2. Taxonomy of Pattern Classification Algorithms

Two or three hundred different pattern classification algorithms have been suggested in literature during the last 50 years. The main objective of this chapter is to review a selection of known statistical algorithms that can be obtained or improved by training ANN-based classification systems. The first selection contains seven statistical algorithms that can be obtained while training linear and non-linear single layer perceptrons and the second selection contains algorithms that can be approached in ANN training after deriving new non-linear features from the original ones. Particular attention is given to methods which can be used to structure the covariance matrices and describe them by a small number of parameters. This approach is not very popular in statistical pattern recognition, however, together with utilisation of neural networks, it becomes a powerful tool to solve problems in small training-set situations.

Principles of statistical decisions, plug-in and Bayesian sample based rules will be presented shortly, explaining why some of the statistical rules are not optimal. Parzen window and *k-NN* non-parametric classifiers will be described to allow radial basic functions and learning vector quantisation networks to be considered. In order to rationalise the co-operation of neural networks we discuss several classifiers for categorical (discrete valued) features. The reader can find more details regarding classification algorithms in Fukunaga (1990), McLachlan (1990), Duda, Hart and Stork (2000) and other texts on the subject.

2.1 Principles of Statistical Decision Theory

In this section, we use the statistical decision function approach and derive the optimal decision rule, which classifies unknown vectors $\boldsymbol{X}$ with the smallest probability of misclassification. Suppose we know that an n-variate observation vector $\boldsymbol{X}$ belongs to one of the two classes, ω_1 and ω_2. In the statistical approach it is assumed that the multivariate observations are random vectors (see e.g. Fukunaga, 1990). Let $p_i(\boldsymbol{X})$ be the class conditional density function of vector $\boldsymbol{X}$ belonging to class ω_i, and let P_i be the a priori probability of the class ω_i. Then using the Bayes Theorem, a posterior probability $P(\omega_i \mid \boldsymbol{X})$ of the vector $\boldsymbol{X}$ to belong to the class ω_i can be expressed as

$$P(\omega_i|\boldsymbol{X}) = \frac{P_i\, p_i(\boldsymbol{X})}{p(\boldsymbol{X})} \quad ,$$

where, in the two category case, $p(\boldsymbol{X}) = P_1\, p_1(\boldsymbol{X}) + P_2\, p_2(\boldsymbol{X})$ is an unconditional probability density function of vector $\boldsymbol{X}$.

A decision rule based on the posterior probabilities classifies an unknown observation vector $\boldsymbol{X}$ into class ω_1 if

$$P(\omega_1|\boldsymbol{X}) > P(\omega_2|\boldsymbol{X})) \tag{2.1}$$

and into the class ω_2 otherwise. Condition (2.1) may be rewritten by using the discriminant function (DF)

$$h_o(\boldsymbol{X}) = \frac{P_1\, p_1(\boldsymbol{X}) - P_2\, p_2(\boldsymbol{X})}{p(\boldsymbol{X})} \quad . \tag{2.2}$$

Hence, the **a posteriori probability decision rule** is: to classify an unlabeled observation $\mathbf{X}$ into class ω_1 if $h_o(\mathbf{X}) > 0$ and into class ω_2 otherwise.

Now we can demonstrate that this classification rule is optimal in a sense of yielding the *minimum probability of misclassification* (PMC). The PMC may be expressed as

$$\varepsilon = P_1 \int_{\Omega 2} p_1(\boldsymbol{X})\, d\boldsymbol{X} + P_2 \int_{\Omega_1} p_2(\boldsymbol{X})\, d\boldsymbol{X}, \tag{2.3}$$

where Ω_1 and Ω_2 splits the measurement space Ω into two nonintersecting parts: if the observation $\boldsymbol{X} \in \Omega_i$, then it is classified into population ω_i, $i = 1, 2$.

Since Ω_1 and Ω_2 are disjoint, and $\Omega_1 \cup \Omega_2 = \Omega$, we have that

$$\int_{\Omega_1} p_1(\boldsymbol{X})\, d\boldsymbol{X} + \int_{\Omega 2} p_2(\boldsymbol{X})\, d\boldsymbol{X} = 1.$$

Therefore, Equation (2.3) may be rewritten as

$$\varepsilon = P_1 \left[1 - \int_{\Omega_1} p_1(\boldsymbol{X})\, d\boldsymbol{X}\right] + P_2 \int_{\Omega_1} p_2(\boldsymbol{X}) d\boldsymbol{X}$$

$$= P_1 - \int_{\Omega_1} [P_1\, p_1(\boldsymbol{X}) - P_2\, p_2(\boldsymbol{X})]\, d\boldsymbol{X}. \tag{2.4}$$

From expression (2.4) we see that the probability of misclassification will be minimised by assigning observation $\boldsymbol{X}$ to ω_1 only if discriminant function

$$h(\boldsymbol{X}) = P_1\, p_1(\boldsymbol{X}) - P_2\, p_2(\boldsymbol{X}) \tag{2.5}$$

is positive.

