map (SOM) network for developing these methods. Along with SOM, we also use fuzzy sets theory and Dempster-Shafer theory of evidence to design classifiers with enhanced performance. In the thesis we also report results of some empirical studies on the robustness of SOM with respect to topology preservation under several modification of basic SOM. In the following we provide a brief introduction to the pattern recognition problem and its different aspects, including the techniques used in the thesis.

Pattern recognition (PR) is the most important trait of cognitive ability, be it of humans or animals. The ability to recognize patterns is central to intelligent behavior. We receive signals from environment through our sensory organs which are processed by the brain to generate suitable responses. The whole process involves extraction of information from the sensory signals, processing it using the information stored in the brain to reach a decision that induces some action. All these information we work with are represented as patterns. We recognize voices, known faces, scenes, written letters and a multitude of other objects in our everyday life. In other words, our very survival hinges on our pattern recognition ability. Even more remarkable is the fact that more often than not we perform these tasks in non-ideal or noisy environments. Not only we use pattern recognition with actual signals from the environment, but also we are capable of doing it at intellectual level. For example, faced with a problem of abstract nature, often we recall a similar problem we have faced earlier or have read about, and get a clue to the solution of the problem in hand.

The pattern recognition ability is so natural to us that we almost take it for granted. However, the exact details of the underlying process is still mostly shrouded in mystery and is in itself a vast field of research involving several disciplines like neurobiology, psychology etc. Here we are mainly concerned with *automated pattern recognition* or the *pattern recognition tasks performed by machines*. Thus we are faced with the task of teaching a machine to recognize patterns. This is, to say the least, a formidable task. Many experts defined the task from different perspectives. Some of them are as follows:

Duda and Hart [88] "pattern recognition, a field concerned with machine recognition

of meaningful regularities in noisy or complex environments."

Pavlidis [273] "the word *pattern* is derived from the same root as the word *patron* and, in its original use, means something which is set up as a perfect example to be imitated. Thus pattern recognition means the identification of the ideal which a given object is made after."

Bezdek [31] "pattern recognition is a *search* for *structure* in data."

Schalkoff [301] "Pattern recognition (PR) is the science that concerns the description or classification (recognition) of measurements."

Essentially the discipline of Pattern Recognition (henceforth we shall use the term "pattern recognition" to mean "pattern recognition by machine" unless stated otherwise) deals with the problem of developing algorithms and methodologies/devices that can enable the computer-implementation of many of the recognition tasks that humans normally perform. The motivation is to perform these tasks more accurately, or faster, and perhaps, more economically than humans and, in many cases, to release humans from drudgery resulting from performing routine recognition tasks repetitively and mechanically. The scope of PR also encompasses tasks humans are not good at, such as reading bar codes, supervising manufacturing and other processes under hazardous conditions etc. The goal of pattern recognition research is to devise ways and means of automating certain decision-making processes that lead to classification and recognition.

The field of pattern recognition has witnessed a tremendous amount of research activity in last few decades. This fact can be discerned from the numerous books [32, 82, 89, 271, 270, 273, 301] devoted to it. Pattern recognition can be broadly divided into three branches, namely, feature analysis, pattern classification and clustering [261]. Excellent review of the major approaches in these branches can be found in [227], [139, 189] and [140, 353] respectively. Though the subject has considerably matured during the last five decades, new challenges continue to come up. There are several factors contributing to the high level of activities we find in PR. The cross-fertilization of ideas from several disciplines such as computer science, mathematics, statistics, physics, psychology, neurobiology, engineering and cognitive science opens new approaches to the solutions of the PR problems. Secondly, rapid developments in technology, especially for sensors, has enabled deployment of automated systems in different fields such as manufacturing, remote sensing, surveillance, medical diagnostic etc. whose functioning are largely dependent on various pattern recognition techniques. Lastly, due to cheap availability and huge proliferation of computers and information technology in a wide variety of organizations, there is a constant need for developing PR techniques for summarizing/exploring mountain-load of accumulated data for useful nuggets of knowledge. So there is an explosive growth of new fields of applications. Here we name a few, data mining for business

and scientific applications, bioinformatics and computational genomic, financial computation for prediction of various financial and economic indicators, land cover analysis and weather prediction from remotely sensed data, image and document retrieval etc. All these applications fuel an ever increasing demand for new PR techniques capable of handling more challenging and computationally demanding tasks.

The rest of this chapter is organized as follows: in section 1.2 an overview of the various steps of pattern recognition task is provided, in section 1.3 different approaches for solving the pattern recognition problems are discussed and in section 1.4 the scope of the current thesis is provided.

1.2 Steps in a pattern recognition task

A typical pattern recognition system consists of three phases, namely, *data acquisition, feature extraction/selection* and using the extracted/selected features for performing *classification* or *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 extraction/selection phase, where the dimensionality of the data is reduced by retaining the discriminating power of the original data set to the extent possible. This stage significantly influences the performance of the PR system because only good features can lead to good PR systems. Finally, in the classification/clustering phase, the extracted/selected features are passed on to the classifier/clustering algorithm that evaluates the incoming information and makes a decision. This phase basically establishes a transformation between the features and the classes/clusters. There are various approaches, such as statistical, neural and fuzzy set-theoretic approaches for realizing this transformation.

- There are many statistical methods of classification including Bayesian rule, nearest neighbor rule, linear discriminant functions, perceptron rule, nearest prototype rule. Statistical clustering algorithms include k-means algorithm, mixture decomposition algorithm and others. [82, 89].

- Using artificial neural networks also there are many clustering and classification algorithms, such as Multi-layer Perceptron (MLP) [297], Radial Basis Function (RBF) network [213], Self-organizing Map (SOM) [169], Support Vector Machine (SVM) [333], Learning Vector Quantizer (LVQ) [169] etc.

- Fuzzy set theory has also been used to design classification and clustering algorithms. To name a few, we have Fuzzy Rule based Classifiers [32, 191], Fuzzy

k-nn classifiers [158], Fuzzy c-means (FCM) clustering algorithm [31] and its many variants, possibilistic clustering algorithms [32] etc.

1.2.1 Data acquisition

Pattern recognition techniques are applicable in a wide variety of problem domains, where the data 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 as $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, a set of N feature vectors in the p-dimensional measurement space Ω_X. An s-th object, $s = 1, 2, \ldots, N$, observed in the process has vector $\mathbf{x}_s$ as its numerical representation; x_{si} is the i-th $(i = 1, 2, \ldots, p)$ feature value associated with the s-th object. Relational data is a set of N^2 numerical relationships, say $\{r_{sq}\}$, between pairs of objects. In other words, r_{sq} represents the extent to which s-th and q-th objects are related in the sense of some binary relationship ρ. If the objects that are pairwise related by ρ are called $O = \{o_1, o_2, \ldots, o_N\}$, then $\rho : O \times O \rightarrow \Re$. In the current thesis we shall deal exclusively with object data.

The data acquired is utilized to develop a system in the following manner:
Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \subset \Re^p$ (often known as *source* space) and $Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\} \subset \Re^t$ (*target* space) and let there be an unknown function $\mathbf{S} : \Re^p \Rightarrow \Re^t$ such that $\mathbf{y}_k = \mathbf{S}(\mathbf{x}_k)\ \forall k = 1, ..., N$. In other words, there is an unknown function $\mathbf{S}$ which transforms $\mathbf{x}$ to $\mathbf{y}$. $\mathbf{S}$ can represent many types of systems. $\mathbf{S}$ can be an *algebraic system*, where $\mathbf{S}$ does not evolve with time, and thus can be represented by a set of algebraic equations. On the other hand, $\mathbf{S}$ can be a dynamical system, which evolves with time, and hence differential equations are required to characterize them. Again, $\mathbf{S}$ can be characterized by the type of output it produces. The output may be continuous numerical valued or they can be decisions as in a classifier.

Finding $\mathbf{S}$ from the input-output data (X, Y) is often called *system identification*. $\mathbf{S}$ can be of two types: regression or function approximation (FA) type when the elements in Y are continuous, and classifier type when Y contains class labels (or categorical decisions). In both cases it is assumed that the input-output data (X, Y) are generated from a time invariant but unknown probability distribution. The data set $T = (X, Y) = \{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, ..., N\}$ that is used to find $\mathbf{S}$ is called a training set. During training, $\mathbf{x}_i, (i = 1, 2, ..., N)$ is used as the input and the outputs $\mathbf{y}_i, (i = 1, 2, ..., N)$ acts as a teacher. The design methodology uses this training set T to learn (build) a system with a hope that the obtained system will work well not only on the training inputs but also on data points which are not present in the training set. The performance of such a

system is measured by its performance on future data, commonly called test data. The test data set is independent of the training set, and it is also assumed that the test data follow the same probability distribution as that of the training data. There is another kind of application where the target data set Y is not available. Only the input X is present and the problem is to find homogeneous groups in X. This is clustering of X.

1.2.2 Feature selection/extraction

Feature extraction and selection tasks, along with *feature ranking* form an important branch of pattern recognition activity known as **feature analysis**. All features that are present in a data set may not be useful for the task at hand. Some features present may be redundant and some may be bad too. Thus selecting the most relevant and useful features from a given set of features is useful as it helps in building systems with low complexity and may save computational time as well as economize future data collection efforts. Also it has been shown that due to finite sample size effects [123, 139] reducing the number of features may lead to an improvement of the prediction accuracy. Moreover, a system built with a smaller number of features is more readable, has less parameters and is expected to have better generalization abilities.

Feature analysis deals with finding a transform $\Phi : \Re^p \rightarrow \Re^q$ using a criterion J on a training set $T = (X, Y) = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \Re^p, \mathbf{y}_i \in \Re^t, i = 1, 2, ..., N\}$. Typically, J is related to the problem that we intend to solve using T. The transform Φ is said to perform *dimensionality reduction* if $q < p$.

Feature analysis can be divided into three basic types: feature extraction, feature selection and feature ranking. Feature extraction is a method to generate a q dimensional vector from a given p dimensional input vector. For feature extraction it is not necessary that $q < p$. There may be applications where one needs to find a richer description of objects by increasing the number of features. Extracting additional features from a given feature vector is common in many image processing and other signal processing applications. However, while designing a pattern recognition system, the designer is mostly interested in transformations which perform a reduction in the dimensionality of the input space.

A special case of feature extraction is feature selection, where the components of $\Phi(\mathbf{x})$ are some components of $\mathbf{x}$, i.e., in this process some components of the original vectors are discarded and others are retained as they are. Thus, in feature selection a subset of the original set of features is selected using some criteria.

Feature ranking methods aim at ranking features according to their suitability for the task at hand. The ranking algorithm assigns some real number to each of the features,

and one can order the features according to these real numbers. Feature selection may then be performed based on these ranks. But ranking individual features is not the best way to look at the problem because two correlated features may get high ranks, while only one of them is required for the task.

Feature selection algorithms can be characterized based on two attributes namely the feature subset evaluation criterion function J and the search technique. J can be of two types, problem independent criteria and problem dependent criteria [227]. The algorithms using the first category are called *filter algorithms*. Such criteria are based on distance or separability measures, information measures, dependency or correlation measures and consistency measures. For the second category, the criteria is based on the performances of feature subsets in a specific pattern recognition task such as the classification error rate. [71]. The algorithms employing such criterion are also called *wrapper methods*. Once the criteria J gets fixed, the problem of feature selection reduces to a search problem. There are 2^p possible subsets of the feature. The number of feature subsets with q or less dimensions is $\sum_{i=1}^{q} \binom{p}{i}$. An exhaustive search for an optimal q-space would require examining $\binom{p}{q}$ possible q subsets of the original feature set. The number of possibilities grows exponentially making exhaustive search impractical even for moderate values of p. However, there exists heuristics like *branch and bound* and *beam search* that reduce the search space. A recent survey of feature selection algorithms can be found in [227]. In [227] the authors also provide a taxonomy of state-of-the-art feature selection algorithms based on the evaluation criteria, search techniques and target pattern recognition tasks. Also there are many methods which do not explicitly use a search technique, but use a transformation on the available features to obtain a reduced set of features. In this thesis we do not consider the feature analysis further.

1.2.3 Classification and clustering

The problem of designing a classifier 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. In real life, the complete description of the classes is not known. We have instead, a finite and usually a small number of samples which often provide partial information for optimal design of classification algorithms. Typically, 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 classifier systems are designed. If the class-conditional probability densities are known, statistical methods such as Bayes decision theory can be applied directly to obtain the optimal solution [88]. However, if the class-conditional densities are not known, the system must be developed

by *learning* from the data.

In pattern recognition, the learning tasks are mainly of two types. When the training data contain two parts, input and output, and the learning algorithm uses this output information, the learning is known as *supervised* learning. Here the goal of learning is to model the mapping of the input data points to the output. If the outputs are discrete class labels, the system is called a classifier; while for continuous valued output, the system is known as a function approximation type system. On the other hand, when the training data set does not contain the label information, the learning is known as *unsupervised* learning. Here the aim is to discover homogeneous groups in the data.

To realize a PR system the system designer can choose among several available and well developed approaches apart from pure statistical ones. These approaches include *neural networks* and *fuzzy set theoretic approaches*. There is also immense research activity directed towards developing pattern recognition systems based on more than one of these approaches, known collectively as *computational intelligence* approaches, which apart from the above include genetic algorithms and rough sets.

1.3 Approaches to Pattern Recognition

In this section we discuss in some details four approaches for solving the pattern recognition problems. They are namely, statistical approach, neural network based approach, fuzzy set theoretic approach and Dempster-Shafer evidence theoretic approach. The works reported in the current thesis use mostly the later three approaches. However, the statistical approach being the most widely used one and historically often used as a benchmark for comparing other approaches, we include it in our discussions. Each of these approaches provide a number of algorithms for performing pattern classification as well as clustering tasks. In the following we discuss some of these algorithms under each approaches for both the tasks.

1.3.1 Statistical Pattern Recognition

Given a pattern vector $\mathbf{x} \in \Re^p$, the classification task is viewed as one of assigning $\mathbf{x}$ into one of a set of c *known* classes $\{\omega_1, \omega_2, \ldots, \omega_c\}$, where a class can be viewed as a source of patterns whose distribution in the feature space is governed by a probability density specific to the class [140]. On the other hand, the task of clustering is defined in [140] as follows:

> Clustering techniques attempt to group patterns so that the classes thereby

obtained reflect the different pattern generation processes represented in the pattern set.

Thus, according to this definition the task of clustering is aimed at discovering *previously unknown* groups from the data set. Here first we look into the classification tasks, then we shall discuss clustering in the later part of this subsection.

Classification

Bayesian decision theory [26] is a fundamental statistical approach to the problem of pattern classification [89]. Under this approach, the decision problem is posed in probabilistic terms and it is assumed that all relevant probability values are known. A pattern is represented in terms of p measurement or feature values and treated as a point in the p-dimensional Euclidean space $\Re^p$, often called the *feature space*. Here each pattern belonging to a class is considered to fall within a compact region of the space. Thus, the distribution of the pattern classes can be represented by a set of probability density (mass) functions, known as *class conditional probability* $p(\mathbf{x}|\omega_i), i = 1, 2, \ldots, c$. To classify a new pattern, the Bayesian approach uses the parameters of the density functions to calculate the posterior probabilities $p(\omega_i|\mathbf{x}), i = 1, 2, \ldots, c$ of the new pattern belonging to different classes. Bayes rule is used for computing the *posterior probability* $p(\omega_i|\mathbf{x})$ as

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

where $P(\omega_i)$ is the *a priori* probability for class ω_i and

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j).$$

The Bayes decision rule minimizes the *risk* or *expected loss* by assigning $\mathbf{x}$ to class ω_i such that the conditional risk $R(\omega_i|\mathbf{x}) < R(\omega_j|\mathbf{x})\forall i \neq j$, where

$$R(\omega_i|\mathbf{x}) = \sum_{j=1}^{c} L(\omega_i, \omega_j)p(\omega_i|\mathbf{x}).$$

$L(\omega_i, \omega_j)$ is the element of loss matrix indicating the loss incurred in assigning $\mathbf{x}$ to ω_i while its true class is ω_j. In case of 0/1 loss function the decision rule simplifies to *maximum a posteriori* (MAP) rule:

$$\text{Assign } \mathbf{x} \text{ to } \omega_i \text{ if } p(\omega_i|\mathbf{x}) > p(\omega_j|\mathbf{x})\forall i \neq j.$$

Bayes rule produces the optimal classification performance if all the class-conditional probabilities are completely known [88]. However, they are usually not known in real-life

problems and must be learned from the available training samples. If the form of the class-conditional densities are known (e.g., multivariate normal), but the parameters of the densities are unknown, then first we estimate the parameters using the training data and use these estimated values to make decisions. Such a classifier is often known as Bayes plug-in classifier [139]. If the form of class-conditional densities is not known then one uses nonparametric density estimation methods like Parzen window [88]. A detailed exposition of Bayesian approach of pattern recognition can be found in [30].

There is a general class of nonparametric classifiers, known as *prototype based classifiers*. Prototypes are a set of patterns, representative of the training data set. A new point $\mathbf{x}$ is classified using the prototypes according to the label(s) of the prototype(s) most similar to it. However, the actual design of the classifier varies depending on the method adopted for generating the set of prototypes. We shall discuss more about prototype based classifiers and develop a novel method for generation of prototype sets in Chapter 4. A special case of prototype based classifiers are *k-nearest neighbor* (k-NN) classifier, where the training set, also called the *reference set*, itself is the set of prototypes. To classify a new pattern $\mathbf{x}$, first k points in the reference set nearest to $\mathbf{x}$ are identified. Then $\mathbf{x}$ is classified using the k-NN rule as follows:

Assign $\mathbf{x}$ to class ω_i if majority of k nearest neighbors of $\mathbf{x}$ are from class ω_i.

The properties of nearest neighbor classifiers are studied by Cover and Hart [68]. They proved that for 1-NN classifiers, the probability of error $P = \lim_{n\to\infty} P_n(e)$, where $P_n(e)$ is the n-sample error rate, is bounded with respect to the optimal Bayesian error rate P^B as follows:

$$P^B \leq P \leq P^B \left(2 - \frac{c}{c-1} P^B\right).$$

Thus, the error rate is bound from above with $2P^B$. It was further shown by Devroye [83] that with increasing k the upper bound approaches the lower bound.

The simplicity as well as performance make NN classifiers lucrative choice for designing classification systems. However, for application in real-life problems, the method, in its naive form, suffers from space complexity of $O(n)$ and computational complexity of $O(pn)$, for classifying each pattern $\mathbf{x} \in \Re^p$. Various methods for reducing the complexity are studied by researchers and a comprehensive account of them can be found in [70]. There are mainly three techniques for reducing the complexity. Using *partial distance*, i.e., the distance calculated in a subspace of the original space, relies upon the fact it is nondecreasing with contribution of additional dimensions [110]. There are methods employing *prestructuring* those build search trees based on the distances among the

training patterns for efficient searching of nearest neighbors [103]. The *editing* techniques remove from the sample data set the samples whose k nearest neighbors come from the same class as itself, thus retaining only the samples corresponding to decision boundaries [349]. There are many variants of k-NN classifiers [70] including those using fuzzy set theory [158] and Dempster-Shafer theory of evidence [76].

The other class of statistical classifiers is based on discriminant functions. Here the form of a discriminant function (i.e., decision boundary) is decided first, then the optimal values of parameters of the function are learned from the training data through gradient descent or relaxation or by other search methods. This approach has its origin in classic work of Fisher [98]. A review of early works on linear discriminant function can be found in [127]. In its simplest form, the linear discriminant functions are capable of classifying patterns from linearly separable classes only. However, for dealing with more complex problems the *generalized* linear discriminant functions [89] map the data nonlinearly into higher dimensional spaces and create linear decision surfaces therein. This is inspired by the famous Cover's *theorem on separability of patterns* [67], which states:

> A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.

Discriminant analysis approach is also known as geometric approach and bears functional similarity with some neural network methods [139]. One such method, the support vector machine (SVM) has become very popular. Here we present a brief overview of two-class SVMs.

The SVM [63, 333, 124] is a machine learning method with its root in statistical learning theory and is being used in many pattern recognition problems. In its basic form SVM is formulated for two-class pattern classification problems. If the training data have linearly separable classes, then SVM finds an optimum decision hyperplane that maximizes the separation between the decision surface and each of the two classes.

Consider a training set $\{\mathbf{x}_i, d_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \Re^p$ is a pattern vector, and $d_i \in \{-1, +1\}$ is the class label associated with $\mathbf{x}_i$. The learning problem of SVM is to find optimum values of the weight vector $\mathbf{w}$ and the bias b of the decision hyperplane such that they satisfy the constraints

$$\mathbf{w}.\mathbf{x}_i + b \geq 1 \text{ for } \forall d_i = +1$$

and

$$\mathbf{w}.\mathbf{x}_i + b \leq 1 \text{ for } \forall d_i = -1,$$

written together

$$d_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 \text{ for } \forall i.$$

This problem is solved by finding the value of $\mathbf{w}$ that minimizes the cost function

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

subject to the above constraints. This is a quadratic optimization problem and can be solved using the method of Lagrange Multipliers. A unique solution exists if the classes are linearly separable. For the optimal hyperplane, the data points which satisfy the equality constraints are known as 'support vectors'. Actually, the dual problem of the above primal optimization problem, set entirely in terms of the training data set, is solved to find the support vectors first and then they are used to compute the optimal hyperplane.

If the classes are overlapping, the optimality problem is reformulated using a set of non-negative scaler variables $\{\xi_i \mid i = 1, ..., N\}$, known as slack variables. Now the constraints take the form

$$d_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } \forall i$$

and the cost function becomes

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i,$$

where C is a regularization parameter that controls the tradeoff between the complexity of the machine and the number of nonseparable points.

To handle classes not linearly separable in the input space, SVMs use inner-product Kernels. This in effect maps the training data in a higher dimensional feature space using a nonlinear mapping ϕ. This action is in accordance with the Cover's theorem of separability [67] stated earlier. Then SVM computes the optimal hyperplane separating the classes in the higher dimensional space. For computing the decision hyperplane inner-product kernels of the form

$$K(\mathbf{x}, \mathbf{x}_i) = (\boldsymbol{\phi}(\mathbf{x}).\boldsymbol{\phi}(\mathbf{x}))$$

are used. They facilitate easier solution of the dual optimization problem. Three popular kernels used in SVMs are

Polynomial kernel of degree d: $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}.\mathbf{x}_i + 1)^d$

Radial Basis Function (RBF): $K(\mathbf{x}, \mathbf{x}_i) = e^{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2}$

and

Sigmoid: $K(\mathbf{x}, \mathbf{x}_i) = \tanh(k\mathbf{x}.\mathbf{x}_i + \theta)$, for $k > 0$ and $\theta < 0$.

In Chapter 4 of this thesis we have compared our proposed classification methods with k-NN and SVM classifiers.

Clustering

Next we consider the clustering problem. In this case, no class label is available. The problem is to find homogeneous groups in the data. For most of the clustering algorithms either the number of groups is known or it is to be assumed. Depending on the information available, there are mainly two types of statistical methods of unsupervised learning, namely, *mixture resolving* and *cluster analysis* [139]. The former is adopted when partial information about the data is available. The set of conditions under which mixture resolving can be applied is as follows [89]:

1. The samples come from a known number of c classes.
2. The forms of class-conditional probability densities $p(\mathbf{x}|\omega_j, \theta_j), j = 1, 2, \ldots, c$ are known.
3. The value of the c parameter vectors $\theta_1, \ldots, \theta_c$ are unknown.

Let $P(\omega_j)$ be the prior probability of class j. Then the probability density of the samples, known as *mixture density*, is represented as a mixture or superposition of c component densities $p(\mathbf{x}|\omega_j, \theta_j)$s as follows:

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \theta_j)P(\omega_j), \text{ where } \theta = [\theta_1, \ldots, \theta_c]^T.$$

The goal of mixture resolving is to estimate the unknown parameter vector θ and $p(\omega_j)$ from the samples and thus decomposing the mixture into its constituent components. Mostly the *Maximum-likelihood Estimate* method is employed for mixture resolving and often the component densities are assumed to be Gaussian [89]. However, due to finite sample size, mixture resolving methods sometime face the problem of *unidentifiability* [322], where no unique solution exists.

There are other types of algorithms which do not make any assumption about the densities but try to partition the data based on similarity. The points belonging to

the same cluster should have more similarity than those belonging to different clusters. Clearly, the concept of *similarity* is fundamental to the definition of a cluster and a measure of similarity between two patterns is essential to the clustering procedures. Commonly, instead of similarity, a measure of *dissimilarity* or distance between the patterns in feature space is used [140]. Most popular metric for continuous features is the Euclidean distance. In addition to this, there are many other distances such as cosine distance, Mohalanobis distance, Tanimoto distance. Various distance measures for strings etc. are also used in clustering procedures depending on the nature of data and problem domain. There are even some non-metric distance measures used with success by several researchers [140].

The clustering algorithms can broadly be divided into two families, hierarchical clustering and partitional clustering. A hierarchical clustering algorithm produces a nested grouping of all patterns and similarity levels at which the groupings are formed. Typically, hierarchical clusters can be represented by a dendrogram. The dendrogram can be broken at various levels to produce different clustering of the data set [140]. There are several variants of hierarchical algorithms such as, single-link, complete-link etc. However, for real-life problems with large data sets the construction of a dendrogram is computationally prohibitive [140]. Henceforth we shall not be discussing the hierarchical algorithms.

A partitional clustering algorithm produces a partition of the data. Most partitional algorithms require the number of clusters to be supplied externally, as a parameter, to the algorithm. A partitional clustering algorithm typically uses a criterion function whose optimization leads to the production of the clusters. Most intuitive and frequently used criterion function is the *squared error criterion*, which is known to work very well if the clusters are isolated and compact. Given a data set $X = \{\mathbf{x}_i | \mathbf{x}_i \in \Re^p \text{ and } i = 1, \ldots, N\}$, and a clustering $\mathcal{L}$, where $\{\mathbf{c}_j | \mathbf{c}_j \in \Re^p \text{ and } j = 1, \ldots, K\}$ are the set of K cluster centroids, the squared error is defined as

$$e^2(X, \mathcal{L}) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \left\| \mathbf{x}_i^{(j)} - \mathbf{c}_j \right\|^2,$$

where $\mathbf{c}_j$ is the cluster center closest to $\mathbf{x}_i^{(j)}$ and n_j is the number of data points for which $\mathbf{c}_j$ is the closest cluster center. Popular k-means algorithm [89] (also known as Generalized Lloyd's Algorithm (GLA) [228]) uses the squared error criterion. The algorithm is as follows:

1. Choose k cluster centers randomly from the hypervolume containing the data or from the data points themselves.

2. Assign each pattern to the closest cluster center.

3. Recompute the cluster centers using the set of closest patterns.

4. If a convergence criterion is not met, go to step 2. Typical convergence criteria include "no reassignment of the patterns to cluster centers" or "insignificant decrease in squared error".

Though very popular, the algorithm is highly sensitive to the initialization. Several variants of the basic algorithm, mostly involving improved initialization for attaining global minima can be found in [10]. Further, the number of clusters is fixed in k-means algorithm, while ideally that should depend on the distribution of data. To change the number of clusters, ISODATA algorithm [18] allows merging and splitting of the clusters depending on the data distribution. Among other variants, *dynamic clustering algorithm* [85] permits cluster representations other than centroids, Mao and Jain [238] used regularized Mahalanobis distance to obtain hyperellipsoidal clusters. In a recent paper Yu [364] provided a generalized framework for k-means clustering that encompasses other objective function based partitional algorithms including various variants of their fuzzy counterparts also. The paper also reviews the issue of cluster validity. A survey of state-of-the-art clustering techniques can be found in [353]. An excellent survey of clustering techniques from data mining perspective can be found in [29]. We shall not be working with the statistical approaches in this thesis.

1.3.2 Neural Networks for Pattern Recognition

Artificial Neural Network (ANN) or simply Neural Network (NN) is a computational paradigm inspired by the information processing system found in biological world. The brain is at the core of the system which has a huge number of small processing units called "neurons". They are heavily interconnected and operate parallelly. Though each neuron individually performs some elementary transformation of the signal at a very slow speed compared to a digital/electronic device, the brain as a whole accomplishes very complex perceptual and other cognitive tasks within a very small time. The study of neural networks tries to analyze and implement this computational paradigm using electronic devices. A neural network can be defined as follows [124]:

A neural network is a massively parallel distributed processor made up of simple processing units, which has a propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. *Knowledge is acquired by the network from its environment through a learning process.*

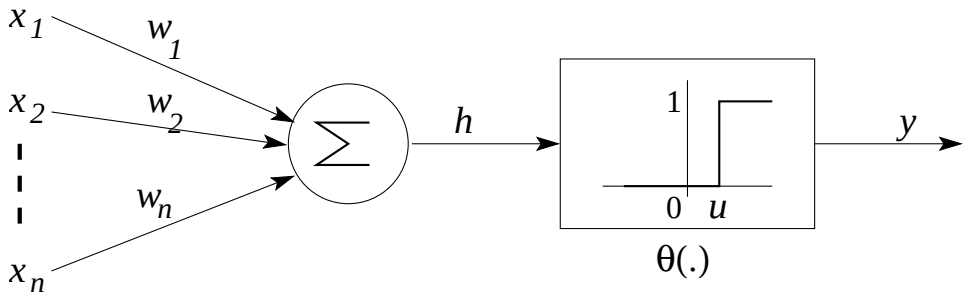

Figure 1.1: McCulloch-Pitts neuron

2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

Research in neural networks started from the search of a feasible computational model of the nervous system. In 1943, McCulloch and Pitts proposed a binary threshold unit (Figure 1.1) as a computational model for an artificial neuron [243]. Mathematically, the neuron computes a weighted sum of its input x_is and apply a threshold function to generate a binary (0/1) output. Thus, formally the output y can be written as:

$$y = \theta\left(\sum_{i=1}^{n} w_i x_i - u\right), \text{ where, } \theta(.) \text{ is a unit step function at 0.}$$

The positive weights correspond to *excitatory* synapses and negative weights model *inhibitory* ones. They proved that a synchronous arrangement of such units with suitably chosen weights are capable of universal computation. The McCulloch-Pitts neuron can be generalized in various ways, most importantly using activation functions $\theta(.)$, other than threshold, such as linear, piecewise linear, logistic, Gaussian etc. In 1949, Hebb [125] pointed out the fact that neural pathways are strengthened each time they are used, a concept fundamentally essential to the ways in which a human learns. These two works probably were the pioneers in neural network research. In the 1950's, Rosenblatt's work [296] resulted in a two-layer network, the perceptron, which was capable of learning certain classification tasks by adjusting connection weights. Although the perceptron was successful in classifying certain patterns, it had some limitations. The perceptron was not able to learn the classic XOR (exclusive or) problem or in general, any linearly nonseparable classification problem. In [248] Minsky and Papert systematically pointed out this and few other deficiencies of perceptron. Further, though multilayer networks are capable of solving linearly nonseparable problems, at that point of time no effective learning algorithm for them was known. Consequently, interest in neural networks dipped for quite sometime. However, the perceptron had laid foundations for later work in neural computing. Interest in neural networks again revived in the eighties with some pathbreaking works like self-organizing map [169], the Hopfield network [130], multilayer

perceptron [297] as well as error-backpropagation learning algorithm [346], adaptive resonance theory [43] etc. An excellent commentary on the development of neural networks can be found in the survey paper [244].

The historical perspective of neural networks was in building biologically plausible systems. The objective was more towards building a model for the nervous system. But in the last decade, it was observed that neural networks, both the traditional architectures, and their variants, have great potential to solve real life problems even though they may not mimic computation in the brain. The thrust of research for the last ten years were more to develop systems which can be used to solve practical problems than to look for biological plausibility. Of course, there have been many advances towards modelling the biological nervous systems and thereby gaining new insights into the process of brain functioning. These issues are nowadays dealt within the subject of computational neuroscience [92, 327]. We shall not be venturing further into those issues.

Neural networks are lucrative for their interesting properties of learning and generalization. They can be applied without any assumptions regarding the distributions and other statistical characteristics of data. Theoretical issues about learning have been extensively studied. Valiant [330] proposed a mathematical theory of learning that addressed the issues of learnability and computational complexity of learning in a restricted framework called the "Probably approximately correct" (PAC) learning. The PAC framework provides a reasonable setup to study the theoretical basis of learning. There have been a variety of theoretical studies encompassing learning in general and in particular, learning in neural networks [333, 336]. All these studies helped to gain a better understanding of the learning and generalization process. Consequently these have helped to develop better learning systems.

Classification

Artificial neural networks have been successfully used in pattern recognition and many other fields. There are specific neural architectures which can perform the various pattern recognition tasks [37, 124]. For example, there are neural networks that can learn the input-output mapping from a set of examples (or training data) through a learning algorithm. During the learning stage the network adjusts the connection weights to implement the desired mapping. As discussed earlier, for pattern classification task, usually labelled data is used and the training is supervised one. Perceptrons, the simplest of the NNs can implement a linear decision boundary of the form

$$y = \sum_{i=1}^{p} w_i x_i - b = 0,$$

and classify the patterns from two linearly separable classes according to the rule:

$$\mathbf{x} \in \text{ class 1 if } y \geq 0, \text{ otherwise } \mathbf{x} \in \text{ class 2.}$$

The required values of the weights $\{w_i\}$ can be learned from labelled (1 for class 1 and 0 for class 2) training data using the perceptron learning algorithm based on error-correction principle [296] as follows:

1. Initialize the weights and the threshold b to small random numbers.
2. Present an augmented (including an input 1 corresponding to bias $b = w_0$) pattern vector $\mathbf{x} = [1, x_1, x_2, \ldots, x_p]^T$ as the input and evaluate the output y.
3. Update the augmented weight vector $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_p]^T$ using the eq:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta(d - y)\mathbf{x}.$$

   where d is the desired output (i.e., the label of $\mathbf{x}$), t is the iteration number and $\eta (0 \geq \eta \geq 1)$ is the learning parameter. The learning rule is also known as *delta rule* and attributed to Widrow and Hoff [347].

The algorithm is terminated when all the training data are correctly classified. Rosenblatt [296] also proved that if the training data comes from two linearly separable classes, the algorithm converges after a finite number of iterations (*perceptron convergence theorem*).

The popular feed-forward networks like the multilayer perceptrons (MLP) [297], radial basis function (RBF) [213] networks can learn any non-linear input-output mapping under a fairly general set of conditions [124, 131]. MLP architecture is consisted of one input layer of nodes (a fan-out layer) and one or more hidden layers along with one output layer. The output of the nodes in one layer form the input of the nodes in the next layer. The nodes are completely connected between two successive layers. Like perceptron, the learning algorithm of MLP is also based on error-correction. However, due to multilayer architecture, the error of hidden nodes can be estimated indirectly only. Thus the popular *error back-propagation* learning algorithm [346] computes the local gradient δ_i^l for the i-th node in layer l as follows:

$$\begin{aligned}
\delta_i^L &= g'(h_i^L)\left[d_i(t) - y_i^L\right] \quad \text{if } l = L, \text{ i.e., output layer} \\
&\text{and} \\
\delta_i^l &= g'(h_i^l)\sum_j w_{ij}\delta_j^{l+1} \qquad \text{for } l = (L-1, \ldots, 1), \text{ nodes in hidden layers,}
\end{aligned}$$

where, g' is the derivative of the activation function g, h_i^l is the weighted sum of the inputs to the node and w_{ij} is the weight connecting the node i in layer l and node j in

layer $l+1$. The weights are updated using the equation:

$$w_{ji}^{l}(t+1) = w_{ji}^{l}(t) - \eta \frac{\partial E(t)}{\partial w_{ji}^{l}(t)} = w_{ji}^{l}(t) + \eta \Delta w_{ji}^{l}(t)$$

where

$$\Delta w_{ji}^{l} = \eta \delta_{i}^{j} y_{j}^{l-1}.$$

This is also known as *generalized delta rule*. Following Werbos's original backpropagation algorithm [346], various modified as well as alternative approaches of learning has been used by researchers. Detailed accounts of these can be found in [37, 124, 346].

While the MLP generally realizes complicated decision boundaries, RBF works on the principle of finding a linear decision boundary in a higher dimension, to which the data is mapped using a set of nonlinear basis functions. The theoretical grounding for RBF can be traced to the Cover's theorem of separability [67, 124]. RBF is a three-layer network, with one hidden layer containing radially symmetric nonlinear activation functions and output layer containing perceptrons with linear activation function. The hidden layer nodes map the input data nonlinearly to a (usually) higher dimensional space, where the classes are linearly separable. The output nodes implement the linear discriminant functions in the higher dimensional space. Usually the learning algorithm is a hybrid one. It is performed in two stages, first the parameters of the basis functions are learned in an unsupervised manner, then the weights of the output nodes are learned using a supervised gradient descent method. However, fully supervised, single stage learning algorithms also exist [124]. These networks have been successfully used as classifiers and function approximation tools for variety of applications.

Variants of the *Learning Vector Quantizer* (LVQ) [169], LVQ1 and LVQ2 are supervised learning techniques, which fine tune the placement of prototype vectors obtained by a vector quantization [110] algorithm for improved classification performance. In [169] Kohonen described these variants of LVQ techniques, where the vector quantization is performed with Self-organizing Map (SOM). More details about various neural networks used for classification can be found in [37, 124].

Clustering

Most of the Neural network based clustering techniques employ competitive learning rules. In a competitive learning network, the output nodes compete among themselves for activation and only one output unit is activated for an input. The idea of competitive learning may be traced back to the works of von der Malsburg [341]. Competitive learning also found to exist in biological neural networks [90]. Competitive learning categorizes the input data so that similar patterns are grouped by the network and represented by a single output unit.

In its simplest form, a competitive learning network consists of an input layer and a single output layer. Each of the units i in the output layer are connected to each unit j in the input layer through a weighted connection w_{ij}. Thus, each output node is associated with a weight vector $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{ip}]^T$. Each output unit is also connected with other output units via *inhibitory* connections while has a self-feedback excitatory connection. When an input $\mathbf{x} \in \Re^p$ is presented to the network, only the output unit with weight vector most similar to the input becomes winner. If Euclidean distance is used, then for the winner node r

$$\|\mathbf{w}_r - \mathbf{x}\| \leq \|\mathbf{w}_i - \mathbf{x}\| \, \forall i \neq r$$

The competitive learning rule can be stated as

$$\Delta \mathbf{w}_i(\mathbf{x}) = \begin{cases} \eta(\mathbf{x} - \mathbf{w}_i) & \text{if } i = r \\ 0 & \text{if } i \neq r. \end{cases}$$

It may be noted that only the winner node is updated. This is known as *winner-takes-all* strategy. Competitive learning can be used to find clusters while the weight vectors become the cluster centers or prototypes. It can also be seen that for stability of the system, the learning parameter should reduce gradually, approaching zero towards the end of training. Otherwise, in successive iteration a data point may be assigned to different nodes.

Self-organizing Map (SOM) [169] and Adaptive Resonance Theory (ART) [43] are two very popular neural clustering techniques [353]. In SOM the output nodes are arranged in a regular lattice structure and a neighborhood function of the winner over the lattice is defined. During training the winner as well as neighboring nodes are updated. This results in several interesting properties of SOM. Since SOM is discussed in details in the next chapter, here we do not duplicate the effort. The ART is developed by Carpenter and Grossberg to solve the *stability-plasticity* dilemma. The ability of a network to adapt to new data is known as the *plasticity* of the network. However, as mentioned earlier, for achieving stability in competitive learning networks, it is imperative that the learning rate decreases gradually towards *zero*. This leads to freezing of the network. Hence, it loses the ability to adapt to new data. ART is capable of learning arbitrary input patterns in a stable, fast and self-organizing way, thus, overcoming the effect of stopping the learning that plagues many other competitive networks [353]. However, ART itself is not a neural network, rather a learning theory, that deals with resonance in neural circuits resulting in a fast learning. The theory encompasses a large class of neural networks.

The simplest ART network is a vector quantizer – it accepts as input a vector and classifies it into a category depending on the stored prototype it most closely resembles.

Once a prototype is found, it is modified (trained) to resemble the input vector. If the input vector does not match any stored prototype within a certain tolerance, represented by a vigilance parameter $\rho(0 \leq \rho \leq 1)$, then a new category is created by storing a new prototype similar to the input vector. Consequently, no stored prototype is ever modified unless it matches the input vector within a certain tolerance. This means that an ART network has both plasticity and stability; new categories can be formed when the environment does not match any of the stored prototypes, but the environment cannot change stored prototypes unless they are sufficiently similar. There are many different variations of ART available today. For example, ART1 performs unsupervised learning for binary input patterns, ART2 is modified to handle both analog and binary input patterns and ART3 performs parallel searches of distributed recognition codes in a multilevel network hierarchy. ARTMAP combines two ART modules to perform supervised learning. A survey of various ART networks can be found in [353]. The same article also includes a survey of many other clustering algorithms.

Among other neural networks, the Hopfield network can be used as an associative memory, which can store and recall patterns [126, 371]. In addition, Hopfield nets have been successfully used for various computationally hard combinatorial optimization problems [126]. There are numerous work in the literature which discusses variants of these networks and use of them for various practical problems. A glimpse of these voluminous research can be found in many books related to neural networks [37, 124, 126, 336, 371, 372, 61].

In the current thesis, we are mainly interested in solving the problems of pattern classification and vector quantization. The works described in the thesis use Self-organizing Map (SOM) extensively. These include empirical studies on some properties of SOM relevant to pattern recognition tasks, development of SOM-based algorithms for designing prototype based classifiers and fuzzy rule based classifiers, design of vector quantizers for image compression and development of fast codebook searching methods for VQ. We shall provide a detailed description of SOM in the next chapter. In the current thesis we shall be referring to other NN models mainly for the purpose of comparing their performances with the methods proposed here.

1.3.3 Fuzzy Set Theoretic Approach to Pattern Recognition

Human mind is adept at dealing with uncertain, imprecise information. It is crucial to us because our sensory organs allow us to receive signals from the physical environment in a vague, imprecise form and our brain processes them without the aid of any kind of exact mathematical reasoning. Still we are capable of reaching at surprisingly good and robust decisions. The procedure is often known as *approximate reasoning*. So it is

expected that the PR systems developed with capability of handling uncertainty would be more robust as well as their working might be closer to human understanding. However, handling uncertainty using conventional mathematical framework is almost impossible, since conventional mathematical structures are based on the concept of classical sets, where an object is either an element of a given set or it is not, i.e., the membership can have only two values, 1 and 0. This situation is changed in 1965 by Zadeh [365] with the introduction of *fuzzy sets*. The concept of fuzzy sets is based on the premise that key elements of human thinking involve dealing with objects whose membership to one category or another is graded or gradual rather than abrupt. It also focuses on the fact that the logic behind human reasoning process is not bi-valued or multivalued but a logic dealing with fuzzy truth and fuzzy rules of inferences. Thus fuzzy sets can be used for representing imprecision/vagueness in everyday life in a novel way. This theory provides a reasoning system that deals with the imprecise concepts embodied in fuzzy sets. It also provides an approximate yet robust means for modelling systems which are too complex or ill-defined to admit precise mathematical analysis [366, 367]. Fuzzy set theory is a generalization of classical set theory and it has greater flexibility to capture and deal with various aspects of uncertainty, imprecision and incompleteness/imperfection in information about a situation. Fuzzy set theory allows one to model complex and nonlinear input-output mappings in terms of fuzzy if-then rules [282], thus simplifying the task of system modelling. A brief overview on the application of fuzzy sets for pattern recognition can be found in [279].

In 1975 Mamdani and Assilian proposed a method of developing fuzzy controllers [236]. It had the striking feature that it did not require any mathematical model of the system to be controlled, instead it had used a set of simple control rules supplied by the human operator. In essence, the idea is to model an experienced human operator. In controller applications the input variables as well as the output signal are almost always continuous valued. Hence, in a Mamdani controller a fuzzy rule is viewed as a fuzzy Cartesian product of its fuzzy inputs and fuzzy outputs, i.e, a rule represents a fuzzy point or granule (often overlapping) in the input-output space. Thus, the rule base is a collection of fuzzy points determining the input-output relationship.

In pattern recognition problems we are faced with imprecise information from several fronts. Instrumental error and environmental noise contribute to imprecision in measurements. The class boundaries often overlap due to natural pattern distribution and/or fuzziness or randomness of observation. When a new pattern falls into such a region, classifying that to a particular class may lead to erroneous decisions. Instead, it is much desirable for the system to generate a graded membership pertaining to the belonging of the pattern to multiple classes. The membership values will reflect the confidence of the system in favor of one decision or another. Further, since it is inspired by human rea-

soning process, fuzzy set theoretic methods lead to the design of interpretable systems. So one can study the system to get a good understanding of the process the system is modelling. All these facts have made fuzzy set theory a very powerful tool for designing pattern recognition systems.

Classification

Though there is no clear-cut definition of a fuzzy classifier. According to Kuncheva [191], some semi-formal definitions are as follows:

1. A fuzzy classifier is any classifier which uses fuzzy sets either during its training or during its operation.

2. A fuzzy or probabilistic classifier, is any possibilistic classifier for which

$$\sum_{i=1}^{c} \mu_i(\mathbf{x}) = 1$$

   where the $\mathbf{x} \in \Re^p$ is a data point and c is the number of classes.

   A possibilistic classifier [32] is defined as a mapping

$$D_p : \Re^p \rightarrow [0,1]^c - \mathbf{0}$$

   i.e, it assigns a soft class label (vector) $\mu(\mathbf{x}) = [\mu_1(\mathbf{x}), \ldots, \mu_c(\mathbf{x})]^T$, where the value $\mu_i(\mathbf{x})$ denotes the degree of confidence for the hypothesis that $\mathbf{x}$ belongs to the i-th class.

3. A fuzzy classifier is a fuzzy if-then inference system (a fuzzy rule base system) which yields a class label (crisp or soft) for $\mathbf{x}$.

Clearly these three definitions are not equivalent. A probabilistic classifier may not use fuzzy sets, hence do not qualify according to 1. A fuzzy rule based classifier uses fuzzy sets but they may not conform to the definition 2. Further, there may be fuzzy classifiers other than rule based ones, e.g., fuzzy k-NN classifiers [158]. However, the definitions give us a rough idea of fuzzy classifiers. Fuzzy rule based classifiers are by far most popular of fuzzy classifiers. In Chapter 5 of the current thesis we develop a data-driven method for designing fuzzy rule based classifiers. A discussion on them is included in the same chapter. More detailed discussions can be found in [191] and [32].

Though the rule based classifiers occupy place of pride among fuzzy classifiers, there are several other approaches investigated by the researchers. Among early approaches Watada et al. [345] proposed a method for fuzzy linear discriminant analysis, where

elements of data set are represented as fuzzy numbers and fuzzy arithmetic is used for computation. However, the requirement for implementation of fuzzy arithmetic limited its scope of application to real life problems. In [159] fuzzy perceptron was proposed. It used data with fuzzy labels for training. Classifiers based on *fuzzy relations* are also studied by several researchers [276, 282]. However, these methods are now more of theoretical value rather than useful [191]. Among others, fuzzy k-nearest neighbors classifiers and its variants as well as prototype based classifiers with fuzzy labels (sometimes also known as fuzzy prototype based classifiers) are investigated by many researchers [191].

Fuzzy k-NN classifiers [158], in their simplest form are straightforward extension of their crisp counterpart, provided the reference data set has soft/fuzzy label, i.e., the label for a data point $\mathbf{x}_i$ is $\mathbf{u}(\mathbf{x}_i) \in [0,1]^c$. Thus for a data point $\mathbf{x}'$, if $X^{(\mathbf{x}')} = \left\{\mathbf{x}^{(1)}(\mathbf{x}'), \ldots, \mathbf{x}^{(k)}(\mathbf{x}')\right\}$ is the set of k nearest neighbors, then

$$\mu(\mathbf{x}') = \frac{1}{k}\sum_{i=1}^{k} \mathbf{u}(\mathbf{x}^{(i)}(\mathbf{x}')). \tag{1.1}$$

The simplest classification rule uses the maximum membership value as follows:

$$\text{Assign } \mathbf{x}' \text{ to class } \omega_i \text{ if } \mu_i(\mathbf{x}') = \max_{1\le j\le c}\left\{\mu_j(\mathbf{x}')\right\}.$$

Evidently this is equivalent to crisp k-NN rule when the reference vectors are crisply labelled (i.e., $\mathbf{u}(\mathbf{x}') \in \{0,1\}^c$) and the classification is done by the maximum membership value. Though in (1.1) summation is used, other fuzzy aggregation operators can also be used [191]; however, there is no theoretical guideline available for making the choice. Given a data point $\mathbf{x}'$, many authors considered the distance of the neighbors for calculating a soft class label $\mu_i(\mathbf{x}')$ for class i. For example, Bereau and Dubuisson [28] propose

$$\mu_i(\mathbf{x}') = \max_{1\le j\le k}\left\{u_i(\mathbf{x}^{(j)}(\mathbf{x}')).\exp\left[-\lambda\left(\frac{d_j}{d_i^m}\right)\right]\right\},$$

where d_j is the distance between $\mathbf{x}'$ and its j-th nearest neighbor $\mathbf{x}^{(j)}(\mathbf{x}')$, d_i^m is the average distance for class ω_i (in [28] it is said to be the average distance between the data points in the reference set X with high membership in ω_i) and $\lambda \in [0,1]$ is a parameter controlling the decrease of the exponent term. Since these methods work on the data with soft labels, if the data points have crisp labels, then a suitable pre-labelling scheme for converting the labels into soft ones is used before applying them. Various such schemes are discussed in [191]. In [360] Yang and Chen proposed a generalized framework incorporating various fuzzy k-NN models. They also showed some theoretical results about their asymptotic behavior.

Fuzzy prototype based classifiers have been studied by many researchers. Here the prototypes have fuzzy label and unlike their crisp counterpart, the number of prototypes

may be less than the number of classes. A common framework, known as Generalized Nearest Prototype Classifier (GNPC) is proposed by Kuncheva and Bezdek [193]. According to the framework:

> *The Generalized Nearest Prototype Classifier (GNPC) is the 5-tuple* $(V, \mathbf{L}_V, s, \mathcal{T}, \mathcal{S})$ *where*
>
> - $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_v\}, \mathbf{v}_i \in \Re^p$ *is the set of prototypes.*
> - $\mathbf{L}_V \in [0,1]^{c \times v}$ *is the label matrix for the prototypes in* c *classes.*
> - $s(\Delta(\mathbf{x}, \mathbf{v}_i); \theta)$ *is a norm-induced similarity function, where* $\Delta(.)$ *is a norm metric in* $\Re^p$ *and* θ *is a set of parameters of* s*.*
> - $\mathcal{T}$ *is a t-norm defined over fuzzy sets and* $\mathcal{S}$ *is an aggregation operator.*

For an input $\mathbf{x} \in \Re^p$, the similarity vector $\mathcal{I} = [s_1, \ldots, s_v]^T$ is calculated. Then the label vector $\mu(\mathbf{x})$ is computed by the composition

$$\mu(\mathbf{x}) = \mathbf{L}_V \circ \mathcal{I},$$

where the composition operator $\circ$ consists of $\mathcal{T}$ and $\mathcal{S}$. The classification is made as

$$\text{Assign } \mathbf{x} \text{ to class } \omega_i \text{ if } \mu_i(\mathbf{x}) = \max_{1 \le j \le c} \{\mu_j(\mathbf{x})\}.$$

Depending on the choice of the elements of the 5-tuple, GNPC model encompasses various families of prototype based classifier. A detailed discussion on various fuzzy classifiers can be found in [191] and [32]. Other fuzzy set theoretic methods such as weighted fuzzy pattern matching [87], fuzzy integral based methods [157], fuzzy decision tree method [48] etc. are also used for designing classifiers.

Clustering

The clustering techniques discussed so far are called hard or crisp clustering since they assign each data point to only one cluster. In fuzzy clustering, an object can belong to all of the clusters with a certain degree of membership. This allows us discover more detailed relationship between an object and the set of clusters. The *fuzzy c-means* (FCM) [31] is by far most popular fuzzy clustering algorithm. Given a set of data points $X = \{\mathbf{x}_j | \mathbf{x}_j \in \Re^p, j = 1, \ldots, N\}$ FCM attempts to find a fuzzy c-partition that minimizes the cost function

$$J(U, V) = \sum_{i=1}^{c} \sum_{j=1}^{N} (u_{i,j})^m D_{ij}$$

where $U = [u_{i,j}]_{c \times N}$ is the fuzzy partition matrix, $u_{i,j} \in [0,1]$ is the membership of $\mathbf{x}_j$ to i-th cluster, $V = [\mathbf{v}_1, \ldots, \mathbf{v}_c]$ is the cluster prototype matrix, $m \in [1, \infty)$ is the fuzzification parameter and $D_{ij} = D(\mathbf{x}_j, \mathbf{v}_i)$ is an inner product induced distance measure between $\mathbf{x}_j$ and $\mathbf{v}_i$. In the following we provide a sketch of basic FCM algorithm where Euclidean distance is used as the distance measure:

1. Select values for m, c and a small number ε. Initialize V randomly. Set $t = 0$.

2. Calculate (at $t = 0$) or update (for $t > 0$) the membership matrix U as follows:

$$u_{ij}^{(t+1)} = 1 \Bigg/ \left(\sum_{l=1}^{c} \left(D_{lj} / D_{ij} \right)^{1/(1-m)} \right) \text{ for } i = 1, \ldots, c \text{ and } j = 1, \ldots, N.$$

3. Update the prototype matrix V as follows:

$$\mathbf{v}_i^{(t+1)} = \left(\sum_{j=1}^{N} \left(u_{ij}^{(t+1)} \right)^m \mathbf{x}_j \right) \Bigg/ \left(\sum_{j=1}^{N} \left(u_{ij}^{(t+1)} \right)^m \right) \text{ for } i = 1, \ldots, c.$$

4. Repeat steps 2 and 3 until $\left\| V^{(t+1)} - V^{(t)} \right\| < \varepsilon$.

There are numerous variants of FCM reported in the literature. They vary in the way cost function is formulated, the distance measure used, the way fuzziness is controlled and the optimization technique adopted. An account of many of these variants as well as other fuzzy clustering techniques (e.g. mountain method [355], possibilistic c-means (PCM) [187], conditional FCM [277, 283] etc.) can be found in [19, 31, 32, 353, 278].

Neural networks and fuzzy set theory, each has its own advantages and disadvantages. So, many attempts have been made in PR research to design systems exploiting the benefits of both. They employ both neural network as well as fuzzy set theoretic techniques [216, 215, 218, 214, 270, 152, 280]. Such combined techniques are often called *neural-fuzzy* or *neuro-fuzzy* techniques. Apart from the huge volume of research literature, there are also a large number of books [31, 32, 164, 191, 216, 271, 270] covering various aspects of fuzzy set theoretic pattern recognition.

In the current thesis we develop methods for designing and fine tuning fuzzy rule based classifiers. We also investigate the issue of context-sensitive tuning of the rule base. Further we develop schemes using the outputs of the fuzzy rule base as possibilistic label vectors to aggregate contextual information for more robust classification.

1.3.4 Evidence Theoretic Approach to Pattern Recognition

Dempster-Shafer theory of evidence [304] deals with a particular form of uncertainty, different from fuzziness. We shall describe it with the help of an example. Consider a

patient visits a doctor in a hospital. The doctor conducts several examinations (physical and clinical) and thinks about a possible set of diseases $\{D_1, D_2, D_3\}$, one of which the patient might be suffering from. Depending on the results of examinations the doctor finds different amount of evidences supporting the diagnosis that the disease may be identified as D_1 or D_2 or D_3, or any one of D_1 and D_2 and so on, i.e., there could be evidence in support of any subset of the set of diseases. The quantified amount of such evidences in support of different subsets of the set of diseases are called a *body of evidence*. Note that, here there is no fuzziness involved with the fact that the patient is suffering from a disease. The uncertainty arises solely from the available evidence that expresses the degree of belongingness of the ailment to different crisp subsets of diseases. This type of uncertainty can be handled using the evidence theory. Further, consider that the doctor has also referred the patient to one of his colleagues who has also come up with a body of evidence over the same set of diseases. Naturally the two bodies of evidence are not likely to be identical. Now the doctors can meet and discuss their findings and come up with a diagnosis, which is essentially making a final diagnosis about the ailment and identify the disease(s). This requires a way of aggregating the available evidences from multiple sources (two doctors here). Such situations where information is available from multiple sources in form of bodies of evidence, the Dempster-Shafer theory of evidence can be applied to handle the information and their aggregation in a mathematically consistent manner.

The Dempster-Shafer theory is used in designing pattern classification systems for a long time. However, it was mainly used for aggregation of information from different sources, for example, classification of multispectral [211] or multisource [112] images. Using DS theory for designing general purpose classifiers is a relatively recent approach pioneered by Denœux and his coworkers [76, 369]. In [76] Denœux proposed an evidence theoretic k-NN rule, where each of the k nearest neighbors becomes a source of evidence corresponding to the hypothesis that the test point belongs to the class of the neighbor. The k bodies of evidence are combined using Dempster's rule of combination to make the final classification. Here both the class of the neighbor as well as its distance from the test data is considered for basic probability assignment (BPA) of the corresponding body of evidence. In [369] Zouhal and Denœux studied the evidence theoretic k-NN rule with respect to various parameter optimization. Later Denœux [77] also proposed a feed-forward neural network classifier employing evidence theory based learning. Evidence theoretic rank nearest neighbor rules are also proposed in [265].

Very recently application of evidence theory for clustering of relational data also proposed by Denœux and Masson [78]. The proposed method, given a matrix of dissimilarities between n objects, assigns a belief to each object in such a way that the degree of conflict between the BPAs given to any two objects reflects their dissimilarity. Then a

credal partition is computed to identify the clusters. In another paper [242] the authors proposed evidence theoretic clustering technique for interval-valued dissimilarity data.

In the current thesis we do not aim to develop general purpose evidence theoretic pattern recognition techniques. Here we develop several schemes for using contextual information generated by fuzzy rule base as supplementary evidence. Evidence theory is then used for aggregating the evidence to achieve a more robust decision making. However, one of the schemes developed here follows the spirit of evidence theoretic k-NN method.

1.4 Scope of the Thesis

In this thesis we propose some novel Self-organizing Map (SOM) based methods for performing pattern recognition tasks as well as present results of some empirical studies on properties of SOM. The empirical studies reported here include studies on topology preservation ability of SOM under non-ideal internode connections, study of topology preservation property of some variants of SOM with simplified lateral feedback functions and study of topology preservation as well as vector quantization property of some variants of SOMs with tree-structured neighborhood functions.

The pattern recognition tasks we considered here are of two types: (1) pattern classification and (2) vector quantization. For pattern classification we have developed a dynamic prototype generation algorithm, a method of extracting and fine-tuning fuzzy rules from the data and several evidence theoretic schemes for improved decision making with fuzzy rule based classifiers. In our work on vector quantization we have developed a scheme for designing vector quantizers for image compression using SOM and a surface fitting method that results in better psychovisual qualities of the reproduced images. We have also devised a scheme for fast codebook searching in a SOM-based vector quantizer that enables us to use large codebooks with high dimensional codevectors while avoiding huge computational load associated with such conditions.

The effectiveness of the proposed methods is demonstrated using several synthetic as well as real life data sets. Especially the fuzzy rule based classifiers are used for the complex classification problem of landcover analysis from multispectral satellite images. The proposed methods for VQ design are tested with several benchmark images. The results obtained with all the proposed methods are compared with benchmark results available in the literature. The organization of the thesis is summarized bellow under different chapter headings.

1.4.1 The Self-Organizing Map (SOM)

Since all the works reported in the current thesis utilize the SOM algorithm directly or indirectly, in Chapter 2 we present an overview of the Self-organizing Map. We describe the SOM architecture and algorithm, followed by a discussion on its properties. Here we also provide a brief review of several variants of SOM and research works both on theoretical aspects of SOM as well as its applications in solving various pattern recognition problems.

1.4.2 SOM: robustness, simplification and topology preservation [199, 202]

In Chapter 3 first we introduce a *rank correlation coefficient* based new quantitative index for measuring topology preservation in SOM. We report a comparative study of this index with two other indexes. Next we study the robustness of SOM algorithm in preserving topology in partial absence of lateral feedback connections between the SOM nodes. Such absence or sparsity of links causes irregularity in the neighborhood update. We have considered two types of sparsity, *systematic* and *random*. The experiments show that SOM can withstand systematic sparsity to a large extent, while the topology preservation is very sensitive to random sparsity. The findings may lead to economic hardware design for SOM where systematic sparsity of connection can be allowed.

Our other empirical studies in Chapter 3 includes the study of two classes of variants of SOM. The first variety involves SOMs with simplified lateral feedback functions, specifically, Gaussian neighborhood function without explicit boundary, quadratic neighborhood function and linear neighborhood function. The study reveals their almost equivalent performance in preserving topology. These neighborhood functions are easier to implement in hardware than the one with explicit neighborhood boundary. The other class of variants consists of SOMs with tree-structured neighborhood function. It was found in [147] that SOMs with Minimal Spanning Tree (MST) neighborhood function are better than standard SOM with a square or hexagonal neighborhood function in extracting prototypes from complex and non-linear data. Here we proposed a new SOM with MST neighborhood where the cost of computation of MST is less as well as topology preservation is better. We also proposed a SOM with an arbitrary tree neighborhood that performs equally well at prototype extraction but avoids the cost of computing MST repeatedly altogether. We also studied the ability of skeletonization of shapes by these SOMs.

1.4.3 Extraction of prototypes and designing classifiers [203, 204, 266]

In Chapter 4 we propose a novel SOM-based method for extracting prototypes and designing prototype based classifiers. First we develop a SOM-based algorithm for dynamic prototype generation using a mixture of unsupervised and supervised training. The algorithm is data-driven and generates an adequate number of prototypes without the need of user specifying the same. The prototypes are used successfully to design nearest prototype (NP) classifiers, called "1-Nearest Multiple Prototype (1-NMP)" classifiers. Next we proposed a method of associating a hyperspherical zone of influence with each prototype and fine-tuning them to design "1-Most Similar Prototype (1-MSP)" classifiers with better performance.

1.4.4 Extraction of fuzzy rules for classification [206, 267]

Fuzzy rule based classifiers are popular for their interpretability, ability of handling uncertainty, and ability to deal with large variation in variances of features. These properties make them good choices for classifying complex data sets. In Chapter 5 we propose a comprehensive scheme for designing fuzzy rule based classifiers. The proposed methodology attempts to design fuzzy rule based classifiers with a small but adequate number of rules for computational efficiency. The scheme uses the prototypes generated using the algorithm proposed in Chapter 4. The prototypes are then converted into an initial set of fuzzy rules. A gradient-descent based algorithm is then devised for fine-tuning of the rule base. We have studied two variants of rule base, using (1) product and (2) softmin as conjunction operator. The classifiers are designed and tested for the complex task of landcover classification using multispectral satellite images.

The softmin operator is a special case of a soft-match operator, that can implement a whole family of operators including average, max etc. depending on the value of a parameter. We also investigated the possibility of *context-sensitive* reasoning. Here each rule is allowed to have its own conjunction operator based on the context in the feature space it is representing. We developed an algorithm for tuning the parameter of the soft-match operator and studied its effectiveness. The proposed methods are compared with several existing methods and found to produce superior results.

1.4.5 Evidence theory-based decision making for fuzzy rule based classifiers [201, 207]

The output of a fuzzy rule based classifier can give more information than just a single class label. The rule base provides confidence values with respect to the membership of an input data point to various classes. Often it may happen that for multiple classes the rule base produces high and close values of confidence, thereby indicating a high level of uncertainty. In such a case, one should try to use additional information, if available to resolve the uncertainty. Such an opportunity exists when each data point can be identified in same spatial or temporal context of some other data points and high correlation is expected among the data points in the same context. For such data types it is possible to devise more robust decision making schemes by aggregating contextual information. Multispectral satellite image data provide such correlated spatial contexts.

In Chapter 6 four schemes are proposed for decision making using contextual information provided by the fuzzy rule base proposed in Chapter 5. Of the four schemes developed, one is a simple aggregation scheme similar to fuzzy k-NN rule, while others use Dempster-Shafer theory of evidence for aggregation of contextual information. The fuzzy rule base is used to generate a possibilistic label vector for each pixel. For final classification of a pixel, the possibilistic label vectors associated with the pixel of interest as well as with its neighboring pixels are used. In the simple aggregation scheme the possibilistic label vectors are averaged and the pixel is assigned to the class with the highest average confidence value. In other schemes the information is converted into evidences in support of various hypotheses through *basic probability assignments*(BPAs) and then the BPAs (evidences) are combined to identify the hypothesis with the highest support. The schemes differ in the way bodies of evidence are formed and the basic probability assignment is done. For example, one of the schemes focuses only on singletons (Bayesian belief) while another one focuses on sets with one and two elements. The aggregation schemes are useful for classifying 'mixed pixels' and pixels at class boundaries because for those pixels the uncertainty is higher in the information captured. Efficacy of the aggregation schemes is studied using two multispectral satellite images and the classification performance is found to improve over simple fuzzy rule based classifiers. It is also observed that the suitability of a particular scheme depends, to some extent, on the nature of spatial distribution of the landcover classes.

1.4.6 Designing vector quantizer with SOM and surface fitting for better psychovisual quality [205]

In Chapter 7 we propose a method of designing vector quantizers for image compression. The SOM algorithm is used to generate the initial codebook. Due to the density matching property and topology preservation property of SOM the codebook is expected to result in less 'granular error' and 'overload error'. The SOM is trained with mean-removed vectors obtained from a set of training images. The performance of the scheme is studied on a different set of images. Further, the use of mean-removed vectors makes it possible to quantize large image blocks by exploiting the statistical redundancy in the data more effectively and thus increases the efficiency of the quantization process. For images compressed with VQ the reconstructed images suffer from blockyness, i.e., the boundaries of the image blocks become pronounced, leading to degradation of psychovisual quality.

We propose here a unique surface fitting technique for smoothing the codevectors that leads to refinement of the initial SOM-generated codebook. This results in a substantial reduction of the blocking effect in the reconstructed images. Two quantitative indexes are also developed for measuring the psychovisual quality of the reconstructed images with respect to blockyness. Since the scheme uses mean-removed vector, one needs to store/transmit the block averages along with the indexes for reconstruction of the images. To improve the compression ratio further, we use Huffman coding of the indexes and the difference coded block averages.

1.4.7 Fast codebook searching in a SOM-based vector quantizer [200]

While designing a vector quantizer, apart from the algorithm used for finding good codevectors, one is faced with two design issues: the dimension of the vectors and the size of the codebook. A large vector enables the quantizer to exploit the statistical redundancy existing in the data to a greater degree. However, this may increase the reproduction error unless codebook size is also large. The bigger is the codebook size, the finer is the representation of the input space. While encoding a vector, the encoder needs to search the codebook. An exhaustive search of a large codebook in high dimension requires a high computational overhead. This may result in a performance bottleneck. To circumvent this problem researchers have proposed many methods. Usually some constraints on the structure of the codebook are imposed, which are exploited during the search. In Chapter 8 we propose a method for fast codebook searching in SOM-generated codebooks. The method performs a non-exhaustive search of the codebook to find a good match for a input vector. Our scheme combines two SOM-based codebook search

strategies, each of which is an independent search method, but somewhat complementary in nature.

In SOM due to the neighborhood update strategy employed during the training, some constraint is implicitly imposed on the organization of the nodes which leads to topology preservation. Hence, if two successive vectors are well-correlated, a good code for the later vector is likely to be found among the neighbors of the code for the former in the SOM lattice. This possibility is exploited by the method developed in Chapter 8. The first strategy searches for a good match among the weight vectors of the neighbors within a small window over the lattice centered at the node corresponding to the code of the previous vector. If a match exceeding a quality threshold is found, then that is accepted; otherwise, an exhaustive search is conducted. The second strategy first uses a smaller SOM which is trained with the weights of the basic codebook. Thus the codes of basic codebook are partitioned according to their closeness of the nodes in the smaller SOM. To search the code for a vector, the smaller SOM is searched first for the best matching node, then the best match in the corresponding partition is accepted. In the combined method the search is conducted in the same way as the first strategy, i.e, with a window. But if a good match is not found within the search window, then the search is performed as in the second strategy.

1.4.8 Conclusion and future works

In Chapter 9 we summarize and conclude the contributions in the thesis. Here we also explore various possibilities of extension and/or modification of the work reported in the thesis.

Chapter 2

The Self-Organizing Map (SOM)

2.1 Introduction

The neural network models vary widely in connection topologies, computational element's capabilities and learning algorithms. In [174] Teuvo Kohonen provided a categorization of neural network models as follows:

1. **Signal-Transfer networks**, which map the input signals into output signals. The transformation is usually learned from the data in a supervised manner. Multilayer perceptrons [297], radial-basis-function networks, madaline etc. are representatives of this category. They are also known as feedforward networks.

2. **State-Transfer networks** or feedback networks, which define the initial activations of the processing elements from the input signals, and go through a sequence of state transitions, at the end settling to a stable state and producing the output. Here the connection strengths are usually pre-assigned (not trained) to the network. Examples of this category include Hopfield networks [130], Boltzmann machines [124], bidirectional associative memory (BAM) etc.

3. **Competitive, self-organizing networks**, where the processing units (neurons) compete against each other for activation. They are characterized by existence of lateral interaction among neurons that enables them to compete and activate selectively. The training is done in an unsupervised manner and the network adapts the connection strengths based on the properties of the input data.

The Self-organizing Map (SOM) (also known as Self-organizing Feature Map (SOFM), Kohonen's Self-organizing Map (KSOM)) proposed by Kohonen in early 1980s belongs to the third category and is one of the most widely used neural network models. SOM is a two-layer network with a fan-out input layer of linear nodes, connected to a competitive output layer where the processing units or "neurons" are arranged in a regular lattice structure. Each node in the input layer is connected to every neuron in the competitive layer through weighted connections. During the learning stage the neurons become tuned to different signal patterns so that when a signal is presented to the network only the neurons best representing the signal become active and the response is concentrated in a small neighborhood in the output lattice. During the training process the responses of the neurons become spatially ordered over the lattice, i.e., two similar input signals will activate same or nearby neurons on the SOM lattice. This leads to the distinguishing properties of SOM, namely, 1) topology preservation and 2) density matching. According to Kohonen [174],

> In the pure form, the SOM defines an "elastic net" of points (parameter,

reference, or codebook vectors) that are fitted to the input signal space to approximate its density *in an ordered fashion*.

The model of the SOM is inspired by the findings in physiology and neuroscience. For a long time the researchers in these fields studying the organization of the brain, were aware of the existence of topographical organizations in different areas of the brain, especially in the cerebral cortex. Their studies on functional deficits and behavioral impairments induced by lesions, tumors, hemorrhages etc. have uncovered that different regions in the brain are responsible for different kind of activities. Later various modern methods such as positron emission tomography (PET) have also led to more detailed study unearthing numerous topographic maps in the brain. Popular examples include "tonotopic maps" in auditory cortex, where there are topologically ordered areas for processing different frequencies of auditory signals. In the visual cortex various topographically ordered computational maps exist to represent various features of visual input like angle of tilt of a line, movement of an object. Details of such studies can be found in [11, 165].

Originally Kohonen described the SOM algorithm as a system of coupled differential equations [166, 167]. The model was inspired by the true biological system. In the model, though acting parallelly, each neuron is functionally independent, i.e., there is no synchronizing clock or higher level coordination of the unit activities. The self-organization and topology preservation is effected due to various lateral feedback connections that correlate the activity of nearby units during the winner search stage. The system can be simulated in a serial computer, but is computationally intensive and for a full implementation a parallel computer with many nodes are needed. Thus, use of the original algorithm for practical applications was very difficult. Consequently, for practical applications a simplified algorithm [169, 176] was proposed by Kohonen that avoids mimicking the biological system too closely, but leads to functional appearance of the Self-organizing Maps. The algorithm can be efficiently implemented in a serial computer. Henceforth we shall be using the simplified model only.

2.2 The SOM architecture and algorithm

2.2.1 The SOM architecture

SOM is a two-layer network comprising of a fan-out input layer and a competitive output layer. The number of input nodes is the same as the number of dimensions p of the input data. Each node in the input layer is connected to all the nodes in the output layer by a weighted connection. Thus, with each output node a weight vector $\mathbf{w} \in \Re^p$ is associated. The output nodes are arranged in a regular q dimensional grid $V(\Re^q)$. The nodes occupy

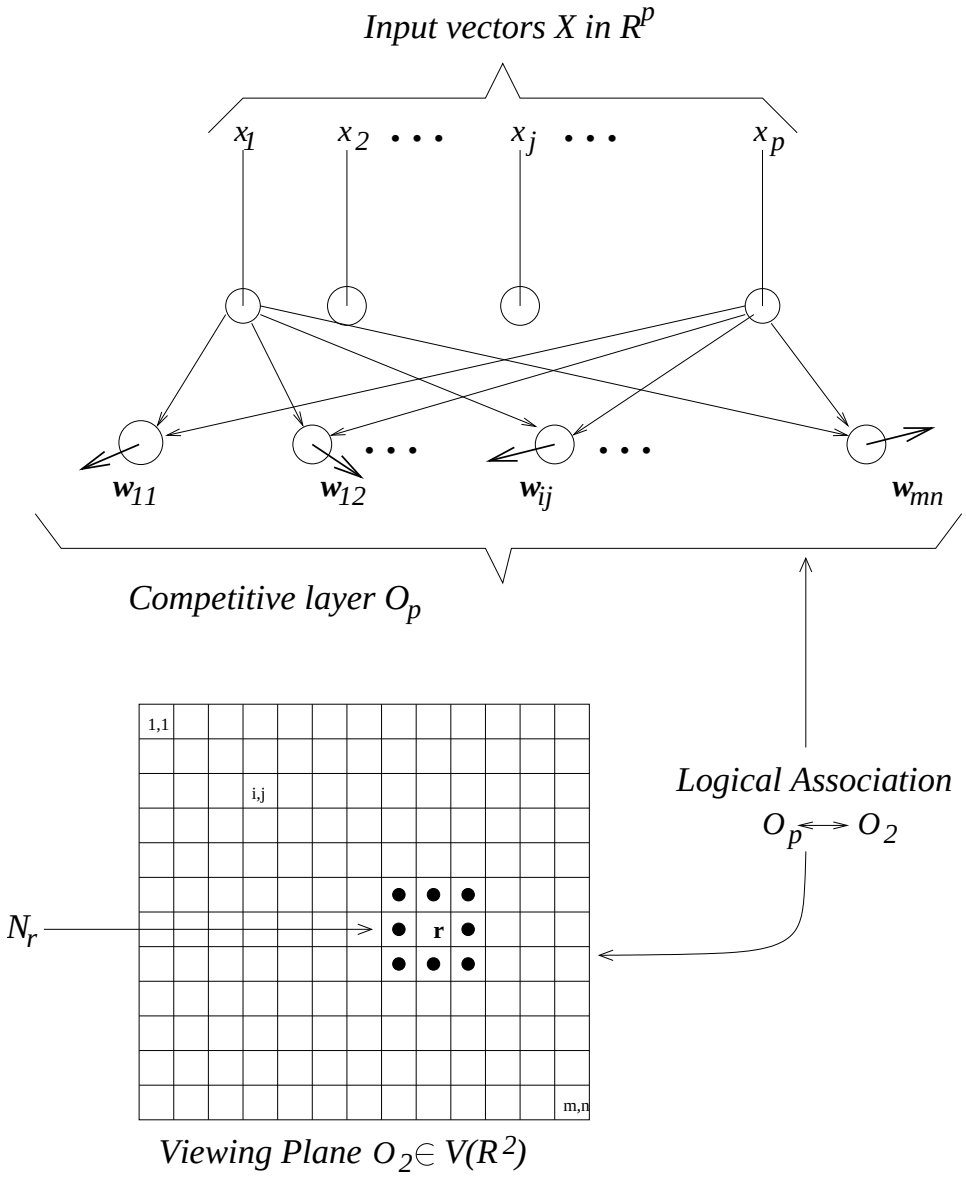

Figure 2.1: SOM architecture.

lattice positions of the rectangular (or sometimes hexagonal) grid. Figure 2.1 depicts the architecture of the SOM for $q = 2$. There also exists (not shown in the figure) lateral feedback connections among the output nodes through which the winner neuron can inhibit other neurons from activating and induce update to its neighbors. Since SOM is often treated as an algorithmic display transformation from $\Re^p$ to $V(\Re^q)$ [34], it is denoted as $A^D_{SOM} : \Re^p \rightarrow V(\Re^q)$.

