In memory of my Father
and in memory of my mathematics teacher –
who was like a real father to me at school.

Foreword

Automatic (machine) recognition, description, classification, and groupings of patterns are important problems in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, and remote sensing. Given a pattern, its recognition/classification may consist of one of the following two tasks: (1) supervised classification (also called discriminant analysis); the input pattern is assigned to one of several predefined classes, (2) unsupervised classification (also called clustering); no pattern classes are defined *a priori* and patterns are grouped into clusters based on their similarity. Interest in the area of pattern recognition has been renewed recently due to emerging applications which are not only challenging but also computationally more demanding (e.g., bioinformatics, data mining, document classification, and multimedia database retrieval).

Among the various frameworks in which pattern recognition has been traditionally formulated, the statistical approach has been most intensively studied and used in practice. More recently, neural network techniques and methods imported from statistical learning theory have received increased attention. Neural networks and statistical pattern recognition are two closely related disciplines which share several common research issues. Neural networks have not only provided a variety of novel or supplementary approaches for pattern recognition tasks, but have also offered architectures on which many well-known statistical pattern recognition algorithms can be mapped for efficient (hardware) implementation. On the other hand, neural networks can derive benefit from some well-known results in statistical pattern recognition. Issues related to the training and test sample sizes, feature space dimensionality, error rate estimation, and the discriminatory power of different classifiers have been extensively studied in the statistical pattern recognition literature. It often appears that some of the neural network researchers attempting to solve pattern recognition problems are not aware of these results.

Professor Raudys' book is a timely addition to the literature on pattern recognition. Professor Raudys is eminently qualified to write a monograph which presents a balanced view of classifier design and promotes an integration of statistical pattern recognition and neural network approaches. Even though his early work, published in Russian, was not easily accessible to the pattern recognition community, he is now well-known and recognised for his early contributions on the topic of "curse of dimensionality" and its practical implications in designing a pattern recognition system. His intuition and knowledge of the performance of various classifiers has enabled him to show the interrelationships among them. The book contains detailed descriptions of various classifiers which, to my knowledge, are not readily available in other textbooks. In addition to deriving analytical results concerning the relationship between sample size, dimensionality and model parameters, he also reports simulations results. These results are not available in other well-known textbooks on pattern recognition and neural networks.

In most pattern recognition books and review papers, it is the statistical approach which is used to compare and analyse classification and regression methods based on neural networks. Šarūnas Raudys was the first to show that ANN based classifiers and regression evolve into, and realise, a number of popular statistical pattern recognition methods. These findings, together with his earlier results, enable him to propose a way to utilise positive attributes of both approaches simultaneously in classifier design. His extensive coverage of the curse of dimensionality and related topics, together with a new approach to introduce statistical methodology into the neural networks design process, constitute the novelty of this book.

In summary, this book is an excellent addition to the literature on statistical pattern recognition and neural networks. It will serve as a valuable reference to other excellent books by Duda, Hart, and Stork, Ripley, Bishop and Haykin.

September 19th, 2000, East Lansing, Michigan Anil K. Jain

Preface

– Les hommes de chez toi, dit le petit prince,
cultivent cinq mille roses dans un même jardin...
et ils n'y trouvent pas ce qu'ils cherchent...
– Ils ne le trouvent pas, répondis-je...
Et cependant ce qu'ils cherchent pourrait être
trouvé dans une seule rose ou un peu d'eau...
– Bien sûr , répondis-je.
Et le petit prince ajouta:
– Mais les yeux sont aveugles. Il faut chercher
avec le coeur.

Antoine de Saint-Exupery "Le Petit Prince",
Chapitre XXV

In his book Antoine de Saint-Exupery wrote: "... In your country, people plant five thousand roses in the same garden ... and they do not find what they are searching for. Meanwhile, they could find everything they are seeking for in a single rose, or in a drop of water... However, eyes are blind. You have to seek by your heart." I am fond of Antoine de Saint Exupery. I like his books and I believe in these words.

When, after 25 years of research work in multivariate statistical analysis and statistical pattern recognition, I became interested in Artificial Neural Networks (ANN), I remembered de Saint-Exupery's words about the single rose. Instead of investigating multilayer perceptrons with thousands of neurones, I at first began to use statistical methods to analyse a single neurone – a single layer perceptron (SLP) and plain back-propagation (BP) training algorithm. The SLP and BP training are simplified mathematical models of complex information processing phenomena that take place in nature. I discovered that a single neurone can explain much about complex brain-learning behaviour.

After several years of research work, I learned that during the training phase, the single layer perceptron's weights are increasing and, therefore, the statistical properties of the cost function that is minimised during the training process are also changing. In its dynamical evolution, the SLP classifier can actually become one of several statistical classifiers that differ in their complexity.

At first the SLP classifier behaves as a simple Euclidean distance classifier. In this situation each pattern class is characterised by a sample mean vector. In further training, the SLP classifier begins to evaluate correlations and variances of features and almost becomes the standard linear Fisher classifier. Further, the SLP begins to behave as a robust classifier that ignores atypical training patterns. Later on in the training phase the SLP behaves as a classifier that minimizes the number of incorrectly classified training patterns. If there are no training errors, the SLP maximizes the margin between the decision boundary and the closest training-pattern vectors. Thus, the decision boundary is halfway between the training patterns and behaves as a support vector classifier where a fixed number of training patterns determine a position of the decision boundary. Thus, the SLP classifier begins as the simplest possible statistical classifier designed under the assumption of Gaussian pattern classes and ends as a complex classifier that can perform well with non-Gaussian pattern classes.

One more interesting peculiarity of the SLP classifier is that the performance (the generalization error) of the perceptron depends on the initial conditions. If the starting perceptron weight vector is almost optimal, the SLP classifier initially contains much useful information. This information can be utilized to determine the final weight vector. To preserve and use the information in the initial weights, one must not overtrain the SLP. This is a very valuable property of adaptively trained neural networks.

The more I work in the ANN discipline, the more I marvel at the remarkable qualities of neural networks. Specifically, I am amazed that the SLP classifier dynamics progress from the simplest algorithms to the most complex algorithms in a natural manner. Statisticians and engineers have long understood that in designing decision-making algorithms from experimental data one needs to progress from simple algorithms to complex algorithms. The artificial neurone accomplishes this complexity progression in a natural manner. Statisticians required several decades to develop a number of statistical classification and regression rules: Fisher (1936) proposed his linear discriminant function more than six decades ago and Vapnik devised his support vector machine (Vapnik, 1995) only recently. The neurone, however, implements this algorithm design progression in a logical sequence. One can think of this progression as nature's method.

There is a plethora of useful information currently available in the field of statistical pattern recognition. The main element of statistical pattern recognition is the assumption that the pattern vectors are random vectors that can be modelled as observations from multivariate distributions. In the parametric approach, the classifier designer assumes he knows this multivariate distribution precisely. In order to determine the classification algorithm, one needs to estimate the unknown multivariate density function parameters using the training data. Researchers have found that the more complex the multivariate density model assumed, or equivalently, the greater the number of parameters to be estimated, the greater the number of training vectors that must be employed to adequately determine the classifier. Therefore, a large number of parsimonious multivariate distribution densities have been formulated for this purpose.

One of the most interesting and important facts utilized in parametric classification is that if some of the pattern-class densities' parameters are in common, these parameters have a negligible influence on the increase in the generalization error. This is a very favourable property of the statistical parametric classifier approach. On the other hand, incorrect assumptions about the type of the probabilistic distribution assumed for the pattern vectors lead to an increase in the classification error. This is one of the main shortcomings of the parametric statistical classifier approach.

Being interested in both statistical pattern recognition and artificial neural network theory, I perceived a certain conflict between these two classification paradigms, and sometimes even a dissonance among proponents of these two classification methods. Statisticians generally have good mathematical backgrounds with which to analyse decision-making algorithms theoretically. They have proven many rigorous results concerning the optimality of statistical classification algorithms. However, they often pay little or no attention to the applicability of their own theoretical results and generally do not heed practical or even theoretical results obtained by ANN researchers.

ANN scientists advocate that one should make no assumptions concerning the multivariate densities assumed for the pattern classes. They, instead, propose that one should assume only the structure of the decision-making rules, for example a linear discriminant function in the space of original or transformed features, and then estimate the unknown rule coefficients (weights) directly from the training data. For this they suggest one minimize the number of errors incurred while classifying the training vectors (empirical error). Many such algorithms have been suggested to solve practical problems. Some of these algorithms have a theoretical justification and some have no theoretical elucidation yet.

Known properties and deficiencies of both statistical and neural classification algorithms hints that one should *integrate* the two classifier design strategies and utilize their good qualities. There are three key aspects of this integration. The first key is the fact that the correct initial weights of the perceptron contain information that can be saved for use in future training processes. Thus, we can utilize pattern classifiers based on statistical methods to define the initial perceptron weight vector. The second key is the fact that, during training, the perceptron changes its statistical properties and evolves from simple classification algorithms to more complex classification algorithms. The third key is the fact that, one can use the diversity of statistical methods and the multivariate models to perform different whitening data transformations, where the input variables are decorrelated and scaled in order to have the same variances. Then while training the perceptron in the transformed feature space, we can obtain the Euclidean distance classifier after the very first iteration. In the original feature space, the weight vector of this classifier is equivalent to the decision making rule found by utilizing the statistical methods and the multivariate models just mentioned. Further training can diminish the negative aspects of approximately correct or incorrect statistical assumptions.

Thus, it is possible to merge the statistical and neural approaches. Specifically, instead of using statistical methods and the multivariate models directly to design the classifier, we can use them to whiten the data. We can then train the perceptron paying special attention to the optimal stopping time. The data whitening reduces the generalisation error and simultaneously speeds up the training process. This approach merges the qualities of both statistical and neural classification algorithm design strategies. Investigations of the visual cortex in biological systems, however, have shown that the input decorrelation technique is already realized in natural systems. This is one more piece of evidence that the data decorrelation and scaling technique performed prior to perceptron training is a natural method of information processing.

An objective of this book is to explain the details necessary to understand and utilise the integrated statistical and neural net approach to design the classification rules. We, therefore, present a discussion of the diversity of linear and non-linear statistical pattern classification algorithms that can be utilised in an advanced neural network analysis. Special attention is paid to the assumptions used to design the algorithm, the generalisation error, and the training-set size relationships. Knowledge of these relationships allows one to analyse and compare the amount of information obtained from the training data, the assumptions, or from the educated guesses utilised to construct the decision-making algorithm. This is perhaps the central question that arises in machine learning and classifier design theory.

Performance, complexity, and training-set size relationships in the nonparametric neural net approach have been discussed in a number of books (Vapnik, 1982, 1995; Wolpert, 1995; Cherkassky and Mulier, 1996; Vidyasagar, 1997, etc.). According to Wolpert, “the statistical and neural net approaches have their own jargon, their own mathematical models, their own concern, and their own results. And, for the most part, they don’t interact”. This book primarily takes a statistical point of view but does not ignore other approaches. Alternative texts are Raudys (1976), Aivazian *et al*. (1988), Fukunaga (1990), McLachlan (1992), Devroye, Gyorfi and Lugosi (1996), Duda, Hart, and Stork (2000). The present book, however, is more focused on the integration of statistical and neural approaches to design the classification algorithms. In order to focus on performance, complexity, and design set size relationships more deeply in this book, I employ a simple formulation of the pattern recognition problem. For more general formulations of the pattern recognition problem and related questions I refer interested readers to broader texts such as Fukunaga’s book. To make the book accessible to more readers, I adopt Fukunaga’s notation.

The book is targeted to graduate students and research workers in data modelling, pattern recognition, and artificial neural networks. No special background beyond a good working knowledge of probability and statistics, elements of linear algebra, and calculus at the undergraduate level is required. However, one will benefit by having a popular pattern recognition or neural networks book (e.g., Fukunaga (1990) or Haykin (1998)) close at hand.

