1

Introduction

1.1 Introduction

Human beings are very adept in recognizing patterns, a task that they continually execute. In contrast, getting a machine to learn and recognize patterns can be extremely difficult and challenging. Machine recognition of patterns can be viewed as a twofold task – learning the invariant and common properties of a set of samples characterizing a class, and deciding that a new sample is a possible member of the class by noting that it has properties common to those of the set of samples. It is generally conjectured that animal brains process the sensory data obtained from different senses, apply rules that are embedded in the brain (which are usually thought to be learnt starting from birth) and recognize patterns. Pattern recognition deals with the theories and methodologies involved in making a machine recognize patterns which are characterized by numerical and nonnumerical attributes. The major hurdle in this task is that the functioning of the brain is much less understood. The mechanisms with which it stores huge amounts of information, processes them at lightning speeds and infers meaningful rules, and retrieves information as and when necessary have till now eluded the scientists. Attempts have therefore been made to develop algorithms and methodologies that are motivated by theoretical considerations and speculations about the working of the brain.

Objects in the real world are captured using some measurements on them. For example, a camera captures the scene in front of it as an array of pixels. Each pixel value may range from 0 to 255 if it is a gray-scale image. For color images, each pixel value will be a vector of, say, three values, indicating the intensities in the red, blue and green bands. Figure 1.1 shows an example of how a scene is converted to image data that can be stored in a computer for further processing. Similarly, a flower may be characterized by certain measurements taken on it. For example, the famous Iris flower data, described in Appendix B, measures the petal length, petal width, sepal length and sepal width of Iris flowers which belong to three classes — Setosa, Versicolor and Virginica. Thus a flower may be represented using a vector in $I\!R^4$ of the above

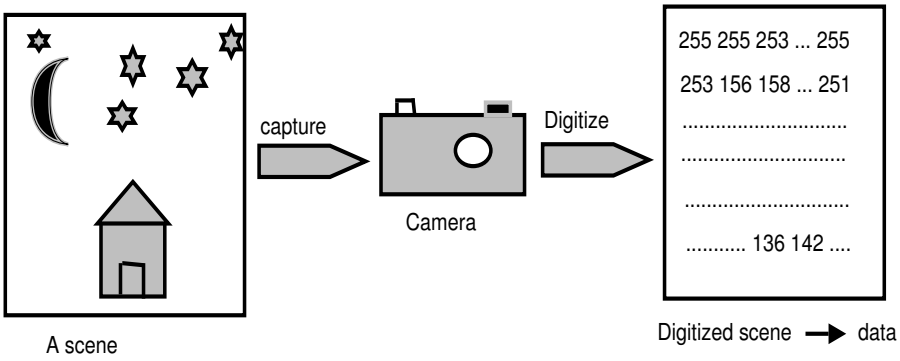

Fig. 1.1. Example of object representation

four measurement values, and it may be labelled by one of the three classes. In other words, a flower may be mapped to a point in a four-dimensional space, and this point may have an associated class label. The task of any pattern recognition system essentially involves taking some measurements of the objects to be recognized, thereby mapping them into points in some space. Thereafter, if the system has some labelled points then it has to infer some rules about the "belongingness" of points to the classes. Given an unknown point, the system will then have to decide about its classification, based on the rules that it has already learnt. On the other hand, if the system is not provided any labelled information, then its task may be to find some relationship among the unlabelled points that it has collected.

Machine learning is an area of artificial intelligence that deals with the task of making machines capable of learning from their environment. Evolution is probably the most powerful learning mechanism in existence. Over millions of years, different species have evolved in such a way that they have become more suited to their environment. In biological evolution, changes occur at the genetic level through crossover and mutation. These changes are propagated to future generations, where the process gets repeated. Usually, these changes are such that they provide some survival advantage to an individual with respect to the local environments. Changes that are disadvantageous tend to decrease in frequency, while those that are beneficial are more likely to be preserved because they aid survival.

Inspired by the powerful evolutionary learning mechanism, scientists attempted to design computer algorithms that mimic these processes. The result was a class of computer algorithms called evolutionary computation (EC). EC is a computing paradigm comprising population-based metaheuristic optimization techniques that are based on the principles of biological evolution. The essential components of EC are a strategy for representing or encoding a solution to the problem under consideration, a criterion for evaluating the

fitness or goodness of an encoded solution and a set of biologically inspired operators applied on the encoded solutions. Genetic algorithms (GAs) are the most well-known and widely used techniques in this computing paradigm. Other commonly known techniques are evolutionary strategies and genetic programming. Because of the robustness and effectiveness of GAs, they find widespread applications in various engineering and scientific communities like pattern recognition, image processing, bioinformatics, data mining, Web mining, mechanical engineering, telecommunications, mobile computing, VLSI design, and embedded and real-time systems.