Since $p(\boldsymbol{X})$ defined in (2.2) is always positive, we see that both discriminant functions, (2.2) and (2.5), are equivalent. Because $P_i\ p_i(\boldsymbol{X})$ is positive, the discriminant function which yields the minimum error rate can be expressed as

$$h^*(\boldsymbol{X}) = \log p_1(\boldsymbol{X}) - \log p_2(\boldsymbol{X}) + \log \frac{P_1}{P_2}. \tag{2.6}$$

Figure 1.6, presented in the previous chapter, illustrates the decision making scheme. Two curves $p_i(\boldsymbol{X})$ represent densities multiplied by the prior probabilities $P_1 = P_2 = 0.5$. The total classification error consists of two parts:

ε_1 (the error of the first kind) is the probability that any object from ω_1 is allocated to ω_2; and

ε_2 (the error of the second kind) is the probability that any object from ω_2 is allocated to ω_1.

The minimum of the classification error sum is obtained when we select the "decision threshold" v_0^* at a point where $P_1\, p_1(\boldsymbol{X}) = P_2\, p_2(\boldsymbol{X})$. The use of any other "decision threshold" will lead to higher sum classification error rates (in Figure 1.6, the area $\Delta\varepsilon$ appears). In the multiclass case, the minimum of the classification error sum (and the maximal probability of correct classification) is obtained if one makes the classification according to a maximum of the product $P_i\, p_i(\boldsymbol{X})$, $(i = 1, \ldots, L)$.

A further generalisation of the decision making process occurs, if we assume that the costs of incorrect decisions of the first kind and of the second kind are not equal among themselves. Let $C_{1/2}$ be the cost of incorrect decisions of the second kind, $C_{2/1}$ be the cost of incorrect decisions of the first kind, then the competitive losses of the two types of decisions are $C_{1/2}P_1\, p_1(\boldsymbol{X})$ and $C_{2/1}P_2\, p_2(\boldsymbol{X})$. Thus, in order to minimise the sum of losses for each $\boldsymbol{X}$, one needs to select a solution with a minimal loss

$$\min_i \{ \sum_{j \neq i} C_{i/j} P_i\, p_i(\boldsymbol{X}) \}.$$

It is easy to generalise the above solution for the multiclass problem and for the case when we would like to have one more decision – to refuse classification in ambiguous cases. In the latter case, one needs to introduce an additional rejection class and to take into account the additional analysis and computation costs involved (see e.g. Fukunaga, 1990, chapter 3).

In a limit case, one can choose one of the costs $C_{1/2}$ or $C_{2/1}$ to be extremely large. Then in a decision-making process, we minimise only one kind of probability of misclassifications.

Expressions (2.2), (2.5) and (2.6) are the most commonly used forms to compose the discriminant function in the statistical pattern recognition. These classification rules are optimal in the sense of yielding the minimum error rate (this is a risk function, if we use the loss instead of using the probability of misclassification).

In order to implement such classification rules, one must know a priori of probabilities P_1 and P_2 and the class-conditional densities $p_1(\boldsymbol{X})$ and $p_2(\boldsymbol{X})$. In most practical problems, the models generating the observation vectors $\boldsymbol{X}$ are unknown and the density functions $p_1(\boldsymbol{X})$ and $p_2(\boldsymbol{X})$ have to be estimated using the training sets. Both parametric (e.g., multivariate Gaussian densities) and nonparametric (e.g., Parzen window, *k*-nearest neighbour approach) estimation methods can be utilised for the density estimation problem.

In the **parametric approach**, a functional form of the densities $p_i(\boldsymbol{X})$ is supposed to be known up to an unknown parameter vector $\mathbf{Y}_i$, i.e. $p_i(\boldsymbol{X}) = p_i(\boldsymbol{X} \mid \mathbf{Y}_i)$.

Example 1. One assumes the *n*-variate vector $\boldsymbol{X}$ is multivariate Gaussian with independent components. Its distribution density function is

$$p_i(x_1, x_2, ..., x_n) = \prod_{j=1}^{n} (2\pi)^{-1/2} \sigma_{ij}^{-1} e^{-1/2 (x_j - m_{ij})^2 / \sigma_{ij}^2}. \qquad (2.7)$$

Here, the 2*n*-variate parameter vector $\mathbf{Y}_i$ is composed from $m_{i1}, m_{i2}, \ldots, m_{in}$ components of the mean vector, and $\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{ip}$ components of variances (one for each feature). Thus, $\mathbf{Y}_i = (m_{i1}, m_{i2}, \ldots, m_{in}, \sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{ip})^T$. One of the possibilities for the classification rule design is to estimate vectors $\mathbf{Y}_1$ and $\mathbf{Y}_2$ from the training data and to insert the estimate obtained into the expressions for the densities $p_i(x_1, x_2, ..., x_n)$ and the discriminant function (2.6). For the Gaussian classes just discussed and $P_1 = P_2 = ½$ we obtain following quadratic discriminant function

$$h^*(\boldsymbol{X}) = \sum_{j=1}^{n} [- ½ (x_j - \hat{m}_{1j})^2 s_{1j}^{-2} + ½ (x_j - \hat{m}_{2j})^2 s_{2j}^{-2} + ½ \ln \frac{s_{2j}}{s_{1j}}], \quad (2.8)$$

where $\hat{m}_{i1}, \hat{m}_{i2}, \ldots, \hat{m}_{in}$ are estimates of the components of the mean vector $\boldsymbol{M}_i$ and $s_{i1}, \ldots, s_{in}$ are estimates of variances of each single feature in the *i*-th class:

$$\hat{m}_{ij} = 1/N_i \sum_{t=1}^{N_i} x_{jt}^{(i)}, \quad s_{ij} = 1/(N_i - 1) \sum_{t=1}^{N_i} (x_{jt}^{(i)} - \hat{m}_{ij})^2,$$

$x_{jt}^{(i)}$ is the j-th component of training vector $\boldsymbol{X}_t^{(i)}$ $i = 1, 2$; $t = 1, 2, \ldots, N_i$ and N_i is a number of training vectors from the i^{th} pattern class.