The book is organized somewhat like a reference book. At the same time I pay particular attention to the ideas used to design and analyse statistical classification algorithms that can be useful for understanding artificial neural network classifiers. For analysis of neural networks and statistical algorithms, the most important aspect is assumptions utilised in the algorithm design process. Therefore, in order to have a comprehensive point of view of the problem, I omit a part of the details concerning the estimation of parameters of well known statistical algorithms that can be found in the popular literature. To make understanding the main ideas easier, I provide a number of simple illustrative examples.

In the first chapter, I present the main definitions and terms and review the effects of finite training-set size on classifiers. In the second chapter, I review principles of the statistical decision theory, review important statistical multivariate data models, and give a taxonomy of pattern classification algorithms that can be obtained or improved while training ANN classification systems. In the third chapter, I present known results concerning the performance and generalisation error relationships for a number of parametric and nonparametric classification algorithms. In the fourth chapter, I consider training peculiarities and the generalisation and complexity of neural classifiers. In the fifth chapter, I explain the integration of the statistical and neural classification approaches. In the sixth and final chapter, I consider the topic of model selection, paying special attention to the accuracy of solutions to this important topic.

The main contents of this book crystallised during my work as a tenured researcher at the Institute of Mathematics and Informatics in Vilnius. However, other parts of the book were generated while visiting and collaborating with researchers, professors and graduate students at several institutions. These include the Department of Computer Science at Michigan State University; the Department of Information Systems at the Hankamer School of Business, Baylor University, USA; the Departments of Electrical Engineering and Applied Physics, Delft University of Technology, The Netherlands; LAFORIA, Institute Blaise Pascal, University Paris 6; Department of Electrical and Electronic Engineering, Bogazici (Bosforus) University, Istanbul; Laboratory of Information Representation, RIKEN, Tokyo; Ford Motor Company Scientific Research Laboratories. Many ideas and solutions were developed while closely collaborating with researchers of A. N. Kolmogorov Laboratory of Statistical Methods in Moscow State University. In particular I wish to thank Yurij Blagoveschenskij, Lev Meshalkin, Vladimir Vapnik, Gennadij Lbov, Anil Jain, Dean Young, Eric Backer, Bob Duin, Françoise Fogelman-Soulie, Patrick Gallinari, Bulent Sankur, Shun-ichi Amari, Andrzej Cichotski, Gintaras Puškorius and many others for useful and encouraging discussions and their hospitality and aid.

I would also like to thank my colleagues and former and present graduate students at the Institute of Mathematics and Informatics (Vilnius), Kaunas Vytautas Magnus University and Vilnius University for their challenging discussions and for their assistance in simulation studies. Special acknowledgement is expressed to Professor Laimutis Telksnys, Vitalijus Pikelis, Marina Skurichina, Tautvydas Cibas, Valdas Dičiūnas, Aistis Raudys, Aušra Saudargienė and Arūnas Janeliūnas. Exceptional thanks are expressed to Roy Davies, Edvardas Povilonis, Dean Young and Laura Thompson for their useful discussions and their aid in editing this book.

The author acknowledges the financial support from the Lithuanian State Science and Studies Foundation.

Vilnius, August 2000 *Šarūnas Raudys*

Contents

Abbreviations and Notations

ANN	artificial neural network
BP	back-propagation
DF	discriminant function
GCCM	Gaussian with common covariance matrix
EDC	Euclidean distance classifier
FS	feature space
k-NN	*k*-nearest neighbour
LDF	linear discriminant function
LOOM	leaving-one-out method
LVQ	learning vector quantisation
MEE	minimum empirical error
MLP	multilayer perceptron
PMC	probability of misclassification
PW	Parzen window
QDF	quadratic discriminant function
RBF	radial basis function
RDA	regularised discriminant analysis
RM	resubstitution method to estimate a training-set (empirical) error
SLP	single layer perceptron
SG	spherical Gaussian
SV	support vector
VC	Vapnik-Chervonenkis
ZEE	zero empirical error

$\boldsymbol{X} = (x_1, x_2, \dots, x_n)^T$ is an n-variate vector to be classified; a subscript ^{T} denotes a transpose operation

n is a dimensionality of the feature vector $\boldsymbol{X}$

L is a number of pattern classes (categories, populations), $\omega_1, \omega_2, \dots, \omega_L$

$p_i(\boldsymbol{X})$ is the class conditional probability density function (PDF) of vector $\boldsymbol{X}$ belonging to class ω_i

P_i is the a priori probability that observation $\boldsymbol{X}$ belongs to class ω_i

Ω is a feature space

One part of the design set is called *a training (learning) set*, while the other one is called *a validation set.* A set used to evaluate performance of the final variant is called *the test set*

N_i is the number of training vectors from class ω_r

$N = N_1+N_2$ if $N_2=N_1$, we denote $\overline{N} = N_1=N_2= N/2$

$\boldsymbol{M}_r$ is a mean vector, of the pattern class ω_r

$\hat{\boldsymbol{M}}_r$ is an arithmetic mean of the training-set of the class ω_r

$$\hat{\boldsymbol{M}}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{X}_j^{(i)}$$

$\boldsymbol{X}_j^{(i)}$ is the j-th training set observation from ω_i

$\hat{\boldsymbol{M}} = ½ (\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)$ is a centre of the training set (if $N_2=N_1$)

Σ_i is an $n \times n$ covariance matrix (CM) of the of category ω_i. It is a function of n variances of all n features and $n(n$-1) correlations between them. When both pattern classes share the same CM, we denote it by $\overline{\Sigma}$

$$\begin{bmatrix} \Sigma_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Sigma_2 & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Sigma_H \end{bmatrix}$$ is the block diagonal covariance matrix

$\hat{\Sigma}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)(\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)^T$ is a sample estimate of Σ_i

$\hat{\overline{\Sigma}}$ is a sample estimate of $\overline{\Sigma}$. $\hat{\overline{\Sigma}} = N_1/(N_1+N_2)\,\hat{\Sigma}_1 + N_2/(N_1+N_2)\,\hat{\Sigma}_2$

$\hat{\Sigma} = \Phi\Lambda\Phi^T$ is a singular value decomposition of matrix $\hat{\Sigma}$

$\hat{\Sigma}^{+} = \Phi \begin{bmatrix} \lambda^{-1} & 0 \\ 0 & 0 \end{bmatrix} \Phi^T$ is a pseudoinversion of $\hat{\Sigma}$

$\hat{\Sigma} + \lambda\mathbf{I} = \Phi(\Lambda + \lambda\mathbf{I})\Phi^T$ is a ridge estimate of the sample covariance matrix $\hat{\Sigma}$

$\delta^2 = (\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\, \bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2)$ is a squared generalised (Mahalanobis) distance between two pattern classes

$n^* = \dfrac{(\boldsymbol{M}^T\boldsymbol{M})^2 (tr\, \bar{\Sigma}^2)}{(\boldsymbol{M}^T\bar{\Sigma}\boldsymbol{M})^2}$ is an effective dimensionality of EDC for GCCM data

$||\boldsymbol{X} - \hat{\boldsymbol{M}}_r||^2 = (\boldsymbol{X} - \hat{\boldsymbol{M}}_r)^T(\boldsymbol{X} - \hat{\boldsymbol{M}}_r)$ is an Euclidean distance between $\boldsymbol{X}$ and $\hat{\boldsymbol{M}}_r$

$h(X) = \boldsymbol{X}^T\boldsymbol{V} + v_0$ is a linear discriminant function (DF)

$v_0, \boldsymbol{V}^T = (v_1, v_2, \dots, v_n)$ are weights of the discriminant function

$N_X(\boldsymbol{M}_i, \Sigma_i)$ is an n-dimensional Gaussian distribution density

$N_X(\boldsymbol{X}, \boldsymbol{M}_i, \Sigma_i) = (2\pi)^{-n/2}\,|\Sigma_i|^{-1/2}\, e^{-1/2\, (\boldsymbol{X} - \boldsymbol{M}_i)^T\, \Sigma_i^{-1}\, (\boldsymbol{X} - \boldsymbol{M}_i)}$

$cost_t = \dfrac{1}{N_1 + N_2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} (t_j^{(i)} - f(\boldsymbol{V}^T\boldsymbol{X}_j^{(i)} + v_0))^2$ is a cost function

$f(\boldsymbol{V}^T\boldsymbol{X}_j^{(i)} + v_0)$ is an activation function

$t_j^{(i)}$ is a desired output (a target) for $\boldsymbol{X}_j^{(i)}$, the j-th training vector from ω_i

$\Phi\{a\} = \int_{-\infty}^{a} (2\pi)^{-1/2}\, \sigma^{-1} \exp\{-½\, t^2/\sigma^2\}dt$ is a standard Gaussian cumulative distribution function

$\varepsilon_B = \Phi\{-½\, \delta\}$ is a Bayes error for two Gaussian populations with common CM

ε_∞^{A} is an asymptotic error for the classifier A

ε_N^{A} is a conditional probability of misclassification (PMC), a conditional generalisation error

$\bar{\varepsilon}_N^{A}$ is an expected probability of misclassification or simply an expected generalisation error

tr is a trace of the matrix (a sum of diagonal elements)

$\mathbf{I}_r$ is an $r \times r$ identity matrix (ones on the diagonal and zeros outside)

1. Quick Overview

The main objective of this chapter is to define the terminology to be used and the primary issues to be considered in depth in the chapters that follow. In the first section, I present three of the simplest and most popular classification algorithms, and in the second section, I describe single and multilayer perceptrons and the training rules that are used to obtain classification algorithms. I give special attention to the most popular perceptron-training rule, back-propagation (BP). In the third section, I show that the three statistical-based classifiers described in Section 1.1 can be obtained via single-layer perceptron (SLP) training. The fourth section deals with performance, complexity and training-set size relationships, while the fifth section explains how to utilise these relationships while determining the optimal complexity of the decision-making algorithm. In the final section, I explain the overtraining effect that can arise while training artificial neural networks. Some bibliographical and historical remarks are included at the end of the chapter.

1.1 The Classifier Design Problem

The simplest formulation of a pattern classification task is to allocate an object characterised by a number of measurements into one of several distinct pattern classes. Let an unknown object (process, image) be characterised by the D-dimensional vector

$$\boldsymbol{S}_D = \begin{bmatrix} s_1 \\ s_2 \\ \\ s_D \end{bmatrix},$$

and let $\omega_1, \omega_2, \ldots, \omega_L$ be labels representing each of the potential pattern classes.

Then in allocatory pattern recognition, the objective is to construct an algorithm that will use the information contained in the components $s_1, s_2, \ldots, s_D$ of the vector $\boldsymbol{S}_D$ to allocate this vector to one of the L distinct categories (pattern classes) listed. Typically, the pattern classifier calculates L functions (similarity measures) $o_1, o_2, \ldots, o_L$ and allocates the unlabeled vector $\boldsymbol{S}_D$ to the class with the highest similarity measure.

In order to design such a classification algorithm one needs to fix the list of potential pattern classes. The measurements are chosen by their degree of relevance to the pattern classification problem at hand. The D measurements are obtained to formalise the large amount of information containing various a priori, unknown relationships among the pattern classes, among components of the vector $\boldsymbol{S}_D$, etc.

Many useful ideas and techniques have been proposed during 50 years of research into the field of pattern recognition. In order to simplify the design process, researchers have proposed that one split the pattern recognition problem into two stages: the feature extraction stage and the pattern classifier design stage. In the feature extraction stage, a comparatively small number of informative features (n) are obtained from a large number of initial measurements by means of linear or non-linear transformations (typically, n<<D). Then one develops an algorithm to classify an n-variate vector

$$\boldsymbol{X} = (x_1, x_2, \dots , x_n)^T$$

into one of L pattern classes, where x_1, x_2, … , x_n are components of the unclassified vector $\boldsymbol{X}$ and ^{T} denotes the transpose operation.

Example 1. Let s_1, s_2, … , s_D contain D values of an acoustic signal. Suppose s_1, s_2, … , s_D are realisations of a stationary random process described by the autoregression equation $s_t = x_1 s_{t-1} + x_2 s_{t-2} + \dots + x_n s_{t-n}$, where n<<D. Then we can estimate the unknown coefficients x_1, x_2, … , x_n from each segment of the signal and use x_1, x_2, …, x_n as the components of the feature vector $\boldsymbol{X}$.