In particular, many tasks involved in the process of recognizing a pattern need appropriate parameter selection and efficient search in complex and large spaces in order to attain optimal solutions. This makes the process not only computationally intensive, but also leads to a possibility of losing the exact solution. Therefore, the application of GAs for solving certain problems of pattern recognition that require optimization of computation requirements, and robust, fast and close approximate solution, seems appropriate and natural. Additionally, the existence of the proof that GAs will definitely provide the global optimal solution as the number of iterations goes to infinity [60] further strengthens the theoretical basis of its use in search problems.

The basic concepts related to machine recognition of patterns is first discussed in Sect. 1.2. The different approaches to pattern recognition are mentioned in Sect. 1.3. These are followed by brief descriptions of the connectionist approach in Sect. 1.4, genetic approach in Sect. 1.5, fuzzy set-theoretic approach in Sect. 1.6 and other hybrid approaches in Sect. 1.7. Section 1.8 mentions some real-life applications of pattern recognition and learning. Finally, a summary of the chapter and the scope of the book are mentioned in Sect. 1.9.

1.2 Machine Recognition of Patterns: Preliminaries

Pattern recognition and machine learning [131, 152, 313, 384, 458] form a major area of research and development activity that encompasses the processing of pictorial and other nonnumerical information obtained from the interaction between science, technology and society. A motivation for the spurt of activity in this field is the need for people to communicate with computing machines in their natural mode of communication. Another important motivation is that the scientists are also concerned with the idea of designing and making intelligent machines that can carry out certain tasks that we human beings do. The most salient outcome of these is the concept of future-generation computing systems.

Machine recognition of patterns, as mentioned before, can be viewed as a twofold task, comprising learning the invariant properties of a set of samples characterizing a class, and of deciding that a new sample is a possible member

of the class by noting that it has properties common to those of the set of samples. A typical pattern recognition system consists of three phases (Fig. 1.2). These are *data acquisition, feature extraction* and *classification*. In the data acquisition phase, depending on the environment within which the objects are to be classified, data are gathered using a set of sensors. These are then passed on to the feature extraction phase, where the dimensionality of the data is reduced by measuring and retaining only some characteristic features or properties. In a broader perspective, this stage significantly influences the entire recognition process. Finally, in the classification phase, the extracted features are passed on to the classifier that evaluates the incoming information and makes a final decision. This phase basically establishes a transformation between the features and the classes. Therefore pattern recognition can be described as a transformation from the measurement space M to the feature space F and finally to the decision space D, i.e.,

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

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

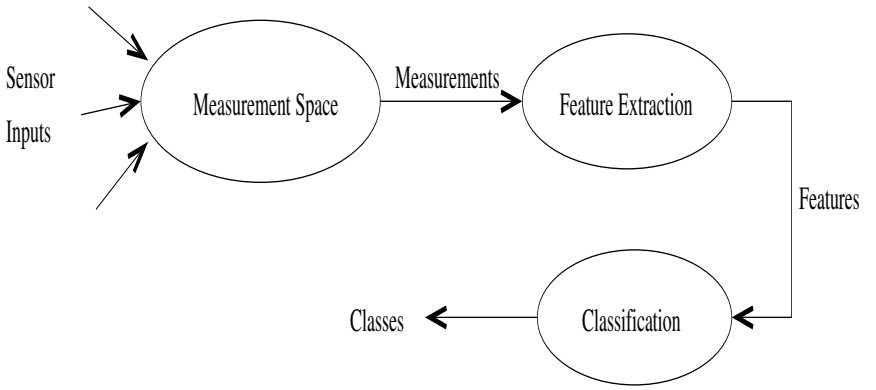

Fig. 1.2. A typical pattern recognition system

In the following sections we describe, in brief, some tasks commonly undertaken in typical pattern recognition systems. These are data acquisition, feature selection, and some classification and clustering techniques.

1.2.1 Data Acquisition

Pattern recognition techniques are applicable in a wide domain, where the data may be qualitative, quantitative or both; they may be numerical, linguistic, pictorial or any combination thereof. Generally, the data structures that

are used in pattern recognition systems are of two types: *object data vectors* and *relational data*. Object data, sets of numerical vectors of Q features, are represented in what follows as $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_t\}$, a set of t feature vectors in the Q-dimensional measurement space Ω_Y. The ith object, $i = 1, 2, \ldots, t$, observed in the process has vector Y_i as its numerical representation; y_{ij} is the jth ($j = 1, 2, \ldots, Q$) feature associated with the object i.

Relational data is a set of t^2 numerical relationships, say $\{r_{ii'}\}$, between pairs of objects. In other words, $r_{ii'}$ represents the extent to which object i and i' are related in the sense of some binary relationship ρ. If the objects that are pairwise related by ρ are called $O = \{o_1, o_2, \ldots, o_t\}$, then $\rho : O \times O \rightarrow \mathbb{R}$.

Sometimes, the data acquisition phase includes some preprocessing tasks like noise reduction, filtering, encoding and enhancement for extracting pattern vectors. For example, if the input pattern is an image, these preprocessing operations play a crucial role for extracting salient features for its recognition.