In principle, SOM can be used for transforming the data $X = \{\mathbf{x}_i \in \Re^p \mid i = 1, 2, \ldots N\}$ onto a display lattice $\Re^q$ for any q. However, in practice, one seldom uses $q > 3$ and in most cases choices are restricted to $q = 1$ and 2. In the current thesis we shall be concentrating on SOMs with one dimensional and two dimensional output spaces with m and $m \times n$ nodes respectively.

Input vectors $\mathbf{x} \in \Re^p$ received by the input nodes are distributed to each of the $m \times n$ output nodes in the competitive output layer. Each node in the output layer has a weight vector $\mathbf{w}_{ij}$ attached to it as shown in Fig. 2.1. Let $O_p = \{\mathbf{w}_{ij}\} \subset \Re^p$ denote the set of $m \times n$ weight vectors. Thus O_p is (logically) connected to a display grid $O_2 \subset V(\Re^2)$. (i, j) in the index set $\{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ is the logical address of the cell. Thus, the k-th component of the weight vector $\mathbf{w}_{ij}$, w_{ij_k} is the weight of the connection between output node (i, j) and the k-th input node that receives the k-th component of the input vector $\mathbf{x}$. There is a one-to-one correspondence between the $m \times n$ p-vectors $\mathbf{w}_{ij}$ and the $m \times n$ cells $(\{i, j\})$,i.e., $O_p \longleftrightarrow O_2$. In the literature, the display cells are sometimes called nodes, or even neurons, in deference to possible biological analogues.

2.2.2 The SOM algorithm

SOM begins with a (usually) random initialization of the weight vectors $\{\mathbf{w}_{ij}\}$. For notational clarity we suppress the double subscripts. Now let $\mathbf{x} \in \Re^p$ enter the network and let t denote the current iteration number. Let $\mathbf{w}_{r,t-1}$ be the weight vector that best matches $\mathbf{x}$ in the sense of minimum Euclidian distance in $\Re^p$. $\mathbf{w}_{r,t-1}$ is called the *winner* in the t-th iteration. Note that, the winner can be decided using other distance functions also. Some of them will be discussed later. This vector has a (logical) "image" which is the cell in O_2 with subscript r. Next a topological (spatial) neighborhood $N_r(t)$ centered at r is defined in O_2, and its display cell neighbors are located. For example, as shown in Figure 2.1, a 3×3 window $N(r)$, centered at r corresponds to nine prototypes in $\Re^p$, associated with the cell r and its eight neighbors on the lattice. Finally, $\mathbf{w}_{r,t-1}$ and the other weight vectors $\mathbf{w}_{i,t-1}$ associated with cells within the spatial neighborhood $N_t(r)$ are updated using the rule

$$\mathbf{w}_{i,t} = \begin{cases} \mathbf{w}_{i,t-1} + h_{ri}(t)(\mathbf{x} - \mathbf{w}_{i,t-1}) & \text{if} i \in N_r \\ \mathbf{w}_{i,t-1} & \text{if} i \notin N_r \end{cases} \tag{2.1}$$

Here r is the index of the "winner" prototype found as:

$$r = \underbrace{arg\,\min}_{i}\{\|\mathbf{x} - \mathbf{w}_{i,t-1}\|\}, \tag{2.2}$$

where $\|*\|$ is the Euclidian norm on $\Re^p$. The function $h_{ri}(t)$ which expresses the strength of interaction between cells r and i in O_2, usually decreases with t, and for a fixed t it decreases as the distance (in O_2) from cell r to cell i increases. $h_{ri}(t)$ is usually expressed as the product of a learning parameter α_t and a lateral feedback function $g_t(dist(r, i))$. A common choice for g_t is $g_t(dist(r, i)) = \exp^{-dist^2(r,i)/\sigma_t^2}$. α_t and σ_t both decrease with time t. The topological neighborhood $N_t(r)$ also decreases with time. This scheme, when

repeated long enough, usually preserves spatial order in the sense that weight vectors which are metrically close in $\Re^p$ generally have, at termination of the learning procedure, visually close images in the viewing plane. A schematic description of the SOM algorithm is provided in Figure 2.2. Figure 2.3(b) shows 1-D SOM trained with data resembling noisy sine curve (2.3(a)), Figures 2.4 and 2.5 depict 1-D and 2-D SOMs respectively trained with 2-D data uniformly distributed over a square.

In the above we assumed the data as real-valued vectors and used Euclidean distance as the metric. However, the applicability of the algorithm is not restricted to this choice. It can deal with other types vectorial data such as binary vectors, also the distance metric chosen can vary according to the requirement of the problem. One can use city-block distance, Minkowsky distance, Tanimoto measure etc [174]. However, the winner matching and updating laws should be compatible with the metric used. For example, if "dot-product" similarity is used as the metric, the winner is found as,

$$r = \underbrace{arg\max}_{i}\{\mathbf{x} \bullet \mathbf{w}_{i,t-1}\},$$

and the update equation becomes

$$\mathbf{w}_{i,t} = \begin{cases} \frac{\mathbf{w}_{i,t-1}+h_{ri}(t)\mathbf{x}}{\|\mathbf{w}_{i,t-1}+h_{ri}(t)\mathbf{x}\|} & \text{if} i \in N_r \\ \mathbf{w}_{i,t-1} & \text{if} i \notin N_r \end{cases}$$

In neural network literature it is often suggested that the input data be normalized for better result [124]. In case of SOM, normalization is not necessary in principle [174], though it may improve numerical accuracy to some extent.

2.2.3 The generalized SOM algorithm

The above SOM algorithm is often termed as the "original incremental SOM algorithm" [174]. Here the input data are restricted to numerical vectors, for which suitable distance metrics can be used. A generalization of the algorithm for items that may not be vectorial was proposed by Kohonen [172] in 1996. In such a case the incremental learning rule (2.1) is no more applicable. Instead a batch computation procedure is adopted resulting in so called "batch map" algorithm. Here we provide a brief sketch of the generalized algorithm.

Consider the network as an ordered, one or two-dimensional array of nodes, each node having a general model m_i associated with it (here we use the notation m_i instead of $\mathbf{w}_i$ due to possible nonvectorial nature of the models and data). The initial values of the m_i may be selected at random, preferably from the domain of the input samples. Next

Algorithm $A^d_{SOM}(Kohonen)$:

Begin

Input X /** unlabelled data set $X = \{\mathbf{x}_i \in \Re^p : i = 1, 2, \ldots, N\}$ **/

Input m, n /** the display grid size a rectangle $m \times n$ is assumed **/

Input $maxstep$ /** maximum number of updating steps **/

Input N_0 /** initial neighborhood size **/

Input α_0 /** the initial step size (learning coefficient) **/

Input σ_0 and σ_f /** parameters to control effective step size **/

/** Learning phase **/

Randomly generate initial weight vectors

$\{\mathbf{w}_{ij}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n, \mathbf{w}_{ij} \in \Re^p\}$

$t \leftarrow 0$

While$(t < maxstep)$

Select randomly $\mathbf{x}(t)$ from X;

find $r = \underbrace{arg\ min}_{i}\{\|\mathbf{x}(t) - \mathbf{w}_i(t)\|\}$

/** r and i stands for two-dimensional indices that uniquely identify a weight vector in O_p **/

$\mathbf{w}_i(t+1) \leftarrow \mathbf{w}_i(t) + \alpha_t g_t(dist(r,i))[\mathbf{x}(t) - \mathbf{w}_i(t)]\ \forall i \in N_t(r)$

$\mathbf{w}_i(t+1) \leftarrow \mathbf{w}_i(t)\ \forall i \notin N_t(r)$

/** $dist(i,j)$ is the Euclidean distance between the centers of nodes r and i on the display lattice,$g_t(d)$ is the lateral feedback function, usually $g_t(d) = e^{-d^2/\sigma_t^2}$ **/

$t \leftarrow t + 1$

$\alpha_t \leftarrow \alpha_0(1 - t/maxstep)$

$N_t \leftarrow N_0 - t(N_0 - 1)/maxstep$

$\sigma_t \leftarrow \sigma_0 - t(\sigma_0 - \sigma_f)/maxstep$

/** there are other ways to readjust α_t, N_t and σ_t, and many choices for g_t **/

End While

/** Display phase **/

For each $\mathbf{x} \in X$ find

$r = \underbrace{arg\ min}_{i}\{\|\mathbf{x} - \mathbf{w}_i\|\}$,and mark the associated cell r in O_2.

End.

Figure 2.2: Kohonen's Self-organizing Map algorithm.

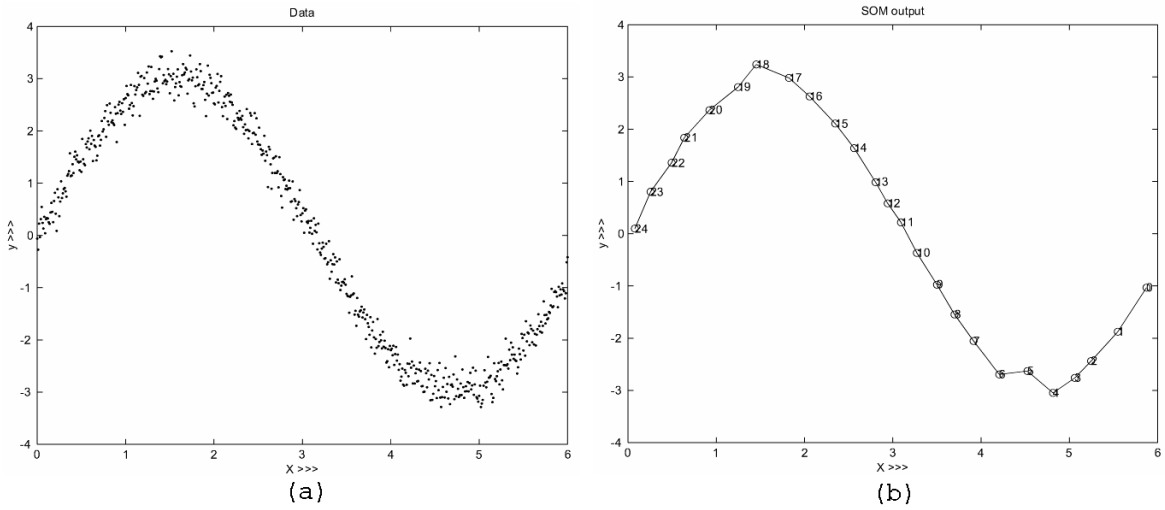

Figure 2.3: 1-D SOM trained with 2-D data distribution of a noisy sine curve.

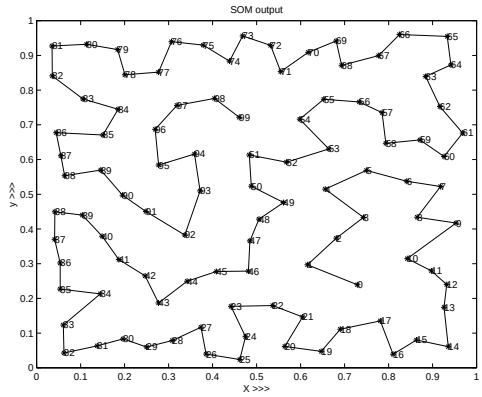

Figure 2.4: 1-D SOM trained with 2-D data uniformly distributed over a square.

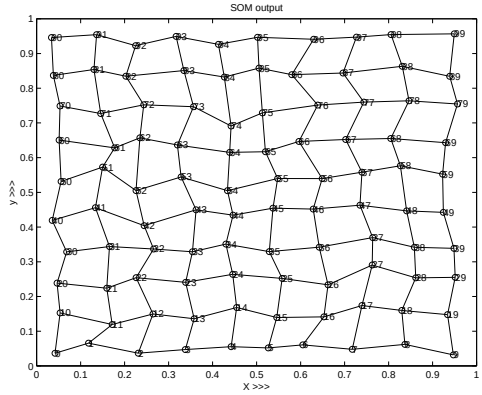

Figure 2.5: 2-D SOM trained with 2-D data uniformly distributed over a square.

consider a list of input samples $x(t)$, t being an integer valued index. In this scheme the samples $x(t)$ and models m_i may be vectors, strings of symbols or some more general item. It will suffice for the algorithm that a distance measure $d(x, m_i)$ is defined over all occurring x and a sufficiently large set of models m_i. The training algorithm is as follows:

For the input sample $x(t)$ the best-matching node $r(t)$ is found as

$$r(t) = \underbrace{\arg\min}_{i}\{d(x(t), m_i)\}. \tag{2.3}$$

Now $x(t)$ is added to a sublist S_r of input samples maintained for the winner node $r(t)$. The process is continued for all input samples and they are distributed into sublists under respective best-matching models. Now for the update of model m_i, the union of the sublists of the models within the corresponding neighborhood set N_i is selected. Then the "middlemost" sample $\overline{x}_i$, i.e., the sample with smallest sum of distance from all the samples $x \in \bigcup_{j \in N_i} S_j$ is computed. $\overline{x}_i$ is called the *generalized median* of the union of sublists. If $\overline{x}_i$ is restricted to the set of samples $x(t)$, it is called the *generalized set median*. Note that, for Euclidean vectors the generalized median is simply the *arithmetic mean*. The models m_is are updated by replacing their values with respective $\overline{x}_i$s. The procedure is carried out through several iterations after which the models approximate the input samples in an orderly fashion.

The convergence of the algorithm, at least to a unique equilibrium, is not yet mathematically proven [174]. In practice, convergence means the sublists do not change with iterations when repeated for sufficiently large number of iterations. Convergence proof of the algorithm under slightly modified conditions for Euclidean space can be found in [53].

Kohonen and Sumervuo used the generalized algorithm for building SOM of protein sequences [179, 180]. In a recent work Günter and Bunke [115] proposed a further generalization of SOM for items represented as graphs, which includes strings and trees as special cases. Thus, it extends the utility of SOM to structural pattern recognition also.

2.3 Properties of SOM

The SOM induces an algorithmic transformation from the input space $\Re^p$ onto the output/viewing plane $V(\Re^q)$ that preserves the *metric-topological* relationships and *distributional density* properties of the feature vectors/signals X in $\Re^p$ [174]. These properties

are popularly known as the *topology preservation* property and *density matching* property of SOM. Both of these properties are results of updating neighborhood neurons during the training phase and distinguishes the SOM from other competitive learning networks. Most of the applications based on SOM depend critically on these properties (explicitly or implicitly) for their success.

2.3.1 Topology preservation property of SOM

A map, be it natural or artificial [165, 174], projects a pattern in an input space onto a position in an output space, thus encoding the information as the location of an activated node in the output space. Further, an essential property of these maps is the preservation of neighborhood relations. This dictates the requirement that nearby vectors in the input space are mapped onto the neighboring locations in the output space. This is known as the "topology preservation property" of the maps. Apart from SOM, there exists a number of neural network models for adaptively forming topology preserving maps [90, 320, 341, 348]. However, Kohonen's SOM is by far most widely studied and applied one.

The topology preservation property is a direct consequence of the update equation (2.1), that forces the weight vector $\mathbf{w}_r$ of the winning neuron $r(\mathbf{x})$ and those of its neighbors toward the input vector $\mathbf{x}$. Thus the map can be visualized as an *elastic/virtual* net with topology of a one or two-dimensional lattice (when the output nodes are arranged in a 1-D or 2-D grid) in the output space and weight vectors of the nodes as coordinates in the input space $\Re^p$. During the training process the elastic net folds onto the "cloud" formed by input data.

The degree of perfection to which a SOM preserves the topology is dictated by various factors. The two most important factors are the complexity of the distribution of input patterns and dimensionality mismatch between the output space and input space. The learning parameters sometimes also play important roles. There could also be cases of various kind of distortions such as twisting of the map for certain initialization. In case of one or two dimensional maps with one or two dimensional (and to some extent three dimensional) input patterns, it is possible to subjectively assess the quality of topology preservation by visual inspection (e.g. Figures 2.3, 2.4 and 2.5). But for higher dimension it is very difficult to assess the quality of the map formed. Many researchers tried to quantitatively measure the topology preservation property [22, 73, 80, 79, 241, 294, 317, 339, 370]. A review of different approaches to measure the topographic order preservation in SOM can be found in [285]. Though the concept of topology preservation is intuitively well understood, it is an extremely difficult task to arrive at an operational definition and derive a quantitative measure. Different authors

define it in different ways and derive the measure accordingly. So they might reflect different facets of the problem. The *topographic product* introduced by Bauer and Pawelzik [22] is one of the popular measures. However, this measure reflects more strongly the dimensional mismatch between the intrinsic feature space and the output space and often it is difficult to interpret when the lack of topology preservation is due to other causes. Su et al. [317] have recently proposed a measure of topology violation based on an intuitive understanding of the property and it produces a fairly good estimate irrespective of the cause. Here we give brief descriptions of these two measures. In the next chapter we shall introduce a new measure of topology preservation based on Kendall's *rank correlation coefficient* [160]. We shall be using these measures in the next chapter for studying topology preservation property of SOM under various modified conditions.

Topographic Product: a measure of topology preservation

The topographic product [22] is a measure of the preservation of neighborhood relations by SOM. For notational convenience we shall denote the output space R^q and input space (i.e., weight space) R^p as U and V respectively in the following discussion.

The Euclidian distance in U is denoted as

$$d^U(j, j^{'}) = \|j - j^{'}\| \tag{2.4}$$

and in V

$$d^V(\mathbf{w}_j, \mathbf{w}_{j^{'}}) = \|\mathbf{w}_j - \mathbf{w}_{j^{'}}\| \tag{2.5}$$

where $j, j^{'} \in \{1, 2, \ldots, m \times n\}$.

The notation of nearest neighbor indices is as follows:
Let $n_k^U(j)$ denote the k-th nearest neighbor of node j with the distance measured in output space, i.e.,

$$n_1^U(j) : d^U(j, n_1^U(j)) = \min_{j^{'} \in U \setminus \{j\}} d^U(j, j^{'})$$

$$n_2^U(j) : d^U(j, n_2^U(j)) = \min_{j^{'} \in U \setminus \{j, n_1^U(j)\}} d^U(j, j^{'})$$

$$\vdots$$

In the same way let $n_k^V(j)$ denote the kth neighbor of j but with the distance measured in the input space between $\mathbf{w}_j$ and $\mathbf{w}_{j'}$:

$$n_1^V(j) : d^V(\mathbf{w}_j, \mathbf{w}_{n_1^V(j)}) = \min_{j^{'} \in V \setminus \{j\}} d^V(\mathbf{w}_j, \mathbf{w}_{j^{'}})$$

$$n_2^V(j) : d^V(\mathbf{w}_j, \mathbf{w}_{n_2^V(j)}) = \min_{j^{'} \in V \setminus \{j, n_1^V(j)\}} d^V(\mathbf{w}_j, \mathbf{w}_{j^{'}})$$

$$\vdots$$

Using the nearest neighbor indexing, define the ratios

$$Q_1(j,k) = \frac{d^V(\mathbf{w}_j, \mathbf{w}_{n_k^U})}{d^V(\mathbf{w}_j, \mathbf{w}_{n_k^V})} \tag{2.6}$$

and

$$Q_2(j,k) = \frac{d^U(j, n_k^U)}{d^U(j, n_k^V)}. \tag{2.7}$$

From these definitions we have $Q_1(j,k) = Q_2(j,k) = 1$ only if the nearest neighbors of order k in the weight space and output space coincide. But this is highly sensitive to local stretching of the map induced by a gradient in the input stimulus density [22].

This problem can be overcome by multiplying the $Q_\nu(j,k)$ for all orders of k. Authors in [22] defined two new indices

$$P_1(j,k) = \left(\prod_{l=1}^{k} Q_1(j,l) \right)^{1/k} \tag{2.8}$$

and

$$P_2(j,k) = \left(\prod_{l=1}^{k} Q_2(j,l) \right)^{1/k}. \tag{2.9}$$

For these indices P_1 and P_2 we have

$$P_1(j,k) \geq 1$$

and

$$P_2(j,k) \leq 1.$$

In P_1 and P_2 a different ordering of nearest neighbors is cancelled, as long as the first set of k nearest neighbors in U and in V are the same (not regarding their order).

The products P_1 and P_2 have the important property of being insensitive to constant gradients of the map and remain close to 1 as long as the second order contributions average out locally [22]. For the case where second derivatives do not average out locally, we combine P_1 and P_2 multiplicatively in order to find

$$P_3(j,k) = \left(\prod_{l=1}^{k} Q_1(j,l) Q_2(j,l) \right)^{1/2k}. \tag{2.10}$$

As a consequence of the inverse nature of P_1 and P_2, the contributions of curvature are suppressed while the violations of neighborhoods are detected by $P_3 \neq 1$. Also, since $P_1 > 1/P_2$ if the input space folds itself into the output space, and $P_1 < 1/P_2$ if the

output space folds itself into the input space, deviation of P_3 above or below 1 indicates whether the embedding dimension D^U is too large or too small, respectively [22].

A simple averaging of logarithm of $P_3(j,k)$ by summing over all nodes and all neighborhood orders gives the full-blown formula for the topographic product P:

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log(P_3(j,k)) \tag{2.11}$$

P is a numerical estimate of overall topology preservation. $P = 0$ indicates perfect preservation of the topology.

A measure of topology violation

The SOM transforms the patterns in feature space into responses of nodes in one or two dimensional lattice of neurons. This transformation retains metric-topological relationship among the feature vectors. A quantitative measure of topology preservation captures the extent to which the metric-topological relationship is retained. There can be different choices for constraints to be satisfied for a "perfect" topology preservation. Perhaps, the strongest is that for each pair of points in the feature space, the distance should be equal or proportional to the distance of the mapped points. A weaker one demands that the distances in the two spaces should have the same order. The topographic product [22], a popular measure, is based on the weaker constraint. Su et. al [317] proposed another kind of weaker constraint and a measure based on that. They observed that if a map is topologically ordered then the weight vector of each node should be more similar to the weight vectors of its immediate neighbors (8 neighbors for a 2-D SOM) on the lattice than to the weight vectors of its non-neighbors. Their measure is designed to detect the violation of this condition. The measure is especially suitable if the SOM is used for visualizing cluster structure of the data.

The method for 2-D SOM can be formulated as follows:
Let Λ_r be the set containing the immediate 8 neighbors of node r and Ω_r denote the set containing the nodes which are not immediate neighbors of node r. Let the size of the map be $m \times n$. Consider a node $i \in \Omega_r$ and another node $i_r \in \Lambda_r$ such that

$$i_r = \underbrace{argmin}_{k \in \Lambda_r} \|\mathbf{p}_i - \mathbf{p}_k\|,$$

where, $\mathbf{p}_i = (p_{i1}, p_{i2})$ is the position vector of the node i in the lattice plane and $\|\mathbf{p}_i - \mathbf{p}_k\|$ is the Euclidean distance between the nodes i and k. Since node r is closer to the neighboring node i_r than to i in the lattice plane, the weight vector of node r should be more similar to the weight vector of the node i_r than to the weight vector of the node

i. Therefore, if the map is preserving the topology then for each node r the following relation should hold:

$$\|\mathbf{w}_i - \mathbf{w}_r\| \geq \|\mathbf{w}_{i_r} - \mathbf{w}_r\| \text{ for } 1 \leq r \leq m \times n, i_r \in \Lambda_r \text{ and } i \in \Omega_r. \tag{2.12}$$

Now the quantitative measure of topology violation V is defined as:

$$V = \sum_{r=1}^{m \times n} \sum_{i \in \Theta_r} [1 - \exp^{-\|\mathbf{p}_i - \mathbf{p}_r\|^2}] \frac{\|\mathbf{w}_{i_r} - \mathbf{w}_r\| - \|\mathbf{w}_i - \mathbf{w}_r\|}{\|\mathbf{w}_{i_r} - \mathbf{w}_r\|}, \tag{2.13}$$

where $\Theta_r = \{i : \|\mathbf{w}_i - \mathbf{w}_r\| < \|\mathbf{w}_{i_r} - \mathbf{w}_r\| \text{ for } i \in \Omega_r \text{ and } i_r \in \Lambda_r\}$ is the set of nodes in Ω_r which violate the condition (2.12) with respect to node r. The measure of violation V has the following properties:

1. $V = 0$ if $\Theta_r = \emptyset$, i.e., the topology is perfectly preserved.
2. The larger the value of V the greater is the violation.
3. If $i \in \Theta_r$ and the nodes r and i is far apart in the lattice plane, their contribution to V will be high due to the factor $(1 - \exp^{-\|\mathbf{p}_i - \mathbf{p}_r\|^2})$.

Usually, perfect topology preservation (i.e., $V = 0$) is achieved if the dimensionality match is perfect and the distribution of the training data has strong similarity with the distribution of the nodes in the lattice plane. Otherwise, even though the dimensionality matches, there could be some topology violation due to existence of cluster structures and variation of density within the data. However, if the dimensionality matching exists, the violations are typically small and can be attributed to the disturbances produced due to the non-uniformity of the data in unfolding of the map during training. On the other hand, for high dimensional data, when the violation is caused by dimension mismatch, the value of the topology violation for same training data increases rapidly with the size of SOM.

2.3.2 The density matching property

The density matching property of the SOM is essentially a combination of two related properties. The first property concerns the ability of the SOM to *approximate the input space* by the set of weight vectors $\{\mathbf{w}_i\}$. This property relates SOM with *vector quantizers*. The other property is unique to SOM and concerns its ability to place weight vectors in the input space in (approximate) proportion of the probability of occurrence of sample data, thus making it possible to *reflect the statistics* of the data. These properties make the SOM algorithm attractive for many applications including clustering,

data reduction and visualization of the input data. For clarity we shall discuss these properties separately in the following.

Approximation of the input space

The SOM, considered a collection of weight vectors $\{\mathbf{w}_i\} \subset \Re^p$ provides a good approximation of the input space. In other words, the SOM stores the information about a large set of input vectors $\mathbf{x} \subset \Re^p$ by finding a smaller set of prototypes $\{\mathbf{w}_i\}$. This aspect of SOM relates it to the theory of *vector quantization* [110]. The relation between vector quantization and SOM algorithm is studied by Luttrell [230, 231]. A vector quantizer (VQ) can be thought of consisting two parts, an encoder and a decoder. Given an input vector $\mathbf{x}$ the encoder selects from a codebook, containing vectors representing the input space, a codevector $\mathbf{c}(\mathbf{x})$. The decoder reproduces the $\mathbf{x}'(\mathbf{c})$ which is an approximation of the original vector $\mathbf{x}$. The error/distortion introduced in the process is $d(\mathbf{x}, \mathbf{x}')$. The optimal design of the VQ strives to minimize the *expected distortion* defined by

$$D = \frac{1}{2} \int_{-\infty}^{\infty} f_{\mathbf{x}}(\mathbf{x}) d(\mathbf{x}, \mathbf{x}') d\mathbf{x},$$

where $f_{\mathbf{x}}(\mathbf{x})$ is the *probability density function* for the distribution of the input vectors. Commonly the square of the Euclidean distance between the vectors are used as the distortion measure. Then D takes the following form:

$$D = \frac{1}{2} \int_{-\infty}^{\infty} f_{\mathbf{x}}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}'\|^2 d\mathbf{x}. \tag{2.14}$$

The necessary conditions for minimization of the expected distortion D are embodied in *generalized Lloyd algorithm (also known as LBG algorithm)* [110] as follows:

Condition A**:** Given the input vector $\mathbf{x}$, choose the code $\mathbf{c} = \mathbf{c}(\mathbf{x})$ to minimize the squired error distortion $\|\mathbf{x} - \mathbf{x}'\|^2$ (*Nearest-neighbor condition*).

Condition B**:** Given the code $\mathbf{c}$, compute the reconstruction vector $\mathbf{x}' = \mathbf{x}'(\mathbf{c})$ as the centroid of the input vectors $\mathbf{x}$ satisfying condition A (*Centroid condition*).

The generalized Lloyd algorithm operates in batch mode and alternately optimizes the encoder and the decoder in accordance with conditions A and B respectively.

Now if we consider existence of additive noise ν in the communication channel between the encoder and decoder contaminating $\mathbf{c}$, the expected distortion takes the form:

$$D = \frac{1}{2} \int_{-\infty}^{\infty} d\mathbf{x} f_{\mathbf{x}}(\mathbf{x}) \int_{-\infty}^{\infty} d\nu \pi(\nu) \|\mathbf{x} - \mathbf{x}'(\mathbf{c}(\mathbf{x}) + \nu)\|^2, \tag{2.15}$$

Table 2.1: Correspondence between the generalized Lloyd algorithm with noise and the SOM algorithm.

Generalized Lloyd Algorithm with additive noise	SOM Algorithm
Encoder $\mathbf{c}(\mathbf{x})$	Best-matching node $i(\mathbf{x})$
Reconstruction vector $\mathbf{x}'(\mathbf{c})$	Weight vector $\mathbf{w}_i$
Probability density function $\pi(\mathbf{c}-\mathbf{c}(\mathbf{x}))$	Neighborhood function $g_{i,j}(\mathbf{x})$

where $\pi(\nu)$ is the probability density function of the noise component ν. With this scenario Luttrell [230, 231] showed that the conditions A and B must be modified as following:

Condition A': Given the input vector $\mathbf{x}$, choose the code $\mathbf{c}=\mathbf{c}(\mathbf{x})$ to minimize the distortion measure

$$D' = \int_{-\infty}^{\infty} d\nu \pi(\nu) \|\mathbf{x} - \mathbf{x}'(\mathbf{c}(\mathbf{x}) + \nu)\|^2. \tag{2.16}$$

Condition B': Given the code $\mathbf{c}$, compute the reconstruction vector $\mathbf{x}'=\mathbf{x}'(\mathbf{c})$ to satisfy the condition

$$\mathbf{x}'(\mathbf{c}) = \frac{\int_{-\infty}^{\infty} d\mathbf{x} f_{\mathbf{x}}(\mathbf{x}) \pi(\mathbf{c}-\mathbf{c}(\mathbf{x}))\mathbf{x}}{\int_{-\infty}^{\infty} d\mathbf{x} f_{\mathbf{x}}(\mathbf{x}) \pi(\mathbf{c}-\mathbf{c}(\mathbf{x}))}. \tag{2.17}$$

It was further demonstrated by Luttrell that if the pdf $\pi(\nu)$ is a smooth function and the additive noise ν is small compared to the original reproduction distortion $\|\mathbf{x}-\mathbf{x}'\|^2$, the condition A' can be approximated by condition A, while condition B' can be realized by stochastic learning. The stochastic learning involves choosing the input vectors $\mathbf{x}$ at random using the factor $\int d\mathbf{x} f_{\mathbf{x}}(\mathbf{x})$, and update the reconstruction vectors $\mathbf{x}'(\mathbf{c})$ as follows:

$$\mathbf{x}'_{new}(\mathbf{c}) \leftarrow \mathbf{x}'_{old}(\mathbf{c}) + \eta\pi(\mathbf{c}-\mathbf{c}(\mathbf{x}))[\mathbf{x}-\mathbf{x}'_{old}(\mathbf{c})] \tag{2.18}$$

where η is the learning-rate parameter and $\mathbf{c}(\mathbf{x})$ is the nearest neighbor encoding approximation to condition A. The update equation is identical to the SOM algorithm with the correspondence listed in Table 2.1. Thus it can be stated that the generalized Lloyd algorithm for vector quantization is equivalent to the batch training version of SOM with zero neighborhood size.

So the SOM algorithm can be treated as a vector quantization algorithm, which provides a good approximation of the input space. Further, as demonstrated by Luttrell, the SOM update equation can also be derived from the viewpoint of vector quantization.

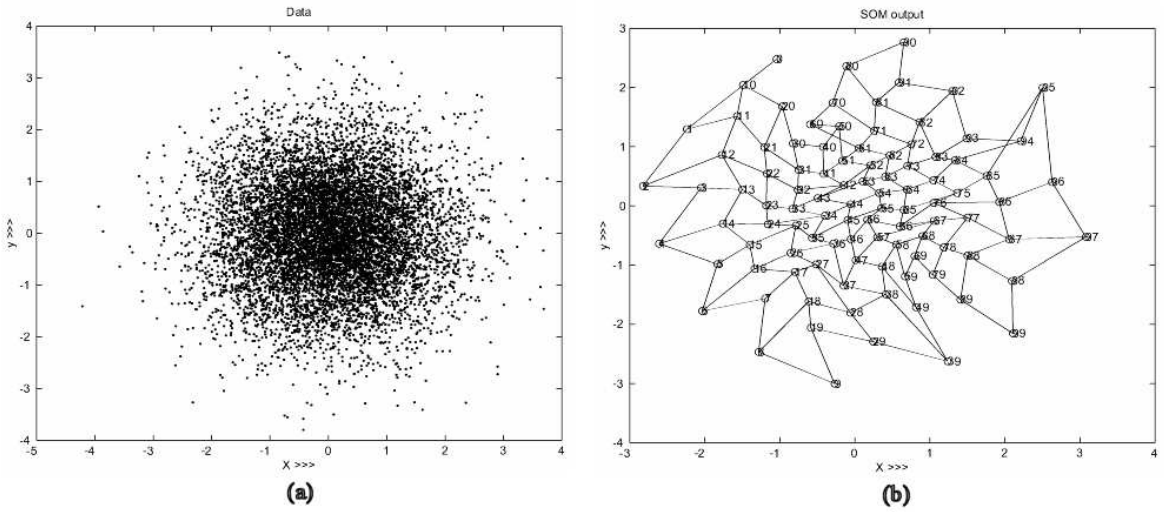

Figure 2.6: Reflecting statistics of input data (a) 2-D normally distributed input data (b) A 10×10 SOM trained with the input data.

The derivation also supports modelling of the neighborhood function in form of a probability density function, which is a common practice. In fact Luttrell [232] later proved that a zero mean Gaussian model is appropriate for such a system.

Reflecting sample statistics

Apart from approximating the input space, the SOM exhibits the property of capturing the variations in the statistics of the training data. The regions of the input space $\Re^p$ from which the samples are drawn with higher probability of occurrence are mapped onto larger domains of the output space than the regions in $\Re^p$ from which samples are drawn with lower probability of occurrence. As a consequence the higher probability regions are represented in the map with better resolution. Figure 2.6 shows (a) a normally distributed 2-D data and (b) a 2-D SOM trained with the data. From the figures the similarity in the distributional property can be easily discerned visually. However, it can also be seen that the capturing of the input statistics is not an exact one, rather approximate.

To discuss the property formally, let $f_{\mathbf{x}}(\mathbf{x})$ be the multidimensional probability density function of the random input vectors $\mathbf{x}$. By definition

$$\int_{-\infty}^{\infty} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = 1.$$

Now let $m(\mathbf{x})$ be the *magnification factor* of the map. Thus if there is a total number

of l nodes in the SOM, then

$$\int_{-\infty}^{\infty} m(\mathbf{x}) d\mathbf{x} = l.$$

Exact matching of the input density demands

$$m(\mathbf{x}) \propto f_{\mathbf{x}}(\mathbf{x}).$$

Unfortunately, for a general SOM, the magnification factor $m(\mathbf{x})$ cannot be expressed as a simple function of $f_{\mathbf{x}}(\mathbf{x})$. Only for the case of 1-dimensional SOMs two different results based on two variants of encoding scheme are available:

1. Luttrell [232] showed that if all the higher order terms in the distortion measure D' in Eq. (2.16) due to the noise $\pi(\nu)$ are retained then

$$m(\mathbf{x}) \propto f_{\mathbf{x}}^{1/3}(\mathbf{x}).$$

2. Ritter [291] showed that for standard form of the SOM algorithm we have

$$m(\mathbf{x}) \propto f_{\mathbf{x}}^{2/3}(\mathbf{x}).$$

Thus the property of SOM regarding reflecting input statistics is not a perfect one as shown for 1-D case, but an approximate one. Though theoretical results for other configurations of SOM are not available, computer simulations show that regions of low input density tends to be overrepresented while those of high density underrepresented.

Many researchers proposed various modifications to the standard SOM algorithms to make the representations faithful. Two of the important approaches involve (1) modification of the competitive process [81] that reduces the probability of winning for frequently winning neurons and (2) modification of the adaptive process [24, 219], where the weight update rule for each neuron within the neighborhood is controlled.

In a different approach Linsker [224] derived the learning rules for formation of topographic map using the principle of *maximization of mutual information* between the output and input signals. In this model, an exact correspondence between the output and input distribution is achieved. Similar information-theoretic approach was used by Van Hulle [331, 332].

2.3.3 Some limitations of SOM

Though the SOM algorithm is easy to implement, comprehensive theoretical analysis of SOM still eludes the researchers. Theoretical treatments of SOM under some special/simplified conditions have been carried out, an overview of which is provided in

Section 2.4.1. SOM is capable of vector quantization as well as vector projection. The vector projection is achieved through the nonlinear topology preserving mapping of the vectors onto the output (viewing) plane. The training of SOM and the formation of the map depend on several factors including the initialization, the sampling pattern of input data, the learning rate and the size of the neighborhood. Among these the learning rate and the size of the neighborhood decrease monotonically with time. Variation of these factors may lead to different results for the SOMs trained even with the same input data. Online SOM is order dependent, i.e., the final weight vectors are affected by the order of input sequence. In some cases different types of distortions of the map such as twists, rotation and mirroring may occur which lead to different neighborhood properties of SOMs [39]. The desired size of the output lattice, the step size and the size of the neighborhood may have to be found empirically for each data set to achieve useful results [19]. In [39] de Bodt et al. discussed these issues and proposed some statistical tools for assessing the reliability of SOMs.

The mapping produced by SOM is a very useful tool for visualization of the inherent structure and interrelationship of the data. However, for satisfactory results the network architecture of SOM need to be adequate with respect to the input data. SOM uses a fixed architecture in terms of number of nodes as well as their arrangements, which has to be defined prior to the training. Therefore, in case of unknown input data it is a non-trivial task to determine a network architecture beforehand that will give satisfactory result [240]. There are a few other problems with visualization. For example, in case of a discontinuous data cloud, interpolating units are positioned between data clusters. In visualization, these may give false cues of the data shape [334]. Typically the map is rectangular in shape, but the axes of the map grid usually do not have any clear interpretation. Moreover, The projection implemented by SOM alone is quite crude. The projection of a data sample is typically defined as the location of its best matching unit (BMU). It is often very hard to infer about the global shape of the data using the raw map grid alone. Sometimes this drawback is addressed by making a separate vector projection of the prototype vectors using e.g., Sammon's projection [145].

Further, the dimensionality of the input data may affect the topology preservation property of SOM. It was found that if the *intrinsic* dimension of the input data is significantly higher than the dimension of the SOM lattice, the network may not be able to fully represent the structure of the data [185]. This is caused by folding of the low dimensional SOM lattice in the high dimensional data space. As a result, data points from distinctly separate and distant clusters may be mapped onto neighboring nodes without any apparent order. On the other hand, given a data point there may be several nodes from different parts of the map with matches almost as good as the match with the BMU [334]. Conceptually, SOM can be built with lattice of any dimension to match

the dimensionality of the input data. However, in practice, the dimension of SOM lattice is at most 3 and usually 1 or 2. This limitation is a result of our inability to visualize a map beyond 3 dimensions.

The use of neighborhood update is crucial to the usefulness of SOM. However, for the nodes near the border of the lattice the neighborhood definition is not symmetric. Thus, compared to a node near the center, a winner node at the border will have less number of neighboring nodes that will be updated. This results in difference in density estimation for the border nodes compared to the center nodes. This is known as the *border effect* [209].

Although SOM have many other usages, it is often used as a clustering tool. In [19] Baraldi and Blonda discussed SOM in connection with several clustering algorithms. They have pointed out some limitations of SOM. For example, since SOM does not minimize any known objective function, termination is not based on optimizing any model of the process or its data. They also opined that "SOM should not be employed in topology-preserving mapping when the dimension of the input space is larger than three. SOM tries to form a neighborhood-preserving inverse mapping ϕ_T^{-1} from lattice G to input manifold X, but not necessarily a neighborhood-preserving mapping ϕ_T from X to G. To obtain a topologically correct map by running SOM algorithm, the adjacency structure of the preset graph G has to match the topological structure of the unknown manifold X".

Despite several limitations, SOM has been used successfully in numerous applications. Various modifications and variants of SOM have been invented to overcome many of these limitations when required by the applications. In Section 2.3.5 and Section 2.4.2 we shall discuss some of them.

2.3.4 Relationship between SOM and k-means clustering algorithm

While SOM is a neural network originally inspired by the neurobiological maps found in the brain [174], the k-means clustering algorithm owes its origin to statistical data analysis [233]. Both of them employ unsupervised learning and often classified as competitive learning methods [107]. However, due to the neighborhood update during the training, some authors [19, 107] distinguish SOM learning method from that of k-means (also known as hard k-means) as soft competitive learning. There are many variants of k-means algorithm [10]. Most popular one is the batch k-means algorithm [89, 140] discussed in Section 1.3.1. The batch algorithm minimizes a squared error criterion and converges to a local minima in a finite number of steps. Its convergence properties can

be studied as a limiting case of fuzzy c-means [31] algorithm. On the other hand, though batch SOM algorithm is useful for various types (even non-numerical) of data, the basic and popular version of SOM uses online updates and is described in Section 2.2. Theoretical studies on finding a single, closed form objective function whose optimization can lead to the formulation of the SOM learning algorithm still eludes the researchers [174]. It is to be noted that the version of k-means proposed by MacQeen [233] is also an online algorithm. However, MacQeen proposed decreasing the learning rate following the harmonic series $\varepsilon(t) = \frac{1}{t}$. Since the series diverges, there is no strict convergence of the algorithm [107]. Nevertheless, MacQueen's k-means algorithm and SOM algorithm are quite similar when the later is run with *winner-only* update scheme (which is often incorporated in the final stage of SOM learning). In the following, unless stated otherwise, we shall consider the batch k-means and the online version of SOM only.

SOM performs two-fold task, vector quantization and topology preserving mapping of the input data on a regular lattice. On the other hand, k-means algorithm is similar to the popular LBG algorithm [222], also known as generalized Lloyd algorithm (GLA) [228] used for vector quantization (VQ) [107, 300]. Thus, SOM and k-means are often compared from VQ perspective. The relation between them has been theoretically studied by Luttrell [230, 231] and briefly presented in Section 2.3.2. In [232] Luttrell studied the relationship between the SOM algorithm with Gaussian lateral feedback function and VQ with zero-mean Gaussian additive noise in the communication channel. Some authors [229, 363, 62] studied the optimality of SOM as vector quantizer and found that SOM is naturally an optimal VQ in minimizing the mean-square-error between reference vectors and data space [361] and often superior to k-means algorithm [62].

Note that, like k-means, even though SOM weight vectors generate a Voronoi tessellation of the data space, the goal originally envisaged for SOM is not clustering in the usual pattern recognition sense (finding homogeneous groups). This is an excellent tool for data reduction and visualization of the structure in high dimensional data through a low dimensional lattice space. The number of nodes in SOM is usually much larger than the number of clusters. Thus SOM places multiple weight vectors within a cluster and tries to match the density of the data distribution. However, some researchers compared SOM and k-means algorithm empirically [13, 17, 247, 303] from clustering perspective. Their approach is to use data sets with known number of clusters and train SOM with the same number of nodes for comparison. However, no definitive result has emerged. For example, Balakrisnan et al. [17] reported worse clustering performance of SOM compared to the k-means. Mingoti and Lima [247] reported similar results. On the other hand, Schreer et al. [303] reported better or equivalent performance of SOM in clustering. Similarly, Bacão et al. [13] obtained consistently better performance of SOM compared to the k-means algorithm. A well-known drawback of k-means algorithm is its

dependency on the initialization of the cluster centers [88]. With improper initialization, there may be several 'dead units', i.e., initial cluster centers that are not updated [107]. The SOM, due to its neighborhood update usually overcomes this problem. Further, for the same reason SOM is less likely to be trapped in a local minima [13, 19]. Thus SOM is expected to perform well in problems with multiple optima [13].

2.3.5 Variants of SOM

Though the basic SOM is a very useful neural network model, many modifications have been proposed by various researchers [174]. The objectives behind such modifications are to improve the efficiency and scope of application of SOM. One of the major modifications, as discussed earlier, is the generalized SOM or "batch map" algorithm, that can self-organize nonvectorial objects also. In [146, 147] the use of weighted Euclidean distance in the matching law is proposed. In the following we shall briefly discuss some other important variants of SOM.

Tree-structured SOM for reducing the complexity of searching the winner has been proposed by Koikkalainen and Oja [181]. Here starting from the root level one neuron SOM, larger SOMs are trained successively in lower levels. While training a lower level SOM, the higher level trained SOM is used for searching the "winner" within a subset of neurons in the lower level using a tree. Another technique for quick search of winner is proposed by Kohonen [173]. The technique is based on maintaining a table of pointers to *tentative winners* for the training samples based on the result of previous winner search. Thus, if the SOM is already roughly ordered, to search for the winer for a training vector, one can use the pointer to the previous winner and is likely to find the new winner within its neighborhood. SOM with Minimal Spanning Tree (MST) neighborhood is proposed in [147]. With such a neighborhood, though the self-organizing process does not lead to spatially ordered mapping, it makes vector quantization significantly faster and stabler [174]. In [118] Hagiwara derived a momentum term for accelerating the SOM training process. Walter and Ritter [342] investigated the application of a parameterized SOM that is capable of rapid learning from small training data sets, for computer vision and robotics. In a recent work Su and Chang [315] proposed a three-stage algorithm for fast training of SOM. In this method first k-means algorithm is used to select N^2 points, then a heuristic is applied for distributing these points over a $N \times N$ array. Finally the SOM algorithm with a fast cooling regime is applied for fine-tuning the map.

Many researchers noted that the fixed structure and size of conventional SOM impose some limitation on its ability to form quality mapping. Neural maps project data from an input space onto a neuron position in an output space (often lower dimensional) preserving neighborhood relation. A map-learning algorithm can achieve an optimal

neighborhood preservation *only if the output space topology roughly matches the effective structure of the data in the input space.* Martinetz and Schulten [240] observed the following:

> To obtain optimal result concerning the conservation of topology of the mapping as well as optimal utilization of all neural units, the topology of the employed network has to match the topology of the manifold of data which is represented. This requires prior knowledge about the topological structure of the manifold M, which is not always available or might be difficult to obtain if the topological structure of M is very heterogeneous, e.g., composed of subsets of different effective dimensions or disjunct and highly fractured.

To address the above problem different adaptive network architectures are proposed. Martinetz and Schulten [240] introduced the "neural-gas" networks, where there is no prefixed neighborhood relation. During the training neighborhood links are established between nearby nodes in the input space. Neural-gas algorithm results in a compact network with good preservation of neighborhood relations. However, the resulting map usually have the same dimension as the input space, making it difficult to use for visualization and no dimension reduction is achieved. In [105] Fritzke proposed "growing neural gas" network, where the neural gas network can increase the number of nodes adaptively based on the accumulated quantization error of the nodes. In another approach, to accommodate data visualization and dimensionality reduction along with the ability of adapt, Fritzke proposed [104, 106] "growing grid" self-organizing network. Here the nodes are located in an array of neurons, however, during the training new rows and/or columns of neurons can be inserted to the network. In [25, 338] Bauar and Villmann proposed "growing SOM (GSOM)" network that adapts with the data by growing along the existing dimension as well as increasing the dimension, if needed. Thus here the output space is restricted to hypercubic lattice. It can be observed that in all the above variants the focus is to produce an equal distribution of the input patterns across the map by adding neighbor units to the nodes which represent disproportionately high number of input data. In [74, 75] Deng and Kasabov proposed a dynamic version of the SOM, where network structure is evolved in an on-line adaptive mode.

To handle the data with hierarchical structures Miikkulainen [246] proposed a hierarchical feature map. Here instead of training a single flat SOM, a balanced hierarchical structure of SOMs is trained. Data mapped onto a unit is represented with more detail in the lower layer SOM assigned to the unit. In a recent paper [289] Rauber et al. proposed the "growing hierarchical self-organizing map (GHSOM)" that is consisted of growing grid model [106] of SOMs arranged in a tree hierarchy. Starting with the top level map, each map grows in size to represent a collection of data at a specific level of

detail. After a certain level of granularity of data representation is achieved, the units are analyzed. The units representing higher level of input data diversity are expanded to form new growing SOM at the next level. The lower level SOM will represent the data mapped into parent unit in greater detail. Recently Atukorale and Suganthan [15] proposed a hierarchical overlapped neural-gas network for classifier design. Marsland et al. [239] proposed a neural-gas network that adaptively grows with requirement, which is suitable for mapping dynamic data (i.e., when the distribution of input data changes with time). In a different approach to represent the hierarchical data Ritter [292] proposed SOM with non-Euclidean, especially "hyperbolic" output space. He observed that the hyperbolic space geometry is such that the size of a neighborhood around a point increases *exponentially* with its radius r. This property fits nicely with exponential scaling behavior of hierarchical, tree-like structures, where number of items r steps away from the root grows as b^r, where b is the branching factor. Hence the mapping, under suitable projections, aids interactive visualization of the hierarchical structure. Kasabov and Peev [151] used hierarchical SOM and fuzzy systems for phoneme recognition. In a recent work [260] Ontrup and Ritter proposed a hierarchically growing hyperbolic SOM for interactive data analysis and demonstrated the method with a large data set. They also introduced a fast winner-search strategy that substantially reduces the training time. In most of the tree-structured and hierarchical variants of SOM the training is performed layer by layer. In a recent paper [352] Xu et al. proposed a new architecture "Self-organizing topological tree (SOTT)" for training all the layers simultaneously in a tree-structured SOM. In the training they have considered not only spatial neighborhood within the same layer, but also the inter layer neighborhood through parent-child relationship among the nodes. Further, to avoid the non-optimality of tree-based winner search, they introduced a multi-path winner search procedure and the concept of "winner path".

Along with these structural variants discussed above, there are several functional variants of SOM proposed by the researchers. The "adaptive subspace SOM (ASSOM)" introduced by Kohonen [171] learns to detect input patterns invariant to translation, rotation and scaling. In ASSOM various map units adaptively develop filters of basic invariant features. The novel aspect of ASSOM is that the mathematical form of the filters are not prefixed, but learned from transformation occurring in the observations. A detailed account of it can be found in [174]. In [281] Pedrycz and Card introduced a fuzzy version of SOM. Another interesting class of modifications of SOM deals with sequential data. In its original form the input data to SOM are static vectors. In sequential data analysis along with individual observations, their context information is also important. Often "windowing" of the input stream is employed for capturing the context information, thus converting the sequential data into static data. However, this

approach leads to loss of information as well as increase in dimensionality [314]. In response to this problem various modifications of SOM are proposed to accommodate unsupervised recurrent learning for straight sequence processing. Among the prominent methods, in "temporal Kohonen map (TKM)" [50], the neurons implement recurrence in terms of leaky signal integration. For a sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_t)$, where $\mathbf{x}_i \in \Re^p$, the integrated distance of neuron i is computed by

$$d_i(t) = \sum_{j=1}^{t} \alpha(1-\alpha)^{(t-j)} \left\| \mathbf{x}_j - \mathbf{w}_i \right\|^2 .$$

This distance is used for winner search and takes into account an exponentially weighted past along with the current input. The parameter $0 \leq \alpha \leq 1$ controls the rate of signal decay during the summation. However, the update rule is the same as standard SOM using the unintegrated current input. In [183] Koskela et al. proposed "recursive SOM (RSOM)" which is an improvement over TKM. It takes into account integrated direction change for distance calculation as well as weight update. Among other methods "recursive SOM" proposed by Voegtlin [340] maintains the context information in the form of activation of the entire map in the previous time step and the distance calculation as well as the update use the context information. In [314] Strickert and Hammer describes the "merge SOM" where the weight and context of the last winner is merged by a weighted linear combination to form a context descriptor. Principe et al. [286] studied a self-organizing network motivated by reaction diffusion mechanisms. The network is capable of creating localized spatio-temporal neighborhoods for clustering. A good review of recursive SOM models can be found in [119].

Recently the use of SOM has been extended to the field of *structural pattern recognition*. In structural pattern recognition usually the objects are represented as nodes of a graph and the relation between these objects are modelled through edges connecting the nodes. Labels or attribute vectors are often attached to the nodes and edges of the graph. Thus it is possible to capture both unary properties of individual objects as well as interrelationship among them with a graph representation. In [115] Günter and Bunke proposed a self-organizing map for clustering of structured data represented as graphs. Strings can also be treated as special cases of the general graph. They have used *graph edit distance* as a distance measure and also derived a suitable update rule based on the edit distance. In [117] Hagenbuchner et al. proposed an extension of self-organizing map for processing structured data, namely labelled directed acyclic graphs (DAGs). A discussion on more variants of SOM can be found in [174].

2.4 A brief survey of the SOM: Theory and Application

So far we have talked about different properties of general nature and different variants of SOM. Now we shall discuss some theoretical results on SOM and present some applications of SOM. We have seen that SOM algorithm is easy to understand and implement. Often its results are visually verifiable. The algorithm was received with much enthusiasm in the neural network community and within a short time many successful applications based on SOM were reported. However, the theoretical studies on different properties of SOM did not achieve the same degree of success. In most of the cases successful theoretical studies are restricted to various special cases, a general theoretical treatment of the SOM algorithm still remains elusive. To our knowledge, the first comprehensive survey of research works on SOM appeared in 1990 in [169]. Two almost exhaustive bibliography of research papers on SOM can be found in [155] and [257] covering the periods from 1981 to 1997 (more than 3000 papers) and from 1998 to 2001 (more than 2000 papers) respectively. In this section first we provide a brief overview of some theoretical results. Then we shall have a brief look into the recent developments in SOM in response to different problems arising out of some emerging fields of applications. A more detailed account of the researches on SOM can be found in [174].

2.4.1 Theoretical analysis of SOM algorithm

SOM presents a neural model of self-organization. Algorithmically SOM can be considered a generalization of well-known competitive learning or vector quantization algorithm. Only significant difference is the use of neighborhoods among the neurons arranged in a regular lattice. Cottrell et al. [65] provide an excellent survey of the theoretical studies on SOM. According to them, the theoretical issues investigated by the researchers regarding SOM can roughly be divided into four types. They are as follows:

1. Vector quantization properties of SOM.
2. Derivation of SOM learning algorithm from some energy functions.
3. Self-organization or ordering of the units in the map.
4. Convergence of the weight vectors in the map.

Investigations into the vector quantization properties of SOM is aimed at comparing the performance of SOM with generalized Lloyd algorithm, also known as Linde-Buzo-Gray (LBG) algorithm and an alternative derivation of the SOM algorithm from VQ

perspective. Both issues are considered by Luttrell [230, 231]. Luttrell proved SOM as a generalization of the LBG algorithm where additive noise in the communication channel between the encoder and the decoder can be included into the model. Luttrell also succeeded to demonstrate that under such a condition a stochastic learning algorithm can be derived, which is equivalent to the SOM learning algorithm. A brief sketch of this work is already provided in Section 2.3.2.

Formal studies on SOM could have been much simplified if one can derive the adaptation laws as a process of minimization of some energy function. Then standard techniques like Liapunov function approach can be applied to study the convergence properties. Use of energy functions for the standard SOM algorithm have been studied in several papers [93, 170, 295, 324]. Here we provide a brief sketch of the formulation of energy function by Erwin et al. [93].

Erwin et al. used one dimensional SOM with one dimensional input for the derivation. A chain of N SOM nodes divide the input space Ω into Voronoi cells. Let us denote the Voronoi cell corresponding to the i-th node by $\Omega(i)$. At the end of the training the nodes are ordered such that for two indexes a and b if $a < b$ then $w_a < w_b$, where w_a and w_b are the weights of the nodes a and b respectively. During the training (i.e., when the nodes are not yet ordered) at any instance the nodes can be represented by a permutation of the index values. For each input, due to the update, a new permutation will arise and the index of a node will change. Let the transformation from a new index a to the old one i of a node be $i = T(a)$.

The adaptation will induce an average change in the weight w_i of the i-th node as

$$V_i(w) = \alpha \sum_{c=1}^{N} h_{ci} \int_{x \in \Omega(c)} (x - w_i) P(x) dx, \tag{2.19}$$

where c is the index of the winning node, h_{ci} is the neighborhood function and $P(x)$ is the probability density function of the input x. Assuming the range of the input patterns is 0 and 1, i.e., $x \in [0, 1]$, the Voronoi cells can be given as:

$$\begin{aligned} \Omega(1) &= \{x \mid 0 < x < (w_{T(1)} + w_{T(2)})/2\} \\ \Omega(n) &= \{x \mid (w_{T(n-1)} + w_{T(n)})/2 < x < (w_{T(n)} + w_{T(n+1)})/2\}, \forall 1 < n < N \\ \Omega(N) &= \{x \mid (w_{T(N-1)} + w_{T(N)})/2 < x < 1\}. \end{aligned} \tag{2.20}$$

Further assuming the sample pdf $P(x)$ constant [93] (i.e. uniform) and taking into account the above Voronoi cell definitions, we obtain the average change

$$
\begin{aligned}
V_{T(a)}(w) &= \sum_{b=2}^{N-1} h_{T(b)T(a)}[(w^2_{T(b+1)} - w^2_{T(b-1)})/8 \\
&\quad + w_{T(b)}(w_{T(b+1)} - w_{T(b-1)})/4 + w_{T(a)}(w_{T(b+1)} - w_{T(b-1)})/2] \\
&\quad + h_{T(1)T(a)}[(w_{T(1)} + w_{T(2)})^2/8 - w_{T(a)}(w_{T(1)} + w_{T(2)})/2] \\
&\quad + h_{T(N)T(a)}[1/2 - w_{T(a)} - (w_{T(N)} + w_{T(N-1)})^2/8 \\
&\quad + w_{T(a)}(w_{T(N)} + w_{T(N-1)})/2] \qquad (2.21)
\end{aligned}
$$

The change can be considered an average force acting on the weight $w_{T(a)}$. Thus the problem of finding the energy function reduces to finding out a potential function $E(w)$ such that

$$\frac{\partial E(w)}{\partial w_i} = -V_i(w). \qquad (2.22)$$

The potential function $E(w)$ would be the required Liapunov function. Erwin et al. [93] proved that such a function cannot exist for input signal with a continuous probability density function $P(x)$.

However, even though no global energy function can be found, in an earlier work Tolat [324] suggested that the update equations can be derived by minimizing a set of energy functions separately, one for each node. In [324] a simple system of three nodes in a one dimensional SOM is considered and the corresponding energy functions are derived. In [93] the set of energy functions are derived with more generality. The energy function for the node $i = T(a)$ is

$$
\begin{aligned}
E_{T(a)}(w) &= \tilde{E}_{T(a)}(w) + X_{T(a)}(w) \\
&= \epsilon \sum_{b=1}^{N} h_{T(a)T(b)} \int_{x \in \Omega(b)} \frac{1}{2}(x - w_{T(a)})^2 P(x) dx \qquad (2.23) \\
&\quad + \frac{\epsilon}{48} \sum_{b=2}^{N} (w_{T(b)} - w_{T(b-1)})^3 (1 - h_{T(b)T(b-1)}) P(\frac{1}{2}(w_{T(b)} + w_{T(b-1)})).
\end{aligned}
$$

The first term in the above energy function is the weighted quantization error due to the locations of the weight values, while the second term is a correction to the total energy due to the change in Voronoi cells during the training process. At the beginning of the learning process the second term dominates while as the map becomes ordered the first term becomes bigger.

Though no global energy function is found to exist for continuous input pdf, in [291] and [170] existence of an error function is demonstrated for discrete cases of SOM, i.e., when the input is generated from a finite set of samples. Ritter et al. [291] considered the

SOM algorithm a Markov process with some transition probabilities between different states $\{w_i \mid i = 1, \ldots, N\}$. They were able to show that for a discrete probability density $P(x) = \sum_{i=1}^{N} p_i \delta(x - x_i)$ there exists an error function $V(w)$ such that

$$V(w) = \frac{1}{2} \sum_{ci} h_{ci} \sum_{x_j \in F_c(w)} p_j (x_j - w_i)^2, \tag{2.24}$$

where $F_c(w)$ is the Voronoi cell of the best matching unit c.

Kohonen [170] derived a similar error function. A locally weighted mismatch of sample x was defined as

$$E_1 = E_1(x, w_1, w_2, \cdots, w_N) = \sum_{i=1}^{N} h_{ci} \|x - w_i\|^2. \tag{2.25}$$

For a discrete set of inputs $S = \{x_1, x_2, \cdots, x_T\}$ one can define a global mismatch metric

$$E_2 = \frac{1}{T} \sum_{t=1}^{T} \sum_{i} h_{ci} \|x_t - w_i\|^2. \tag{2.26}$$

This should be minimized. Using stochastic approximation technique, E_2 can be minimized using local approximation, i.e., by minimizing E_1 per sample basis. Thus we have

$$\begin{aligned} w_i(t+1) &= w_i(t) - \frac{1}{2} \alpha(t) \nabla_{w_i(t)} E_1 \\ &= w_i(t) - \alpha(t) h_{ci} [x(t) - w_i(t)], \end{aligned} \tag{2.27}$$

which is the standard SOM update equation. In the same paper Kohonen suggested an error functional for the continuous case as follows:

$$E = \sum_{i} \int_{x \in \Omega_i} \sum_{k} h_{ik} \|x - w_k\|^2 p(x) dx, \tag{2.28}$$

where the integration is done over the Voronoi cells Ω_i.

Almost all successful studies on the self-organization of SOM are limited to one dimensional SOM with one dimensional input space. The usual approach is to use Markov chain tools to prove:

> *Given the input space $x \in [0, 1]$ with pdf $P(x)$ and a set of SOM nodes with weights $\{w_i\}$ arranged in a linear chain, after the training the weights converge to one of the ordered configurations given by the following sets*
>
> $$F_N^+ = \{0 < w_1 < w_2 < \cdots < w_N < 1\}$$

and

$$F_N^- = \{0 < w_N < w_N - 1 < \cdots < w_1 < 1\}.$$

The two sets F_N^+ and F_N^- are called absorbing states.

In the Markov chain approach, an arrangement of nodes $F(t)$ is considered as a Markov process defined on the common probability space $(\Psi, \mathcal{T}, \pi)$, where Ψ is a sample space, $\mathcal{T}$ is a σ-algebra over Ψ and π is a probability measure on $\mathcal{T}$ such that $\pi(\Psi) = 1$. To prove self-organization one requires to show that $\exists\ T < \infty$ and $\delta > 0$ for which

$$\pi_{F(0)}(\{\psi \in \Psi : \tau_d \leq T\}) \geq \delta \ \ \forall\ F(0), \tag{2.29}$$

or the probability $\pi_{F(0)}$, of finding the set of samples ψ in the sample space Ψ which takes the neuron weights F from all initial conditions $F(0)$ to the organized configuration in a finite time τ_d is non-zero. Many researchers were able to prove self-organization using different assumptions about the neighborhood function and/or the input pdf $P(x)$. Cottrell and Pagès [66] have shown that from all initial conditions with $w_i \neq w_j \forall i \neq j$ and with a uniform pdf $P(x)$ and using two neighbors, the weights almost surely converge to an ordered state. In successive works [40, 94, 99, 100, 102, 298] the self-organization under different generalizing conditions are demonstrated.

However, there is little progress in the case of multidimensional SOM. This is mainly hindered by lack of comprehensive definition of the ordering in a multidimensional map. Even with 0-neighborhood case the Markov chain is found to be irreducible (i.e., no absorbing state exists). For details see Cottrell et al. [65].

Though the convergence of the weight values are related to the ordering property, it requires a separate treatment because the weight values may converge to unique fixed values even if the units are not ordered. Such a situation can occur when the algorithm has local minima due to an unsuitable neighborhood function. Erwin et al. [94] demonstrated existence of metastable states due to local minima. In [168] Kohonen studied the convergence of weight values for one-dimensional SOM with five nodes and neighborhood function having width 2 (i.e, winner and its 2-neighbors). He was able to prove the existence of an equilibrium solution. Solutions for more generalized cases are considered by Cottrell et al. [65]. They suggested a method using the ordinary differential equations (ODE) for the purpose. The issue of magnification factor of the map is also related to the convergence. We have already mentioned some related results in Section 2.3.2.

2.4.2 Applications of SOM

There are literally numerous applications where SOM is successfully applied. As early as in 1990 Kohonen [169] listed the following problems where SOM was applied successfully:

- statistical pattern recognition, especially recognition of speech,
- control of robot arms and other problems in robotics,
- control of industrial processes, especially diffusion processes in the production of semiconductor substrates,
- automatic synthesis of digital systems,
- adaptive devices for various telecommunication tasks,
- image compression,
- radar classification of sea ice,
- optimization problems,
- sentence understanding,
- application of expertise in conceptual domain and even
- classification of insect courtship song.

Since then the SOM has been used for solving such a wide spectrum of problems, it is no longer possible to enumerate them individually. Instead, Oja et al. [257] provide the following taxonomy of generic problem domains for which SOM has been used:

- image and video processing (548),
- pattern recognition (487),
- mathematical techniques (427),
- artificial intelligence (347),
- software (300),
- engineering in biology and medicine (268),
- information theory and coding (265),
- speech recognition (166),
- control problems (158),
- signal processing (157),
- design of circuits (92),

- information science and documentation (83),
- business and administrative problems (73),

where the numbers in the parenthesis indicate the number of papers related to the respective topics as found in the bibliographies [155] and [257]. A good description of several applications of SOM in various fields such as process and machine monitoring, speech processing, robotics, telecommunication, text processing (including a detailed discussion on famous WEBSOM project) etc. can be found in the book by Kohonen [174]. In [178] application of SOM in engineering problems are described.

Instead of trying to discuss the whole range of applications of the SOM, in the following we briefly review the recent advances on a few emerging fields of application, where SOM is increasingly being used for pattern recognition tasks. Namely, we choose data mining, information retrieval, bioinformatics, multichannel satellite image analysis, and image processing and coding. Some of these applications are related to the present thesis.

The SOM algorithm is often used in conjunction with other techniques such as fuzzy logic to perform these tasks. Several researchers proposed modifications of the SOM architecture and learning algorithm to make it more suitable for solving particular problems.

Data mining is the most vital step in *knowledge discovery in databases (KDD)*. In recent time, with the easy availability of computational equipment and a mature database technology, processes across various fields are being automated to enhance operational efficiency. This trend resulted in capture and storage of massive amount of data, often running into terabytes. This accumulated data hold the promise of containing useful nuggets of information, which if discovered and understood, likely to lead to better decision making and improved performance. However, our ability to analyze and understand such large data sets lags far behind our ability to gather and store the data. The emerging field of "knowledge discovery in databases" (KDD) is aimed at developing computational techniques and tools for extraction of useful knowledge from rapidly growing volumes of data. KDD is defined by Fayyad et al. [96] as

> *The nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data.*

The KDD process typically consists of several steps such as data selection, data cleaning and preprocessing, data mining, interpretation and validation of the result and finally using the discovered knowledge. The data mining stage concerns applying appropriate techniques or algorithms to understand and/or discover useful knowledge from the

data. Data mining includes tasks such as data visualization, classification, clustering, regression, predictive modelling, association analysis etc.

In data mining applications, usually the data are in high dimension and are enormous in volume. The analyst usually starts with very little knowledge of the patterns/knowledge hidden in the huge mass of data. So the data mining process is iterative and interactive in nature and extensively utilizes various exploratory data analysis and visualization tools/techniques. In this respect SOM is one of the very useful tools. Visualization techniques allow the analyst to choose proper methods for subsequent analysis as well as help to form hypotheses about the clusters/patterns existing in the data. Proper visualization of high-dimensional data requires structure-preserving projection of the data points to a lower dimensional space. The SOM algorithm itself produces a topology preserving nonlinear projection of the input data to the lattice plane. The SOM essentially performs two tasks: vector quantization and vector projection. Vector quantization produces from the original data set a smaller but representative data set in the form of weight vectors or prototypes which follow the distribution of the original data closely. This allows one to use SOM for data reduction. Due to the vector projection property, SOM forms a topologically ordered nonlinear mapping of the original high dimensional data onto a low dimensional regular grid. These properties can be exploited to form various hypotheses about the data.

The visualization of the raw map is possible only for data up to 3-dimension. However, researchers have developed several visualization methods for discovering the properties of the data. One of the simple yet effective methods for visualizing the possible cluster structures in the data on a 2-D SOM grid uses the *unified distance matrix* (U-matrix) introduced by Ultsch and Siemon [329]. At each node on the grid, it depicts the average distance of the associated prototypes with its neighbors using a gray scale representation. Thus it shows the clusters as patches of light color (or valley, if 3-d representation is made) separated by regions of darker color (or mountains in 3-D). A comprehensive account of various visualization methods for high dimensional data using standard 2-D SOM can be found in [334]. Recently several sophisticated schemes for SOM-based data visualization and analysis have appeared. In [156] Kaski et al. proposed a scheme for coloring the nodes of SOM through a nonlinear projection of the map into a suitable color space. The projection method preserves the important local distances as well as the global order. Subsequent projection to a lower dimension (say, 2-D) reveals the cluster structures in the data. Though SOM produces a topology preserving mapping, it is not distance-preserving. On the other hand Sammon's projection [145] tries to preserve the distance, but does not produce any mapping function. Thus to accommodate new data points, whole projection has to be computed again. In [264] Pal and Eluri proposed a method of using SOM to perform data reduction first, then apply Sammon projection to the

set of weights generated by SOM. Then a multi-layer perceptron (MLP) is trained with the weight vectors as input and the corresponding Sammon projections as the target. Thus, the MLP is able to realize the mapping which is equivalent to the Sammon's projection. Now the MLP is used for projection of complete data set for achieving a distance preserving projection. The method also serves the purpose of dimensionality reduction of the data. In [182] König proposed a scheme which first uses SOM for data reduction and then Sammon's non-linear mapping (NLM) of the prototypes followed by a NLM recall (NMLR) algorithm for a distance and topology preserving mapping of the whole data set. In the same paper he also proposed a fully connectionist method similar to Pal and Eluri. He applied the methods to large data sets and found that the hybrid method (using NMLR) is more stable and efficient than the fully connectionist method. In [316] Su and Chang proposed a new variant of SOM called "double self-organizing map" (DSOM) where the position vector of the nodes are also modified along with the weight vectors so that for similar data both the weights as well as node positions of the nodes are close. In [362, 361] Yin introduced a novel method called "visualization-induced SOM" (ViSOM) for multivariate data projection. Here also the aim is to produce a mapping with preservation of both topology and distance. Here the SOM update rule for the neighbors are modified to constrain the lateral contraction force between the neighboring neurons and the winner neuron. In [177] Kohonen and Oja proposed a method for visual feature analysis using SOM. Among others Rauber et al. [289] used a growing hierarchical SOM (GHSOM) for exploratory analysis of high-dimensional data. The method enables the analyst to discover hierarchical structure in massive volume of data. Alahakoon et al. [7, 8] used a dynamic growing SOM that can be used to discover cluster structures in the data in different level of details. Marsland et al. [239] proposed a growing SOM suitable for handling dynamic input distribution and novelty detection. A number of visualization methods for exploration of financial data are presented in [72].

Apart from using SOM for gaining information about the structure of the data distribution, it has also been used for finding the actual clusters. However, unlike most clustering algorithms, which attempt to find as many prototypes as the number of clusters, in SOM each potential cluster usually is presented by many prototypes. To make the clustering result interpretable, smaller number of clusters need to be identified. Hence to detect the clusters the SOM is usually augmented by a clustering of its prototypes. Such an approach can be found in [335], where SOM is used for data reduction and subsequently clustering of the prototypes is performed. Apart from the generic SOM based methods for data mining described above, SOM has been used in many application-specific data mining schemes. Among such recent works, in [308] SOM is used to prepare a map of web pages based on user navigation pattern. Kim and Cho [163] used fuzzy integral based method with an ensemble of adaptive SOMs for classifying web pages. Laakso et

al. [195] used SOM for mapping web link information. In [210] and [223] SOM-based tools for customer segmentation is developed. In [132] and [133] Hsieh used SOM for developing behavioral scoring model and credit scoring model respectively. SOM is also used for mining association rules in [49]. Kasbov et al. [150] used SOM for On-line decision making and prediction of financial and macroeconomic parameters. In [23] Bauer developed a SOM based tool for strategic decision-making in investment.

The SOM is also finding increasing attention for information retrieval tasks such as classification and indexing of text documents and images. With huge proliferation of text documents in electronic format, the traditional methods for searching documents such as keyword searching, manual categorization and indexing are becoming increasingly inadequate. For effective retrieval of documents one needs automated methods for content-based organization and retrieval. The first attempt to use SOM for information retrieval using text documents was probably due to Lin et al. [221] back in 1991. They formed a small map of scientific documents using the words in the titles only. However, to devise such methods effectively, one requires to represent the information content of a whole document in form of a numerical vector. A common method is to form a vector describing a document such that each component refers to some word and it is represented by some increasing function of the word frequency multiplied by an importance factor. This is known as *vector space model*. However, here each document vector has the dimension equal to the size of the vocabulary, which is vary large (usually of the order of tens of thousands for document collections of even moderate size). To reduce the dimensions to a manageable level various methods such as manual restrictions on the size of the vocabulary, latent semantic indexing, MatchPlus method etc have been devised [154, 245]. There are other issues like context sensitivity, synonymous words etc. which complicates the problem. In a comprehensive work, Kohonen and his coworkers developed the WEBSOM [154] method based on SOM algorithm that produces a document map based on the similarity of document contents. The method addresses several issues involved with text document organization and retrieval. Here *random mapping method* [293] used for dimensionality reduction without reduction of vocabulary. The average context of the words are encoded using SOM by forming a "word category map". The final document map is generated using the response of the word category map to the input vectors. The resulting map can be produced graphically to provide a graphic interface for document searching. A web-based demonstration of the method can be found in the WWW address http://websom.hut.fi/websom/. The method has been further refined with introduction of a batch version of SOM algorithm in [175]. Among other works on document classification using SOM, Lin [220] extended the earlier work in [221] to use full-text documents. The hierarchical SOM is used by Merkl [245] to cluster documents containing descriptions of software library components. Chen et al. [51] used SOM for

categorization and searching of internet documents. Recently Pullwitt [287] attempted to enhance the document clustering performance by integrating contextual information. Yang and Lee [358] proposed a SOM-based text mining method for generation of web directories. In [259] Ong et al. demonstrated an application of a hierarchical knowledge map for online news.