Utilisation of the sample estimates instead of the true parameters in Expression (2.6) is called a **plug-in approach** to designing the classification rule. It is the most widely used technique to design the statistical classifiers. In chapter 3 we will see that in case $N_1 >> N_2$ (or *vice versa*), this approach does not minimise the probability of misclassification and does not lead to optimal use of the training-set observations to design the classification rule.

An alternative to the plug-in technique is a **Bayes predictive approach** that theoretically leads to optimal classification rules. A cardinal assumption of the Bayes approach is an assumption that the classifier designer knows a priori of distributions $p_{prior}(\mathbf{Y}_i)$ of the parameter vectors $\mathbf{Y}_1$, $\mathbf{Y}_2$. It is a *very strong assumption*. Often it cannot be satisfied in practical situations. An incorrect definition of $p_{prior}(\mathbf{Y}_1, \mathbf{Y}_2)$ leads to an increase in the classification error. Hence, in statistical pattern recognition, the predictive Bayes approach did not gain a wide recognition. In recent years, this approach has obtained considerable attention in the analysis of regularisation processes in ANN training. Therefore, in Section 2.4, we present some details and a small discussion.

There exist a great variety of mathematical models to parameterise multivariate distribution density functions $p_i(\boldsymbol{X})$ and to estimate those parameters. Also there is a number of **nonparametric estimation methods** such as the Parzen window (PW) and the *k*-nearest neighbour (*k-NN*) rules. Each of these models or methods can be used in expressions (2.2), (2.5) and (2.6) and results in a specific classification algorithm. In principle, all of them can be called Bayes algorithms, since they are designed using the Bayes formula, and statistical decision function approach. However, it is approximately true, since all the training-set based estimates of the densities $p_i(\boldsymbol{X})$ are not the true densities.

2.2 Four Parametric Statistical Classifiers

In Section 1.2, we introduced two popular statistical classifiers, the Euclidean distance and the standard Fisher linear DF classifiers. We have shown that these classification rules can be obtained in the adaptive SLP training. Now we are able to show that these classifiers can be obtained by applying the principles of the statistical decision functions discussed in the previous section.

2.2.1 The Quadratic Discriminant Function

In statistical pattern recognition, most often the pattern vectors are supposed to be multivariate Gaussian. The multivariate Gaussian density has the form

$$p_i(\boldsymbol{X}) = N_X(\boldsymbol{X}, \boldsymbol{M}_i, \Sigma_i) = (2\pi)^{-n/2} \mid \Sigma_i \mid^{-1/2} \mathrm{e}^{-1/2 (\boldsymbol{X}- \boldsymbol{M}_i)^T \Sigma_i^{-1} (\boldsymbol{X}- \boldsymbol{M}_i)} \tag{2.9}$$

where $\boldsymbol{M}_i$ is an n-variate mean vector of the i-th pattern class and Σ_i is an $n \times n$ symmetric, positive definite covariance matrix (CM). The symmetric covariance matrix, $\Sigma = ((\sigma_{ij})) = ((\sigma_i \sigma_j \rho_{ij}))$, characterises variances σ_i^2, and correlations ρ_{ij} between the components of the feature vector $\boldsymbol{X}$.

For the multivariate Gaussian vectors, the maximum likelihood estimators of $\boldsymbol{M}_i$ and Σ_i are

$$\hat{\boldsymbol{M}}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{X}_j^{(i)}, \tag{2.10a}$$

$$\hat{\Sigma}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)(\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)^T. \tag{2.10b}$$

Inserting estimates (2.10*a* and 2.10*b*) into the density (2.9) and then into (2.6) yields the following sample-based plug-in *quadratic discriminant function* Q:

$$\hat{h}^{\mathrm{Q}}(\boldsymbol{X}) = (\boldsymbol{X} - \hat{\boldsymbol{M}}_2)^T \hat{\Sigma}_2^{-1} (\boldsymbol{X} - \hat{\boldsymbol{M}}_2) - (\boldsymbol{X} - \hat{\boldsymbol{M}}_1)^T \hat{\Sigma}_1^{-1} (\boldsymbol{X} - \hat{\boldsymbol{M}}_1) + \ln \frac{|\Sigma_2|}{|\Sigma_1|} + \ln \frac{P_1}{P_2}. \tag{2.11}$$

This rule often is called a standard quadratic DF. It forms the second-order decision boundary in the multivariate feature space. The MLP with the soft-limiting activation function also can form similar non-linear boundaries.

Example 2. In Figure 2.1, by "crosses" and "and small pluses", we depict 500+500 bi-variate vectors from two spherical Gaussian populations and the non-linear decision boundaries of the quadratic DF (boundary Q) and MLP with two hidden neurones (boundary MLP). In spite of visible difference in decision boundaries, both classifiers result in approximately 5½% error.