1.2.2 Feature Selection

Feature selection is one of the major tasks in any automatic pattern recognition system. The main objective of *feature selection* [47, 127] is to retain the optimum salient characteristics necessary for the recognition process and to reduce the dimensionality of the measurement space Ω_Y so that effective and easily computable algorithms can be devised for efficient classification. The problem of feature selection has two aspects: the formulation of a suitable criterion to evaluate the goodness of a feature and the selection of optimal subsets from the available features.

The major mathematical measures so far devised for the estimation of feature quality are mostly statistical in nature, and can be broadly classified into two categories:

- feature selection in the measurement space, and
- feature extraction in the transformed space.

The techniques in the first category generally reduce the dimensionality of the feature set by discarding redundant information carrying features. On the other hand, those in the second category utilize all the information contained in the pattern vectors, and map a higher-dimensional pattern vector to a lower-dimensional one.

Feature selection is the process of selecting a map of the form $X = f(Y)$, by which a sample Y $(=y_1, y_2, \ldots, y_Q)$ in a Q-dimensional measurement space Ω_Y is transformed into a point X $(=x_1, x_2, \ldots, x_N)$ in an N-dimensional feature space Ω_X, where $N < Q$.

The pioneering research on feature selection mostly deals with statistical tools. Later, the thrust of the research shifted to the development of various other approaches to feature selection, including fuzzy, neural and genetic approaches [133, 342, 351, 401, 434, 465].

1.2.3 Classification

The problem of classification basically establishes a transformation $F \rightarrow D$ between the features and the classes (Fig. 1.2). In other words, it provides a partitioning of the feature space into regions, one region for each category of input. That is, it attempts to assign every data point in the entire feature space to one of the possible (say K) classes. Different forms of transformation can be a Bayesian rule of computing a posteriori class probability, nearest neighbor rule, linear discriminant functions, perceptron rule, nearest prototype rule, etc. [13, 127, 130, 153, 155, 161, 172, 364, 461]. Classifiers are usually, but not always, designed with labelled data, in which case these problems are sometimes referred to as *supervised classification* (where the parameters of a classifier function D are learned). Some common examples of the *supervised* pattern classification techniques are the nearest neighbor rule, Bayes' maximum likelihood classifier and the perceptron rule. Some of the commonly used classification (supervised) methods, which have been used in different chapters of this book for the purpose of comparison, are now described here.

Let $C_1, C_2, \ldots, C_i, \ldots, C_K$ be the K possible classes in an N-dimensional feature space. Let $\mathbf{x} = [x_1, x_2, \ldots, x_j, \ldots, x_N]'$ be an unknown pattern vector. In a deterministic classification approach, it is assumed that there exists only one unambiguous pattern class corresponding to each of the unknown pattern vectors. If the pattern $\mathbf{x}$ is a member of class C_i, the *discriminant (decision) function* $D_i(\mathbf{x})$ associated with the class C_i, $i = 1, 2, \ldots, K$, must then possess the largest value. In other words, a classificatory decision would be as follows:

$$\text{Decide} \quad \mathbf{x} \in C_i, \qquad \text{if} \quad D_i(\mathbf{x}) > D_l(\mathbf{x}), \tag{1.1}$$

$(\forall\ i, l)$ in $(1, \ldots, K)$ and $i \neq l$. Ties are resolved arbitrarily. The decision boundary in the feature space between regions associated with the classes C_i and C_l would be governed by the expression

$$D_i(\mathbf{x}) - D_l(\mathbf{x}) = 0. \tag{1.2}$$

Many different forms satisfying Eq. (1.2) can be selected for $D_i(\mathbf{x})$. The functions that are often used are linear discriminant functions, quadratic discriminant functions and polynomial discriminant functions. One commonly used piecewise linear discriminant function, which is used in the nearest neighbor (NN) classifier, involves distance as a similarity measure for classification. This is described below.

NN rule

Let us consider a set of n patterns of known classification $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where it is assumed that each pattern belongs to one of the classes $C_1, C_2, \ldots, C_i, \ldots, C_K$. The NN classification rule [130, 155, 461] then assigns

a pattern $\mathbf{x}$ of unknown classification to the class of its nearest neighbor, where $\mathbf{x}_i \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is defined to be the nearest neighbor of $\mathbf{x}$ if

$$D(\mathbf{x}_i, \mathbf{x}) = \min_l \{D(\mathbf{x}_l, \mathbf{x})\}, \qquad l = 1, 2, \ldots, n. \tag{1.3}$$

Here D is any distance measure definable over the pattern space.

Since the aforesaid scheme employs the class label of only the nearest neighbor to $\mathbf{x}$, this is known as the 1-NN rule. If k neighbors are considered for classification, then the scheme is termed as the k-NN rule. The k-NN rule assigns a pattern $\mathbf{x}$ of unknown classification to class C_i if the majority of the k nearest neighbors belongs to class C_i. The details on the k-NN rule along with the probability of error are available in [130, 155, 461].