We discuss feature extraction methods in Chapter 6. In this book, we restrict ourselves to specific classifier design tasks and we assume that the n features are already extracted so that we can perform the classification of the n-dimensional vectors $\boldsymbol{X}$. For simplicity, we mostly consider only the two category case (L = 2) throughout the book.

Typically, in designing the classifier, one selects a number of n-dimensional vectors with known class membership and uses this data set as the main information source to develop a pattern recognition algorithm. This data set is called *a design set*. Often, while designing the pattern recognition algorithms, one develops a number of classifier variants differing in the feature system, in the methods used to design the classifier, in the parameter estimation methods, etc. A problem arises in determining which classifier variant to use. In order to obtain a reliable unbiased estimate of the performance of different variants for the recognition algorithm, one should use part of the design set for estimation of the algorithm's parameters (training) and another part for performance evaluation. The first part of the design set is called *a training (learning) set*, while the second part is called *a validation set*.

There exist different modifications for the data-splitting strategies. One of them is to perform the second test where the design and the validation sets are interchanged. We consider other strategies in Chapter 6. In order to evaluate the performance (generalisation error) of the final variant, we must use a new data set that have not been used for either training or for validation. This *test set* should be used only once.

An important source of “information” in developing a pattern classification algorithm involves assumptions and hypotheses concerning the structure and configuration of the pattern classes and the type of classifier decision boundary desired. One of the most popular hypotheses is that the *n*-dimensional vectors of distinct pattern classes are situated in different areas of a multivariate *n*-dimensional pattern space. One validation of this assumption is that these areas are “compact”. Going further, one can assume that the arithmetic mean $\hat{\boldsymbol{M}}_r$ represents the pattern class ω_r sufficiently well. Then one can allocate an unknown vector $\boldsymbol{X}$ according to its Euclidean distance to the means $\hat{\boldsymbol{M}}_1, \hat{\boldsymbol{M}}_2, \dots, \hat{\boldsymbol{M}}_L$ of the L distinct categories. The Euclidean distance between two vectors is defined as

$$\|\boldsymbol{X} - \hat{\boldsymbol{M}}_r\|^2 = (\boldsymbol{X} - \hat{\boldsymbol{M}}_r)^T (\boldsymbol{X} - \hat{\boldsymbol{M}}_r), \tag{1.1}$$

where $\boldsymbol{a}^T\boldsymbol{b}$ denotes a scalar product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

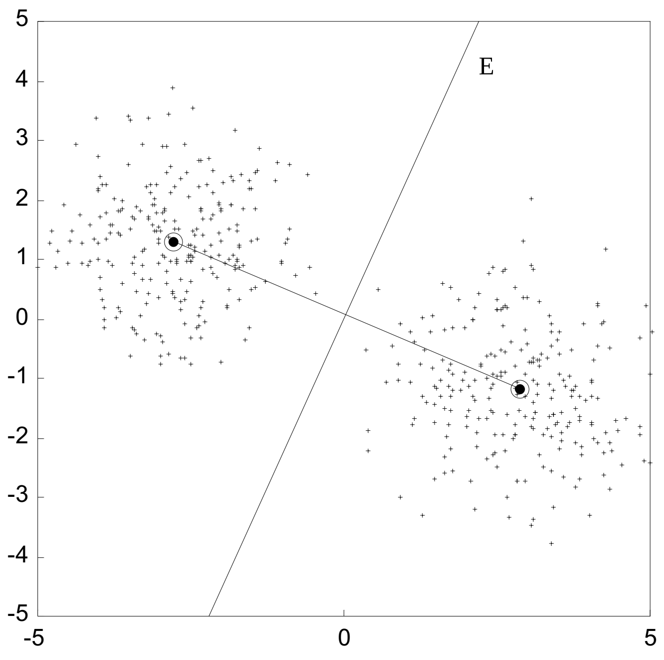

Fig. 1.1. Bivariate vectors sampled from two pattern classes, the sample means of the classes (circles), and the linear decision boundary of the Euclidean distance classifier.

This classifier is called the Euclidean distance classifier (EDC) or the nearest means classifier (Sebestyen, 1962). To design the EDC, one assumes that the pattern vectors of each class are similar among themselves and close in proximity to a “typical member” of this class (the centre represents the sample mean vector of this class). In Figure 1.1, we have depicted 300 bivariate vectors belonging to each of two pattern classes and the linear decision boundary of the EDC. The sample means of the classes, $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$, are denoted by circles.

In the two pattern-classes case, we perform allocation of the vector $\boldsymbol{X}$ according to the difference

$$||\boldsymbol{X} - \hat{\boldsymbol{M}}_1||^2 - ||\boldsymbol{X} - \hat{\boldsymbol{M}}_2||^2,$$

and the classification algorithm can be written as a linear discriminant function (DF) $h(\boldsymbol{X}) = \boldsymbol{X}^T \boldsymbol{V}^{\mathrm{E}} + v_0$ with

$$\boldsymbol{V}^{\mathrm{E}} = \hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2 \quad \text{and} \quad v_0 = -\tfrac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)^T \boldsymbol{V}^{\mathrm{E}}. \qquad (1.2)$$

The allocation of the unclassified vector $\boldsymbol{X}$ to one of the two pattern classes is performed according to the sign of the discriminant function. This classification rule can also be obtained from the statistical decision theory if one assumes that the vector $\boldsymbol{X}$ is random and its distribution is the spherical Gaussian distribution $N_X(\boldsymbol{M}_r, \sigma^2\mathbf{I})$. Here, $\boldsymbol{M}_r$ denotes the n-dimensional mean vector of the r-th pattern class and $\sigma^2\mathbf{I}$ is the covariance matrix (CM). For this pattern-class data model the CM is equal to the identity matrix $\mathbf{I}$ multiplied by the scalar σ^2 (the variance of each feature). For this theoretical data model, the EDC is an asymptotically (when the number of training vectors $N \to \infty$) optimal classifier, according to the criterion of classification error minimisation.

In the general case, the $n \times n$ covariance matrix Σ of a multivariate pattern-class distribution is a function of n variances and $n(n-1)$ correlations. If one assumes that both pattern classes share *the same* CM, we denote this assumption by $\bar{\Sigma}$. Otherwise, Σ_r represents the associated CM for the pattern-class ω_r. Generally, $\bar{\Sigma} \neq \sigma^2\mathbf{I}$, and the EDC is no longer an asymptotically optimal classification rule in terms of minimising the classification error.

Example 2. In Figure 1.2, we have a graph of 300 bivariate correlated vectors ($\times$- marks and small pluses) of two Gaussian pattern classes $N_X(\boldsymbol{M}_r, \bar{\Sigma})$ and the linear decision boundary of the EDC, depicted as line E. The sample means of the classes are denoted by circles, parameters of the data model are:

$$\boldsymbol{M}_1 = -\boldsymbol{M}_2 = (-2, 2)^T, \text{ and } \bar{\Sigma} = (2, 1; 1, 9)).$$

Obviously, for this data model the EDC is not the best classification rule. To design an appropriate classifier, one needs to take into account the fact that the feature variances are unequal and that the features are correlated. For such a data model one can use a linear discriminant function of the form $h(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{V}^{\mathrm{F}} + v_0$ with

$$\boldsymbol{V}^{\mathrm{F}} = \hat{\Sigma}^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2) \quad \text{and} \quad v_0 = -\tfrac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)^T \boldsymbol{V}^F, \qquad (1.3)$$

where $$\hat{\Sigma} = \frac{1}{N_1 + N_2 - 2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)(\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)^T \qquad (1.4)$$

is the pooled sample covariance matrix, and allocation of the unlabeled vector $\boldsymbol{X}$ is performed according to the sign of the discriminant function (1.3). This is the first classification rule proposed in the statistical classification literature (Fisher, 1936) and it is now called the standard Fisher DF. The decision boundary corresponding to the Fisher DF is depicted by line F in Figure 1.2.

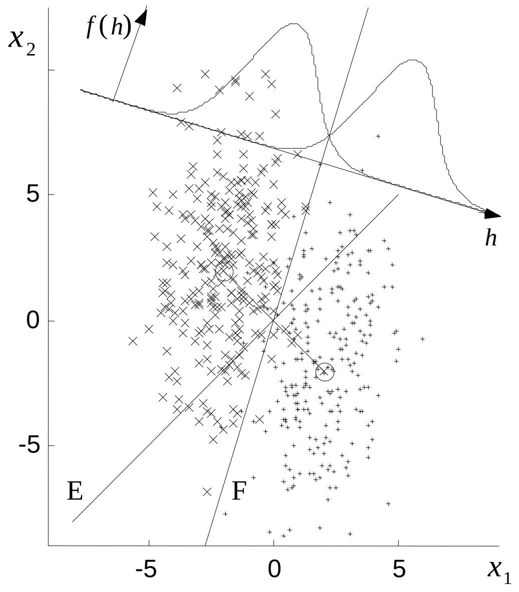

Fig. 1.2. A plot of 250+250 bi-variate vectors (×-marks and small pluses) sampled from two Gaussian pattern classes, sample means of the classes (circles), and the linear decision boundaries: the Euclidean distance classifier E, and the Fisher linear DF. Above is a plot of the two pattern-class densities *f(h)* in the direction *h* of their best separability.

To obtain the Fisher rule, we project the training vectors onto a straight line h perpendicular to the decision boundary F. In the upper part of Figure 1.2, we have depicted the distribution densities $f_1(h)$ and $f_2(h)$ of the discriminant function values $h(\boldsymbol{X})= \boldsymbol{X}^T \boldsymbol{V}^F + v_0$ on a projection line h. In the new univariate space, both pattern classes are almost nonintersecting. This projection is the best linear projection possible onto a line.

In the classification literature, a large number of other parametric classification rules have been proposed. In the statistical approach, we postulate that $\boldsymbol{X}$, being the vector to be classified, is random. To obtain the classification rule (1.3), we can assume that the multivariate density of the vector $\boldsymbol{X}$ is Gaussian. Then we can estimate the unknown parameters of the Gaussian densities from the data and apply statistical decisions theory.

There is another approach to designing classification rules. Advocates of this alternative approach believe that one is unwise to make assumptions about the type of multivariate pattern-class density model and its parameters. Also, advocates of this approach insist that one should not estimate the classification-rule parameters in an indirect manner. Instead of constructing a parametric statistical classifier, these classifier designers suggest that a much better classifier-design method is to make assumptions about the classification rule structure (for example a linear discriminant function) and then estimate the unknown coefficients (weights) of the classification rule directly from the training data. As a criterion of goodness, they propose that one minimise the number of classification errors (empirical error) while classifying the training vectors. This is a typical formulation of the classifier design problem utilised in artificial neural network theory. One of several possible approaches to determining the unknown weight vector $\boldsymbol{V}$ is to adapt its components in an iterative manner. This weight-determination method is explained in the following section. Another entirely hypothetical way to estimate the weights is to generate M (a large number) random vectors $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ … , $\boldsymbol{V}_M$, and to estimate the empirical errors for all M variants and then to choose the best set of weights. However, this random search procedure is utilised only in theoretical analyses.

The EDC is an asymptotically optimal classification rule for spherically Gaussian pattern classes with unequal mean vectors. Here, the word "asymptotically" means that the training-set size N tends to infinity. The standard linear Fisher DF is asymptotically optimal for Gaussian pattern classes that share a common covariance matrix $\overline{\Sigma}$ but have unequal mean vectors. However, this rule is no longer asymptotically optimal when the CMs of different classes are unequal or when the data is not Gaussian. For such types of data, the minimum empirical error (MEE) classifier is a good alternative to the EDC and Fisher's DF, which are parametric statistical classifiers.

Example 3. In order to see the differences among the classifiers in Figure 1.3, we present their decision boundaries and the data from two bivariate Gaussian classes $N_X(\boldsymbol{M}_r, \overline{\Sigma})$ contaminated with 10% additional Gaussian noise $N_X(\mathbf{0}, \mathbf{N})$.