2.2.2 The Standard Fisher Linear Discriminant Function

Suppose both pattern classes share a common covariance matrix $\overline{\Sigma} = \Sigma_1 = \Sigma_2$, and let $P_2 = P_1$. Then inserting sample estimates of the mean vectors and the covariance matrix into Equation (2.11), and omitting the term $\ln(P_1/P_2)$, we obtain the Fisher linear discriminant function (1.3), defined in Section 1.1. This classification rule is historically the first one. It was proposed by Ronald Fisher (1936), where he mapped training vectors $\boldsymbol{X}_j^{(1)}$ from class ω_1 and vectors $\boldsymbol{X}_j^{(2)}$ from class ω_2 on a straight line and searched a direction of this line in order to get the best separability of the classes. To define the "separability", he used a measure

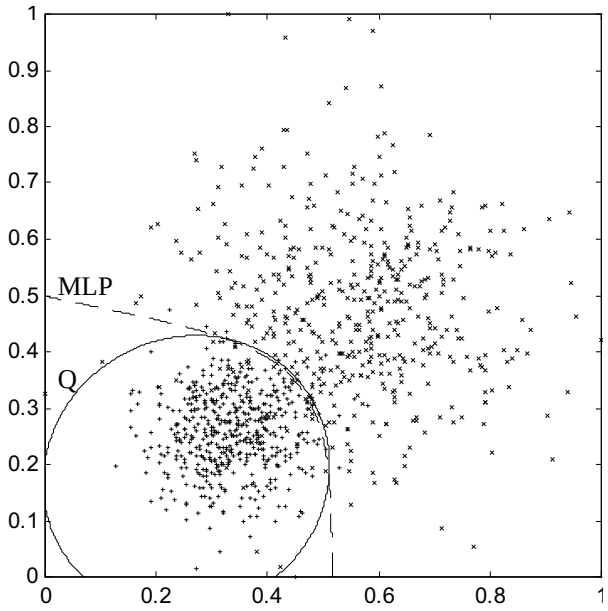

Fig. 2.1. 200 bi-variate vectors from two Gaussian classes $N_X((-1, -1)^T, \mathbf{I})$, $N_X((2, 2)^T, 4\mathbf{I})$ and the non-linear decision boundaries of the quadratic DF and the MLP with 2 hidden neurones.

$$T^2 = \frac{[\bar{h}_1(\boldsymbol{X}) - \bar{h}_2(\boldsymbol{X})]^2}{\hat{V}[h_1(\boldsymbol{X})] + \hat{V}[h_2(\boldsymbol{X})]} = \left(\frac{\boldsymbol{V}^T \hat{\boldsymbol{M}}_1 - \boldsymbol{V}^T \hat{\boldsymbol{M}}_2}{\sqrt{\boldsymbol{V}^T \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{V}}} \right)^2, \tag{2.12}$$

where $\bar{h}_i = \boldsymbol{V}^T \hat{\boldsymbol{M}}_i + v_0$ is a sample mean value of DF, $h(\boldsymbol{X}) = \boldsymbol{X}^T \boldsymbol{V}^F + v_0$, evaluated from all training vectors from class ω_i, and $\hat{V}[h_i(\boldsymbol{X})]$ is a sample variance.

The measure (2.12) is a difference in the sample means of the projections divided by a standard deviation. Maximisation of (2.12) gives the weights of the standard linear Fisher DF defined in Equation (1.3). Illustration of the above mapping have been presented in Figure 1.2.

2.2.3 The Euclidean Distance Classifier

While designing the standard linear DF, it was supposed that both classes share a common covariance matrix $\bar{\Sigma}$. Suppose now that $\bar{\Sigma}$ is proportional to the identity matrix $\mathbf{I}$, i.e. $\bar{\Sigma} = \sigma^2 \mathbf{I}$ (we suppose the variables to be uncorrelated and have identical variances). For equal prior probabilities of the classes, we classify according to the sign of the discriminant function. Then the parameter σ^2 becomes unimportant. As a result we obtain the Euclidean distance classifier with the weights defined by the Equation (1.2).

2.2.4 The Anderson–Bahadur Linear DF

The linear DF is preferable when the covariance matrices of the different classes are equal among themselves. When the covariance matrices differ radically, it is preferable to use the quadratic discriminant function (2.11). There is an alternative to the quadratic DF suggested by Anderson and Bahadur (1962). In this classifier, the estimates of the covariance matrices are weighted by factors γ and $1 - \gamma$. The weight vector has the form:

$$\boldsymbol{V}^{\mathrm{AB}} = \left(\gamma\, \hat{\boldsymbol{\Sigma}}_1 + (1-\gamma)\, \hat{\boldsymbol{\Sigma}}_2\right)^{-1} (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2), \tag{2.13}$$

where factor γ is a certain function of $\hat{\boldsymbol{M}}_1, \hat{\boldsymbol{M}}_2$ and $\hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2$.

We denote this classifier as AB. This rule approximates the non-linear decision boundary of the quadratic DF by a certain plane. A difference between these decision boundaries is most important in the area where the both pattern classes intersects. Often (but not always) the AB rule results in a close performance compared to the quadratic DF.