In most of the practical problems, the features are usually noisy and the classes in the feature space are overlapping. In order to model such systems, the feature values $x_1, x_2, \ldots, x_j, \ldots, x_N$ are considered as random values in the probabilistic approach. The most commonly used classifier in such probabilistic systems is the *Bayes' maximum likelihood classifier* [9, 461], which is now described.

Bayes' Maximum Likelihood Classifier

Let P_i denote the a priori probability and $p_i(\mathbf{x})$ denote the class conditional density corresponding to the class C_i $(i = 1, 2, \ldots, K)$. If the classifier decides $\mathbf{x}$ to be from the class C_i, when it actually comes from C_l, it incurs a loss equal to L_{li}. The expected loss (also called the conditional average loss or risk) incurred in assigning an observation $\mathbf{x}$ to the class C_i is given by

$$r_i(\mathbf{x}) = \sum_{l=1}^{K} L_{li}\ p(C_l/\mathbf{x}), \tag{1.4}$$

where $p(C_l/\mathbf{x})$ represents the probability that $\mathbf{x}$ is from C_l. Using Bayes' formula, Eq. (1.4) can be written as

$$r_i(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{l=1}^{K} L_{li}\ p_l(\mathbf{x}) P_l, \tag{1.5}$$

where

$$p(\mathbf{x}) = \sum_{l=1}^{K} p_l(\mathbf{x}) P_l.$$

The pattern $\mathbf{x}$ is assigned to the class with the smallest expected loss. The classifier which minimizes the total expected loss is called the *Bayes' classifier*.

Let us assume that the loss (L_{li}) is zero for a correct decision and greater than zero but the same for all erroneous decisions. In such situations, the expected loss, Eq. (1.5), becomes

$$r_i(\mathbf{x}) = 1 - \frac{P_i p_i(\mathbf{x})}{p(\mathbf{x})}. \tag{1.6}$$

Since $p(\mathbf{x})$ is not dependent upon the class, the Bayes' decision rule is nothing but the implementation of the decision functions

$$D_i(\mathbf{x}) = P_i p_i(\mathbf{x}), \qquad i = 1, 2, \ldots, K, \tag{1.7}$$

where a pattern $\mathbf{x}$ is assigned to class C_i if $D_i(\mathbf{x}) > D_l(\mathbf{x})$, $\forall l \neq i$. This decision rule provides the minimum probability of error. It is to be noted that if the a priori probabilities and the class conditional densities are estimated from a given data set, and the Bayes' decision rule is implemented using these estimated values (which may be different from the actual values), then the resulting classifier is called the *Bayes' maximum likelihood classifier*.

Assuming normal (Gaussian) distribution of patterns, with mean vector μ_i and covariance matrix $\sum_i$, the Gaussian density $p_i(\mathbf{x})$ may be written as

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp[\, -\frac{1}{2}(\mathbf{x} - \mu_i)' \Sigma_i^{-1} (\mathbf{x} - \mu_i)], \tag{1.8}$$
$$i = 1, 2, \ldots, K.$$

Then, $D_i(\mathbf{x})$ becomes (taking the natural logarithm)

$$D_i(\mathbf{x}) = \ln P_i - \tfrac{1}{2} \ln |\Sigma_i| \, -\frac{1}{2}(\mathbf{x} - \mu_i)' \Sigma_i^{-1} (\mathbf{x} - \mu_i), \tag{1.9}$$
$$i = 1, 2, \ldots, K.$$

Note that the decision functions in Eq. (1.9) are hyperquadrics, since no terms higher than the second degree in the components of $\mathbf{x}$ appear in it. It can thus be stated that the Bayes' maximum likelihood classifier for normal distribution of patterns provides a second-order decision surface between each pair of pattern classes. An important point to be mentioned here is that if the pattern classes are truly characterized by normal densities, on an average, no other surface can yield better results. In fact, the Bayes' classifier designed over known probability distribution functions provides, on an average, the best performance for data sets which are drawn according to the distribution. In such cases, no other classifier can provide better performance on an average, because the Bayes' classifier gives the minimum probability of misclassification over all decision rules.

1.2.4 Clustering

As already mentioned, when only unlabelled data are available, then the classification methodology adopted is known as *unsupervised classification* or *clustering*. Many clustering algorithms are used as precursors to the design of a classifier in such situations. In clustering [8, 127, 187, 212, 461] a set of patterns, usually vectors in a multidimensional space, are grouped into clusters

in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. Clustering in N-dimensional Euclidean space $\mathbb{R}^N$ is the process of partitioning a given set of n points into a number, say K, of groups (or clusters) based on some similarity/dissimilarity metric. Let the set of n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be represented by the set S, and the K clusters be represented by $C_1, C_2, \ldots, C_K$. Then

$$\begin{array}{l} C_i \neq \emptyset, \qquad for\ i = 1, \ldots, K, \\ C_i \bigcap C_j = \emptyset,\ for\ i = 1, \ldots, K, \quad j = 1, \ldots, K,\ \text{and}\ i \neq j,\ \text{and} \\ \bigcup_{i=1}^{K} C_i = S. \end{array}$$

For this it is necessary to first define a measure of similarity which will establish a rule for assigning patterns to the domain of a particular cluster center. One such measure of similarity may be the Euclidean distance $\mathbf{D}$ between two patterns $\mathbf{x}$ and $\mathbf{z}$ defined by $\mathbf{D} = \|\mathbf{x} - \mathbf{z}\|$. The smaller the distance between $\mathbf{x}$ and $\mathbf{z}$, the greater is the similarity between the two, and vice versa.