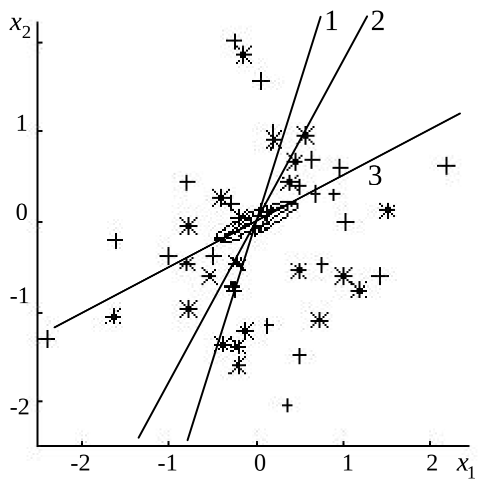

Fig. 1.3. Decision boundaries of the Euclidean distance classifier (1), the Fisher linear DF (2), and the MEE classifier (3). The noise patterns are denoted by "$*$" and "+".

The pattern class models have different means $\boldsymbol{M}_1 = -\boldsymbol{M}_2 = \boldsymbol{M}$ and share a common covariance matrix $\overline{\Sigma}$. The Gaussian components are represented by two small ellipses. Parameters of the data model are

$$\boldsymbol{M} = \begin{bmatrix} 0.10 \\ 0.05 \end{bmatrix}, \; \overline{\Sigma} = \begin{bmatrix} 0.040 & 0.018 \\ 0.018 & 0.01 \end{bmatrix}, \; \mathbf{N} = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}.$$

In this book, we refer to this data configuration as data model **A**.

1.2 Single Layer and Multilayer Perceptrons

The **non-linear single layer perceptron** (SLP) is a simplified mathematical model of an elementary information processing cell in living organisms – a neurone. Animal and human brains are composed from billions of such cells, which are interconnected. The standard belief is that the neurones are organised into layers, and neurones in one layer are interconnected with neurones in the next one. The complexity of connections between the neurones resembles nets. In an analogous model, the information processing units, i.e. artificial neurones, are also structured into layers. The units in one layer are interconnected with units in the adjacent layer. Thus the term *artificial neural nets* appeared.

According to Rumelhart *et al.* (1986) the SLP has several inputs (say n), $x_1, x_2, ..., x_n$, and one output, o, that is calculated according to the equation

$$o = f(\boldsymbol{V}^T\boldsymbol{X} + v_0), \tag{1.5}$$

where $f(net)$ is a non-linear activation function, e.g. a sigmoid function:

$$f(net) = 1/(1+\mathrm{e}^{-net}) = f^{sigmoid}(net), \tag{1.6}$$

and v_0, $\boldsymbol{V}^T = (v_1, v_2, ..., v_p)$ are the weights of the discriminant function (DF).

The output of the SLP can be used to perform classification of the unclassified vector $\boldsymbol{X}$. When the sigmoid function (1.6) is used, we compare the output o with the threshold value of 0.5. The decision boundary is linear as are the three linear boundaries depicted in Figure 1.3. To find the weights for the SLP, we minimise a certain training set's cost function. The most popular cost function is the sum of squares cost function

$$cost_t = \frac{1}{N_1 + N_2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} \left(t_j^{(i)} - f(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0) \right)^2, \tag{1.7}$$

where $t_j^{(i)}$ is the desired output (a target) for $\boldsymbol{X}_j^{(i)}$, the j-th training observation from class ω_i, and N_i is the number of training vectors from class ω_i.

In this pattern recognition algorithm, the desired outputs are associated with the class label. If the sigmoid function (1.6) is used, then typically one uses either $t_j^{(1)} = 1$, $t_j^{(2)} = 0$ or $t_j^{(1)} = 0.9$, $t_j^{(2)} = 0.1$. Outputs of the activation function (1.6) can vary in the interval of (0, 1). Therefore, we refer to the target values of 1 and 0 as “boundary” values, and we refer to the values 0.1 and 0.9 as “proximal” values (the values recommended by Rumelhart *et al*., 1986). Some authors use another activation function:

$$f^{tanh}(net) = (e^{net} - e^{-net}) / (e^{net} + e^{-net}) = 2f^{sigmoid}(2net) - 1, \tag{1.8}$$

For this function $t_j^{(1)} = 1$, $t_j^{(2)} = -1$ (the boundary values), or $t_j^{(1)} = 0.8$, $t_j^{(2)} = -0.8$ (proximal values). Both activation functions (1.6) and (1.8) are smooth and can be differentiated. Such functions are commonly referred to as “soft-limiting” functions. In extreme cases, there are times when we analyse the hard-limiting activation function

$$f^{hard}(net) = \begin{cases} 1 \text{ if } net > 0 \\ 0 \text{ if } net \leq 0 \end{cases}. \tag{1.9}$$

In this case, $t_j^{(1)} = 1$ and $t_j^{(2)} = 0$. Typically, to determine the weights v_0, $(v_1, v_2, \dots, v_n) = \boldsymbol{V}^T$, we minimise the cost function (1.7). In the gradient descent optimisation algorithm (delta rule), we calculate derivatives of the cost function and find the weights in an iterative manner. At step *t*, we update the weight vector $\boldsymbol{V}_{(t)}$ according to equations

$$\boldsymbol{V}_{(t+1)} = \boldsymbol{V}_{(t)} - \eta \frac{\partial cost_t}{\partial \boldsymbol{V}}, \; v_{0(t+1)} = v_{0(t)} - \eta \frac{\partial cost_l}{\partial v_0}, \tag{1.10}$$

where η is the learning-step and $\frac{\partial cost_t}{\partial \boldsymbol{V}}$ is the gradient of the cost function.

When the hard-limiting cost function is used, we cannot calculate the gradient. In such cases, we must use other optimisation methods. To realise the training procedure (1.10), we must first choose the initial weights, $\boldsymbol{V}_{(t=0)}$ and $v_{0(t=0)}$. For this purpose, the neural net literature recommends one use small random numbers. In this book we advocate that one use non-random weight selection methods that can often lead to classifiers with a smaller generalisation error.

If expertly used, the training rule (1.10) can produce different algorithms to design the statistical classifiers. In fact, all three boundaries depicted in Figure 1.3 can be obtained while training the non-linear single layer perceptron. In Section 1.3 and Chapter 4, we demonstrate how this is accomplished.

The **multi-layer perceptron** (MLP) is the oldest and most frequently used type of the artificial neural network (ANN). A typical MLP classifier consists of several layers of artificial neurones containing single layer perceptrons (see Figure 1.4). Input nodes of the MLP classifier correspond to the *n* components of the feature vector. The output-layer can have one or several (say *L*) output neurones. The input signals propagate from the input layer to the output layer. Such networks are called the feedforward ANN. Typically, each neurone in the output layer corresponds to a pattern class label. Inputs to L output neurones are outputs $f_1, f_2, ..., f_h$ of the h neurones that constitute a lower (hidden) layer. Complex MLP classifiers can have many hidden layers.

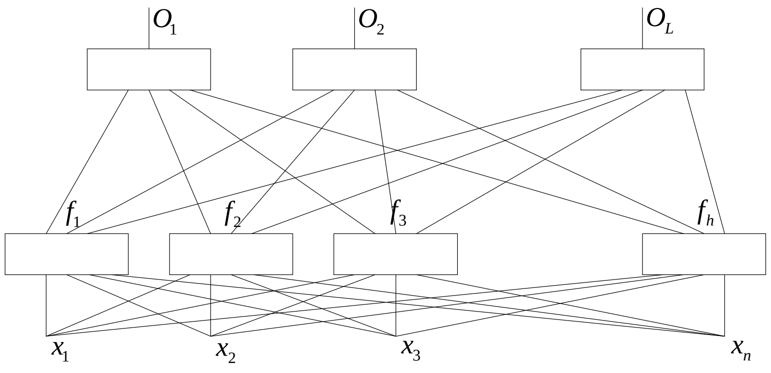

Fig. 1.4. MLP with one hidden layer.

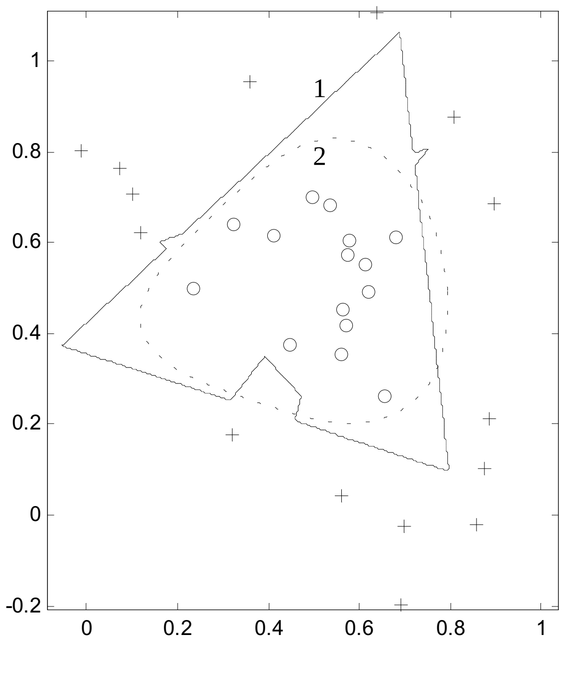

Fig. 1.5. 30 bivariate training vectors and decision boundaries of MLP with 10 hidden units and hard-limiting (1) and smooth (2) activation functions.

In order to find the classifier weights one must minimise a cost function similar to function (1.7). In the gradient descent optimisation algorithm, we need to calculate derivatives of the cost function and update the weights for all neurones in an iterative manner similar to the training of the SLP. While calculating the gradients of the hidden layer neurones, we propagate the error signal $t_j^{(i)} - f(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0)$ back to the lower layers. This technique is called *Back Propagation* (BP). The reader can find equations for the gradients of the output and hidden layer neurones in practically all artificial neural network monographs and textbooks.

When the hard limiting function is used, we obtain a piecewise-linear decision boundary. Smooth activation functions assist in obtaining smooth decision boundaries (compare boundaries 1 and 2 in Figure 1.5). The degree of non-linearity of the smooth activation function can be changed by multiplication of all components of the hidden layer weights by a certain positive constant α (Section 4.6.9). In fact, both boundaries in Figure 1.5 have been obtained by multiplying the weights of all neurones in the hidden layer of a well trained perceptron by the factor α, where α =1000 for the hard limiting activation function and α = 1/3 for the soft limiting function. This bivariate data model is denoted by **SF2** and is discussed in more depth in Chapter 4. We see that the *non-linearity* of the activation function is an essential factor that affects the shape of the decision boundary. In general, the perceptron with one hidden layer, a soft-limiting activation function, and a sufficiently large number of hidden units can realise arbitrarily complex decision surfaces.