Example 3. In Figure 2.2 we depict a major part of bi-variate Gaussian training vectors and decision boundaries for four types of parametric classifiers: E − the Euclidean distance classifier, F − the standard Fisher linear DF, Q − the standard quadratic DF and AB − the *Anderson–Bahadur linear DF*. In spite of the difference in the covariance matrices of the classes, the AB and Q classifiers result in a similar classification performance. Below we present the training and test errors and their average: an estimate of the asymptotic classification error:

$$\hat{\varepsilon}_{\infty}^{\mathrm{E}} = (0.027 + 0.031)/2 = 0.029,$$
$$\hat{\varepsilon}_{\infty}^{\mathrm{F}} = (0.024 + 0.031)/2 = 0.027,$$
$$\hat{\varepsilon}_{\infty}^{\mathrm{Q}} = (0.008 + 0.008)/2 = 0.008,$$
$$\hat{\varepsilon}_{\infty}^{\mathrm{AB}} = (0.009 + 0.019)/2 = 0.014.$$

While choosing the distribution parameters for this illustration, efforts have been made to provide a large difference between $\varepsilon_{\infty}^{\mathrm{Q}}$ and $\varepsilon_{\infty}^{\mathrm{AB}}$.

2.3 Structures of the Covariance Matrices

In Section 1.4, we have seen that the ratio between the number of features and the number of training examples is essential in determining the generalisation abilities of the statistical classifiers. In the two-category case, for the linear DF, we have to estimate $2n$ components of the mean vectors and $n(n+1)/2$ components of the covariance matrix. For the quadratic classifier, we have to estimate two mean vectors and two matrices, i.e. $2n + n(n+1)$ parameters. When the number of features is large, the number of parameters can become extremely large.

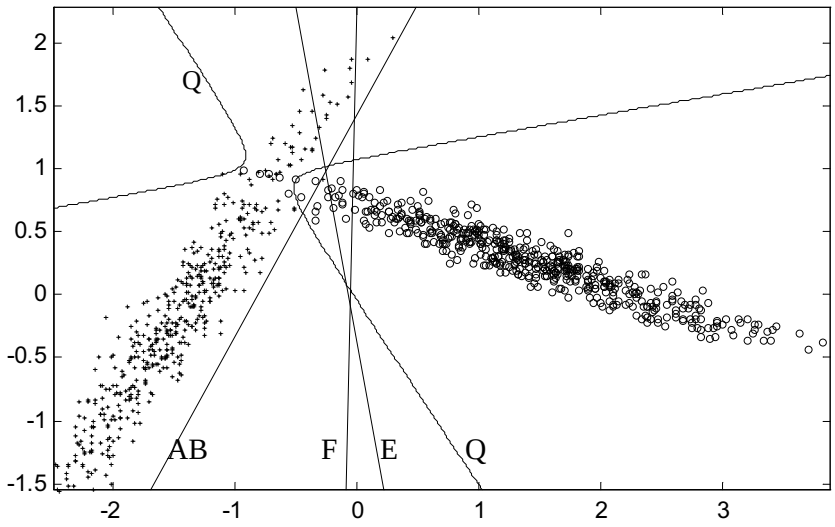

Fig. 2.2. Decision boundaries of four parametric classifiers.

Thus, to design the classifier, one needs a very large number of training examples. In many real world problems, only a finite number of examples is available. Therefore, the designer is forced to reduce the number of parameters. One way to do this is to reduce the number of features. Another way is to simplify the classifier. To do this, one needs to make some assumptions (constraints) on the structure of the covariance matrix.

2.3.1 A Set of Standard Assumptions

In Section 2.1 an example of such an assumption was already made – the covariance matrices of two classes were assumed to be identical. Further reduction in the number of parameters is available by constraining a type of dependency between the separate features. Constraints on the type of the covariance matrix help to reduce the number of parameters to be estimated from the training-set. If these constraints are not far from the true reality, then we can obtain a reduction in the generalisation error. The largest gain in reduction in the number of parameters is obtained in high-dimensional cases. In the ANN design, one can benefit from the utilisation of the constraints in following situations:

- we use the weights of the statistical classifier with the constrained covariance matrix to initialise the perceptron weights;

- we use the constrained estimates of the covariance matrix to whiten the data i.e. to rotate and scale the feature space in order to obtain pattern class distributions as close as possible to a spherical distribution. Then the Euclidean distance classifier obtained after the first training iteration of the SLP is a good classifier for the spherical pattern classes.

A set of the most powerful assumptions (constraints) is made if one assumes that:

(a) the covariance matrices of the different classes are identical;
(b) the features are statistically independent (this assumption was discussed at the end of Section 2.1);
(*c*) the variances of all features are identical.

If $P_2 = P_1$, the first constraint induces *the Fisher linear DF*. If all three assumptions are made, then the covariance matrix has a structure $\mathbf{\Sigma}_i = \mathbf{I}\,\sigma^2$. The use of this particular set of assumptions and the density (2.9) induces *the Euclidean distance classifier*.

If we use assumption (*b*), then we will have a quadratic classifier for independent features. This discriminant function (2.8) was already discussed in Section 2.1. Here, in order to design the classifier for the two-category case, one needs to estimate two *n*-variate mean vectors $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ and two diagonal matrices $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, with *n* variances in their diagonals, altogether 4*n* parameters. To invert sample based matrices $\hat{\boldsymbol{D}}_1$ and $\hat{\boldsymbol{D}}_2$, one needs to invert only the diagonal elements. We denote this as a *quadratic classifier,* $\mathrm{D_1D_2}$.