Clustering techniques have been broadly categorized into partitional and hierarchical methods. One commonly used clustering technique in the partitional category is the K-means algorithm [461], where the number of clusters K is assumed to be known a priori. The K-means algorithm has been widely recommended after extensive studies dealing with comparative analysis of different clustering methods [129, 155, 307]. Popular techniques in the hierarchical category are the single linkage, complete linkage and average linkage algorithms. A distinguishing feature of the hierarchical clustering techniques from the partitional ones is that while the former provide a valid clustering of the data at each iteration of the algorithm, the latter do not do so (they provide a valid clustering only on termination of the algorithm). Minimal spanning tree-based graph-theoretic clustering techniques are also quite popular in the pattern recognition community. A description of some existing techniques like the K-means and single linkage algorithms are described in Chap. 8 of this book.

1.3 Different Approaches

Pattern recognition, by its nature, admits many approaches, sometimes complementary, sometimes competing, to the approximate solution of a given problem. These include the decision-theoretic approach (both deterministic and probabilistic), syntactic approach, connectionist approach, genetic approach and fuzzy set-theoretic approach.

In the decision-theoretic approach [127, 131, 155, 488], once a pattern is transformed, through feature selection, to a vector in the feature space, its characteristics are expressed only by a set of numerical values. Classification can be done using deterministic or probabilistic techniques. Classification techniques described in Sect. 1.2.3 fall under the decision-theoretic approach.

On the other hand, when a pattern is rich in structural information (e.g., picture recognition, character recognition, scene analysis), i.e., the structural information plays an important role in describing and recognizing the patterns, syntactic approaches, which deal with representation of structures via sentences, grammars and automata, are usually adopted. In the syntactic method [153, 461], the ability to select and classify the simple pattern primitives and their relationships represented by the composition operations is a vital criterion for making a system effective. Since the techniques of composition of primitives into patterns are usually governed by the formal language theory, the approach is often referred to as the linguistic approach. An introduction to a variety of approaches based on this idea can be found in [153].

For any pattern recognition system, one desires to achieve robustness with respect to random noise and failure of components. Another important requirement is to have output in real time. It is also desirable for the system to be adaptive to changes in the environment. Moreover, a system can be made artificially intelligent if it is able to emulate some aspects of the human processing system. Connectionist (neural network-based) approaches [250, 406] to pattern recognition [64, 420] are attempts to achieve these goals. An artificial neural network (ANN) is a system composed of several simple processing elements that operate in parallel and that are connected by some interconnection structure. The behavior of a neural network is determined by its structure, connection weights and the function that is implemented at each node. Some ANN models are closely related to statistical pattern recognition [213, 223], with each approach benefiting from the theories/applications developed in the other. For example, many classical pattern recognition algorithms like principal component analysis, K-means clustering, etc., can be mapped to ANNs for faster hardware implementation. Similarly, ANNs derive benefit from well-known results in statistical pattern recognition, such as Bayes' decision theory, and nearest neighbor rules.

A pattern recognition problem can be mapped to one of search and optimization of certain criteria (e.g., search for a suitable decision boundary that provides the maximum abstraction of the data, search for a suitable subset of features that provides the most compact representation of the data). Often the search spaces in such problems are extremely huge, complex and multimodal. Hence genetic approaches to pattern recognition have been attempted by several researchers for constructing more accurate and efficient systems.

Methods and technologies developed under the above-mentioned categories may, again, be fuzzy set-theoretic in order to handle uncertainties arising from vague, incomplete, linguistic, overlapping patterns, at various stages of pattern recognition systems. This approach is developed based on the realization that a pattern may belong to more than one class, with varying degrees of class membership. Accordingly, fuzzy decision-theoretic, fuzzy syntactic and fuzzy neural approaches are developed [57, 59, 224, 225, 351, 368].

Some of the classification methods under the aforesaid approaches are briefly described, in the following sections, along with their relevance and characteristic features.

1.4 Connectionist Approach: Relevance and Features

Neural networks can be formally defined as *massively parallel interconnections of simple (usually adaptive) processing elements that interact with objects of the real world in a manner similar to biological systems.* Their origin can be traced to the work of Hebb [189], where a local learning rule is proposed. The benefit of neural nets lies in the high computation rate provided by their inherent massive parallelism. This allows real-time processing of huge data sets with proper hardware backing. All information is stored distributed among the various connection weights. The redundancy of interconnections produces a high degree of robustness, resulting in a *graceful degradation* of performance in the case of noise or damage to a few nodes or links. The characterizing features of ANNs are

- simple processing units
- high degree of interconnection
- simple scalar messages
- adaptive interaction within elements

In [353] the relevance of neural networks is summarized as follows: Neural networks are natural classifiers having resistance to noise, tolerance to distorted images or patterns (or the ability to generalize), superior ability to recognize partially occluded or degraded images or overlapping pattern classes or classes with highly nonlinear boundaries, and potential for parallel processing. Neural network models have been studied for many years with the hope of achieving human-like performance (artificially), particularly in the field of pattern recognition, by capturing the key ingredients responsible for the remarkable capabilities of the human nervous system. Note that these models are extreme simplifications of the actual human nervous system.