Content-based image retrieval (CBIR) is another challenging task related to information retrieval. Researchers have applied SOM to develop quite a few CBIR methods. In [51] an interactive method was developed where the local color features of a region-of-interest are used for image retrieval. Laaksonen et al. [196, 198, 197] developed PicSOM, that uses tree-structured SOM for interactive image retrieval. The method utilizes relevance feedback to fine tune the search process. Liu et al. [225] developed a retrieval method for lung CT images that utilizes the texture and geometrical shapes in the query image. Wu et al. [351] developed a SOM-based system for image retrieval from web that utilizes textual information about the image also. In [318] and [359] CBIR methods based on the shape of the objects in the images are developed.

The SOM has been used extensively for pattern recognition tasks in bioinformatics area (268 research papers as identified in [257]). Analysis of gene expression data and protein sequence data are among the most challenging tasks in bioinformatics. In [253] Nikkilä et al. developed a SOM based method for analysis and visualization of gene expression data. In the same article they also provided a review of contemporary works in the field. In a more recent work Ressom et al. [290] proposed a method using adaptive double self-organizing map for clustering of gene expression profiles. Similarly, SOM has been used extensively for analysis of proteins [12, 97, 111, 134]. Recently Kohonen and Somervuo [180, 312] developed an extension of SOM algorithm that can handle nonvectorial data which is suitable for analyzing sequence data in its original symbolic form.

Analysis of satellite images is another area of immense practical value where there is an ever increasing demand for sophisticated pattern recognition techniques. Neural network-based techniques have always been used extensively for this purpose [14, 27, 36, 112, 272]. Many researchers have developed SOM based systems for the same [19, 46, 60, 258, 323, 337, 354]. In [217] Lin et al. used multiple SOMs, each working with different set of features, cascaded with a neural fuzzy network for multispectral satellite image classification. In the current thesis we develop SOM based methods for generating and tuning fuzzy rule bases for classification task which are tested successfully for the analysis of multichannel satellite images with good performance. Here we also develop classifiers those use context information for the final decision making. Such contextual schemes are eminently suitable for satellite image analysis tasks.

The image and signal processing are other areas where SOM has been used extensively [255, 256] (548 papers on image and video processing and 157 papers on signal processing as counted by Oja et al. [257]). The excellent vector quantization capability of SOM algorithm makes it especially suitable for designing vector quantizers for image/signal compression. First use of SOM for designing vector quantizer for image compression probably was due to Nasrabadi and Feng [249] back in 1988. Subsequently many researchers developed data compression systems using SOM [9, 20, 35, 44, 45, 86, 120, 121, 212, 235, 284, 311, 321, 357]. In the current thesis we develop SOM based methods for vector quantizer design. These methods deal with two aspects of VQ design, (1) design of VQ aimed at reducing blockyness in the reproduced images and (2) fast codebook search methods for efficient encoder design.

Chapter 3

SOM: Robustness, Simplification and Topology Preservation[1]

[1]First part of this chapter has been published in [199] and the rest has been published in [202].

3.1 Introduction

We have seen in Section 2.3 that SOM has some interesting as well as useful properties. As a consequence, SOM algorithm has been investigated by many researchers and been applied to diverse fields of applications (Section 2.4). In the current thesis we develop some methods based on SOM for designing classifiers and vector quantizers. Before we describe them, in this chapter we report the results of some empirical studies on the robustness of the SOM algorithm. One of the most interesting properties of SOM is the topology preserving mapping of the input data to a regular output lattice. Several methods for quantitatively measuring the topology preservation of an SOM has been proposed by researchers [22, 73, 80, 79, 241, 294, 317, 339, 370]. In Section 2.3.1 we have described the popular measure "topographic product (P)" due to Bauer and Pawelzik [22] and the "topology violation measure (V)" introduced by Su et al. [317]. In this chapter we introduce a new quantitative measure of topology preservation based on the Kendall's rank correlation coefficient [160]. We use these three measures in two empirical studies reported in this chapter. We also propose a visualization scheme, that helps one to assess the level of topology preservation quite easily. First we study the robustness of SOM algorithm with respect to systematic as well as random absence of lateral feedback connections. Subsequently, we shall study some simplified variants of SOM with respect to topology preservation and vector quantization.

3.2 A new quantitative measure of topology preservation

The measure for topology preservation developed here is based on its most intuitive definition, i.e., under a topologically ordered mapping, two data points, close to each other in the input space are mapped to the same point (node) or nearby points in the output space of the map. Since the weight vectors of the SOM nodes are known to represent the distribution of input data, if topological order is preserved, the nodes located further on SOM lattice should have higher dissimilarity between their wight vectors than those which are closer together on the SOM lattice. Thus if we consider an individual node of the map, if topological ordering is preserved in the map, then the list of remaining nodes in SOM ranked by their distance from the chosen node in the input space and the same in the output space (i.e., SOM lattice) should have significant similarity. A quantitative measure of this similarity will indicate topology preservation with respect to the selected node. Kendall's rank correlation coefficient [160] can be used for such a measure. For assessing the overall topology preservation of the map the

average of the measures for individual nodes can be used.

3.2.1 Rank Correlation

When objects are arranged in order according to some quality/properties which they all possess to a varying degree, they are said to be ranked with respect to that quality. The arrangement as a whole is called a *ranking*. If the objects possess another such quality, another ranking of the same objects can be obtained. Thus several rankings of the same set of objects are possible depending on different qualities they possess. Naturally, if there is any relationship among these qualities, then it is likely to be reflected in the corresponding rankings also. Kendall's τ [160] coefficient is a measure of such relation. It is called rank correlation between coefficient two rankings.

Kendall's τ coefficient is computed as follows:

> Let R_1 and R_2 be two rankings of a set of n objects. Define the natural order 1,2,... as direct order (i.e., the pair, say, 2,3 is said to be in direct order and 3,2 is said to be in inverse order). Now for every distinct pair of objects from the set of n objects, set the value $v_1 = +1$ if they are in direct order in R_1, set $v_1 = -1$ if they are in inverse order. Similarly set v_2 according to the order in R_2. Multiply v_1 and v_2 to obtain the score for the pair of the objects. Let S be the sum of the scores for all pairs of objects (total $\frac{n(n-1)}{2}$ pairs). Then τ is defined as,

$$\tau = \frac{2S}{n(n-1)} \tag{3.1}$$

τ has the following properties:

1. If the rankings are in perfect agreement, i.e., every object has the same rank in both, then τ is +1, indicating a perfect positive correlation.

2. If the rankings are in perfect disagreement, i.e., one ranking is the inverse of other, then τ is -1, indicating a perfect negative correlation.

3. For other arrangements τ should lie between these limiting values. Increase of values from -1 to +1 corresponds to increasing the agreement between the ranks.

However, it may happen that several objects possess a quality to the same degree. This is the case of tied ranks. The common practice is to mark such objects in the rankings and make their contribution to the score 0 (thus, the score due to a tied pair in any of

the ranking becomes 0). If there are u objects tied among themselves in R_1, then $\frac{u(u-1)}{2}$ pairs will contribute zero to the score S. Similarly, v tied objects in R_2 will cause $\frac{v(v-1)}{2}$ pairs to contribute 0 to S. So total number of tied pairs in R_1 is

$$U = \frac{1}{2}\sum u(u-1)$$

and in R_2 is

$$V = \frac{1}{2}\sum v(v-1)$$

where the summation $\sum$ is over all tied scores in the respective ranking. Thus, τ for tied rankings is defined as

$$\tau = \frac{S}{\sqrt{\left[\frac{1}{2}n(n-1) - U\right]\left[\frac{1}{2}n(n-1) - V\right]}} \tag{3.2}$$

3.2.2 Rank Correlation-based measure of topology preservation

In an earlier work Bezdek and Pal [33] proposed a rank correlation based index of topology preservation using Spearman coefficient [160]. However, they have used the rankings of the elements of internode distance matrix computed in the input space and the output grid. In our approach, in the line of [22], we first consider each individual node and then aggregate the result to arrive at the overall figure. Consider a node j in a SOM with N nodes. Given node j, two ordered lists of $N-1$ nodes can be prepared based on their distances from the node j in the output space and in the input space. Similar orderings of nodes are also used for computing topographic product [22]. Though we have discussed them in the earlier chapter (Section 2.3.1), for the sake of completeness of the discussion we present the concept again.

Let us denote the output space, a finite regular grid $V(\Re^q) \in \Re^q$ and input space (i.e., weight space) R^p of SOM as U and V respectively in the following discussion.

The Euclidean distance in U is denoted as

$$d^U(j, j^{'}) = \|j - j^{'}\| \tag{3.3}$$

and in V

$$d^V(\mathbf{w}_j, \mathbf{w}_{j^{'}}) = \|\mathbf{w}_j - \mathbf{w}_{j^{'}}\| \tag{3.4}$$

where $j, j^{'} \in \{(x, y) \mid x = 1, 2, \ldots, m \text{ and } y = 1, 2, \ldots, n\}$ for $q = 2$.

The notation of nearest neighbor indices is as follows:
Let $n_k^U(j)$ denote the k-th nearest neighbor of node j with the distance measured in output space, i.e.,

$$n_1^U(j) : d^U(j, n_1^U(j)) = \min_{j^{'} \in U \setminus \{j\}} d^U(j, j^{'})$$

$$n_2^U(j) : d^U(j, n_2^U(j)) = \min_{j' \in U \setminus \{j, n_1^U(j)\}} d^U(j, j')$$

$$\vdots$$

In the same way let $n_k^V(j)$ denote the k-th neighbor of j but with the distance measured in the input space between $\mathbf{w}_j$ and $\mathbf{w}_{j'}$:

$$n_1^V(j) : d^V(\mathbf{w}_j, \mathbf{w}_{n_1^V(j)}) = \min_{j' \in V \setminus \{j\}} d^V(\mathbf{w}_j, \mathbf{w}_{j'})$$
$$n_2^V(j) : d^V(\mathbf{w}_j, \mathbf{w}_{n_2^V(j)}) = \min_{j' \in V \setminus \{j, n_1^V(j)\}} d^V(\mathbf{w}_j, \mathbf{w}_{j'})$$

$$\vdots$$

It is easy to see from the definition of $n_k^U(j)$ and $n_k^V(j)$ that the degree of topology preservation with respect to the node j is reflected in the degree of similarity of these two rankings. For good topology preservation these two rankings should be highly correlated. We can measure the degree of correlation using Kendall's rack correlation coefficient τ. However, due to regular grid structure of the output space, there will be considerable number of ties in the distance values measured in the output space (i.e., the sequence $\{n_k^U(j)\}$). Therefore, we need to use the Eq. (3.2) for computing τ_j, the rank correlation between $\{n_k^U(j)\}$ and $\{n_k^V(j)\}$. An overall estimate of topology preservation for the whole map can be obtained by averaging the τ_js as follows:

$$T = \frac{\sum_{j=1}^{N} \tau_j}{N}. \tag{3.5}$$

The topology preservation measure T proposed here do not attend the extreme values (+1/-1) due to large number of ties in the distance values over the output space, however as we shall demonstrate, it is very useful in comparing the topology preservation between SOMs of equal sizes.

3.2.3 Experimental study on the topology preservation measures

To have a closer look at the behaviors of three quantitative measures of topology preservation, we conduct an experiment. We take several snapshots of a 6×6 map being trained with 2-D data uniformly distributed over a square, at various intermediate stages of training. During the training the topological ordering builds up with increasing iteration. We measure the degree of topology preservation for these snap shots using the indexes P, V and T and compare them to get an idea of effectiveness of these measures. Before that,

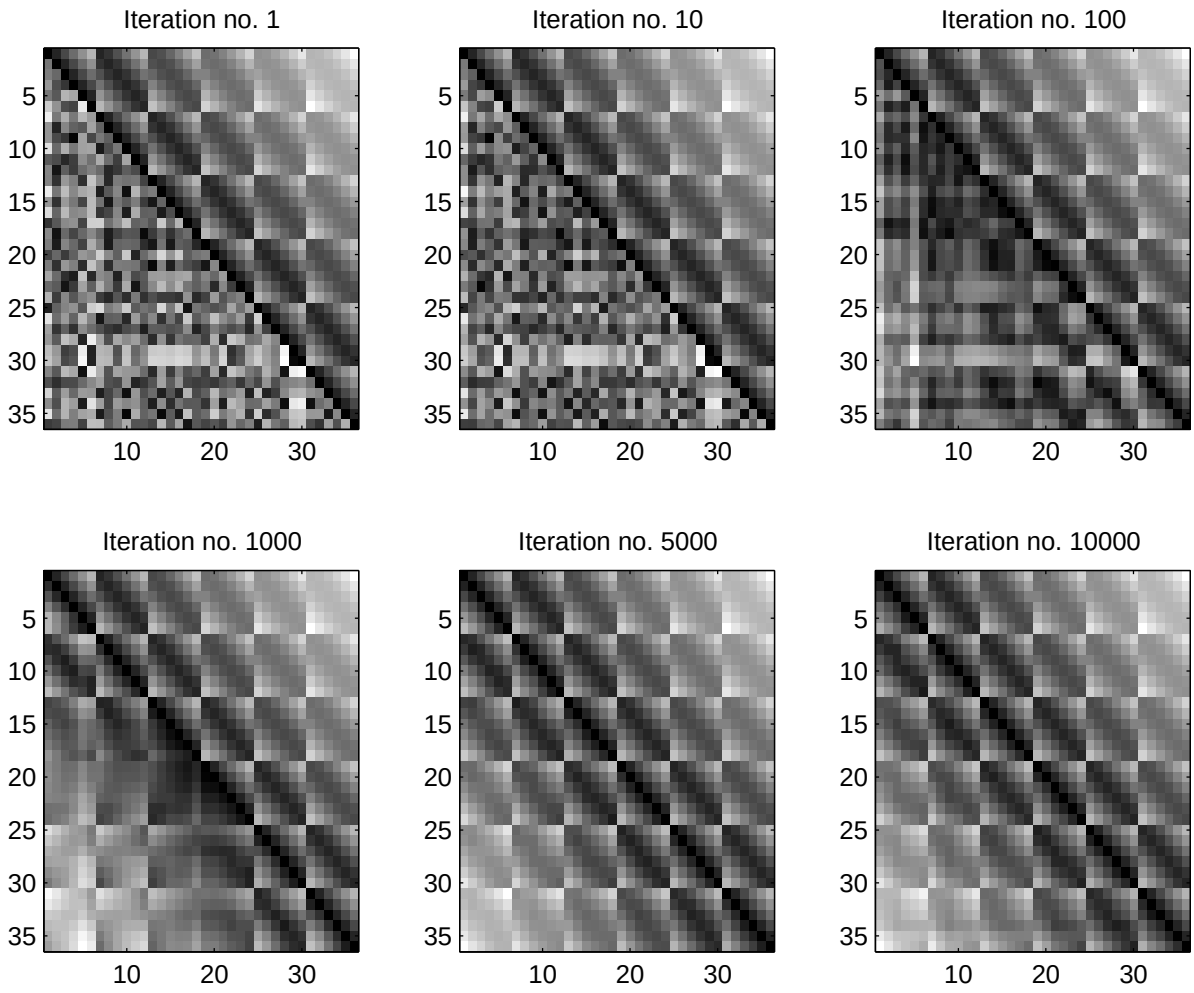

Figure 3.1: Distance based visualization of topological ordering process during the training. The gray values depict the normalized internode distances. The upper right triangle of the matrix depicts the inter-node distances on the SOM grid, the lower left triangular matrix depicts the internode distances in the input space.

to visualize the progress of topological ordering with iteration, we present a internode distance based visualization of the SOM under training. In this visualization we combine the internode distances in the SOM grid and in the input space in a single matrix. The upper right triangle contains the internode distances on the SOM lattice and the lower left triangle contains the internode distances in the input space. Both types of distances are scaled to lie in [0,1]. To visualize the matrix elements are represented by gray values [0,255] in proportion of their values. The distance matrices for the training snapshots are shown in Figure 3.1. From the figure the level of topological ordering is easily discernible from the level of similarity of two triangular halves of the matrices. The successive panels in the figure clearly show the gradual improvement in topological ordering of the SOM with progress of the training.

The variation of topology preservation property during the training as measured with

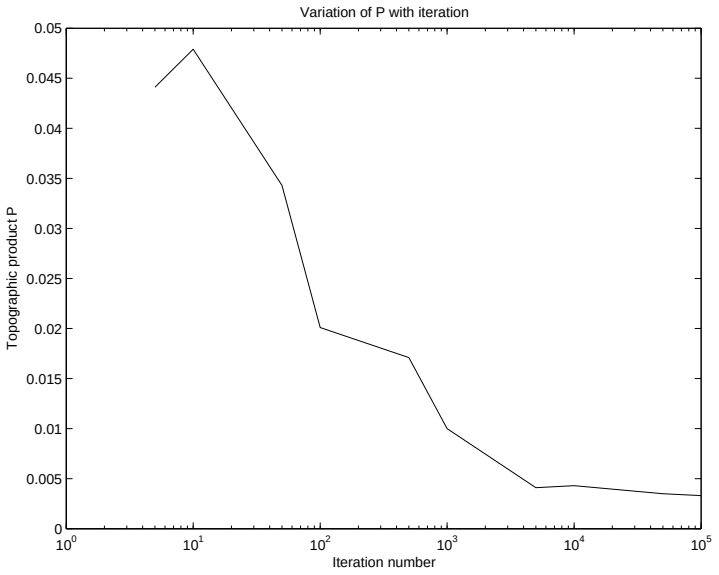

Figure 3.2: Topology preservation during different stage of training of SOM measured with topographic product P

the indexes P, V and T is depicted in Figures 3.2, 3.3 and 3.4 respectively. It can be seen from the figures that all of them depict the gradual improvement of topology preservation of the SOM as the training progresses. However, some differences in the behavior of the measures can be observed.

The topographic product P approaches the value zero for the perfect topology preservation. The magnitude of difference from zero indicates the lack of topology preservation. Though, in Figure 3.2 the variation of P clearly demonstrates the progress of SOM with iterations in preserving topology, the range of values for P is very small. Even at the beginning (i.e., before the training started), its value is only 0.05. Hence, this measure is difficult to use meaningfully for a single SOM. It can be better used in comparing properties of different SOMs. This measure is also affected heavily with dimension mismatch between the SOM lattice and the input data.

The measure for topology violation V becomes zero when topology is preserved and increases rapidly with violation. As can be seen from Figure 3.3, it starts with a high value (77) and varies rapidly. Thus it gives a good resolution when topology is significantly violated. However, as can be seen from the figure, it attains low values at iteration 500 onwards. Thus, it lacks differentiation ability when topology preservation is reasonably good and cannot detect further improvement. Its range of values increases rapidly with increase of SOM size. Hence, it is difficult to use it to compare SOMs of different sizes.

The rank correlation based index T (Figure 3.4) proposed here is an intuitive yet useful measure with good resolution both at the lower and the higher ends. Its value increases with increase of topology preservation. The value varies here over a range of 0 to 0.35

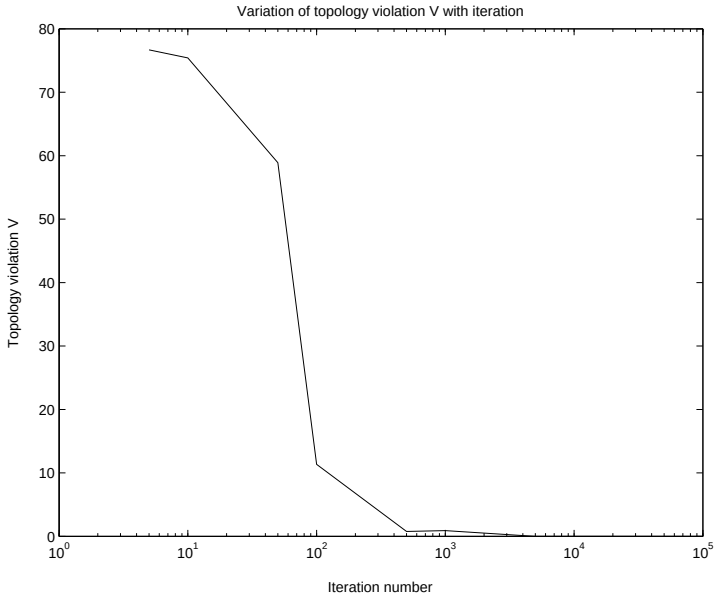

Figure 3.3: Topology violation during different stage of training of SOM measured with V

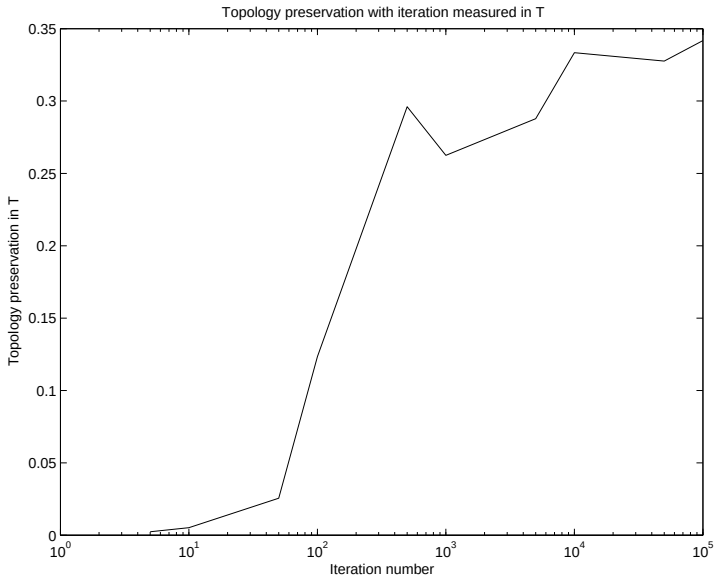

Figure 3.4: Topology preservation during different stage of training of SOM measured with T

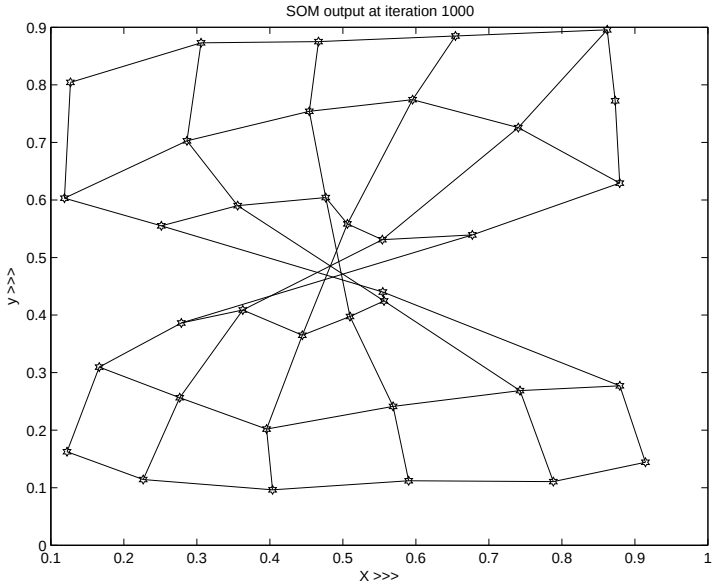

Figure 3.5: The twisted SOM at an intermediate stage of training

(approximately). Thus, it overcomes the major shortcomings of other two measures studied here. Further, it can be seen in Figure 3.4 that the value drops at iteration 1000. This is due to the fact that the map is twisted (Figure 3.5) at that stage of training. Out of the three measures studied, only T is capable of identifying such a situation clearly.

3.3 Robustness of SOM in preserving topology with respect to link density

Robustness of a system implies its ability to retain functional properties under non-ideal operating conditions. The theoretical studies of robustness of neural networks are also part of the general theoretical studies on them. Here, usually the aim is to explore the boundary conditions of the system and environmental parameters within which the network performance can be assured. The theoretical studies on the properties of SOM are reviewed in Section 2.4.1. However, in case of SOM, satisfactory theoretical results are available mostly in very simple cases, whereas SOM is applied in practice to handle many complex problems. Here we are interested in studying the capability of SOM to retain its important functional property "topology preservation" when certain type of partial system failure occurs. In [226] Liu and Michel analyzed the robustness of a class of bipolar associative memory neural networks with sparse interconnection under perturbation. They analyzed the upper bound of the perturbation of parameters and presented an economic design procedure for VLSI implementation of the networks. In this work we analyze empirically the robustness of SOM under certain conditions.

The SOM is originally inspired by the discovery of different spatially ordered maps in the brain [11, 165, 174]. Many of them are found in cerebral cortex area for different perceptual tasks. Originally Kohonen modelled the SOM algorithm in the form of a system of coupled differential equations [166, 167]. The computer simulation using them treated each neuron independently and did not demand any synchronization among the neurons. The topological ordering is achieved due to various lateral feedback connections among the neighbors. However, such a simulation (mimicking the biological neurons) is computationally intensive and ill-suited for practical applications. For practical purposes a simpler algorithm [169] is used which leads to functional appearance of the topologically ordered maps.

The simplified SOM algorithm (described in Figure 2.2) is usually implemented by simulating it on serial computers. During the training, in each iteration, the update of the winner and its neighbors are done in a way equivalent to a parallel operation. For quick learning, initially the neighborhood is defined large enough to cover almost whole viewing plane (i.e., SOM lattice). Thus there is an *implicit* assumption at work that every neuron (a possible winner) is connected to all the other neurons by some kind of direct communication link, so that the winner can induce the update of another neuron if it happens to fall within the neighborhood of the winner. Such a scenario of complete connection among biological neurons is highly unlikely. Also in hardware implementation of SOM, where parallel operations can be achieved, this kind of complete connection is difficult to maintain. Here we investigate the effect of partial absence of such connections on the topology preservation property of the practical SOM algorithm. The absence of the connection between a neuron (say a) and another (say b) will result in no update of b when a is a winner, even though b falls within the neighborhood of a. We shall call the proportion of existing connections between a neuron and rest of the neurons as the "link density" of the neuron, expressed in percentage value. A value of 100% means complete connection (the usual SOM), while 0% signifies total absence of connection, that results in absence of neighborhood update (i.e., the learning becomes simple competitive learning).

In this context, we can think about two different types of deviations from full connectivity. The first case can be thought as natural or systematic one. A neuron is more likely to be connected to the nearby neurons than to those far away. This can be modelled using a probability distribution such that the probability of a link between two nodes being present is inversely proportional to the Euclidean distance in the output space between the nodes. This situation is analogous to what can be expected in biological networks or may be desirable for designing hardware implementations economically.

The other situation involves random absence of links. The situation is analogous to

damage caused in the biological maps due to accident and diseases or random failures of components in hardware implementations. In the following we investigate empirically both the situations separately with different levels of link densities. The study will provide some indication of the robustness/fault-tolerance of the algorithm especially if the training algorithm is implemented on hardware to run in parallel mode. To measure the topology preservation property we use three indexes, (1) topographic product [22] P (Eq. (2.11)), (2) measure of topology violation V proposed by Su et. al [317] (Eq. (2.13)) and (3) rank correlation based measure T (Eq. (3.5)) introduced in this chapter.

3.3.1 Experimental Results

The data set used in all experimental studies reported here contains ten thousand 2D points uniformly distributed over a square. We have studied both the *cases* described above: 1) when the probability of a link existing between two nodes is inversely proportional to their distance on the SOM lattice and 2) when the absence of links are random. The former situation is simulated using the procedure ***SelectNeighbors(j)***. Given a target value of link density LD, in a SOM of size $N \times N$, each node j should be connected to $n = (N^2 - 1) \times (LD/100)$ other nodes in the SOM. The outline of the procedure for node j with lattice coordinate (a, b) is given in the Figure 3.6.

For the second case, for a node j, n connected nodes are chosen randomly from the set of $N^2 - 1$ remaining nodes.

For both the cases we have experimented with link densities varying from 0% to 100% in steps of 5%. For each value of link density 5 SOMs are trained. All the SOMs are of size 10×10. For a particular link density, all parameters except the randomization seeds for different runs, are identical. The results for case 1 and case 2 are summarized graphically in Figures 3.7 and 3.8 respectively. Each of the figures is divided into four panels. The panel (a) contains the view of the maps for some selected SOMs with different link densities. The panels (b)-(d) depict the variation of topology preservation property w.r.t the link density measured in topographic product P (Eq. 2.11), index of topology violation V (Eq. 2.13) and the rank correlation based index T (Eq. 3.5) respectively. In each of these panels the measurements for 5 SOMs for each link density is marked with the symbols $\circ$, $+$, $\times$, $\diamond$ and $\square$ respectively and the solid line represents the averages of 5 measurements.

As expected, in both of the cases, with decrease in link density the topology preservation suffers. All the three indexes agree over that. However, the topological product values are difficult to interpret in this context since its deviation from 0 (the perfect preservation) is more of a indicative of dimensional mismatch between input and out-

Procedure $NeighborSet^j$ = **SelectNeighbors**(j)

For all nodes k with lattice coordinate (x, y), $k \neq j$ compute

$d_{inv}(k) \leftarrow \frac{1}{\sqrt{(a-x)^2+(b-y)^2}}$

End For

For $i = 1$ to $i = N^2 - 1$

$Pr(i) \leftarrow \frac{d_{inv}(i)}{\sum_{k=1}^{N^2-1} d_{inv}(k)}$

End For

For $i = 2$ to $i = N^2 - 1$

$CumPr(i) \leftarrow Pr(i) + Pr(i-1)$

End For

$NeighborSet^j = \phi$

While $\mid NeighborSet^j \mid < n$

Generate a random number R from a uniform random number generator.

Perform a binary search in $CumPr$ to find the entry, nearest and larger than R.

Convert the index of the entry found to the corresponding lattice coordinate (x', y').

If $(x', y') \notin NeighborSet^j$

$NeighborSet^j \leftarrow (x', y')$.

End If

End While

Figure 3.6: The procedure for selecting connected nodes in case 1

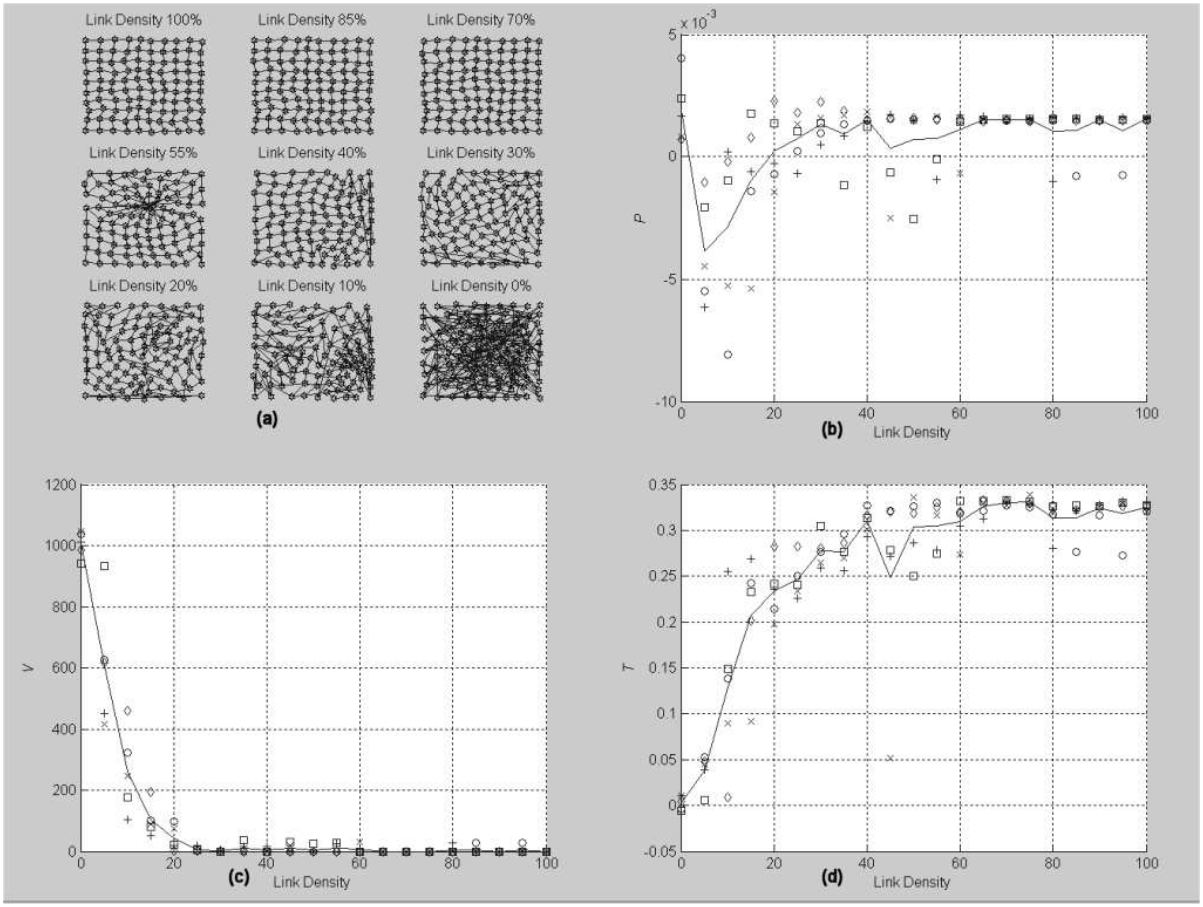

Figure 3.7: The graphical results when the probability of absence of a link is proportional to the interneuron distance. (a) Views of some SOMs with different link densities. (b) Variation of topographic product P with link density. (c) Variation of V with link density. (b) Variation of T with link density.

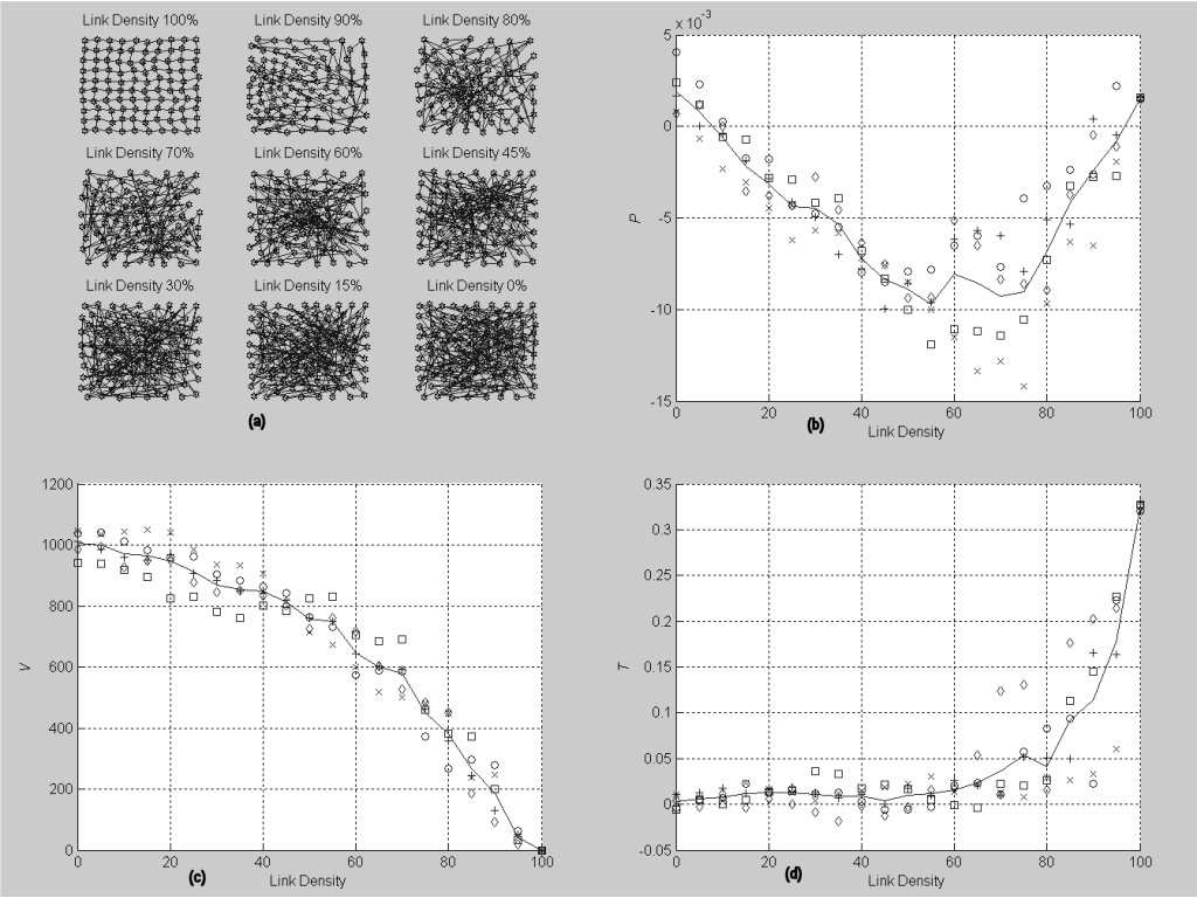

Figure 3.8: The graphical results when the probability of absence of a link is random. (a) Views of some SOMs with different link densities. (b) Variation of topographic product P with link density. (c) Variation of V with link density. (b) Variation of T with link density.

put spaces. The other two indexes reflect the situations better, though V shows relative insensitivity to lack of topology preservation due to cases like twisting of the map. Moreover, beyond a threshold on link density, V increases rapidly. The variation of the third coefficient T is more consistent with the visual inspection of the maps. T depicts, what one can expect looking at the maps. It is also sensitive to the changes due to twisting of map.

All three measures indicate good robustness of the SOM algorithm in case 1. It can be seen from Figure 3.7 that the significant decrease of topology preservation cannot be detected until the link density falls to nearly 40%. In contrast, in case 2 considerable decrease in topology preservation can be observed even at link density as high as 90%.

3.4 Simplified SOMs

The topology preservation property of SOM is attributed to the neighborhood update mechanism of SOM during the training stage. During the training, for a winner node r, the node i is updated according to the rule [169]:

$$\mathbf{w}_{i,t} = \begin{cases} \mathbf{w}_{i,t-1} + h_{ri}(t)(\mathbf{x} - \mathbf{w}_{i,t-1}) & \text{if } i \in N_t(r) \\ \mathbf{w}_{i,t-1} & \text{otherwise,} \end{cases} \tag{3.6}$$

where $h_{ri}(t)$ is the product of a learning parameter α_t and a *lateral feedback function* $g_t(dist(r,i))$. Thus the update of a node i for winner node r depends on two factors, 1) the strength of lateral feedback function and 2) the width of the neighborhood function $N_t(r)$. The first factor determines the extent of the update while the second determines whether the node qualifies for update at all. These two restrictions together implement a localized nature of the learning. In [174] the "mexican hat" function (Figure 3.9(a)) was recommended as an ideal lateral feedback function as it is closer in functionality to the kind of interaction observed in biological systems. However, in practice that is seldom used. Instead the Gaussian function (Figure 3.9(b)) $g_t(dist(r,i)) = \exp^{-dist^2(r,i)/\sigma_t^2}$ is used almost exclusively. Though the Gaussian function is a monotonically decreasing one, it attains value 0 only at infinite distance. Hence to use the function alone without explicit neighborhood boundary will require update of all nodes, though for most of them the amount of update is negligible. This necessitates explicit definition of the neighborhood boundary to achieve computational efficiency in simulation of standard SOM algorithm on a sequential computer. On the contrary, for a hardware implementation of SOM that could exploit the high degree of parallelism inherent in the architecture, the design becomes more complicated by the notion of an explicit neighborhood boundary and a complex form of lateral feedback function used in the algorithm. Hence, it would be

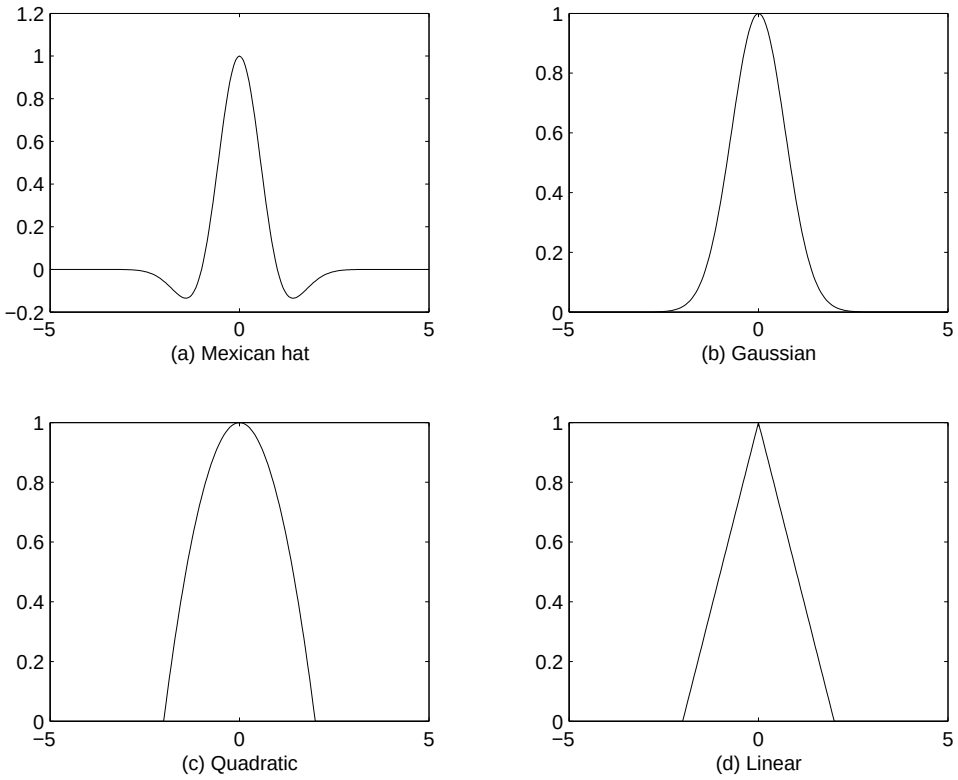

Figure 3.9: Lateral feedback functions.

useful if simpler lateral feedback functions, for which explicit neighborhood boundary need not be specified, can be found with similar performance. They can ease hardware implementations of SOM significantly.

On the other hand, there are cases where a special neighborhood function might be useful. It was noted by the researchers that due to fixed size and lattice structure of standard SOM, for good quality of mapping it is necessary that the intrinsic dimension of the input data and the SOM lattice should approximately match [240]. We have discussed this issue and various variants of SOM for high dimensional complex data proposed by researchers in the section 2.3.5. Though topology preservation is one of the most interesting property of SOM, it is also an attractive tool where the aim is to extract prototypes or data reduction. This is due to the fact that with neighborhood update during the training, under-utilization of the prototypes is less likely compared to clustering algorithms like k-means. This problem is considered in [147] and it was found that for standard SOM due to its rigid lattice structure and the conventional neighborhood function defined over the lattice, for complicated data sets, there might sill be some prototypes placed in a region where no data point exists. Kangas et al. [147] proposed the use of alternative neighborhood functions. They specifically studied the "minimal spanning tree (MST)" neighborhood and reported that to be superior to standard SOM with square neighborhood function for prototype extraction tasks from

non-uniform input data distribution. It must be noted that by using such a specialized neighborhood function, the topology preservation is likely to suffer [174]. However, if the major aim is prototype extraction, a neighborhood function compatible with the data distribution is likely to perform better.

In the following we describe several simplified variants of SOM and their performances. We divide the variants into two broad classes. The first one consists of SOMs with different forms of lateral feedback functions which are detailed in subsection 3.4.1, the other has the SOMs with several variation of tree neighborhoods and these are described in subsection 3.4.2. We call the former class simplified SOM (SSOM) and the later tree SOM (TSOM). The topographic product P and the rank correlation coefficient-based measure T are used as measures of the topology preservation. The data sets used include (1) uniform square (uniform distribution of 2-D points over a square), (2) two spheres (3-D points distributed uniformly over two spheres), (3) iris data (150 4-d points), and (4) Y-data (2-d points randomly distributed on a Y).

3.4.1 Simplified SOMs

Though the possibility of several different lateral feedback functions for Kohonen's SOM is mentioned in [124], no detailed studies of these possibilities have come to our notice. Here we explore three lateral feedback functions. The first one is the Gaussian lateral feedback function:

$$g_t(d) = \exp^{d^2/\sigma_t^2},$$

σ_t being the standard deviation of the Gaussian distribution. But the update of neighbors are not limited by the explicit neighborhood $N_t(r)$. All the nodes are updated. The extent of update depends on the value of $g_t(d)$, where $d = dist(r, i)$, r being the winner node. We call this one as SSOM with pure Gaussian neighborhood. We call it simplified as we abandon the use of any explicit neighborhood limit.

Other two SSOMs explored here employ quadratic and linear feedback functions as follows:

The quadratic function:

$$g_t(d) = 1 - \frac{d^2}{b_t^2} \text{ (Figure 3.9(c))}$$

The linear function:

$$g_t(d) = 1 - \frac{d}{b_t} \text{ (Figure 3.9(d))}$$

where d is the distance on the lattice of the node from the winner and b_t is the radius of the circle centered at the winner beyond which $g_t(d) = 0$. b_t decreases with time.

3.4.2 SOMs with Tree Neighborhood

Tree structures are suitable for defining an unambiguous neighborhood on the set of weight vectors because they are acyclic by definition. The effect of the neighborhood defined over a minimal spanning tree (MST) constructed on the set of weight vectors is studied in [147]. It was found to be more efficient in terms of prototype placement for non-uniform data. In [147] it was not clearly mentioned on which underlying graph the MST was constructed. To construct an MST we can use different graphs depending on various constraints imposed on the internode connectivities over the SOM lattice. Here we study two possibilities: (1) a complete graph on the weight vectors so that there exists one edge between every pair of nodes, which we assume to be the case studied in [147] and (2) a graph on the weight vectors in which each node is connected to its immediate neighbors in the logical output space (i.e., SOM lattice). We call the MST constructed over the former as complete MST (CMST) and the later as restricted MST (RMST). As evident from their construction, RMST requires much less computations than CMST because for the former the underlying graph has much less number of edges. We investigate the performance of both in terms of prototype placement as well as topology preservation.

In recent time the SOMs with tree neighborhood has been used for skeletonization of shapes in an image [306]. For this problem, along with good prototype extraction, a spatial connectivity among the prototypes is very useful for automated processing of the skeletonized shape. This can be achieved with suitable TSOMs. In [306] Singh et al. used MST-based SOM proposed in [147]. The TSOM with restricted MST neighborhood proposed here can be a computationally less expensive alternative for the task than the TSOM with CMST neighborhood.

Authors of [147] suggested re-computation of the MST once in every 200-500 iterations. However, computation of MST is quite expensive. If the aim is to extract prototypes from complex data without under-utilization of the prototypes, not topology preservation, then the essential requirement is to define an unambiguous neighborhood function, so that for a given winner there exists an unambiguously identifiable set of other nodes to be updated. In this regard, any tree neighborhood defined over the nodes will suffice. Also the recomputation of the tree is not necessary. In this section we propose a computationally inexpensive tree neighborhood. Here a complete graph is assumed, a list of available nodes is kept, from which nodes are deleted as soon as they are included in the tree. Initially the list contains all nodes. A starting node is picked at random and then on each step an available node is picked at random and added to the tree as a neighbor of a node randomly picked from those already on the tree. This is continued till all the available nodes are exhausted. This tree is used to define the neighborhood

SOM Type	Uniform Square			Two Spheres			Iris			Y-Data		
	P	V	T	P	V	T	P	V	T	P	V	T
SOM *Standard*	0.0031	0	0.3546	0.0096	0.3609	0.2979	0.0287	2.3089	0.2065	0.0294	5.5005	0.1935
SSOM *Gaussian*	0.0025	0	0.3944	0.0123	0.2928	0.2928	0.0279	1.0966	0.2441	0.0326	4.6675	0.1974
SSOM *Quadratic*	0.0030	0	0.3484	0.0054	1.6225	0.2353	0.0243	1.4319	0.2552	0.0303	5.5246	0.1630
SSOM *Linear*	0.0031	0	0.3652	0.0072	1.3682	0.2162	0.0235	2.2408	0.2378	0.0929	5.4401	0.1878

Table 3.1: Performance measure of the standard SOM and SSOMs with respect to topology preservation.

throughout the training, i.e., the tree is not recomputed during the training. We call this one arbitrary tree neighborhood (ATN). This variant of TSOM is not likely to preserve topology, but can be used for prototype extraction in an inexpensive way. This can be more preferable than a pure competitive learning network (i.e., with no neighborhood update) since it has less chance of under-utilizing the prototypes.

3.4.3 Experimental Results

The visual displays of the standard SOM and the three SSOMs trained with the Uniform-square data and plot of their topographic products are shown in Figure 3.10. The visual displays show, the simplified versions are at least as good as, if not better than, the usual SOM. This is also confirmed by the results obtained on topology preservation as presented in Table 3.1. In this respect SOM is quite robust with respect to lateral feedback function and choice of neighborhood.

Table 3.1 shows the quantitative measure of the topology preservation of the SOMs measured by topographic product P, topology violation V and rank correlation-based measure T for 4 training data sets. In all experiments a 6×6 SOM is used. The reported figures are averages of five runs with different initializations. For each data set the SOMs are trained for the same number of iterations with the same computational protocols.

Figure 3.11 shows the visual displays of standard SOM and three TSOMs trained with Y-data. For Y-data prototype placement is better for the SOMs with tree neighborhood. When the data have linear structure, tree neighborhood is expected to be better for prototype generation. The performances of the TSOMs for several input data sets with respect to topology preservation are summarized in the Table 3.2. Table 3.2 includes the results for standard SOM also for an ease of comparison. As expected, topology preservation is not good with tree neighborhood.

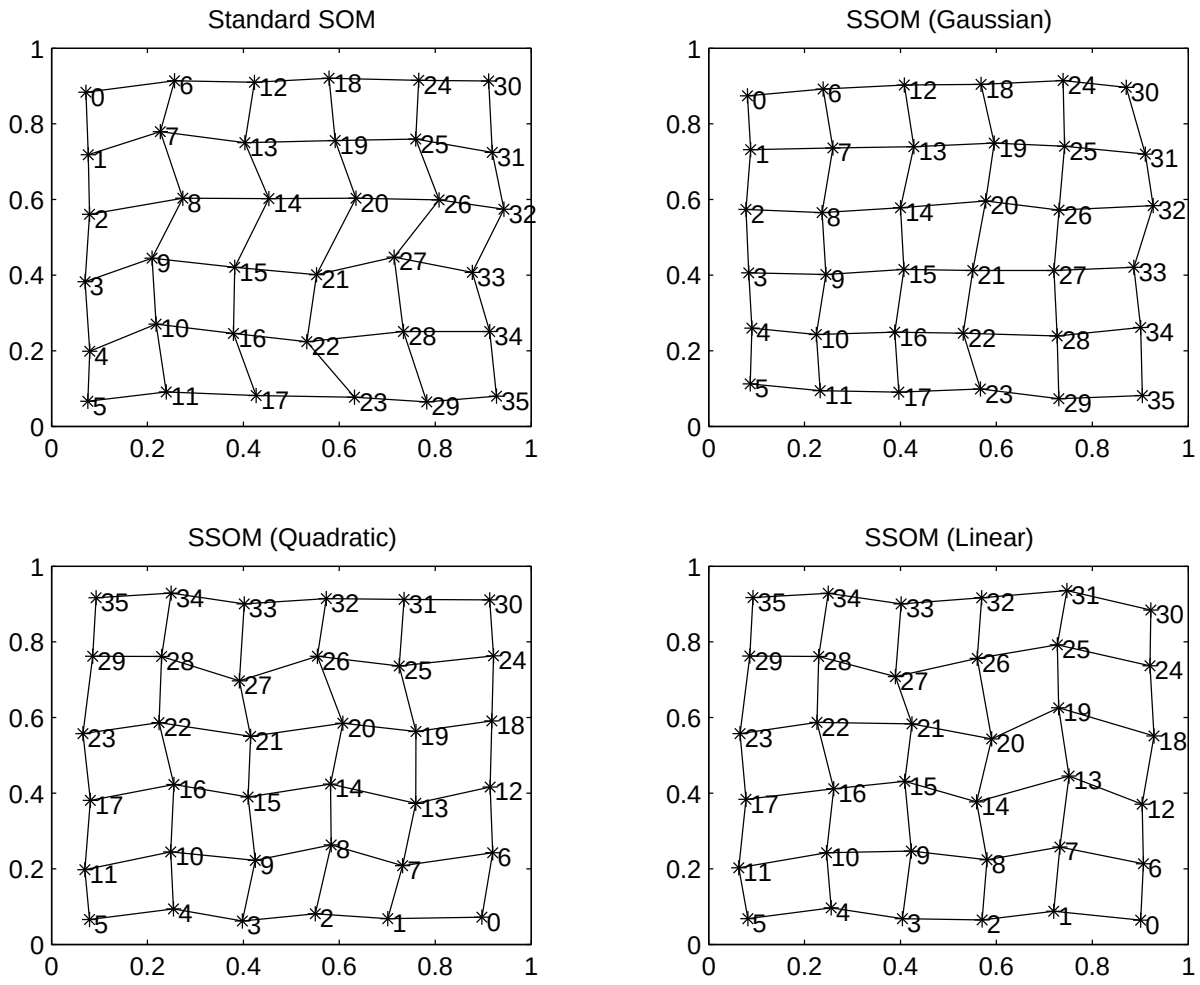

Figure 3.10: (a)Visual display of standard SOM trained with Uniform square data. (b),(c) and (d) are visual displays for SSOM(Gaussian), SSOM(Quadratic) and SSOM(Linear) respectively

SOM	Uniform Square			Two Spheres			Iris			Y-Data		
Type	P	V	T	P	V	T	P	V	T	P	V	T
SOM *Standard*	0.0031	0	0.3546	0.0096	0.36	0.2979	0.0287	2.31	0.2065	0.0294	5.50	0.1935
TSOM *Comp. MST*	0.0081	59.85	0.0265	0.0510	62.70	0.0232	0.0823	63.83	0.0286	0.0947	59.69	0.0261
TSOM *Rest. MST*	0.0078	20.36	0.0962	0.0273	22.52	0.1297	0.0425	24.10	0.1011	0.0453	16.91	0.1194
TSOM *Arb. Tree*	0.0141	74.83	0.0117	0.0576	88.21	0.0093	0.0718	75.65	0.0082	0.1108	86.81	0.0124

Table 3.2: Performance measure of the standard SOM and TSOMs with respect to topology preservation.

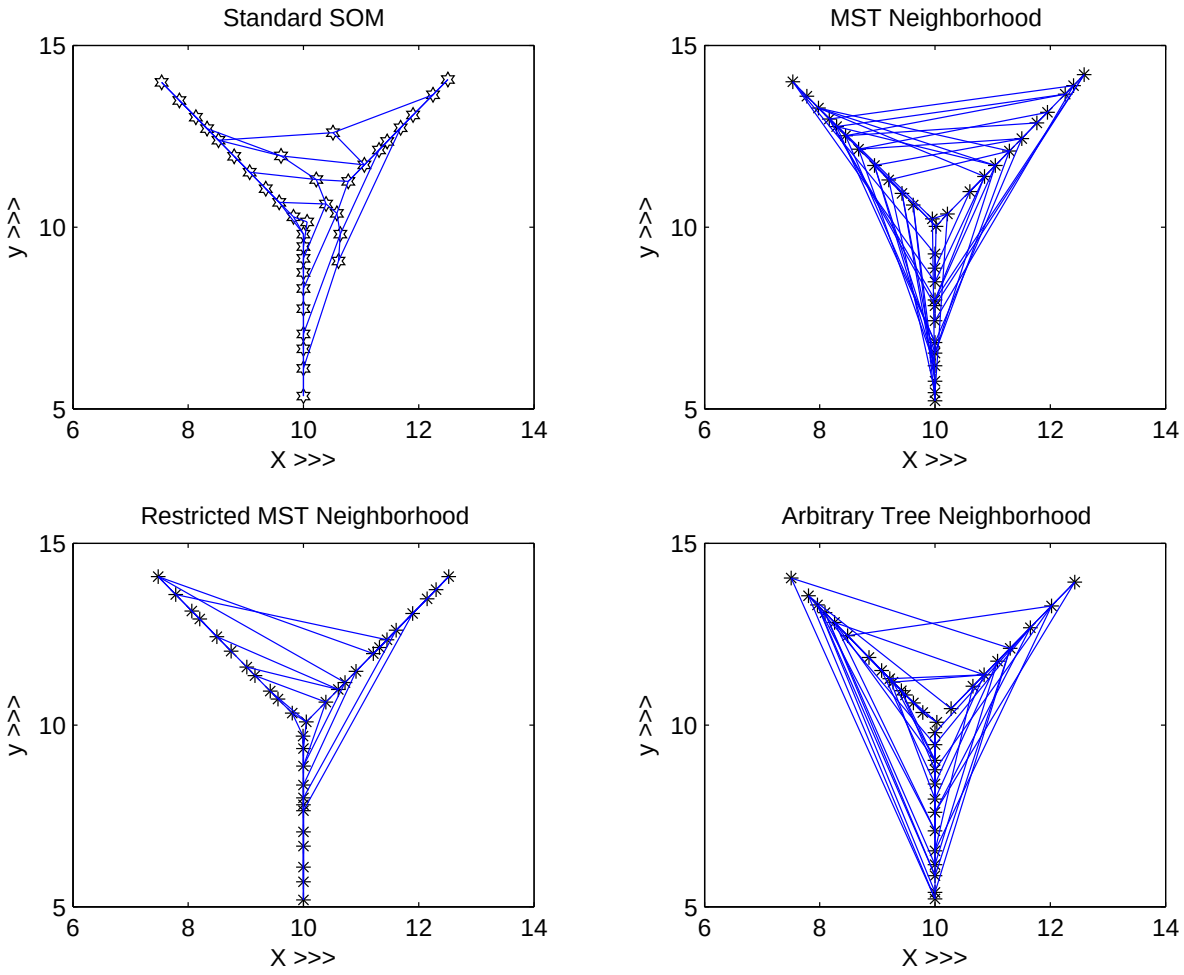

Figure 3.11: (a)Visual display of standard SOM trained with of Y-data. (b),(c) and (d) are visual displays for TSOMs using complete tree, restricted tree and arbitrary tree neighborhood respectively.

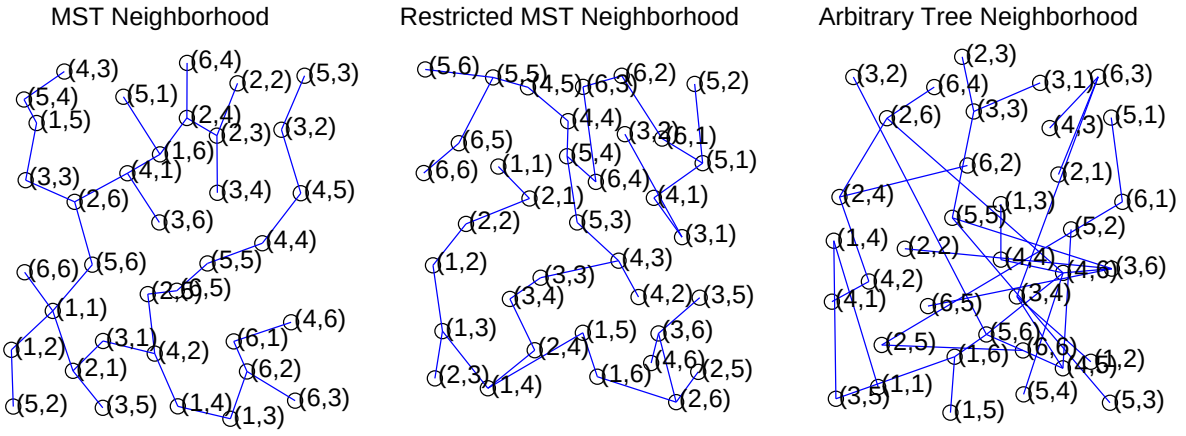

Figure 3.12: Visual display of the neighborhood function of three TSOMs for uniform square data. The nodes are marked with their lattice coordinates.

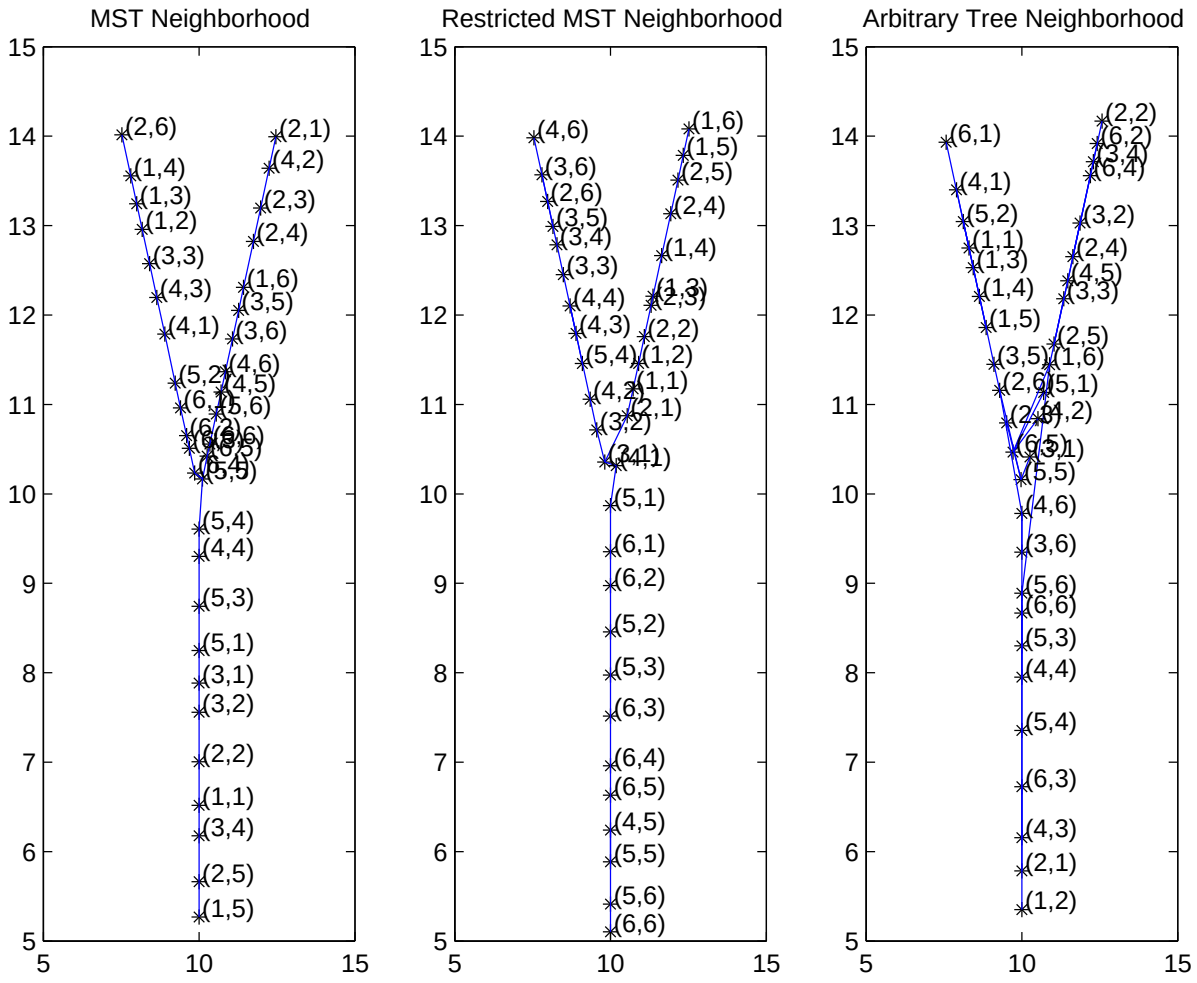

Figure 3.13: Visual display of the neighborhood function of three TSOMs for Y data. The nodes are marked with their lattice coordinates.

SOM type	Uniform Square	Two Spheres	Iris	Y-data
SOM *Standard*	47.19	476.77	12.46	12.32
TSOM *Comp. MST*	57.36	483.92	11.46	9.86
TSOM *Rest. MST*	57.50	514.11	11.90	9.80
TSOM *Arb. Tree*	58.77	490.86	11.32	10.65

Table 3.3: Total quantization errors for standard SOM and three TSOMs for four data sets

It can be seen from Figure 3.11 that for Y-data standard SOM places quite a few prototype in wrong positions. This is clearly due to incompatibility between the data distribution and the network topology. Three TSOMs exhibit better performances than the standard SOM for non-uniform data in terms of prototype placement but their performances degrades when topology preservation is considered. Among the TSOMs studied, the TSOM with RMST neighborhood shows the best performance in topology preservation. In the Figures 3.12 and 3.13 the tree neighborhoods of the trained TSOMs for the uniform square data and Y data are presented. In these figures the nodes are marked with their lattice coordinates. Visual observation of these figures also reveals that for the TSOM with RMST neighborhood there is a tendency of the nodes close on the lattice being placed closer in the tree neighborhood. This is due to the fact that the graph considered for constructing the RMST neighborhood takes into account the spatial neighborhood of the nodes in the output space. This is completely ignored in other two TSOMs.

Since the major goal of the TSOMs are to extract prototypes efficiently, their ability in this regard can be measured by computing the "total quantization error" of the prototypes. Let X_i be the set of training samples mapped onto the i-th node (prototype), i.e.,

$$X_i = \{\mathbf{x}_k \mid\mid \mathbf{w}_i - \mathbf{x}_k \mid\mid \leq \mid\mid \mathbf{w}_j - \mathbf{x}_k \mid\mid \forall j \neq i\},$$

where $\mathbf{w}_i$ is the weight vector of the i-th node. Then the quantization error due to node i is

$$QE_i = \sum_{\mathbf{x}_k \in X_i} \mid\mid \mathbf{w}_i - \mathbf{x}_k \mid\mid^2 .$$

Thus, for a $N \times N$ SOM the total quantization error is

$$TQE = \sum_{i=1}^{N^2} QE_i = \sum_{i=1}^{N^2} \sum_{\mathbf{x}_k \in X_i} \mid\mid \mathbf{w}_i - \mathbf{x}_k \mid\mid^2 .$$

The total quantization error of the TSOMs along with those of standard SOM for comparison are presented in Table 3.3. The values included in the table are the average

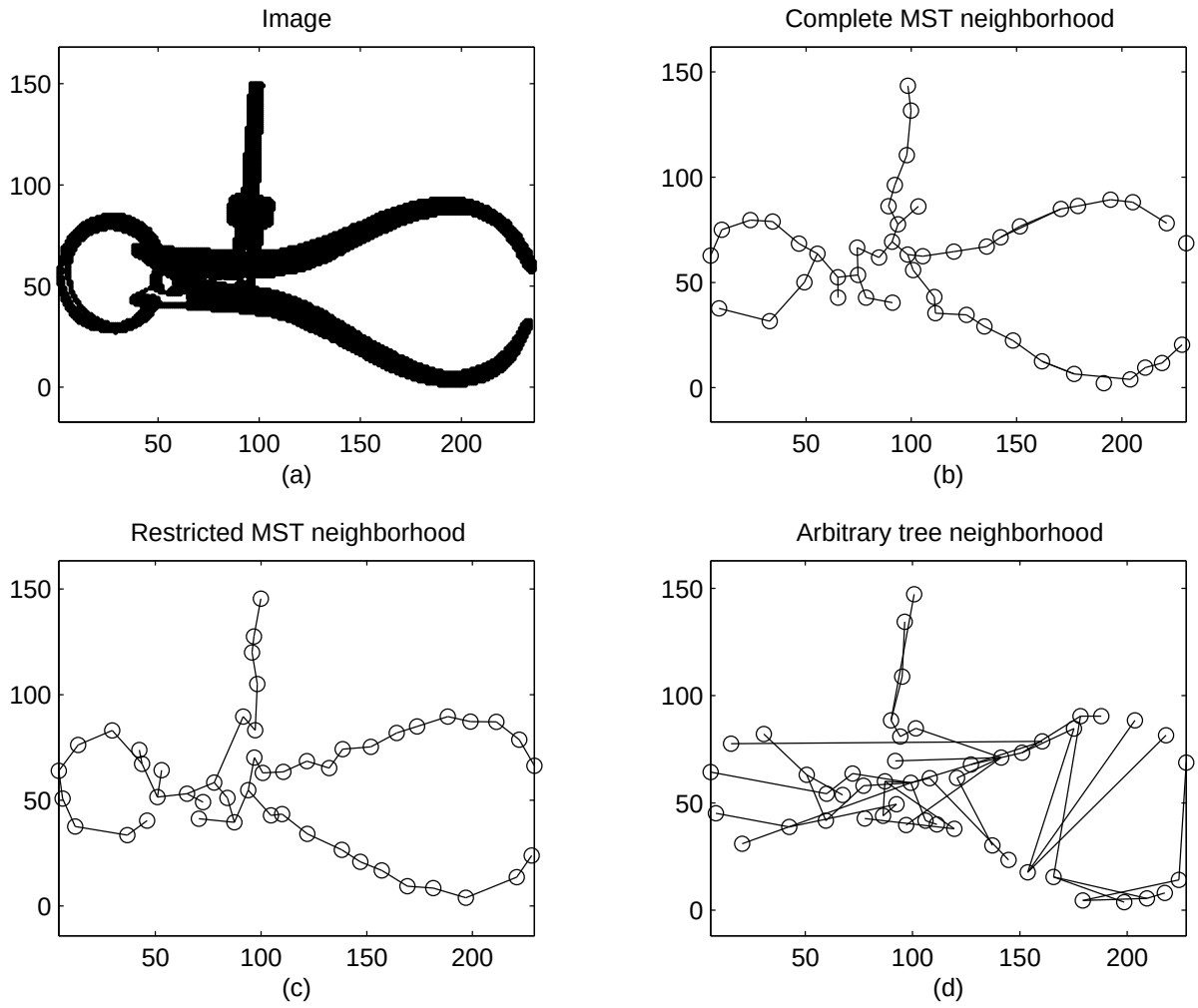

Figure 3.14: Result of skeletonization experimentations with 3 TSOMs.

of five SOMs for each type and each data set. It can be seen from the table that for Uniform Square the standard SOM has the least error, while with TSOMs the error increases by roughly 20%. This is expected, since for this data the SOM lattice structure matches very closely with the data distribution. For the Two Sphere data though the error for standard SOM is marginally lower, the errors for all four SOMs are fairly close with a mean about 490 and a $\pm 4\%$ variation. For the Iris data all three TSOMs have lower error with the TSOM with arbitrary tree neighborhood having the lowest error (nearly 10% lower than standard SOM). For the Y data also the TSOMs have lower error, with TSOM using restricted MST neighborhood scoring lowest value 9.8, more than 20% less than the standard SOM. Therefore, if the prototype extraction is the sole aim and the data distribution may not match the regular lattice structure of SOM, the SOMs with tree neighborhoods make good choices. Further, the TSOM with arbitrary tree neighborhood (ATN) can be a lucrative choice due to its lesser computational cost.

In Figure 3.14 the results of the skeletonization experiments with three TSOMs along

with the training image (Figure 3.14(a)) is presented. In the figures (b-d) the prototypes are connected according to their connectivity on the neighborhood tree at the end of training. It can be observed that both TSOMs with CMST and RMST neighborhoods perform an excellent skeletonization. The adjacency of the prototypes on the tree gives the connectivity information for further processing for higher level tasks such as recognition or indexing. However, for the TSOM with arbitrary tree neighborhood, though good placement of the prototypes is achieved, the tree represents the connectivity information very poorly. This is expected since the tree is built randomly and not altered during training. Thus adjuscent parts of the tree get tuned to the data region far away.

3.5 Conclusions

In this chapter we reported some empirical studies on properties of SOM. The studies mainly involve topology preservation property of SOM in various circumstances. In section 3.2 we introduced a quantitative measure of topology preservation property based on the concept of rank correlation. In section 3.3 we pointed out the underlying assumption about the existence of links between each pair of nodes in the network, so that neighborhood update can be achieved (as in a parallel hardware or a biological system). We studied the effect on topology preservation ability of the network due to different degrees of presence (link density) of the links. The experimental results suggest that when the links to nearer nodes are more likely to be present than those to far away nodes, even with a fairly low link density ($\approx 40\%$) the SOM preserves topology satisfactorily. This also supports the localized nature of SOM learning.In other words, if the local connections remain more or less undisturbed, the system performs satisfactorily. On the other hand, the experiments with all links having equal probability of being absent reveal that the performance of the SOMs degrade rapidly with link density. This is due to the randomness of the disturbance that affect the local structure much more severely. As a whole the study brings out several facet of SOM algorithm such as, localized learning, fault-tolerance to systematic disturbances, sensitivity to random disturbances etc. These findings may be exploited for designing of SOM hardware economically.

In section 3.4.1 we have studied the topology preservation properties of three variants of SOM with simplified lateral feedback functions, namely Gaussian without explicit neighborhood boundary, quadratic and linear. They are found to perform at par with the standard SOM which uses the Gaussian lateral feedback function with explicit neighborhood boundary. Such simplified lateral feedback functions hold much promise in simplifying the hardware implementation of SOM.

In section 3.4.2 we have studied variants of SOM with tree structured neighborhoods

for update. Two of the variants use minimal spanning trees to define the neighborhood of the nodes. The MSTs are calculated over the space of the weight vectors, using the inter-weight distance as the weight of the associated edge. In the first variant all possible pairs of nodes are considered, thus treating the SOM as a complete graph. We called this complete MST (CMST) neighborhood. In contrast, the other MST is computed over the graph with edges existing between nodes which are spatial neighbors on the SOM lattice. We named this as restricted MST (RMST). The third variant uses an arbitrary tree structure for neighborhood (ATN) definition. The results showed that the tree neighborhoods do not perform well in terms of topology preservation, while they show superior performance in prototype extraction from non-uniform data compared to the standard SOM. However, the SOMs with RMST neighborhood exhibit a significant level of topology preservation also. Further, for shape skeletonization tasks, it appears to be a computationally economic alternative to TSOM with CMST neighborhood. If prototype extraction is the only consideration, then the SOMs with ATN neighborhood can be quite useful due to their lower computational overhead.

Chapter 4

Extraction of Prototypes and Designing Classifiers[1]

[1]Parts of this chapter have been published in [203] and [266], the whole of it is published in [204].

4.1 Introduction

A classifier designed from a data set $X = \{\mathbf{x}_i \mid i = 1, ..., N, \mathbf{x}_i \in \Re^p\}$ can be defined as a function

$$\mathcal{D} : \Re^p \rightarrow N_c,$$

where $N_c = \{\mathbf{e}_i \mid i = 1, ..., c, \mathbf{e}_i \in \Re^c\}$ is the set of label vectors, p is the number of features and c is the number of classes. If $\mathcal{D}$ is a fuzzy classifier then $e_{ij} \geq 0$ and $\sum_{j=1}^{c} e_{ij} = 1$. If $\mathcal{D}$ is a crisp classifier, $\mathbf{e}_i$ is a basis vector with components $e_{ij} = 0 \ \ \forall \ i \neq j$ and $e_{ii} = 1$; consequently, here also $\sum_{j=1}^{c} e_{ij} = 1$. However, for a possibilistic classifier $\sum_{j=1}^{c} e_{ij} \leq c$ [32]. Designing a classifier involves finding a good $\mathcal{D}$. $\mathcal{D}$ may be specified parametrically, e.g., Bayes classifier [30], or nonparametrically e.g., nearest neighbor (NN) classifiers (crisp and fuzzy) [88, 32], nearest prototype (NP) classifiers (crisp or fuzzy) [88, 32], neural networks [37, 124] etc.