1.3 The SLP as the Euclidean Distance and the Fisher Linear Classifiers

The objective of this section is to demonstrate that in the adaptive BP training for the two-category case (L = 2), the SLP can realise three known statistical classifiers. In this section, we assume $N_2 = N_1 = \bar{N} = N/2$ with centred data, i.e., $\hat{\boldsymbol{M}} = ½\,(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2) = \mathbf{0}$ and analyse the cost function of the sum of squares (1.7) with activation function (1.8) and targets $t_j^{(1)}=1$, $t_j^{(2)}= -1$. Let the prior weights be zero, i.e., $v_{0(0)} = 0$, $\boldsymbol{V}_{(0)} = \mathbf{0}$. We train the SLP with the standard delta rule (1.10) in a batch-mode (the total gradient). Note that in the total gradient, we change the weights after calculating the gradients for *all* training vectors. For the stochastic gradient, we change the weights after calculating the gradient for *each single* training vector. During the first iteration, the weights and a scalar product, $v_0 + \boldsymbol{V}^T\boldsymbol{X}_j^{(i)}$, are close to zero. Therefore, for the tanh(*net*) activation function (1.8) we have

$$o(v_0 + \boldsymbol{V}^T\boldsymbol{X}_j^{(i)}) \cong v_0 + \boldsymbol{V}^T\boldsymbol{X}_j^{(i)}, \quad \text{and} \quad \partial\, o(v) \cong dv\,.$$

Thus,

$$\frac{\partial\, cost_l}{\partial v_{0(t)}} = -\frac{2}{N}\sum_{i=1}^{2}\sum_{j=1}^{\bar{N}}(t_j^{(i)} - v_{0(t)} - (\boldsymbol{X}_j^{(i)})^T\boldsymbol{V}_{(t)}) = 2v_{0(t)} + 2\,\hat{\boldsymbol{M}}^T\boldsymbol{V}_{(t)} = 0,$$

$$\left.\frac{\partial\; cost_l}{\partial \boldsymbol{V}}\right|_{V=V_{(t)}} = -\frac{2}{N}\sum_{i=1}^{2}\sum_{j=1}^{\bar{N}}\boldsymbol{X}_j^{(i)}(t_j^{(i)} - v_{0(t)} - (\boldsymbol{X}_j^{(i)})^T\boldsymbol{V}_{(t)}) = -\Delta\hat{\boldsymbol{M}} + 2\mathbf{K}\boldsymbol{V}_{(t)}\,, \tag{1.11}$$

where

$$\Delta\hat{\boldsymbol{M}} = \hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2\,,\ \text{and}\ \ \mathbf{K} = \frac{1}{N}\sum_{i=1}^{2}\sum_{j=1}^{\bar{N}}\boldsymbol{X}_j^{(i)}(\boldsymbol{X}_j^{(i)})^T.$$

After the first learning iteration in the training process, we get

$$\boldsymbol{V}_{(1)} = \boldsymbol{V}_{(0)} - \eta\,\frac{\partial cost_l}{\partial \boldsymbol{V}} = \eta\,\Delta\hat{\boldsymbol{M}}\,,\ v_{0(1)} = 0. \tag{1.12}$$

This is the weight vector of the Euclidean distance (nearest means) classifier (EDC) scaled by η. In the first section, we mentioned that the EDC is the asymptotically optimal classification rule for the data model with two spherically Gaussian pattern classes. The fact that a single-layer perceptron becomes a comparatively good statistical classifier after only one iteration is a very nice property of the SLP! To demonstrate this point, we need the following conditions:

E1) the centre of the data $\hat{\boldsymbol{M}}$ is moved to the zero point;
E2) the training process starts from zero weights;
E3) if $N_2 = N_1 = \bar{N}$, we use symmetrical targets, i.e. for activation function (1.8) $t_2 = -t_1$;
E4) the total gradient training (batch mode) method is used.

Second order optimisation methods allow us to obtain the minimum of the quadratic cost function in one iteration. Equating the derivative of the cost (1.7) to zero for non-singular matrix **K**, we obtain $v_0 = 0$, and $\boldsymbol{V} = ½\mathbf{K}^{-1}\Delta\hat{\boldsymbol{M}}$.
For $\hat{\boldsymbol{M}} = \mathbf{0}$ (the centred data) we can use the representation

$$\mathbf{K} = \frac{\bar{N}-1}{\bar{N}}\,\hat{\Sigma} + \frac{1}{4}\,\Delta\hat{\boldsymbol{M}}\,\Delta\hat{\boldsymbol{M}}^T, \tag{1.13}$$

where $\hat{\Sigma}$ is the pooled sample covariance matrix defined in (1.4).

Using Bartlett’s formula for the inversion of a symmetric positive-definite matrix,

$$(\mathbf{A} + \mathbf{V}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{V}\mathbf{V}^T\mathbf{A}^{-1}}{1 + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{V}}\,, \tag{1.14}$$

where **A** is an $n \times n$ symmetric positive-definite matrix, we can express (1.13) in the form

$$\boldsymbol{V} = k_n\ \hat{\Sigma}^{-1} \Delta \hat{\boldsymbol{M}}\ , \tag{1.15}$$

where $k_n = 2/(D^2 + 4(\overline{N} - 1)/\ \overline{N}\ )$, and $D^2 = \Delta \hat{\boldsymbol{M}}^T \hat{\Sigma}^{-1} \Delta \hat{\boldsymbol{M}}$.

If $\hat{\boldsymbol{M}} = \mathbf{0}$ and the classification task is performed according to the sign of the discriminant function, then the positive constant k_n plays no role and expressions (1.15) and (1.3) are identical. Thus, in the linear SLP training process, we can obtain the standard Fisher linear discriminant function. This classifier can also be obtained using back propagation training when one uses the linear SLP (with a linear activation function, e.g. $f(net) = net$). We can approximate the Fisher classifier while training the non-linear perceptron. However, to do this we need to utilise the proximal targets.

Let us now consider the hard-limiting activation function (1.9). Then the cost (1.7) expresses the empirical classification error – the frequency of errors when the training-set is classified. Obviously, successful minimisation of this cost function leads to the minimum empirical error classifier. Unfortunately, in such a case the cost function is no longer differentiable and has many local minima. To tackle the local minima problems special discrete optimisation methods must be applied. In Chapter 4, we show how to approach the minimum empirical error classifier by the BP technique when the smooth activation function is utilised. Moreover, in zero empirical error situations, the expert training of the SLP can lead to a maximum margin classifier. For this classifier, the decision boundary is placed exactly in the middle between the closest training vectors of opposite categories. In Chapter 4, we also demonstrate that in BP training, more statistical classifiers can be obtained.

1.4 The Generalisation Error of the EDC and the Fisher DF

Asymptotic and Bayes error rates. In machine learning and classifier design theory, one of the main performance characteristics is the probability of misclassification. There are several types of classification errors or probabilities of misclassification. In order to differentiate among them, we consider the problem of classifying observations from the two Gaussian with a common covariance matrix (GCCM) pattern classes, $N_X(\boldsymbol{M}_1, \overline{\Sigma}\ )$ and $N_X(\boldsymbol{M}_2, \overline{\Sigma}\ )$.

Assume that the vectors from the first and second categories are equiprobable, i.e. the prior probabilities, P_1 and P_2, of the pattern classes are equal. Thus, $P_2 = 1 - P_1 = ½$. When all parameters are known, statistical discriminant analysis provides an optimal discriminant function that is linear, namely $h(\boldsymbol{X}) = v_0 + \boldsymbol{V}^T\boldsymbol{X}$, with weights

$$\boldsymbol{V}^{\text{optimal}} = \bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) \text{ and } v_0 = -\hat{\boldsymbol{M}}^T\boldsymbol{V}^{\text{optimal}}. \tag{1.16}$$

The probability of misclassification (PMC) of this rule is

$$\varepsilon = P_1\varepsilon_1 + P_2\,\varepsilon_2,$$

where ε_1 is the probability of error of the first sort, i.e. the probability that a vector from the first class will be allocated to the second pattern class: $\varepsilon_1 = Probability\{h(\boldsymbol{X}) < 0 \mid \boldsymbol{X} \in \omega_1\}$, and ε_2 is the probability of error of the second sort, i.e. the probability that a vector from the second class will be allocated to the first pattern class: $\varepsilon_2 = Probability\{h(\boldsymbol{X}) \geq 0 \mid \boldsymbol{X} \in \omega_2\}$.

For the Gaussian data model described above, $h(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{V} + v_0$ is a Gaussian random variable. Thus,

$$Probability\{\, h(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_i) < 0 \mid \boldsymbol{X} \in \omega_i\,) = \Phi\{\, (-1)^i \frac{Eh(\boldsymbol{X}|\boldsymbol{X} \in \omega_i)}{\sqrt{Vh(\boldsymbol{X}|\boldsymbol{X} \in \omega_i)}}\,\}, \tag{1.17}$$

where

$\Phi\{a\} = \int_{-\infty}^{a} (2\pi)^{-1/2}\sigma^{-1}\exp\{-t^2/(2\sigma^2)\}dt$ is the standard Gaussian cumulative distribution function (a quick reminder: $\Phi\{a\} \to 0$ as $a \to -\infty$, $\Phi\{a\} = 0.5$ when $a = 0$, and $\Phi\{a\} \to 1$ when $a \to \infty$),

$E\,h(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_i)$ is the conditional mean:

$$E\,h(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_i) = E(\boldsymbol{X}^T\boldsymbol{V}^{\text{optimal}} + v_0) =$$

$$h(\boldsymbol{X}) = \boldsymbol{M}_i^T\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) + v_0 = ½(-1)^{i+1}(\boldsymbol{M}_1 + \boldsymbol{M}_2)^T\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) = ½(-1)^{i+1}\delta,$$

and $V\,h(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_i)$ is the conditional variance:

$$V\,h(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_i) =$$

$$E\,(\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\bar{\Sigma}^{-1}(\boldsymbol{X} - \boldsymbol{M}_i)\,(\boldsymbol{X} - \boldsymbol{M}_i)^T\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2)$$

$$= (\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\bar{\Sigma}^{-1}\bar{\Sigma}\,\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) = (\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2) = \delta^2,$$

where $\delta^2 = (\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\bar{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2)$ (1.18)

is the squared generalised distance between the two pattern classes.

When $\bar{\Sigma} = \mathbf{I}$ (the $n \times n$ identity matrix, here all n features are uncorrelated and have variances equal to one), δ^2 denotes the conventional squared Euclidean distance between the two vectors, $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$. The generalised squared distance (1.18) was introduced by the Indian statistician P.C. Mahalanobis and, therefore, it carries his name. We use the Mahalanobis distance as a measure of the separability of two GCCM pattern classes.

Inserting the conditional means and the conditional variance into (1.17) yields

$$\varepsilon = \Phi\{ -½\, \delta \}. \tag{1.19}$$

Equation (1.19) shows that, for the pattern-class models $N_X(\boldsymbol{M}_1, \overline{\Sigma})$ and $N_X(\boldsymbol{M}_2, \overline{\Sigma})$, the Mahalanobis distance uniquely determines the classification error of the optimal rule. For example, if $\delta = 2.56$, then $\varepsilon = 0.1$, if $\delta = 3.76$, then $\varepsilon = 0.03$, and if $\delta = 4.65$, then $\varepsilon = 0.01$.

Statistical decision-theory based classifiers are formulated using Bayes formula for conditional probabilities. This classifier-design method allows optimal classification rules to be obtained when all parameters are known. Thus, for the GCCM data model, the discriminant function (1.16) is a Bayes rule, and $\Phi\{-½\,\delta\}$ is its associated error rate. The minimal probability of misclassification is called the Bayes error and is denoted by ε_B. For the data model $N_X(\boldsymbol{M}_1, \overline{\Sigma})$ and $N_X(\boldsymbol{M}_2, \overline{\Sigma})$ the Bayes error coincides with the asymptotic error for the Fisher linear DF. As a reminder, the asymptotic error determines the performance of the classification rule when an arbitrarily large training-set is used to determine the classification rule's coefficients (weights). In this case, we assume we know the parameters $\boldsymbol{M}_1$, $\boldsymbol{M}_2$, and $\overline{\Sigma}$ exactly. Therefore, for the GCCM data model the asymptotic error of the Fisher classifier is $\varepsilon_\infty^{F} = \Phi\{-½\delta\}$.

Consider the Euclidean distance classifier designed for pattern classes with known parameters. The weights of the discriminant function are:

$$V^{E} = \boldsymbol{M}_1 - \boldsymbol{M}_2 \text{ and } v_0 = -½(\boldsymbol{M}_1 + \boldsymbol{M}_2)^T(\boldsymbol{M}_1 - \boldsymbol{M}_2). \tag{1. 20}$$

The asymptotic error rate of this classifier is

$$\varepsilon_\infty^{E} = \Phi\{-½\, \delta^*\},$$

where δ^*, the effective distance between two pattern classes, is defined to be

$$\delta^* = \frac{(\boldsymbol{M}_1 - \boldsymbol{M}_2)^T(\boldsymbol{M}_1 - \boldsymbol{M}_2)}{\sqrt{(\boldsymbol{M}_1 - \boldsymbol{M}_2)^T\overline{\Sigma}(\boldsymbol{M}_1 - \boldsymbol{M}2)}}. \tag{1.21}$$

Calculations show that $\delta^* \le \delta$. Hence, for the GCCM data model we have that $\varepsilon_\infty^{E} \ge \varepsilon_\infty^{F}$. Note that the Bayes error does not depend on the particular classifier design method. Rather, it depends on the characteristics of the data model.