The perceptron [115, 188, 357, 406] is the earliest development in this direction. It may be broadly categorized into two classes: *single-layer perceptrons* and *multilayer perceptrons.* The single-layer perceptron, given two classes of patterns, attempts to find a linear decision boundary between them. If the classes are linearly separable, the perceptron algorithm is guaranteed to find a separating hyperplane in a finite number of steps. However, as shown by Minsky and Papert [312], if the pattern space is not linearly separable, the perceptron algorithm will not converge. In fact, it is shown to fail for the simple XOR problem, which is linearly inseparable.

This limitation was overcome by Rumelhart, Hinton and Williams [405], who developed the generalized delta rule for training of multilayer perceptrons

(MLP). The multilayer perceptron, known for discriminating nonlinearly separable classes as well, has since then been widely studied and applied in various fields [279]. Below we briefly explain the working principle of an MLP.

Multilayer Perceptron (MLP)

An MLP [115, 188, 357] consists of several layers of simple neurons with full connectivity existing between neurons of adjacent layers. Figure 1.3 shows an example of a three-layer MLP which consists of an input layer (*layer 1*), one hidden layer (*layer 2*) and an output layer (*layer 3*).

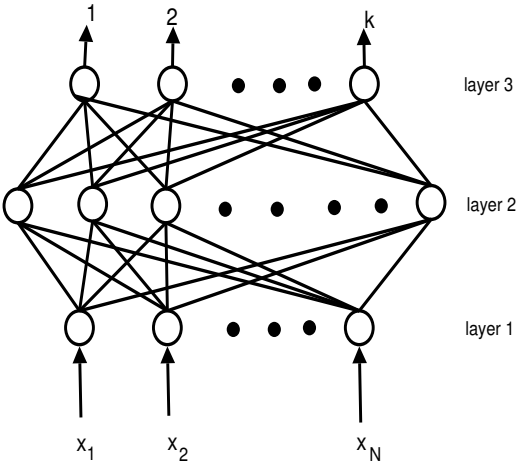

Fig. 1.3. Multilayer perceptron

The neurons in the input layer serve the purpose of fanning out the input values to the neurons of *layer 2*. Let

$$w_{ji}^{(l)}, \qquad l = 2,\ 3 \tag{1.10}$$

represent the connection weight on the link from the ith neuron in layer $l-1$ to the jth neuron in layer l. Let $\theta_j^{(l)}$ represent the threshold of the jth neuron in layer l. The total input, $x_j^{(l)}$, received by the jth neuron in layer l is given by

$$x_j^{(l)} = \sum_i y_i^{(l-1)}\ w_{ji}^{(l)} + \theta_j^{(l)}, \tag{1.11}$$

where $y_i^{(l-1)}$ is the output of the ith neuron in layer $l-1$. For the input layer

$$y_i^{(1)} = x_i, \tag{1.12}$$

where x_i is the ith component of the input vector. For the other layers

$$y_i^{(l)} = f(x_i^{(l)}) \qquad l = 2,\ 3. \tag{1.13}$$

Several functional forms like threshold logic, hard limiter and sigmoid can be used for $f(.)$.

There are several algorithms for training the network in order to learn the connection weights and the thresholds from a given training data set. Back-propagation (BP) is one such learning algorithm, where the least mean square error of the network output is computed, and this is propagated in a top-down manner (i.e., from the output side) in order to update the weights. The error is computed as the difference between the actual and the desired output when a known input pattern is presented to the network. A gradient descent method along the error surface is used in BP. More information on MLP and other neural methods are available in [115, 188, 279, 357].

1.5 Genetic Approach: Relevance and Features

Genetic algorithms (GAs) [19, 164, 195, 475], first proposed by Holland, are a class of computational models that mimic the principles of natural evolution for performing search and optimization. They use two important features from natural evolution: handing down of information from one generation to another, or *inheritance*, and competition for survival, or *survival of the fittest*. The main advantages of GAs that make them suitable for solving real-life problems are

- They are adaptive.
- They possess inherent parallelism.
- They are efficient for solving complex problems where the main focus is on obtaining good, not necessarily the best, solutions quickly.
- They are easy to parallelize, without much communication overhead.