Although Bayes classifier is statistically optimal, it requires complete knowledge of prior probabilities $P_j; j = 1, 2, \cdots, c$ and class densities $p(\mathbf{x} \mid j); j = 1, 2, \cdots, c$, which is almost never possible in practical cases. Usually no knowledge of the underlying distribution is available except that it can be estimated from the samples.

In the present chapter two new approaches to design prototype-based classifiers are introduced. The problem of finding the required number of prototypes as well as the prototypes themselves are addressed together. A set of c (the number of classes) initial prototypes is generated from a set of labelled training data (without using the label information) using Kohonen's SOM algorithm [169]. Then the prototypes are labelled using the class information following a "*most likely class*" heuristic. The proposed algorithm, during the tuning stage evaluates the performance of the prototypes and depending on the evaluation result, prototypes are deleted, merged and split. This is continued until the performance of the prototypes stabilize. The procedure takes into account the intra-class and inter-class distribution of the training data and tries to produce an "optimal" set of prototypes. This set of prototypes is then used to design a *nearest prototype* classifier. This is our first classifier. To keep parity with the next classifier designed here we call it "1-Nearest Multiple Prototype (1-NMP) classifier". 1-NMP classifier has some limitations as will be demonstrated later. To address these limitations the set of prototypes is further processed through a fine-tuning stage. During this stage, a *zone of influence* for each prototype is defined. A "similarity measure" and an error function are also defined. Then the position of each prototype and its zone of influence are modified iteratively in order to minimize the error function. Any unknown data point is then classified according to its maximum similarity with the prototypes. We call this classifier "1-Most Similar Prototype (1-MSP) classifier". Both classifiers are tested with several data sets and compared with some benchmark results as well as k-NN classifiers

and support vector machine (SVM) classifiers.

4.2 Generation of a good Set of Prototypes

Prototype-based classifiers are conceptually simple, easy to understand and computationally efficient. Evidently, their performance depends on the quality of the prototypes, i.e., the degree to which the inter-class as well as intra-class distributions of the data are captured by the prototypes used. Overall, while designing a prototype-based classifier one faces three fundamental issues.

1. How to generate the prototypes?
2. How many prototypes are to be generated?
3. How to use the prototypes for designing the classifier?

Depending on the schemes adopted to address these questions, there are a large number of classifiers. To illustrate the point we discuss the simplest of them, the nearest-prototype classifiers with one prototype per class. The number of prototypes is equal to the number of classes in the training data, i.e., $\hat{c} = c$. In this kind of classifier the prototype for a class is usually generated by taking the mean of the training data vectors from that class. The third issue is commonly addressed using a distance function δ. A vector $\mathbf{x} \in \Re^p$ is classified using the prototype set $V = \{\mathbf{v}_i, \mathbf{l}_i \mid i = 1, ...c, \mathbf{v}_i \in \Re^p, \mathbf{l}_i \in N_c\}$, where c is the number of classes and $\mathbf{l}_i$ is the label vector associated with $\mathbf{v}_i$, as follows.

$$\begin{aligned} \text{Decide} \quad & \mathbf{x} \in \text{class } i \\ \Leftrightarrow \quad & \mathcal{D}_{V,\delta}(\mathbf{x}) = \mathbf{l}_i \\ \Leftrightarrow \quad & \delta(\mathbf{x}, \mathbf{v}_i) \leq \delta(\mathbf{x}, \mathbf{v}_j) \ \forall \ j \neq i. \end{aligned}$$

This simple scheme and its variants work quite well for many problems. However, such a classifier has been proved inadequate if the data from one class are distributed into more than one cluster or if the data from two different classes cannot be separated by a single hyperplane, as demonstrated in the famous "XOR" data [371]. Therefore, a generalized classifier design scheme must keep provision for *multiple prototypes* for a class.

If multiple prototypes are used for a class, the three issues concerning the prototypes become more difficult to address, but in this case a lot of sophisticated techniques become

available for dealing with them. Usually the three issues can be addressed independently, but in some cases the strategy used for answering one issue may depend on the strategy used for dealing with others or a single strategy may take care of more than one issue.

We now briefly review some commonly used schemes for addressing these issues. The optimal number of prototypes required depends on both the inter-class as well as the intra-class distributions of the data. One can use suggestions of a domain expert, or some clustering algorithm to find cluster centroids in the training data set with enough prototypes to represent each class. However, most of the clustering algorithms (e.g. c-means algorithm) require the of number clusters to be supplied externally or to be determined using some cluster validity index . The k-NN algorithms [88] use each data point in the training data set as a prototype.

Given the number of prototypes there are many procedures for generating them; for example, clustering algorithms like c-means [88], fuzzy c-means [31], mountain clustering method [355], etc. Other approaches include learning vector quantization (LVQ) methods, neural network based methods such as Kohonen's SOM, random search methods, gradient search methods etc. Each method has its own advantages and limitations. Often more than one method are used together for producing a good set of prototypes.

Once the prototypes are generated, there are several ways of using the prototypes in the classifier. To mention a few, in a nearest prototype classifier a data point is assigned the class label of the prototype closest to it. In a fuzzy rule-based classifier, each prototype can be used to generate a fuzzy rule to define a fuzzy rule base [137]. This rule base is then used for classifying the data.

Here we develop a scheme for generation of a set of prototypes using Kohonen's Self-Organizing Map (SOM) algorithm [169]. The scheme provides an integrated approach to the solution of first two of the three issues mentioned above. Once a good set of prototypes is obtained, we present two approaches to classifier design using the prototypes.

4.2.1 The SOM-based prototype generation scheme

The prototype generation method developed here uses SOM algorithm as its backbone. SOM algorithm is used here in to find a good set of prototypes which is adequate for the classification task. As customary with approaches to classifier design, a set of labelled training data is used for designing purposes. First a SOM is trained using the feature vectors (not the labels) of the training data. The weight vectors of the trained SOM are treated as the initial set of unlabelled prototypes/cluster centers. Use of SOM for clustering is advantageous in several ways. In most of the clustering algorithms such as c-means the resulting clusters sensitive to initialization. That may often result in

under-utilization of the prototypes. The algorithm can end up with several prototypes representing no data point (i.e., due to initialization condition they were not updated subsequently). Further, existence of outlier points may cause the cluster centers to get stuck to them.

In SOM under-utilization of nodes is unlikely to happen, since even if some node is initialized outside the data, due to neighborhood update they will eventually move into the region of feature space containing the input data. Also due to the same neighborhood update feature a node near an outlier is eventually likely to move towards a dense region. This makes SOM algorithm relatively insensitive to noise.

In clustering algorithms, the aim is to find substructures in the data. However, when we want to use the cluster centers as the prototypes for classification purpose, it may not be adequate to find them with unsupervised method only. In general, finding cluster centers and assigning them class labels do not result in a good set of prototypes for classifier design. The distribution of the classes may not form distinct clusters, classes may be very close together and/or overlapping. Also it may not be possible to represent a class with a single prototype. Usually deciding on an adequate number of prototypes for classification is a formidable problem.

For computational efficiency it is desirable to have a small number of prototypes for each class. The number of prototypes should be just enough to capture the class-distributional property of the training data. Here we develop a user-friendly scheme for extraction of prototypes. In this method the user need not *guess* or go through *multiple trials* to determine the number of prototypes needed, the algorithm determines the number of prototypes automatically and generates them.

In the second stage, the proposed method uses the initial unlabelled prototypes from the SOM and the training data along with the class labels. First the initial prototypes are tentatively labelled using a majority voting scheme. Then the nearest-prototype classification performance of the prototypes are evaluated in each iteration. Depending on their performance, the prototypes may go through one or more of the following operation, *deletion*, *merge*, *split* and *modification* to generate a new set of prototypes. Then the new set of prototypes are again fine tuned using SOM algorithm. Iterations continue till the error rate falls to a satisfactory level. This drastically reduces the number of parameters to be decided by the user. One needs to specify only *two* parameters regarding the acceptability of a prototype, which basically embodies the intuitive concept of the user regarding quality of prototypes, namely *representativeness*, i.e., a prototype should represent enough number of training data and *purity*, i.e, it should predominantly represent data from a single class.

Though the user need not guess the final number of prototypes, one has to decide

upon the number of nodes in the first stage SOM. There are two very easy guidelines to that. First, one can start with a reasonably large SOM, the redundant nodes will be removed in the second stage. In the second approach one can start with minimal number of nodes, which is equal to the number of classes. If the class structure corresponds to the cluster structure, then they will eventually become final prototypes, otherwise in the second stage additional prototypes will be generated. Clearly the second option is computationally more efficient and does not need the user to exercise any specific choice for the required number of clusters. We follow this approach in this chapter.

4.2.2 Generation and labelling of initial set of prototypes

In this work we use a 1-D SOM, but the algorithm can be extended to 2-D SOM also. First we train a one-dimensional SOM using the training data. Although, the class information is available, SOM training does not use it. The number of nodes in the SOM is the same as the number of classes c. This choice is inspired by the fact that the *minimum number of prototypes that are required is equal to the number of classes*. At the end of the training the weight vector distribution of the SOM reflects the distribution of the input data. These unlabelled prototypes are then labelled using class information. For each of N input feature vectors we identify the prototype closest to it, i.e., the winner node. Since no class information is used during the training, some prototypes may become the winner for data from more than one classes, particularly when the classes overlap or are very close to each other. For each prototype $\mathbf{v}_i$ we compute a score D_{ij}, which is the number of data points from class j to which $\mathbf{v}_i$ is the closest prototype. The class label C_i of the prototype $\mathbf{v}_i$ is determined as

$$C_i = \underbrace{arg\max}_{j} D_{ij}. \tag{4.1}$$

This scheme will assign a label to each of the c prototypes, but such a set of prototypes may not classify the data satisfactorily. For example, from Eq. (4.1) it is clear that $\sum_{j \neq C_i} D_{ij}$ data points will be wrongly classified by the prototype $\mathbf{v}_i$. Hence we need further refinement of the initial set of prototypes $V_0 = \{\mathbf{v}_{10}, \mathbf{v}_{20}, ..., \mathbf{v}_{c0}\} \subset \Re^p$.

4.3 Generating a better set of prototypes

The prototypes generated by the SOM algorithm represent the overall distribution of the input data. A set of prototypes useful for classification job must be capable of dealing with class specific characteristics (such as class boundaries) of the data. In this section

we present a DYNAmic prototype GENeration algorithm (DYNAGEN) for generating a set of prototypes that is a better representative of the distribution of training data and takes into account the class specific characteristics. The strategy involves a modification procedure starting with the initial set of prototypes V_0. In each iteration the prototype set is used in a nearest-prototype classifier and the classification performance is observed. Depending on the performance of the individual prototype in the classifier, prototypes are modified, merged, split and deleted, leading to a new set of prototypes. This process of modification is repeated until the number of prototypes and their performance stabilize within an acceptable level. The final set of prototypes, when used to design prototype-based classifiers is expected to enhance the performance of the classifier. We use the term 'performance of a prototype' to indicate the 'performance of the prototype when used in a nearest-prototype classifier'.

In the t-th iteration, the prototype set V_{t-1} from previous iteration is used to generate the new set of prototypes V_t. The labelled set of prototypes V_{t-1} is used to classify a set of training data. Let W_i be the number of training data to which prototype $\mathbf{v}_i$ is the closest one (winner). Let $S_i = \max_j \{D_{ij}\} = D_{iC_i}$. Thus, when $\mathbf{v}_i$ is labelled as a prototype for class C_i, S_i training data points will be correctly classified by $\mathbf{v}_i$ and $F_i = \sum_{j \neq C_i} D_{ij}$ data points will be incorrectly classified. Consequently, $W_i = S_i + F_i$ and $W_i = \sum_j D_{ij}$.

In the training data $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, let there be N_j points from class j. The refinement stage uses just *two* parameters K_1 and K_2 to dynamically generate $c+1$ retention thresholds known as *global retention* threshold α and a set of *class-wise retention* threshold β_k (one for each class), to evaluate the performance of each prototype. The thresholds α and β_k are computed dynamically (not fixed) for the t-th iteration using the following formulae:

$$\alpha_t = \frac{1}{K_1 \mid V_{t-1} \mid} \tag{4.2}$$

and

$$\beta_{kt} = \frac{1}{K_2 \mid V_{kt}^* \mid}, \tag{4.3}$$

where $V_{kt}^* = \{\mathbf{v}_i \mid \mathbf{v}_i \in V_{t-1}, C_i = k\}$. *We emphasize that the algorithm uses just two (not $c+1$) user supplied parameters.*

Based on the classification performance of the prototypes, different operations are performed to generate a new set of prototypes. In the following first we define these operations, later while describing the algorithm we state when to use them.

4.3.1 The operations

Merging of a prototype with respect to a class: Let a prototype $\mathbf{v}_i$ represent D_{ik} training data point from class k. To merge $\mathbf{v}_i$ w.r.t. class k, we identify the prototype $\mathbf{v}_l$ closest to $\mathbf{v}_i$ where $C_l = k$ (i.e., $\mathbf{v}_l$ is a prototype for class k). Let us denote X_{ij} as the set of training data vectors from class j whose nearest prototype is $\mathbf{v}_i$. When we merge $\mathbf{v}_i$ with $\mathbf{v}_l$ w.r.t. class k, $\mathbf{v}_l$ is updated according to the equation,

$$\mathbf{v}_l = \frac{W_l \mathbf{v}_l + \sum_{\mathbf{x} \in X_{ik}} \mathbf{x}}{W_l + D_{ik}}. \tag{4.4}$$

Note that, we do not say here when to merge. This will be discussed later.

Modifying a labelled prototype: A prototype $\mathbf{v}_i$ is modified according to the following equation,

$$\mathbf{v}_i = \frac{\sum_{\mathbf{x} \in X_{iC_i}} \mathbf{x}}{D_{iC_i}}. \tag{4.5}$$

Splitting a prototype: If $\mathbf{v}_i$ represent data from r classes, then the prototype $\mathbf{v}_i$ is split into r new prototypes for r different classes according to Eq. (4.6).

$$\mathbf{v}_l = \frac{\sum_{\mathbf{x} \in X_{iC_l}} \mathbf{x}}{D_{iC_l}}. \tag{4.6}$$

The subscript l in Eq. (4.6) is used to indicate one of r classes represented by $\mathbf{v}_i$. The prototype $\mathbf{v}_i$ is deleted. So after the splitting, the number of prototypes is increased by $r - 1$.

Deleting a prototype: A prototype, say $\mathbf{v}_i$, is removed from the set of prototypes so that number of prototypes is reduced by one.

4.3.2 The DYNAmic prototype GENeration (DYNAGEN) algorithm

Now we are in a position to describe the DYNAGEN algorithm for dynamic generation and enhancement of the prototypes.

Algorithm DYNAGEN:

- Repeat for all $\mathbf{v}_i \in V_{t-1}$ until the termination condition is satisfied:

- 1. If $W_i \neq D_{iC_i}$ and $W_i < \alpha_t N$ and there is at least another prototype for class C_i then delete $\mathbf{v}_i$. (Global deletion).

     Comment: *If a prototype is not a pure one (i.e., it represents data from more than one class) and does not represent a reasonable number of points, it fails to qualify to become a prototype. However, if there is no other prototype for class C_i the prototype is retained.*

  2. Else if $W_i > \alpha_t N$ but $D_{ij} < \beta_{jt} N_j$ for all classes then merge $\mathbf{v}_i$ for the classes for which $D_{ij} > 0$ using Eq. (4.4) and delete $\mathbf{v}_i$. (Merge and delete)

     Comment: *The prototype represents a reasonable number of points, but does not represent an adequate number of points from any particular class, so it cannot qualify as a prototype for any particular class. But we cannot ignore the prototype completely. We logically first split $\mathbf{v}_i$ into s prototypes $\mathbf{v}_{i1}, \mathbf{v}_{i2}, ...\mathbf{v}_{is}, s \leq c$, s is the total number of classes for which $D_{ij} > 0$, and then merge $\mathbf{v}_{ij}$ to its closest prototype from class j. $\mathbf{v}_i$ is then deleted.*

  3. Else if $W_i > \alpha_t N$ and $D_{iC_i} > \beta_{C_i t} N_{C_i}$but $D_{ij} < \beta_{jt} N_j$ for all $j \neq C_i$ then merge $\mathbf{v}_i$ with respect to all the classes other than C_i for which $D_{ij} > 0$ using Eq. (4.4) and modify $\mathbf{v}_i$ using Eq. (4.5). (Merge and modify)

     Comment: *The prototype represents points from more than one class; however, the points from only one class are well represented by the prototype. According to our labelling scheme the prototype is labelled with the most represented class. Thus, we merge $\mathbf{v}_i$ with respect to the classes other than C_i using Eq. (4.4) and then modify $\mathbf{v}_i$ by Eq. (4.5).*

  4. Else if $W_i > \alpha_t N$ and $D_{ij} > \beta_{jt} N_j$ for more than one class then merge $\mathbf{v}_i$ w.r.t. classes for which $D_{ij} < \beta_{jt} N_j$ by Eq. (4.4) and split $\mathbf{v}_i$ into new prototypes for the classes for which $D_{ij} > \beta_{jt} N_j$ by Eq. (4.6). Add these new prototypes to the new set of prototypes V_t. (Merge and split)

     Comment: *The prototype represents reasonably well number of points from more than one class. So we merge the prototype with respect to the classes whose data are not well represented and split the prototype into one for each class whose data are reasonably well represented by $\mathbf{v}_i$.*

- Let V_t be the union of all unaltered prototypes of V_{t-1} and the modified as well as new prototypes.

- Run the SOM algorithm on V_t with *winner-only update* (i.e., no neighbor is updated) strategy using the same training data as input.

  Comment:*At this stage we want only to fine tune the prototypes. If the neighbors are also updated the prototypes again might migrate to represent points from more than one class.*

- Repeat the procedure with the new set of prototypes V_t if any of the termination conditions is not met.

- **Termination Conditions**

  The algorithm terminates under any one of the following two conditions.

  1. Achievement of a satisfactory recognition score defined in terms of percentage of correct classifications (ϵ).
  2. A specified number of iterations (I_{max}) is reached.

     Comment:*Proper use of condition (1) requires some knowledge of the data. However, even if we do not have the same, we can always set a high (conservative) percentage for (ϵ), say 95%.*
     Condition (2) is used to protect against infinite looping of the algorithm for some data with highly complex structures for which the chosen values of ϵ may not be reachable.

We now design the nearest prototype classifier, which we call 1-NMP classifier, using the set of prototypes generated by DYNAGEN.

4.3.3 Results of 1-NMP classifier

We report the performance of the classifier for ten data sets. The data sets are divided into two groups A and B. Group A consists of 6 data sets **Iris, Glass, Breast Cancer, Vowel, Norm4**, and **Two-Dishes**; while group B contains **Cone-torus, Normal Mixture, Satimage1** and **Phoneme**. For group A data sets some results are available in the literature but the details of the experimental protocols (such as training-test division, learning parameters etc.) used are not available. Hence, we have randomly divided each data set in group A into two (approximately) equal subsets. We then use these subsets alternately as training and test sets and report the performance. For the group B data sets many benchmark results for different classifiers are documented in [191] along with computational protocols. We have used the same partitions as reported in [191] and have compared our results with those reported in [191].

Table 4.1: Distribution of data points among different classes for group A data sets

Class	Iris	Glass	B. cancer	Vowel	Norm4	Two-Dishes
	Training Set					
1	25	35	178	36	100	250
2	25	38	106	44	100	500
3	25	8		86	100	
4		6		75	100	
5		4		103		
6		14		90		
Total	75	105	284	434	400	750
	Test Set					
1	25	35	179	36	100	250
2	25	38	106	45	100	500
3	25	9		86	100	
4		7		76	100	
5		5		104		
6		15		90		
Total	75	109	285	437	400	750

Iris data [11] have 150 points in 4-dimension that are from 3 classes, each with 50 points. *Glass* data [129] consist of 214 samples with 9 attributes from 6 classes. *Breast Cancer* [237] have 569 points in 30-dimension from two classes. A normalized version of the Breast Cancer data is generated by dividing each component by the largest value of that component. The *Vowel* [268] data set consists of 871 instances of discrete phonetically balanced speech samples for the Telugu vowels in consonant-vowel nucleus-consonant (CNC) form. It is a 3-dimensional data containing the first three formant frequencies. The 871 data points are unevenly distributed over 6 classes. The data set *Norm4* [262] is a sample from a mixture of four normals in 4-dimension. It has 800 points, with 200 points each from the four components. The *Two-Dishes* is a synthetically generated data set consisting of 1500 2-D data points distributed uniformly over two well-separated dishes of different radii. The scatterplot of this data set is shown in Figure 4.3. Each dish has 750 points. The distribution of the data points among different classes for group A data sets is shown in Table 4.1. Table 4.1 reveals that for Glass, Breast Cancer and Vowel data sets there are large variations in the distribution of data points in different classes.

In group B the *Cone-Torus* data set has 201 and 199 2-D points in the training and test sets respectively [191, 1]. There are three classes each representing a different shape,

Table 4.2: Distribution of data points among different classes for group B data sets

Class	Cone-torus	Normal Mixture	Satimage1	Phoneme
	Training Set			
1	46	125	104	354
2	50	125	68	146
3	105		108	
4			47	
5			58	
6			115	
Total	201	250	500	500
	Test Set			
1	46	500	1429	3464
2	49	500	635	1440
3	104		1250	
4			579	
5			649	
6			1393	
Total	199	1000	5935	4904

a cone, a half torus and a Gaussian shape. In the *Normal Mixture* data each of the two classes comes from a 2-D normal distribution [2, 191]. The training set contains 250 points and the test set contains 1000 points. The *Satimage1* data set is generated from a Landsat Multi-Spectral Scanner image [191, 108]. The present data set covers an area of 82×100 pixels. The data set, as available from [108] has 36 dimensions containing the gray values of 9 (a 3×3 neighborhood) pixels captured by 4 sensors operating in different spectral regions. However, adhering to the protocol followed in [191], we also use the data for the central pixel only, thus reducing the dimension to four. The data set has 6 classes representing different kinds of ground covers. The training set has 500 points and the test set has 5935 points. The *Phoneme* data set contains vowels coming from 1809 isolated syllables (for example: pa, ta, pan, ...) in French and Spanish languages [191, 108]. Five different attributes are chosen to characterize each vowel. They are the amplitudes of the five first harmonics AHi, normalized by the total energy Ene (integrated on all frequencies), AHi/Ene. Each harmonic is signed positive when it corresponds to a local maximum of the spectrum. Otherwise, it is signed negative. The Phoneme data set has two classes, nasal and oral. The training set contains 500 points and the test set contains 4904 points. The class-wise distribution of the data points for group B data sets is presented in Table 4.2.

Table 4.3: Performance of the 1-NMP classifier for the group A data sets.

Data Set	Size		No. of prototypes		% of Error		Average Test Error
	Trng.	Test	Initial	Final	Trng.	Test	
Iris	75	75	3	5	2.66%	8.0%	
	75	75	3	5	4.0%	5.33%	6.66%
Glass	105	109	6	28	16.19%	34.86%	
	109	105	6	26	19.26%	33.33%	34.09%
Breast	284	285	2	4	10.91%	11.58%	
Cancer	285	284	2	5	9.12%	12.67%	12.13%
Normalized	284	285	2	8	8.8%	5.61%	
Breast Cancer	285	284	2	4	6.66%	10.56%	8.08%
Vowel	434	437	6	21	18.4%	19.22%	
	437	434	6	15	20.82%	22.35%	20.78%
Norm4	400	400	4	4	4.25%	3.5%	
	400	400	4	4	4.0%	4.0%	3.75%
Two-Dishes	750	750	2	2	6.13%	5.33%	
	750	750	2	2	5.6%	6.53%	5.93%

Tables 4.3 and 4.4 summarize the classification performances of 1-NMP classifier for group A and group B data sets respectively. We used the values $K_1 = 3, K_2 = 6, \epsilon = 95\%$ and $I_{max} = 10$ for all data sets. The % of error column shows the percentage of misclassification for the training and the test data.

It is well known that classes 2 and 3 of Iris have some overlap and the typical re-substitution error with a nearest prototype classifier using 3 prototypes obtained by some clustering algorithm is about 10% error [32]. The average test performance of the proposed system with 5 prototypes is quite good resulting only 6.66% error.

Glass data shows a high percentage of error; this is possibly unavoidable, because a

Table 4.4: Performance of the 1-NMP classifier for the group B data sets.

Data Set	Size		No. of prototypes		% of Error	
	Trng.	Test	Initial	Final	Trng.	Test
Cone-Torus	201	199	3	12	18.25%	22.5%
Normal Mixture	250	1000	2	4	14.0%	9.5%
Satimage1	500	5935	7	27	13.6%	15.65%
Phoneme	500	4904	2	5	21.8%	21.31%

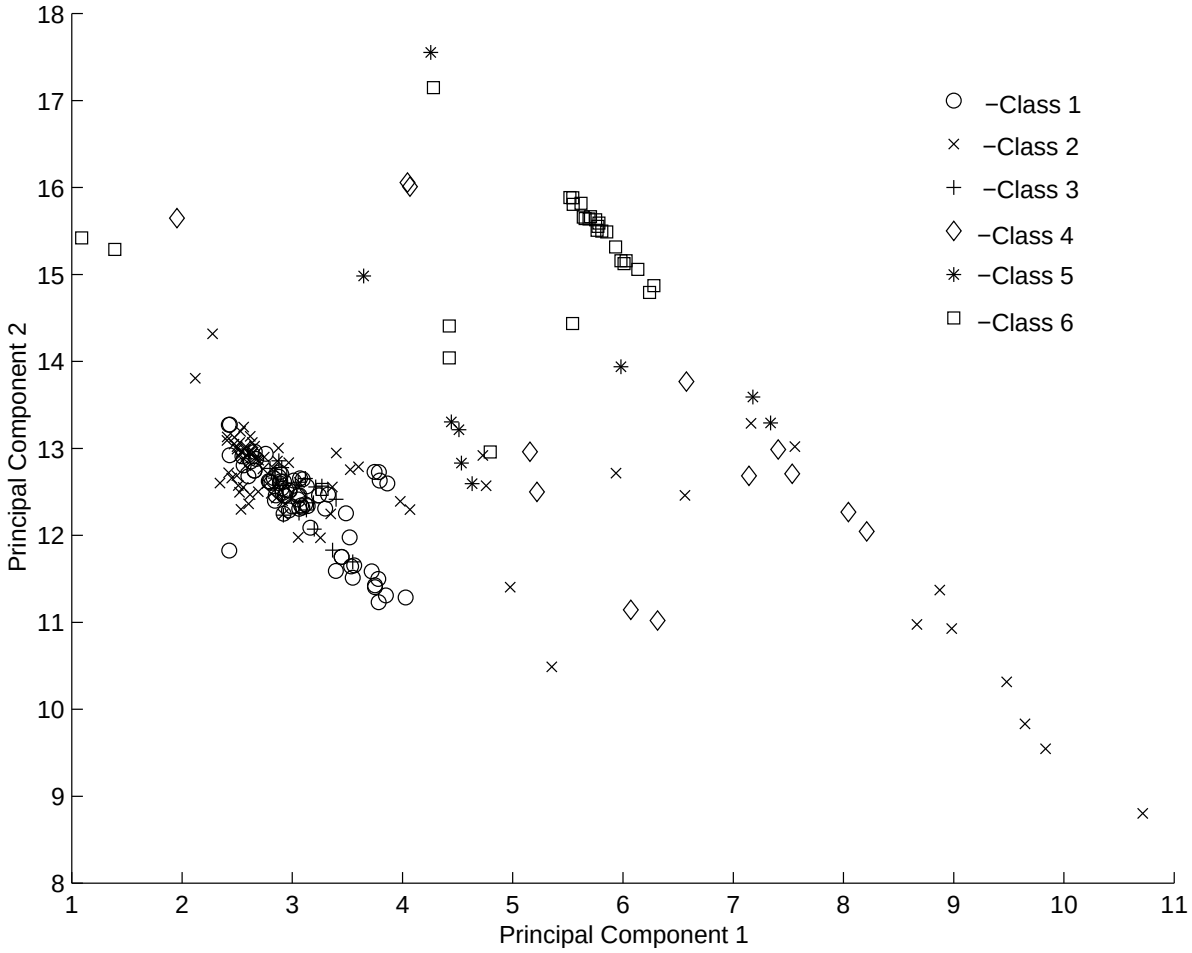

Figure 4.1: Scatterplot of the Glass data along two most significant principal components.

scatterplot (Figure 4.1) of the first two principal components shows that the data for class 3 are almost randomly distributed among the data points from other classes. In fact, the points from class 3 (represented by +) are not visible in the scatterplot. In [137] the recognition score reported for the Glass data is 64.4%, i.e., about 35% error. Our classifier could realize more than 66% accuracy with 26 prototypes in the best test result and the average test error is 34.09%.

Breast Cancer data have been used in [237] to train a linear programming-based diagnostic system by a variant of multisurface method (MSM) called MSM-Tree and about 97.5% accuracy was obtained. Breast cancer data of a similar kind have also been used in a recent study [137] with 74.0% accuracy with **100** rules. Our classifier could achieve as low as 12.13% average test error with only **4-5** prototypes and it is quite good. With the normalized Breast Cancer data the 1-NMP classifier exhibits a much better performance.

Although the Vowel data set has three features, like other authors [268] we used only the first two features. Bayes classifier for this data set [269] gives an overall recognition score of 79.2%. Figure 4.2, the scatterplot of Vowel data (different classes are represented

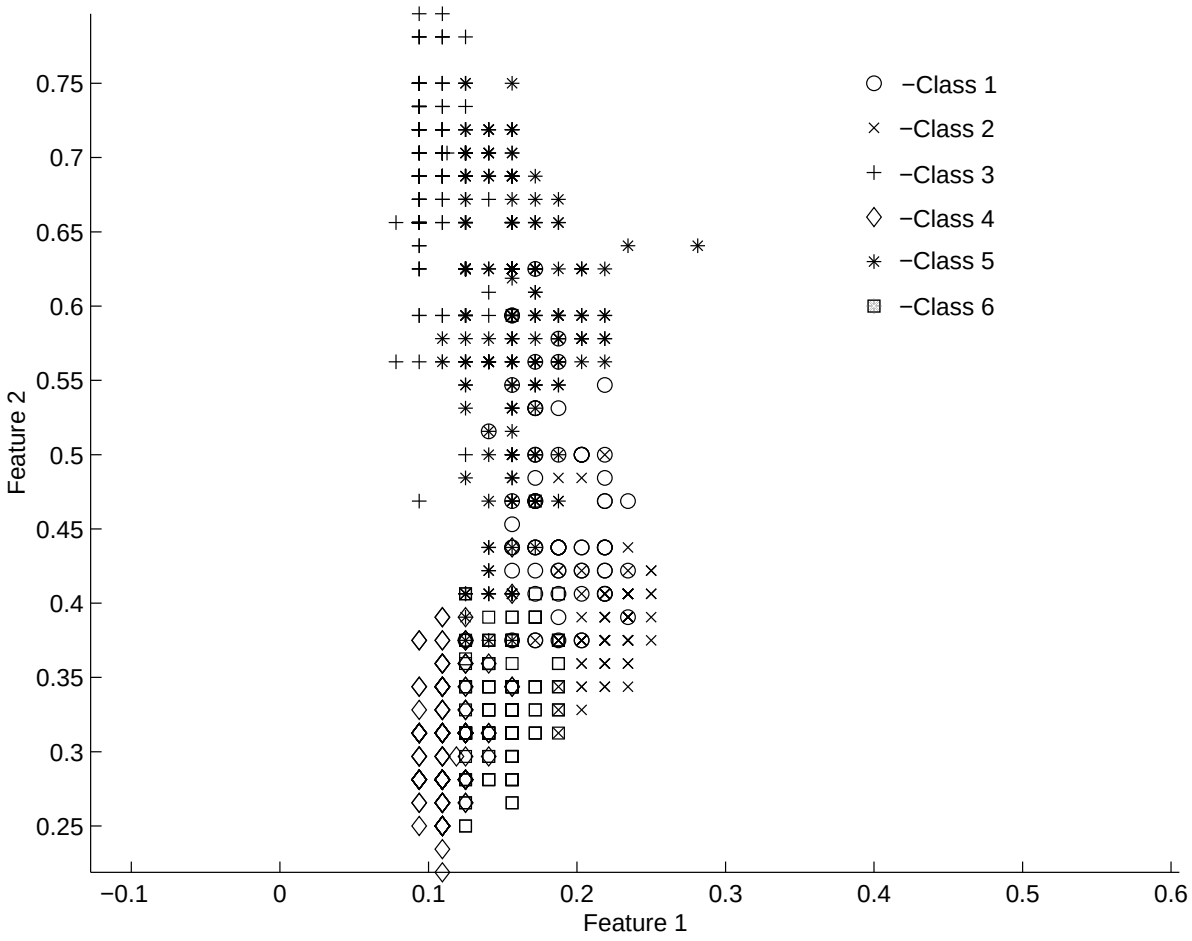

Figure 4.2: Scatterplot of first two features of the Vowel data.

by different symbols) depicts that there are substantial overlaps among different classes and hence some misclassification is unavoidable. The proposed classifier could achieve an average of 79.22% correct classification.

The performance on Norm4 [262] with only 4 prototypes, i.e., one prototype per class is quite good. For this relatively simple data set the DYNAGEN based classifier could achieve up to 96% accuracy with only 4 prototypes.

The Two-Dishes data is a synthetic one generated for investigating the limitations of the 1-NMP classifier. Although the classes are linearly separable, the 1-NMP classifier achieves an accuracy of 96% with two prototypes only. Our tuning algorithm could have produced more prototypes and thereby could reduce the error. But we used only two prototypes to demonstrate the effectiveness of our next classifier described in the following section.

Table 4.4 shows that for all of the group B data sets, the 1-NMP classifier generalizes quite well. For two data sets the performance on the test sets is little better than that on the training sets while for the other two cases the performance on the training sets are little better.

4.3.4 Properties of 1-NMP classifiers

It is evident from the results obtained that the 1-NMP classifiers designed with the prototypes generated by the DYNAGEN algorithm are quite efficient. As with any prototype based classifier, the performance of these classifiers depend on the quality of the prototype set used, i.e., how faithfully the distribution of the data is represented by the set of prototypes. In our case the DYNAGEN algorithm is ultimately responsible for the quality of the prototype set. On closer examination of the DYNAGEN algorithm, one can find following general tendencies acting on the prototypes.

- The DYNAGEN algorithm tries to generate a small number of prototypes while at the same time it tries to keep the error rate as low as possible. This is achieved by means of two opposing actions: (1) splitting of prototypes and (2) deletion of prototypes. The former leads to an increase in the number of prototypes, with a view to ensuring that class boundaries are taken care of; while the later decreases the number of prototypes by deleting prototypes, which do not represent an adequate number of data points. These two actions together try to realize that there are enough prototypes to represent the distribution of the data and each prototype represents a substantial number of data points. In other words, the number of prototypes does not increase in an uncontrolled manner to become comparable with the number of training data points.

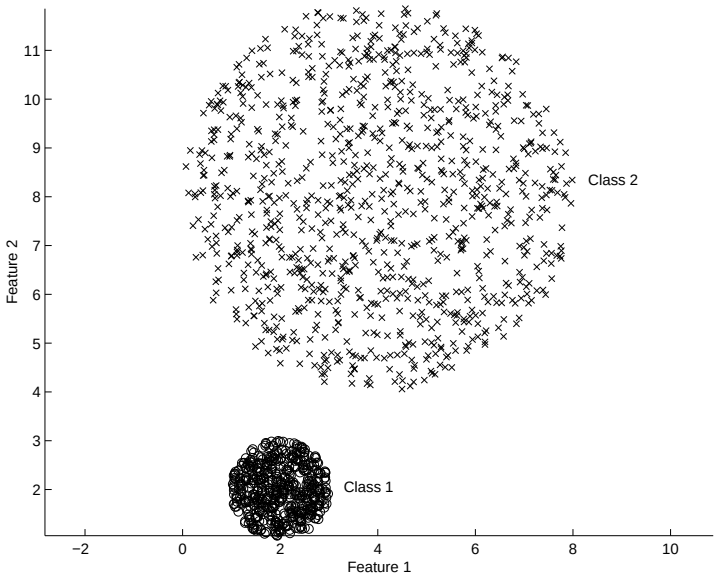

Figure 4.3: Scatterplot of the Two-Dishes data.

- The DYNAGEN algorithm tries to place each prototype at the center of a dense set of points. This is achieved by the retraining of the prototype set using SOM algorithm with the winner-only update scheme at the end of each modification cycle. This step reduces the effect of outlier data points on the positioning of the prototypes in the feature space.

The above capabilities make the tuning algorithm suitable for generating quality prototypes even for complex training data sets with linearly non-separable classes or for data sets for which points from the same class are divided into more than one cluster. These desirable qualities of the prototype set is reflected in the performance of the classifiers.

However, the result obtained for the Two-Dishes data set (Figure 4.3) indicates that just placement of prototypes at the center of each class in some situations can turn into a weakness of the algorithm. Though the 1-NMP classifier with 2 prototypes generates only 5.93% error rate for Two-Dishes, it is not good enough. Figure 4.3 shows that the data from the two classes are linearly separable and each class forms a distinct cluster. But the two classes have considerably different variances. One can reasonably expect that this data set should be classified with very high accuracy with only two prototypes. But it is beyond the capability of 1-NMP classifier. This is due to the fact that the tuning process places a prototype at the "center" of a cluster and in case of clusters having considerably different variances, such a set of prototypes, when used in a nearest prototype classifier may not classify some of the data points correctly.

To address this problem, we propose a new strategy of fine-tuning the prototypes and design a new classifier. In this new strategy, each prototype is associated with a *zone of*

influence. This modification is expected to improve the classification performance over the nearest prototype classifier and it might reduce the number of prototypes also. The scheme is described in the next section.

4.4 A new classifier and fine-tuning of prototypes

We describe the design of a new prototype based classifier, the "1 Most Similar Prototype (1-MSP)" classifier and the fine-tuning of the prototype set in order to achieve better classification performance.

4.4.1 The 1-MSP classifier

Given a set of labelled prototypes $V = \{\mathbf{v}_i \mid \mathbf{v}_i \in \Re^p \text{ and } i = 1, \ldots, \hat{c}\}$, we define a zone of influence for each prototype. The influence of a prototype $\mathbf{v}_i$ at a point $\mathbf{x} \in \Re^p$ is defined by the equation,

$$\alpha(\mathbf{v}_i, \mathbf{x}) = \exp^{-\|\mathbf{v}_i - \mathbf{x}\|^2/\sigma_i} \tag{4.7}$$

where $\sigma_i > 0$ is a parameter associated with the prototype $\mathbf{v}_i$. $\alpha(\mathbf{v}_i, \mathbf{x})$ is a decreasing function of the Euclidean distance $\|\mathbf{v}_i - \mathbf{x}\|$. The surface of constant influence for a prototype is a hypersphere in $\Re^p$ centering at the prototype. Note that, other choices of α are also possible.

The function $\alpha(\mathbf{v}_i, \mathbf{x})$ also serves as a measure of similarity between the prototype $\mathbf{v}_i$ and the point $\mathbf{x}$. The 1-MSP classifier is designed to classify a data point $\mathbf{x}$ as follows:

$$\begin{aligned} \text{Decide} \quad & \mathbf{x} \in \text{class } i \\ \Leftrightarrow \quad & \mathcal{D}_{V,\alpha}(\mathbf{x}) = \mathbf{l}_i \\ \Leftrightarrow \quad & \alpha(\mathbf{x}, \mathbf{v}_i) \geq \alpha(\mathbf{x}, \mathbf{v}_j) \quad \forall\ j \neq i. \end{aligned}$$

4.4.2 Fine tuning of the prototypes

Let V be a set of labelled prototypes ($\mid V \mid = \hat{c} \geq$ the number of classes) generated from the training data X by the DYNAGEN algorithm discussed in Section 4.3 or by some other means. Let $\mathbf{x}_i \in X$ be from class c and $\mathbf{v}_{ci}$ be the prototype for class c having the greatest similarity α_{ci}, with $\mathbf{x}_i$. Also let $\mathbf{v}_{\neg ci}$ be the prototype from an incorrect class having the greatest similarity $\alpha_{\neg ci}$ with $\mathbf{x}_i$. Thus,

$$\alpha_{ci} = \exp^{-\|\mathbf{x}_i - \mathbf{v}_{ci}\|/\sigma_{ci}}$$

and

$$\alpha_{\neg ci} = \exp^{-\|\mathbf{x}_i - \mathbf{v}_{\neg ci}\|/\sigma_{\neg ci}},$$

where σ_{ci} and $\sigma_{\neg ci}$ are the values of the σ parameters associated with the prototypes $\mathbf{v}_{ci}$ and $\mathbf{v}_{\neg ci}$ respectively.

We use an error function E,

$$E = \sum_{\mathbf{x} \in X} E_{\mathbf{x}} = \sum_{\mathbf{x} \in X} (1 - \alpha_{ci}(\mathbf{x}) + \alpha_{\neg ci}(\mathbf{x}))^2. \tag{4.8}$$

Such an error function has been used by Chiu [58] in the context of designing a fuzzy rule-based classifier. The fine-tuning algorithm developed here involves minimization of the instantaneous error $E_{\mathbf{x}}$ following the steepest descent search to modify the prototypes and their zones of influence. The algorithm is presented bellow.

Prototype fine-tuning algorithm:

Begin

- Set the learning parameters η_m and η_s.
- Set a parameter reduction factor $0 < \varepsilon < 1$.
- Set the maximum number of iteration $maxiter$.
- Compute E_0 using Eq. (4.8) for $V_0 = (\mathbf{v}_1^0, \mathbf{v}_2^0, ..., \mathbf{v}_{\hat{c}}^0)$.
- Compute the misclassification M_0 of 1-MSP classifier using V_0.
- While $(t < maxiter)$ do
  - For each $\mathbf{x}_i \in X$
    - Find the prototypes $\mathbf{v}_{ci}^{t-1}$ and $\mathbf{v}_{\neg ci}^{t-1}$.
    - Compute α_{ci} and $\alpha_{\neg ci}$.
    - Modify the prototypes $\mathbf{v}_{ci}^{t-1}$ and $\mathbf{v}_{\neg ci}^{t-1}$ and their zones of influence using the following equations:

$$
\begin{aligned}
\mathbf{v}_{ci}^{t} &= \mathbf{v}_{ci}^{t-1} - \eta_m \frac{\partial E}{\partial \mathbf{v}_{ci}^{t-1}} \\
&= \mathbf{v}_{ci}^{t-1} + \eta_m (1 - \alpha_{ci} + \alpha_{\neg ci}) \frac{\alpha_{ci}}{\sigma_{ci}^{t-1}} (\mathbf{x}_i - \mathbf{v}_{ci}) \\
\mathbf{v}_{\neg ci}^{t} &= \mathbf{v}_{\neg ci}^{t-1} - \eta_m \frac{\partial E}{\partial \mathbf{v}_{\neg ci}^{t-1}} \\
&= \mathbf{v}_{\neg ci}^{t-1} - \eta_m (1 - \alpha_{ci} + \alpha_{\neg ci}) \frac{\alpha_{\neg ci}}{\sigma_{\neg ci}^{t-1}} (\mathbf{x}_i - \mathbf{v}_{\neg ci}) \\
\sigma_{ci}^{t} &= \sigma_{ci}^{t-1} - \eta_s \frac{\partial E}{\partial \sigma_{ci}^{t-1}} \\
&= \sigma_{ci}^{t-1} + \eta_s (1 - \alpha_{ci} + \alpha_{\neg ci}) \frac{\alpha_{ci}}{\sigma_{ci}^{t-1^2}} \parallel \mathbf{x}_i - \mathbf{v}_{ci} \parallel^2 \\
\sigma_{\neg ci}^{t} &= \sigma_{\neg ci}^{t-1} - \eta_s \frac{\partial E}{\partial \sigma_{\neg ci}^{t-1}} \\
&= \sigma_{\neg ci}^{t-1} - \eta_s (1 - \alpha_{ci} + \alpha_{\neg ci}) \frac{\alpha_{\neg ci}}{\sigma_{\neg ci}^{t-1^2}} \parallel \mathbf{x}_i - \mathbf{v}_{\neg ci} \parallel^2
\end{aligned}
$$

  - End For
  - Compute E_t using Eq. (4.8) for the new set of prototypes V_t.
  - Compute the misclassification M_t of 1-MSP classifier using V_t.
  - If $M_t > M_{t-1}$ or $E_t > E_{t-1}$ then
    * $\eta_m \leftarrow (1 - \varepsilon)\eta_m$
    * $\eta_s \leftarrow (1 - \varepsilon)\eta_s$
    * $V_t \leftarrow V_{t-1}$

      Comment:*If the error is increased, then possibly the learning coefficients are too large. So, decrease the learning coefficients, and retain V_{t-1} and continue.*
  - End If
  - If $M_k = 0$ or $E_k = 0$ then
    * Stop
  - End If

- End While

End

In each iteration, for each data point, the algorithm finds *the prototype which has the maximum possibility of being responsible for correctly classifying the point* and *the*

prototype which has the maximum possibility of being responsible for wrongly classifying the point. Then the parameters associated with these two prototypes are modified so that the possibility of correct classification of the data point increases while that of wrong classification decreases.

When the algorithm terminates we have the final set of prototype V_{Final}, which is expected to give a very low error rate when used with the 1-MSP classifier.

4.4.3 Implementation and results of 1-MSP classifiers

To implement the above prototype refinement and classification scheme one question needs to be answered: How to get an initial estimate of the zones of influence, i.e., σ_i's for the initial prototypes?

In our implementation we have used the prototype set generated by our previous SOM-based DYNAGEN algorithm. If we set the initial values of σ_is as

$$\sigma_i = \text{ constant } \forall\ i,$$

then theoretically the tuning starts with a set of prototypes whose performance in a 1-MSP classifier is the same as its performance in a 1-NMP classifier since

$$\alpha(\mathbf{v}_i, \mathbf{x}_k) \geq \alpha(\mathbf{v}_j, \mathbf{x}_k)$$
$$\Leftrightarrow \quad \|\mathbf{v}_i - \mathbf{x}_k\| \leq \|\mathbf{v}_j - \mathbf{x}_k\| \quad \forall\ \mathbf{v}_i, \mathbf{v}_j, \mathbf{x}_k \in \Re^p.$$

Then with the progress of tuning, the prototypes and σ_is will be modified to reduce E. However, there are a few practical considerations. If the constant is not chosen judiciously, the tuning may not yield the desired result. To elaborate this point, if the σ_is are too small then there will be a sharp fall of α with distance and this may cause loss of important digits due to finite precision on a digital computer. Consequently, unless high precision is used, the performance of 1-NMP and untuned 1-MSP classifiers may be slightly different. On the other hand, if the σ_is are too large then for a data point, many prototypes may produce high α values and during training, the reduction of the error may be too slow making the tuning less effective. So, we get an initial estimate of the zones of influence (σ_is) for the prototypes as follows.

For each prototype $\mathbf{v}_i^0$ in the set $V_0 = \{\mathbf{v}_i^0 \mid i = 1, ..., \hat{c}, \mathbf{v}_i^0 \in \Re^p\}$ (the prototype set generated by the DYNAGEN algorithm and used as the initial set for tuning), let X_i be the set of training data closest to $\mathbf{v}_i^0$. For each $\mathbf{v}_i^0$ a set

$$S_i = \{\sigma_{ij} \mid j = 1, ..., p, \sigma_{ij} = (\sqrt{(} \sum_{\mathbf{x}_k \in X_i} (x_{kj} - v_{ij})^2))/ \mid X_i \mid\}$$

Table 4.5: Performance of the 1-MSP classifier for the group A data sets.

Data Set	Size		No. of prototypes	% of Error		Average Test Error
	Trng.	Test		Trng.	Test	
Iris	75	75	5	2.66%	2.66%	
	75	75	5	2.66%	4.0%	3.33%
Glass	105	109	28	14.28%	34.86%	
	109	105	26	13.76%	27.62%	31.24%
Normalized	284	285	8	5.63%	4.21%	
Breast Cancer	285	284	4	4.91%	7.04%	5.62%
Vowel	434	437	21	16.82%	18.53%	
	437	434	15	17.39%	17.51%	18.02%
Norm4	400	400	4	4.25%	4.25%	
	400	400	4	4.0%	3.5%	3.87%
Two-Dishes	750	750	2	0%	0.13%	
	750	750	2	0%	0%	0.07%

is computed. Then the set of initial σ_is is computed as

$$S = \{\sigma_i \mid i = 1, ..., \hat{c}, \sigma_i = (\sum_{\sigma_{ij} \in S_i} \sigma_{ij})/p\}.$$

Thus, σ_i is the average of the standard deviations of the individual components of data vectors $\mathbf{x} \in X_i$ about the prototype $\mathbf{v}_i^0$. This is just a choice for the initial σ_is, other choices are possible too.

Table 4.5 summarizes the result of 1-MSP classifier for six data sets in Group A. We have used the same training and test division for all data sets as used for the 1-NMP classifier. For most of the data sets the results show marked improvement over those of 1-NMP classifier while for the rest the performance remains almost the same. We emphasize on the Two-Dishes data set to explain our motivation for the design of 1-MSP classifier. The result shows an average of 0.07% error rate for Two-Dishes with only two prototypes and the best result obtained shows zero error.

However, the 1-MSP classifier fails in case of the original Breast Cancer data set. Our algorithm assumes hyperspherical clusters in the feature space. This is expected when in a cluster the standard deviations of all features are nearly equal. But the Breast Cancer data have 30 features, of which majority have values of the order of 10^{-2} and some are of the order of 10^3, and the standard deviations of different features are considerably different. Our initialization scheme computes the average of these standard deviations to associate a hyperspherical zone of influence with each prototype. Since most of the

components have small values, the σ_is are small and αs for most of the data points become very small, leading to loss of precision and high misclassification rates. However, if the σ_is are increased artificially (by a factor of 400, say), the performance matches that of the 1-NMP classifier, but the reduction of the error function E becomes very slow during training. Such a data could be better handled if the zones of influence of the prototypes are not limited to hypersphere. But such schemes (like using Mahalanobis distance instead of Euclidean distance) would lead to more computational overhead. So, here we try to get around this problem by normalizing the data over each component. The Normalized Breast Cancer data exhibits an excellent performance of the 1-MSP classifier.

Apparently the 1-MSP classifier has some similarity with the Radial Basis Function (RBF) network, since each prototype is associated with a zone of influence and the strength of the influence is determined by a Gaussian function of the Euclidean distance from the prototype. The hidden nodes in a RBF network also use the Gaussian functions as activation functions (there are other possibilities too). However, in the conventional RBF network, a linear aggregation function is used between the hidden layer and the output layer. There is complete connection between the hidden layer and the output layer. If we think of an RBF-like architecture for the 1-MSP classifier we cannot have the complete connection between the hidden layer and the output layer. The aggregation function in each output node would be a *max* operator, not a linear function. Moreover, the α-function used here need not necessarily be a basis function. As we shall see, in general, the RBF network needs more nodes for similar performances. However, our algorithm can be used as a preprocessing stage for RBF classifier, where the weight vectors of RBF nodes can be initialized with the positions of the prototypes and the spreads of the activation functions can be initialized with the value of the corresponding σ. To make a comparison we have classified the group A data sets (same as in Table 4.5) using RBF networks with the same number of RBF nodes as the number of prototypes used by the 1-MSP classifier. We have used the routines available in the Neural Network Toolbox of Matlab-5. For all experiments we used 2.0 for the spread of the radial basis functions. The results for the RBF classifiers are summarized in Table 4.6.

Table 4.6 shows higher error rates for the RBF classifiers compared to 1-MSP classifiers for most of the data sets. For Iris and normalized Breast Cancer the RBF network shows a slightly better performance. However, it should be noted that depending on the data set, the performance of RBF classifiers is highly sensitive to the choice of various parameters such as number of hidden nodes, the spread parameters etc. In contrast, while building 1-MSP classifiers, the user needs to specify only two parameters K_1 (eq. 4.2) and K_2 (eq. 4.3). Further, the choice of these parameters does not need to be varied much for different data sets. In fact, for all classifiers reported in this thesis, we have used

Table 4.6: Performance of the RBF networks for the group A data sets.

Data Set	Size		Network configuration	% of Error		Average Test Error
	Trng.	Test		Trng.	Test	
Iris	75	75	4-5-3	4.0%	2.66%	
	75	75	4-5-3	2.66%	2.66%	2.66%
Glass	105	109	9-28-6	22.8%	49.54%	
	109	105	9-26-6	30.27%	43.8%	46.67%
Normalized	284	285	30-8-2	3.17%	2.45%	
Breast Cancer	285	284	30-4-2	3.51%	5.98%	4.21%
Vowel	434	437	2-21-2	55.06%	56.97%	
	437	434	2-15-2	48.28%	50.46%	53.71%
Norm4	400	400	3-4-4	36.5%	36.25%	
	400	400	3-4-4	37.25%	38.5%	37.37%
Two-Dishes	750	750	2-2-2	9.6%	10.13%	
	750	750	2-2-2	6.66%	6.0%	8.06%

Table 4.7: Performance of the 1-MSP classifier for the group B data sets.

Data Set	Size		No. of prototypes	% of Error	
	Trng.	Test		Trng.	Test
Cone-Torus	201	199	12	16.5%	14.75%
Normal Mixture	250	1000	4	13.4%	9.7%
Satimage1	500	5935	27	13.4%	15.6%
Phoneme	500	4904	5	19.8%	20.53%

the same values for these two parameters and obtained fairly consistent and satisfactory classification performance.

Table 4.7 summarizes the performance of the 1-MSP classifiers for the group B data sets. For each of the data sets the training error is lower for the 1-MSP classifier than the 1-NMP classifier. The test error for Cone-Torus data decreases substantially. Test error for Normal Mixture and Satimage1 remains almost the same, while for the Phoneme data there is some improvement in the performance on the test set.

In case of the Normal Mixture data set it can be observed that the test error is substantially lower than the training error for both 1-NMP and 1-MSP classifiers. Though such a result is not very common, it may occur due to the nature of the data distribution and its representation in the training-test partition. At least two situations can be readily identified that can produce such results. In the first situation the classes are overlapped

and there is a relatively higher concentration of the data points in the training set from the overlapping region compared to that in the test set. Typically a classifier makes more mistakes for the data points in the overlapped region. Since the test data contain lower proportion of points in the overlapped region compared to the training data, overall classification performance on the test data may be better than that on the training data. This has happened with the training-test partition used for the Normal Mixture data set. Similar trend can be observed in the results for other classifiers using the same partition (MLP in Table 4.8 and RBF in Table 4.9). On the other hand, when different partitions are used, as in the cross-validation experiments described later in Table 4.11 and Table 4.12, the test error is close to the training error.

In the other situation, even if the classes are not overlapped, it can happen when the class distribution of training data is different from that of the test data. If there is some class for which the classifier performs better on the training set and if that class has a higher proportions in the test data, then the classifier may show a higher performance on the test data. This has happened with the data sets Satimage2 and Satimage3 used in the thesis. In these cases the training data sets contain equal number of points from each class, while in the test sets there is a large variation in the number of points from different classes.

The data sets in group B, as mentioned earlier, have been extensively investigated in [191]. We reproduce here some results using MLP (multilayer perceptron) and RBF on these data sets in Table 4.8 and Table 4.9 respectively. To make a fair comparison, for both networks, we have chosen two architectures having the number of nodes closest to the number of prototypes used by our 1-MSP classifier.

Comparison of Table 4.7 with Table 4.8 shows that for the Cone-Torus data the performance of the 1-MSP classifier with 12 prototypes is comparable to the average performance of the MLP with 10 and 15 hidden nodes. For Normal Mixture, the performance of 1-MSP with 4 prototypes is also comparable with the performance of MLP networks with 5 and 10 hidden nodes. For Satimage1 the 1-MSP classifier produces 13.4% training error and 15.6% test error, but the MLP even with 64 hidden nodes produces 24.4% training error and 23.08% test error. The performance of MLP with 20 hidden nodes is very poor (79.2% training error and 75.92% test error!). For the Phoneme data the MLP with 5 hidden nodes produces a little better result than 1-MSP with 5 prototypes. Comparing Table 4.9 with Table 4.7 we find that the performance of 1-MSP classifier and RBF network is comparable for Normal Mixture; for Satimage1 and Phoneme the 1-MSP classifier performs a little better than the RBF network, while for Cone-Torus the performance of RBF network is a little better than the 1-MSP classifier.

Table 4.8: Results with MLP networks for group the B data sets [191].

Data Set	Network Size	Trng Error	Test Error
Cone-Torus	2-10-3	15.25%	14.25%
	2-15-3	13.50%	12.00%
	Average	14.37%	13.12%
Normal Mixture	2-5-2	12.00%	10.00%
	2-10-2	12.80%	10.30%
	Average	12.40%	10.15%
Satimage1	4-20-6	79.20%	75.92%
	4-65-6	24.40%	23.08%
	Average	51.80%	49.50%
Phoneme	5-5-2	14.00%	18.23%
	5-10-2	16.80%	21.04%
	Average	15.40%	19.63%

Table 4.9: Results with RBF networks for the group B data sets [191].

Data Set	Network Size	Trng Error	Test Error
Cone-Torus	2-10-3	16.50%	13.75%
($\sigma = 3$)	2-15-3	17.00%	14.00%
	Average	16.75%	13.87%
Normal Mixture	2-5-2	14.00%	9.50%
($\sigma = 1.5$)	2-10-2	12.40%	10.00%
	Average	13.20%	9.75%
Satimage1	4-15-6	14.80%	17.02%
($\sigma = 10$)	4-20-6	12.80%	15.52%
	Average	13.80%	16.27%
Phoneme	5-5-2	21.00%	23.65%
($\sigma = 2$)	5-10-2	18.80%	21.31%
	Average	19.90%	22.48%

4.5 Cross-validation and comparison with k-NN and SVM classifiers

While studying the performance of a pattern classification scheme, it is important to study the stability or consistency of the learning method adopted by the scheme. For this we have conducted 10-fold cross-validation experiments and analyzed the performance of different classifiers using error statistics.

In this section we first report the results of cross-validation experiments with the proposed 1-NMP and 1-MSP classifiers. Then we present the results of cross-validation with the k-NN and support vector machine (SVM) classifiers and compare them with those of the proposed classifiers.

4.5.1 Results of cross-validation with proposed classifiers

For convenience, we list in Table 4.10 the sizes of training-test partitions of different data sets for the cross-validation experiments. The results are summarized in Table 4.11 and Table 4.12 for 1-NMP classifiers and 1-MSP classifiers respectively. For the reasons explained earlier in this chapter we do not include the Breast Cancer data set but use the normalized Breast Cancer data set. Columns 3 and 4 of Table 4.11 display the average training and test error percentages along with their standard deviations (standard deviations are shown within parentheses). In addition to the error rates, since generation of prototypes is a part of the learning task performed by the 1-NMP classifier, in Table 4.11 we also report the average and standard deviation of number of prototypes. Since the corresponding 1-MSP classifiers also have the same number of prototypes, we do not repeat these statistics in Table 4.12.

Comparing the test error statistics reported in Table 4.11 and Table 4.12, it can be observed that 1-MSP classifiers show significantly lower average error rates than the 1-NMP classifiers for the Iris, Normalized Breast Cancer, Vowel, Two-dishes and Phoneme data sets. Among other data sets, for Glass, Cone-torus and Satimage1, 1-MSP classifiers show marginally lower average error rates; while for Norm4 and Normal Mixture 1-NMP classifiers perform marginally better. Overall, for most of the data sets, the 1-MSP classifiers exhibit better or marginally better average error rates than those for 1-NMP classifiers. This is also evident from Fig. 4.4, which displays the average test error for all data sets for the two classifiers. The standard deviations of the error rates are very close for the two types of classifiers for most of the data sets. For Glass, Vowel and Cone-Torus data sets the 1-MSP classifiers resulted in higher standard deviation values. This is probably due to the overlapped class structure of the data sets.

Table 4.10: Training and test partition sizes for the cross-validation experiments

Data Set	Size of Training Set	Size of Test Set
Iris	135	15
Glass	189	21
N. B. Cancer	504	56
Vowel	783	87
Norm	720	80
Two-Dishes	1350	150
Cone-Torus	720	80
Normal Mixture	1125	125
Satimage1	5787	643
Phoneme	4860	540

Table 4.11: Results of 10-fold cross validation for 1-NMP classifiers

Data Set	No. of prototypes Avg. (St. D.)	E_{tr} (%) Avg. (St. D.)	E_{te} (%) Avg. (St. D.)
Iris	5.00 (0.00)	9.33 (0.80)	10.00 (4.71)
Glass	27.70 (3.80)	22.33 (2.31)	34.29 (10.48)
N. B. Cancer	4.50 (0.85)	8.02 (0.72)	7.86 (2.69)
Vowel	23.40 (2.50)	19.80 (1.39)	20.34 (3.25)
Norm4	4.00 (0.00)	4.14 (0.28)	4.00 (1.42)
Two-Dishes	2.00 (0.00)	6.73 (0.81)	7.00 (2.42)
Cone-Torus	10.80 (1.62)	9.89 (0.87)	9.88 (2.73)
Normal Mixture	4.00 (0.00)	9.10 (0.52)	9.04 (2.85)
Satim1	23.60 (1.90)	15.45 (0.41)	15.72 (1.27)
Phoneme	6.10 (0.74)	22.11 (0.57)	21.81 (1.96)

Table 4.12: Results of 10-fold cross validation for 1-MSP classifiers

Data Set	E_{tr} (%) Avg. (St. D.)	E_{te} (%) Avg. (St. D.)
Iris	4.52 (1.37)	6.00 (4.92)
Glass	20.37 (2.47)	33.81 (10.15)
N. B. Cancer	5.22 (0.44)	5.36 (3.04)
Vowel	18.28 (1.13)	18.53 (5.23)
Norm4	4.11 (0.25)	4.50 (1.47)
Two-Dishes	0.00 (0.00)	0.00 (0.00)
Cone-Torus	8.86 (1.28)	9.13 (5.14)
Normal Mixture	8.80 (0.41)	9.60 (2.69)
Satim1	15.50 (0.51)	15.61 (1.46)
Phoneme	20.67 (0.24)	20.81 (2.05)

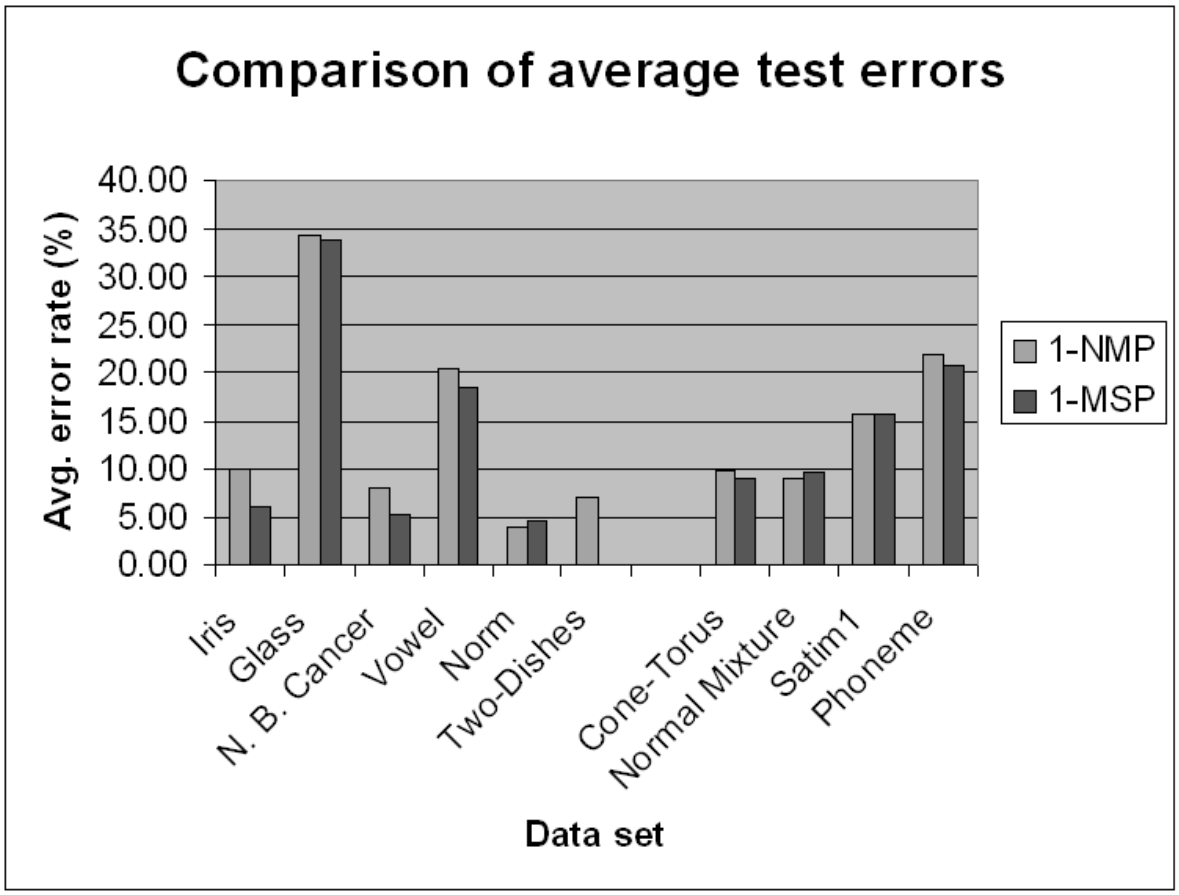

Figure 4.4: Comparison of average test error rates of 10-fold cross-validation for the proposed methods

4.5.2 Comparison of the proposed classifiers with k-NN and SVM classifiers

The k-NN classifier, due to its simplicity, is one of the most popular classification techniques. We have discussed it in Section 1.3.1 along with other statistical classifiers. The SVM classifier for two-class problems has also been discussed in the same section. Although there have been a few attempts to design multi-class SVMs [6], typically two-class SVM is used to solve the multiclass problem. Usually either One vs. One [252] or One vs. All [333] strategy is used. We use the One vs. One method for our multiclass data sets. In this method, for a c class problem we build $\frac{c(c-1)}{2}$ two-class classifiers, one for each pair of distinct classes, trained with the training data from the corresponding pair of classes. An unknown sample $\mathbf{x}$ is classified by each of the classifiers into one of the classes for which it is trained. The sample is assigned to the class which receives the majority of the votes.

To compare the proposed schemes experimentally with the k-NN and SVM classifiers we have conducted 10-fold cross-validation experiments with group A and group B data sets. For k-NN classifiers, for each data set we have experimented with 5 different values of k, namely, 1, 3, 5, 7 and 9. For SVM classifiers we conducted experiments with two variants, namely, linear SVM and SVM with RBF kernel. For building SVM classifiers we have used the *SVMLight* [142] package which has a built-in mechanism to compute the value of the regularization parameter C. For the SVMs with RBF kernel we have used the spread parameter $\gamma = 0.1$. We have experimented with a few other choices of γ and for this choice, the performance of the classifiers for most of the data sets were found quite good.

Table 4.13 and Table 4.14 depict the error statistics of the k-NN and SVM classifiers for both group A and group B data sets. The test error statistics of 1-NMP and k-NN classifiers in Table 4.11 and Table 4.13 reveal that for Iris, Normalized Breast Cancer, Two-dishes and Phoneme data sets the k-NN classifiers perform significantly better. This is due to the fact that for data sets with good class structures the k-NN works very well as a large number of representative points are maintained in the training set. For the Vowel data set the 1-NMP classifier performs better. For Cone-torus data set with complex class structures the 1-NMP classifier result in a significantly lower average error rate. For Norm4 and Normal Mixture also 1-NMP classifiers show better performance. For the Glass data set, with lower values of k (1,3) k-NN performs better, but for higher k values (5,7 and 9), the performance degrades and becomes worse than that of 1-NMP. This is probably because of the very complex and overlapped structure of the data. For Satimage1 the performance of k-NN is marginally better for larger values of k (5,7 and 9), while for lower k values (1,3) 1-NMP outperforms k-NN.

Table 4.13: Result of 10-fold cross-validation experiment with k-NN classifiers: The statistics of test misclassification rates are reported. The numbers in the parentheses denote the values of standard deviation.

Data Set	k=1 Av.(St. D.)	k=3 Av.(St. D.)	k=5 Av.(St. D.)	k=7 Av.(St. D.)	k=9 Av.(St. D.)
Iris	4.00 (4.66)	4.00 (4.66)	3.33 (4.71)	2.67 (4.66)	3.33 (4.71)
Glass	27.14 (5.96)	31.43 (7.84)	35.71 (7.86)	36.67 (11.89)	39.52 (11.46)
N. B. Cancer	3.21 (2.50)	2.68 (2.27)	3.04 (2.80)	2.68 (2.27)	3.04 (2.39)
Vowel	33.68 (3.34)	26.90 (3.12)	23.10 (3.09)	23.68 (2.72)	20.69 (3.07)
Norm4	7.13 (2.89)	5.00 (2.76)	5.13 (3.25)	4.50 (2.90)	4.63 (2.70)
Two-Dishes	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Cone-Torus	17.50 (5.77)	17.00 (6.21)	17.50 (4.71)	17.00 (4.38)	15.00 (4.41)
Normal Mixture	11.36 (2.89)	10.00 (2.45)	10.48 (2.40)	10.00 (2.83)	10.00 (3.00)
Satimage1	18.37 (2.24)	15.89 (1.26)	15.58 (0.97)	14.87 (0.97)	14.56 (0.82)
Phoneme	9.43 (1.54)	10.74 (1.47)	11.28 (1.76)	12.35 (1.73)	12.74 (1.61)

Table 4.14: Result of 10-fold cross-validation experiment with SVM classifiers

Data Set	Linear		RBF kernel	
	E_{tr} (%) Avg. (St. D.)	E_{te} (%) Avg. (St. D.)	E_{tr} (%) Avg. (St. D.)	E_{te} (%) Avg. (St. D.)
Iris	6.67 (2.18)	9.33 (7.83)	3.04 (1.13)	2.67 (3.44)
Glass	63.81 (0.87)	62.96 (7.82)	35.71 (2.62)	43.92 (8.17)
N. B. Cancer	3.83 (0.28)	3.93 (2.35)	3.75 (0.22)	4.11 (2.39)
Vowel	24.21 (0.59)	24.25 (6.38)	24.21 (0.59)	24.25 (6.38)
Norm4	4.10 (0.20)	5.00 (2.57)	3.47 (0.60)	4.75 (2.34)
Two-Dishes	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Cone-Torus	25.19 (0.66)	25.75 (5.28)	16.92 (0.69)	17.50 (3.54)
Normal Mixture	11.76 (0.44)	11.76 (3.42)	11.13 (0.39)	11.12 (3.51)
Satim1	15.81 (0.26)	15.85 (1.79)	9.57 (0.16)	16.10 (1.09)
Phoneme	22.49 (0.23)	22.67 (1.56)	18.65 (0.33)	19.07 (2.24)

Compared to the 1-MSP classifiers (Table 4.12) the k-NN classifiers show better test performance on Iris, Normalized Breast Cancer and Phoneme data, while both classifiers show zero test error rates for the Two-Dishes data set. On the other hand, for Vowel, Norm4, Cone-torus and Normal Mixture data sets the 1-MSP classifiers show lower average test error rates and comparable standard deviations. For Glass data set, for k=5,7 and 9 the 1-MSP outperforms k-NN. For the Satimage1, like 1-NMP, the 1-MSP classifier shows better performance than the k-NN classifiers with k=1 and 3.

We have demonstrated that the 1-MSP classifier, which is an improved version of the 1-NMP classifier, performs better than the 1-NMP classifier for majority of the data sets studied. So here we compare the 1-MSP classifier with the SVM classifier. Table 4.12 and Table 4.14 reveal that except for Normalized Breast Cancer and Two-dishes, the 1-MSP classifier outperforms the linear SVM in terms of average test error rates. For Two-dishes both classifiers achieve zero error rates and for Iris, Glass, Vowel, Cone-torus, Normal Mixture and Phoneme the average error rates for the 1-MSP classifiers are much lower than those for the SVM classifiers. In most of the cases the standard deviations of error rates are comparable for both types of classifiers.

While using SVMs with RBF kernels, the performance improves significantly for several data sets. Compared to 1-MSP classifiers, the SVMs with RBF kernels result in lower average test error rates for Iris, Normalized Breast Cancer and Phoneme data sets. However, for Glass, Vowel, Cone-torus and Normal Mixture data sets the 1-MSP classifier achieves lower average error rates. For Norm4 also the 1-MSP classifier performs marginally better. It is interesting to observe that for several data sets with complex and overlapped class structure the 1-MSP classifier is found to perform better.

There are a few points those distinguish the proposed classifier design approaches from the k-NN and SVM classifiers. Though k-NN classifiers are simple and usually perform quite well, a straightforward implementation of a nearest-neighbor classifier requires large memory (the complete set of training samples X has to be kept in memory) and involves large computational overhead (the distance between $\mathbf{x}$ and each of the training data points has to be computed). This makes it difficult to use the k-NN methods for large and high dimensional data. There have been several attempts to minimize the computational overhead of k-NN algorithm [109, 70], some of which approximate the k-NN scheme. We have not considered such implementations here. Theoretically, with increase in the value in k the performance of k-NN classifier improves and approaches the optimal Bayes error rate [83]. However, due to the finite size of the training data it is often not so in real life cases. It can be observed from Table 4.13 that for the data sets Glass and Phoneme the average error rates increase with k.

In prototype-based classifiers, such as those proposed here, the training data set is

represented by a set of prototypes $V = \{(\mathbf{v}_i, \mathbf{l}_i) \mid i = 1, ..., \hat{c}\}$ where, $\hat{c} \geq c$, which is much smaller than the number of training samples. Each prototype can be thought as a representative of a subset of X. Thus, they overcome the problems of large memory and computational requirements. Moreover, the prototype based classifiers are more readable than k-NN classifiers because prototypes represent typical cases in the data set. Further, our objective in the next chapter is to develop fuzzy rule based classifiers, where the prototypes are converted into fuzzy rules. To retain the advantage of interpretability of a fuzzy rule based system, it is most desirable for us to devise a scheme that uses small number of prototypes with good performance.

The support vector machines provide a robust means of building classifiers with good performance. The theoretical foundation behind its formulation also makes it attractive to use. A system built with linear SVM is quite understandable but when kernel functions are used for implicit nonlinear projection to high dimensional space, the understandability of the structure of data as well as of the decision boundaries suffers heavily. In contrast, the prototype based classifiers proposed in this chapter are conceptually very simple and easily understandable. Moreover, from the experiments it is found that their performance is quite comparable with those of the SVM for several data sets. For a given data set often the performance of SVM is highly sensitive to the choice of the kernel parameters. This is not the case with the proposed classifiers.