Let only assumptions (*a*) and (*b*) be made: the features are uncorrelated, and the variances of all features are identical. Then we have the linear discriminant function with following weights

$$\boldsymbol{V}^{\mathrm{D}} = \hat{\boldsymbol{D}}^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2) \quad \text{and} \quad v_0 = -\,\Delta\hat{\boldsymbol{M}}^{T}\boldsymbol{V}^{\mathrm{D}}, \tag{2.14}$$

where $\hat{\boldsymbol{D}}$ is the sample estimate of the diagonal variance matrix composed from diagonal elements of $\hat{\mathbf{\Sigma}}$, the sample covariance matrix and $\Delta\hat{\boldsymbol{M}} = \tfrac{1}{2}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)$.

We will call this classifier, the linear classifier for diagonal CM, or a *diagonal classifier* and denote it by the letter D.

2.3.2 Block Diagonal Matrices

A generalisation of the "diagonal" classifier is a discriminant function for Gaussian vectors having a block structured dependence between the variables. This DF structurally resembles the multi-layer perceptron with one hidden layer. Now, let us present more details. Let *n* components of random vector $\boldsymbol{X}$ be divided into *H* blocks $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_H$ with dimensionalities $n_1, n_2, \ldots, n_H$ ($\sum_{j=1}^{H} n_j = n$).

Inside each block, the features are supposed to be mutually dependent. The separate blocks are supposed to be statistically independent. Then the covariance matrix of vector $\boldsymbol{X}$ can be represented in a block-diagonal way:

$$\overline{\Sigma} = \begin{bmatrix} \Sigma_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Sigma_2 & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Sigma_H \end{bmatrix} \tag{2.15}$$

and one can use the following linear discriminant function

$$h^{\mathrm{BL}}(\boldsymbol{X}) = \sum_{j=1}^{H} [\boldsymbol{X}_j - \tfrac{1}{2}(\hat{\boldsymbol{M}}_{j1} + \hat{\boldsymbol{M}}_{j2})]^T \hat{\boldsymbol{\Sigma}}_j^{-1}(\hat{\boldsymbol{M}}_{j1} - \hat{\boldsymbol{M}}_{j2}) + c, \tag{2.16}$$

where $\hat{\boldsymbol{M}}_{j1}$ and $\hat{\boldsymbol{M}}_{j2}$ are n_j-variate vectors, the estimates of the mean vectors of each block of each of two classes, $\hat{\boldsymbol{\Sigma}}_j$ is the sample maximum likelihood estimate of $\boldsymbol{\Sigma}_j$, the covariance matrix of the j-th block, and c is a constant. Typically $c = 0$, however, in order to take into account an effect of unequal training-set sizes, N_1 and N_2, we will use a special c value (for a discussion of this problem, see Section 3.5.3).

The difference between the diagonal D and the block-diagonal classifier BL is that, instead of n independent separate variables, we have H independent blocks of variables with $n_H = \sum_{j=1}^{H} n_j(n_j+1)/2$ parameters to be estimated from the training-set. When both n and H are large, we have a drastic reduction in the number of parameters of the covariance matrix that have to be estimated from the training-set. Our experiments with a dozen of real-world data sets have shown that this model is useful.

It is very important that the parameters of the block covariance matrix are same for both competing pattern classes while analysing the influence of the number of parameters on the generalisation error (discussed in the next chapter). Here we can trace a similarity with the multilayer perceptron, where the weights of the hidden layer neurones affect all outputs, i.e. these weights are common for all outputs.

2.3.3 The Tree Type Dependence Models

An interesting and useful model which requires a small number of parameters to describe a joint dependence between all n variables is a model based on an approximation of a joint probability distribution by the first order *tree dependence*. According to the fundamentals of the probability theory, each multivariate density can be represented as

$$p(x_1, x_2, ..., x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1)...p(x_n \mid x_{n-1},..., x_2, x_1). \qquad (2.17)$$

In the first order tree dependence model, it is assumed that each variable is conditioned, at most, upon one other variable. One can say that each variable has only one "boss" component on which it depends. Then probability density function (2.17) can be written in the following simplified form:

$$p(x_1, x_2, ..., x_n) = \prod_{j=1}^{n} p(x_j \mid x_{mj}) \quad (1 \le m_j \le n) \qquad (2.18)$$

where a sequence $m_2, \ldots, m_n$ constitutes a graph of connections (an unknown permutation of the integers 1, 2, ..., n) and $p(x_i \mid x_0)$, by definition, is equal to $p(x_i)$.

In a general case, the covariance matrix has $n \times n$ non-zero elements. An inverse of this matrix $\bar{\Sigma}^{-1}$, however, has $2n$-1 non-zero elements. It is a result of the assumption that each component of the vector $\boldsymbol{X}$ depends only on one other component. For complex data, one has to construct the second and higher order tree dependence models. This important model is rarely mentioned in pattern recognition textbooks and remains practically unnoticed in the pattern recognition community. Therefore in Appendix 2, we present more details of this model and provide a numerical example.