Example 4. Consider the data model with GCCM pattern classes with $n = 20$, $\Delta\boldsymbol{M} = \boldsymbol{M}_1 - \boldsymbol{M}_2 = (-1.7040, 0.5244, 0.49700, \dots, 0.0872, 0.0599, 0.0326)^T$. All correlations among the variables are equal: $\rho = 0.213$, and all n variables have unit variances. For this data model the effective and Mahalanobis distances are $\delta^* = 3.7616$ and $\delta = 4.65$, respectively, and the asymptotic classification errors are $\varepsilon_\infty^{\mathrm{E}} = 0.03$ and $\varepsilon_\infty^{\mathrm{F}} = 0.01$. In this book, we refer to this data configuration as data model **C**.

The expressions for the Mahalanobis and the effective distances, (1.18) and (1.21), respectively, as well as numerical calculation show that the asymptotic errors depend on the type of classifier design method. Both the EDC and the Fisher DF are linear classifiers. However, they use different methods to determine their respective weights. Therefore, in the general case, the asymptotic error values differ as well. We note that theoretically, for certain data models, both classifiers can have the same asymptotic errors.

Generalisation errors. Consider the *sample-based* EDC with the discriminant function (1.2). The sample mean vectors $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$ are functions of the training vectors $\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_2^{(1)}, \boldsymbol{X}_3^{(1)} \dots, \boldsymbol{X}_{N_1}^{(1)}$, and $\boldsymbol{X}_1^{(2)}, \dots, \boldsymbol{X}_{N_2}^{(2)}$. Therefore, $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$ are random vectors. Thus, due to sampling error, $\hat{\boldsymbol{M}}_1$, and $\hat{\boldsymbol{M}}_2$ differ from the true population mean vectors, $\boldsymbol{M}_1$, $\boldsymbol{M}_2$, and the vector $\boldsymbol{V}^{\mathrm{E}}$ and the threshold v_0 in (1.2) differ from the ideal values determined by Equation (1.20). Hence, the decision boundary of the sample-based rule (1.2) differs from the ideal boundary of the Euclidean distance classifier (1.20). The probability of misclassification of the sample-based rule (the generalisation error) is larger than that of the ideal rule.

Example 5. The increase in the generalisation error due to estimation of the optimal classification boundary using the training data is easy to demonstrate graphically by inspecting the univariate case. In Figure 1.6, we present two Gaussian densities $N_x(m_1, 1)$ and $N_x(m_2, 1)$ with means $m_1 = -m_2 = 1.3$ and unit variances. In the univariate case, we must define only one parameter of the classification rule and that is the threshold v_0. For the ideal rule $v_0^* = ½(m_1 + m_2) = 0$. Therefore, we allocate the observation x to the first class, if $x > 0$, and to the second one, if $x \leq 0$. A part of the observations is classified incorrectly. In Figure 1.6, the probability of misclassification is proportional to the two vertically dashed areas in the centre of the figure. For a randomly chosen training-set, the threshold of the sample-based classification rule $\hat{v}_0 = -(\hat{m}^{(1)} + \hat{m}^{(2)})/2$ will always differ from v_0^* (it can coincide with the ideal threshold v_0^* only incidentally). Thus PMC

$$\varepsilon_N^{\mathrm{E}} = P_1 \, Probability\{\, x < \hat{v}_0 \mid x \in \omega_1\} + P_2 \, Probability\{\, x \geq \hat{v}_0 \mid x \in \omega_2\}, \quad (1.22)$$

which is larger than the PMC of the ideal rule.

The increase in the PMC, $\Delta\varepsilon$, is proportional to the black triangle in Figure 1.6. This increase is always positive. Thus, any shift from the ideal threshold v_0^* will always increase the classification error or PMC.

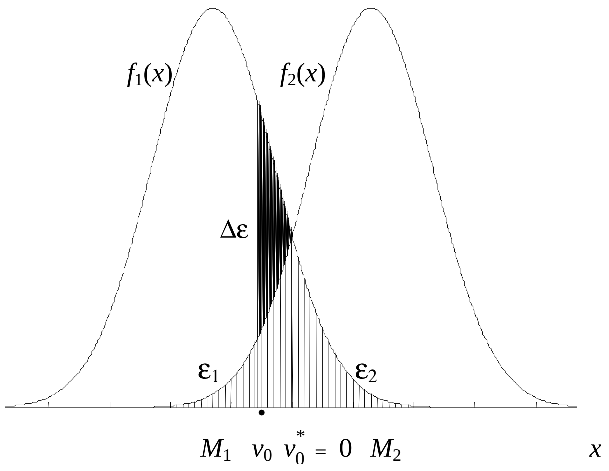

Fig. 1.6. Two univariate Gaussian pattern-class densities $N_x(M_1, 1)$ and $N_x(M_2, 1)$ and the asymptotic errors ε_1 and ε_2 (two dashed areas). The increase in the conditional error, $\Delta\varepsilon$, is marked by the solid black area.

The probability ε_N^E is conditioned on a particular training-set. Therefore, we call the probability ε_N^E the *conditional probability of misclassification* or the *conditional generalisation error.* In statistical analysis, the training vectors $\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_2^{(1)}, \boldsymbol{X}_3^{(1)} \dots, \boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{X}_1^{(2)}, \dots, \boldsymbol{X}_{N_2}^{(2)}$ are considered as independent random vectors. Then the sample mean vectors $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$ may be considered to be random vectors, as well. In this context, the conditional probability of misclassification ε_N^E is also a random variable. We call an expected value, $\bar{\varepsilon}_N^E$, the *expected probability of misclassification* or, simply, an *expected (mean) generalisation error.*

For the model of two spherical Gaussian classes $N_X(\boldsymbol{M}_1, \bar{\Sigma})$, $N_X(\boldsymbol{M}_2, \bar{\Sigma})$, the sample means $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$ are Gaussian: $\hat{\boldsymbol{M}}_1 \sim N(\boldsymbol{M}_1, \frac{1}{N_1}\bar{\Sigma})$, $\hat{\boldsymbol{M}}_2 \sim N(\boldsymbol{M}_2, \frac{1}{N_2}\bar{\Sigma})$. For this data model, and a case of equal training sample sizes, ($N_2 = N_1 = \bar{N} = N/2$), the expected generalisation error of the EDC is

$$\bar{\varepsilon}_N^E \approx \Phi\left\{ -\frac{\delta^*}{2} \frac{1}{\sqrt{T_{\boldsymbol{M}}}} \right\}, \tag{1.23}$$

where δ^* is the effective distance between two pattern classes (defined in (1.21)),

$$T_{\boldsymbol{M}} = 1 + \frac{4n^*}{\delta^{*2} N}, \text{ and} \tag{1.24}$$

$$n^* = \frac{((\boldsymbol{M}_1 - \boldsymbol{M}_2)^T (\boldsymbol{M}_1 - \boldsymbol{M}_2))^2 (\mathrm{tr}\overline{\Sigma}^2)}{((\boldsymbol{M}_1 - \boldsymbol{M}_2)^T \overline{\Sigma} (\boldsymbol{M}_1 - \boldsymbol{M}_2))^2} \tag{1.25}$$

is called the *effective dimensionality*.

Equations (1.23) and (1.24) show that $\overline{\varepsilon}_N^{\mathrm{E}} \geq \varepsilon_\infty^{\mathrm{E}}$. Thus, the expected error $\overline{\varepsilon}_N^{\mathrm{E}}$ approaches the asymptotic error $\varepsilon_\infty^{\mathrm{E}} = \Phi\{-½\delta^*\}$ as the training-set size N increases without bound. The increase in the expected generalisation error, $\Delta\overline{\varepsilon}_N^{\mathrm{E}} = \overline{\varepsilon}_N^{\mathrm{E}} - \varepsilon_\infty^{\mathrm{E}}$, is a function of N^{-1}.

For the moment, consider the spherical Gaussian case where $\overline{\Sigma} = \sigma^2\mathbf{I}$. Note that, for this data model, the EDC is an asymptotically optimal classifier. Also, note that $\delta^* = \delta$ and $n^* = n$ and, therefore, $T_{\boldsymbol{M}} = 1 + \frac{4n}{\delta^2 N}$. We see that the increase in the expected generalisation error depends on the asymptotic error rate and is proportional to n/N, the dimensionality divided by the training-set size. For fixed N and δ, the generalisation error increases as n increases.

Consider now the more general case with $\overline{\Sigma} \neq \sigma^2\mathbf{I}$. Then $n^* \neq n$, where $1 < n^* \leq \infty$. Thus, in theory, pattern-class densities exist where the EDC is very insensitive to the training-set size (when n^* is close to 1) and pattern-class densities exists where the EDC is very sensitive to the training-set size (when n^* is very large). This is a very important conclusion that is utilised in the further analysis of certain neural network classifiers in later chapters.

One can readily see that the Fisher discriminant function is more complex than the Euclidean distance classifier. To design the EDC one needs to estimate only the mean vectors. To design the Fisher linear DF, one needs to estimate the mean vectors μ_1 and μ_1 and the covariance matrix Σ. The sample-based estimator $\hat{\Sigma}$ causes greater fluctuations in the coefficients v_0 and $\boldsymbol{V}^{\mathrm{F}}$ of the Fisher linear DF than in the coefficients of the sample EDC. For the GCCM data model, the estimator $\hat{\Sigma}$ is a Wishart random matrix (see e.g. Anderson, 1958). The expected generalisation error of the Fisher linear DF is

$$\overline{\varepsilon}_N^{\mathrm{F}} \approx \Phi\left\{ -\frac{\delta}{2} \frac{1}{\sqrt{T_{\boldsymbol{M}} T_{\overline{\Sigma}}}} \right\}, \tag{1.26}$$

where $T_{\overline{\Sigma}} = 1 + \frac{n}{N-n}$.

The term $T_{\overline{\Sigma}}$ arises due to the estimation of the covariance matrix $\overline{\Sigma}$. Comparison of (1.23) and (1.26) demonstrates that in the spherical Gaussian case when $\overline{\Sigma} = \sigma^2\mathbf{I}$, the more complex Fisher linear DF is more sensitive to the training-set size than the Euclidean distance classifier. The term $T_{\overline{\Sigma}}$ explicitly shows the price we pay in accuracy because we must estimate the common pattern-class covariance matrix (n variances and $n(n-1)/2$ correlations). Note that the difference between the two linear classifiers is more noticeable when N, the training-set size, is close to n.

The theoretical results considered in (1.26) are important for the design and application of neural network classifiers. In adaptive SLP training, the term $T_{\overline{\Sigma}}$ shows the cost involved in order to move the decision boundary E towards the boundary F in Figure 1.2. Also, this term shows that if N is close to n after minimising the cost function, the SLP classifier can acquire poor small sample properties (curse of dimensionality).

These two theoretical facts lead us to conclude that in the small training-set case, we need to train the SLP fewer iterations than in the large sample case. We wish to reemphasise two important facts concerning SLP training:

- after the first iteration, we can have an EDC classifier that evolves toward the Fisher classifier as training continues.
- in general, the Fisher classifier requires a larger training-set than the EDC.

1.5 Optimal Complexity – The Scissors Effect

The term T_{Σ} shows the amount of training data that is required in order to move from boundary E towards boundary F in Figure 1.2. This amount can become quite large if n, the dimensionality, is large. Thus, in the small training-set case, it is somewhat questionable as to whether or not one should attempt to move the boundary from E to F by continued training. Possibly, there are times one should stop the perceptron training process before one attempts to move the boundary through further training. The main emphasis in this book is to address these types of issues. Later, we explain that the answer to the boundary-movement question depends on the training-set size, the complexity of the problem, and the data dimensionality.