4.6 Classifying landcover types from multispectral satellite images

In the following two chapters we shall develop schemes for building fuzzy rule based classifiers for complex pattern recognition tasks. Those schemes use the DYNAGEN algorithm described in this chapter for generation of initial prototypes. The fuzzy rule based classifiers are designed with the aim of performing classification of large and complex data sets. To test the performance of these fuzzy rule based classifiers we use two multispectral satellite images for classifying landcover types. We call these images **Satimage2** and **Satimage3** respectively. Apart from the fact that these two data sets pose difficult classification problems, we use them because in Chapter 6 we shall propose several classifiers that exploit *spatial contextual* information. Though the fuzzy rule based classifiers outperform both 1-NMP and 1-MSP classifiers for these data sets, for the sake of continuity we report here the performances of 1-NMP and 1-MSP classifiers with these multispectral satellite images.

4.6.1 The multispectral satellite image data

Satimage2 is a Landsat Thematic Mapper (Landsat-TM) image available along with full ground truth in the catalog of sample images of the ERDAS software and used for testing various algorithms [190]. The image covers the city of Atlanta, USA and its surrounding. *Satimage3* is also a Landsat-TM image depicting outskirts of the city of Vienna, Austria [36].

The Satimage2 is of size 512×512 pixels, captured by seven detectors operating in different spectral bands from Landsat-TM3 multispectral scanner. Each of the detectors generates an image with pixel values varying from 0 to 255. The 512×512 ground truth data provide the actual distribution of classes of objects captured in the image. From this data we produce the labelled data set with each pixel represented by a 7-dimensional feature vector and a class label. Each dimension of a feature vector comes from one channel and the class label comes from the ground truth data.

Kumar et al. [190] studied Satimage2 using a number of techniques including the maximum likelihood classification, artificial neural network and fuzzy integral method. They found the fuzzy integral method to produce the best performance in their study.

Satimage3 also is a seven channel Landsat-TM image of size 512×512. However, due to some characteristic of the hardware used in capturing the image the first row and the last column of the images contain gray value 0. So we did not include those pixels in our study and effectively worked with 511×511 image. The ground truth containing four classes is used for labelling the data. More details about Satimage2 and Satimage3 are provided in the next chapter.

Satimage3 has been studied by Bischof et al. [36] using the maximum likelihood classifier (MLC) and neural networks. They reported better performance with neural networks compared to MLC.

In our studies we generated 4 sets of training samples for each of the images. For Satimage2 each training set contains 200 data points randomly chosen from each class. This choice is made to conform to the protocol followed in [190]. For Satimage3 we include in each training set 800 randomly chosen data points from each of the four classes. The training set used in [36] contains unequal number of points from different classes. The lowest number of points chosen from a class in [36] was 922 (water). We use 800 points from each class in a training set for the purpose of comparing our result with the reported ones. The classifiers designed with the training data are tested on the whole images. *Note that, the training sets used here are very small compared to the sizes of the whole data sets. For Satimage2 the training set is 0.61% of the total data and for Satimage3 the same is 1.22% only.* We shall be using the same training sets for building

Table 4.15: Classification performances of 1-NMP classifiers designed using different training sets for the multispectral satellite images

Data	Training Set	No. of Prototypes	Error Rate in Training Data	Error Rate in Whole Image
Satimage2	1.	30	26.7%	22.2%
	2.	25	24.4%	23.0%
	3.	25	22.7%	18.7%
	4.	27	25.8%	18.2%
Satimage3	1.	14	20.0%	17.6%
	2.	14	19.5%	19.8%
	3.	12	21.3%	21.8%
	4.	11	21.9%	23.9%

the fuzzy rule based classifiers also.

4.6.2 Experimental results

Each training set is used to design a 1-NMP classifier and a 1-MSP classifier and then tested on the whole image. The performance of the 1-NMP classifiers and 1-MSP classifiers are summarized in Table 4.15 and Table 4.16 respectively. Table 4.15 shows that the best result for Satimage2 on the test data yields 81.8% correct classification and the average classification performance is 79.47%, which is better than the best classification rate 78.15% achieved by Kumar et al. [190] using fuzzy integral based method on the same image. 1-MSP classifiers for Satimage2 shows improvement in performance ranging from 1% (training set 3) to 4.8% (training set 2), while for training set 4 we get lowest error rate of 16.4%.

For Satimage3 the reported result [36] shows 84.7% accuracy with the maximum-likelihood classifier and 85.9% accuracy with a neural network based classifier. For Satimage3 the best 1-NMP classifier achieves 82.4% test accuracy with training set 1, which is comparable with the reported result. Other classifiers show slightly higher error rates. The performance of the 1-MSP classifiers are marginally better, except for training set 3, which shows a slight reduction in accuracy.

Table 4.16: Classification performances of 1-MSP classifiers designed using different training sets for the multispectral satellite images

Data	Training Set	No. of Prototypes	Error Rate in Training Data	Error Rate in Whole Image
Satimage2	1.	30	24.8%	21.1%
	2.	25	22.3%	18.2%
	3.	25	22.3%	17.7%
	4.	27	22.6%	16.4%
Satimage3	1.	14	21.1%	17.2%
	2.	14	18.8%	18.9%
	3.	12	21.5%	22.8%
	4.	11	21.8%	22.7%

4.7 Conclusion

We have presented two comprehensive schemes for designing prototype based classifiers. The schemes address all major issues involved with the design. Based on the training data set, the proposed SOM-based DYNAGEN algorithm takes care of prototype generation as well as determination of an adequate number of prototypes needed for the classification tasks by a nearest-prototype classifier, here named 1-NMP classifier. It may seem that other clustering algorithms can be used for prototype generation. However, the proposed DYNAGEN algorithm offer several advantages. If hard c-means (HCM) clustering is used, the number of prototype need to be guessed. There also remains the possibility of underutilization of the prototypes. On the other hand, if fuzzy c-means (FCM) clustering algorithm is used, though underutilization of prototypes is avoided, while tuning the prototypes, the training starts from the beginning, i.e., the information extracted till the last iteration is lost. In contrast, the SOM-based approach proposed here is incremental in nature. However, neural gas algorithm [240] (which is a variant of SOM) may be used for prototype generation with the same effect.

The performance of the 1-NMP classifier is quite good. However, as pointed out earlier, the nearest-prototype classifier cannot take care of large variations in the variances of the data from different classes. To equip the classifier to deal with such a situation it must use the variance of the data represented by each prototype. In our second classifier (1-MSP) we associate with each prototype a zone of influence that tries to cover the spread of the data represented and thereby it accounts for the class variance. To measure the proximity we use a (Euclidean) norm-induced *similarity measure* that implicitly defines

the limit of the zone of influence of each prototype.

Apparently one can think of two ways of using Mahalanobis distance to deal with data having non-hyperspherical class/subclass structure: (1) Running some clustering algorithm on the whole training data, and computing the covariance matrix for each cluster. But in this method the class information is ignored and a cluster may contain data from different classes. This may particularly defeat the purpose of the exercise. (2) Clustering the data from each class separately to generate the prototypes and computing the covariance matrix of the data points closest to each prototype. This 'closeness' criterion is again based on Euclidean distance. So the computation of Mahalanobis distance based on the covariance matrix thus obtained may not solve the problem, unless it is updated during training, as in Gustafson and Kessel [116]. Moreover, although this option uses the class label information, it cannot capture interaction between classes and this has to be accounted during refinement of the prototypes.

In the following chapter we shall develop a generalization of 1-MSP classification scheme leading to realization of a fuzzy rule-based classifier, where the rules are obtained in a data-driven manner.

Chapter 5

Extraction of Fuzzy Rules for Classification [1]

[1] Part of this chapter has been published in [206] and the whole of it is published in [267].

5.1 Introduction

In recent times fuzzy rule based classifiers [31, 32, 137, 191] have attracted attention of many researchers due to their several useful features compared to more traditional distance based classifiers. At the conceptual level their working is closer in spirit to human reasoning. At the practical level they can be used very easily to handle several problematic situations. For example, outliers can be detected by small firing strengths of the rules, highly overlapped regions can be detected by high firing strength of rules from different classes. Most interestingly, the problem of high variation in the variances of different features, which often degrades the performance of a distance based classifier substantially, can be handled naturally by fuzzy rules due to the atomic nature of the antecedent clauses. Although theoretically the Maximum Likelihood Classifiers (MLCs) assuming Gaussian mixture model can also handle this problem, they face the formidable problem of density estimation.

In this chapter we describe a comprehensive scheme for designing fuzzy rule based classifiers. This is a multi-stage scheme. In the first stage a set of labelled prototypes representing the distribution of the training data is generated using the Self-organizing Map (SOM) based prototype generation algorithm proposed in Chapter 4. Other methods of labelled prototype generation can also be used. The algorithm employs a combination of unsupervised and supervised clustering of the training data to generate an adequate number of prototypes representing the overall as well as class-specific distribution of the training data. Then each of these prototypes is converted to a fuzzy rule. Each of the fuzzy rules represents a region, may be overlapped, in the feature space. We call this region the **context** of the rule. Note that, throughout this chapter the word "context" is used to mean a region in the feature space, not pixels in the neighborhood of another pixel in an image.

Next we develop a tuning algorithm for the fuzzy rules that fine-tunes the peaks as well as the spreads of the fuzzy sets associated with the rules. We call this the *context tuning* stage. The exact implementation of the tuning algorithm is dependent on the conjunction operator used to represent the AND connective in the antecedent part of the rules. Here we develop the tuning algorithms for two different conjunction operators, namely the **product** and the **softmin**. The tuned rules are used to classify the unknown samples based on the firing strengths of the rules. A test sample is assigned to the class of the rule generating the highest firing strength.

The softmin operator, based on the value of a parameter, can approximate a whole family of operators including *min, average* and *max*. This raises the possibility of using a set of rules where each rule is free to use a different conjunction operator depending

on the context it operates on. This is known as **context-sensitive reasoning**. Here, depending on the context the reasoning scheme may change. To realize this possibility we also develop an algorithm for tuning the parameters of the softmin operators on a per-rule basis. The schemes, which use the same conjunction operator (reasoning scheme) for all rules will be called **context-free reasoning** in this chapter.

The chapter is organized as follows: in section 5.2 we discuss various issues related to designing a fuzzy rule based classifier and how these issues are dealt with by other researchers. Section 5.3 we describe the proposed method for extraction of fuzzy rules from data using prototypes generated by SOM-based method developed in the previous chapter. The idea of tuning the fuzzy rule base for context-sensitive inferencing is developed in Section 5.4. The implementation details and experimental results are presented in Section 5.5. We conclude the chapter in Section 5.6.

5.2 Fuzzy rule based systems for pattern classification

5.2.1 Which type of fuzzy rules to use

A fuzzy rule based system consists of a set of fuzzy rules $\{R_i \mid i = 1, 2, \cdots, M\}$. The rules can have different structures. The structure of the Mamdani and Assilian [236] or MA rules has the following form

$$R_i : \textbf{If } x_1 \textbf{ is } A_{i1} \textbf{ AND } x_2 \textbf{ is } A_{i2} \textbf{ AND } \cdots \textbf{ AND } x_p \textbf{ is } A_{ip} \textbf{ then } y \textbf{ is } B_i,$$

where x_k is the k-th component of the input vector $\mathbf{x} \in \Re^p$, A_{ik} is a fuzzy set used in the i-th rule and defined on the domain of x_k, i.e., on the universe of the k-th input feature and B_i in the consequent part is a fuzzy set defined on the domain of the output variable y. These fuzzy sets are often designed to have semantic meaning (such as low, medium, high etc.) with respect to the particular feature and also called linguistic values. The use of linguistic values, allows one to model human-like reasoning systems with fuzzy rules. Naturally such a system can be used as tool for approximate reasoning with good degree of tolerance to imprecision and uncertainty that may exist in real-life systems. Geometrically each rule can be viewed as a fuzzy point or information granule in the combined input-output space. When a sample data point $\mathbf{x} \in \Re^p$ is presented to the system, the LHS or the antecedent part of the rule calculates a firing strength

$$\alpha_i(\mathbf{x}) = T(\mu_{i1}(x_1), \mu_{i2}(x_2), \cdots, \mu_{ip}(x_p)), , \tag{5.1}$$

where, $\mu_{ik}(x_k)$ is the membership value of x_k to the fuzzy set A_{ik} and T is a T-norm [164] implementing the AND operation between the antecedent clauses. The output of R_i is the fuzzy set B_i clipped at the height α_i. In general, more then one rules fire with non-zero firing strength. These rules together produce a fuzzy set which is the union of all clipped B_is. This output fuzzy set is then "defuzzified" using some defuzzification method such as *Center of Gravity (COG)* or *Mean of Maxima (MOM)*, to obtain a crisp value for the output of the system [164]. As it can be observed, MA rules deal with continuous valued output variables and they have found numerous applications in control, prediction, function approximation etc. [251].

There is also another type of rules due to Takagi and Sugeno [319]. This type of rules are known as Takagi-Sugeno rules or TS rules. The general form of this rule is as follows:

$$R_i : \textbf{If } x_1 \textbf{ is } A_{i1} \textbf{ AND } x_2 \textbf{ is } A_{i2} \textbf{ AND } \cdots \textbf{ AND } x_p \textbf{ is } A_{ip} \textbf{ then } y \textbf{ is } \mathbf{u}_i(\mathbf{x}).$$

The set of functions $\{\mathbf{u}_i : \Re^p \mapsto \Re^q : 1 \leq i \leq M\}$ comprise the consequent part of the rule base. In general, each $\mathbf{u}_i$ is a vector field whose components are scalar fields of some specified form (e.g., constant, linear, affine, quadratic, Gaussian etc.). The output of the rule base is the vector $\mathbf{S}_{TS}$ computed by the convex combination of M output functions as

$$\mathbf{S}_{TS}(\mathbf{x}) = \frac{\sum_{i=1}^{M} \alpha_i(\mathbf{x}) \cdot \mathbf{u}_i(\mathbf{x})}{\sum_{j=1}^{M} \alpha_j(\mathbf{x})}. \tag{5.2}$$

When in a rule base $\mathbf{u}_i$'s are all polynomials of the same order, the TS system is said to have the same order. When the components of the output function are all constant, TS rules are said to be of 0-th order. TS rules have also found many application in control, forecasting, function approximation and pattern recognition [32, 191].

For pattern classification tasks the situation is different. Here the output is not a continuous variable, but a class label. The requirement here is to find information granulation in the input space and associate each of the granules with a crisp class label. In this case, MA and TS type of rules are not appropriate. The fuzzy rules for classification task should have a categorical label value in its consequent part. Thus the fuzzy rules for classification are of the type [59]:

$$R_i : \textbf{If } x_1 \textbf{ is } A_{i1} \textbf{ AND } x_2 \textbf{ is } A_{i2} \textbf{ AND } \cdots \textbf{ AND } x_p \textbf{ is } A_{ip} \textbf{ then class is } j.$$

Since output of such rules are class labels, even though they bear an apparent similarity with TS rules of 0-th order, aggregation of outputs using (5.2) is not meaningful. In this case we need a different strategy to compute the rule base output. For classification,

when a sample data point $\mathbf{x} \in \Re^p$ is presented to the system, the *firing strength* α_i of each rule is computed. For a given $\mathbf{x}$, if the firing strength of a rule is α, and the rule represents, say class k, then one can interpret α as the degree of support that $\mathbf{x}$ *is in class k*. The simplest decision rule that can be used is:

$$\begin{aligned} \text{Decide } \mathbf{x} \quad &\in \quad \text{Class } k, \\ \text{If } \alpha_j(\mathbf{x}) = \underbrace{\arg\max}_{i}\{\alpha_i\} \quad &\text{and} \quad k = \mathit{consequent}(R_j). \end{aligned}$$

Note that, in the literature use of MA type rules for classification is also suggested where the rule consequents are fuzzy sets defined on [0,1] and interpreted as certainty (not class labels) [191]. For such systems the center of gravity type defuzzification has been used but this is conceptually not attractive. We shall explain the issue with the help of an example:

Suppose, for a data point two rules R_1 and R_2 are fired. The firing strength of R_1 is 1.0 and that of R_2 is 0.2. The consequent of R_1 is a fuzzy set **very low** with *high specificity*, while the consequent of R_2 is **very high** with *low specificity*. In this case, although the output certainty should be close to **very low**, due to the effect of COG type defuzzification, it will be heavily biased towards **very high**. This is counter-intuitive. This happens, because the output linguistic variable is artificial and the defuzzification scheme is not appropriate (certainty is not an additive concept in context of a rule-base).

In view of the above reason, in this thesis we use the rules where the consequents are class label, i.e., the rule type due to Chiu [59] as described above.

5.2.2 Design issues

For designing a fuzzy rule based classifier there are three issues that need to be addressed:

I_1 How many rules are needed?

I_2 How to generate the rules?

I_3 How to use the rules to decide a class?

The simplest way of tackling the above issues is to take the help of a domain expert and create the fuzzy rules to represent his/her domain knowledge. But in a typical pattern classification problem, such domain knowledge is usually not available. So a scheme is needed for designing fuzzy rule based classifiers based on the training samples. In other words, one need to extract the rules from the training data.

Two major approaches to fuzzy rule extraction can be identified, namely, *decision tree based approach* and *prototype based approach*. The tree based approach was introduced by Chang and Paladins in their famous paper [48] in 1977. In such a classifier the number of rules are equal to the number of leaf nodes. Subsequently many researchers [56, 141, 234, 344] developed methods of fuzzy rule extraction from data. In general, these methods produce interpretable rules and often are capable of handling patterns with both numerical and non-numerical features. Typically, first the tree is generated and then from the tree a set of rules are generated. When rules are generated from the tree, all rules may not use all the input variables in the antecedent.

The prototype based approaches try to exploit geometric properties of the data. This approach can be regarded as special case of more general *clustering based approach to rule extraction* from data, which can be also used for function approximation and control, apart from classifier design. Actually, pattern classification tasks can be thought as a special case of function approximation task, where the output variable has discrete values. There are many theoretical studies available on the *universal approximation* (UA) properties of fuzzy rule based systems. A good review of such works can be found in [186]. In [192] Kuncheva discussed specifically the UA properties of fuzzy rule based classifiers. Though such results provide theoretical assurance about the quality of fuzzy rule based systems, they help little in the way of actually designing a system. In the next paragraph we briefly mention some of the approaches to designing the rule base.

Given a set of input and output observations, extracting fuzzy rules involves creating fuzzy partitions in the input and output space by associating fuzzy sets with each variable. There are chiefly two ways of doing it: data independent partition and data dependent partition. In the former, the input space is partitioned in a predetermined grid-like fashion, i.e., domain of each input feature is divided into a predetermined number of fuzzy sets, so that the input space is partitioned into a number of fuzzy hyperboxes (each corresponding to a rule). The partition of output space is then performed in a supervised manner. In [343] Wang and Mendel adopt this approach. Despite being very simple and highly interpretable, this approach suffers from two drawbacks. 1) The information in the training set is not fully exploited. 2) The scheme suffers from "curse of dimensionality", i.e., with increase in dimension, the number of fuzzy rules increases exponentially. However, in [137] Ishibuchi et al. used such a grid type partition for high dimension data, where they utilized "don't care" as an antecedent fuzzy set and thereby retaining only a small set of useful rules. They used genetic algorithm to find a set of good rules. In the other approach the partitions are dictated by the structure of the data. They generally involve first learning the structure of the data and then creating partitions. In [254] ensemble of rule tables with different fineness of granulation is used. A grade of certainty was associated with the rules during training. Finally, the rules from

the multi-table are pruned using the certainty grades. Abe and Lan [3] extracted rules from data by means of creating action hyperboxes and inhibition hyperboxes in the input space. In [4] Abe et al. developed classifiers with ellipsoidal regions in the input space, which are initially found through a clustering algorithm. Then the membership functions are tuned by changing their slopes in a supervised manner. Since there are a large number of parameters to be learned during the data dependent rule extraction, neural network based methods (also known as neuro-fuzzy systems) [52, 153, 149, 215, 218] and genetic algorithm [57, 136, 128, 138, 350] have also been used for fine tuning the rule bases.

SOM has been used for generation of fuzzy rules by Chi et al. in [54] and [55] for classification of handwritten numeric characters. Their approach involved training an SOM with the data to obtain the prototypes. Then each prototype is labelled and converted into a fuzzy rule using triangular membership function for each linguistic variable according to the method proposed by Dickerson and Kosko [84]. The number rules are reduced by merging narrowly separated fuzzy regions. In a typical example reported in [55] they used a 30×30 SOM and generated 809 fuzzy rules.

5.3 Designing fuzzy rule based classifiers

In this section we develop a prototype based method for designing a fuzzy rule based classifier. The scheme tries to achieve computational efficiency by using a *small number of rules that is enough to span the data distribution and representing class structures.* Here the prototypes are generated using the SOM-based prototype generation method DYNAGEN described in Section 4.3 of the previous chapter. We have already observed that the prototypes obtained from DYNAGEN show very good performance when used in a nearest prototype classifier. Hence these prototypes can be used to generate an initial set of rules which can be further fine tuned to obtain a good fuzzy rule based system. We use a gradient descent based technique of fine tuning both the peak and width of each fuzzy set.

5.3.1 Generating the fuzzy rule base

A prototype (representing a cluster of points) $\mathbf{v}_i$ for class k can be translated into a fuzzy rule of the form :

$$R_i\text{: If } \mathbf{x} \text{ is CLOSE TO } \mathbf{v}_i \text{ then the class is } k.$$

Where the fuzzy set "CLOSE TO" can be represented by a multidimensional membership function such as

$$\mu_{CLOSETO}(\mathbf{x}) = \exp^{-\frac{||\mathbf{x}-\mathbf{v}_i||^2}{{\sigma_i}^2}}, \tag{5.3}$$

where $\sigma_i > 0$ is a constant. This is equivalent to using prototypes with hyperspherical zones of influence centered at $\mathbf{v}_i$s. The 1-MSP (most similar prototype) classifier uses such membership values and it has been studied in [204] and described in the previous chapter. Such a classifier does not perform quite well when different features have considerably different variances.

To overcome this shortcoming, "$\mathbf{x}$ is CLOSE TO $\mathbf{v}_i$" can be written as a conjunction of p atomic clauses :

$$x_1 \text{ is CLOSE TO } v_1 \text{ AND } \cdots \text{ AND } x_p \text{ is CLOSE TO } v_p.$$

So the i-th rule R_i representing one of the c classes takes the form

$$R_i : x_1 \text{ is CLOSE TO } v_{i1} \text{ AND } \cdots \text{ AND } x_p \text{ is CLOSE TO } v_{ip} \text{ then class is } k.$$

Note that, this is just one possible interpretation of "$\mathbf{x}$ is CLOSE TO $\mathbf{v}_i$". The first form requires a multidimensional membership function while the second form requires several one dimensional membership functions and a conjunction operator. In general, the two forms will not produce the same output because they are not exactly equivalent. Depending on the choice of the membership function and the conjunction operator, the forms may lead to the same output.

The fuzzy set CLOSE TO v_{ij} can be modelled by a triangular, trapezoidal or Gaussian membership function. In this investigation, we use the Gaussian membership function,

$$\mu_{ij}(x_j; v_{ij}, \sigma_{ij}) = \exp -(x_j - v_{ij})^2/{\sigma_{ij}}^2. \tag{5.4}$$

Given a data point $\mathbf{x}$ with unknown class, we first find the firing strength of each rule. Let $\alpha_i(\mathbf{x})$ denote the firing strength of the i-th rule on a data point $\mathbf{x}$. We assign the point $\mathbf{x}$ to class k if $\alpha_r = \max_i(\alpha_i(\mathbf{x}))$ and the r-th rule represents class k. The computational method used for the firing strength $\alpha_i(\mathbf{x})$ depends on the choice of the conjunction operator (for AND operation).

The performance of the classifier depends crucially on the adequacy of the number of rules used and proper choice of the fuzzy sets used in the antecedent part of the rules. In our case each fuzzy set is characterized by two parameters v_{ij} and σ_{ij}. Let the initial set

of fuzzy rules be $R^0 = \{R_i^0 \mid i = 1, 2, \ldots, \hat{c}\}$. The parameters v_{ij}^0 and σ_{ij}^0 for fuzzy sets in the antecedent part of a rule $R_i^0 \in R^0$ are obtained from the prototype $\mathbf{v}_i^{final} \in V^{final}$ as follows:

$$v_{ij}^0 = v_{ij}^{final} \tag{5.5}$$

$$\sigma_{ij}^0 = k_w(\sqrt{(\sum_{\mathbf{x}_k \in X_i} (x_{kj} - v_{ij}^{final})^2)})/|X_i|, \tag{5.6}$$

where X_i is the set of training data closest to $\mathbf{v}_i^{final}$ and $k_w > 0$ is a constant parameter that controls the initial width of the membership function. If k_w is small, then specificity of the fuzzy sets defining the rules will be high and hence each rule will model a small area in the input space. On the other hand, a high k_w will make each rule cover a bigger area. Since the spreads are tuned, in principle k_w should not have much impact on the final performance, but in practice the value of k_w may have a significant impact on the classification performance for complicated data sets because of the local minimum problem of gradient descent techniques. In the current work the values of k_ws are found experimentally. One can use a validation set for this.

The initial rule base R^0 thus obtained can be further fine tuned to achieve better performance. But the exact tuning algorithm depends on the conjunction operator (implementing AND operation for the antecedent part) used for computation of the firing strengths. The firing strength can be calculated using any T-norm [32]. Use of different T-norms result in different classifiers. The **product** and the **minimum** are among most popular T-norms used as conjunction operators. Using product, the firing strength of the r-th rule is computed as follows:

$$\alpha_r(\mathbf{x}) = \Pi_{j=1}^{j=p} \mu_{rj}(x_j; v_{rj}, \sigma_{rj}). \tag{5.7}$$

and the same when computed using the minimum is

$$\alpha_r = \min_j \{\mu_{rj}(x_j; v_{rj}, \sigma_{rj})\}. \tag{5.8}$$

Clearly it is much easier to formulate a calculus based tuning algorithm if product is used.

In the current study we design two different classifiers one using product and the other using the **Softmin** operator. We shall see that the softmin operator enables us to realize a novel context-sensitive inferencing scheme.

5.3.2 Tuning the rule base

Let $\mathbf{x} \in X$ be from class c and R_c be the rule from class c giving the maximum firing strength α_c for $\mathbf{x}$. Also let $R_{\neg c}$ be the rule from the incorrect classes having the highest firing strength $\alpha_{\neg c}$ for $\mathbf{x}$.

We define the error function E as follows:

$$E = \sum_{\mathbf{x} \in X} E_{\mathbf{x}} = \sum_{\mathbf{x} \in X} (1 - \alpha_c(\mathbf{x}) + \alpha_{\neg c}(\mathbf{x}))^2. \tag{5.9}$$

This kind of error function has been used by Chiu [58] also. With a view to minimizing E, we consider the instantaneous error $E_{\mathbf{x}}$. Note that, $E_{\mathbf{x}}$ is a function of the parameters v_{cj}, $v_{\neg cj}$ and σ_{cj}, $\sigma_{\neg cj}$ of the two rules R_c and $R_{\neg c}$. The tuning algorithm will refine the rules with respect to their contexts.

Here the index j corresponds to clause number in the corresponding rule, i.e., for the first antecedent clause $j = 1$, for the second clause $j = 2$ and so on. The tuning process is repeated until the rate of decrement in E becomes negligible or a maximum number of iterations are over. Next we give an algorithmic description of the rule refinement algorithm when product is used to compute the firing strength.

Rule refinement (context-tuning) algorithm:

Begin

- Choose learning parameters η_m and η_s.
- Choose a parameter reduction factor $0 < \varepsilon < 1$.
- Choose the maximum number of iteration $maxiter$.
- Compute the error E_0 for the initial rule base R^0.
- Compute the misclassification M_0 Corresponding to initial rule base R^0.
- $t \leftarrow 1$
- While $(t \leq maxiter)$ do
  - For each vector $\mathbf{x} \in X_{Tr}$ (The training set)
    - Find the rules R_c and $R_{\neg c}$ using α_c and $\alpha_{\neg c}$.

* Modify the parameters of rules R_c and $R_{\neg c}$ as follows:
* For $k = 1$ to p do

(A) $v_{ck}^{new} = v_{ck}^{old} - \eta_m \frac{\partial E}{\partial v_{ck}^{old}} = v_{ck}^{old} + \eta_m(1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_c}{\sigma_{ck}^{old^2}}(x_k - v_{ck}^{old})$

(B) $v_{\neg ck}^{new} = v_{\neg ck}^{old} - \eta_m \frac{\partial E}{\partial v_{\neg ck}^{old}} = v_{\neg ck}^{old} - \eta_m(1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_{\neg c}}{\sigma_{\neg ck}^{old^2}}(x_k - v_{\neg ck}^{old})$

(C) $\sigma_{ck}^{new} = \sigma_{ck}^{old} - \eta_s \frac{\partial E}{\partial \sigma_{ck}^{old}} = \sigma_{ck}^{old} + \eta_s(1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_c}{\sigma_{ck}^{old^3}}(x_k - v_{ck}^{old})^2$

(D) $\sigma_{\neg ck}^{new} = \sigma_{\neg ck}^{old} - \eta_s \frac{\partial E}{\partial \sigma_{\neg ck}^{old}} = \sigma_{\neg ck}^{old} - \eta_s(1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_{\neg c}}{\sigma_{\neg ck}^{old^3}}(x_k - v_{\neg ck}^{old})^2$

* End For

– End For

– Compute the error E_t for the new rule base R^t.

– Compute the misclassification M_t for R^t.

– If $M_t > M_{t-1}$ or $E_t > E_{t-1}$ then

* $\eta_m \leftarrow (1 - \varepsilon)\eta_m$
* $\eta_s \leftarrow (1 - \varepsilon)\eta_s$
* $R^t \leftarrow R^{t-1}$
* End If

Comment:*If the error is increased, then possibly the learning coefficients are too large. So, decrease the learning coefficients, and retain* R^{t-1}.

– If $M_t = 0$ or $E_t = 0$ then

* Stop.

– End If

– $t \leftarrow t + 1$

- End While

End

At the end of the rule base tuning we get the final rule base R_{final} which is expected to give a very low error rate.

Since a Gaussian membership function is extended to infinity, for any data point all rules will be fired to some extent. In our implementation, if the firing strength is less than a threshold, ϵ (≈ 0.01), then the rule is not assumed to be fired. The threshold is set considering approximate 2σ limit of the Gaussian membership functions. Thus, under this situation, the rule base extracted by the system may not be complete with respect to the training data. This can happen even when we use membership functions

with triangular or trapezoidal shapes. This is not a limitation but a distinct advantage, although for the data sets we used, we did not encounter such a situation. If no rule is fired by a data point, then that point can be thought of as an outlier. If such a thing happens for some test data, then that will indicate an observation not close enough to the training data and consequently no conclusion should be made about such test points.

5.3.3 *Softmin* as a conjunction operator

Though product is a valid T-norm and has some attractive mathematical properties, its use is conceptually somewhat unattractive. To illustrate the point let us consider a rule having two atomic clauses in its antecedent. If the two clauses have truth values a and b, then intuitively the antecedent is satisfied at least to the extent of $\min(a, b)$. However, if product is used as the conjunction operator, we always have $ab \leq \min(a, b)$. Thus we always under-determine the importance of the rule. This does not cause any problem for non-classifier fuzzy systems as the defuzzification operator usually performs some kind of normalization with respect to the firing strength. But in classifier type applications a decision may appear to be taken with very low confidence, when actually it is not the case. Further, in certain cases, such as under evidence theory framework used in next chapter, this may lead to overemphasis on total ignorance. For example, if each antecedent clause is satisfied to the extent 0.9 and there are 10 antecedent clauses, the firing strength becomes $0.9^{10} = 0.3487$! Thus to avoid the use of product and at the same time to be able to apply calculus to derive update rules we use a softmin operator.

The **soft-match** of n positive number $x_1, x_2, ..., x_n$ is defined by

$$SM(x_1, x_2, ..., x_n, q) = \left\{ \frac{(x_1^q + x_2^q + ... + x_n^q)}{n} \right\}^{1/q}.$$

where q is any real number. SM is known as an aggregation operator with upper bound of value 1 when $x_i \in [0, 1] \forall i$. This operator is used by different authors [91, 263] for different purposes. It is easy to see that

$$\lim_{q \to \infty} SM(x_1, x_2, ..., x_n, q) = \max(x_1, x_2, ..., x_n)$$

and

$$\lim_{q \to -\infty} SM(x_1, x_2, ..., x_n, q) = \min(x_1, x_2, ..., x_n).$$

Thus we define the softmin operator as the soft-match operator with a sufficiently negative value of the parameter q. The firing strength of the r-th rule computed using softmin is

$$\alpha_r(\mathbf{x}) = \left\{ \frac{\sum_{j=1}^{j=p} (\mu_{rj}(x_j; v_{rj}, \sigma_{rj}))^q}{p} \right\}^{1/q}. \tag{5.10}$$

In the current study we use $q = -10.0$.

Using the same error function E, as in the previous section we derive the rule update equations L, M, N, and O bellow.The tuning algorithm remains the same except equations A, B, C and D are replaced by L, M, N, and O respectively.

$$\text{(L) } v_{ck}^{new} = v_{ck}^{old} - \eta_m \frac{\partial E}{\partial v_{ck}^{old}} = v_{ck}^{old} + \eta_m (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_c}{\sum_{j=1}^{j=p} \mu_{cj}^q} \frac{\mu_{cj}^q}{\sigma_{ck}^{old^2}} (x_k - v_{ck}^{old})$$

$$\text{(M) } v_{\neg ck}^{new} = v_{\neg ck}^{old} - \eta_m \frac{\partial E}{\partial v_{\neg ck}^{old}} = v_{\neg ck}^{old} - \eta_m (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_{\neg c}}{\sum_{j=1}^{j=p} \mu_{\neg cj}^q} \frac{\mu_{\neg cj}^q}{\sigma_{\neg ck}^{old^2}} (x_k - v_{\neg ck}^{old})$$

$$\text{(N) } \sigma_{ck}^{new} = \sigma_{ck}^{old} - \eta_s \frac{\partial E}{\partial \sigma_{ck}^{old}} = \sigma_{ck}^{old} + \eta_s (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_c}{\sum_{j=1}^{j=p} \mu_{cj}^q} \frac{\mu_{cj}^q}{\sigma_{ck}^{old^3}} (x_k - v_{ck}^{old})^2$$

$$\text{(O) } \sigma_{\neg ck}^{new} = \sigma_{\neg ck}^{old} - \eta_s \frac{\partial E}{\partial \sigma_{\neg ck}^{old}} = \sigma_{\neg ck}^{old} - \eta_s (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_{\neg c}}{\sum_{j=1}^{j=p} \mu_{\neg cj}^q} \frac{\mu_{\neg cj}^q}{\sigma_{\neg ck}^{old^3}} (x_k - v_{\neg ck}^{old})^2$$

The use of softmin is also consistent with our perception of AND connective in the antecedent parts of the fuzzy rules. Since for reasonably big negative value (such as -10.0) of q the softmin approaches direct min, when the trained system is used for testing/deployment, direct min may be used for computational benefit.

5.4 Context-sensitive inferencing

The use of softmin operator opens up a host of theoretical possibilities. The important fact to be noted is that the softmin operator is just one member of the family of aggregation operators generated by the **soft-match** operator for $q \in [-\infty, \infty]$. The family of operators covers a large spectrum from *minimum* to *maximum* including the *average* (for $q = 1$). Figure 5.1 shows that softmin varies from the minimum of its arguments to their maximum via the average.

In fuzzy rule based systems for pattern classification tasks, we use rules of the form

$$R_i : \text{If } x_1 \text{ is } A_{i1} \text{ AND } \cdots \text{ AND } x_p \text{ is } A_{ip} \text{ then class is } j,$$

Even if we use a tunable conjunction operator typically all rules in a system use the same conjunction operator. We can raise a fundamental question at this point. Is it really necessary to have the same conjunction operator for all rules in a system? It is very difficult to have a definite answer. A rule is considered to be a tool for reasoning in a small area of input feature space, each rule can be thought as a different *context of reasoning*. Thus drawing an analogy with the reasoning of human experts, we can recognize the possibility that within the same system there could be rules using different conjunction operators. So a system may contain some rules for which minimum is the

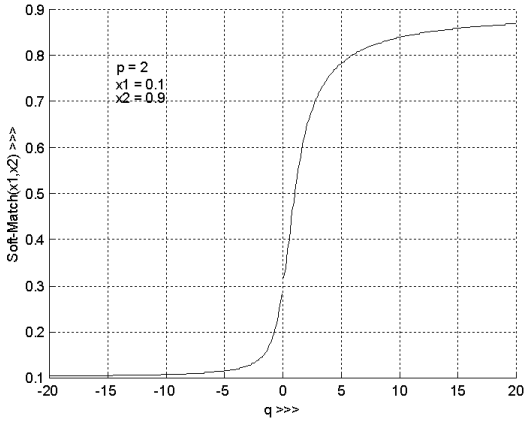

Figure 5.1: Plot of soft-match operator against q.

appropriate conjunction operator while there could be others whose firing strength is larger than the minimum of the membership values of atomic propositions. There could even be some rules whose conjunction operator are closer in spirit to the maximum. This leads to a concept called *context-sensitive inferencing*. Human being often do context-sensitive inferencing. Depending on the cost involved with a decision an expert may adopt different level of conservatism in inferencing [263].

Thus while designing a scheme for rule extraction from the data, if the conjunction operator for each rule can also be learnt from the data, the resulting system is expected to achieve better performance.

In our present scheme we use the softmin as the conjunction operator. As mentioned earlier, it can act as different conjunction operators for different values of its parameter q. So we can calculate the firing strength for the r-th rule R_r as

$$\alpha_r(\mathbf{x}) = \left\{ \frac{\sum_{j=1}^{j=p} (\mu_{rj}(x_j; v_{rj}, \sigma_{rj}))^{q_r}}{p} \right\}^{1/q_r}. \tag{5.11}$$

i.e., q_r is the parameter for conjunction operator corresponding to the r-th rule. The change in the value of the parameter q_i changes the nature of inferencing implemented by rule R_i. Therefore, we call the parameter q_i the *inferencing parameter* of rule R_i. Also for the sake of clarity, in places where distinction is needed, we shall call the parameters controlling the position and spread of fuzzy sets v_{ij} and σ_{ij}s as *rule parameters* or *rule base parameters*. Hence, the error function, as a function of inferencing parameters q_is can be written as

$$E(q_1, q_2, \cdots q_{|R|}) = \sum_{\mathbf{x} \in X} E_{\mathbf{x}} = \sum_{\mathbf{x} \in X} (1 - \alpha_c(\mathbf{x}) + \alpha_{\neg c}(\mathbf{x}))^2, \tag{5.12}$$

where R is the set of rules. Now we can use calculus to formulate an update scheme for q_is for reducing the error function. The algorithm for tuning consequent operators is given bellow.

The conjunction operator (inferencing parameter) refinement algorithm:

Begin

- Choose learning parameter η_q.
- Choose a parameter reduction factor $0 < \varepsilon < 1$.
- Choose the maximum number of iteration $maxiter$.
- Compute the error E_0 for the initial rule base R^0.
- $t \leftarrow 1$
- While $(t \leq maxiter)$ do
  - For each vector $\mathbf{x} \in X$
    - Find the rules R_c and $R_{\neg c}$ using α_c and $\alpha_{\neg c}$.
    - Modify the parameters q_c and $q_{\neg c}$ of rules R_c and $R_{\neg c}$ respectively as follows:

$$q_c^{new} = q_c^{old} - \eta_q \frac{\partial E}{\partial q_c^{old}} = q_c^{old} + \eta_q (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_c}{q_c} \left(\frac{\sum_{j=1}^{j=p} \mu_{cj}^{q_c} \ln \mu_{cj}}{\sum_{j=1}^{j=p} \mu_{cj}^{q_c}} - \ln \alpha_c \right)$$

$$q_{\neg c}^{new} = q_{\neg c}^{old} - \eta_q \frac{\partial E}{\partial q_{\neg c}^{old}} = q_{\neg c}^{old} - \eta_q (1 - \alpha_c + \alpha_{\neg c}) \frac{\alpha_{\neg c}}{q_{\neg c}} \left(\frac{\sum_{j=1}^{j=p} \mu_{\neg cj}^{q_{\neg c}} \ln \mu_{\neg cj}}{\sum_{j=1}^{j=p} \mu_{\neg cj}^{q_{\neg c}}} - \ln \alpha_{\neg c} \right)$$

  - End For
  - Compute the error E_t for the modified rule base R^t.
  - If $E_t > E_{t-1}$ then
    - $\eta_q \leftarrow (1 - \varepsilon)\eta_q$
    - $R^t \leftarrow R^{t-1}$

      Comment:*If the error is increased, then possibly the learning coefficients are too large. So, decrease the learning coefficients, and retain* R^{t-1}.
  - End If
  - $t \leftarrow t + 1$

- End While

End

The initial set of rules used in this algorithm can be the set obtained from the tuning algorithm described in the previous section. Thus, the earlier tuning scheme finds a suitable context, i.e., the rule parameters for each rule using softmin and then we tune the inferencing scheme depending on the context. In the new algorithm, unlike the previous two, the stress is put on the reduction of total error as defined in Eq. (5.12). The algorithm starts with the same inferencing parameter in the soft-match operators for all rules and then the operator for each rule is tuned separately.

5.5 Implementation and results

The classifier designing scheme proposed in this chapter is a general one and can be used for any pattern classification task with object data. However, the major motivation of developing this scheme is to design classifiers for complex data sets while using a small training set for the design purpose. Though the 1-NMP and 1-MSP classifiers proposed in the previous chapter work well with several data sets, their performance for complex data sets like Satimage2 and Satimage3 (Section 4.4.3) can be further improved. Thus in this chapter we conduct detailed experiments with the data sets Satimage1 (from Group B data sets in previous chapter), Satimage2 and Satimage3. Before that, we present the performance of fuzzy rule based classifiers for other data sets (i.e., the group A data sets and the group B data sets excluding Satimage1) studied in Chapter 4.

5.5.1 Classification of group A and group B data sets

The classifiers reported here employ context-free reasoning for both variants of rules proposed in this chapter, namely, *product rules* using Eq. 5.7 as conjunction operator and *softmin rules* using Eq. 5.10 as conjunction operator. A detailed description of these data sets and the training-test partitions are provided in Section 4.3.3. Here we use the *same* training-test partitions.

The initial fuzzy rule base for each of the data sets is derived from the corresponding prototype sets used to build the 1-NMP classifiers as reported Chapter 4. Thus the number of rules for a classifier is the same as the number of prototypes in corresponding 1-NMP classifiers. The designing of the rule base involves a very few parameters and we have used the same values of parameters for all data sets. Strictly speaking, designing

Table 5.1: Performance of fuzzy rule based classifiers for group A data sets

Data set	No. of Rules	% of error					
		Product Rules			Softmin Rules		
		Trng.	Test	Avg.	Trng.	Test	Avg.
Iris	5	2.67	2.67		1.33	4.00	
	5	2.67	6.67	4.67	0.00	4.00	4.00
Glass	28	38.10	53.21		22.86	35.78	
	26	19.27	34.29	43.75	24.77	35.24	35.51
Breast	4	14.08	12.63		6.69	3.86	
Cancer	5	14.74	16.90	14.77	3.86	8.45	6.16
Normalized	8	7.04	6.32		4.93	4.91	
B. Cancer	4	5.96	7.75	7.04	2.46	7.75	6.33
Vowel	21	18.20	19.22		16.59	17.62	
	15	18.31	19.35	19.29	16.70	19.59	18.61
Norm	4	4.75	4.25		3.25	3.75	
	4	3.25	4.75	4.50	3.50	5.25	4.50
Two-Dishes	2	0.00	0.00		0.00	0.00	
	2	0.00	0.00	0.00	0.00	0.00	0.00

Table 5.2: Performance of fuzzy rule based classifiers for group B data sets

Data set	No. of Rules	% of error			
		Product Rules		Softmin Rules	
		Trng.	Test	Trng.	Test
Cone-Torus	12	14.57	19.40	15.58	14.43
Normal Mixture	4	11.60	10.00	13.66	9.40
Phoneme	5	19.60	20.39	20.80	21.24

of the rule based classifier involves only three parameters, the two learning coefficients η_m, η_s and k_w. An appropriate value of k_w may expedite the learning, its optimal choice will depend on the structure in the data and can be determined by a few trials with the training set. However, the ultimate output is not much dependent on its value because it just initializes the spreads of the membership functions which are finally tuned using the training data. Thus, this parameter can be ignored, i.e., we can use k_w =1 for all data sets. The other two parameters are the learning coefficients. If we keep them fixed throughout the training, then their choice, like any other gradient search, will decide the local minima that the rule base settles at. However, in our tuning algorithms when the average error increases with the current learning rate we have an in-built mechanism of reverting to the previous rule base and reducing the learning rates. This makes the scheme much less sensitive to the choice of the learning parameters. We have used the same values of learning parameters ($\eta_m = 0.1$, η_s= 0.05 and k_w= 2.0) for all data sets.

The DYNAGEN algorithm discussed in Chapter 4 uses two parameters K_1 and K_2. Although K_1 and K_2 are not parameters of the fuzzy rule based systems, we discuss about their choices because the prototypes extracted by the DYNAGEN are used to initialize the rule base. Essentially, these two parameters are used to judge whether a prototype is adequately represented by training points with respect to the overall data structure as well as with respect to a class. Smaller values of K_1 and K_2 demand each prototype to be represented by a larger number of points and thus lower the number of prototypes, and hence the number of rules. Loosely speaking these two parameters implement a kind of lower bound for the concept of adequacy and hence it is possible to find a fixed set of values that work reasonably well with most data sets. That is what we did here. Based on a few trails, we suggested $K_1 = 3$ and $K_2 = 6$ for all data sets and used the same for all results reported.

The results for group A data sets are presented in Table 5.1. Results for both product rule classifiers as well as the softmin rule classifiers are included. For each data set, similar to the results reported in Table 4.3, two classifiers are built with the training and test partitions interchanged. The 5-th and 8-th columns in Table 5.1 report the average of test errors of two classifiers for each data set. Comparing with the 1-NMP classifiers, it can be seen that for all but the Glass data set, the fuzzy rule based classifiers, especially those using softmin rules show significantly better performance. Except for Glass the performance of the fuzzy rule based classifiers are fairly close to those of the corresponding 1-MSP classifiers (Table 4.5). The 1-MSP classification scheme, as reported earlier, did not work well for the Breast Cancer data set due to high variation of the range of its components. The fuzzy rule based classifiers could overcome this difficulty. The softmin rules based classifiers achieve significantly lower error rates (average 6.16%) for this data set compared to the 1-NMP classifiers (average 12.13%).

Table 5.3: Fuzzy rule based classifiers: results of 10-fold cross validation

Data Set	Product rules		Softmin rules	
	E_{tr} (%)	E_{te} (%)	E_{tr} (%)	E_{te} (%)
	Avg. (St. D.)	Avg. (St. D.)	Avg. (St. D.)	Avg. (St. D.)
Iris	3.78 (1.07)	6.00 (4.92)	2.37 (0.77)	4.00 (4.66)
Glass	22.80 (3.24)	42.38 (7.60)	20.37 (4.15)	37.62 (6.13)
N. B. Cancer	6.43 (1.69)	7.50 (3.05)	3.95 (0.92)	5.71 (1.84)
Vowel	17.82 (1.25)	18.74 (4.85)	16.93 (1.09)	20.46 (3.46)
Norm4	4.17 (0.30)	4.13 (1.67)	3.81 (0.29)	4.75 (2.19)
Two-Dishes	0.77 (0.10)	0.80 (0.88)	0.00 (0.00)	0.00 (0.00)
Cone-Torus	8.15 (1.14)	9.25 (3.94)	6.28 (0.47)	7.88 (2.13)
Normal Mixture	9.44 (0.42)	10.00 (2.45)	9.12 (0.28)	9.52 (2.81)
Satim1	14.96 (0.34)	14.74 (1.15)	14.32 (0.37)	14.53 (1.45)
Phoneme	20.52 (0.36)	20.69 (1.83)	18.15 (0.52)	19.26 (2.27)

The performance of the classifiers for three group B data sets is reported in Table 5.2. The training-test partitions are the same as those used in Table 4.4. Comparing Table 5.2 with the Table 4.4 and Table 4.7, it can be observed that for these data sets, the fuzzy rule based classifiers perform better than corresponding 1-NMP classifiers and their performance are comparable to those of corresponding 1-MSP classifiers.

5.5.2 Results of cross-validation experiments

We have also reported in Table 5.3 the results of 10-fold cross-validation experiments for both the group A and group B data sets. Comparing the results in Table 5.3 with cross-validation experiments for the 1-MSP classifiers in Table 4.12, we find that for the Iris data set the product rule classifier shows the same performance as the 1-MSP classifier while the softmin rule classifier exhibits better performance. For Glass and Normalized Breast Cancer the 1-MSP performs better but the performance of softmin rule classifier is fairly close. For both these data sets the performance of fuzzy rule based classifiers is more consistent because the standard deviation of error rates are much smaller than those of 1-MSP. For the Vowel and Norm4 data sets the product rule classifiers exhibit better performance while softmin rule classifiers have marginally higher error rates compared to 1-MSP classifiers. The product rule classifier shows a small error rate for the Two-dishes but the softmin rule classifier achieves zero error rate. For Cone-torus and Normal Mixture the 1-MSP classifiers perform marginally better than the product rule classifiers, but the softmin rule classifiers achieve lower error rates than the 1-MSP classifier. For Satimage1 and Phoneme data sets both fuzzy rule based classifiers achieve lower error

rates. It is interesting to note that for all group B data sets the softmin rule classifiers perform better than the 1-MSP classifiers.

5.5.3 Landcover classification from multispectral satellite images

Classification of multispectral satellite images is a very important field of application of pattern recognition techniques. Currently huge amount of information about the earth is routinely being generated by the satellite-based sensors. As mentioned in Chapter 4, this information is often captured by multispectral scanners, that acquire data at several distinct spectral bands producing multispectral images. Such data sets capture more information since a sensor operating in certain spectral region might be more sensitive to certain classes of objects than the others. Analysis of such data demands advanced techniques for data fusion and pattern recognition. This requirement led to development of numerous techniques for data fusion and classification. To name a few, statistical methods [309, 272], Dempster-Shafer theory [211], neural networks [14, 27, 36, 272] etc.

Many researchers have studied different fuzzy methodologies for landcover classification using multispectral satellite images. For example, in [101, 42] authors have used fuzzy c-means algorithm [31], Kumar et al. [190] have applied fuzzy integral based method. Fuzzy rule base has been used for classification by many researchers [32, 137] for diverse fields of application. Fuzzy rules are attractive because they are interpretable and can provide an analyst a deeper insight into the problem. There have been a few attempts to use fuzzy rule based systems for land cover analysis. In a recent paper Bárdossy and Samaniego [21] have proposed a scheme for developing a fuzzy rule based classifier for analysis of multispectral images. They employed simulated annealing for optimizing the performance of a randomly selected initial set of rules. In [188] Kulkarni and McCaslin have used a fuzzy neural network for rule extraction.

Analysis of satellite images has many important applications such as prediction of storm and rainfall, assessment of natural resources, estimation of crop yields, assessment of natural disasters, and landcover classification. In this work we focus on land cover classification from multispectral satellite images. We consider a set of independent detectors of a sensor, operating in different spectral bands and producing homogeneous data (i.e., same type of information, namely pixel values). Each pixel is represented by a vector, each component of which comes from a detector.

Satimage1 is prepared from a four channel Landsat image and consisting of 6435 pixels. So there are 6435 vectors in $\Re^4$. There are six (c=6) types of landcover classes as shown in Table 5.4. This is a benchmark data set and available on the web at [108]. Performance

Table 5.4: Different classes and their frequencies for Satimage1

Land-cover types	Frequencies
Red soil	1533
Cotton crop	703
Gray soil	1358
Damp gray soil	626
Soil with vegetation stubble	707
Very deep gray soil	1508
Total	6435

of many classifiers for this data set can be found in [191]. In the previous chapter also this data set (Satimage1 data set in group B) is used for testing the 1-NMP and 1-MSP classifiers.

For Satimage1 the data set X is partitioned into X_{Tr} and X_{Te} such that $X = X_{Tr} \cup X_{Te}$ and $X_{Tr} \cap X_{Te} = \phi$. The training-test partition used in [191] is available at [108]. The number of pixels of different land-cover types in the training set are : Red soil = 104; Cotton crop = 68; Gray soil = 108; Damp gray soil = 47; Soil with vegetation stubble = 58; Very deep gray soil = 115. We use this partition as our first partition and generate three more random partitions keeping the same number of representations from different classes in the training and test sets.

We have already mentioned in Chapter 4 that Satimage2 is a Landsat-TM image available along with full ground truth in the catalog of sample images of the ERDAS software. In Satimage2, each pixel is represented by a 7-dimensional feature vector. Each dimension of a feature vector comes from one channel. The class distribution of the pixels in Satimage2 is given in Table 5.5. Figure 5.2 shows the channel 1 image for Satimage1 and Figure 5.4 shows the ground truth for it where different classes are indicated with different colors.

Kumar et al. [190] studied Satimage2 using a number of techniques including the maximum likelihood classifier, artificial neural network and a fuzzy integral based method. The fuzzy integral based method is reported to produce the best performance in their study.

Satimage3 is also a Landsat-TM image. Different classes and their frequencies for Satimage3 are described in Table 5.6. Figure 5.3 shows the channel 4 image for Satimage3

Table 5.5: Classes and their frequencies in the Satimage2.

Classes	Frequencies
Forest	176987
Water	23070
Agriculture	26986
Bare ground	740
Grass	12518
Urban area	11636
Shadow	3197
Clouds	358
Total	262144

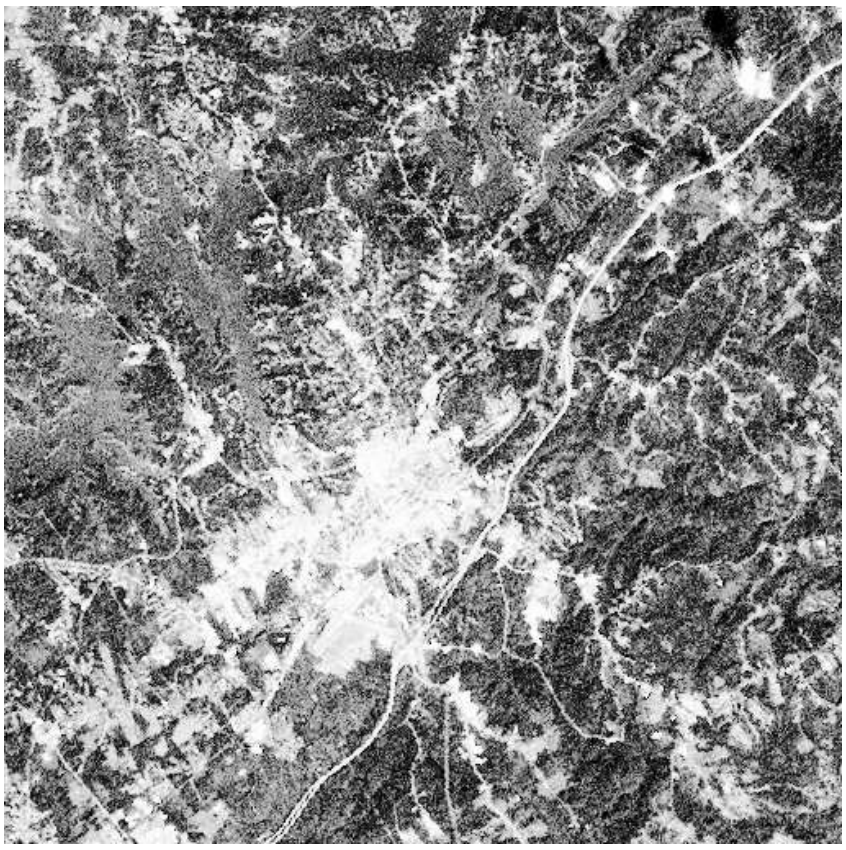

Figure 5.2: Band-1 of Satimage2 (After histogram equalization).

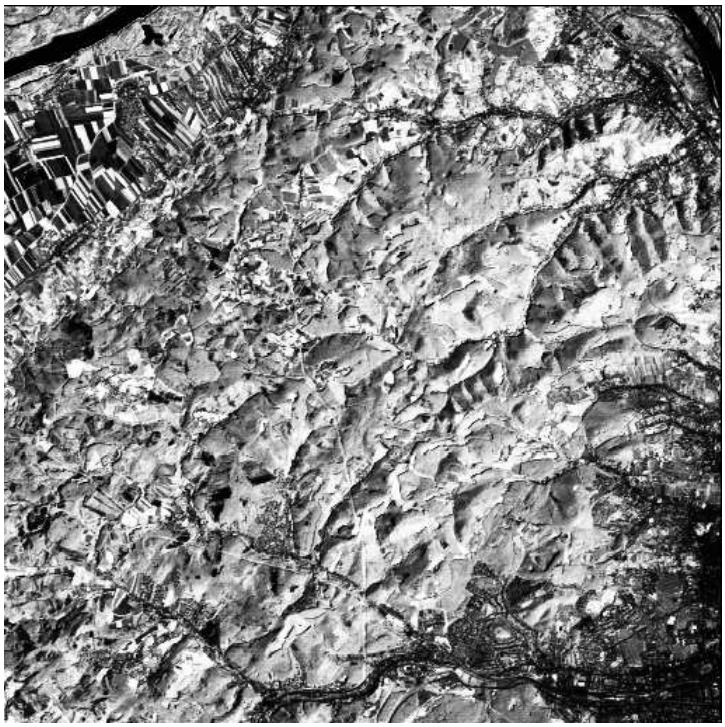

Figure 5.3: Band-4 of Satimage3 (After histogram equalization).

Table 5.6: Classes and their frequencies in the Satimage3.

Classes	Frequencies
Built-up land	48502
Forest	161522
Water	2643
Agriculture	48554
Total	261121

and Figure 5.5 shows the ground truth for it where different classes are indicated with different colors.

As described in Section 4.6.2, in our study we generated 4 sets of training samples for both Satimage2 and Satimage3. For Satimage2 each training set contains 200 data points randomly chosen from each class. For Satimage2 we include in each training set 800 randomly chosen data points from each of the four classes. The classifiers designed with the training data are tested on the whole images. The performance of 1-NMP classifiers and 1-MSP classifiers for these data sets are presented in Section 4.6.2.

We divide the remaining part of this section in two parts. In the first part (*context-free inferencing*, Section 5.5.4) we describe performance of two types of fuzzy rule based classifiers. The first type uses **product** as the conjunction operator. The other type

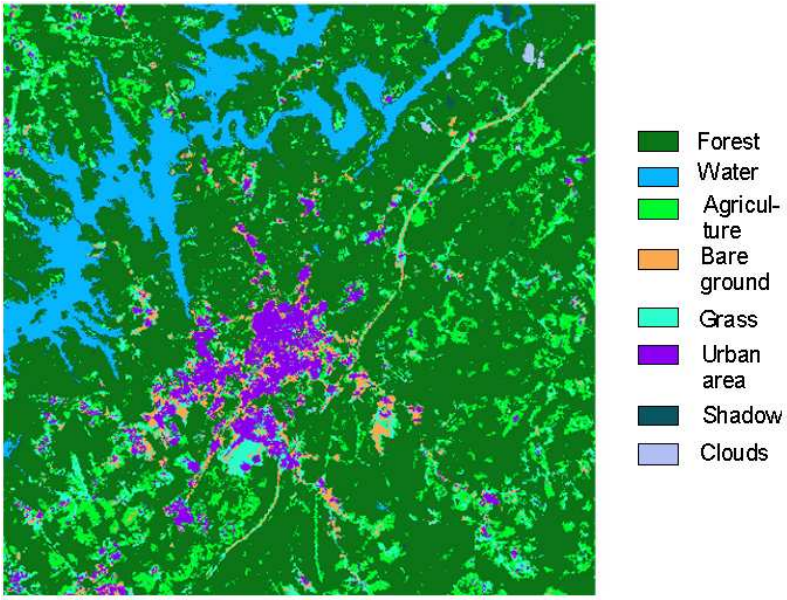

Figure 5.4: The ground truth for Satimage2. The classes are represented by different colors.

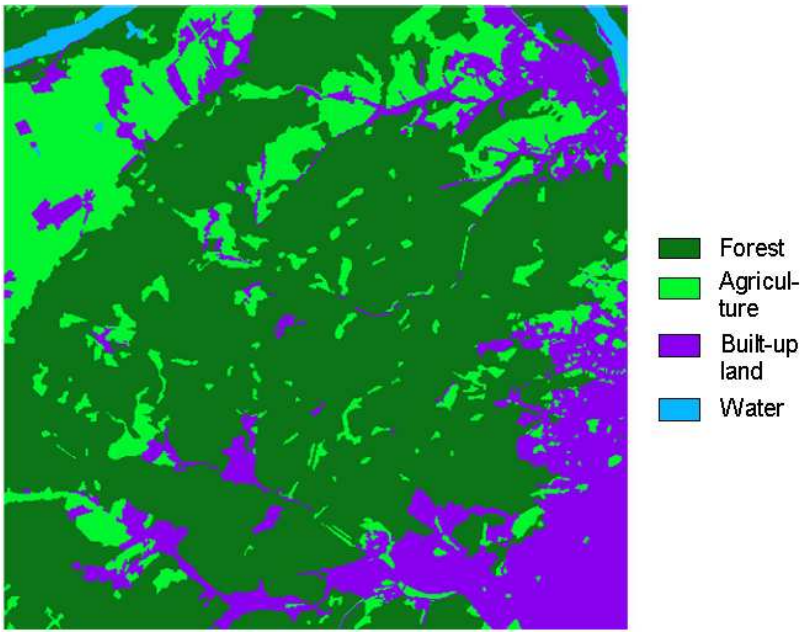

Figure 5.5: The ground truth for Satimage3. The classes are represented by different colors.

Table 5.7: Performance of the fuzzy rule based classifiers designed with context-free reasoning scheme for 4 different partitions of Satimage1 data set.

Partition Number	Number of Rules	k_w	% of Error			
			Product rules		Softmin rules	
			Trng.	Test	Trng.	Test
1	27	2.0	12.8%	15.51%	10.8%	15.58%
2	26	2.0	12.2%	15.6%	10.0%	16.16%
3	25	2.0	16.0%	15.26%	12.8%	15.92%
4	20	2.0	12.0%	17.1%	9.4%	16.29%

uses **softmin** as the conjunction operator with a constant q value for all rules in a classifier. In the current study all these classifiers use $q = -10$. In the second part (*context-sensitive inferencing*, Section 5.5.5) we report results using the context-sensitive inferencing scheme, i.e., different conjunction operator for different rules.

5.5.4 Performance of the classifiers with context-Free inferencing

Table 5.7 depicts the performance of the fuzzy rule based classifier on Satimage1. Both fuzzy rule based classifiers show consistently almost similar performances. In all cases the softmin based classifiers show some improvement in the training error. While for test sets the product based classifier shows slightly better performance in three cases and the softmin based classifier is better in the remaining case. Satimage1 has been extensively studied in [191]. Comparing our results in Table 5.8 with the results in [191] (reproduced in Tables 4.8 and 4.9 in previous chapter) we find that our classifier outperforms MLP of comparable sizes and at par with RBF networks. For example, in [191] the test error reported using an MLP is 23.08% while that by RBF networks varied between 14.52% - 15.52%.

In Table 5.8 we present the results on four different random partitions of Satimage2. It shows an excellent performance of the rule based classifiers. Both types (product based and softmin based) classifiers exhibit consistent and comparable performances. For all partitions the softmin based classifiers show lower training errors while for the whole data the product based classifier performs better in only one case. Figure 5.6 shows a typical classification result (corresponding to the first partition) which is almost identical to Figure 5.4.

Table 5.8: Performances of fuzzy rule based classifiers designed with context-free reasoning scheme for different training sets for Satimage2

Training Set	No. of rules	k_w	Product rules		Softmin rules	
			Error Rate in Training Data	Error Rate in Whole Image	Error Rate in Training Data	Error Rate in Whole Image
1.	30	5.0	19.3%	13.8%	12.0%	13.6%
2.	25	6.0	14.4%	13.6%	14.3%	14.47%
3.	25	5.0	16.5%	13.7%	12.0%	13.03%
4.	27	4.0	16.0%	13.8%	12.6%	12.5%

Table 5.9: Performances of fuzzy rule based classifiers designed with context-free reasoning scheme for different training sets for Satimage3

Training Set	No. of rules	k_w	Product rules		Softmin rules	
			Error Rate in Training Data	Error Rate in Whole Image	Error Rate in Training Data	Error Rate in Whole Image
1.	14	2.0	17.03%	15.41%	16.31%	14.14%
2.	14	2.0	16.75%	15.22%	16.30%	14.04%
3.	12	2.0	18.72%	14.63%	17.09%	14.01%
4.	11	2.0	18.59%	15.43%	17.34%	14.23%

The same data set (Satimage2) has been used by Kumar et al. [190] in a comparative study using several classification techniques. The best result obtained by them using a *fuzzy integral* based scheme achieves a classification rate of 78.15%. *In our case, even the worst performance is about 5% better than the results in* [190].

The performance of the fuzzy rule based classifiers for Satimage3 is summarized in Table 5.9. For this image, the softmin rules perform slightly better than the product rules for all four training sets. Performance of all classifiers are significantly better than 1-MSP classifiers using the same training sets which are reported in Table 4.16 in the previous chapter.

For Satimage3 the reported result [36] shows 84.7% accuracy with maximum likelihood classifier (MLC) and 85.9% accuracy with a neural network based classifier. In our case, for all training-test partitions the fuzzy rule based classifiers using product rules perform similar to the MLC classifiers and classifiers with softmin rules outperform the MLC and

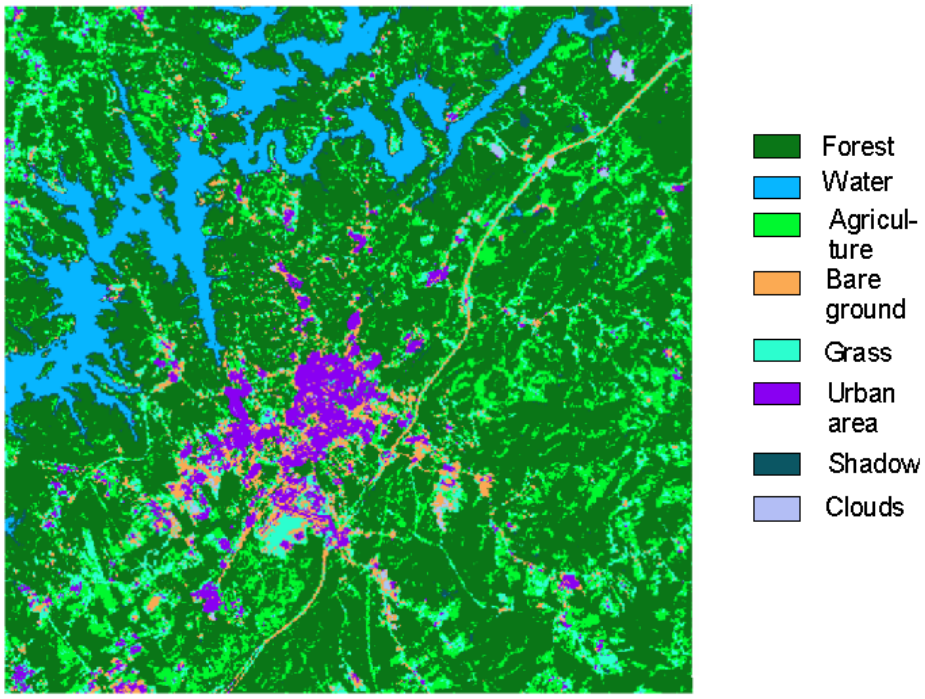

Figure 5.6: The classified image for Satimage2 (Training set 1). The classes are represented by different colors.

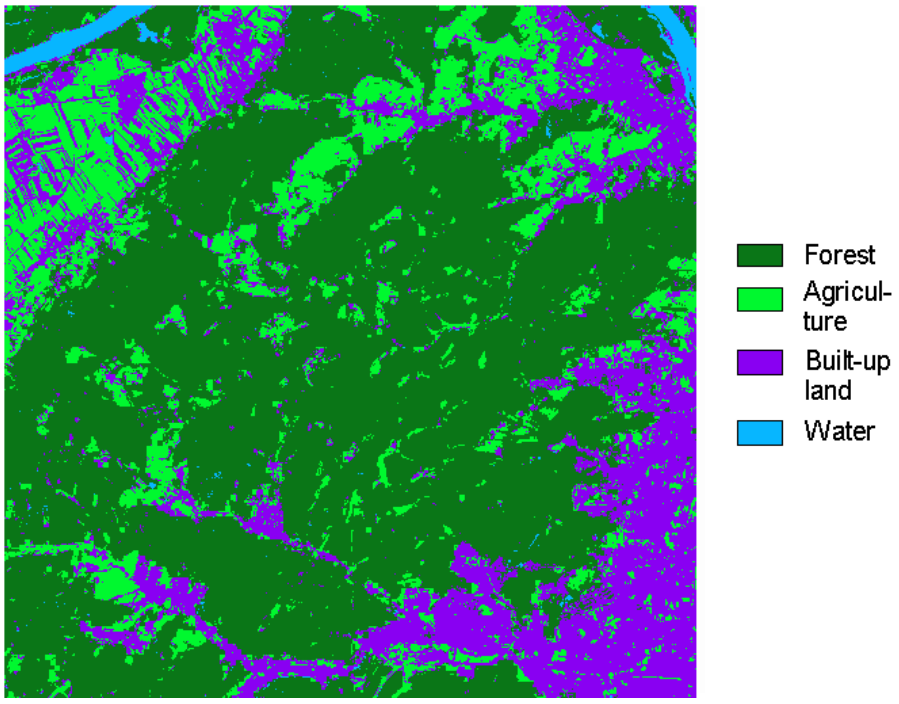

Figure 5.7: The classified image for Satimage3 (Training set 2). The classes are represented by different colors.

are at par with the result reported for neural network (actually for the second and third training sets our results depict marginal improvement). Note that, the training set used in [36] is considerably larger and the points are carefully chosen to represent different sub-categories of the data. It was also reported in [36] that though there are four classes, there exists a number of distinct sub-categories in each class. They identified through statistical analysis, twelve such subcategories in the data for which the spectral data can be represented approximately by normal distributions. Our method dynamically extracted nearly the same number (14, 14, 12 and 11 respectively) of fuzzy rules, which is consistent with the observation reported in [36]. This fact points to the effectiveness of the dynamic rule extraction procedure developed and used here. A typical classification result for Satimage3 using the softmin rules (Training set 1) is shown in Figure 5.7.

It can be observed from the above tables that in a few cases classifiers with fewer number of rules perform better than those with larger number of rules. This is due to the fact that each classifier is trained with different randomly generated training sets and there is some randomness involved in the SOM based prototype generation scheme. These result in different rule bases, where one with more number of rules may have a few rules in partial conflict.

In Table 5.10 we show the rules obtained for classes 1, 2 and 3 of Satimage2 using the training set 1. Column 1 of Table 5.10 shows the class number while column 2 lists the rule identification number. Column 3 describes the rules using the membership functions in the antecedent part of the rules. Since it is a seven dimensional data, each rule involves seven atomic antecedent clauses (fuzzy sets). Each fuzzy set is represented by a 2-tuple (μ_{ij}, σ_{ij}), where μ_{ij} and σ_{ij} are the center and spread of a Gaussian membership function modelling the j-th antecedent clause of i-th rule. Thus for clause 1 of rule number 2, the tuple $(\mu_{21}, \sigma_{21}) = (65.7, 11.9)$ represents a clause "gray value from channel 1 is CLOSE to 65.7" where the fuzzy set CLOSE to 65.7 is represented by a Gaussian function with center at 65.7 and spread 11.9. Inspection of the parameters of rules for class 1 reveals that between rule 2 and rule 6 all features change, but features 4, 5 and 7 changes significantly. Comparing rules 6 and 8 we find that features 3, 5, 6 and 7 change significantly. This indicates that each rule represents a distinct volume (fuzzy granule) in the feature space.

Table 5.10 also suggests that data from class 2 probably form a nice cluster in the feature space that can be modelled by just a single rule. This is further confirmed by the fact that the spread of the membership functions for features 2, 4, 5, 6 and 7 are relatively small.

Similarly, for class 3, different rules model different areas in the feature space. For rules 14 and 15 although μs for features 1 and 2 do not change much, the μs for other features

Table 5.10: The rules for classes 1, 2 and 3 with corresponding fuzzy sets for Satimage2. These rules are obtained using the training set 1.

Class	Rule No.	Fuzzy sets in form of (μ_{ij}, σ_{ij}) tuples, $j = 1, 2, \cdots, 7$
1	2	(65.7,11.9), (23.6,6.8), (22.8,10.2), (50.9,19.8) (41.7,16.2), (130.3,6.1), (13.9,12.4)
	6	(67.5,11.4), (24.8,7.0), (25.4,10.6), (56.5,22.1) (56.5,19.7), (131.9,4.8), (18.3,4.2)
	8	(68.8,11.7), (27.4,8.7), (30.7,12.2), (57.0,19.4) (62.1,24.8), (126.5,8.5), (7.9,11.9)
2	4	(65.5,12.1), (22.0,7.5), (17.1,9.4),(9.1,2.2) (4.9,6.7), (128.7,6.9), (2.1,8.2)
3	7	(67.1,13.3), (24.9,7.7), (20.6,11.3), (45.4,21.7) (59.1,17.6), (130.7,5.7), (27.1,4.3)
	9	(66.2,8.5), (25.0,9.3), (24.7,10.1), (53.9,20.2) (77.3,12.3), (137.1,6.4), (27.1,14.5)
	14	(71.5,13.3), (27.9,7.7), (23.8,9.9), (88.4,17.3) (89.5,16.9), (137.2,12.1), (41.8,13.7)
	15	(71.9,22.3), (26.3,22.7), (27.2,45.0), (72.7,39.5) (84.7,46.6), (133.8,12.0), (29.7,40.2)

Table 5.11: Performance of the rule based classifiers designed with context-sensitive reasoning scheme for 4 different partitions of Satimage1 data set.

Partition Number	Number of Rules	% of Error	
		Trng.	Test
1	27	11.0%	15.58%
2	26	9.8%	16.19%
3	25	12.8%	15.96%
4	20	9.6%	16.22%

Table 5.12: Classification performances designed with context-sensitive reasoning scheme for 4 different training sets for Satimage2.

Training Set	No. of rules	Error Rate in Training Data	Error Rate in Whole Image
1.	30	11.87%	13.5%
2.	25	14.6%	14.45%
3.	25	12.19%	13.18%
4.	27	12.75%	13.1%

change considerably between the two rules. Depending on the complexity of the class structure, the required number of rules also changes. The number of rules for a given class may also vary depending on the training set used. However, this variation across the training sets is not much. For example, the number of rules for class 1 obtained using 4 training sets are 3, 2, 3 and 4 respectively. Similar results are obtained for other classes too. This indicates good robustness of the proposed rule generation and tuning algorithm.