2.3.4 Temporal Dependence Models

The recognition of the temporal and spatial objects and phenomena requires working with vectors of very high dimensionality. It is an advantageous area of application for the constrained statistical classifiers. There is a number of models that allow evaluation of temporal or spatial dependence between the data points. Undoubtedly, the expertise gained in this area can be useful in ANN design.

Let the components $x_1, x_2, ..., x_{n-1}, x_n$ of the multivariate vector be measurements differing in time or in space and assume they are realisations of a *stationary random process*. Then the covariance matrix has the following structure

$$\bar{\Sigma} = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \cdots & \delta_{n-1} & \delta_n \\ \delta_2 & \delta_1 & \delta_2 & \cdots & \delta_{n-2} & \delta_{n-1} \\ \delta_3 & \delta_2 & \delta_1 & \cdots & \delta_{n-3} & \delta_{n-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \delta_{n-1} & \delta_{n-2} & \delta_{n-3} & \cdots & \delta_1 & \delta_2 \\ \delta_n & \delta_{n-1} & \delta_{n-2} & \cdots & \delta_2 & \delta_1 \end{bmatrix}. \qquad (2.19)$$

We see there are only n parameters $\delta_1, \delta_2, \delta_3, \ldots, \delta_n$ that describe the structure of dependence between the variables. A number of special models, such as auto-regression, moving average, ARMA and circular, allow further reduction of the

number of parameters (see e.g., Appendix 3 and Fukunaga, 1990). For example, in the moving average model, we assume

$$x_t = \mu_t + b_0\, v_t + b_1\, v_{t\text{-}1} + \ldots + b_{q\ vt\text{-}q},$$

where μ_t, b_0, $b_1, \ldots, b_q$ are q+2 independent parameters of the model.

2.4 The Bayes Predictive Approach to Design Optimal Classification Rules

The plug-in approach of designing the classification rule does not lead to the optimal use of the training-set observations to design the classifier. An alternative to the plug-in approach is a Bayes (predictive) approach. This method theoretically leads to the optimal classification rules.

2.4.1 A General Theory

Let $p_1(\boldsymbol{X} \mid \mathbf{Y}, \mathbf{Y}_1)$, $p_2(\boldsymbol{X} \mid \mathbf{Y}, \mathbf{Y}_2)$, be the conditional probability density function (PDF) of the classes. Here, we assume that the vector $\mathbf{Y}$ is common for the distribution densities of both classes. In the Bayes predictive approach, two principal assumptions are:

- parameter vectors $\mathbf{Y}$, $\mathbf{Y}_1$, $\mathbf{Y}_2$ are random;
- the designer knows their *a priori* distribution density $p^{prior}(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)$.

We use these two assumptions and derive the optimal discriminant function with a minimal generalisation error over *all possible classification problems* defined by the density $p^{prior}(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)$.

Let $\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_2^{(1)}, \ldots, \boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots, \boldsymbol{X}_{N_2}^{(2)}$ be the training vectors. Then according to the Bayes formula the *a posteriori* distribution of $\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2$ is

$$p_i^{aposteriori}(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2 \mid \boldsymbol{X}_1^{(1)}, \boldsymbol{X}_2^{(1)}, \ldots, \boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots, \boldsymbol{X}_{N_2}^{(2)}) = \frac{p(\boldsymbol{X}_1^{(1)}, \ \ldots\ , \ \boldsymbol{X}_{N_2}^{(2)} \mid \mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)\, p^{prior}(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)}{\int p(\boldsymbol{X}_1^{(1)}, \ \ldots\ , \ \boldsymbol{X}_{N_2}^{(2)} \mid \mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)\, p^{prior}(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2)\, d\mathbf{Y} d\mathbf{Y}_1 d\mathbf{Y}_2} \tag{2.20}$$

and the *a posteriori* distribution density of vector $\boldsymbol{X}$ is

$$p_i^{aposteriori}(\boldsymbol{X} \mid \boldsymbol{X}_1^{(1)}, \boldsymbol{X}_2^{(1)}, \ldots, \boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots, \boldsymbol{X}_{N_2}^{(2)}) =$$

$$\int p_i(\boldsymbol{X} \mid \mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2) p(\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2 \mid \boldsymbol{X}_1^{(1)}, \ldots, \boldsymbol{X}_{N_1}^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots, \boldsymbol{X}_{N_2}^{(2)})\, d\mathbf{Y} d\mathbf{Y}_1 d\mathbf{Y}_2. \tag{2.21}$$

This predictive density is used to obtain the optimal classification rule.

2.4.2 Learning the Mean Vector

Now we will illustrate how this approach can lead to the Euclidean distance classifier. Let the vector $\boldsymbol{X}$ be Gaussian with the density in Equation (2.9). Suppose the covariance matrix $\bar{\Sigma}$ of this distribution is common for both pattern classes and known. Let the *a priori* distribution of the mean vector $\boldsymbol{M}$ be Gaussian

$$p_i^{prior}(\boldsymbol{M}_i) = N_{\boldsymbol{M}}(\mu, \mathbf{K}), \tag{2.22}$$

where μ is n-variate vector and $\mathbf{K}$ is an $n \times n$ symmetric positive defined matrix.