Consider the case when EDC and the Fisher DF are used to classify two equiprobable non-spherical Gaussian classes $N_X(\boldsymbol{M}_1, \overline{\Sigma})$ and $N_X(\boldsymbol{M}_2, \overline{\Sigma})$. When all parameters are known, the optimal Bayes classifier function is Fisher's linear DF

$$h(\boldsymbol{X}) = (\boldsymbol{X} - ½\,(\boldsymbol{M}_1 + \boldsymbol{M}_2))^T\,\overline{\Sigma}^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2).$$

Note that when $\overline{\Sigma}$ is not proportional to **I** (the identity matrix), the asymptotic error rate $\varepsilon_\infty^{\mathrm{E}}$ of the ideal EDC with all parameters known is larger than the Bayes error ε_{B}.

Example 6. Consider a 20-variate Gaussian data model **C** with Mahalanobis distance $\delta = 4.65$ and $\delta^* = 3.7616$ (asymptotic errors $\varepsilon_\infty^{\mathrm{F}} = 0.01$ and $\varepsilon_\infty^{\mathrm{E}} = 0.03$). For the data model **C,** the effective dimensionality $n^* = 20$. Thus, from (1.23) and (1.26), for different training-set sizes $\overline{N}$ we have:

for $\overline{N} = 15$, $\bar{\varepsilon}_N^{\mathrm{E}} \approx 0.0469$, $\bar{\varepsilon}_N^{\mathrm{F}} \approx 0.1094$;

for $\overline{N} = 25$, $\bar{\varepsilon}_N^{\mathrm{E}} \approx 0.0400$, $\bar{\varepsilon}_N^{\mathrm{F}} \approx 0.0441$;

for $\overline{N} = 40$, $\bar{\varepsilon}_N^{\mathrm{E}} \approx 0.0362$, $\bar{\varepsilon}_N^{\mathrm{F}} \approx 0.0259$;

for $\overline{N} = 80$ $\bar{\varepsilon}_N^{\mathrm{E}} \approx 0.0330$, $\bar{\varepsilon}_N^{\mathrm{F}} \approx 0.0163$.

The dependence of the generalisation error of the EDC and the Fisher DF on the training-set size $\overline{N}$ is illustrated graphically in Figure 1.7.

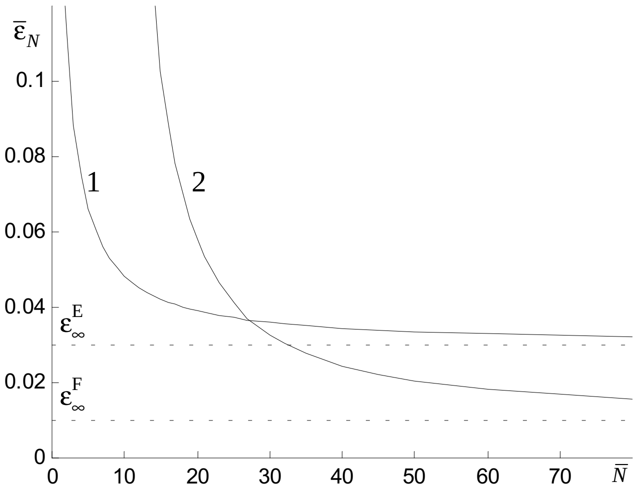

Fig. 1.7. The scissors effect – the generalisation error versus the training-set size $\overline{N}$: 1 is the Euclidean distance classifier, 2 is the standard Fisher linear classifier (adapted from Raudys, 1970).

With the given example from Figure 1.7, we see that in this case, for small training sets (up to $\overline{N} \approx 30$), one would prefer the simple-structured EDC as the classifier of choice. Furthermore, for large training sets (over $\overline{N} \approx 30$), one would prefer the more complex-structured Fisher classifier as the classifier of choice.

This phenomenon, called the scissors effect, has been known in the Statistical Pattern Recognition field for the past 30 years:

> *in small training-set cases, often it is preferable to use simple structured classification rules than the complex ones, and, vice versa, in large training-set cases, complex classifiers can be used more efficiently.*

In Equation (1.26), the generalisation error of the Fisher DF classifier indicates that when *n*, the number of features, is close to *N*, the training-set size, the generalisation error can increase considerably. In practice, the features used for classification have varying degrees of importance. Typically, there is a portion of the original set of variables that have little discriminative power. Moreover, the features are usually correlated and, with the addition of new reinforcing features, the original features can possibly have a decreasing amount of discriminative power. Thus, one may hope that a subset of the original features can be omitted with little or no loss of discriminative power. Therefore, while adding more features with decreasing importance, the generalisation error decreases at first, reaches a minimum, and then increases again. This phenomenon is known as the *peaking effect* and is well known in the field of statistical data analysis.

Example 7. Consider a 20-variate Gaussian data model **C** with correlated variables. For this data model, the Fisher linear DF is the asymptotically optimal classification rule. Let us increase the number of features sequentially, adding the features one after another. In the 20-variate data model **C**, all *n* features have the same discriminative power. The positive correlations between the features causes the added features to contribute less than the previous features to the Mahalanobis distance. Therefore, as the number of the features, *n*, increases, the asymptotic and the Bayes errors decrease quickly at first, but later, the speed of the decrease lessens (solid curves in Figure 1.8). This is a typical situation in real world pattern recognition problems.

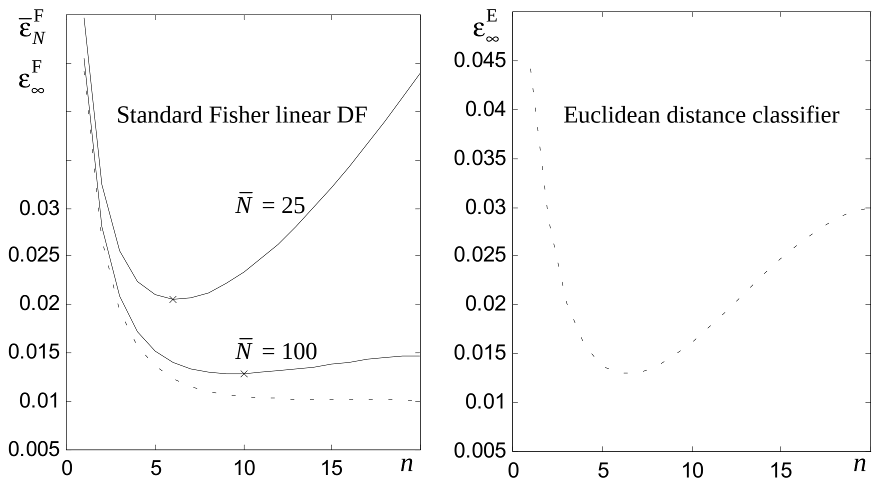

Fig. 1.8. The asymptotic (the dotted curves) and generalisation errors (solid) of the Fisher linear DF and the Euclidean distance classifier as functions of the number of the features *n*.

For the given dimensionality *n*, the generalisation error depends on both the Mahalanobis distance $\delta(n)$ and the training-set size *N*. When the mean vectors $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and the covariance matrix $\bar{\Sigma}$ must be estimated and the training-set size *N* is small, we introduce considerable error (noise) to the unknown components of these parameters. The statistical nature of the training-set vectors causes a certain amount of the "inaccuracy" to be included in the estimators $\hat{\boldsymbol{M}}_1$, $\hat{\boldsymbol{M}}_2$, $\hat{\Sigma}$ and, therefore, to the weight vector $\boldsymbol{V}^F = \hat{\Sigma}^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)$. For a fixed training-set size *N*, the inaccuracy in the estimators $\hat{\boldsymbol{M}}_1$, $\hat{\boldsymbol{M}}_2$, and $\hat{\Sigma}$ increases as the number of features increase. Hence, with the addition of new features, the generalisation error decreases at first, reaches a minimum, and then begins to increase. We obtain the peaking effect early (curve for $\bar{N}$ = 25) when N is small. For large N, the "noise" or inaccuracy in the estimators of $\hat{\boldsymbol{M}}_1$, $\hat{\boldsymbol{M}}_2$, $\hat{\Sigma}$ is less. Thus, we obtain the peaking effect later (curve for $\bar{N}$ = 100).

In comparison with the Fisher classifier, the less complex EDC requires the estimation of a smaller number of parameters from the training vectors because we estimate only the pattern-class means $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ and not the CM $\bar{\Sigma}$. Therefore, for the same training-set size, the "noise" level is smaller for the EDC. Usually, while utilising the EDC, we obtain the peaking effect later.

Above, we have explained the peaking phenomenon by the presence of a small training-set size. In reality, the peaking phenomenon can be observed even in the case of an infinite training-set size due to the use of a *non-optimal classifier design procedure*. A discrepancy between the true pattern-class distributions and the hypothetical pattern-class distributions induces error in estimating the parameters of the classification rule. This fact can give rise to the peaking phenomenon even in a case when the number of training samples is infinitely large.

Example 8. The Gaussian data model **C** with correlated features is not an ideal model of the pattern classes for the EDC. Theoretically, the addition of the new features can diminish the distance measure δ^* instead of increasing it. Therefore, in this data model with equally correlated features, *the asymptotic error of the EDC has a minimum* because of a peaking effect (Figure 1.8, right).

One must concede that the configuration model **C** is probably not a typical example of real world pattern-classification problems. This model, however, nicely explains one more factor that can cause the peaking phenomenon. One may conjecture that for some specific pattern classes (e.g. multimodal), a similar phenomenon can be observed in the SLP design.

The number of features where the conditional generalisation error $\varepsilon_N^{\mathsf{A}}$ reaches its minimum is called the *optimal number of features.* In practical applications, the curves $\varepsilon_N^{\mathsf{A}}$ = $f(n)$ may have more than one minimum. The character of the curve $\varepsilon_N^{\mathsf{A}}$ = $f(n)$ depends on the type of classification rule A (the complexity of the classifier), the pattern recognition problem to be solved, the order in which the features are presented, and the training-set size. The optimal number of features increases as the number of the training observations increases and decreases as the complexity of the classifier increases.

An important problem encountered when designing pattern classification systems is **the feature definition problem**. We have demonstrated above that in order to obtain a classifier with good generalisation performance, one needs to use a small number of effective (informative) features. A number of methods have been developed to help select the best subset of features from the totality of previously selected ones. This problem is called the feature subset selection problem and it has much in common with pruning and incremental learning algorithms for neural networks.

While selecting the best subset of features, one usually performs this type of work with a certain inaccuracy. Therefore, the order in which the features are presented may differ from the ideal ordering. Instead of having one minimum, one can have multiple minima. Moreover, if the features or, more strictly speaking, the subsets of features are ranked poorly, then we cannot observe a minimum. Statistical analysis can help evaluate these effects numerically. We consider these problems in Chapter 6.

1.6 Overtraining in Neural Networks

In perceptron training, we minimise the cost function to determine the weights. The cost value is closely related to the empirical classification error. At the beginning of the training process, the weights are small and the values of the weighted sums $v_0 + \boldsymbol{V}^T\boldsymbol{X}_j^{(i)}$ are also small. Around the zero point the activation function acts almost as a linear function. Thus the cost function approximately characterises the empirical error. When the weights are large, the weighted sums $v_0 + \boldsymbol{V}^T\boldsymbol{X}_j^{(i)}$ are also large. Then the smooth activation function acts as the hard limiting threshold function and the cost (1.7), in essence, represents the empirical classification error.

With an increase in the number of iterations, the empirical classification error customarily decreases at first. The generalisation error also initially decreases. After it reaches a minimum, it then begins to increase. This phenomenon is called overtraining or overfitting and has the same origin as the scissors and the peaking effects.

Theoretical analysis of the generalisation error and of the dynamics in the SLP classifier training reveals reasons for this phenomenon and indicates methods to utilise this fact in the best possible way. Below, we present and comment on two examples that are analysed more thoroughly in Chapter 4.