GAs are particularly suited for applications involving search and optimization, where the space is huge, complex and multimodal, and the need for finding the exact optimal solution is not all important. Several tasks in the domain of pattern recognition meet these requirements. For example, consider the problem of clustering n observations into m groups such that some clustering measure is optimized to the largest possible extent. The number of possibilities is $\frac{1}{m!}\sum_{k=0}^{k=m}(-1)^{m-k}\binom{m}{k}k^n$. This number becomes immensely large even for small problems. It is expected that application of an effective search strategy like GAs is likely to provide significantly superior performance compared to schemes that are ad hoc or mostly local in nature.

Moreover, in many pattern recognition problems, one is on the lookout for an acceptable good solution rather than the exact optimal one. Therefore, using genetic algorithms for pattern recognition problems that need optimization

of computation requirements, and robust, fast, and close approximate solutions appear to be natural and appropriate. The following characteristics of GAs make them adequate for use in pattern recognition problems:

(a) Because of implicit parallelism [164], GAs can search huge spaces in less time.
(b) Being stochastic in nature they have the ability to come out of local optima.
(c) They make no assumptions (e.g., differentiability requirement) about the search space, except that it is bounded.

Note that, unlike ANNs, which were developed keeping pattern classification tasks in mind, GAs were developed for the purpose of optimizing a given criterion. However, another way to look at classification is as a problem of searching for the best, or optimum, distinguishing boundary between the different classes. Use of GAs for performing this search will lead to the development of a nonparametric classifier that is applicable to a wide variety of data sets.

In addition to pattern classification and feature selection [160, 356, 434], GAs find widespread applications in solving different problems in image processing and scene recognition [14, 193, 347, 424], rule generation and classifier systems [65, 207, 208, 209, 210, 216], neural network design [66, 185, 290, 316, 321, 346, 407, 417, 491], scheduling problems [21, 92, 249, 277, 278, 390], VLSI design [85, 108, 302, 480], path planning [91, 109], the travelling salesman problem [176, 196, 218, 306, 333], graph coloring [113], and numerical optimization [309].

1.6 Fuzzy Set-Theoretic Approach: Relevance and Features

Uncertainty can arise either implicitly or explicitly in each and every phase of a pattern recognition system. It results from the incomplete, imprecise or ambiguous input information, the ill-defined and/or overlapping boundaries among the classes or regions, and the indefiniteness in defining/extracting features and relations among them. Any decision taken at a particular level will have an impact on all other higher-level activities. It is therefore required for a pattern recognition system to have sufficient provision for representing these uncertainties involved at every stage, so that the ultimate output of the system can be obtained with the least uncertainty.

Fuzzy sets were introduced in 1965 by Zadeh [509] as a way to represent vagueness in everyday life. Since this theory is a generalization of the classical set theory, it has greater flexibility to capture various aspects of incompleteness, impreciseness or imperfection in information about a situation.

The relevance of fuzzy set theory in the realm of pattern recognition [57, 59, 224, 225, 351, 352, 367, 368] is adequately justified in

- representing input patterns as an array of membership values denoting the degree of possession of certain properties
- representing linguistically phrased input features for processing
- providing an estimate (representation) of missing information in terms of membership values
- representing multiclass membership of ambiguous patterns and in generating rules and inferences in linguistic form
- extracting ill-defined image regions, primitives, and properties and describing relations among them as fuzzy subsets

1.7 Other Approaches

In the previous sections we have described, in addition to some conventional approaches to pattern recognition, three modern approaches based on tools like neural networks, genetic algorithms and fuzzy logic. These three tools are essential components of the soft computing paradigm. Soft computing is a relatively recent term that is used to denote a consortium of methodologies that works synergistically and provides in one form or another flexible information processing capability for handling real-life ambiguous situations. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to achieve tractability, robustness and low-cost solutions. The guiding principle is to devise methods of computation that lead to an acceptable solution at low cost by seeking for an approximate solution to an intractable problem.

There have been several attempts over the last decade to derive hybrid methods by combining the merits of several existing techniques. Moreover, other pattern recognition paradigms like support vector machines, case-based reasoning and rough sets are also being extensively studied and developed. Let us now discuss some such attempts in this section.

An integration of neural network and fuzzy theory, commonly known as neuro-fuzzy computing, is one such hybrid paradigm [215, 353] that enables one to build more intelligent decision-making systems. In the context of pattern recognition, it incorporates the capability of neural networks in generating required decision boundaries, which may be linearly nonseparable, along with the capability of fuzzy logic for handling uncertain, imprecise, linguistic and incomplete data.

Application of rule-based systems in pattern recognition has also gained popularity in the recent past. By modelling the rules and facts in terms of fuzzy sets, it is possible to make interfaces using the concept of approximate reasoning [45, 128]. An approach to classifying unknown patterns by combining the k-NN rule with the Dempster–Shafer theory of evidence [427] has been formulated in [125]. In this regard, GAs have also been used, under the genetic-fuzzy integration, to evolve the fuzzy membership functions as well as the rule base [192, 197, 207, 208, 370, 508]. Tettamanzi [457] describes a

way in which the features of fuzzy sets can be used for improving the performance of genetic algorithms. A new method of fuzzy coding making use of linguistic terms modelled as a collection of fuzzy sets is proposed in [372]. This fuzzy coding establishes a relevant level of information granularity and provides some specific search orientation (guidance). However, calculation of the fitness function now needs to be performed prudently. Under genetic-neural integration, there have been several attempts at using GAs to train a given network, determine the network topology, or perform both determining and training an appropriate network for a given problem. Branke [71] provides an overview of some such attempts.