5.5.5 Performance of the classifiers with context-sensitive inferencing

To study the effect of context-sensitive inferencing scheme on the classifiers we performed several experiments with the multispectral satellite image data sets. In this section we report the results on a small data set Satimage1 and a large data set Satimage2. We take the set of rules obtained after context tuning (i.e., the rule tuning for context-free inferencing) stage. As mentioned earlier, when the context tuning of the rules is performed, we have used a fixed value of q (=-10.0) for all rules. Now, this rule set is

Table 5.13: Performance analysis of context-sensitive inferencing for rule based classifiers (when the initial rules were context tuned with q=-10.0 for all rules)

Training data	Initial q_i	Training) Error (E)		Training Misclassification		Misclassification in test data or whole image	
		Initial	Final	Initial	Final	Initial	Final
Satimage1	-10.0	176.26	175.9	54 (10.8%)	55 (11.0%)	925 (15.6%)	917 (15.4%)
	1.0	359.71	176.02	72 (14.4%)	55 (11.0%)	1000 (16.8%)	915 (15.4%)
	5.0	468.09	176.28	116 (23.2%)	54 (10.8%)	1536 (25.9%)	908 (15.3%)
Satimage2	-10.0	688.07	685.06	192 (12.0%)	189 (11.8%)	35657 (13.6%)	35202 (13.4%)
	1.0	1283.42	724.98	274 (17.1%)	202 (12.6%)	37241 (14.2%)	35585 (13.5%)
	5.0	1487.01	684.86	414 (25.8%)	190 (11.8%)	50330 (19.2%)	35082 (13.4%)

tuned for q_is (i.e., every rule is allowed to have its own value of q).

The performances of the context-sensitive classifiers after conjunction operator tuning are summarized in Tables 5.11 and 5.12 for Satimage1 and Satimage2 respectively. It can be observed that for the case of tuning the rule set, with q_is initialized at -10.0, the performances in terms of classification rate do not improve significantly. However, some improvement in terms of training error as defined by Eq. (5.12) is observed in all cases. It is further seen that after tuning, the q_is for all rules remain negative, which makes every rule to use approximately the minimum as the conjunction operator. This is hardly surprising, since the initial rules are already context tuned with a fixed value of $q = -10.0$ for every rule. So, it indicates that the context tuning with *fixed* q has developed the rule set to correspond to a minima of the energy function.

To investigate this issue further, we take the same set of rules (i.e., the context tuned rules with fixed $q = -10.0$). We initialize the q_is with different values, namely, 1.0 and 5.0 and tune the reasoning parameter q_is. All classifiers, with the same training data set have shown strong tendencies of convergence to similar set of conjunction operators, irrespective of the initial value of the q_is. We present the result of the classifiers designed with the partition 1 of Satimage1 and Satimage2 with initial values of q_is 1.0 and 5.0 in Table 5.13. We emphasize that the initial value of 1.0/5.0/-10.0 corresponds to the initial value of q_i of every rule for the tuning of the inference parameter. But the rule base parameters v_{ij}s and σ_{ij}s are already tuned using a fixed value of $q = -10.0$. Column 2 of Table 5.13 (and 5.14) shows the initial value of q_is for all rules in the rule base when the q-tuning for every rule (to realize *context-sensitive inferencing*) starts.

Table 5.13 shows that for both cases with initial q_is 1.0 and 5.0, though the initial value of error and misclassification rates are quite high, after the tuning they are much lower and very close to those corresponding to classifiers with initial q_i -10.0 for the respective data set. This fact is reflected for the training sets as well as the test sets. It was further observed that for all initial settings of q_i the final values of q_i become negative making

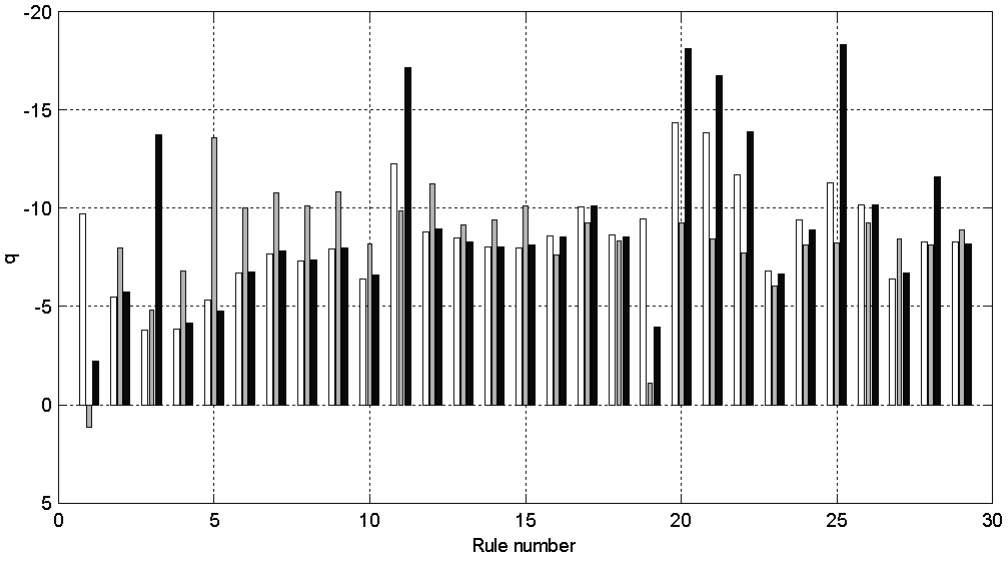

Figure 5.8: Bar diagram of q_is of the rules for Satimage2 for initial q_i -10, 1 and 10 (when the initial rules were context tuned with q=-10.0 for all rules).

Table 5.14: Performance analysis of context-sensitive inferencing for rule based classifiers (when the initial rules were not context tuned)

Training data	Initial q_i	Training) Error (E)		Training Misclassification		Misclassification in test data or whole image	
		Initial	Final	Initial	Final	Initial	Final
Satimage1	-10.0	258.42	235.39	70 (14.0%)	67 (13.4%)	1035(17.4%)	958(16.14%)
	1.0	345.41	245.15	74 (14.8%)	67 (13.4%)	1097 (18.5%)	941 (15.8%)
	5.0	471.22	246.02	116 (23.2%)	68 (13.6%)	1680 (28.3%)	1042 (17.5%)
Satimage2	-10.0	1198.47	1081.71	426 (26.6%)	398 (24.8%)	68070 (25.9%)	62873 (23.9%)
	1.0	1280.22	1096.01	417 (26.0%)	382 (23.8%)	63645 (24.2%)	56464 (21.5%)
	5.0	1446.36	1143.70	472 (29.5%)	404 (25.2%)	74299 (28.3%)	57174 (21.8%)

the operator more or less equivalent to the *minimum* operator. These are observed to attain values close in magnitude also. A small but interesting exception occurs in case of Satimage2 (training set 1). Figure 5.8 shows the bar diagram of the tuned q values of the rules for Satimage2. In the figure each group of three bars shows the tuned q values for a particular rule for three different initial values of $q = -10.0$ (white), 1.0 (gray) and 5.0 (black) respectively. For rule 1 with initial $q_1 = 1.0$, it can be observed that q_1 remains 1.0. We analyzed the case and found that the rule was not fired at all and the value remained the same as the initial value. The results in Tables 5.13 and 5.14 suggest that the context tuning using softmin has successfully reduced the error and developed a set of rules such that the optimal performance of the classifiers can be obtained with minimum-like operators.

To investigate the validity of the above we carried out another set of experiments. Here

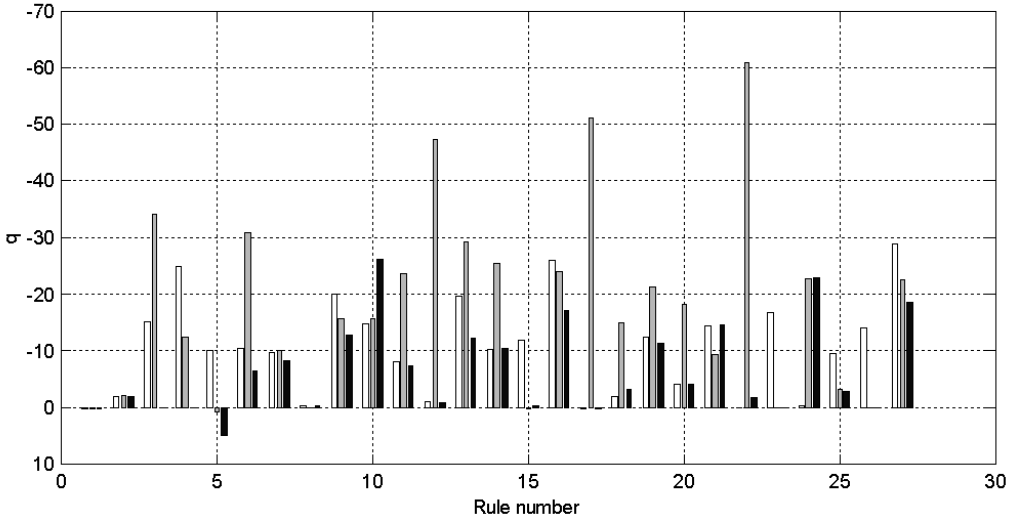

Figure 5.9: Bar diagram of q_is of the rules for Satimage1 for initial q_i -10, 1 and 5 (when the initial rules were not context tuned).

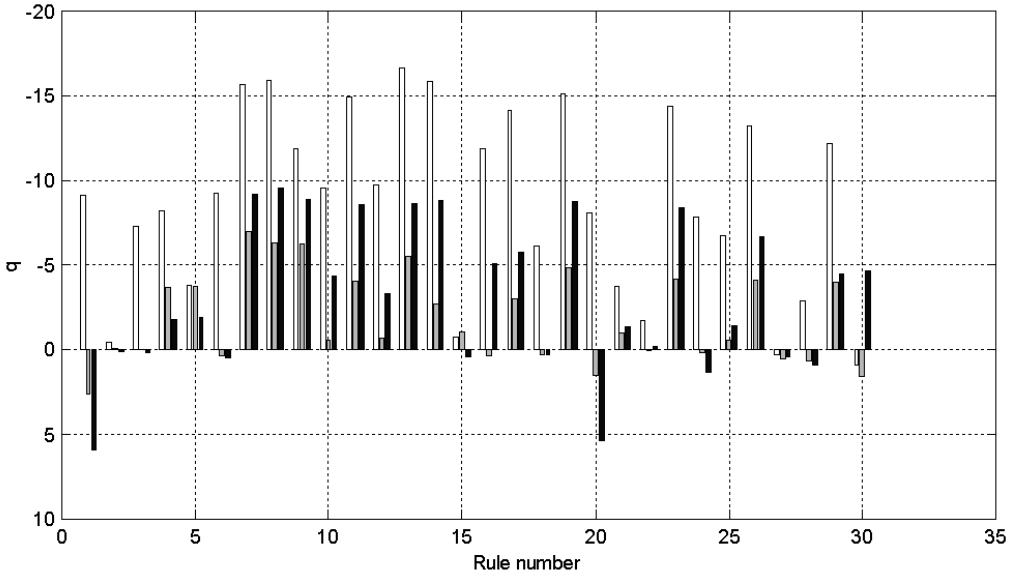

Figure 5.10: Bar diagram of q_is of the rules for Satimage2 for initial q_i -10, 1 and 5 (when the initial rules were not context tuned).

we used the initial rule set as obtained from the prototype generation stage, i.e., the rules are *no more context-tuned*. In other words, the rule base parameters are initialized from the set of prototypes, but not tuned. We tuned inferencing parameters q_is for these rules with initial q_is -10.0, 1.0 and 5.0 respectively. The results of tuning for the partition 1 of Satimage1 and Satimage2 are summarized in Table 8. As evident from the results, in all the cases both initial and final performances are worse than those in the previous experiment. However, the results also reveal that the conjunction operator (context-sensitive) tuning have produced a substantial improvement in performances for all cases. Thus it shows that if the initial rule base is not very good, then the context-sensitive inferencing can result in substantial improvement in performance.The bar plots of the tuned q_i values of the rules for Satimage1 and Satimage2 are shown in figures 5.9 and 5.10 respectively. It can easily be discerned from the figures that the final set of q_i values, unlike the previous experiment, differ substantially for different initial values of the q_is. This clearly indicates that the system in this case attains different error minima in q_i-space for different initialization.

5.6 Conclusion

Fuzzy rule based classifiers are capable of dealing with data having different variances in different features and also different variances for different classes. In such a system a rule represents a region in the input space, which we call the **context** of the rule. Here we proposed a scheme for designing a fuzzy rule based classifier. It starts with generation of a set of prototypes. Then these prototypes are converted into fuzzy rules. The rules are then tuned with respect to their context. We have developed two variants of the context tuning algorithm, one for rules using product as the conjunction operator and the other is used when the conjunction operator is *softmin*. The rule based classifiers are tested with several data sets and found to produce very good classification results.

The fuzzy rule based classifiers are then extensively used to classify multispectral satellite images and the performances obtained are very good. The classifier performance for such image data may further be improved using features other than gray levels. Also use of other T-norms can alter the performance of the classifier.

If softmin is used as the conjunction operator, then using different q (a parameter of softmin, we call it the *inferencing parameter*) for different rules, each rule can be made to use effectively a different conjunction operator. Such a scheme is called **context-sensitive inferencing**. We have developed an algorithm for tuning the q parameter for different rules.

Our experimental result suggests that if the context tuning (with fixed q value for

all rules) is carried out properly then subsequent context-sensitive inferencing offers marginal (if any) improvement. However, if the rules are not context tuned, the context-sensitive inferencing can improve the performance of the classifier substantially.

Some other facts have also come to light from these experiments. The results of q tuning of the context-tuned rules suggest possible existence of a deep error minima with respect to the q parameters to which the system reaches when it is context-tuned. But the results for the rule sets without context-tuning indicate that the system lands up in different minima for different initializations of q_is.

In light of above we can suggest two possible benefits for using context-sensitive reasoning for classification: (1) if the context tuning is not good enough, the performance may be enhanced with context-sensitive reasoning and (2) the results of tuning the q_is with different initializations may be used for judging the quality of a rule base as a whole.

Chapter 6

Evidence Theory based Decision Making for Fuzzy Rule based Classifiers [1]

[1] Part of this chapter has been published in [201] and the whole of it is published in [207].

6.1 Introduction

In the previous chapter we have developed a comprehensive scheme for designing fuzzy rule based classifiers. The method starts with generating a set of prototypes using the SOM-based prototype generation algorithm introduced in Chapter 4. Then each of the prototypes is converted into a fuzzy rule which is further fine tuned to obtain the final rule base. The decision making procedure in the system involved classifying an unknown data into the class of the rule producing the highest firing strength for the data. Thus the classification result is obtained in form of a crisp label with an accompanying confidence value, i.e., the firing strength α of the rule used for decision making. The magnitude of α (to be precise $(1-\alpha)$) is an indicator of the uncertainty involved in the classification of the data. The uncertainty may be due to different factors, such as, measurement error, random noise, overlapping of the class boundaries etc. However, we have not used this information to take any action (other than detecting the points with unacceptably low value of α as outlier).

A closer look at the functionality of the fuzzy rule base reveals that the rule base can be used to produce outputs with more information. The output of the rule base can be obtained in the form of a possibilistic label vector $\boldsymbol{\alpha} \in \Re^c$, where c is the number of classes and the k-th component of $\boldsymbol{\alpha}$, α_k $(0 \leq \alpha_k \leq 1)$ is the highest firing strength among the rules belonging to class C_k. Thus the possibilistic label vector allows us not only to find the option with the highest confidence, but also to look at other alternatives, which sometimes may be very close to the highest one. This difference in firing strength could be genuine (i.e., true class corresponds to the highest firing strength) or it may be due to the uncertainty of the classification process (i.e., the true class is one of the alternatives with slightly lower firing strength). In such a situation it might be prudent to use additional information, if available, for the final decision making. In an image, usually, the spatial neighbors are correlated. This becomes more relevant for satellite image analysis as one can exploit the information available with the spatial neighborhood. If we can judiciously use information from the neighbors, it is likely to produce more accurate decisions. In this chapter we use Dempster-Shafer theory of evidence to develop several decision making strategies for classifying landcover types in multispectral satellite images. Hereafter we shall develop the decision making schemes from the perspective of multispectral satellite image analysis but they can be applied with little modification to other problem domains (such as speech data) where contextual correlation can be expected. In these schemes the output of the rule base (possibilistic label vector) for a pixel of interest as well as its spatial neighbors are used together to decide the class membership of the pixel of interest.

The rest of the chapter is organized as follows. We discuss the problem of landcover

classification from multispectral satellite images in Section 6.2. Here we also point out the need for aggregating information from spatial context for the task. In section 6.3 we provide an overview of the Dempster-Shafer theory of evidence. The process of generating possibilistic label vectors using fuzzy rule base is discussed in Section 6.4. In section 6.5 we describe four methods for aggregation of spatial contextual information for decision making. We present the experimental results and discussions in section 6.6. Section 6.7 concludes the chapter.

6.2 The Landcover analysis problem

Although, in the previous chapters, we have briefly talked about landcover classification, this chapter focus on only landcover classification from multispectral satellite images. In particular, we develop some schemes for aggregation of spatial (contextual) information. Hence, we provide a reasonably detailed discussion of the landcover analysis problem.

Landcover is one of the most fundamental geographical variables and plays a crucial role in a host of enquiries like land erosion rate, depletion of forest cover, species dispersion routes, use of water bodies, resource planning and utilization etc. It was long recognized that the landcover data collected through traditional ground based methods are often out-of-date or of poor quality [325] and also difficult to acquire [95]. So land cover classification using remotely sensed images is considered to be a cost effective and reliable method for generating up-to-date landcover information [101]. As mentioned earlier, usually such images are captured by multispectral scanners (such as Landsat TM), that acquire data at several distinct spectral bands producing multispectral images. However, analysis of such data calls for sophisticated techniques.

In response to the challenges posed by this problem numerous techniques have been developed. Most widely used techniques employ statistical modelling involving discriminant analysis and maximum likelihood classification [272, 309]. Though such classifiers can theoretically offer optimal performance [88], the underlying assumptions for such a claim is that all class-conditional probability densities are known. This condition is never achieved in practice. To avoid this problem several methods employing artificial neural networks [14, 27, 36, 272] have been studied. These systems learn about the data from a set of training samples, without any prior assumptions.

However, neural network based classifiers always classify a sample to the class for which maximum support is obtained, no matter how small this amount of support may be. Such system will use the same principle to make decision even if there is another class for which the support is very close to the maximum. This particular feature received considerable criticism from several authors [101, 326] in context of their applicability to

landcover classification. They argued for suitability of "soft" classifiers which can produce a measure of confidence in support of the decision as well as indicate measures of confidence in support of alternative decisions, which can be used for further processing presumably using auxiliary information and this can result in a more robust and accurate system. Their arguments stemmed from the following observations regarding the landcover analysis problems:

- A user defined class may actually correspond to different sub-classes in the data. For example, an analyst may be interested in determining the portion of land under cultivation. But different crops may have different spectral signatures, even the same crop at different stages of maturity may have different spectral signatures.
- The spectral signature of the same object may vary significantly due to microclimatic variations such as existence of cloud, slope of the ground etc.
- Due to the limit of spatial resolution of the sensor, each pixel in an image corresponds to a significant area of the land. If the area corresponding to a pixel contains objects from different classes, each of them will contribute their spectral signature to the radiance captured by that pixel. Such pixels are termed as "mixed" pixels as against "pure" pixels which correspond to the landcover area containing objects belonging to a single class.

In developing soft classifiers for landcover analysis two approaches have gained popularity. These are based on (1) fuzzy set theory [87] and (2) Dempster and Shafer's evidence theory [304]. There is, of course, the probabilistic approach, this we do not pursue further. In the previous chapter we have developed a comprehensive method for designing fuzzy rule based classifiers and used them for classifying landcover types from multispectral satellite images. A short review of different fuzzy set theoretic approaches used for the same purpose is provided in Section 5.5.3.

The other approach to design soft classifiers uses the evidence theory developed by Dempster and Shafer [304]. Designing general purpose classifiers using Dempster-Shafer theory is an active area of research [76, 265, 369, 143]. As observed by Lee et al. in [211], for multispectral image analysis, there may be a great incentive for applying Dempster-Shafer theory of evidence. Since the theory of evidence allows one to combine evidences obtained from diverse sources of information in support of a hypothesis, it seems a natural candidate for analyzing multispectral images for landcover classification. This approach is found to produce good results when applied to multispectral image analysis [313, 275, 162]. In all these works, the approach is to treat each channel image as a separate source of information. Each image is analyzed to associate each pixel with

some degree of belief pertaining to its belonging to each member of a set of hypotheses known as the *frame of discernment*. Usually some probabilistic techniques are employed to assign the degree of belief. In the next stage these beliefs from all images for a pixel are combined using Dempster's rule of combination [304] to calculate the total support for each hypothesis.

In a satellite image, usually the landcover classes form spatial clusters, i.e., a pixel belonging to a particular class is more likely to have neighboring pixels from the same class rather than from other classes. Thus, inclusion of contextual information from the neighboring pixels is likely to increase classification accuracy. An overview of common contextual pattern recognition methods can be found in [122]. Recently contextual methods based on Markov Random Field (MRF) models have become popular for classification of multispectral [299] as well as multisource [310, 328] satellite images. Typically a Bayesian framework is used to model the posterior probability. To determine the values of the model parameters an energy function is minimized using optimization techniques like simulated annealing, genetic algorithm etc. These methods typically utilize the contextual information in the training stage. Also their accuracy depends on the correctness of the assumption about the class-conditional density functions.

In this chapter we propose a scheme for classifier design that uses both fuzzy sets theory and Dempster-Shafer theory of evidence. This is a two stage scheme. In the first stage a fuzzy rule based classifier is developed. This classifier is noncontextual and for each pixel it generates a possibilistic label vector. In the next stage we aggregate the responses of the fuzzy rules over a 3×3 *spatial* neighborhood of a pixel to make the classification decision about that pixel. Thus, the decision making process takes into consideration the information available from the spatial neighborhood of the pixel. In other words, *spatial contextual* information is exploited to make a better decision. Note that, unlike MRF model-based methods, the spatial neighborhood information of the pixels are used in the decision making stage only, not in the training stage. Further, in this scheme there is no need to make assumptions about class-conditional densities. We propose four methods for decision making. The first one is a simple aggregation of the fuzzy labels of the neighboring pixels while the rest are based on the theory of evidence. The evidence theoretic methods vary in three respects, (1) how the *sources of evidence* are identified, (2) which *focal elements* are selected and (3) how the *basic probability assignments* (BPAs) are computed.

6.3 Handling uncertainty with theory of evidence

We face different types of uncertainty while designing information processing systems. Fuzzy sets provide us with tools for modelling one form of uncertainty. For a fuzzy subset A defined over the universal set Θ, each element $x \in \Theta$ belongs to A by the amount $\mu_A(x)$, in other words, $\mu_A(x)$ expresses the degree of belongingness of x to A. However, we are often faced with a situation when we need to assign a crisp label to $\mathbf{x}$. But due to inadequacy of information we cannot be very specific and instead of assigning only one label, we may be able to assign some confidence over a subset of possibilities, when the true label (say θ) is one of the elements in the subset. To put it more concretely we cite here an example from [164]:

"Consider the jury members for a criminal trial who are uncertain about the guilt or innocence of the defendant. ... the set of people who are guilty of the crime and the set of innocent people are assumed to have very distinct boundaries. The concern, therefore, is not with the degree to which the defendant is guilty but with the degree to which the evidence proves his or her membership in either the crisp set of guilty people or in the crisp set of innocent people. We assume that perfect evidence would point to full membership in one and only one of these sets. Our evidence, however, is rarely, if ever, perfect, and some uncertainty usually prevails."

As described in the above example, the universal set is consisted of two clearly defined crisp sets, an element cannot be a partial member of both sets (i.e., he/she must be either guilty or innocent). This type of uncertainty can be represented by assigning a value to each possible crisp subset to which the element can belong. This value represents the degree of evidence supporting the element's inclusion in the set. A *fuzzy measure* can be used to represent this type of uncertainty. A fuzzy measure is defined as follows:

Let Θ be the universal set and $P(\Theta)$ be its power set. Any function

$$g : P(\Theta) \rightarrow [0, 1]$$

is a fuzzy measure if it satisfies the following three axioms [164]:

- $g1 : g(\emptyset) = 0$ and $g(\Theta) = 1$.
- $g2$: For every $A, B \in P(\Theta)$, if $A \subset B$ then $g(A) \leq g(B)$.
- $g3$: For every sequence $(A_i \in P(\Theta) \mid i = 1, 2, \cdots)$ of subsets of Θ, if either $A_1 \subseteq A_2 \subseteq \cdots$ or $A_1 \supseteq A_2 \supseteq \cdots$, then

$$\lim_{i \rightarrow \infty} g(A_i) = g(\lim_{i \rightarrow \infty} A_i).$$

Axiom $g1$ states that the element cannot be in the empty set and must be within universal set. In other words, since the empty set does not contain any element, it cannot contain our element of interest. The universal set contains all the elements, hence it must contain the element of interest. This is often known as the "closed world assumption". According to axiom $g2$ the evidence supporting the membership of an element in a set is at least as great as the evidence that the element belongs to any subset of that set.

$g3$ is applicable only for infinite universe and in the present context since Θ is finite, $g3$ can be disregarded. *Belief* and *Plausibility* functions introduced by Dempster and Shafer [304] are two important and well developed special types of fuzzy measures. They provide a comprehensive and easy framework of representing the uncertainty arising out of lack of perfect evidence. They also allow combination of evidence available from different sources to reach at an aggregated evidence in a consistent manner.

6.3.1 Dempster-Shafer theory of evidence

A Belief measure is a function $Bel : P(\Theta) \rightarrow [0, 1]$ that satisfies the axioms $g1$ through $g3$ of fuzzy measures and the following additional axiom:

- $Bel(A_1 \cup A_2 \cup \cdots \cup A_n) \geq \sum_i Bel(A_i) - \sum_{i<j} Bel(A_i \cap A_j) + \cdots + (-1)^n Bel(A_1 \cap \cdots \cap A_n)$, for every n and for every collection of subsets of Θ.

There is a plausibility measure with each belief measure defined by

$$Pl(A) = 1 - Bel(A^c) \forall A \in P(\Theta).$$

In evidence theory framework the universal set Θ is known as the *frame of discernment* [304]. These measures can be interpreted as follows:

- Given a hypothesis, $Bel(A)$ is a quantitative measure of the belief that the "true" hypothesis is contained in A.
- $Pl(A)$ is the measure of belief that the "true" hypothesis is *not* contained in the complementary set of A.

Every belief measure and its dual plausibility measure can be expressed in terms of a Basic Probability Assignment (BPA) function m defined as:

$m : P(\Theta) \rightarrow [0, 1]$ is called a BPA whenever $m(\emptyset) = 0$ and $\sum_{A \subseteq \Theta} m(A) = 1$.

Here $m(A)$ is interpreted as the degree of evidence supporting the claim that the "truth" is in A and in absence of further evidence no more specific statement can be made. A belief measure and a plausibility measure are uniquely determined by m through the formulae:

$$Bel(A) = \sum_{B \subseteq A} m(B). \tag{6.1}$$

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \ \ \forall A \subset \Theta. \tag{6.2}$$

From Eq. (6.1) and Eq. (6.2) we see that,

$$Pl(A) \geq Bel(A) \ \ \forall A \in P(\Theta).$$

Every set $A \in P(\Theta)$ for which $m(A) > 0$ is called a *focal element* of m. The BPA $m(\Theta)$ assigned to whole frame of discernment is interpreted as the amount of *ignorance*.

Evidence obtained in the same context from two distinct sources and expressed by two BPAs m^1 and m^2 on some power set $P(\Theta)$ can be combined by Dempster's rule of combination to obtain a joint BPA $m^{1,2}$ as:

$$m^{1,2}(A) = \begin{cases} \frac{\sum_{B \cap C = A} m^1(B) m^2(C)}{1-K} & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset. \end{cases} \tag{6.3}$$

Here

$$K = \sum_{B \cap C = \emptyset} m^1(B) m^2(C).$$

Equation (6.3) is often expressed with the notation

$$m^{1,2} = m^1 \oplus m^2.$$

The rule is commutative and associative. Evidence from any number (say k) of distinct sources can be combined by repetitive application of the rule as:

$$m = m^1 \oplus m^2 \oplus \cdots \oplus m^k = \oplus_{i=1}^{k} m^i.$$

If m^1 is vacuous (i.e., $m^1(\Theta) = 1$ and $m^1(A) = 0 \forall A \subset \Theta$), then m^1 and m^2 are combinable and $m^1 \oplus m^2 = m^2$. If m^1 is Bayesian (i.e., $m^1(A) = 0 \forall A$ such that $\mid A \mid > 1$) and m^1 and m^2 are combinable, then $m^1 \oplus m^2$ is also Bayesian.

6.3.2 Pignistic probability

Given a body of evidence we are often required to make decisions based on the available evidence. In such case Θ becomes the set of decision alternatives and the function Bel

denotes our belief about the choice of an optimal decision $\theta_0 \in \Theta$. However, in general it is not possible to select the optimal decision directly from the evidence embodied in the function Bel. We need to construct a probability function P^{Θ} on Θ in order to select the optimal decision. The transformation between the belief function Bel and the probability function P^{Θ} is known as *pignistic transformation* [307] and it is denoted here by Γ_{Θ}. Thus

$$P^{\Theta} = \Gamma_{\Theta}(Bel).$$

P^{Θ} is called a *pignistic probability*, which can be used for making decision. The pignistic probability for $\theta \in \Theta$ can be expressed in terms of BPAs as follows:

$$P^{\Theta}(\theta) = \sum_{A \subseteq \Theta, \theta \in A} \frac{m(A)}{\mid A \mid} \tag{6.4}$$

Optimal decision can now be chosen in favor of θ_0 with the highest pignistic probability.

6.4 Generating fuzzy label vectors from the fuzzy rule base

We use the fuzzy rule base extracted from the data using the method described in the previous chapter. However, the output of the fuzzy rule base for a data point $\mathbf{x}$ is computed in a different manner. Let there be c classes, $\mathcal{C}=\{C_1, C_2, \cdots C_c\}$. For each data point $\mathbf{x}$ the fuzzy rule base generates a fuzzy label vector

$$\boldsymbol{\alpha}(\mathbf{x}) = [\alpha_1(\mathbf{x}), \alpha_2(\mathbf{x}), \ldots, \alpha_c(\mathbf{x})]^T \in \Re^c,$$

such that the value of the k-th component of $\boldsymbol{\alpha}(\mathbf{x})$, $\alpha_k(\mathbf{x})$ $(0 \leq \alpha_k \leq 1)$ represents the confidence of the rule base supporting the fact that the data point $\mathbf{x}$ belongs to class C_k. Strictly speaking, $\boldsymbol{\alpha} \in \Re^c$ is a possibilistic label vector [32]. The value of the k-th component $\alpha_k(\mathbf{x})$ of the label vector $\alpha(\mathbf{x})$ is computed in the following manner:

Let the size of the rule base $\{R_i\}$ be $\hat{c}$ (i.e., $\mid \{R_i\} \mid = \hat{c}$), where $\hat{c} \geq c$. Since $c \leq \hat{c}$, there could be multiple rules corresponding to a class. Let $\{R_i^k\} \subset \{R_i\}$ be the subset of rules corresponding to class C_k. We designate firing strength of a rule $R_j \in \{R_i^k\}$ for $\mathbf{x}$ as $\alpha_j^k(\mathbf{x})$. Then, the k-th component of the possibilistic label vector $\boldsymbol{\alpha}$ is

$$\alpha_k(\mathbf{x}) = \underbrace{\arg\max}_{j,\ R_j \in \{R_i^k\}} \{\alpha_j^k(\mathbf{x})\}.$$

In other words, $\alpha_k(\mathbf{x})$ is the highest firing strength produced by the rules corresponding to the class C_k for $\mathbf{x}$. We treat this value as the confidence measure of the rule base

pertaining to the membership of $\mathbf{x}$ to the class C_k. However, if α_k is less than a threshold, say 0.01, it is set to 0.

In the present context each data vector $\mathbf{x}$ is associated with a pixel p of the multispectral image and has a spatial coordinate denoting its position in the image. This allows us to identify a data vector in its spatial context.

6.5 Aggregation of spatial contextual information for decision making

We develop four decision making schemes. The first one involves a simple aggregation of the possibilistic label vectors of the neighboring pixels. The scheme is modelled after the well known fuzzy k-nearest neighbor algorithm proposed by Keller et al. [158]. The other three methods use Dempster-Shafer theory of evidence.

In our decision making schemes, we consider a pixel together with the pixels within its 3×3 spatial neighborhood. We identify the pixel under consideration as p^0 and its eight neighbors as $p^1, p^2, \cdots, p^8$. The corresponding possibilistic label vectors assigned to these nine pixels are denoted as $\boldsymbol{\alpha}^0, \boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^8$.

6.5.1 Method 1: Aggregation of possibilistic labels by fuzzy k-NN rule

This is a simple aggregation scheme modelled after the fuzzy k-NN method of Keller et al. [158]. Given a set of nine label vectors $\{\boldsymbol{\alpha}^0, \boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^8\}$ an aggregated possibilistic label vector α^A is computed as follows:

$$\boldsymbol{\alpha}^A = \frac{\sum_{i=0}^{8} \boldsymbol{\alpha}^i}{9}. \tag{6.5}$$

The pixel p^0 is assigned to class C_k such that

$$\alpha_k^A \geq \alpha_i^A \quad \forall\ i = 1, 2, \cdots, c.$$

Note that, though the label vector $\boldsymbol{\alpha}^0$ corresponds to the pixel to be classified, no special emphasis (importance) is given to it in this aggregation scheme.

6.5.2 Method 2: Aggregation with Bayesian belief modelling

This aggregation method as well as the next two use the evidence theoretic framework of Dempster-Shafer. Within this framework the set of classes, $\mathcal{C}$, is identified as the *frame of discernment* Θ. A body of evidence (BOE) is represented by a BPA m over the subsets of $\mathcal{C}$. There are 2^c possible subsets of $\mathcal{C}$. The value $m(A)$ denotes the belief in support of the proposition that *the true class label of the pixel of interest is in* $A \subset \mathcal{C}$. In the context of our problem we shall

1. identify distinct bodies of evidence (BOE),
2. formulate a realistic scheme of assigning the BPAs to the relevant subsets of $\mathcal{C}$,
3. combine evidences provided by all BOEs,
4. compute the pignistic probability for each class from the combined evidence and
5. use maximum pignistic probability rule to make the decision.

In this scheme we identify eight BOEs for eight neighbor pixels with corresponding BPAs denoted as $m^1, m^2, \cdots, m^8$. We consider the Bayesian belief structure, i.e., each focal element has only one element. Assigning BPA to a subset essentially involves committing some portion of belief in favor of the proposition represented by the subset. So the scheme followed for assigning BPAs must reflect some realistic assessment of the information available in favor of the proposition. We define m^i as follows:

$$m^i(\{C_k\}) = \frac{(\alpha_k^i + \alpha_k^0)}{S}, k = 1, 2, ..., c \tag{6.6}$$

where $S = \sum_{k=1}^{k=c}(\alpha_k^i + \alpha_k^0)$ is a normalizing factor. Thus, each BPA contains c focal elements, one corresponding to each class and the assigned value $m^i(\{C_k\})$ is influenced by the magnitudes of firing strengths produced by the rule base in support of class C_k for the pixel of interest p^0 and its i-th neighbor p^i. Clearly the label vector $\boldsymbol{\alpha}^0$ influences all eight BPAs. Hence it is expected that in the final decision making, the influence of $\boldsymbol{\alpha}^0$ will be higher than any neighbor. Such an assignment is motivated by the fact, the spatial neighbors usually are highly correlated, i.e., pixel p^0 and its immediate neighbors are expected from the same class.

Thus for eight neighboring pixels we obtain eight separate BOEs. Since each of the sources has the same set of focal elements, the evidences are combinable and the global BPA for the focal elements can be computed by applying the Dempster's rule repeatedly. It can be easily seen that the global BPA m^G is also Bayesian and can be computed as

$$m^G(\{C_k\}) = \frac{\prod_{i=1}^{8} m^i(\{C_k\})}{\sum_{l=1}^{c}\prod_{i=1}^{8} m^i(\{C_l\})}. \tag{6.7}$$

In this case the pignistic probability $P^{\mathcal{C}}(C_k)$ is the same as $m^G(\{C_k\})$. So the pixel p^0 is assigned to class C_k such that

$$m^G(\{C_k\}) \geq m^G(\{C_l\}) \ \ \forall\ C_l \in \mathcal{C}.$$

6.5.3 Method 3: Aggregation with non-Bayesian belief

Here also the BOEs are identified in the same way as in the previous method. However, we allow the assignment of belief to subsets of $\mathcal{C}$ having two elements i.e., the subsets $\{C_l, C_m : l < m \text{ and } l, m = 1, 2, \cdots, c\}$. We define m^i as follows:

$$m^i(\{C_k\}) = \frac{(\alpha_k^i + \alpha_k^0)}{S}, k = 1, 2, ..., c \tag{6.8}$$

$$m^i(\{C_l, C_m : l < m\}) = \frac{(\alpha_l^i + \alpha_m^0) + (\alpha_m^i + \alpha_l^0)}{2S}, l, m = 1, 2, ..., c \tag{6.9}$$

where

$$S = \sum_{k=1}^{k=c}(\alpha_k^i + \alpha_k^0) + \sum_{l=1}^{l=c-1}\sum_{m=l+1}^{m=c}[(\alpha_l^i + \alpha_m^0) + (\alpha_m^i + \alpha_l^0)].$$

In Method 2 we exploited only the correlation of spatial neighbors. For satellite images when a pixel falls at the boundary of some landcover type, it may correspond to more than one landcover type. Since the chance of a pixel to have representation from more than two landcover types is not high, we restrict the cardinality of the focal elements to two. Using Eq. (6.8) and Eq. (6.9) for eight neighboring pixels we obtain eight separate bodies of evidence. Since each of them has the same set of focal elements, the evidences are combinable and the global BPA for the focal elements can be computed applying Dempster's rule repeatedly. In [76, 265] some comprehensive formulae are derived for computing the global BPA in one step using evidences from a number of sources. It was possible because they worked with *simple support functions*, where each source of evidence assigns a BPA to one focal element containing only one class and the rest of the belief is assigned to the whole frame of discernment. In our previous method we could use Eq. (6.7) to compute the global BPA since we dealt with singleton focal elements only. Unfortunately, in the present case the belief functions are not simple support functions. We have to compute the global BPA in steps, combining two bodies of evidence at a time and preparing an intermediate BOE which will be combined with another BOE and so on. Thus in our scheme the combined global BPA m^G is computed as follows:

$$m^G = \oplus_{i=1}^{8} m^i = (\cdots((m^1 \oplus m^2) \oplus m^3) \oplus \cdots m^8). \tag{6.10}$$

It is easily seen that:

$$
\begin{aligned}
m^{(i,j)}(\{C_k\}) &= m^i(\{C_k\}) \oplus m^j(\{C_k\}) \\
&= \frac{\left\{\begin{array}{l} m^i(\{C_k\})m^j(\{C_k\}) \\ +m^i(\{C_k\})\sum_{l\neq k} m^j(\{C_k, C_l\}) \\ +m^j(\{C_k\})\sum_{l\neq k} m^i(\{C_k, C_l\}) \\ +\sum_{l\neq k} m^i(\{C_k, C_l\})\sum_{m\neq k,l} m^j(\{C_k, C_m\}) \end{array}\right\}}{1-K}, \\
& \qquad\qquad k = 1, 2, ..., c \quad (6.11)
\end{aligned}
$$

and

$$
\begin{aligned}
m^{(i,j)}(\{C_l, C_m\}) &= m^i(\{C_l, C_m\}) \oplus m^j(\{C_l, C_m\}) \\
&= \frac{m^i(\{C_l, C_m\})m^j(\{C_l, C_m\})}{1-K}, l, m = 1, 2, ..., c, l \neq m \quad (6.12)
\end{aligned}
$$

where K is given by

$$
\begin{aligned}
K &= \sum_{k=1}^{c-1} m^i(\{C_k\}) \sum_{l=k+1}^{c} m^i(\{C_l\}) \\
&\quad + \sum_{k=1}^{c} m^i(\{C_k\}) \sum_{l,m\neq k}^{c} m^j(\{C_l, C_m\}) \\
&\quad + \sum_{k=1}^{c} m^j(\{C_k\}) \sum_{l,m\neq k}^{c} m^i(\{C_l, C_m\}) \\
&\quad + \sum_{l\neq r,s, \text{ and } m\neq r,s}^{c} m^i(\{C_l, C_m\})m^j(\{C_r, C_s\}).
\end{aligned}
$$

Once m^G is obtained the pignistic probability for each class is computed. The following formula is used for computing the pignistic probability of class C_k:

$$
P^{\mathcal{C}}(C_k) = m^G(\{C_k\}) + \frac{\sum_{l=1,\, l\neq k}^{c} m^G(\{C_k, C_l\})}{2}. \quad (6.13)
$$

The pixel p^0 is assigned to the class C_k such that

$$
P^{\mathcal{C}}(C_k) \geq P^{\mathcal{C}}(C_l) \;\; \forall \; C_l \in \mathcal{C}.
$$

6.5.4 Method 4: Aggregation using evidence theoretic k-NN rule

This method is fashioned after the evidence theoretic k-NN method [76] introduced by Denœux. Here nine BPAs $m^0, m^1, \cdots, m^8$ are identified corresponding to the possibilistic label vectors $\boldsymbol{\alpha}^0, \boldsymbol{\alpha}^1, \cdots \boldsymbol{\alpha}^8$. The BPA m^i is assigned as follows:

$$
\text{Let } q = \underbrace{\arg\max}_{k}(\alpha_k^i), \quad (6.14)
$$

then

$$m^i(\{C_q\}) = \alpha_q^i \tag{6.15}$$

$$m^i(\mathcal{C}) = 1 - \alpha_q^i \tag{6.16}$$

$$m^i(A) = 0 \; \forall A \in P(\mathcal{C}) \backslash \{\mathcal{C}, \{C_q\}\} \tag{6.17}$$

Thus there is only one focal element containing one element in each BOE. The rest of the belief is assigned to the frame of discernment $\mathcal{C}$, which can be interpreted as the amount of ignorance. The evidences are then combined as follows:
Let Φ^q be the set of BPAs for which the focal element is $\{C_q\}$. If $\Phi^q \neq \emptyset$ then the corresponding BPAs can be combined as $m^q = \bigoplus_{m^i \in \Phi^q} m^i$ as follows,

$$m^q(\{C_q\}) = 1 - \prod_{m^i \in \Phi^q} (1 - m^i(\mathcal{C})), \tag{6.18}$$

$$m^q(\mathcal{C}) = \prod_{m^i \in \Phi^q} m^i(\mathcal{C}). \tag{6.19}$$

If $\Phi^q = \emptyset$, then m^q is simply the BPA associated with the vacuous belief function: $m^q(\mathcal{C}) = 1$. The global BPA m^G can be obtained by combining these class specific BPAs as $m^G = \bigoplus_{q=1}^{c} m^q$ such that

$$m^G(\{C_q\}) = \frac{m^q(\{C_q\}) \prod_{r \neq q} m^r(\mathcal{C})}{K} \;\; q = 1, 2, \cdots, c \tag{6.20}$$

$$m^G(\mathcal{C}) = \frac{\prod_{q=1}^{c} m^q(\mathcal{C})}{K} \tag{6.21}$$

where $K = \sum_{q=1}^{c} m^q(\{C_q\}) \prod_{r \neq q} m^q(\mathcal{C}) + \prod_{q=1}^{c} m^q(\mathcal{C})$.

The pignistic probability $P^G(C_k)$ can be calculated as

$$P^G(C_k) = m^G(\{C_q\}) + \frac{m^G(\mathcal{C})}{c}.$$

However, since the term $\frac{m^G(\mathcal{C})}{c}$ is same for all the classes, the pixel p^0 is classified to class C_k if

$$m^G(\{C_q\}) \geq m^G(\{C_l\}) \quad \forall \; C_l \in \mathcal{C}.$$

Like Method 1, in this method also no special emphasis is given to the pixel under consideration, p^0, over the neighboring pixels. Whereas in Method 2 and Method 3, $\boldsymbol{\alpha}^0$ influences all the BPAs, i.e., $\boldsymbol{\alpha}^0$ plays a special role. It can be seen later in experimental results that these two approaches provide different classification efficiencies depending on the nature of the spatial distribution of the classes in the image. Intuitively, if pixels corresponding to different landcover types are scattered all over the image, neighboring

pixels should not be given the same importance as that of the central pixel for optimal performance of the classifier. On the other hand, if pixels of a particular landcover type form compact spatial clusters, then giving equal importance to the neighboring pixels may be desirable. However, the optimal weight to be given to the neighboring pixels to get the best performance should depend on the spatial distribution of different landcover types on the image being analyzed. To realize this, we modify the BPA assignment scheme of current method as follows:

$$\text{Let } q = \underbrace{\arg\max}_{k}(\alpha_k^i),$$

then

$$\begin{aligned}
m^0(\{C_q\}) &= \alpha_q^0, \text{ for } i = 0 && (6.22)\\
m^i(\{C_q\}) &= w\alpha_q^i, \text{ otherwise} && (6.23)\\
m^0(\mathcal{C}) &= 1 - \alpha_q^0, \text{ for } i = 0 && (6.24)\\
m^i(\mathcal{C}) &= 1 - w\alpha_q^i, \text{ otherwise} && (6.25)\\
m^i(A) &= 0 \; \forall A \in P(\mathcal{C}) \backslash \{\mathcal{C}, \{C_q\}\} && (6.26)
\end{aligned}$$

where $0 \leq w \leq 1$ is a weight factor that controls the contribution of the neighboring pixels in the decision making process. The optimal value of w for the best classification performance depends on the image under investigation and can be learnt during training using a grid search.

6.6 Experimental results

We report the performances of the proposed classifiers for the two multispectral satellite images, Satimage2 and Satimage3. Though in the previous chapter we have used three multispectral satellite images, the pixel position information, which is crucial for the methods proposed here, is not available for the first of them (Satimage1). Hence we use only Satimage2 and Satimage3 in the current study. The description of these data sets is already provided in Section 4.6.1 as well as in Section 5.5.3 with more details. The channel 1 image of Satimage2 and the ground truth is presented in the Figures 5.2 and 5.4. The channel 4 image and ground truth for Satimage3 are depicted in Figures 5.3 and 5.5 respectively.

The performances of the fuzzy rule based classifiers for Satimage2 and Satimage3 using firing strengths directly for decision making are reported in Tables 5.8 and 5.9 respectively in the previous chapter. Comparison of these results with the published results is provided in Section 5.5.4. Here we report the performances of the four methods that use contextual information for decision making.

Table 6.1: Performances of fuzzy rule based classifiers using spatial neighborhood information aggregation methods for decision making for different training sets for Satimage2

Training Set	No. of rules	Error Rate in Whole Image			
		Method 1	Method 2	Method 3	Method 4
1.	30	13.32%	11.93%	11.48%	13.43%
2.	25	14.00%	13.01%	12.62%	14.38%
3.	25	13.66%	11.54%	11.23%	13.90%
4.	27	12.98%	11.01%	10.45%	13.18%

Table 6.2: Performances of fuzzy rule based classifiers using spatial neighborhood information aggregation methods for decision making for different training sets for Satimage3

Training Set	No. of rules	Error Rate in Whole Image			
		Method 1	Method 2	Method 3	Method 4
1.	14	11.55%	12.96%	12.64%	11.75%
2.	14	11.55%	12.75%	12.48%	11.65%
3.	12	11.24%	12.45%	12.23%	11.35%
4.	11	11.24%	12.53%	12.28%	11.44%

Tables 6.1 and 6.2 summarize the performances of the four classifiers using different methods of aggregation of spatial information for Satimage2 and Satimage3 respectively. We used the same set of fuzzy rules as used in the previous chapter to generate the inputs to the four proposed methods.

Comparison between Table 6.1 and Table 5.8 shows that for Satimage2 Method 3 performs the best with improvements varied between 1.12% and 2.05% and the best performing classifier (for training set 4) achieves an error rate as low as 10.45%. This is closely followed by Method 2. Method 1 and Method 4 show marginal improvement for training sets 1 and 2 while for training sets 3 and 4 their performance degrades a little. Comparison of Table 6.2 and Table 5.9 show that although the classifiers using Method 3 increase the classification accuracy by about 1.5%, Method 1 is clearly the best performer for Satimage3 with improvements ranging between 2.49% and 2.99%. Similar performances are achieved by Method 4. Method 4 used $w = 1$ for both Table 6.1 and Table 6.2.

Figure 6.1 shows the classified image corresponding to training set 1 using Method 3 for Satimage2. Figure 6.2 displays the classified image (for training set 1 using Method 4) for Satimage3.

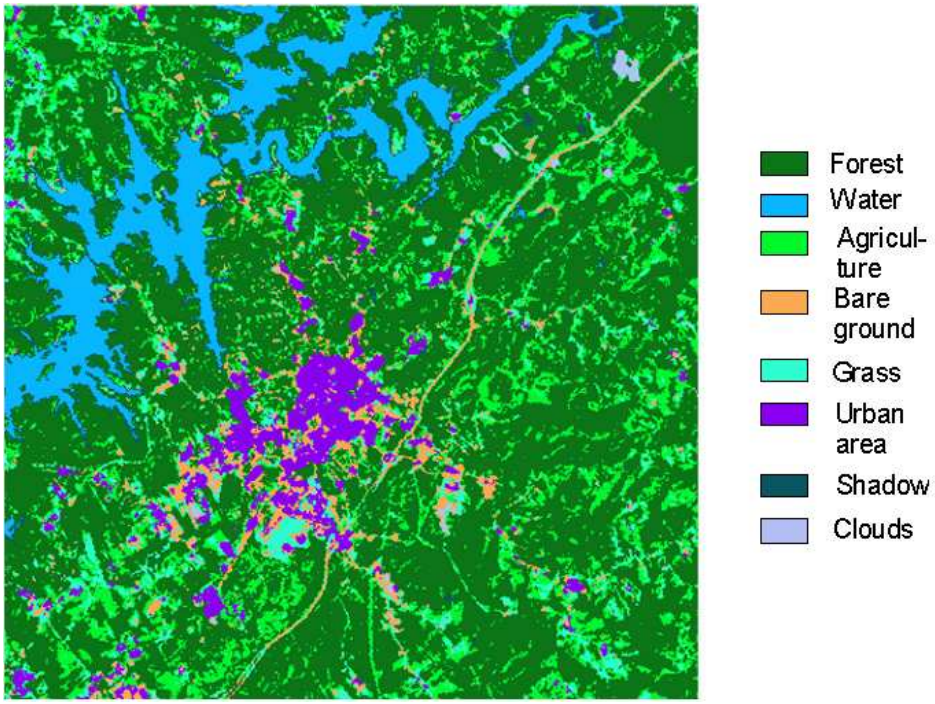

Figure 6.1: The classified image for Satimage2 (Training set 1, Method 3). The classes are represented by different colors.

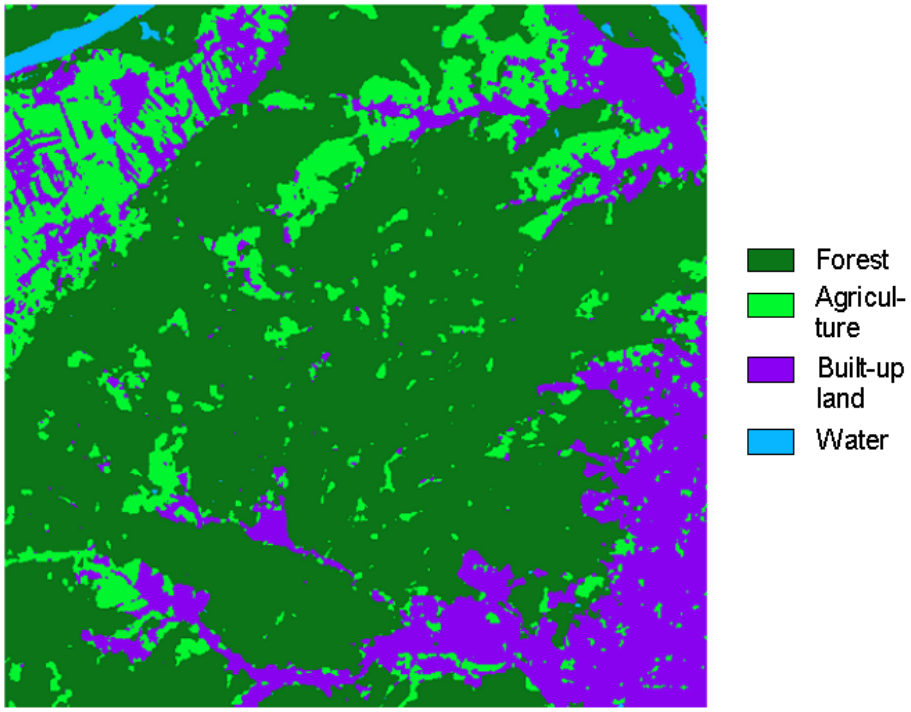

Figure 6.2: The classified image for Satimage3 (Training set 1, Method 4). The classes are represented by different colors.

Table 6.3: Class-wise average classification performance for Satimage2

Landcover types	Average classification rate (%)			
	Method 1	Method 2	Method 3	Method 4
Forest	93.25	90.80	91.33	94.47
Water	90.95	90.42	90.64	92.29
Agriculture	68.49	83.60	83.12	60.12
Bare ground	58.15	65.46	65.92	55.10
Grass	53.19	73.75	75.65	55.59
Urban area	80.82	80.83	79.76	75.32
Shadow	48.97	96.09	95.68	40.42
Clouds	99.78	99.78	99.78	99.79

Our experimental results demonstrate that for Satimage2 Method 2 and Method 3 work well while the other methods work better for Satimage3. These differences in performances can be explained if we look into the way the neighborhood information is aggregated in each method and the nature of the spatial distribution of the classes in the images. In Method 1 the fuzzy label vectors of the central pixel (the pixel of interest) and its eight neighboring pixels are treated in same way for aggregation. The same is true for Method 4 though evidence theoretic approach is used for information aggregation. In Method 2 and Method 3 eight BPAs are defined, each of them is assigned using the possibilistic label vector of the central pixel and that of one of the eight neighboring pixels. Thus all the BPAs are heavily influenced by the central pixel, and consequently, so is the final decision. Hence, it is expected that for the images dominated by large stretches of homogeneous areas (i.e., area covered by single landcover type) can be classified better by Method 1 and Method 4. Also for Method 4 there may exist an optimal weight for neighborhood information (different from $w = 1$) with improved performance. We shall see later that this is indeed the case. Comparison of Figure 5.5 (ground truth for Satimage3) with Figure 5.4 (ground truth for Satimage2) reveals that Satimage2 mostly consists of small (often very small) patches of landcover types, while Satimage3 has large patches of landcover types. So for Satimage2 neighborhood information needs to be used judiciously to get an improved classifier. This is what achieved by Method 2 and Method 3.

To have a closer look at class-wise performances of the proposed methods we have presented in Table 6.3 and Table 6.4 the class-wise classification performances for Satimage2 and Satimage3 respectively. It can be seen from the table that for Satimage3 all four methods perform comparably for each of the four classes, with a slender edge in

Table 6.4: Class-wise average classification performance for Satimage3

Landcover types	Average classification rate (%)			
	Method 1	Method 2	Method 3	Method 4
Built-up land	85.83	84.09	84.38	85.55
Forest	95.46	94.00	94.22	95.68
Water	96.02	93.77	94.32	96.13
Agriculture	68.20	68.03	67.79	66.92

favor of Method 1 and Method 4. However, for Satimage2, which contains more complicated spatial distribution of the classes, there is a significant class-specific variation of performance among different methods. For the classes Forest, Water, Urban area and Clouds all methods perform nearly equally (the variation is within 5% approximately) well, however for other classes the performances differ significantly. For Agriculture, the second largest class, Method 2 and Method 3 have accuracy of over 83%, while Method 1 and Method 4 are 68.49% and 60.12% accurate respectively. For the class Bare ground, Method 2 and Method 3 outperform the others comfortably. For the class Shadow, there is a huge performance gap between the {Method 2, Method 3} ($\approx 96\%$) and the {Method 1 (49%), Method 4 (40%)}. However, since the frequency of the Shadow class is very small, this variation does not affect the overall accuracy significantly.

Now we use the modified Method 4 (Eqs. 6.22-6.26) which incorporates a weight factor w that controls the contribution of the information from the neighborhood in the decision making process. The value $w = 1.0$ makes the method same as the original Method 4. To find the optimal values of w we use a 100×100 sub-image from each image and find optimal w based on these blocks. We use grid search to find the optimal w. Note that, the rule bases are the same as used earlier. For example, for Satimage2, for each of the four rule sets we find the optimal w using the classification error on the selected block of image. Figure 6.3 depicts the variation of classification error as a function of w for Satimage2. It is interesting to observe that for Satimage2 for all four rule bases the best performances are achieved around $w = 0.35$. So we should use the modified Method 4 with $w = 0.35$ for each of the four rule bases. Table 6.5 displays the classification errors on the whole image for $w = 0.35$. Comparing Table 6.5 with column 6 of Table 6.1, we find a consistent improvement with $w = 0.35$ in all four cases. The improvement varied between 2.14% and 2.75%. We also tried to find the optimal w for each of the four rule bases for Satimage3. Figure 6.4 displays the classification error as a function of w. In this case, for all four rule bases we find an optimal value of $w = 1.0$. This again confirms the fact that when different landcover types form spatially compact clusters, neighbor

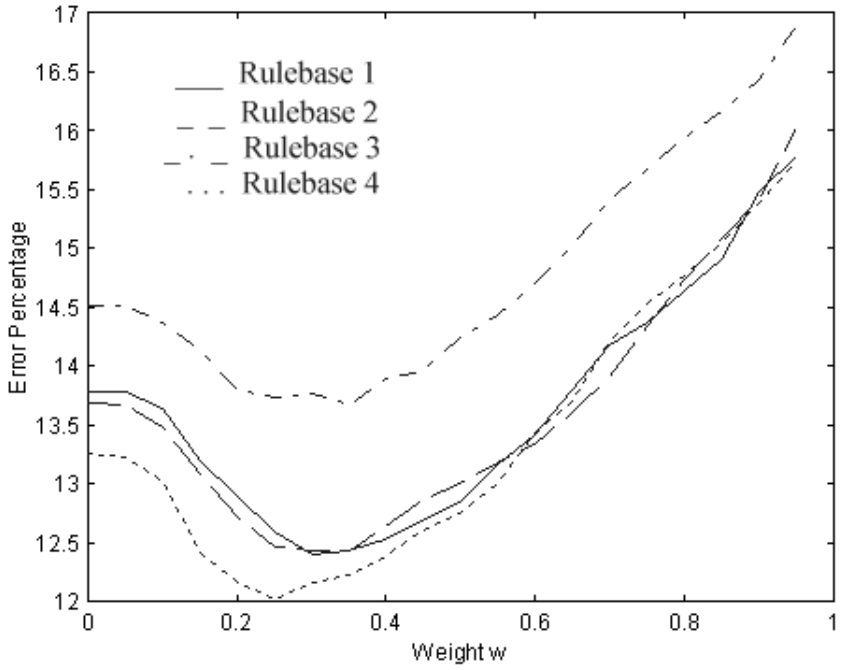

Figure 6.3: The result of grid search for optimal value of w using a 100×100 sub-image of Satimage2.

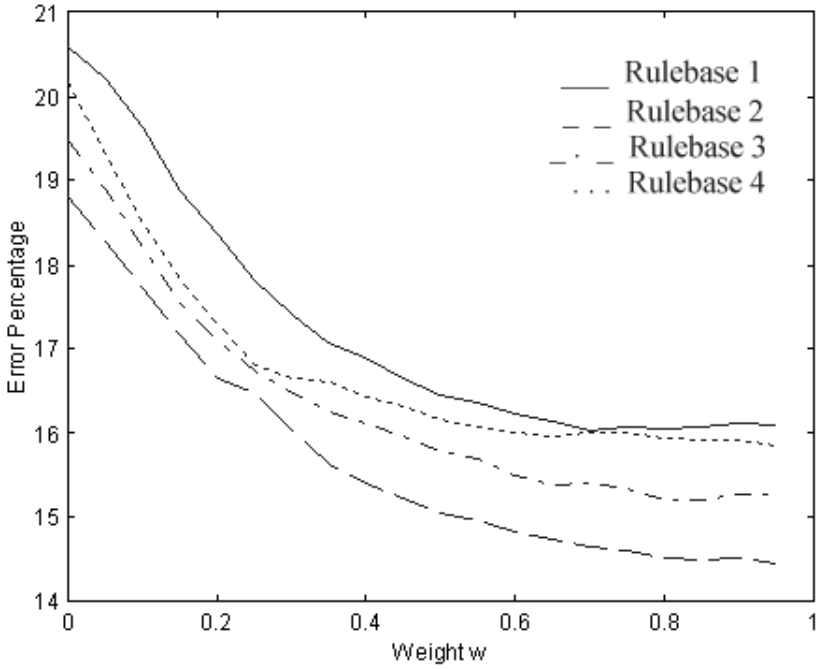

Figure 6.4: The result of grid search for optimal value of w using a 100×100 sub-image of Satimage3.

Table 6.5: Classification performances of classifiers using modified Method 4 with $w = 0.35$ on Satimage2

Training Set	1	2	3	4
Error rate	11.10%	12.24%	11.31%	10.33%

and central pixels play equally important roles in the decision making.

6.7 Conclusion

The problem of landcover classification in multispectral satellite images has some unique features that can be used to achieve better performance. To exploit this advantage we have developed decision making schemes that use information from spatial neighborhood. For assigning a class label to a pixel the schemes use information generated by the fuzzy rule base extracted in he previous chapter. The fuzzy rule base generates possibilistic label vectors for the pixel as well as its eight spatial neighbors. These schemes are based on the rationale that each fuzzy rule captures information regarding the characteristic of a particular landcover type and in an image, more often than not, neighboring pixels belong to the same landcover type.

So for most of the pixels (we may call them *pure* pixels) the maximum firing strength produced by the rule base is high and considerably higher than the firing strengths of all other rules. So the classification scheme developed in the previous chapter works well for such pixels. However, in an image there exists a considerable number of pixels which correspond to the boundary regions of landcover objects from different classes. The spectral signatures captured in such pixels usually do not match well with the characteristic signature of any particular class. We call such pixels *mixed* pixels. For such a pixel, though there will always be a rule that gives the highest firing strength, there may be some other rules whose firing strengths could be pretty close to the highest. Thus we face a problem of *lack of confidence* while making a decision. So, to make a better decision we need to look for more information. Here we have tried to exploit the information provided by the spatially neighboring pixels. We have proposed four different schemes for aggregation of spatial neighborhood information for decision making. Three of these schemes use evidence theoretic framework, the remaining one is based on the fuzzy k-NN rule. Our extensive experiments using Satimage2 and Satimage3 have revealed very good performances by the four proposed methods.

For pure pixels the proposed schemes simply enhance the confidence in the decision that

would have been made considering firing strength directly. However, for mixed pixels these schemes allow one to make a better decision by taking into account the context (spatial neighborhood) information. The suitability of a particular scheme depends to some extent on the nature of the image to be classified. Specifically, Method 1 and Method 4 are more suitable for images having spatially compact classes, while Method 2 and Method 3 are better for images when different classes are sparsely distributed all over the image. One can use the performance of the methods on a validation data to decide on the best classifier for a given situation. One of the methods has a parameter that can be tuned based on the training data to get a classifier with improved performance.

Although, the four proposed classifiers aggregated contextual information, generated by a fuzzy rule based system, the aggregation methods can be used with any other classifier which can produce a fuzzy/probabilistic/possibilistic label vector for each pixel. For example, one can use the output of a neural network and generate such label vectors and the aggregation methods can be used on them.

Chapter 7

Designing Vector Quantizer with SOM and Surface Fitting for Better Phychovisual Quality [1]

[1] Content of this chapter has been published in [205].

7.1 Introduction

So far we have discussed classifier designs based on prototypes generated by Self-organizing Maps. In this chapter and Chapter 8 we consider a different patten recognition application of SOM, i.e., image compression. With the advent of World Wide Web and proliferation of multimedia contents, data compression techniques have gained immense importance. Data compression has become an enabling technology for efficient storage and transmission of multimedia data. In this chapter we propose a method for image compression by vector quantization of the image [110] using Self-organizing Map [169] algorithm. We also propose refinement of the codebook using a method of bicubic surface fitting for reduction of psychovisually annoying *blocking effect*.

A vector quantizer (VQ) [110] $\mathcal{Q}$ of dimension k and size S can be defined as a mapping from data vectors (or "points") in k-dimensional Euclidean space, $\Re^k$ into a finite subset $\mathcal{C}$ of $\mathcal{R}^k$. Thus,

$$\mathcal{Q} : \Re^k \rightarrow \mathcal{C} \tag{7.1}$$

where $\mathcal{C} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_S\}$ is the set of S reconstruction vectors, called a *codebook* of size S and each $\mathbf{y}_i \in \mathcal{C}$ is called a *code vector* or *codeword*. For each $\mathbf{y}_i, i \in \mathcal{I} = \{1, 2, ..., S\}$ is called the index of the code vector and $\mathcal{I}$ is the index set. Encoding a data vector $\mathbf{x} \in \Re^k$ involves finding the index j of the code vector $\mathbf{y}_j \in \mathcal{C}$ such that $\|\mathbf{x}-\mathbf{y}_j\| \leq \|\mathbf{x}-\mathbf{y}_i\| \ \forall\, i \neq j$ and $i, j \in \mathcal{I}$. The decoder uses the index j to look-up the codebook and generates the reconstruction vector $\mathbf{y}_j$ corresponding to $\mathbf{x}$. The distortion measure $d(\mathbf{x}, \mathbf{y}_j) = \|\mathbf{x}-\mathbf{y}_j\|$ represents the penalty of reproducing $\mathbf{x}$ with $\mathbf{y}_j$. If a VQ minimizes the average distortion, it is called the optimal VQ of size S.

Vector quantization has been used for image compression successfully by many researchers [9, 16, 121, 148, 144, 184, 249, 250]. The oldest as well as most commonly used method is the generalized Lloyd algorithm (GLA) [228] or LBG algorithm [222]. The algorithm is similar to the *k-means* algorithm used for statistical cluster analysis [300]. GLA is an iterative gradient descent algorithm that tries to minimize an average squared error distortion measure. Design of an optimal VQ using GLA has been proposed and studied in [222]. However, GLA being a greedy algorithm, its performance is sensitive to initialization and it converges to the local minima closest to the initial point. Fuzzy k-means algorithms [31] and several other fuzzy vector quantization techniques have been studied and used for image compression in [148]. Zeger *et al.* [368] proposed methods for designing globally optimal vector quantizer using stochastic relaxation techniques and simulated annealing.

Self-organizing Map (SOM) [169], as described in Chapter 2, is a competitive learning neural network with excellent vector quantization capability along with several other

interesting properties. Consequently, it has attracted attention of the researchers in the field of vector quantization [9, 249, 356]. The learning scheme of SOM is an application of the least mean square (LMS) algorithm where the weights of the neurons are modified "on the fly", for each input vector, as opposed to the usual batch update scheme of GLA. Thus the codebook is updated using an instantaneous estimate of the gradient, known as stochastic gradient, which does not ensure monotonic decrease of the average distortion. Therefore, the algorithm has a better chance of not getting stuck at a local minima. GLA can also incorporate incremental update through purely competitive learning. However, due to incorporation of neighborhood update (opposed to the "winner only" update in pure competitive learning) in the training stage, SOM networks exhibit the interesting properties of topology preservation and density matching [169]. The former means that the vectors near-by in input space are mapped to the same node or nodes near-by in the output space (lattice plane of the SOM nodes). The density matching property refers to the fact that after training the distribution of the weight vectors of the nodes reflects the distribution of the training vectors in the input space. Thus, more code vectors are placed in the regions with high density of training vectors. In other clustering algorithms, a dense and well separated cluster is usually represented by a single cluster center. Though it is very good if the clusters are subsequently used for pattern classification task, in case of vector quantization, where the aim is to reduce the reconstruction error, this may not be that good.

The total reconstruction error of a VQ is the sum of *granular error* and *overload error* [110]. The granular error is the component of the quantization error due to the granular nature of the quantizer for an input that lies within a bounded cell. Due to the density matching property, SOM places several prototypes in a densely populated region and thus makes the quantization cells small in such area. This leads to the reduction in the granular error component resulting in preservation of finer details. This is a highly desirable property in a VQ. The overload error component of quantization error arises when the input lies in any unbounded cell representing data at the boundary of training sample distribution. Since the distribution of the codewords replicate the distribution of the training data, the overload error is also low when the distribution of the test data is well represented by that of the training data.

In [47] Chang and Gray introduced an on-line technique for VQ design using stochastic gradient algorithm which can be considered a special case of the SOM algorithm and it is shown to perform slightly better than GLA. Nasrabadi and Feng [249] also used SOM for VQ design and demonstrated performance better than or similar to GLA. Yair *et al.* [356] used a combination of SOM and stochastic relaxation and obtained consistently better performance than GLA. Amerijckx *et al.* [9] used SOM algorithm to design a VQ for the coefficients of discrete cosine transform of the image blocks. The output of

VQ encoder is further compressed using entropy coding. They reported performance equivalent to or better than standard JPEG algorithm.

In this chapter we propose a scheme for designing a spatial vector quantizer (SVQ) for image compression using the SOM [169] algorithm and surface fitting. We use a set of training images to design a generic codebook that is used for encoding the training as well as other images. The codebook is designed with *mean-removed* (also known as *residual*) [110, 16, 184] vectors. The mean is added to the reproduced vectors by the decoder. An initial codebook is generated by training a SOM with the training vectors. Then to achieve better psychovisual fidelity, each codevector is replaced with the best-fit cubic surface generated by the training vectors mapped to the respective codevector. The set of code indexes produced by the encoder is further compressed using Huffman coding. A scheme of difference coding for the average values of the blocks is used. The difference coding enables us to utilize Huffman coding for the averages which also leads to more compression.

In Section 7.2 we describe the VQ design scheme in detail. In 7.3 we present the scheme for processing the block averages for efficient Huffman coding. We report the experimental results in Section 7.4. The chapter is concluded in 7.5. We use *peak signal to noise ratio* (PSNR) as one of the performance measures of the VQ. The PSNR in deciBel (dB) for a 256 level image of size $h \times w$ is defined as

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{\frac{1}{hw} \sum_{i=1}^{h} \sum_{i=1}^{w} (x_{ij} - \hat{x}_{ij})^2}, \tag{7.2}$$

where x_{ij} is the value of the ij-th pixel in the original image and $\hat{x}_{ij}$ is that of the reconstructed image. In addition, we also propose two indexes for quantitative assessment of psychovisual quality in the context of blocking effect. We use them to demonstrate the improvement of reconstruction quality by the surface fitting codebooks over the initial SOM generated codebooks.

7.2 Scheme for designing vector quantizers

Here we adopt a multi-stage approach for developing a comprehensive scheme for designing vector quantizers to achieve good psychovisual quality of the reconstructed image with high compression ratio. First we prepare mean-removed (residual) training vectors from a set of training images. A 2-dimensional SOM is trained with them to obtain the initial codebook. Then surface fitting scheme is employed for improving the quality of the codevectors. We also develop a difference coding scheme for the block averages. Finally, variable length codewords are assigned to the difference coded block averages

as well as the block indexes to achieve further compression through Huffman coding. In following we describe our scheme in full detail.

7.2.1 Preparing the image vectors

To produce the feature vectors an $h \times w$ image is divided into $p \times q$ blocks. Then each block is represented by a vector $\mathbf{x} \in \mathcal{R}^k$, where $k = pq$ and each component x_i of $\mathbf{x}$ represents the value of (lexicographically ordered) i-th pixel in the block. The pixel values in the $p \times q$ image block are presented in *row-major* fashion in the feature vector $\mathbf{x}$. We convert each vector $\mathbf{x}$ to a mean-removed or residual vector $\mathbf{x}'$ such that $\mathbf{x}' = \mathbf{x} - \bar{x}\mathbf{1}$ where $\bar{x} = \frac{1}{k}\Sigma_{i=1}^{k} x_i$ is the average pixel value of the block and $\mathbf{1}$ is a k-dimensional vector with all components 1. We use the residual vectors to train the SOM as well as for encoding of the images. The averages are separately transmitted to the decoder and added to the reproduction vectors such that the reconstructed vector $\hat{\mathbf{x}} = \mathbf{y}_i + \bar{x}\mathbf{1}$, where $\mathbf{y}_i$ is the code vector corresponding to $\mathbf{x}$. This conversion essentially has the effect of reducing the dimension of the input space by one, i.e., from k to $k-1$ and thereby can reduce the potential number of clusters. To demonstrate this effect of the process here we present some results using 512×512 Lena image. The original image is shown in Figure 7.1. We divide the image into 8×8 blocks which produces 64 dimensional vectors. We use the vectors to train a 16×16 SOM to prepare a codebook of size 256. The weight vectors of SOM nodes are used as reproduction vectors. The reconstructed image is shown in Figure 7.2(a). The PSNR achieved is 27.62 dB. Figure 7.2(b) shows the result of the same experiment using the mean-removed vectors. In the second case the PSNR achieved is 29.61 dB.

Thus, using mean-removed vectors significant improvement of VQ performance can be achieved. Hereafter in this paper the term vector will be used for mean-removed vectors unless explicitly stated otherwise.

7.2.2 Preparing training data for construction of a generic codebook

In most of the experimental works on vector quantization the codebooks are trained with the test image itself. However, this poses a problem in practical use of such algorithm for transmission/storage of the compressed image. While transmitting a compressed image, both the compressed image and the codebook must have to be transmitted. The overhead in transmitting the codebook diminishes the compression ratio to considerable extent. This problem can be avoided if a generic codebook is used by both the transmitter and

Figure 7.1: Original Lena image.