Example 9. In this example we show that in the spherically Gaussian (SG) data case, only *one iteration* is necessary to train the SLP. In order to demonstrate this fact effectively, we have chosen data with high dimensionality (n=100) and a comparatively small training-set size ($\bar{N}$ = 200 from each pattern class). Let the Mahalanobis distance be δ =3.76. The asymptotic and Bayes error rates are $\varepsilon_\infty^{\mathrm{E}} = \varepsilon_\infty^{\mathrm{F}} = \varepsilon_{\mathrm{B}} = 0.03$. The theory (Equations (1.23) and ((1.26)) indicates that for $\bar{N}$ = 200 the EDC will yield $\bar{\varepsilon}_N^{\mathrm{E}} = 3.48\%$ errors on average, while the Fisher classifier will yield $\bar{\varepsilon}_N^{\mathrm{F}} = 5.79\%$ errors on average. Thus, in the batch-mode, we obtain the best classifier (EDC) immediately after the first iteration. There is no need to further train the perceptron. Recall that in order to obtain the EDC after the first iteration of the training process, the training conditions E1 – E4 should be satisfied. In the present experiment, we satisfied these conditions and trained the SLP with the sigmoid activation function (1.6) and targets $t_j^{(1)} = 1$, $t_j^{(2)} = 0$.

The learning curve, the generalisation error versus the number of iterations, is presented in Figure 1.9 (curve 2). We see after the first iteration that $\varepsilon_N^{\text{SLP}} = 0.034$ is quite close to $\bar{\varepsilon}_N^{\text{E}} = 0.0348$ predicted by the theory. With an increase in the number of iterations in the training process, the decision boundary of the SLP classifier approaches that of the Fisher DF and later it approaches the maximum margin classifier. For both classifiers we have a much higher generalisation error than for the EDC.

In the second part of this experiment, we violated one of the conditions E1 – E4. Specifically, we began training from a random initial weight vector generated by a random Gaussian N_v(0. 0.1) generator. After the first few iterations, we get a comparatively high generalisation error (it is too high to be depicted in Figure 1.9) that decreases until reaching the minimum (approximately after 50 iterations) and then begins increasing. After 100 iterations there is no difference between the two learning curves obtained for differently initialised SLPs. The expert initialisation of $\boldsymbol{V}_{(0)} = \mathbf{0}$ helped reduce the minimal classification error substantially.

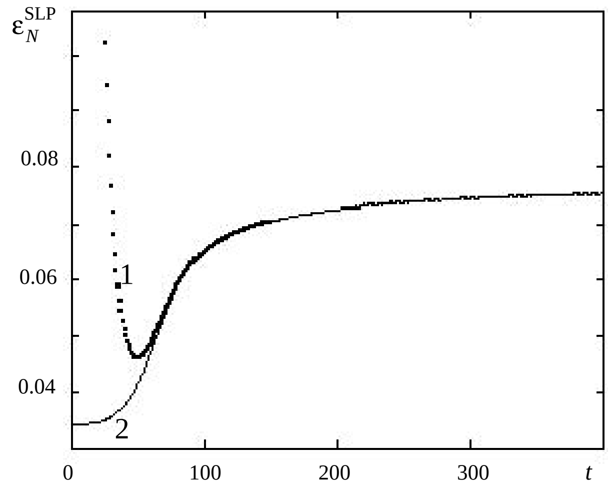

Fig. 1.9. The generalisation error versus t, the number of iterations. Spherical Gaussian classes with $\delta = 3.76$ and $n = 100$, $\bar{N} = 200$. We used the varying learning step $\eta = 1.1^t$ and random $N_x(0, \sigma^2)$ weight initialisation: 1 - $\sigma^2 = 0.1$, 2 - $\sigma^2 = 0.0$ (Reprinted from *Neural Networks*, 11:297-313, S. Raudys, Evolution and generalization of a single neurone, 1998, with permission from Elsevier Science).

Example 10. While solving real world problems, one is unlikely to encounter spherical data. One can expect that in the complex data case, one must train the perceptron more than one iteration. In order to obtain the best result we must control the training process in an intelligent way. One example of "intelligent" training is demonstrated in the learning curves depicted in Figure 1.10. Here, we have used the 20-variate artificial data **C** with correlated features and, while training SLP, we have satisfied conditions E1– E4. In this example, we have a very small training-set – only 14 vectors from each pattern class. In the previous section, we had that for $\bar{N} = 15$, $\bar{\varepsilon}_N^{E} \approx 0.0469$ and $\bar{\varepsilon}_N^{F} \approx 0.1094$. Thus, the sample size $\bar{N} = 14$ is too small to train the Fisher classifier.

For data **C** and $\bar{N} = 14$, we should avoid obtaining this classifier. We can get the Fisher classifier or a classifier close to it by controlling the magnitudes of the weights during the SLP training. If the weights are small, the activation function acts almost like a linear function. In Chapter 4, we consider several methods that can help control the weight magnitudes. One such method is the selection of proper target values. Using proximal target values, one does not allow the weights to become too large and, therefore, forces the classifier to approach the Fisher classifier. Thus, a priori, we see that proximal targets and lengthy training can lead to undesirable results. The learning curve 1 in Figure 1.10 corresponds to targets $t_j^{(1)} = 0.9$ and $t_j^{(2)} = 0.1$ (the values recommended by Rumelhart *et al.*, 1986) and confirms this presumption.

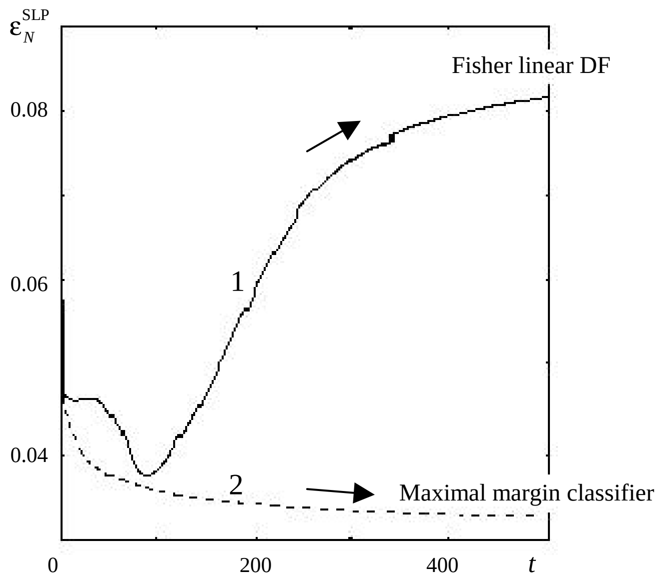

Fig. 1.10. The generalisation error versus t, the number of iterations: 1 - targets $t_1 = 0.9$, $t_2 = 0.1$, 2 - $t_1 = 1$, $t_2 = 0$. The data model **C** with a training-set size of $\bar{N} = 14$ (Reprinted from *Neural Networks*, 11:297-313, S. Raudys, Evolution and generalization of a single neurone, 1998, with permission from Elsevier Science).

In the second demonstration, we use the boundary target values $t_j^{(1)} = 1$, $t_j^{(2)} = 0$. In this case, after a few iterations the empirical error becomes equal to zero and the weights begin increasing. Initially, we are approaching the minimum empirical error classifier and then we move towards the maximal margin classifier. In this extremely small training-set size case, the minimum empirical error classifier is much better than the parametric Fisher classifier (curve 2 in Figure 1.10).

The two examples just presented illustrate that the training conditions greatly affect the resulting neural net classifier. While solving real world problems, we know neither the data model nor its parameters. Therefore, the usefulness of theoretical error estimates is less effective. Nevertheless, in some cases, we can successfully determine an appropriate classifier. In the following chapters, we demonstrate the utility of the theoretical results to the classifier design problem.

Theoretical analysis suggests that good initial weights and optimal stopping of the SLP training process are very useful in reducing the generalisation error. While working with real world data, it is worth whitening the data, i.e. initially transforming the data into spherical data. Then we have to move the data centre to the zero point and begin training with zero initial weights. So, after the first iteration we can obtain the EDC, which is a very good initial classifier for spherical data. In further training, we can utilise all the positive properties of the neural network approach. This allows us to obtain comparatively good classification rules if the training-set size is small or the pattern-class models are non-Gaussian. Therefore, we can utilise statistical methods to transform the data in the most advantageous manner. This approach is, in fact, the integration of the statistical and neural network approaches for designing pattern classification algorithms (for more details see Chapter 5).

1.7 Bibliographical and Historical Remarks

The necessity of analysing artificial neural networks using statistical methods was understood long ago. There are a number of books and review papers devoted to this problem, including: Hertz, Krogh and Palmer (1991), Sethi and Jain (1991), Haykin (1999), Michie *et al.* (1994), Ripley (1994), Cheng and Titterrington (1994), Bishop (1995), Vapnik (1995), Schuermann (1996), Vidyasagar (1997), Cherkassky and Mulier (1998), Jain, Duin and Mao (2000). For a more in-depth study of statistical pattern recognition, the author recommends the following monographs: Duda, Hart and Stork (2000), Young and Calvert (1974), Tou and Gonzales (1974), Devijver and Kittler (1982), Aivazian *et. al.* (1989), Fukunaga (1990), Schalakoff (1992), Devroye, Giorfi and Lugosi. (1996). An excellent presentation of many ideas and results pertaining to statistical discriminant analysis and pattern recognition is McLachlan's monograph (1992). However, some connectionists find this book difficult to read.

The first mathematical model of a neurone was proposed as a binary threshold unit. Later this model was generalised by changing the hard-limiting threshold function to the soft-limiting function. Subsequently, multiple neurones were analysed. The neurones were then organised in layers with feedforward connections between layers. For a more detailed review, see Hertz *et al*. (1991), and Haykin (1999). Around 1960, a number of training algorithms for determining the weights of the network were proposed on the grounds of physiological analysis and ad hoc principles. Originally, the first idea was to show the training patterns sequentially and to make the weight change only if the classification error occurred. Later, Tsypkin (1966) and Amari (1967) showed that different adaptive iterative training algorithms can be obtained formally from the condition of the iterative minimisation of a certain cost function. Thus came the discovery that different cost functions and optimisation methods produce distinctly different training algorithms. The standard cost function of the sum of squares with the linear activation function was proposed by Widrow and Hoff (1960) as an adaline algorithm. The "tanh" non-linear soft-limiting function was introduced by Amari (1967) and the cost function (1.7) was suggested by Pietrantonio and Jurs (1972).

The gradient-type back-propagation training algorithm for determining the weights of the multilayer perceptron was discovered in several different research areas of information processing by Tsypkin (1966, 1973), Amari (1967), Bryson and Ho (1969), Werbos (1974), Parker (1985), Le Cun (1986), Rumelhart, Hinton and Williams (1986). A careful review of different optimisation algorithms in neural network training can be found in Van der Smagt (1994). Koford and Groner (1966) were the first to show that the adaline algorithm leads to the standard Fisher linear discriminant function. Raudys (1995*a*, 1996, 1998*b*) has shown that under certain training conditions after the first iteration the SLP leads to the Euclidean distance classifier and later to other six known statistical classifiers. For the regression task, this question has been considered by Sjoberg and Ljung (1992) and later by Raudys (2000*b*).

The small training-set properties of the EDC (formally, the Fisher linear DF with known covariance matrix) were first analysed by John (1961). Raudys (1967) was the first to suggest the analysis of the sample-based discriminant functions using a double asymptotic approach where the number of the training samples N and the number of the features n increase simultaneously (now this method is called the thermodynamic limit approach). Deev (1970, 1972) and Raudys (1972) obtained generalisation error expressions for the Fisher linear DF.

The dimensionality and fixed training-set size effect was first mentioned by Rao (1949). The optimal dimensionality problem for pattern classification was first considered by Allays (1966), Hughes (1965) and Lbov (1966). The "scissors effect" that arises in the application of pattern classification algorithms of different complexity has been formulated by Raudys (1970) and Kanal and Chandrasekaran (1971). Later, under different names, the complexity and sample size effect was rediscovered and popularised in a number of papers and books (Vapnik and Chervonenkis, 1974; Duin, 1978; Jain and Chandrasekaran, 1982; Raudys and Jain, 1991, Geman *et al*., 1992). The graphs in Figure 1.7 for the EDC and the Fisher classifier are taken from Raudys (1970). Utilisation of the data whitening transformations prior to training the perceptron as a means of integrating the statistical and neural network classifier design approaches was first suggested in Raudys (1998*a* and 1998*c*).