Integration of neural networks, GAs and fuzzy logic have also been studied in some systems [305, 346, 389, 503]. Some other approaches for pattern recognition are based on case-based reasoning [349], support vector machines [105], rough sets [365] and their hybridizations.

1.8 Applications of Pattern Recognition and Learning

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

- *medicine:* medical diagnosis, image analysis, disease classification
- *computation biology:* gene identification, protein modelling, rational drug design
- *natural resource study and estimation:* agriculture, forestry, geology, the environment
- *human-machine communication:* automatic speech recognition and understanding, natural language processing, image processing, script recognition, signature analysis
- *biometry:* face recognition, gesture recognition
- *vehicular:* automobile, airplane, train and boat controllers
- *defense:* automatic target recognition, guidance and control
- *police and detective:* crime and criminal detection from analysis of speech, handwriting, fingerprints, photographs
- *remote sensing:* detection of manmade objects and estimation of natural resources
- *industry:* CAD, CAM, product testing and assembly, inspection, quality control
- *domestic systems:* appliances
- *space science:* Mars rover control, camera tracking
- *investment analysis:* predicting the movement of stocks, currencies, etc., from historical data

1.9 Summary and Scope of the Book

Pattern recognition and learning are two important issues that need to be addressed for making machines that can emulate some human behavior. In this chapter, an introduction to pattern recognition has been provided. The different tasks are described briefly. Various approaches to pattern recognition are mentioned, along with their relevance.

In the past, several computational algorithms have been developed that are based on the principles of biological systems. Effectiveness of integrating these computing techniques with classical pattern recognition and learning tasks has time and again been established by various researchers. GAs, an efficient search and optimization tool based on the mechanism of natural genetics and evolution, are such an important and relatively recent computing paradigm. Remaining chapters of this book describe different methodologies, theories, and algorithms, developed with various real-life applications using different variants of GAs for efficient pattern recognition and learning in both supervised and unsupervised modes.

Chapter 2 provides a detailed discussion on GAs. The different steps of GAs are described followed by the derivation of the schema theorem. The proof of convergence of GAs to the optimal string under limiting conditions is stated. Some implementation issues and various genetic operators are mentioned. A discussion on multiobjective GAs, a relatively recent development in this field, is provided in a part of this chapter. Finally, different applications of GAs are mentioned.

Chapters 3–7 deal with several single and multiobjective GA-based classifiers. Detailed comparative results on artificial and real-life data sets are provided. The basic classification principle used in the genetic classifiers is the approximation of the class boundaries using a fixed or variable number of hyperplanes (as well as higher-order surfaces) such that one or more objective criteria are optimized. While Chap. 3 describes a *GA-classifier* using a fixed number of hyperplanes (and hence, fixed chromosome length), Chap. 5 considers a variable number of hyperplanes (and hence, variable chromosome length) for designing the *VGA-classifier*. The latter facilitates better approximation of the decision boundary with less a priori information. Theoretical analysis of the two genetic classifiers are provided in Chaps. 4 and 5, respectively. A new model of GA, viz., GA with chromosome differentiation, keeping analogy with the sexual differentiation in nature, is described in Chap. 6 for reducing the computation time and improving performance. Its incorporation in the *VGA-classifier* to provide the *VGACD-classifier* is described, along with a real-life application for pixel classificaton in remote sensing imagery. In Chap. 7, the classification problem is modelled as one of multiobjective optimization, and a multiobjective genetic classifier, *CEMOGA-classifier*, is discussed. Different quantitative indices are defined to measure the quality of Pareto front solutions.

The application of genetic algorithms for clustering, both crisp and fuzzy, is demonstrated in Chap. 8. The number of clusters can be fixed or variable, thereby necessitating the use of both fixed- and variable-length genetic representation schemes.

Bioinformatics and Web intelligence are two upcoming areas where the use of genetic learning holds promise. Some studies have already been carried out that demonstrate that the integration of the search capability of GAs in these domains leads to the production of more effective and accurate solutions, more efficiently. Chapters 9 and 10 are devoted to these two important application areas of GAs.

In Chap. 9, the basic concepts of bioinformatics are first explained, followed by a listing of some of the tasks in bioinformatics. The relevance of GAs in bioinformatics is discussed. Thereafter, a detailed description of the bioinformatics tasks is provided along with the applications of GAs for solving them. Some experimental results demonstrating the effectiveness of GAs for designing small ligands are also provided.

Finally, Chap. 10 starts with an introduction to Web intelligence and mining, and the various components and tasks in Web mining. The existing limitations and challenges in this regard are mentioned. Subsequently, the different applications of genetic search for solving Web mining-related problems are discussed.