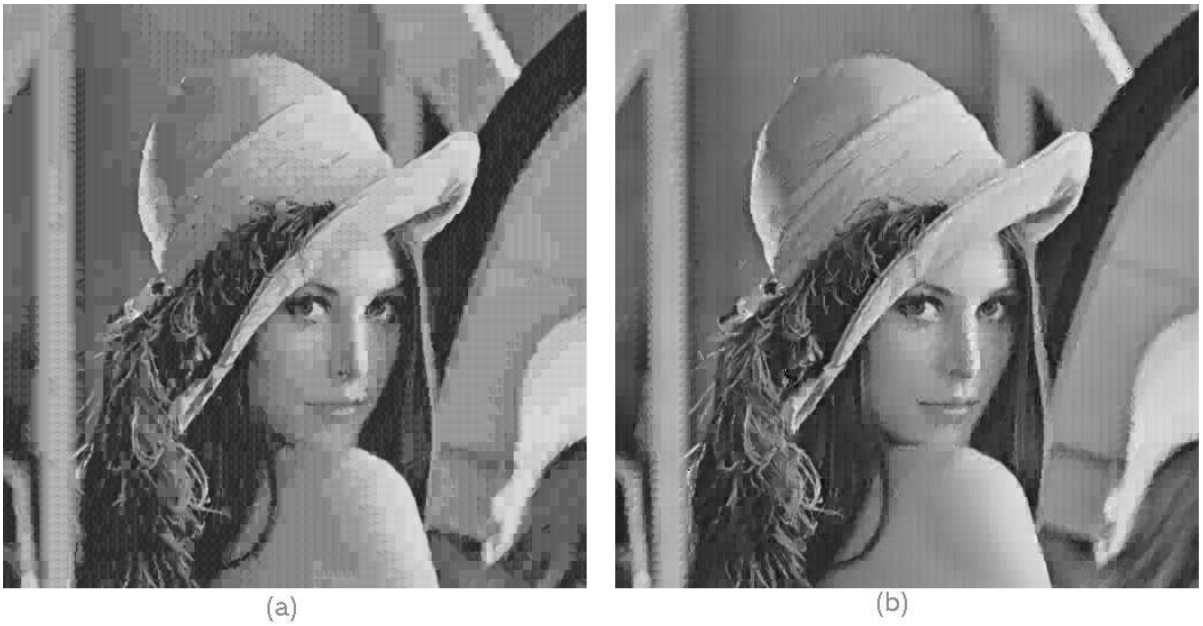

Figure 7.2: (a) Reconstructed Lena image with VQ using original vectors. PSNR = 27.62 dB. (b) Reconstructed Lena image with VQ using mean-removed vectors. PSNR = 29.61 dB. Here the VQ is trained with the Lena image only.

Figure 7.3: The training image.

the receiver [300]. Construction of such a generic codebook poses a formidable problem, since in general, if an image is compressed using a codebook trained on a different image, the reconstruction error tends to be substantially high.

Here we explore the possibility of construction of a generic codebook that can be used to encode any image (i.e., images other than those used to construct the codebook) with acceptable fidelity. Such a codebook can allow us one time construction of the encoder and the decoder and making the codebook a permanent part of it. To achieve this, we select a set of images having widely varied natures in terms of details, contrasts and textures. We use these images together for construction of the generic codebook. We prepare a 768×512 training image (Figure 7.3) which is a composite of six smaller 256×256 images and train the SOM using this training image and then construct the surface fitting codebook for the VQ.

We emphasize that, the generic codebook constructed by the above mentioned method

can neither be claimed as *universal* nor as the *best possible generic codebook*. Our aim is merely to demonstrate experimentally that a generic codebook constructed from some judiciously chosen images can be used for effective compression of other images having "similar characteristics". If the images to be compressed are of radically different nature, say, containing texts or geometric line drawings, use of another generic codebook constructed with images of similar type will be appropriate. Note that, when we say images of "similar" characteristics, we do not refer to images which are visually similar, but images with similar distribution of gray level over small blocks of size, say, 8×8.

7.2.3 Surface fitting

Once the SOM is trained, the codebook can readily be designed using the weight vectors as the reconstruction vectors. Images can be encoded by finding out, for each image vector, the code vector with the least Euclidean distance. However, all spatial vector quantizers produce some blockyness in the reconstructed image [250, 302, 288], i.e., in the reconstructed image the boundaries of the blocks become visible. Even though the reconstructed image shows quite good PSNR, this effect often has some adverse psychovisual impact. Often transform and/or subband coding is used to overcome this effect. But this adds substantial computational overhead since the raw image has to be transformed into frequency domain before encoding using the quantizer and also in decoder the image has to be converted back into the spatial domain from the frequency domain.

In our method we adopt a scheme of *polynomial* surface fitting for modifying the code-vectors generated by SOM algorithm that reduces the blockyness of the reconstructed image and improves its psychovisual quality. In this case the computational overhead occurs only at the codebook design stage, not during encoding or decoding of an image. Although in computer graphics polynomial surfaces serve as standard tools for modelling the surface of graphical objects [274], in image coding their application is mostly restricted to image segmentation and representation of segmented patches. In [194, 305] low degree polynomials are used for local approximation of segmented patches, while in [38] Bezier-Bernstein polynomials are used for image compression by globally approximating many segmented patches by a single polynomial with local corrections. To the best of our knowledge there is no reported work that uses polynomial surfaces in context of vector quantization.

For using polynomial surfaces one has to decide on the degree of the surface to be used. Lower degree polynomials are easier to compute but they have less expressive power. On the other hand, higher degree polynomials, though have more expressive power, they tend to replicate the training data exactly and lose the power of generalization, and thus

unacceptable results may be produced when presented with an image not used in the training set. There is also the issue of solving for the coefficients of the polynomial. If a small block size is used, there may not be enough points in a block to find a solution of a high degree polynomial.

In our work we have experimented with block sizes 4×4, 4×8 and 8×8. It is found that for 4×4 blocks biquadratic surfaces give performances comparable to that of bicubic surfaces, while for larger block sizes bicubic surfaces perform significantly better. Surfaces of degree 4 do not improve the performance significantly for test images. So we use the bicubic surfaces throughout our work.

In this scheme we try to find for each reconstruction block a cubic surface centered at the middle of the block so that the gray value x of a pixel at (α, β) (with respect to the origin set at the middle of the block) can be expressed by the *bicubic* equation,

$$\begin{aligned} x(\alpha,\beta) &= a_0 + a_1\alpha^3 + a_2\beta^3 + a_3\alpha^2\beta + a_4\alpha\beta^2 \\ &\quad + a_5\alpha^2 + a_6\beta^2 + a_7\alpha\beta + a_8\alpha + a_9\beta \\ &= \mathbf{p}^T\mathbf{a} \end{aligned} \tag{7.3}$$

where

$$\mathbf{p} = [1, \alpha^3, \beta^3, \alpha^2\beta, \alpha\beta^2, \alpha^2, \beta^2, \alpha\beta, \alpha, \beta]^T$$

and

$$\mathbf{a} = [a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9]^T.$$

Thus, for a whole block containing k pixels we can write

$$P = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_k^T \end{bmatrix}_{k \times 10}, \tag{7.4}$$

where $\mathbf{p}_i$ is the $\mathbf{p}$ vector for i-th component of feature vector $\mathbf{x}$.

To find the coefficients of the polynomials corresponding to the codevectors generated by the SOM, we partition the training vectors into k subsets such that a training vector belongs to i-th subset, if it is mapped to the i-th codevector. Let $X = \{\mathbf{x} \in \Re^k\}$ be the set of training vectors and $X^{(i)} = \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, ..., \mathbf{x}_{n_i}^{(i)}\} \subset X$ where $n_i = \mid X^{(i)} \mid$, be the set of training vectors mapped to the i-th code vector. We write for the set $X^{(i)}$

$$\mathbf{G}^{(i)} = \begin{bmatrix} \mathbf{x}_1^{(i)} \\ \mathbf{x}_2^{(i)} \\ \vdots \\ \mathbf{x}_{n_i}^{(i)} \end{bmatrix}_{kn_i \times 1}$$

and

$$\mathbf{a}^{(i)} = [a_0^{(i)}, a_1^{(i)}, ...a_9^{(i)}]^T.$$

Then Eq. (7.3) can be written in a matrix form for all the vectors mapped to the i-th code vector as

$$\mathbf{G}^{(i)} = \mathbf{P}^{(i)}\mathbf{a}^{(i)}, \tag{7.5}$$

where

$$\mathbf{P}^{(i)} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{n_i} \end{bmatrix}_{kn_i \times 10} \quad \text{and } P_i = P \ \ \forall\ i = 1, 2, ..., n_i.$$

Thus the total squared error due to reconstruction of all blocks corresponding to vectors in $X^{(i)}$ using the i-th reconstruction block generated by cubic surface specified by the coefficient vector $\mathbf{a}^{(i)}$ can be expressed as

$$E = \|\mathbf{G}^{(i)} - \mathbf{P}^{(i)}\mathbf{a}^{(i)}\|^2. \tag{7.6}$$

Differentiating (7.6) with respect to $\mathbf{a}^{(i)}$ we obtain

$$\frac{\partial E}{\partial \mathbf{a}^{(i)}} = \mathbf{P}^{(i)T}(\mathbf{G}^{(i)} - \mathbf{P}^{(i)}\mathbf{a}^{(i)}). \tag{7.7}$$

Thus the coefficient vector $\mathbf{a}^{(i)}$ corresponding to the minimum of E can be obtained by solving the equation

$$\mathbf{P}^{(i)T}(\mathbf{G}^{(i)} - \mathbf{P}^{(i)}\mathbf{a}^{(i)}) = 0 \tag{7.8}$$

for $\mathbf{a}^{(i)}$. From (7.8) we obtain

$$\mathbf{a}^{(i)} = (\mathbf{P}^{(i)T}\mathbf{P}^{(i)})^{-1}\mathbf{P}^{(i)T}\mathbf{G}^{(i)} = \mathbf{P}^{(i)+}\mathbf{G}^{(i)}, \tag{7.9}$$

where $\mathbf{P}^{(i)+} = (\mathbf{P}^{(i)T}\mathbf{P}^{(i)})^{-1}\mathbf{P}^{(i)T}$ is the *pseudoinverse* of $\mathbf{P}^{(i)}$.

We compute the coefficient vectors $\mathbf{a}^{(i)}$s for all codevectors obtained from SOM. Once the $\mathbf{a}^{(i)}$s are available, the codebook can be designed to contain the coefficients $\{\mathbf{a}^{(i)}\}$. If we do so, then for encoding we have to do the following:

1. For each polynomial ($\mathbf{a}^{(i)}$), generate the surface by computing each of the pq real values corresponding to the position of the pixels of a block.

2. Round these real values subject to the constraint that all values lie in $\{0, 1, ..., L-1\}$, where L is the number of gray value levels.

Notice here, that given a coefficient vector $\mathbf{a}^{(i)}$, the surface generated $\hat{\mathbf{x}}_i$ remains the same irrespective of the spatial location of the associated block on the image. The reason is that for every block we use the central point of the block as the origin. So we can avoid a lot of useless computation, if instead of storing $\mathbf{a}^{(i)}$s, we store the $\hat{\mathbf{x}}_i$s in the code book. Let these generated surfaces be $\{\hat{\mathbf{x}}_i, i = 1, 2, ..., S\}$. Then for every block of size pq of the image we find the closest $\hat{\mathbf{x}}_k$ and use its index to encode. Then, while decoding, we reverse the procedure using $\hat{\mathbf{x}}_i$s and the indexes used to code the blocks. In other words, *we do not store the* $\mathbf{a}^{(i)}s$ *but the code vectors reconstructed using* $\mathbf{a}^{(i)}s$.

We use the new codebook for encoding as well as decoding subsequently. We have tested the effectiveness of this method using several images. Although the quality improvement in terms of PSNR appears marginal, significant improvement in terms psychovisual quality is observed consistently. Figure 7.4(a) shows the reconstructed Lena image using SOM codebook (block size 8×8 and codebook size 256), while Figure 7.4(b) depicts the same using surface fitting codebook. Though the surface fitting codebook marginally increases the PSNR (0.3 dB) for this image, reduction of blockyness and improvement of psychovisual quality are evident. To demonstrate the effect more clearly we show in Figure 7.5 enlarged portions (containing lips) of the images in Figure 7.4(a) and Figure 7.4(b).

7.3 Processing the block averages and indexes for Huffman coding

Usually in image compression schemes using vector quantization, the indexes are encoded using some lossless entropy coding scheme such as Huffman coding to achieve further compression [110]. The rationale behind it is that the codewords do not appear uniformly in a quantized image, rather there is a distinct non-uniform distribution of codewords. Thus encoding the indexes with variable length codes can achieve more compression. In the vector quantization scheme proposed here one needs to store/transmit one code index and a block average value for each quantized block of the image. In this section we develop a suitable transformation for the block average values and describe a scheme for efficient Huffman coding of the indexes and the block averages.

Figure 7.4: (a) Reconstructed Lena image with VQ using SOM weight vectors as reconstruction vectors. PSNR = 28.19 dB. (b) Reconstructed Lena image with VQ using surface fitting. PSNR = 28.49 dB.

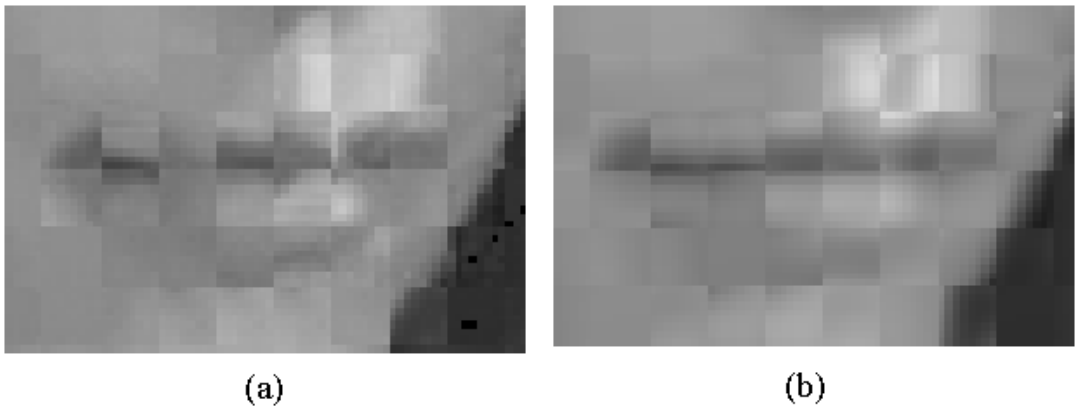

Figure 7.5: (a) Enlarged portion of Lena image shown in figure 3(a). (b) Enlarged portion of Lena image shown in figure 3(b).

7.3.1 Difference coding of the block averages

An image of size $h \times w$ divided into $p \times q$ blocks produces $N = \frac{hw}{pq}$ blocks. In our scheme, while encoding an image, for each block the encoder produces one codebook index and one average pixel value and for the whole image the blocks are processed in a row-major fashion. The encoded image is represented by a set of indexes $I = \{\ell_i : \ell_i \in \mathcal{I}, i = 1, 2, ..., N\}$ and the set of block averages $M = \{m_i : m_i \in [0, 255], i = 1, 2, ..., N\}$. Thus, an average value can be rounded off to nearest integer and encoded with a byte value. However, in images substantial correlation exists among the neighboring blocks. So the difference of the average values between neighboring blocks is usually very small. In the set of differences of block averages $D = \{d_i : d_1 = m_1 \quad \text{and} \quad d_i = m_i - m_{i-1}, i = 2, 3, ..., N\}$, most of the d_is will have small absolute values and thus can be encoded more efficiently using Huffman coding scheme. But this will make the potential range of d_is [-255,255], thereby requiring 511 codewords for Huffman encoding.

Here we develop a scheme of transformation of difference coding that requires 256 codewords for Huffman encoding. The scheme is as follows:

Step 1: Convert the set of averages M to $M' = \{m'_i : m'_i = 2 * (int)(\frac{m_i}{2.0} + 0.5), i = 1.2, ..., N\}$.

Step 2: Compute the set of differences of block averages $D = \{d_i : d_1 = m'_1 \quad \text{and} \quad d_i = m'_i - m'_{i-1}, i = 2, 3, ..., N\}$.

Step 3: Due to the conversion in Step 1 all the d_is are even. So we can use all positive odd numbers to represent the negative difference values. We convert D to $D' = \{d'_i : \text{if} \quad d_i \geq 0 \quad \text{then} \quad d'_i = d_i \quad \text{otherwise} \quad d'_i = -(d_i + 1)\}$.

Thus, the transformation encodes each positive difference value to its nearest positive even number, while each negative difference value is represented by a positive odd number nearest in absolute value. This scheme will allow the decoder to identify the positive and negative difference values unambiguously. The loss in information due this transformation is negligible.

Usually in the images large portions can be found that has no or little variation in brightness. When such a region is divided into blocks, the average gray value of two neighboring blocks differ very little. Thus, all over the image the difference of average gray values of neighboring blocks are more likely to be small. To demonstrate this fact we display in Figure 7.6 the frequency distributions of actual averages as well as the difference coded averages for the Lena image quantized with 8×8 blocks. Similar distributions are observed for all the other images we studied. It is evident from Figure

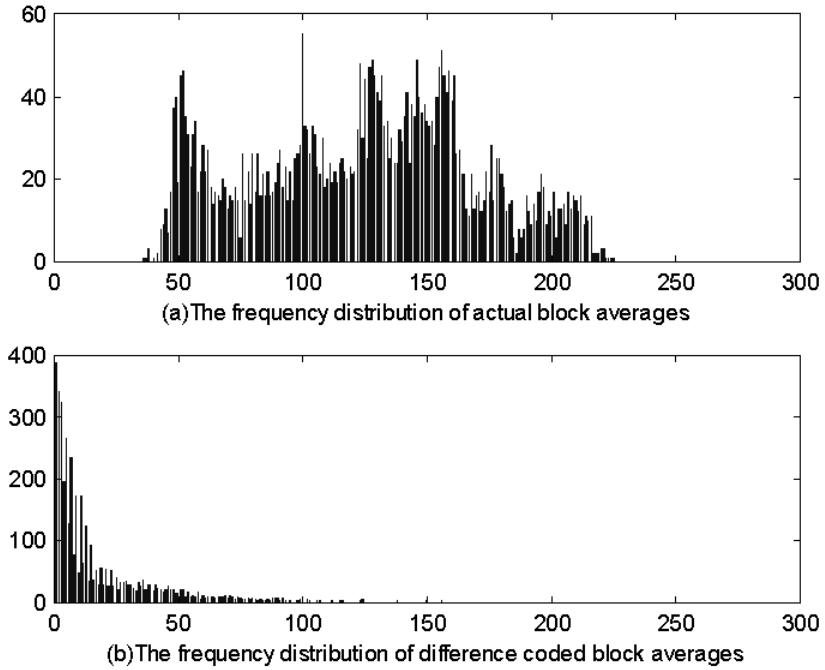

Figure 7.6: The frequency distributions of (a) actual block averages and (b) difference coded block averages for Lena image.

7.6(b) that the distribution of the difference coded average values can be approximated as a *monotonically decreasing* distribution and significant data compression can be achieved using Huffman coding scheme to encode the difference coded averages.

7.3.2 Huffman coding of the indexes and difference coded averages

Huffman coding [135] is a lossless compression scheme that achieves nearly optimal performance in terms of entropy. The scheme involves assigning variable length codewords to the symbols in a set of input symbols. The higher is the probability of occurrence of an input symbol, shorter is the codeword assigned to it. However, the implementation of the scheme requires *a priori* knowledge of the input probability. This information is often not available in practice.

In our scheme we use mean-removed vectors for VQ design. These vectors in effect contain the information about the variation among the components. Further, due to use of SOM clustering algorithm, the distribution of code vectors in $\Re^k$ approximates the distribution of the input vectors. Since a wide variety of images is used for training the system, the distribution of the index values of training image is expected to give a fairly good approximation of the distribution of indexes for other images encoded using the

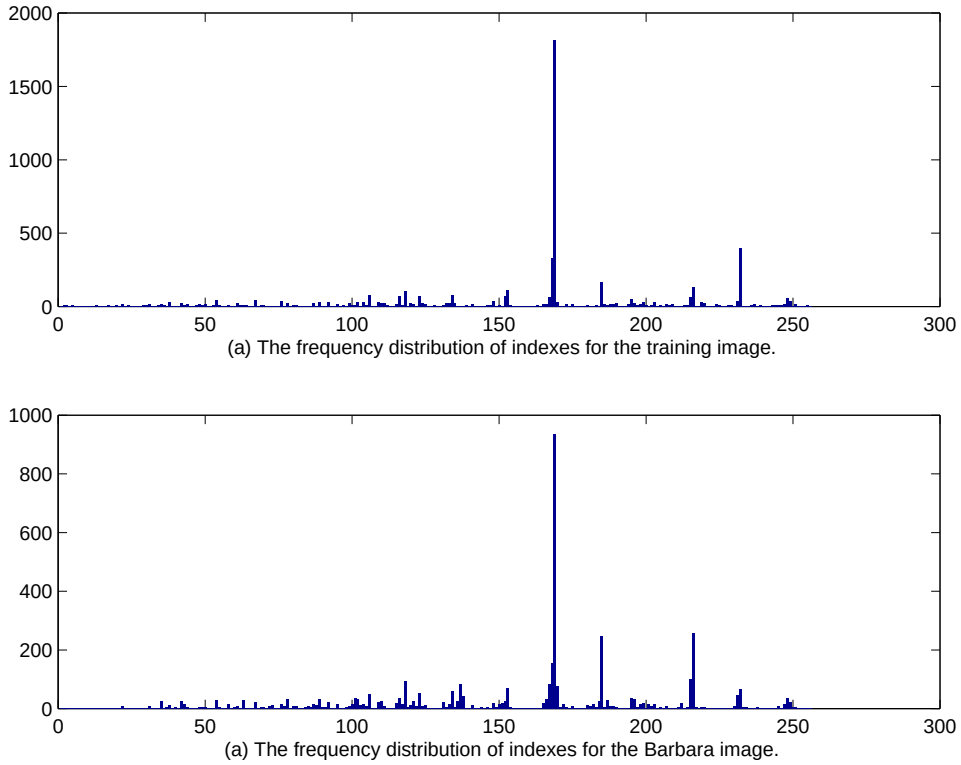

Figure 7.7: The frequency distributions of (a) indexes of encoded training image and (b) indexes of encoded Barbara image.

same codebook. This can be seen from Figure 7.7, that displays the distribution of (a) indexes of the training image and (b) indexes of the Barbara image of size 512×512, both encoded using the same codebook.

Hence, for a given codebook we can use the knowledge of distribution of indexes for the training image as (approximate) *a priori* knowledge about the distribution of indexes for any image encoded using the same codebook. Thus, the same assignment of variable length codewords to the index values can be used for Huffman coding of indexes of the images (not used in the training) encoded using the same codebook. Also the difference coding scheme for the block averages allows us to approximate the distribution of average values as a monotonically decreasing one. So the use of the same set of variable length codewords for the difference coded average values enables us to compress the averages for all encoded images.

7.4 Experimental results

We study 3 vector quantizers using block sizes 8×8 (VQ1), 4×8 (VQ2) and 4×4 (VQ3) respectively. Each of the VQs uses a codebook of size 256 (constructed with a 16×16 SOM)and is trained with *mean-removed vectors*. Then the codebooks are

Table 7.1: Performance of the vector quantizers on training images.

Image	Size	VQ1 (Blocksize 8×8)		VQ2 (Blocksize 4×8)		VQ3 (Blocksize 4×4)	
		PSNR (dB)	CR (bpp)	PSNR (dB)	CR (bpp)	PSNR (dB)	CR (bpp)
Blood Cell	256×256	28.52	0.21	30.75	0.44	33.50	0.89
Cameraman	256×256	23.58	0.16	24.65	0.34	26.51	0.65
Chair	256×256	31.25	0.14	32.39	0.28	34.99	0.54
Keyboard	256×256	30.28	0.17	31.01	0.36	33.92	0.67
Lena	256×256	26.42	0.21	27.82	0.44	30.49	0.82
Peppers	256×256	25.49	0.22	27.05	0.45	29.36	0.86

modified applying the surface fitting method. In later part of this section when we report the study on improvement of psychovisual quality due to surface fitting, we distinguish them as "SOM VQ" and "surface fitting VQ". Thus, to represent each block in the encoded image 1 byte is required for the index and another byte is required for the block average. This makes the compression ratios 0.25 bpp (bits per pixel), 0.5 bpp and 1 bpp for VQ1, VQ2 and VQ3 respectively. To achieve more compression, lossless Huffman encoding is applied separately to the indexes and the block averages. The codeword assignment for the indexes is based on the frequency distribution of the codevectors in the encoded training image. Because of strong correlation between neighboring blocks, the absolute differences between average values of neighboring blocks are found to have a monotonically decreasing distribution and codewords are assigned exploiting this.

Each VQ is trained in two stages using SOM algorithm and surface fitting method using the image shown in Figure 7.3. The training image is a composite of six 256×256 images. The individual images in the training set are **Blood cell, Peppers, Keyboard, Lena, Cameraman** and **Chair**. We report the test results for six images, of which **Lena, Barbara** and **Boat** have size 512×512 and **Bird, House** and **Mattface** are of size 256×256. Please note that the Lena images used in training and test set are different. The former is of size 256×256 while the later has the size 512×512. The performances of the vector quantizers for the training images are summarized in Table 7.1 and those for the test images are summarized in Table 7.2. In Table 7.2 we also present the performances of standard JPEG algorithm. For every image, while using the JPEG algorithm we tried to maintain the same compression rates as achieved by our scheme. We used the routines available in Matlab 5 software to generate the JPEG images.

Our results for the test images show that the proposed algorithm consistently performs better than the JPEG in terms of PSNR for the lowest bit rate studied. In this case the difference of PSNR value between the images produced by our scheme and JPEG varied between 0.23 dB for Lena image to 5.10 dB for the Bird image. For the higher bit rates JPEG shows consistently higher PSNR values than the proposed algorithm at

Table 7.2: Performance of the vector quantizers on test images and their comparison with baseline JPEG.

Image	Compression	VQ1 (Blocksize 8×8)		VQ2 (Blocksize 4×8)		VQ3 (Blocksize 4×4)	
	Algorithm	PSNR (dB)	CR (bpp)	PSNR (dB)	CR (bpp)	PSNR (dB)	CR (bpp)
Lena	Proposed	28.47	0.18	30.14	0.38	33.03	0.74
(512×512)	JPEG	28.24	0.20	33.28	0.38	36.43	0.73
Barbara	Proposed	23.85	0.19	24.33	0.41	25.09	0.79
(512×512)	JPEG	23.48	0.19	27.16	0.41	31.12	0.73
Boat	Proposed	26.44	0.17	28.11	0.35	31.11	0.69
(512×512)	JPEG	24.10	0.17	30.35	0.36	34.14	0.68
Bird	Proposed	30.27	0.15	32.56	0.30	35.60	0.57
(256×256)	JPEG	25.17	0.16	34.42	0.29	38.63	0.54
House	Proposed	25.39	0.16	26.39	0.33	29.51	0.67
(256×256)	JPEG	24.20	0.18	30.15	0.33	34.36	0.64
Mattface	Proposed	28.69	0.18	31.96	0.37	35.46	0.69
(256×256)	JPEG	27.04	0.19	35.35	0.37	39.60	0.66

similar compression rates. To compare the psychovisual qualities we display in Figure 7.8 the Lena images compressed using the proposed algorithm and JPEG algorithm. The images 7.8(a), 7.8(c) and 7.8(e) are compressed using the proposed algorithm with block sizes 8×8, 4×8 and 4×4 respectively. The images 7.8(b), 7.8(d) and 7.8(f) are compressed using the JPEG algorithm with bit rates similar to the corresponding images in the left panels. It is evident form the images shown in Figure 6 that despite a small difference in PSNR values at the lowest bit rate (0.23 dB), the image compressed by the proposed algorithm is quite superior to the JPEG image in terms of psychovisual quality. For higher bit rates though the JPEG images have significantly higher PSNR values, their psychovisual qualities are similar to the corresponding images compressed with the proposed algorithm. Similar results are obtained for other images studied in this paper. For all other test images the original images and the images reconstructed using the proposed algorithm for three VQs are shown in Figs. 7.9-7.11. The PSNR and the compression rates for these images (panels (b), (c) and (d) of Figs. 7.9-7.11) are reported in Table 7.2.

A plethora of studies on the Lena image are available in the literature. Many of them can be found in the excellent survey paper of Cossman et al. [64]. However, they have concentrated exclusively on vector quantization of image subbands and the results reported cannot be readily compared with those of the spatial vector quantization (SVQ) techniques. Instead we compare our results with other works using SVQ techniques. Zeger et al. [368] developed a variant of simulated annealing for VQ design and compared it with GLA. They used Lena and Barbara images of size 512×512 to test the algorithms. They used 4×4 block size and a codebook size of 256. The reported PSNRs are 30.48 dB and 25.80 dB respectively for GLA and 30.59 dB and 25.87 dB respectively for simulated annealing. Thus, the results reported here are comparable for Barbara image

(a) PSNR = 28.47 dB, CR = 0.18 bpp

(b) PSNR = 28.24 dB, CR = 0.20 bpp

(c) PSNR = 30.14 dB, CR = 0.38 bpp

(d) PSNR = 33.28 dB, CR = 0.38 bpp

(e) PSNR = 33.03 dB, CR = 0.74 bpp

(f) PSNR = 36.43 dB, CR = 0.73 bpp

Figure 7.8: (a), (c), (e) Compressed Lena images using proposed algorithm. (b), (d), (f) Compressed Lena images using JPEG.

Figure 7.9: Results on 512×512 Barbara image. (a) Original image, (b) reconstructed image for VQ with 8×8 blocks, (c) reconstructed image for VQ with 4×8 blocks and (d)reconstructed image for VQ with 4×4 blocks.

Figure 7.10: Results on 512×512 Boat image. (a) Original image, (b) reconstructed image for VQ with 8×8 blocks, (c) reconstructed image for VQ with 4×8 blocks and (d)reconstructed image for VQ with 4×4 blocks.

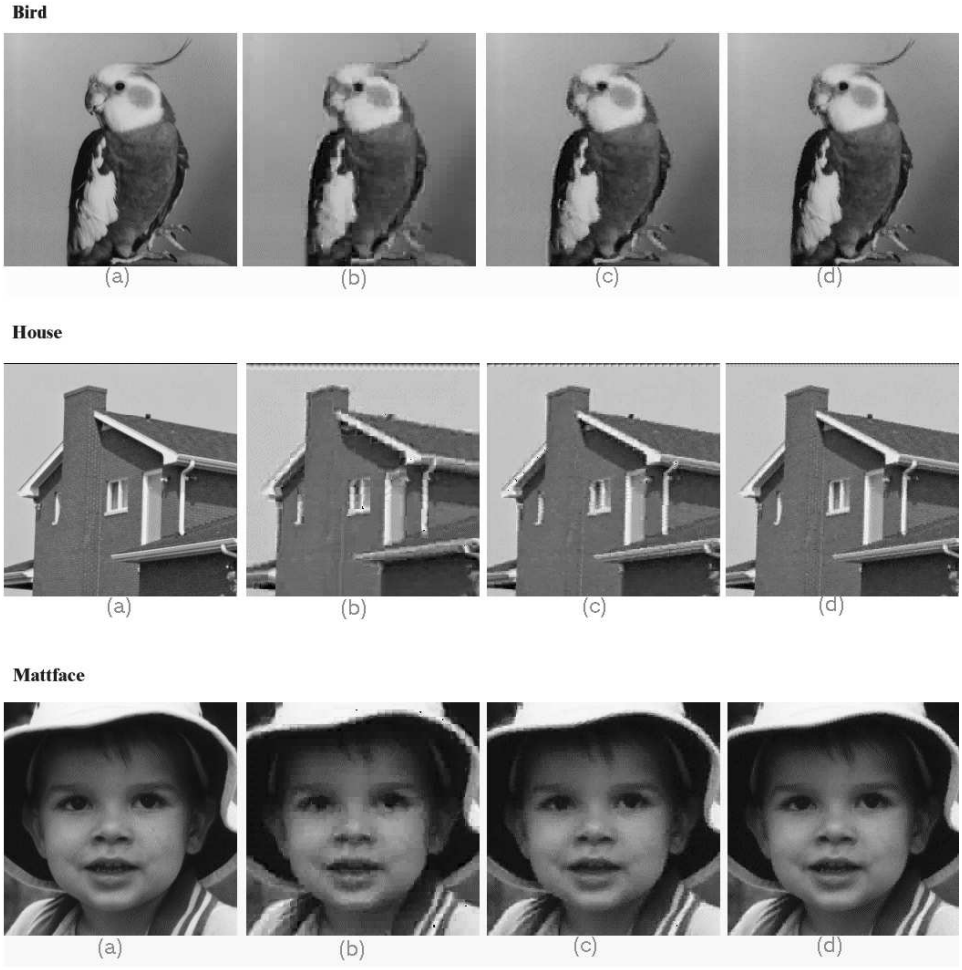

Figure 7.11: Results on 256×256 images. (a) Original images, (b) reconstructed images for VQ with 8×8 blocks, (c) reconstructed images for VQ with 4×8 blocks and (d)reconstructed images for VQ with 4×4 blocks.

and significantly superior for the Lena image. Karayiannis and Pai [148] developed several variants of a fuzzy vector quantization algorithm. They also reported results for VQs using traditional k-means algorithm and fuzzy k-means algorithm. They used the 256×256 Lena image for both training and testing. In their study they used image blocks of size 4×4 and considered codebooks of different sizes. For a codebook of size 256 they reported the PSNRs 27.06 dB using k-means algorithm, 29.91 dB using fuzzy k-means algorithm, 29.60 dB, 29.93 dB and 29.95 dB for three different variants of fuzzy vector quantization algorithms. These results are similar to the results reported here for the Lena image used in the training set. In [9] SOM algorithm is used for designing a VQ. Here SOM is trained with the low order coefficients of discrete cosine transform (DCT) of 4×4 image blocks. The output of the encoder is further compressed using a differential encoding scheme. The result reported for the Lena image shows a PSNR of 24.7 dB with a compression rate 25.22 (i.e., 0.32 bpp approximately). In a recent work Schnaider et al. [302] studied wavelet based lattice vector quantization methods. For the same image they reported a PSNR 32.06 dB at a compression ratio 24:1 (i.e., 0.33 bpp approximately). All other images studied here also show high PSNR with good compression rates.

7.4.1 Quantitative assessment of psychovisual quality preservation

Our goal is to devise a simple scheme of designing VQ that can compress images with good perceptual fidelity. It is a well established fact that the mean squared error (MSE) based distortion measures such as PSNR are not very good for measuring perceptual fidelity. However, there is no universally accepted quantitative measure for psychovisual quality. We presented a surface fitting method for quantization that smoothens the reconstructed image resulting in reduction of blocking effect. Sometimes, it may introduce some blurring of sharp features. However, moderate blurring is not considered annoying by a human observer since it is a "natural" type of distortion [288]. The effectiveness of the proposed scheme for preserving psychovisual quality in the reconstructed images has been demonstrated visually in Figs. 7.8-7.11. The increased ability of reducing the blocky effect by surface fitting scheme is also demonstrated visually in Figure 7.4 and Figure 7.5. Figure 7.5 depicts enlarged views of some portion of the images shown in Figure 7.4. The portion is selected in such a way that it contains fine detail as well as smooth non-linear intensity variation.

Now we define two quantitative indexes that can assess the preservation of psychovisual quality with respect to blocking effect. The development is based on the following observations made by Ramstad et al. [288].

0	-1	0
-1	4	-1
0	-1	0

Figure 7.12: Convolution mask corresponding to the Laplacian operator.

1. The blocking effect is a natural consequence of splitting the image into blocks and independent processing of each block. The quantization errors will lead to appearance of the blocks that the image is split into.

2. Blocking effects are visible in smooth image areas, whereas in complex areas, such as textures, any underlying blocking effect is effectively masked.

Thus the degradation of psychovisual quality is contributed by (1) the reduction of smoothness due to the difference between an image block and the quantized block that replaces it and (2) the additional discontinuity imposed across the block boundary due to quantization. So we develop a pair of quantitative indexes. The first one measures the loss of smoothness per pixel across the block boundary due to vector quantization. We call it *boundary smoothness mismatch index* (BSMI). The second index deals with difference of smoothness per pixel between the original image and the reconstructed image for the non-boundary pixels (i.e., all pixels in a block that are not on the boundary of the block). We call it *inner smoothness difference index* (ISDI). Evidently for both indexes lower values imply better preservation of psychovisual quality.

The development of these indexes are based on the fact that the second derivative, i.e., *Laplacian* of a surface at a point can be used as a measure of the lack of smoothness at that point. This fact is often used for detection of edges in an image [113], where the pixels showing abrupt variation of intensity with respect to their neighbors are detected. In our approach we use the Laplacian to measure the lack of smoothness in intensity variation. The discrete realization of the operator in form of a convolution mask is shown in Figure 7.12. Henceforth we shall denote this mask as $\mathcal{L}(i,j)$, where (i,j) denotes the coordinate of the pixel on which the mask is applied.

Now we present the formulae for computing the indexes. Let P_B denote the set of pixels at the block boundaries and P_I be the set of non-boundary pixels in an image. Then the *boundary smoothness mismatch index* (BSMI) of an image is defined as

$$BSMI = \frac{\sum_{(i,j)\in P_B}[x_{ij} * \mathcal{L}(i,j)]^2}{Number\ of\ pixels\ in\ P_B}. \tag{7.10}$$

The *inner smoothness difference index* (ISDI) is computed for a reconstructed image with respect to the original image and is defined as

Table 7.3: Comparison of performances regarding preservation of psychovisual fidelity between the vector quantizers using SOM code books and surface fitting code books.

Image	Code book used	Blocksize 8×8		Blocksize 4×8		Blocksize 4×4	
		BSMI	ISDI	BSMI	ISDI	BSMI	ISDI
Lena	SOM	527.95	494.35	467.90	398.79	433.22	337.73
(512×512)	Surface fit	488.36	385.90	415.84	375.38	405.29	330.05
Barbara	SOM	666.77	2605.70	979.52	2285.50	1788.10	1562.50
(512×512)	Surface fit	570.73	2556.60	703.03	2535.60	1151.50	2177.90
Boat	SOM	737.51	711.16	761.78	607.40	700.26	487.01
(512×512)	Surface fit	704.76	623.99	636.32	597.27	660.99	491.98
Bird	SOM	429.23	209.12	312.93	185.65	270.50	150.17
(256×256)	Surface fit	388.41	168.04	298.22	163.57	257.19	141.68
House	SOM	758.57	872.20	681.49	932.19	599.03	430.73
(256×256)	Surface fit	756.92	647.84	658.20	978.57	562.31	592.54
Mattface	SOM	540.77	209.37	411.19	190.10	312.87	161.88
(256×256)	Surface fit	512.84	133.09	365.63	136.58	251.44	110.99

$$ISDI = \frac{\sum_{(i,j)\in P_I}[x_{ij} * \mathcal{L}(i,j) - \hat{x}_{ij} * \mathcal{L}(i,j)]^2}{Number\ of\ pixels\ in\ P_I}, \tag{7.11}$$

where x_{ij} and $\hat{x}_{ij}$ denote the intensities of the (i,j)-th pixel in the original image and the reconstructed image respectively. Note that, for both measures, lower the value, the better is the performance. We report the results of our study using the proposed indexes in Table 7.3.

As shown in Table 7.3 for all eighteen cases the surface fitting codebooks show better performances in terms of BSMI. This clearly indicates that for the surface fitting codebooks the block boundaries maintain better continuity. Table 7.3 also reveals that for thirteen (out of eighteen) cases the ISDI values for the surface fitting codebook are smaller than the corresponding ISDI values using SOM codebook. This means that in these thirteen cases the similarity of the blocks reconstructed by the surface fitting codebook is more to the original image than the similarity of SOM VQ reconstructed images with original ones. The remaining five cases involve three images with small block sizes (Barbara and House images with block sizes 4×8 and 4×4 and Boat image with block size 4×4). These results can be attributed to the fact that each of these images (original) contains substantial portions covered with complex texture-like areas and for smaller block sizes the gain due to surface fitting over the SOM is not reflected in ISDI values. Overall, Table 7.3 indicates that the replacement of the codevectors obtained

directly from SOM with the codevectors obtained by least square-error surface fit results in VQs preserving the psychovisual fidelity to a better extent.

7.5 Conclusion

We presented a comprehensive scheme for designing vector quantizers for image compression using generic codebooks that produce reconstructed images with good psychovisual quality. The scheme exploits the special features of SOM for codebook generation and introduces a novel surface fitting scheme for refinement of the codevectors generated by SOM algorithm to reduce the blockyness in the reconstructed images. To achieve better compression, it also puts together some well known concepts such as entropy coding of indexes and difference coded mean values. The proposed scheme, as a whole, achieves compression at low bit rates with good quality reconstructed images.

Due to the density matching and topology preservation properties of SOM, it can be used to generate a good set of code vectors. Use of mean removed vectors reduce the reconstruction error significantly but it necessitates doubling of the amount of the data to be stored or transmitted. However, lower reconstruction error allows us to use larger image blocks with acceptable fidelity. The use of cubic surface fitting for refinement of the codevectors enables us to decrease unpleasant blocking effect that appears in spatial VQs at low bit rates. The improvement due to surface fitting is demonstrated visually as well as quantitatively using two indexes proposed in this chapter. The computational overload due to surface fitting as proposed here is restricted to the codebook generation stage only, unlike the transform or subband coding techniques where every image has to be transformed into frequency domain at the encoder side and inverse transformed into spatial domain at the decoder side.

The use of generic codebook not only enables us to construct the codebook only once, but the knowledge of distribution of indexes for the training images can also be exploited as the *a priori* knowledge of distribution of indexes for the test images. This is used for devising an entropy coding scheme for the index values. Further, Huffman coding of the average values is done through the difference coding of the means of the image blocks.

We have reported results with three VQs using block sizes 8×8 (VQ1), 4×8 (VQ2) and 4×4 (VQ3). Among them, as expected, VQ1 gives the highest compression rate but PSNR is comparatively low. On the other hand, VQ3 produces excellent quality of the reconstructed images with the lowest compression rate. VQ2 paves a middle path by achieving nice reconstruction fidelity at a good compression rate. We have compared our results with standard JPEG algorithm. VQ1 is found to be superior to JPEG at comparable bit rates both in terms of PSNR and psychovisual quality.

VQ2 and VQ3 scored less than the corresponding JPEG images in terms of PSNR, but produced comparable psychovisual quality. We have compared our method with some published work and found that our results are superior or comparable to other spatial VQs. Further, we compared our results with two recently published work using DCT and wavelets respectively and our results for Lena image are comparable to them in terms of PSNR.

We proposed two indexes for quantitative assessment of "blockyness" introduced in the reconstructed images by the VQ process. The first index, BSMI, measures the lack of continuity/smoothness at the block boundaries in the reconstructed images. The other index, ISDI, measures the deviation of the reconstructed image from the original image in terms of smoothness property of the non-boundary pixels. We have compared the images reconstructed using the codevectors generated directly from SOM and those using the codevectors obtained by surface fitting method. We found that the surface fitting codebooks produce images with better psychovisual quality with respect to the blockyness.

Chapter 8

Fast Codebook Searching in a SOM-based Vector Quantizer [1]

[1] Content of this chapter is submitted to [200].

8.1 Introduction

In the previous chapter we have presented a method for generation of codevectors that can realize high compression ratio along with good psychovisual quality of reconstructed images. Apart from finding good codevectors the VQ performance depends crucially on the size of the codebook and the dimension of the vectors. Increasing the codebook size, the input space can be represented to a finer degree. The VQ exploits the inter-block statistical redundancies. A larger block enables the quantizer to exploit the statistical redundancy existing in the data to a greater degree [110]. However, using large number of codevectors and high dimensional vectors introduces a serious performance bottleneck on the part of the encoder. Given any vector to be encoded, the encoder has to search the codebook for the best matching codevector. Thus, to make full search of a large codebook with high dimensional vectors, the encoder incurs a high computational overhead. To circumvent this difficulty various techniques have been developed [110]. These techniques apply various constraints on the structure of the codebook and use suitably altered encoding algorithm. These techniques can be divided into two major categories. The *memoryless* VQs perform encoding/decoding of each vector independently. Well known examples in this category include tree-Structured VQ (TSVQ), classified VQ, shape-gain VQ, multistage VQ, hierarchical VQ etc [110, 114]. Methods in the other category depend on the past information (thus having memory) for their operations. Prominent members include predictive VQ, recursive or feedback VQ, finite state VQ (FSVQ) etc [110, 114]. Comprehensive reviews of these methods can be found in [5]. All these techniques are aimed at reducing the codebook search complexity without significant loss in reproduction quality. Among some of the recent works, in [208] Lai and Liaw developed a fast codebook searching algorithm based on projection and triangular inequality that use the characteristics of the vectors to reject impossible codes. In [161] a method for designing predictive vector quantizer using deterministic annealing is proposed. An approach to design an optimal FSVQ is proposed in [69]. A novel method of designing TSVQ can be found in [41].

8.2 Codebook searching in a SOM

In the previous chapter we discussed the merits of SOM-based VQs in general. Here we explore the advantage of using SOM-based VQs from the perspective of fast codebook searching. If SOM is used to generate the VQ codebook, the training algorithm implicitly introduces some constraints on the structure of the codebook through its "neighborhood update" feature. This results in the topology preservation property and density matching property of SOM. In a SOM with 2-dimensional lattice, the weight vectors corresponding

to the network nodes are used as the codevectors. Thus, a code index can be expressed by a pair of (x, y) coordinates of the corresponding node on the lattice. Due to topology preservation property of SOM, similar vectors are mapped onto same or nearby nodes on the lattice. In an image adjuscent blocks often bear close similarity. Thus, for two adjuscent blocks, if the lattice coordinate of the codeword of one is known, the codeword for the other block is more likely to be found among the nearby nodes of the previous codeword on the lattice. Thus, a limited search in the codebook can find a good match for the second block. There is another possibility of performing fast codebook search with SOM. One can train a smaller SOM with the codewords in the larger SOM-based main codebook. Then the codewords of the main codebook are partitioned according to their proximity to the nodes of the smaller SOM. Now the codebook search for a vector can be performed as a two-step process, finding the best matching node in the smaller SOM and then selecting the best matching codeword from the subset corresponding to that node.

In the current chapter we exploit the above possibilities to develop two strategies for fast codebook search and finally combine them to propose a fast codebook search method with several desirable features. Though in this work we design spatial vector quantizer (SVQ) for image data to demonstrate the power of the proposed methods, they are applicable to vector quantization of any kind of signals where two successive inputs have some correlation. These methods also offer a scope for efficient Huffman coding of the index values.

Like previous chapter, here also we use mean-removed vectors for vector quantization. For preparing the image vectors we follow the procedure described in Section 7.2.1. Thus, here also we need the indexes as well as the block averages to reconstruct the quantized image. We compress indexes and block averages further using Huffman coding. For efficient Huffman coding of the block averages we use the scheme proposed in Section 7.3.1. Now in the following we shall describe the SOM-based fast codebook searching method and the scheme for Huffman coding the indexes generated by the proposed codebook search method.

8.3 Searching the codebook

First we design and analyze two strategies for searching the codebook in a restricted manner to reduce the search complexity. The first one uses the basic SOM generated codebook and exploits its topology preservation property. The second one uses a smaller SOM along with the basic codebook. The smaller SOM is trained with the codevectors of the basic codebook, we call it Level2-SOM (L2-SOM). Then we propose a combined

method that uses the L2-SOM and the basic codebook and utilizes the topology preservation property of the basic codebook, thus exploiting the best features of strategies 1 and 2. It should be noted that the strategies 1 and 2 are also independent methods for searching the codebook. Here we call them strategies simply to differentiate them from the final method that combine both of them. The methods are described bellow.

8.3.1 Strategy 1: Restricted window search over SOM lattice

This strategy is fashioned after Finite State Vector Quantizers (FSVQ)[110]. FSVQs belong to a more general class of VQs known as *recursive* or *feedback* vector quantizers. Given an input sequence of vectors $\{\mathbf{x}_n\}$, $n = 1, 2, \cdots$, the encoder produces both a set of code indexes $\{u_n\}$, $n = 1, 2, \cdots$, and a sequence of *states* $\{s_n\}$, $n = 1, 2, \cdots$, which determines the behavior of the encoder. A FSVQ is characterized by a finite set of K states $\mathcal{S} = \{1, 2, \cdots, K\}$ and state transition function $f(u, s)$. Given the current state s and last code index u, when presented with the next vector $\mathbf{x}$ of the input sequence, the FSVQ enters a new state determined by the state transition function $f(u, s)$. Associated with each state s is a smaller codebook $\mathcal{C}_s$, known as state codebook for state s. The search for the codevector for the new vector $\mathbf{x}$ is now restricted to $\mathcal{C}_s$, which is much smaller than the full codebook, also known as super codebook, $\mathcal{C} = \bigcup_{s \in \mathcal{S}} \mathcal{C}_s$. The major challenge in designing a FSVQ involves the design of the state transition function and finding the codebooks corresponding to each state.

One of the simplest and popular techniques for finding a next-state function is called *conditional histogram design* [110]. This approach is based on the observation that each codeword is followed almost invariably by one of a very small subset of the available codevectors. This happens due to the existence of high degree of correlation between successive vectors in an input sequence. Thus, the performance of the VQ should not degrade much if these small subsets form the state codebooks. So, the training sequence is used to determine the conditional probability $p(j \mid i)$ of generating the j-the codevector following the generation of i-th codevector. The state codebook for state i of size N_i consists of the N_i codevectors $\{\mathbf{y}_j\}$ with highest values of $p(j \mid i)$. However, with this design, especially for data outside the training sequence, the optimal codevector may lie outside the state codebook. Thus often a threshold on the degree of matching is used. If no codevector with match exceeding the threshold is found in the state codebook, the system is said to 'derail' [69]. In such a situation usually an exhaustive search over the super codebook is performed for re-initializing the search process.

While using SOM for generation of the codebook, one can associate with each of the codevectors a tuple describing the position of the corresponding node in the SOM lattice. Further, due to topology preservation property of SOM, the nodes nearby on the lattice

plane encode vectors with higher similarity than the nodes which are located far away on the lattice. Since the input vectors corresponding to two adjacent blocks in the image are likely to be correlated (i.e., expected to have high similarity), the codevectors associated with them are also likely to be associated with nodes those are close to each other on the lattice. Thus, if the codevector for an input vector corresponds to a node c on the lattice, then the codevector for an adjacent block is likely to be within a small neighborhood N_c of c. This property can be exploited to reduce codebook search time.

We define a search window size $s_h \times s_w$ and a quality threshold T. The image vectors to be quantized are prepared as described in Section 7.2.1. For the first vector an exhaustive search is performed and the index in the form of (x_1, y_1) pair is generated. For the subsequent k-th vector the search is performed among the nodes on the SOM lattice falling within the search window centered at the node (x_{k-1}, y_{k-1}) (we call it the previous index). Due to topology preservation property of SOM and the characteristic similarity between neighboring image blocks there is a high possibility of finding a good match within the window. If the best match within the window exceeds the threshold T, then the index (x_k, y_k) of the corresponding node is accepted; otherwise (i.e., in case of derailment), rest of the codebook is exhaustively searched to find the best match. Now (x_k, y_k) becomes the previous index for $k+1$-th vector. Thus, we can identify the full codebook generated by the SOM as the super codebook of FSVQ, the current states as the predecessor index (x_{k-1}, y_{k-1}), and the state codebook as the set of codevectors within the SOM lattice window of size $s_h \times s_w$ centered at (x_{k-1}, y_{k-1}).

There are two issues to be taken care of. The first one concerns the case when the previous index is close to the boundary of the SOM lattice and the window cannot be centered at it. In such a case, we align the edge of the window with the edge of the lattice. The second issue relates to the image blocks at the beginning of a row of blocks, i.e., the blocks at the left edge of the image. Since the image blocks are encoded in a row-major fashion, for other blocks the previous index corresponds to the index for the immediate left block. For a leftmost block we use the index of a leftmost block in the previous row (i.e., the block at the top of current block) as the previous index.

There is also a design issue regarding the choice of the window size as well as the quality threshold T. For a given threshold T, smaller window sizes will reduce the window search time, but the instances of derailment will be higher. Again, for a fixed window size, the higher the threshold, the more is the instances of derailment. It is difficult to predict theoretically a suitable choice of these parameters because optimal choices of these parameters are likely to depend on the distribution of the codebook vectors over the SOM lattice as well as the distribution of the image vectors being quantized. In the 'Experimental results' section we have presented some empirical studies regarding the

choice of these parameters.

Apart from facilitating fast codebook searching, the SOM based method has another advantage. Since the indexes are expressed in terms of 2-tuple values of lattice position and majority of the code indexes are found within a small neighborhood of the previous index, we can express the indexes as 2-tuple of offset values from the previous index. In such a scheme the values in majority of the index tuples are likely to be much smaller than tuples with absolute values. As we shall see later, this feature can be exploited for efficient Huffman coding of the indexes.

8.3.2 Strategy 2: Codebook searching with L2-SOM

Though the above scheme results in a significant speed-up in the codebook searching procedure, for the cases when a suitable match is not found within the search window, full codebook search is employed. To ensure the reproduction quality, we usually set the quality threshold T reasonably high. In turn, this may lead to significant number of cases for which full codebook search is conducted. To avoid the exhaustive searches we develop a two level scheme for codebook searching. The scheme follows the design principle of classified VQs (CVQ) [110], where the super codebook is partitioned into smaller codebooks using some heuristic. In CVQ, quantization of a vector involves first selecting one of the smaller codebooks and then finding the best match within it. Here we train a smaller SOM, known as Level-2 SOM (L2-SOM) with the codevectors of the basic SOM. This allows us to partition the codevectors into groups according to their proximity to the weight vectors of the L2-SOM. Thus, for a vector to be coded, the encoder first looks up to the L2-SOM to find the best matching node. Then the best matching codevector is found from the group of basic codevectors associated with the best matching node. This can also be viewed as a hierarchical SOM.

Under this scheme, we do not use any search window or any threshold for finding the next codevector. Though the scheme results in huge benefit in terms of search time, the index values (the members of the tuples) have the tendency of spreading over a much larger range. This affects adversely the additional compression achieved by Huffman coding of the index values.

8.3.3 Combined Method: Restricted window search with L2-SOM look-up for re-initialization

The proposed method combines the features of strategies 1 and 2 to achieve better results. Here, like strategy 1, to find the code index corresponding to the current vector a

window centered at the node corresponding to the previous index is searched. However, the computation bottleneck in strategy 1 due to exhaustive searches for initialization (1st vector) and re-initialization (when we fail to find a good match within the window according to the threshold condition) is removed using the L2-SOM. In other words, whenever an exhaustive search is required under first strategy, the best matching L2-SOM node is found and the group of basic codevectors corresponding to the node is searched for the best match.

It will be seen in the experimental results that this method will increase the search efficiency, as well as tend to restrict the components of index offset values to smaller ones leading to more efficient Huffman coding of the indexes. Thus, this method incorporates faster codebook searching as well as good compression ratio without significant loss in reproduction quality.

8.4 Transformation and Huffman coding of the indexes

For a codebook generated with an $m \times n$ SOM the range of the components of indexes are 0 to $m-1$ and 0 to $n-1$ respectively. However, due to the topology preservation property, even without restricting the search, neighboring image blocks are mapped into nearby nodes on the SOM lattice. So instead of using the absolute values of the coordinates, if we express the index of a vector in terms of offsets from the previous index, i.e., $(x_k^o, y_k^o) = (x_k - x_{k-1}, y_k - y_{k-1})$ then x_k^o and y_k^o are more likely to have small values. Figure 8.1 depicts the histograms of the index components expressed as offset values for 512×512 Lena image for a VQ using 32×32 SOM with 8×8 blocks (i.e., 64 dimensional vectors) employing exhaustive search. For a restricted search the distribution is expected to have even sharper peak around 0.

Clearly, coding of indexes in terms of offset values will allow us to perform efficient Huffman coding. However, using offset values stretches the range of index component values from $-(m-1)$ to $(m-1)$ and from $-(n-1)$ to $(n-1)$ respectively. Hence we need more code words. We can restrict the range of component values of the offset (x_k^o, y_k^o) values within 0 to $(m-1)$ or $(n-1)$ whichever is greater, if the index values are further transformed into $(x_k^{o'}, y_k^{o'})$ as follows:

- If $x_k^o \geq 0$ then $x_k^{o'} = x_k^o$
- Otherwise $x_k^{o'} = x_k^o + m$
- If $y_k^o \geq 0$ then $y_k^{o'} = y_k^o$

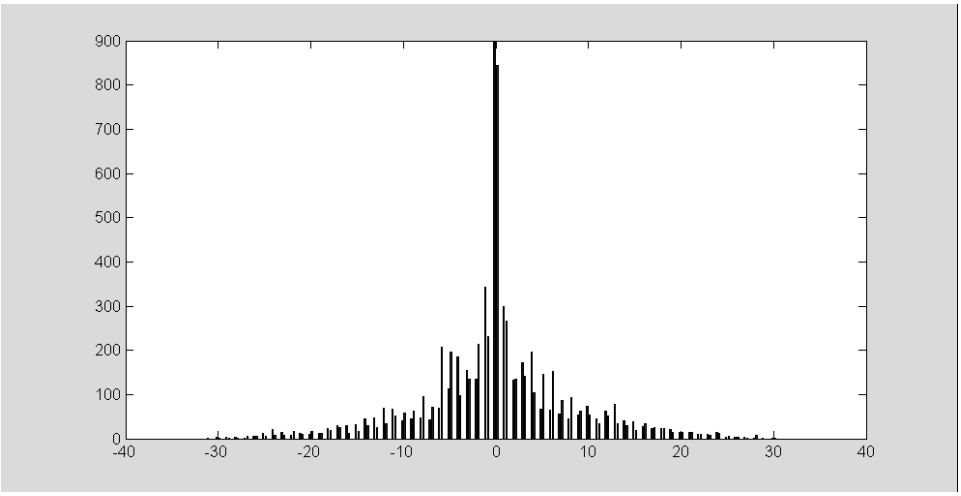

Figure 8.1: Grouped histogram of the index offsets for exhaustive search vector quantization of the Lena image.

- Otherwise $y_k^{o'} = y_k^o + n$

Figure 8.2 depicts the histogram corresponding to the transformed offsets for the Lena image (corresponding to the index offsets shown in Figure 8.1).

The decoder computes the index values (x_k, y_k) from $(x_k^{o'}, y_k^{o'})$ as follows:

- If $x_k^{o'} \leq (m - x_{k-1})$ then $x_k = x_{k-1} + x_k^{o'}$
- Otherwise, $x_k = x_{k-1} + (x_k^{o'} - m)$
- If $y_k^{o'} \leq (n - y_{k-1})$ then $y_k = y_{k-1} + y_k^{o'}$
- Otherwise, $y_k = y_{k-1} + (y_k^{o'} - n)$

8.5 Experimental results

In our experiments we have trained a 32×32 SOM with training vectors generated from a composite of sixteen 256 level images each of size 256×256. The training image is shown in Figure 8.3. The block size used is 8×8. Thus the vectors are in 64 dimension and the basic codebook size is 1024. Here we report the test results with three 512×512 images **Lena**, **Barbara** and **Boat**. Note that, though the Lena and Barbara images are used in training, the test images are not the same as the training images. In the training set 256×256 images are used. The experimental setup for the results reported here are as follows:

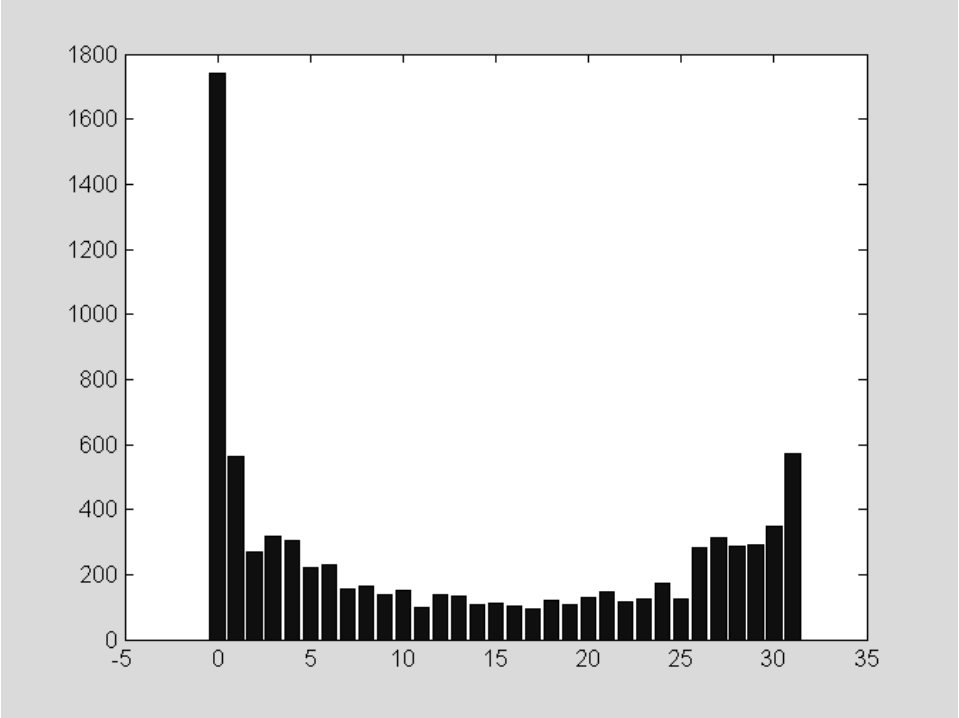

Figure 8.2: Histogram of the transformed offset values for exhaustive search vector quantization of the Lena image.

> The search window is set to 8×8 and the quality threshold T is set at 30 dB PSNR (per block). A 6×6 Level-2 SOM is trained with the 1024 codevectors in the basic codebook. The experimental results for Strategy 1, Strategy 2 and the Combined method are summarized in Table 8.1. We note here that each of Strategy 1 and Strategy 2 are fully implementable restricted search methods on its own.

In Table 8.1 the search complexity is expressed in terms of the number of codevectors examined during the search procedure. For the exhaustive search, for 4096 blocks in a test image the number of codevectors examined is 4096*1024=4194304. For the restricted search methods the complexity is expressed as percentage of the number of codevectors examined with respect to the exhaustive search (shown in parenthesis). It is evident from the results that compared to the exhaustive search all three proposed methods substantially decrease the search time without any significant sacrifice in the reproduction quality. The Strategy 1 nearly halves the search complexities with negligible decrease in PSNR values for all the images. The Strategy 2 reduces the search complexity for all the images to 15%-16%, i.e., 1/6-th with drop of PSNR 0.37, 0.26 and 0.34 dB for Lena, Barbara and Boat images respectively, compared to the exhaustive search. The proposed Combined method reduces the search complexity to about 11% with a decrease of PSNR 0.48 dB for Lena. For Barbara the search complexity to about 14% with a drop in PSNR

Figure 8.3: The training images.

Table 8.1: Comparison of VQ with exhaustive search and restricted searches (Strategies 1,2 and the combined method).

Image	Search method	PSNR (dB)	No. of Distance calculations (% w.r.t. exhaustive search)	Compression ratio(bpp)
Lena	Exhaustive	28.95	4194304	0.227
	Strategy 1	28.82	1993984 (47.5%)	0.218
	Strategy 2	28.58	687393 (16.4%)	0.226
	Combined	**28.47**	**472071 (11.3%)**	**0.218**
Barbara	Exhaustive	24.37	4194304	0.231
	Strategy 1	24.34	2710144 (64.6%)	0.227
	Strategy 2	24.11	654899 (15.6%)	0.232
	Combined	**24.09**	**604710 (14.4%)**	**0.228**
Boat	Exhaustive	26.97	4194304	0.207
	Strategy 1	26.93	2348224 (56.0%)	0.203
	Strategy 2	26.63	656693 (15.6%)	0.207
	Combined	**26.61**	**512804 (12.2%)**	**0.203**

of 0.28 dB, while for Boat the search overhead to 12% with loss of PSNR by 0.36 dB. Therefore, the Combined method results in a significant decrease in search complexity with a very little sacrifice in quality.

The compression ratios reported here are the final value with Huffman coding of the transformed index offsets and the difference coded block averages. As can be seen from the results, Strategy 1 and Combined method produce almost the same compression ratio for all images while the exhaustive search and Strategy 2 results show marked similarity in compression ratio. This is the consequence of the restriction imposed on Strategy 1 and Combined method through the use of the search window. Figure 8.4 shows the reproduced Lena image quantized using standard VQ and Strategy 1, Strategy 2 and Combined method in the panels (a) to (d) respectively. Visual inspection reveals almost no difference of quality among them. Figure 8.5 shows the histograms of offset values for the methods studied.

The Figures 8.5(a) and 8.5(c) show marked similarity while 8.5(b) and 8.5(d) are almost identical with more concentration of the offset values at and around zero. This explains the higher compression rates achieved by the Strategy 1 and Combined method due to more efficient Huffman coding of the indexes.

Now we compare the overall performance of the three schemes based on the three factors, namely *reproduction quality*, *search complexity* and *compression ratio*. Strategy

Figure 8.4: The reproduced images for . (a) Exhaustive search, (b) SOM window search (Strategy 1), (c) Level 2 SOM search (Strategy 2) and (d) Proposed Combined search method for Lena image.

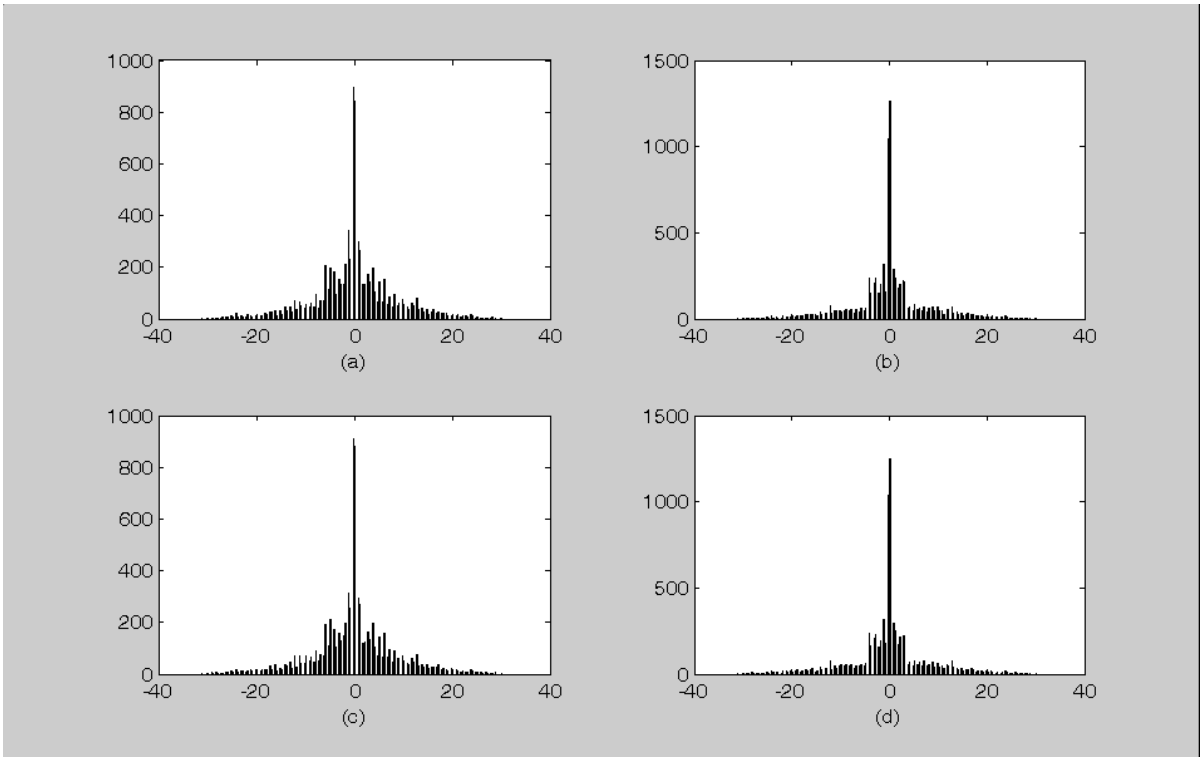

Figure 8.5: The histogram of offset values . (a) Exhaustive search, (b) SOM window search (Strategy 1), (c) Level 2 SOM search (Strategy 2) and (d) Proposed Combined search method for Lena image

1 has the best reproduction quality (see Figure 8.4). Its compression ratio is either lowest (for Barbara) or the same as that of the combined method. However, it has a much higher search complexity compared to the Strategy 2 and Combined method. Strategy 2 reduces the search complexity to a great extent with a slight decrease in reproduction quality, but it has compression ratio higher than Strategy 1. The Combined method reduces the search complexity further with a negligible decrease in reproduction quality, but achieves low compression ratio similar to Strategy 1. Thus considering the three factors together the Combined method outscores both Strategy 1 and Strategy 2 when applied separately.

As mentioned earlier, for the design of the encoder, the choice of the *search window size* $s_h \times s_w$ and the choice of the*quality threshold* T play an important role. We have conducted an empirical study by designing the VQ with the Combined method with various choices of the search window sizes and quality thresholds, and collected the statistics for quantizing the Lena image. In Figure 8.6 the variation in number of distance calculations for different window sizes and quality thresholds are depicted. It can be observed that for both increase of window size as well as threshold value, the number of distance computations increases. However, the variation is much less with respect to threshold value compared to the variation with respect to window size. This clearly indicates the strong possibility of finding a good match within a small window. In Figure

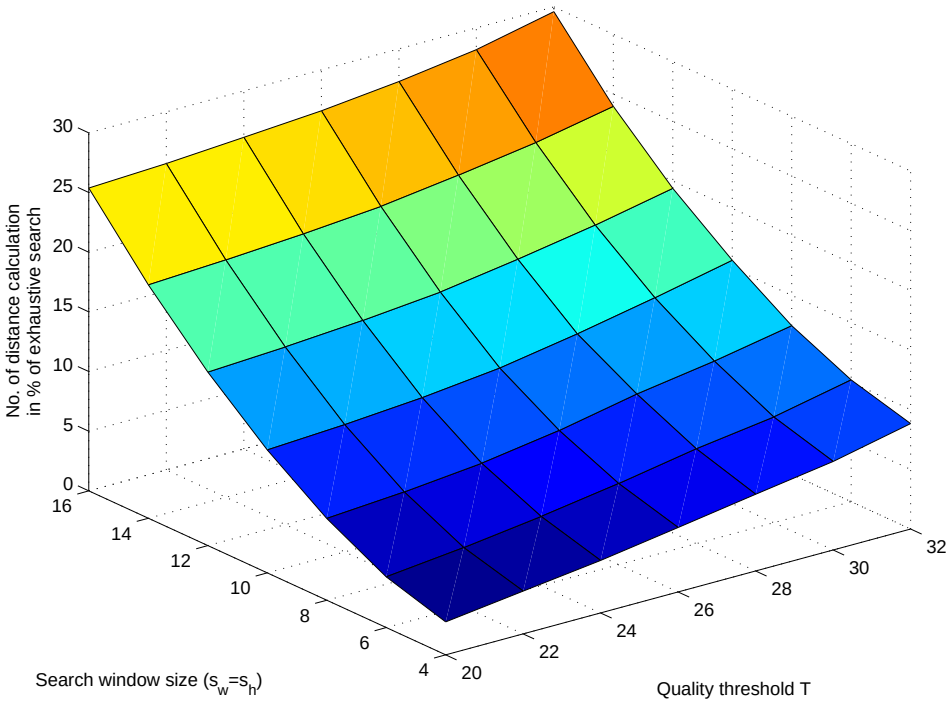

Figure 8.6: Variation of number of distance calculation with search window size and quality threshold for quantizing Lena image.

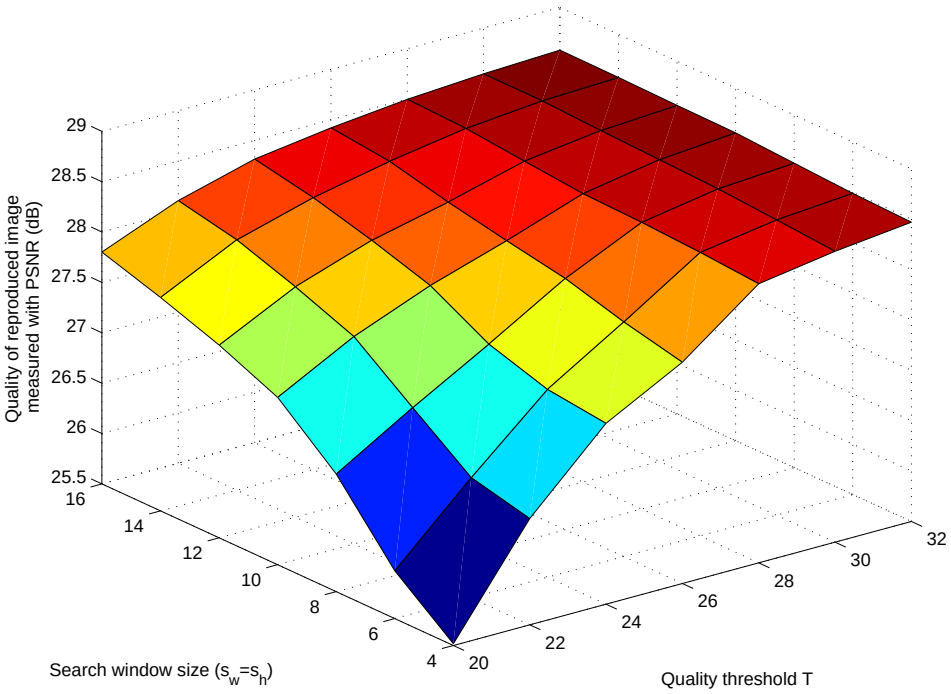

Figure 8.7: Variation of reproduction quality (Measured in PSNR) with search window size and quality threshold for quantizing Lena image.

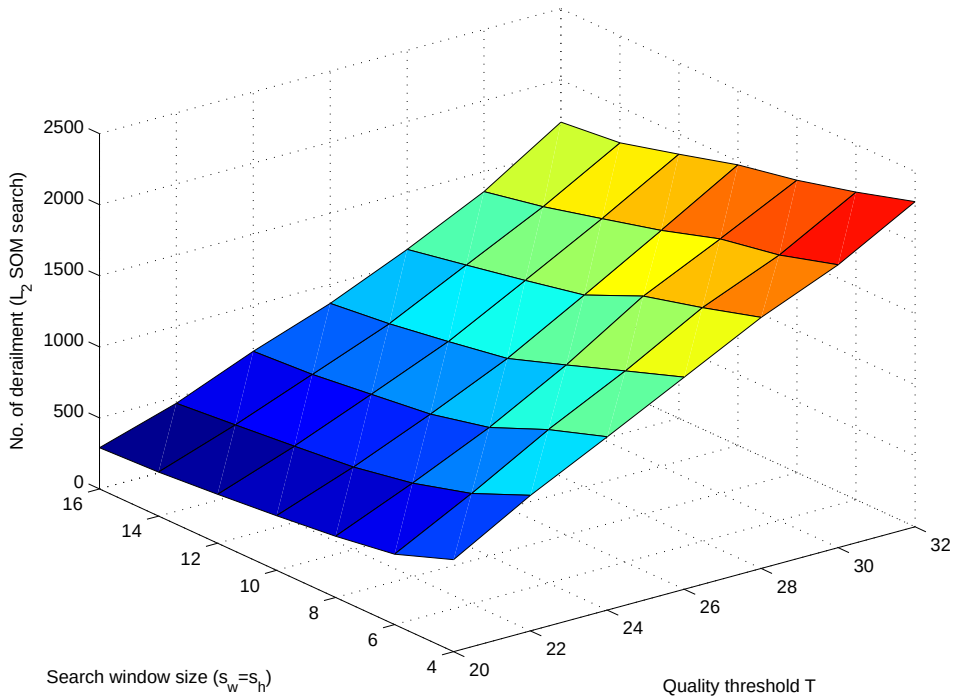

Figure 8.8: Variation of number of level2 SOM searches (i.e, derailments) with search window size and quality threshold for quantizing Lena image

8.7 the variation of the reproduction quality is presented. Here it is evident that the quality threshold has more influence on the reproduction quality than the window size. Figure 8.8 depicts the frequency of Level 2 SOM searches caused by the derailments. This also indicates a greater influence of the quality threshold than the window size.

8.6 Conclusion

Self-organizing Map is used by several researchers for designing the codebook of a vector quantizer. However, these methods are restricted mainly to the use of SOM as a clustering algorithm. In this work, apart from utilizing the clustering property we exploited the other notable properties of SOM, topology preservation and density matching, to formulate encoding method with reduced search complexity. First, we designed two separate strategies using SOM for fast codebook search. The first strategy used the main SOM generated codebook and exploited the topology preservation property by restricting the codebook search to a small window. This strategy is designed in line with the finite state VQs, without explicit calculation of state codebooks. However, exhaustive search of the codebook is performed for re-initializations of the encoder. The second strategy uses along with the basic SOM another smaller SOM (L2-SOM) trained with the weight vectors of the basic SOM and produces a partitioning of the codevectors. This strategy does

not exploit the topology preservation property but partitions the codebook into smaller sub-codebooks. The procedures developed here eliminated the need for an exhaustive search of codebooks altogether. Finally, we proposed a method that combined the best features of both strategies, i.e, the L2-SOM as well as restricted search within a window, to deliver a good overall performance. The use of SOM and restricted search combined with suitable transformations of index values and block averages enabled us to apply Huffman encoding to enhance the compression ratio without compromising reproduction quality.

Choices of two design parameters, "the search window size" and the "quality threshold" influence the computational load of the encoder significantly. However, our empirical study shows that the rate of increase in computational load with increase of the quality threshold for a fixed window size is greater than that when the threshold is kept constant and the window size is increased. This indicates that if a match satisfying certain threshold is to be found within the search window, more often than not it is found within a small neighborhood of the previous index. This finding also indicates the suitability of SOM-based codebook search methods proposed in this work.

Though we have designed the spatial vector quantizer for image compression, the schemes are general in nature and can be used for other data types such as audio where successive inputs are likely to be correlated.

Chapter 9

Conclusion and Future Works

9.1 Conclusion

In this thesis we have presented some Self-organizing Map (SOM)-based new methods for pattern recognition tasks, namely, pattern classification and vector quantization. We have also presented results of some empirical studies on SOMs under various conditions. In Chapter 1 we have presented a brief overview of the field of pattern recognition in general, including brief descriptions of statistical, fuzzy set theoretic, evidence theoretic and neural networks based approaches to solve this problem.

An overview of SOM along with its properties is provided in Chapter 2. We also discussed different variants of the standard SOM, theoretical studies on SOM and applications of SOM in several emerging fields involving pattern recognition tasks.

In Chapter 3 we have introduced a new quantitative measure of topology preservation (T) in SOM. The index proposed here is based on rank correlation coefficient and reflects intuitive understanding of topological ordering. Comparative studies of the proposed index are made with two other measures, topographic product (P) [22] and a measure of topology violation (V) [317]. The proposed index is found to be an effective measure with a good resolution power over a range of topological ordering starting from very low value for the randomly initialized SOM to a high value for the final ordering at the end of the training. It is also found to be sensitive to degradation of topological ordering due to twisting of the map at intermediate stage of the training.

In the same chapter we have then studied the robustness of SOM in preserving topology under sparse lateral feedback interconnections among the SOM nodes. Lateral feedback interconnections in SOM are responsible for communication to neighbors of the winner node so that they can be updated. In hardware realizations of SOM, where parallel operation is achieved, these connections are needed to be implemented in one form or other. Here we defined a concept of "link density" of a node reflecting its level of connection with other nodes. We studied the effect of different levels of link density for two types of sparsity: systematic absence of connections and random absence of connections.

Systematic absence of connections: In this situation the probability of a node to have a lateral feedback connection with another node is taken as inversely proportional to their distance over the SOM lattice. Such a scenario may have a plausible biological analogue or may be introduced intentionally in economic design of a hardware realization.

Random absence of connections: In this case the absence of a lateral feedback connection between any pair of nodes is a random event with equal probability.

The empirical studies have shown that the SOM algorithm has good resilience in the first case tolerating nearly 50% absence of connections. However, in the second case the performance is found to degrade very quickly.

Other studies reported in Chapter 3 include different variants of SOM with simplified lateral feedback functions (we call them simplified SOM or SSOM) and variants of SOM with tree-structured neighborhood function (TSOMs). The SSOMs are easier to implement in hardware because they do not need explicit neighborhood boundary. They also have simpler functional forms. We studied SOMs with Gaussian, quadratic and linear (without explicit boundary) neighborhood functions. We have trained the variants of SOM with several data sets and have found them to perform equally well with respect to topology preservation.

It was observed by many researchers that though SOM is a powerful algorithm for topology preservation and vector quantization, in its standard form, due to fixed lattice structure and size its performance degrades if the data are complex and intrinsic dimension of the data is different from that of SOM lattice.We also proposed two variants of SOM with tree neighborhood functions. The first one has a MST neighborhood function but the graph over which the MST is computed is a restricted one having only the edges over a neighborhood. Thus it reduces the computation required in building MST. Though the TSOMs are not designed for topology preservation, our proposed variant preserves topology to some extent. In the MST-based SOMs the trees need to be recomputed several times (usually once in every 200-500 iterations). We argued that if prototype extraction is the sole aim, one only needs to define an unambiguous neighborhood function for implementing the neighborhood update. Thus we proposed a variant of SOM with Arbitrary Tree Neighborhood (ATN), where a tree is defined randomly over the nodes and no recomputation of the tree is performed during training. We demonstrated that with respect to prototype extraction the SOM with ATN neighborhood is as good as other TSOMs. We also investigated the capability of the TSOMs in skeletonization of shapes from images. It was found that the two SOMs with MST neighborhood perform equally well.

In Chapter 4 first we have developed a SOM-based algorithm for prototype generation. We call it "DYNAmic prototype GENeration (DYNAGEN)" algorithm. In most design schemes for prototype based classifiers usually some clustering algorithm such as k-means, is used for finding the prototypes. Typically, the number of prototypes to be found has to be supplied as a parameter. However, the number of prototypes required for the task is dependent on the data itself and is often very difficult to guess correctly. In the DYNAGEN algorithm developed here the user need not specify the number of prototypes. The algorithm is a combination of unsupervised and supervised learning.

The algorithm performs several operations, such as deleting, splitting and merging of the prototypes in a data-driven manner and produces an adequate number of labelled prototypes.

We have designed "1-Nearest Multiple Prototype (1-NMP)" classifiers for several benchmark data sets using prototypes generated by the DYNAGEN algorithm. They compare favorably with the published results for these data sets. Next we have developed a method of associating a zone of influence with each prototype and an algorithm for fine-tuning the prototypes. This results in a "1-Most Similar Prototype (1-MSP)" classifier with performances superior to 1-NMP classifiers for most of the data sets.

In Chapter 5 we build further upon our work reported in Chapter 4. In this chapter we develop a scheme for converting the prototypes generated with the DYNAGEN algorithm into a set of fuzzy rules. Then a fine-tuning algorithm is used to generate a good quality fuzzy rule base. The rule base is used for classification. Though in this chapter we reported the performance of the classifiers on the data sets used in the previous chapter, its true power is revealed through the landcover analysis task from multispectral satellite images. We have performed landcover classification from three multispectral satellite image data and the performance of the classifiers are found superior to published results on these data sets.

In this chapter we have discussed the problems of using *product* as the conjunction operator for classification tasks and then used *softmin*, a special case of a *soft-match* operator as the conjunction operator. The softmin has functional similarity to the *min* but being analytic, allows an easy computability of tuning rules. Further, we have studied the possibility of context-sensitive inferencing by tuning a parameter of soft-match operator so that each rule can use a different conjunction operator based on the nature of the data context it represents.

Though the simplest way to use a fuzzy rule base for pattern classification is to assign a pattern to the class corresponding to the rule with the highest firing strength, the approach adopted in Chapter 5, it is possible to use the fuzzy rule base to produce more information-rich output. In Chapter 6 we use the fuzzy rule base to generate output in the form of a possibilistic label vector with one component for each of the classes. The value of a component reflects the confidence of the rule base regarding the belongingness of the pattern to that particular class. In real life, it may happen that more than one component value are high and pretty close. In such a scenario it will be wise to use additional information for the final decision making, rather than simply going for the highest value. In some applications additional information can be obtained from the data itself, where each pattern is expected to have high correlation with other patterns within some identifiable context. For example, such situations can be found in image analysis,

where patterns can be associated with other patterns in a spatial context; speech signal analysis, where patterns have temporal context etc. In Chapter 6 we have developed four schemes for using context information and applied them to classify landcover types from multispectral satellite images. To classify a pixel , the proposed methods use the outputs (i.e., possibilistic labels) of the rule base for the pixel under consideration and its neighboring pixels. The first scheme (Method 1) is the fuzzy k-NN scheme for classification that finds the mean of the label vectors for nine pixels (the pixel of interest and its eight neighbors) and classify the pixel to the class corresponding to the maximum component value of the mean label vector.

Other three schemes are based on Dempster-Shafer theory of evidence for aggregation of information. The three methods differ in the ways the sources of evidence are identified, *focal elements* are chosen and how the *basic probability assignments* (BPA) to the focal elements are computed. For the first two (Method 2 and Method 3) methods, eight sources of evidence, each corresponding to the information from the pixel of interest and one of its eight neighboring pixels are identified. In the first case we use the Bayesian belief modelling. In the second case the BPAs focus on sets with one and two elements. In the third evidence theoretic approach (Method 4) we use each of the nine pixels (the pixel of interest and its eight neighbors) as a separate source of evidence. Here each body of evidence has only two focal elements, a singleton containing the class C_k corresponding to highest valued component of the label vector, with BPA $m(C_k)$ equal to the component value. The rest of the belief, $1-m(C_k)$ is assigned to the set of all classes and represents ignorant. In all these method Dempster's combination rule is applied to compute the combined evidence. The final decision is made based on *pignistic probability*. These methods are experimentally tested on two multispectral satellite images are found to perform noticeably better than the straightforward fuzzy rule based methods studied in Chapter 5. It was also observed that the efficiency of these four methods vary depending on the nature of spatial distribution of the classes. It was found that for complex spatial distribution of classes, Method 2 and Method 3 outperform Method 1 and Method 4. While in case the classes form large spatial clusters Method 1 and Method 4 perform better. We have also developed a modified version of Method 4 with a tunable parameter. It was found that after tuning of the parameter, modified Method 4 performed as well as Method 2 and Method 3 for complex spatial distribution of classes.

In Chapter 7 we focus on the problem of vector quantization using SOM for image compression. We have put forward the arguments relating suitability of SOM for designing VQ. In our work we have used mean-removed vector to accommodate large block sizes for increasing efficiency of VQ with respect to exploiting the statistical redundancy. We have studied the construction of a generic codebook also. Further, it is well known that the reconstructed images compressed with VQs show psychovisually annoying blocky

effect. Here we developed a novel "surface fitting" method using polynomial (bicubic) surfaces for smoothing the code vectors. This results in an improvement of psychovisual quality of the reconstructed image with noticeable reduction in blockyness. We have also developed two quantitative indexes for measuring the psychovisual quality with respect to blockyness.

In our work, we have used the mean-removed vectors, which necessitates the transmission/storage of the block averages along with the indexes of codewords for reproduction of the images. We have exploited the correlation among neighboring blocks to develop a difference based representation of the block averages that is suitable for Huffman coding. For the code indexes also, we have observed that there is a significant similarity of codeword distribution between the set of training images and test images. Hence, we have used the codeword distribution for the training set for assigning variable length codes for Huffman coding of the indexes. The scheme is tested with several benchmark test images and the results are compared with published results as well as standard JPEG algorithm. The results are found to be superior or comparable to other published results in terms of PSNR. Also it is found to be superior to JPEG at very low bit rates and similar in psychovisual quality at higher bit rates.

Although a VQ with large vector size and large codebook size is desirable, it demands huge computation while searching the codebook exhaustively for the best-matching codeword. To avoid the computational overhead while using large vectors and codebooks, researchers have developed several approaches to perform non-exhaustive search of codebook to find a code with good, but not necessarily best, match. These approaches usually impose some constraint on the structure of the codebook and exploit the structure in the searching method.

The topology preservation property of SOM places similar vectors in the nearby position on the SOM lattice and the density matching property places more codewords in the densely populated regions of feature space. There is also the property of images that neighboring blocks have, more often than not, high degree of similarity. Exploiting these facts, in this chapter we have developed a method for fast codebook searching in a VQ where the codebook is generated by the SOM algorithm. The method is a combination of two separate but complimentary strategies which are also developed in the same chapter.

The first strategy exploits the topology preservation property of SOM. Since the neighboring image blocks have good similarity, and in SOM similar codewords are positioned nearby on the lattice, in this strategy given the index of one block, the match for the next (neighboring) block is restricted to a small search window defined over the SOM lattice and centered at the position of the codeword of the previous block. If the best match within the window exceeds a prefixed quality threshold, it is accepted as the code-

word. Otherwise, rest of the codebook is exhaustively searched for the best match. In the second strategy we train a smaller SOM with the codewords of the main codebook. Then we partition the codewords in the main codebook according to their closeness to the nodes of the smaller codebook. Now, for finding the codeword for a block first we search for the best-matching node in the smaller SOM and then select the best-matching codeword from corresponding subset of the main codebook.

Our proposed method is combination of the above two strategies. Here, first a search is made within the window as per Strategy 1. If a good match is not found, instead of exhaustive search, the codeword is chosen through the smaller codebook and its associated set of codewords of the main codebook as prescribed in Strategy 2. Both Strategy 1, Strategy 2 and the combined method are tested by constructing a large codebook with large vectors and using some benchmark test images. It was found that a very significant reduction in computation, compared to exhaustive search, can be achieved through these methods, especially the combined method. Further, it can be observed that in most cases, the best codeword is found within a small search window around the previous codeword. Thus, if the indexes of the codewords are expressed by their lattice coordinates and successive indexes are expressed in the form of difference of lattice coordinates, then the indexes will be represented as tuples having mostly small numbers. This observation is used here to develop a scheme for efficient Huffman coding of the indexes.

9.2 Future works

There are several possible extensions and/or improvements that can be envisaged for the works reported in this thesis. In Chapter 3 we have proposed a rank correlation based measure for topology preservation in SOM. Though the measure is very useful, due to the large number of ties among the distances calculated over the SOM lattice, the index effectively does not span the entire range [-1,1]. If the effective range can be extended, the resolution of the measure will also be increased. Further, there is a possibility of using the topology preservation measures for identifying information rich feature subsets from high dimensional data. If in the data there exists some random, information-poor components, their contribution to the formation of the map is expected to be limited. Thus, if an index of topology preservation of the SOM for a subset of features, excluding the random features is computed and that of a subset including the random features is computed, then the index is likely to produce a higher value in the former case.

For the fuzzy rule extraction method reported in Chapter 5 a neural fuzzy system may be designed for the same purpose. Also there is a possibility of devising pruning/merging

techniques for the fuzzy rules. However, in the present case since the effort is made from the beginning to keep the number of rules low, such techniques are not likely to have much effect.

The works on evidence theoretic decision making for fuzzy rule based classifiers reported in Chapter 6 can be extended in several ways. Variants of the schemes proposed here can be customized for other data types with contextual correlation such as speech signal, various financial time series etc. However, in each case the customization of these schemes must take into consideration the domain-specific nature of the problems. There is also a scope of generalizing the proposed schemes for any type of data (i.e. data sets without identified context). There the context of the pattern under consideration has to be identified by the method itself. One such scheme might be using the training data along with the fuzzy rule base, so that "k-nearest neighbors" of the pattern under consideration form the context and the fuzzy labels of the k neighbors along with the fuzzy label of the pattern can be used for decision making. Further, since the k nearest neighbors are at different distances from the pattern of interest, the contribution of the neighbors in decision making can be made weighted.

In Chapter 7 we have presented a spatial vector quantization (SVQ) scheme for image compression. The scheme can be easily modified to work with temporal signals like audio signals. One can also look into the possibility of applying the same for transform coded signals, such as cosine transform or various wavelet transforms. However, then the formulation of the smoothening scheme for the codevectors with surface fitting needs to take into account the transform-specific characteristics. There is another interesting possibility. While forming the surface fitting codebook, the codewords may be stored in form of coefficient vectors and the vector to be quantized is also presented in form of a coefficient vector corresponding to a polynomial surface representing the block. The search for the matching codeword is to be carried out based on similarity of the coefficient vectors. However, the space spanned by coefficient vectors can no longer be treated as Euclidean, since the coefficients corresponding to terms of different order have different level of impact in constructing the surface.

The SOM-based fast codebook search method for VQ can be extended for temporal signals as well as transform coded signals. Further, there is a potential application of the method in encoding image sequences, such as frames in video streams, where successive frames have a high correlation. Here the method can exploit both the spatial correlation as well as temporal correlation of the blocks in the image frames to find a good codevector in a SOM-based codebook.

Bibliography

[1] http://www.bangor.ac.uk/ mas00a/z.txt.

[2] http://www.stats.ox.ac.uk/ riplay/prnn.

[3] S. Abe and M. S. Lan. A method for fuzzy rule extraction directly from numerical data and its application to pattern classification. *IEEE Trans. Fuzzy Systems*, 3(1):18–28, 1995.

[4] S. Abe and R. Thawonmas. A fuzzy classifier with ellipsoidal regions. *IEEE Trans. Fuzzy Systems*, 5(3):358–368, 1997.

[5] H. Abut. *IEEE Reprint Collection*, chapter Vector Quantization. IEEE Press, Piscataway, New Jersey, MAY 1990.

[6] F. Aiolli and A. Sperduti. Multiclass classification with multi-prototype support vector machines. *Journal of Machine Learning Research*, 6:817–850, 2005.

[7] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Trans. Neural Networks*, 11(3):601–614, 2000.

[8] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. *Data Mining and Coputational Intelligence*, volume 68 of *Studies in fuzziness and soft computing*, chapter Mining a growing feature map by data skeleton modelling, pages 217–250. Physica-Verlag, New York, 2001.

[9] C. Amerijckx, M. Verleysen, P. Thissen, and J. Legat. Image compression by self-organized kohonen map. *IEEE Trans. Neural Networks*, 9(3):503–507, 1998.

[10] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, 1973.

[11] J. A. Anderson, A. Pellionisz, and E Rosenfeld, editors. *Neurocomputing 2: Directions for Research*. MIT Press, Cambridge, Mass., 1990.

[12] M. A. Andrade, G. Casari, C. Sander, and A. Valencia. Classification of protein families and detection of the determinant residues with an improved self organizing map. *Biol. Cyb.*, 76:441–50, 1997.

[13] F. Bac ao, V. Lobo, and M. Painho. *Computational Science ICCS 2005*, volume LNCS 3516, chapter Self-organizing maps as substitutes for k-means clustering, pages 476–483. Springer-Verlag, Berlin, 2005.

[14] P. M. Atkinson and A. R. L. Tatnall. Neural networks in remote sensing: An introduction. *Int. J. Remote Sensing*, 18:699–709, 1997.

[15] A. S. Atukorale and P. N. Suganthan. Hierarchical overlapped neural-gas network with application to pattern classification. *Neurocomputing*, 35:165–176, 2000.

[16] R. L. Baker and R. M. Gray. Differential vector quantization of achromatic imagery. *Proc. Int. Picture Coding Symp.*, pages 105–106, 1983.

[17] P. V. Balakrishnan, M. C. Cooper, V.S. Jacob, and P.A. Lewis. A study of the classification capabilities of neural networks using unsupervised learning: a comparison with k-means clustering. *Psychometrika*, 59(4):509–525, 1994.

[18] G. H. Ball and C. J. Hall. Isodata, a novel method of data analysis and clasification. Tech. rep., Stanford University, Stanford, CA, 1965.

[19] A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition part i and ii. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 29(6):778–801, 1999.

[20] J. M. Barbalho, A. D. D. Neto, J. A. E. Costa, and M. L. A. Netto. Hierarchical SOM applied to image compression. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 442–447, 2001.

[21] A. Bárdossy and L. Samaniego. Fuzzy rule-based classification of remotely sensed imagery. *IEEE Trans. Geosci. Remote Sensing*, 40(2):362–374, 2002.

[22] H. Bauer and K. R. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans. on Neural Networks*, 3(4):570–579, 1992.

[23] H. U. Bauer. Exploiting topography of neural maps: A case study on investment strategies for emerging markets. *IEEE IAFE Conference on Computational Intelligence for Financial Engineering (CIFEr)*, pages 216–219, 1998.

[24] H. U. Bauer, R. Der, and M. Hermman. Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8:757–771, 1996.

[25] H. U. Bauer and T. Villmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8(2):218–26, 1997.

[26] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society (London)*, 53:370–418, 1763.

[27] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy. Conjugate-gradient neural networks in classification of multisource and very high-dimension remote sensing data. *Int. J. Remote Sensing*, 14:2883–2903, 1993.

[28] M. Bereau and B. Dubuisson. A fuzzy extended k-nearest neighbor rule. *Fuzzy Sets and Systems*, 44:17–32, 1991.

[29] P. Berkhin. Survey of clustering data mining techniques. *http://citeseer.nj.nec.com/berkhin02survey.html*, 2002.

[30] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, New York, 1996.

[31] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

[32] J. C. Bezdek, J. Keller, R. Krishnapuram, and N. R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Boston, 1999.

[33] J. C. Bezdek and N. R. Pal. Index of topological preservation for feature extraction. *Pattern Recognition*, 28(3):381–91, 1995.

[34] J. C. Bezdek and N. R. Pal. A note on self-organizing semantic maps. *IEEE Trans. on Neural Networks*, 6(5):1029–1036, 1995.

[35] S. M. Bhandarkar, J. Koh, and M. Suk. A hierarchical neural network and its application to image segmentation. *Mathematics and Computers in Simulation*, 41(3–4):337–55, 1996.

[36] H. Bischof, W. Schneider, and A. J. Pinz. Multispectral classification of landsat-images using neural networks. *IEEE Trans. on Geosci. Remote Sensing*, 30(3):482–490, 1992.

[37] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[38] S. Biswas. Segmentation based compression for gray level images. *Pattern Recognition*, 36:1501–1517, 2003.

[39] E. D. Bodt, M. Cottrell, and M. Verlysen. Statistical tools to assess the reliability of self-organizing maps. *Neural Networks*, 15:967–978, 2002.

[40] C. Bouton and G. Pagès. Self-oraganization and asymptotic convergence of the one-dimensional kohonen algorithm with non-uniformly distributed stimuli. *Stochastic Processes and Their Applications*, 47:249–274, 1993.

[41] M. M. Campos and G. A. Carpenter. S-tree: self-organizing trees for data clustering and online vector quantization. *Neural Networks*, 15(505–525), 2001.

[42] R. L. Cannon, J. V. Dave, J. C. Bezdek, and M. M. Trivedi. Segmentation of a thematic mapper image using the fuzzy c-means clustering algorithm. *IEEE Trans. Geosci. Remote Sensing*, 24(3):400–408, 1986.

[43] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizaing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[44] S. Carrato. Image vector quantization using ordered codebooks: Properties and applications. *Signal Processing*, 40(1):87–103, 1994.

[45] G. Cazuguel, A. Cziho, B. Solaiman, and C. Roux. Medical image compression and characterization using vector quantization: an application of self-organizing maps and quadtree decomposition. In S. Laxminarayan and E. Micheli-Tzanakou, editors, *Proceedings. 1998 IEEE International Conference on Information Technology Applications in Biomedicine, ITAB '98*, pages 127–32. IEEE, New York, NY, USA, 1998.

[46] J. C. W. Chan, DeFries R. S., and J. R. G. Townshend. Improved recognition of spectrally mixed land cover classes using spatial textures and voting classifications. In *Computer Analysis of Images and Patterns. 9th International Conference, CAIP 2001. Proceedings (Lecture Notes in Computer Science Vol.2124). Springer-Verlag, Berlin, Germany*, pages 217–27, 2001.

[47] P. C. Chang and R. M. Gray. Gradient algorithms for designing adaptive vector quantizer. *IEEE Trans. ASSP*, ASSP-34:679–690, 1986.

[48] R. L. P. Chang and T. Pavlidis. Fuzzy decision tree algorithms. *IEEE Tr. on Syst. Man and Cyberns.*, 7(1):28–35, 1977.

[49] S. W. Changchien and T. C. Lu. Mining association rules procedure to support on-line recommendation by customers and product fragmentation. *Expert Systems with Application*, 20:325–335, 2001.

[50] G. J. Chappell and J. G. Taylor. The temporal Kohonen map. *Neural Networks*, 6:441–445, 1993.

[51] H. Chen, C. Schuffels, and R. Orwig. Internet categorization and search: a self-organizing approach. *Journal of Visual Communication and Image Representation*, 7(1):88–102, 1996.

[52] Y. Chen and J. Z. Wang. Support vector learning for fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Syst.*, 11(6):716–728, 2003.

[53] Y. Cheng. Convergence and ordering of kohonen's batch map. *Neural Computing*, 9:1667–1676, 1997.

[54] Z. Chi, J. Wu, and H. Yan. Handwritten character numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28(1):59–66, 1995.

[55] Z. Chi and H. Yan. Handwritten character numeral recognition using a small number of fuzzy rules with optimized defuzzification parameters. *Neural Networks*, 8(5):821–827, 1995.

[56] Z. Chi and H. Yan. Id3-derived fuzzy rules and optimized defuzzification. *IEEE Trans. Fuzzy Syst.*, 4(1):24–31, 1996.

[57] C.-K. Chiang, H.-Y Chung, and J.-J. Lin. A self-learning fuzzy logic controller using genetic algorithms with reinforcements. *IEEE Trans. Fuzzy Syst.*, 7:460–467, 1997.

[58] S. L. Chiu. Fuzzy model identification based on cluster estimation. *J. Intell. and Fuzzy Syst.*, 2:267–278, 1994.

[59] S. L. Chiu. *Fuzzy Information Engineering*, chapter Extracting fuzzy rules from data for function approximation and pattern classification, pages 149–162. Wiley and Sons, 1997.

[60] C. I. Christodoulou, S. C. Michaelides, C.S. Pattichis, and K. Kyriakou. Classification of satellite cloud imagery based on multi-feature texture analysis and neural networks. In *IEEE International Conference on Image Processing*, volume 1, pages 497–500, 2001.

[61] I. Cloete and J.M. Zurada, editors. *Knowledge-Based Neurocomputing*. MIT Press, Cambridge, Massachusetts, 2000.

[62] J. A. Corral, M. Guerrero, and P. J. Zufria. Image compression via optimal vector quantization: A comparison between som, lbg and k-means algorithms. In *Proceedings of IJCNN*, pages 4113–4118, 1994.

[63] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[64] P. C. Cossman, R. M. Gray, and M. Vetteri. Vector quantization of image subbands: A survey. *IEEE Trans. Image Processing*, 5, 1996.

[65] M. Cottrell, J. C. Fort, and G. Pagès. Theoretical aspects of som algorithm. *Neurocomputing*, 21:119–138, 1998.

[66] M. Cottrell and G. Pagès. Ètude d'un processus d'auto-organisation (in french). *Annales de l'Institut Henri Poincaé*, 47:1–20, 1987.

[67] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with application in pattern recognition. *IEEE Trans. Electronic Computers*, EC-14:326–334, 1965.

[68] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, IT-13(1):21–27, 1967.

[69] A. Czihó, B. Solaiman, I. Lováni, G. Cazuguel, and C. Roux. An optimization of finite-state vector quantization for image compression. *Signal Proc. Image Comm.*, 15(545–558), 2000.

[70] B. Dasarathy. *Nearest Neighbors Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[71] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.

[72] G. Deboeck and T. Kohonen, editors. *Visual Exploration in Finance with Self-Oraganizing Maps*. Springer-Verlag, London, 1998.

[73] P. Demartines and F. Blyao. Kohonen self-organizing maps: Is the normalization necessary? *Complex Syst.*, 6:105–123, 1992.

[74] D. Deng and N. Kasabov. ESOM: An algorithm to evolve self-organizing maps from on-line data streams. In *Proceedings of the International Joint Conference on Neural Networks*, volume 6, pages 3–8, Piscataway, NJ, 2000. Univ of Otago, IEEE.

[75] D. Deng and N. Kasabov. On-line pattern analysis by evolving self-organising maps. *Neurocomputing*, 51:87–103, 2003.

[76] T. Denœux. A k-nearest neighbor classification rule based on dempster-shafer theory. *IEEE Trans. Systems Man Cyberns*, 25(5):804–813, 1995.

[77] T. Denœux. A neural network classifier based on dempster-shafer theory. *IEEE Trans. Syst., Man, Cybern. A*, 30:131150, 2000.

[78] T. Denœux and M.-H. Masson. Evclas: Evidential clustering of proximity data. *IEEE Trans. Syst., Man, Cybern. B*, 34(1):95–109, 2004.

[79] R. Der, M. Herrmann, and T. Villmann. Time behavior of topological ordering in self-organized feature mapping. *Biol. Cybern.*, to appear, 1994.

[80] R. Der and T. Villmann. Dynamics of selforganizing feature mapping. *New Trends in Neural Computation*, LNCS 686:312–315, 1993.

[81] D. DeSieno. Adding a conscience to competitive learning. *IEEE Intl. Conf. on Neural Networks*, 1:117–125, 1988.

[82] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, New Jersey, 1982.

[83] L. P. Devroye. On inequality of cover and hart in nearest neighbor discrimination. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-3(1):75–78, 1981.

[84] J. Dickerson and B. Kosko. Fuzzy function learning with covariance ellipsoids. In *Proc. IEEE Intl. Conf. Neural Networks*, volume III, pages 1162–1167, San Fransisco, CA, 1993.

[85] E. Diday. The dynamic cluster method in non-hierarchical clustering. *J. Comput. Inf. Sci.*, 2:61–88, 1973.

[86] Robert D. Dony and Simon Haykin. Neural network approaches to image compression. *Proc. of the IEEE*, 83(2):288–303, 1995.

[87] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Application*. Academic, New York, 1980.

[88] R. Duda and P. Hart. *Patern Classification and Scene Analysis*. Wiley Interscience, NY, New York, 1973.

[89] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, New York, 2000.

[90] R. Durbin and G. Mitchison. A dimension reduction framework for understanding cortical maps. *Nature*, 343:644–647, 1990.

[91] H. Dyckhoff and W. Pedrycz. Generalized means as models of compensative connectives. *Fuzzy Sets and Systems*, 14:143–154, 1984.

[92] P. Dylan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001.

[93] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Ordering, convergence property and energy functions. *Biological Cybernetics*, 67(1):47–55, 1992.

[94] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, 67(1):35–45, 1992.

[95] J. E. Estes and D. W. Mooneyhan. Of maps and myths. *Photogrammetric Engineering and Remote Sensing*, 60:517–524, 1994.

[96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smith. The kdd process for extracting useful knowledge from volumes of data. *Communications of ACM*, 39(11):27–34, 1996.

[97] E. A. Ferran. Self-organized neural maps of human protein sequences. *Protein Science*, 3(3):507–521, Mar 1994.

[98] R. A. Fisher. The use of multiple measurement in taxonoic problems. *Annals of Eugenics*, 7 Part II:179–188, 1936.

[99] J. A. Flanagan. Sufficient conditions for self-organization in the one-dimensional som with a reduced width neighborhood. *Neurocomputing*, 21:51–60, 1998.

[100] J. A. Flanagan. Self-organization in the one-dimensional som with a decreasing neighborhood. *Neural Networks*, 14:1405–1417, 2001.

[101] G. M. Foody. Approaches for the production and evaluation of fuzzy land cover classifications from remotely-sensed data. *Int. J. of Remote Sensing*, 17(7):1317–1340, 1996.

[102] J. C. Fort and G. Pagès. On the asymptotic covergence of the kohonen algorithm with a general neighborhood function. *Annals of Applied Probability*, 5(4):1177–1216, 1995.

[103] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for for finding best matches in logarithmic expected time. *ACM Trans. Mathematical Software*, 3(3):209–266, 1977.

[104] B. Fritzke. Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[105] B. Fritzke. *Advances in Neural Information Processing Systems*, volume 7, chapter A growing neural gas network learns topologies, pages 625–632. MIT Press, Cambridge MA, 1995.

[106] B. Fritzke. Growing grid—a self-organizing network with constant neighbourhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, Sept 1995.

[107] B. Fritzke. Some competitive learning methods. Draft document, 1998. http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG.

[108] ftp://ftp.dice.ucl.ac.be/pub/neural nets/ELENA.

[109] F. Fukunaga and P .M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. on Computers*, 24:750–753, 1975.

[110] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[111] A. Golbraikh, P. Bernard, and J. R. Chretien. Validation of protein-based alignment in 3d quantitative structure-activity relationships with coMFA models. *European Journal of Medicinal Chemistry*, 35(1):123–136, Jan 2000.

[112] P. Gong. Integrated analysis of spatial data from multiple sources: Using evidential reasoning and artificial neural network techniques for geological mapping. *Photogrammetric Engineering and Remote Sensing*, 62(5):513–523, 1996.

[113] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addision-Wesley, Reading, Mass., 1992.

[114] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. Information Theory*, 44(6):1–63, 1998.

[115] S. Günter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23:405–417, 2002.

[116] E. E. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. IEEE Conf. on Decision and Control, San Diego*, pages 761–766, Piscataway, NJ, 1979. IEEE Press.

[117] M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Trans. Neural Networks*, 14(3):491–505, 2003.

[118] M. Hagiwara. Self-organizing feature map with a momentum term. *Neurocomputing*, 10(1):71–81, January 1996.

[119] B. Hammer, A. Michelli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, 17:1061–1085, 2004.

[120] R. Hamzaoui. Codebook clustering by self-organizing maps for fractal image compression. *Fractals*, 5(suppl. issue):27–38, 1997.

[121] R. Hamzaoui and D Saupe. Combining fractal image compression and vector quantization. *IEEE Trans. on Image Processing*, 9(2):197–208, 2000.

[122] R. M. Haralick. Decision making in context. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5:417–428, 1983.

[123] T. Hastie, R. Tibshirani, and J Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[124] S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Addison Wesley Longman, Singapore, 1999.

[125] D.O. Hebb. *The Organization of Behaviour*. John Wiley and Sons, New York, 1949.

[126] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Publishing Company, CA, USA, 1995.

[127] W. H. Highleyman. Linear decision functions with application to pattern recognitio. *Proceedings of IRE*, 50:1501–1514, 1962.

[128] H.Ishibuchi, T.Nakashima, and T.Murata. *A fuzzy classifier system that generates linguistic rules for pattern classification problems*, volume 1152 of *Lecture Notes in Artificial Intelligence*, pages 35–54. Springer-Verlag, Berlin, 1996.

[129] R. C. Holte. Very simple classification rules perform well on most commonly used data sets. *Machine Learning*, 11:63–91, 1993.

[130] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

[131] K. Hornik, M. Stinchcombe, and H. White. Multilayered feedforward networks are universal approximators. *Neural Networks*, 2:259–366, 1989.

[132] N. C. Hsieh. An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems with Applications*, 27:623–633, 2004.

[133] N. C. Hsieh. Hybrid minig approach in design of credit scoring model. *Expert Systems with Applications*, 28:655–665, 2005.

[134] W. Huaichun, J. Dopazo, L. G. de la Fraga, Y. P. Zhu, and J. M. Carazo. Self-organizing tree-growing network for the classification of protein sequences. *Protein Science*, 7(12):2613–2622, 1998.

[135] D. A. Huffman. A method of construction of minimum redundancy codes. *Proceedings of IRE*, 40:1098–1101, 1952.

[136] H. Ishibuchi and T. Nakashima. Effect of rule weights in fuzzy rulebased classification systems. *IEEE Trans. on Fuzzy Systems*, 9(4):506–515, August 2001.

[137] H. Ishibuchi, T. Nakashima, and T. Murata. Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems. *IEEE Trans. on Syst. Man and Cybern: B*, 29(5):601–618, 1999.

[138] H. Ishibuchi and T. Yamamoto. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining,. *Fuzzy Sets and Systems*, 141(1):59–88, 2004.

[139] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[140] A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[141] C. Z. Janikow. Fuzzy decision trees: issues and methods. *IEEE Tr. on Syst. Man and Cyberns.-Part B*, 28(1):1–14, 1998.

[142] T. Joachims. *SVM-light support vector machine*, 2002. http://svmlights.joachims.org.

[143] A. Jouan and Y. Allard. Land use mapping with evidential fusion of features extracted from polarimetric synthetic aperture radar an hyperspectral imagery. *Information Fusion*, 5:251–267, 2004.

[144] A. B. R. Klautau Jr. Predictive vector quantization with intrablock predictive support region. *IEEE Trans. on Image Processing*, 8(2):293–295, 1999.

[145] J. W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, C-18:401–409, 1969.

[146] J. Kangas, T. Kohonen, J. Laaksonen, O. Simula, and O. Ventä. Variants of self-organizing maps. In *Proc. IJCNN'89, International Joint Conference on Neural Networks*, volume II, pages 517–522, Piscataway, NJ, 1989. IEEE Service Center.

[147] J. A. Kangas, T. K. Kohonen, and J. T. Laaksonen. Variants of self-organizing maps. *IEEE Trans. on Neural Networks*, 1(1):93–99, 1990.

[148] N. B. Karayiannis. Fuzzy vector quantization algorithms and their application in image compression. *IEEE Trans. Image Processing*, 4(3):1193–1201, 1995.

[149] N. Kasabov. On-line learning, reasoning, rule extraction and aggregation in locally optimised evolving fuzzy neural networks. *Neurocomputing*, 41:25–41, 2001.

[150] N. Kasabov, D. Deng, L. Erzegovezi, M. Fedrizzi, and A. Beber. On-line decision making and prediction of financial and macroeconomic parameters on the case study of the european monetary union. In H. Bothe and R. Rojas, editors, *Proceeding of the ICSC Symposia on Neural Computation (NC'2000)*. ICSC Academic Press, 2000.

[151] N. Kasabov and E. Peev. Phoneme recognition with hierarchical Self Organised neural networks and fuzzy systems—a case study. In Maria Marinaro and Pietro G. Morasso, editors, *Proc. ICANN'94, International Conference on Artificial Neural Networks*, volume I, pages 201–204, London, UK, 1994. Springer.

[152] N. Kasabov and Q.Song. Denfis: Dynamic evolving neural-fuzzy inference system and its application for time series prediction. *IEEE Trans. Fuzzy Systems*, 10(2):144–154, 2002.

[153] N. K. Kasabov. Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy Sets and Syst.*, 82(2):135–149, 1996.

[154] S. Kaski, T. Henkela, K. Lagus, and T. Kohonen. Websom – self-organizing maps for document collections. *Neurocomputing*, 21:101–117, 1998.

[155] S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (som) papers: 1981–1997. *Neural Computing Survays (online Journal at http://www.cse.ucsc.edu/NCS/)*, 1:102–350, 1998.

[156] S. Kaski, J. Venna, and T. Kohonen. Coloring that reveals cluster structures in multivariate data. *Australian-Journal-of-Intelligent-Information-Processing-Systems*, 6:82–88, 2000.

[157] J. M. Keller, P. Gader, H. Tahani, J.-H. Chiang, and M. Mohamed. Advances in fuzzy integration for pattern recognition. *Fuzzy Sets and Systems*, 65:273–283, 1994.

[158] J. M. Keller, M. R. Gray, and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Trans. Syst. Man Cybern*, 15(4):580–585, 1985.

[159] J. M. Keller and D. J. Hunt. Incorporating fuzzy membership function into the perceptron algorithm. *IEEE Trans. Patt. Anal. Machine Intell.*, 7:693–699, 1985.

[160] M. Kendall and J. D. Gibbons. *Rank Correlation Coefficient*. Edward Arnold, 1990.

[161] H. Khalil and K. Rose. Predictive vector quantizer design using deterministic annealing. *IEEE Trans. Signal Processing*, 51(1):244–254, 2003.

[162] H. Kim and P. H. Swain. Evidential reasoning approach to multisource-data classification in remote sensing. *IEEE Trans. on Syst. Man and Cybern*, 25(8):1257–1265, 1995.

[163] K. J. Kim and S. B. Cho. Fuzzy integration of structure adaptive soms for web content mining. *Fuzzy Sets and Systems*, 148:43–60, 2004.

[164] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice-Hall, New Jersy, 1995.

[165] E. I. Knudsen, S. Du Lac, and S. D Esterly. Computational maps in brain. *Ann. Rev. Neurosci.*, 10:41–65, 1987.

[166] T. Kohonen. Automatic formation of topological maps of patterns in a self-organizing sysyem. *Proc. 2nd Scandinavian Conf. on Image Analysis*, pages 214–220, 1981.

[167] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59–69, 1982.

[168] T. Kohonen. *Self-organization and associative memory*. Springer Series in Information Sciences. Springer-Verlag, Berlin, 2 edition, 1988.

[169] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–1480, 1990.

[170] T. Kohonen. *Artificial Neural Networks*, chapter Self-organizing maps: Optimization approaches, pages I–891–990. North Holland, 1991.

[171] T. Kohonen. The Adaptive-Subspace SOM (ASSOM) and its use for the implementation of invariant feature detection. In F. Fogelman-Soulié and P. Gallinari, editors, *Proc. ICANN'95, International Conference on Artificial Neural Networks*, volume I, pages 3–10, Nanterre, France, 1995. EC2.

[172] T. Kohonen. Self-organizing maps of symbol strings. Technical Report A42, Laboratory of Computer and Information Science, Helsinki University of Technology, Finland, 1996.

[173] T. Kohonen. The speedy SOM. Technical Report A33, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[174] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 3 edition, 2001.

[175] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self-organization of a massive document collection. *IEEE Tr. Neural Networks*, 11(3):574–585, 2000.

[176] T. Kohonen and E. Oja. A note on a simple self-organizing process. Report TKK-F-A474, Helsinki University of Technology, Espoo, Finland, 1982.

[177] T. Kohonen and E. Oja. Visual feature analysis by the self-organising maps. *Neural Computing & Applications*, 7:273–286, 1998.

[178] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–84, 1996.

[179] T. Kohonen and P. Somarvuo. Self-organizing maps on symbol strings. *Neurocomputing*, 21:19–30, 1998.

[180] T. Kohonen and P. Somarvuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15:945–952, 2002.

[181] P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *Proc. IJCNN-90, International Joint Conference on Neural Networks, Washington, DC*, volume II, pages 279–285, Piscataway, NJ, 1990. IEEE Service Center.

[182] A. König. Interactive visualization and analysis of hierarchical neural projections for data mining. *IEEE Transactions on Neural Networks*, 11(3):615–624, 2000.

[183] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Recurrent som with local linear models in time series prediction. In *6th European Symposium on Artificial Neural Networks. ESANN'98. Proceedings*, pages 167–172, Brussels, Belgium, 1998. D-Facto.

[184] F. Kossentini, W. Chung, and M. Smith. Conditional entropy constrained residual vq with application to image coding. *IEEE Trans. on Image Processing*, 5:311–321, 1996.

[185] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear prjection method based on kohonen's topology preserving maps. *IEEE Tr. Neural Networks*, 6(3):548–559, 1995.

[186] V. Kreinovich, G. C. Mouzouris, and H. T. Nguyen. *Fuzzy Systems: Modeling and Control*, chapter Fuzzy rule based modelig as a universal approximation tool, pages 136–195. Kluwer Academic Publisher, 1998.

[187] R. Krishnapuram and J. Keller. A possibilistic approach to clustering. *IEEE Trans. Fuzzy Syst.*, 1(2):98–110, 1993.

[188] A. Kulkarni and S. McCaslin. Knowledge discovery from multispectral satellite images. *IEEE Geosci. Remote Sensing Letters*, 1(4):246–250, 2004.

[189] S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh. Learning pattern classification - a survey. *IEEE Trans. Information Theory*, 44(6):2178–2206, 1998.

[190] A. S. Kumar, S. Chowdhury, and K. L. Majumder. Combination of neural and statistical approaches for classifying space-borne multispectral data. *Proc. of ICAPRDT99*, pages 87–91, 1999.

[191] L. Kuncheva. *Fuzzy Classifier Design*. Physica-Verlag, Heidelberg, 2000.

[192] L. I. Kuncheva. How good are fuzzy if-then classifiers? *IEEE Trans. Syst. Man and Cyberns.: Part B*, 30(4):501–509, 2000.

[193] L. I. Kuncheva and J. C. Bezdek. An integrated framework for generalized nearest prototype classifier design. *International Journal of Uncertainty, Fuzzyness and Knowledge-based Systems*, 6(5):437–457, 1998.

[194] M. Kunt, M. Benard, and R. Leonardi. Recent results in high compression image coding. *IEEE Trans. Circuits Systems*, 34:1306–1336, 1987.

[195] S. Laakso, J. Laaksonen, M. Koskela, and E. Oja. Self-organising maps of web link information. In N. Allinson, H. Yin, L. Allinson, and J. Slack, editors, *Advances in Self-Organising Maps*, pages 146–151. Springer, 2001.

[196] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. PicSOM—content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13–14):1199–1207, 2000.

[197] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. Self-organising maps as a relevance feedback technique in content-based image retrieval. *Pattern Anal. Appl.*, 4((2-3)):140–152, 2001.

[198] J. Laaksonen, M. Koskela, and E. Oja. PicSOM—self-organizing image retrieval with mpeg-7 content descriptors. *IEEE Trans. Neural Networks*, 13(4):841–853, 2002.

[199] A. Laha. An empirical study on the robustness of som in preserving topology with respect to link density. In *Proceedings of 11th International Conference on Neural Information Processing (ICONIP 2004)*, volume LNCS 3316, pages 142–149, Berlin, 2004. Springer-Verlag.

[200] A. Laha, B. Chanda, and N. R. Pal. Fast codebook searching in a som-based vector quantizer for image compression. *Image and Vision Computing (Communicated)*.

[201] A. Laha and J. Das. Fuzzy rules and evidence theory for satellite image analysis. In *Proc. 5-th Intl. Conf. Advances in Pattern Recognition (ICAPR 2003)*, pages 453–457, 2003.

[202] A. Laha and N. R. Pal. On different variants of self-organizing feature map and their properties. In *Proceedings of the 1999 IEEE Hong Kong Symposium on Robotics and Controls*, volume 1, pages I–344–I–349, 1999.

[203] A. Laha and N. R. Pal. Dynamic generation of prototypes with self-organizing feature maps for classifier design. *Pattern Recognition*, 34(2):315–321, 2000.

[204] A. Laha and N. R. Pal. Some novel classifiers designed using prototypes extracted by a new scheme based on self-organizing feature map. *IEEE Trans. on Syst. Man and Cybern: B*, 31(6):881–890, 2001.

[205] A. Laha, N. R. Pal, and B. Chanda. Design of vector quantizer for image compression using self-organizing feature map and surface fitting. *IEEE Trans. Image Processing*, 13(10):1291–1303, 2004.

[206] A. Laha, N. R. Pal, and J. Das. Designing prototype-based classifiers and their application to classification of multispectral satellite images. In *Proceedings of 6th International Conference on Soft Computing (IIZUKA2000)*, pages 861–868, 2000.

[207] A. Laha, N. R. Pal, and J. Das. Land cover classification using fuzzy rules and aggregation of contextual information through evidence theory. *IEEE Trans. Geosc. and Remote Sensing*, 44(6):1633–1641, 2006.

[208] J. Z. C. Lai and Y.-C. Liaw. Fast-searching algorithm for vector quantization using projection and triangular inequality. *IEEE Trans. on Image Processing*, 13(12):1554–1558, 2004.

[209] J. Lampinen and T. Kostiainen. Generative probability density model in the self-organizing map. In U. Seiffert and L. Jain, editors, *Self-organizing neural networks: Recent advances and applications*, page 7594. Physica Verlag, Berlin, 2002.

[210] J. H. Lee and S. C. Park. Intelligent profitable customers segmentation system based on business intelligence tools. *Expert Systems with Applications*, 29:145–152, 2005.

[211] T. Lee, J. A. Richards, and P. H. Swain. Probabilistic and evidential approaches for multispectral data analysis. *IEEE Trans. on Geosci. Remote Sensing*, 25:283–293, 1987.

[212] R. Y. Li, J. Kim, and N. Al-Shamakhi. Image compression using transformed vector quantization. *Image and Vision Computing*, 20(1):37–45, Jan 1 2002.

[213] W. Light. *Approximation Theory VII*, chapter Ridge functions, sigmoidal functions and neural networks, pages 163–205. Academic Prss, Boston, 1992.

[214] C. T. Lin, F. B. Duh, and D. J. Liu. A neural fuzzy network for word information processing. *Fuzzy Sets and Systems*, 127(1):37–48, 2002.

[215] C. T. Lin and C. S. G. Lee. Neural network-based fuzzy logic and control and decision system. *IEEE Trans. Comput.*, 40(12):1320–1336, 1991.

[216] C. T. Lin and C. S. G. Lee. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall International, New Jersey, 1996.

[217] C. T. Lin, Y. C. Lee, and H. C. Pu. Satellite sensor image classification using cascaded architecture of neural fuzzy network. *IEEE Tr. Geosci. Remote Sens.*, 38(2):1033–1043, 2000.

[218] C. T. Lin and Y. C. Lu. A neural fuzzy system with fuzzy supervised learning. *IEEE Trans. Syst. Man, and Cybern. B*, 26(5):744–763, 1996.

[219] J. K. Lin, D. G. Grier, and J. D. Kowan. Faithful representation of separable distributions. *Neural Computation*, 9:1305–1320, 1997.

[220] X. Lin. Map displays for information retrieval. *Journal of the American Society for Information Science*, 48:40–54, 1997.

[221] X. Lin, D. Soergel, and G. Marchionini. A Self-organizing semantic map for information retrieval. In *Proc. 14th. Ann. Int. ACM/SIGIR Conf. on R & D In Information Retrieval*, pages 262–269, 1991.

[222] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, COM-28:84–95, 1980.

[223] P. Lingras, M. Hogo, M. Snorek, and C. West. Temporal analysis of clusters of supermarket customers: conventional versas interval set approach. *Information Sciences*, 172:215–240, 2005.

[224] R. Linsker. How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Computation*, 1:402–411, 1989.

[225] C. T. Liu, P. L. Tai, A. Y. J. Chen, C. H. Peng, and J. S. Wang. A content-based scheme for CT lung image retrieval. In *IEEE International Conference on Multi-Media and Expo*, pages 1203–1206, 2000.

[226] D. Liu and A. Mitchel. Robustness analysis and design of a class of neural networks with sparse interconnecting structure. *Neurocomputing*, 12:59–76, 1996.

[227] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[228] S. P. Lloyd. Least-squares quantization in pcm. *IEEE Trans. Inform. Theory*, IT-28:129–137, 1982.

[229] S. P. Luttrell. Hierarchical self-organising networks. In *Proceedings of IEE International Conference on Artificial Neural Networks*, pages 2–6, 1989.

[230] S. P. Luttrell. Self-organization: A derivation from first principle of a class of learning algorithms. *Proc. IEEE Conference on Neural Networks*, pages 495–498, 1989.

[231] S. P. Luttrell. Derivation of a class of training algorithms. *IEEE Tr. Neural Networks*, 1:229–232, 1990.

[232] S. P. Luttrell. Code vector density in topographic mapping: Scalar case. *IEEE Tr. Neural Networks*, 2:427–436, 1991.

[233] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5-th Burkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[234] P. E. Maher and D. St. Clair. Uncertain reasoning is an id3 machine learning framework. In *Proc. IEEE Int. Conf. on Fuzzy Systems*, pages 7–12. IEEE Press, 1992.

[235] B. Mailachalam and T. Srikanthan. A robust parallel architecture for adaptive color quantization. In *Proceedings International Conference on Information Technology:*

Coding and Computing. IEEE Comput. Soc, Los Alamitos, CA, USA, pages 164–9, 2000.

[236] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. of Man-Machine Studies*, 7:1–13, 1975.

[237] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[238] J. Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering. *IEEE Trans. Neural Networks*, 7(1):16–29, 1997.

[239] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organizing network that grows when required. *Neural Networks*, 15:1041–1058, 2002.

[240] T. Martinetz and K. Schulten. A "Neural-Gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Proc. International Conference on Artificial Neural Networks* (Espoo, Finland), volume I, pages 397–402, Amsterdam, Netherlands, 1991. North-Holland.

[241] T. Martinetz and K. Schulten. Topology preserving networks. *Neural Networks*, 7(3):507–522, 1994.

[242] M.-H. Masson and T. Denœux. Clustering interval-valued proximity data using belief functions. *Pattern Recognition Letters*, 25:163171, 2004.

[243] W. S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys*, 5:115–133, 1943.

[244] D. A. Medler. A brief history of connectionism. *Neural Computing Survey (online journal), http://www.icsi.berkeley.edu/ jagota/NCS,*, 1:61–101, 1998.

[245] Dieter Merkl. Text classification with self-organizing maps: some lessons learned. *Neurocomputing*, 21(1):61–77, 1998.

[246] R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2(1):83–101, 1990.

[247] S. A. Mingoti and J. O. Lima. Comparing som neural network with fuzzy c-means, k-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research*, To be published, 2005.

[248] M. Minsky and S. A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[249] N. M. Nasrabadi and Y. Feng. Vector quantization of images based upon the kohonen self-organization feature maps. *Proc. 2nd ICNN Conf.*, 1:101–108, 1988.

[250] N. M. Nasrabadi and R. A. King. mage coding using vector quantization: A review. *IEEE Trans. Commun.*, 36(8):957–971, 1988.

[251] H. T. Nguyen and M. Sugeno, editors. *Fuzzy Systems: Modeling and Control*. Kluwer Academic, Boston, 1998.

[252] M. N. Nguyen and J. C. Rajapakse. Multi-class support vector machines for protein secondary structure prediction. *Genome Informatics*, 14:218–227, 2003.

[253] J. Nikkilä, P. Törönen, S. Kaski, J. Venna, E. Castrén, and G. Wong. Analysis and visualization of gene expression data using self-organizing maps. *Neural Networks*, 15:953–966, 2002.

[254] K. Nozaki, H. Ishibuchi, and H. Tanaka. Adaptive fuzzy-rule based classification system. *IEEE Trans. Fuzzy Systems*, 4(3), 1996.

[255] E. Oja. Neural networks in image processing and analysis. In *Proc. Symp. on Image Sensing and Processing in Industry*, pages 143—152, Tokyo, Japan, 1991. Pattern Recognition Society of Japan.

[256] E. Oja. Self-organizing maps and computer vision. In Harry Wechsler, editor, *Neural Networks for Perception, vol. 1: Human and Machine Perception*, pages 368–385. Academic Press, New York, NY, 1992.

[257] M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (som) papers: 1998–2001. *Neural Computing Survays (online Journal at http://www.cse.ucsc.edu/NCS/)*, 3:1–156, 2002.

[258] S. Oka, Y. Takefuji, and T. Suzuki. Feature extraction of IKONOS images by self-organization topological map. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology. CISST'2000. CSREA Press - Univ. Georgia, Athens, GA, USA*, volume 2, pages 687–91, 2000.

[259] T. Ong, H. Chen, W. Sung, and B. Zhu. Newsmap: a knowledge map for online news. *Decision Support Systems*, 39:583–597, 2005.

[260] J. Ontrup and H. Ritter. A hierarchically growing hyperbolic self-organizing map for rapid structuring of large data sets. In *Proc. 5-th Workshop on Self-organizing Maps (WSOM05)*, page (To appear), 2005.

[261] N. R. Pal. Soft computing for feature analysis. *Fuzzy Sets and Systems*, 103:201–221, 1999.

[262] N. R. Pal and J. C. Bezdek. On cluster validity for the fuzzy c-means model. *IEEE Trans. on Fuzzy Systems*, 3(3):370–379, 1995.

[263] N. R. Pal and C. Bose. Context sensitive inferencing and reinforcement type tuning algorithms for fuzzy logic systems. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 3(4):230–239, 1999.

[264] N. R. Pal and V. K. Eluri. Two efficient connectionist schemes for structure preserving dimensionality reduction. *IEEE Transactions on Neural Networks*, 9(6):1142–1154, 1998.

[265] N. R. Pal and S. Ghosh. Some classification algorithms integrating dempster-shafer theory of evidence with the rank nearest neighbor rule. *IEEE Trans. Syst. Man and Cybern: Part A*, 31(1):59–66, 2001.

[266] N. R. Pal and A. Laha. A multi-prototype classifier and its application to remotely sensed image analysis. *Australian Journal of Intelligent Information Processing*, 6(2):110–118, 2000.

[267] N. R. Pal, A. Laha, and J. Das. Designing fuzzy rule based classifier using self-organizing feature map for analysis of multispectral satellite images. *International Journal of Remote Sensing*, 26(10):2219–2240, 2005.

[268] S. K. Pal and D. Dutta Majumder. Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Trans. Syst. Man, Cybern*, 7:625–629, 1977.

[269] S. K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Trans. on Neural Networks*, 3(5):683–697, 1992.

[270] S. K. Pal and S. Mitra. *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing*. John Wiley, New York, 1999.

[271] S. K. Pal and A. Pal, editors. *Pattern Recognition: From Classical to Modern Approaches*. World Scientific, Singapore, 2001.

[272] J. D. Paola and R. A. Schowengerdt. A detailed comparison of backpropagation neural network and maximum likelihood classifiers for urban land use classification. *IEEE Trans. on Geosci. Remote Sensing*, 33:981–996, July 1995.

[273] T. Pavlidis. *Structural Pattern Recognition*. Springer-Verlag, New York, 1980.

[274] T. Pavlidis. *Algorithms for Graphics and Image Compression*. Springer-Verlag, New York, 1982.

[275] D. R. Peddle. An empirical comparison of evidential reasoning, linear discriminant analysis and maximum likelihood algorithms for land cover classification. *Canadian J. Remote Sensing*, 19:31–44, 1993.

[276] W. Pedrycz. Classification in a fuzzy environment. *Pattern Recognition Letters*, 3:303–308, 1985.

[277] W. Pedrycz. Conditional fuzzy c - means. *Pattern Recognition Letters*, 17:625–632, 1996.

[278] W. Pedrycz. *Industrial Electronics Handbook*, chapter Fuzzy pattern recognition, pages 1207–1230. CRC Press, 1996.

[279] W. Pedrycz. Fuzzy sets in pattern recognition: accomplishments and challenges. *Fuzzy Sets and Systems*, 2:171–176, 1997.

[280] W. Pedrycz. *Computational Intelligence and Applications*, chapter Computational Intelligence: An Introduction, pages 3–17. Physica-Verlag, Heidelberg, 1999.

[281] W. Pedrycz and H. C. Card. Linguistic interpretation of self-organizing maps. In *Proc. IEEE Int. Conf. on Fuzzy Systems*, pages 371–378, San Diego, 1992.

[282] W. Pedrycz and M. Reformat. Rule-based modelling of nonlinear relationships. *IEEE Trans. Fuzzy Systems*, 2:256–269, 1997.

[283] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Trans. Syst., Man, and Cybern.*, 5:787–795, 1997.

[284] S. C. Pei and Y. S. Lo. Color image compression and limited display using self-organization Kohonen map. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(2):191–205, 1998.

[285] D. Polani. *Self-Organizing Neural Networks*, chapter Measures for the Organization of Self-Organizing Maps, pages 13–44. Physica-Verlag, Heidelberg, 2002.

[286] J. Principe, N. Euliano, and S. Garani. Principles and networks for self-organization in space-time. *Neural Networks*, 15:1069–1083, 2002.

[287] D. Pullwitt. Integrating contextual information to enhance som-based text document clustering. *Neural Networks*, 15:1099–1106, 2002.

[288] T. A. Ramstad, S. O. Aase, and J. H. Husøy. *Subband Compression of Images: Principles and Examples*. Elsivier Science B. V., Amsterdam, 1995.

[289] A. Rauber, D. Merkl, and M. Dittenbach. The graowing hierarchical self-organing map: exploratory analysis of high-dimensional data. *IEEE Trans. Neural Networks*, 13(6):1331–1341, 2002.

[290] H. Ressom, D. Wang, and P. Natarajan. Adaptive double self-organizing maps for clustering gene expression profiles. *Neural Networks*, 16:953–966, 2003.

[291] H. Ritter. Asymptotic level density for a class of vector quantization processes. *IEEE Tr. Neural Networks*, 2:173–175, 1991.

[292] H. Ritter. Self-organizing maps on non-euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–110. Elsevier, Amsterdam, 1999.

[293] H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biol. Cybernet.*, 61:241–254, 1989.

[294] H. Ritter, T. M. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps.* Addison Wesly, Reading, MA, 1992.

[295] H. Ritter and K. Schulten. Kohonen self-organizing maps: exploring their computational capabilities. *Proc. Intl. Conf. on Neural Networks*, pages 109–116, 1988.

[296] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–488, 1958.

[297] D. E Rumelhart, G. E. Hinton, and R. J Williams. *Learning Distributed Processing: Exploration in the Microstructure of Cognition. Vol 1: Foundation.* MIT Press, Cambridge, Mass, 1986.

[298] A. Sadeghi. Self-organization property of kohonen's map with general type of stimuli distribution. *Neural Networks*, 11:1637–1643, 1998.

[299] A. Sarkar, M. K. Biswas, B. Kartikeyan, V. Kumar, K. L. Majumder, and D. K. Pal. A mrf model-based segmentation approach to classification for multispectral imagery. *IEEE Trans. Geosci. Remote Sensing*, 40(5):1102–1113, 2002.

[300] K. Sayood. *Introduction to Data Compression (2nd ed).* Morgan Kaufmann, 2000.

[301] R. Schalkoff. *Pattern Recognition: Statistical, Syntactic and Neural Approaches.* John Wiley and Sons, New York, 1992.

[302] M. Schnaider and A. P. Papliński. Still image compression with lattice quantization in wavelet domain. *Advances in Imaging and Electron Physics*, 119, 2000.

[303] J. F. Schreer, R. J. H. O'Hara, and K. M. Kovacs. Classification of dive profiles: A comparison of statistical clustering techniques and unsupervised artificial neural networks. *Journal of Agriculture Biological and Environmental Statistics*, 3(4):383–404, 1998.

[304] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, New Jersey, 1976.

[305] L. Shen and M. Rangayyan R. A segmentation based lossless image coding method for high resolution medical image compression. *IEEE Trans. Med. Imaging*, 16:301–307, 1997.

[306] R. Singh, V. Cherkassky, and N. Papanikolopoulos. Self-organizing maps for the skeletonization of sparse shapes. *IEEE Trans. Neural Networks*, 11(1):241–248, 2000.

[307] P. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–234, 1994.

[308] K. A. Smith and A. Ng. Web page clustering using a self-organizing map of user navigation patterns. *Decision Support Systems*, 35:245–256, 2003.

[309] A. H. S. Solberg, A. K. Jain, and T. Taxt. Multisource classification of remotely sensed data: Fusion of landsat tm and sar images. *IEEE Trans. on Geosci. Remote Sensing*, 32(4):768–777, 1994.

[310] A. H. S. Solberg, T. Taxt, and A. K. Jain. A markov random field model for clasification of multisouirce satellite imagery. *IEEE Trans. Geosci. Remote Sensing*, 34(1):100–113, 1996.

[311] H. Soliman and A. Abdelali. Colored image compression using neural networks. *Parallel and Distributed Computing and Systems. IASTED/ACTA Press, Anaheim, CA, USA; 2000; 2 vol*, 1:229–31, 2000.

[312] P. Somervuo and T. Kohonen. Clustering and visualization of large protein sequence databases by means of an extension of the self-organizing map. In *Discovery Science. Third International Conference, DS 2000. Proceedings (Lecture Notes in Artificial Intelligence Vol.1967). Springer-Verlag, Berlin, Germany*, pages 76–85, 2000.

[313] A. Srinivasan and J. A. Rechards. Knowledge-based techniques for multi-source classification. *Int. J. Remote sensing*, 11:501–525, 1990.

[314] M. Strickert and B. Hammer. Merge som for temporal data. *Neurocomputing*, 64:39–71, 2005.

[315] M. C. Su and H. T. Chang. Fast self-organizing feature map algorithm. *IEEE Transactions on Neural Networks*, 11(3):721–733, May 2000.

[316] M. C. Su and H. T. Chang. New model of self-organizing neural networks and its application in data projection. *IEEE Transactions on Neural Networks*, 12(1):153–158, 2001.

[317] M. C. Su, H. T. Chang, and C. H. Chou. A novel measure for quantifying the topology preservation of self-organizing feature maps. *Neural Processing Letters*, 15(2):137–145, 2002.

[318] P. N. Suganthan. Shape indexing using self-organizing maps. *IEEE Tr. Neural Networks*, 13(4):835–840, 2002.

[319] T. Takagi and M. Sugeno. Fuzzy identification of systems and its implication to modelling and control. *IEEE Trans. on Systems Man and Cybernetics*, 15:116–132, 1985.

[320] A. Takeuchi and S. Amari. Formation of topographic maps and columner microstructure. *Biolaogical Cybernetics*, 35:63–72, 1979.

[321] R. Talumassawatdi and C. Lursinsap. Fault immunization concept for self-organizing mapping neural networks. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 9:781–790, 2001.

[322] H. Teicher. Identifiability of mixtures. *Annals of Mathematical Statistics*, 32(1):244–248, 1961.

[323] B Tian, M. A. Shaikh, M. R. Azimi Sadjadi, Thomas H. Vonder H., and D. L. Reinke. Study of cloud classification with neural networks using spectral and textural features. *IEEE Transactions on Neural Networks*, 10(1):138–151, 1999.

[324] V. V. Tolat. An analysis of kohonen's self-organizing maps using a set of energy functions. *Biological Cybernetics*, 64(2):155–164, 1990.

[325] J. Townshend, C. Justice, W. Li, C. Gurney, and J. McManus. Global land cover classification by remote sensing: present capabilities and future possibilities. *Remote Sensing of Environment*, 35:243–255, 1991.

[326] J. R. G. Townshend. Land cover. *Int. J. Remote Sensing*, 13:1319–1328, 1992.

[327] T. Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, 2002.

[328] B. C. K. Tso and P. M. Mather. Classification of multisource remote sensing imagery using a gentic algorithm and markov random fields. *IEEE Trans. Geosci. Remote Sensing*, 37(3):1255–1260, 1999.

[329] A. Ultsch and H. P. Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In *Proc. INNC'90, Int. Neural Network Conf.*, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer.

[330] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[331] M. M. Van Hulle. Topographic map formation by maximizing unconditional entropy: A plausible strategy for on-line unsupervised competitive learning and nonparametric density estimation. *IEEE Tr. Neural Networks*, 7:1299–1305, 1996.

[332] M. M. Van Hulle. Nonparametric density estimation and regression achieved with topographic maps maximizing the information-theoretic entropy of their outputs. *Biological Cybernetics*, 77:49–61, 1997.

[333] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[334] J. Vesanto. Som-based data visualization methods. *Intelligent Data Analysis*, 3:111–126, 1999.

[335] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[336] M. Vidyasagar. *Theory of Learning and Generalization, with Application to Neural Networks and Control Systems*. Springer, New York, 1997.

[337] T. Villmann, , and E. Merényi. Extensions and modifications of the som and its application in satellite remote sensoring processing. In H. Bothe and R. Rojas, editors, *Proceeding of the ICSC Symposia on Neural Computation (NC'2000) May 23-26, 2000 in Berlin, Germany*, 2000.

[338] T. Villmann and H.-U. Bauer. Applications of the growing self-organizing map. *Neurocomputing*, 21((1-3)):91–100, 1998.

[339] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz. Topology preservation in self-orfanizing feature map: exact definition and measurement. *IEEE Trans. on Neural Networks*, 8(2):256–266, 1997.

[340] T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15, 2002.

[341] von der Malsburg and J. D Willshaw. How to label nerve cells so that they can interconnect in an ordered fashion. *Proc. National Academy of Sciences USA*, 74:5176–5178, 1977.

[342] J. Walter and H. Ritter. Rapid learning with parameterized self-organizing maps. *Neurocomputing*, 12:131–153, 1996.

[343] L. Wang and J. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. Syst. Man and Cyberns.*, 22(6):1414–1427, 1992.

[344] Q. R. Wang and C. Y. Suen. Large tree classifier with heuristic search and global training. *IEEE Trans. Patt. Anal. and Machine Intell*, 9(1):91–102, 1987.

[345] J. Watada, H. Tanaka, and K. Asay. Fuzzy discriminant analysis in fuzzy groups. *Fuzzy Sets and Systems*, 19:261–271, 1986.

[346] P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of IEEE*, 78:1550–1560, 1990.

[347] B. Widrow and M. E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Convention Record*, pages 96–104, 1960.

[348] J. D Willshaw and von der Malsburg. How patterned neural connections can be set up by self-organization. *Proc. Royal Socoiety of London, Section B*, 194:431–445, 1996.

[349] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans Syst. Man and Cybern.*, 2:408–420, 1972.

[350] C.-C Wong and C.-C. Chen. A ga-based method for constructing fuzzy systems directly from numerical data. *IEEE Trans. Syst. Man Cyberns. B*, 30:904–911, 2000.

[351] Q. Wu, S. S. Iyengar, and M. Zhu. Web image retrieval using self-organizing feature map. *Journal of the American Society for Information Science and Technology*, 52(10):868–875, 2001.

[352] P. Xu, C.-H. Chang, and A. Paplinski. Self-organizing topological tree for on-line vector quantization and data clustering. *IEEE Trans. Syst. Man Cybern. :B*, 35(3):515–526, 2005.

[353] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3):645–678, 2005.

[354] M. Yacoub, F. Badran, and S. Thiria. A topological hierarchical clustering: Application to ocean color classification. In *Artificial Neural Networks-ICANN 2001, PROCEEDINGS*, pages 492–499, 2001.

[355] R. R. Yager and D. P. Filev. Approximate clustering by the mountain method. *IEEE Trans Syst. Man and Cyberns*, 24(8):1279–1283, 1994.

[356] E. Yair, K. Zager, and A. Gersho. Competitive learning and soft competition for vector quantizer design. *IEEE Trans. Signal Processing*, 40(2):394–309, 1992.

[357] T. Yamamoto. Vector quantization for image compression using circular structured self-organization feature map. In *IEEE International Conference on Image Processing*, volume 2, pages 443–446, 2001.

[358] H. Yang and C. Lee. A text mining approach on automatic generation of web directories and hierarchies. *Expert Systems with Application*, 27:645–663, 2004.

[359] H. C. Yang. Shape-based image retrieval by spatial topology distances. In *Proceedings of the ACM International Multimedia Conference and Exhibition*, pages 38–41, 2001.

[360] M.-S. Yang and C.-T. Chen. On strong consistency of fuzzy generalized nearest neighbor rule. *Fuzzy Sets and Systems*, 60:273–281, 1993.

[361] H. Yin. Data visualization and manifold mapping using visom. *Neural Networks*, 15:1005–1016, 2002.

[362] H. Yin. ViSOM-a novel method for multivariate data projection and structure visualization. *IEEE Transactions on Neural Networks*, 13(1):237–243, January 2002.

[363] H. Yin and N. M. Allinson. On the distribution and convergence of feature space in self-organising map. *Neural Computation*, 7:1178–1187, 1995.

[364] J. Yu. General c-means clustering model. *IEEE Trans. Pat. Anal. Mach. Intel.*, 27(8):1197–1211, 2005.

[365] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[366] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning: Parts 1, 2 and 3. *Information Sciences*, 8, 8 and 9:199–249, 301–357, 43–80, 1975.

[367] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[368] K. Zeger, J. Vaisey, and A. Gersho. Globally optimal vector quantizer design by stochastic relaxation. *IEEE Trans. Signal Processing*, 40(2):310–322, 1992.

[369] L. M. Zouhal and T. Denœux. An evidence theoretic k-nn rule with parameter optimization. *IEEE Trans. Syst. Man and Cybern: Part C*, 28(2):263–271, 1998.

[370] S. Zrehen. Analyzing kohonen maps with geometry. *Proc. Int. Conf. Artificial Neural Networks*, pages 609–612, 1993.

[371] J. M. Zurada. *Introduction to Artificial Neural Systems*. West Publ. Co., St. Paul, MN, 1992.

[372] J. M. Zurada, W. Jedruch, and M. Barski. *Neural Networks*. Polish Scientific Publishers, Warsaw, Poland, 1996.

List of Publications of the Author

1. A. Laha and N. R. Pal, Some novel classifiers designed using prototypes extracted by a new scheme based on Self-Organizing Feature Map, *IEEE Trans. on Syst. Man and Cybern: B*, Vol. 31, no. 6, pp. 881–890, 2001.

2. A. Laha and N. R. Pal, Dynamic generation of Prototypes with Self-Organizing Feature Maps for classifier design, *Pattern Recognition*, Vol. 34, no. 2, pp. 315–321, 2000.

3. A. Laha, N. R. Pal and B. Chanda, Design of Vector Quantizer for Image compression using Self-organizing Feature Map and surface fitting, *IEEE Trans. Image Processing*, Vol. 13, no. 10, pp. 1291–1303, 2004.

4. N. R. Pal, A. Laha and J. Das, Designing fuzzy rule based classifier using self-organizing feature map for analysis of multispectral satellite images, *International Journal of Remote Sensing*, Vol. 26, no. 10, pp. 2219–2240, 2005.

5. A. Laha, N. R. Pal and J. Das, Land cover classification using fuzzy rules and aggregation of contextual information through evidence theory, *IEEE Trans. Geosc. and Remote Sensing*, Vol. 44, no. 6, pp. 1633–1641, 2006.

6. N. R. Pal and A. Laha, A Multi-prototype classifier and its application to remotely sensed image analysis, *Australian Journal of Intelligent Information Processing*, Vol. 6, no. 2, pp. 110–118, 2000.

7. A. Laha, B. Chanda and N. R. Pal, Fast codebook searching in a SOM-based vector quantizer for image compression, *Image and Vision Computing (Communicated).*

8. A. Laha, An empirical study on the robustness of SOM in preserving topology with respect to link density, *Proceedings of 11th International Conference on Neural Information Processing (ICONIP 2004)*, Vol. LNCS 3316 (Springer-Verlag), pp. 142–149, 2004.

9. A. Laha and N. R. Pal, On different variants of Self-Organizing Feature Map and their properties, *Proceedings of the 1999 IEEE Hong Kong Symposium on Robotics and Controls*, Vol 1, pp. I-344–I-349, 1999.

10. N. R. Pal and A. Laha, Design of a Nearest-Prototype Classifier with Dynamically Generated Prototypes Using Self-Organizing Feature Maps, *Proceedings of International Conference on Neural Information Processing (ICONIP'99), Perth, Australia*, Vol. 2, pp. 746–751, 1999.

11. A. Laha, N. R. Pal and J. Das, Designing Prototype-based Classifiers and Their Application to Classification of Multispectral Satellite Images, *Proceedings of 6th International Conference on Soft Computing (IIZUKA2000)*, pp. 861–868, 2000.

12. A. Laha, N. R. Pal and J. Das, Satellite Image Analysis with Fuzzy Rules and Theory of Evidence, *Fuzzy Set Theory and its Mathematical Aspects and Applications, A. K. Srivastav (Ed.)*, Allied Pub. Pvt. Ltd, New Delhi, 2003.

13. A. Laha and J. Das, Fuzzy Rules and Evidence Theory for Satellite Image Analysis, *Proc. 5-th Intl. Conf. Advances in Pattern Recognition (ICAPR 2003)*, pp. 453–457, 2003.