We will need the equality

$$\int N_X(\boldsymbol{M}, \Sigma) N_{\boldsymbol{M}}(\mu, \mathbf{K}) d\boldsymbol{M} = N_X(\mu, \Sigma + \mathbf{K}). \tag{2.23}$$

Use of (2.22) and (2.23) in (2.20) and (2.21) results in

$$p_i^{aposteriori}(\boldsymbol{M}_i \mid \boldsymbol{X}_1^{(i)}, \boldsymbol{X}_2^{(i)}, ..., \boldsymbol{X}_{N_i}^{(i)}) =$$

$$N_{\boldsymbol{M}}\left((\mathbf{K}^{-1} + \Sigma^{-1} N_i)^{-1} (\Sigma^{-1} \hat{\boldsymbol{M}}_i N_i + \mathbf{K}^{-1}\mu), (\mathbf{K}^{-1} + \Sigma^{-1} N_i)^{-1}\right). \tag{2.24}$$

$$p_i^{aposteriori}(\boldsymbol{X} \mid \boldsymbol{X}_1^{(i)}, \boldsymbol{X}_2^{(i)}, ..., \boldsymbol{X}_{N_i}^{(i)}) =$$

$$N_X\left((\mathbf{K}^{-1} + \Sigma^{-1} N_i)^{-1} (\Sigma^{-1} \hat{\boldsymbol{M}}_i N_i + \mathbf{K}^{-1}\mu), (\mathbf{K}^{-1} + \Sigma^{-1} N_i)^{-1} + \Sigma\right). \tag{2.25}$$

Equation (2.25) is too complex to follow the influence of the prior distribution on the estimate of probability density functions of the classes. Assume the prior mean vector μ is common for both classes. Let $\Sigma = \mathbf{I}\sigma_x^2$, $\mathbf{K} = \mathbf{I}\sigma_a^2$. Then the *a posteriori* density for vector $\boldsymbol{X}$.

$$p_i^{aposteriori}(\boldsymbol{X} \mid \boldsymbol{X}_1^{(i)}, \boldsymbol{X}_2^{(i)}, ..., \boldsymbol{X}_{N_i}^{(i)}) =$$

$$N_X\left(\hat{\boldsymbol{M}}_i \frac{\sigma_a^2}{\sigma_a^2 + \sigma_x^2 / N_i} + \mu \frac{\sigma_x^2 / N_i}{\sigma_a^2 + \sigma_x^2 / N_i}, \mathbf{I}\left(\sigma_a^2 \frac{\sigma_x^2 / N_i}{\sigma_a^2 + \sigma_x^2 / N_i} + \sigma_x^2\right)\right), \tag{2.26}$$

From Equation (2.26), we see that the more we have training vectors, the effects of the prior distribution is less noticed. When the number of training vectors is small or the covariance matrix of the prior distribution is "narrow" then the contribution of prior distribution is significant. Inserting (2.26) into (2.6), results in a quadratic

discriminant function. When $N_2 = N_1$ we obtain the linear classifier (1.2) i.e. the standard EDC. We see for the spherical Gaussian distributions of $\boldsymbol{X}$ and priors for $\boldsymbol{M}_i$, the plug-in classifier (the EDC) is not optimal in the Bayes predictive sense if $N_2 \neq N_1$.

Equations (2.25) and (2.26) show that analytical expressions of the Bayes approach are much more complex than that of the plug-in approach. Moreover, in the Bayes approach, an important and very difficult problem is the determination of parameters of the prior distribution.

One of suggestions most often used in the literature is to choose a very "flat" density as a prior. For the above example, it would be a choice of a very large value of σ_a^2. When $\sigma_a^2 \to \infty$ then, from (2.26), it follows that

$$p^{aposteriori}(\boldsymbol{M}_i) = N_{\boldsymbol{M}i}(\hat{\boldsymbol{M}}_i, \mathbf{I}\frac{\sigma_x^2}{N_i}). \tag{2.27}$$

$$p_i^{aposteriori}(\boldsymbol{X} \mid \boldsymbol{X}_1^{(i)}, \boldsymbol{X}_2^{(i)}, \ldots, \boldsymbol{X}_{N_i}^{(i)}) = N_X(\hat{\boldsymbol{M}}_i \times 1 + \mu \times 0, \mathbf{I}(\frac{\sigma_x^2}{N_i} + \sigma_x^2)). \tag{2.28}$$

To show the similarity with (2.26), in Equation (2.28) we presented the mean as a sum of two terms. The variance in the above equation depends on N_i and indicates once more that the difference between the classifiers obtained by the plug-in and the Bayes approaches arises when $N_2 \neq N_1$.

In order to see a parallel between network training and the Bayes predictive approach, let us represent the estimate of the mean of the *a posteriori* distribution as a recurrent equation. Let us have t observations from ω_i and add a new, (t+1)-th, observation $\boldsymbol{X}_{t+1}^{(i)}$. As before, let $\sigma_a^2 \to \infty$. For the new observation vector $\boldsymbol{X}_{t+1}^{(i)}$, we use the *a posteriori* density (2.27) as a prior density, i.e. $\mu = \hat{\boldsymbol{M}}_i$, $\sigma_a^2 = \frac{\sigma_x^2}{t}$. Note that $\hat{\boldsymbol{M}}_i$ is estimated from t training vectors. Then applying (2.28) with a new training-set size, N_i =1, we obtain

$$p^{aposteriori}(\boldsymbol{M}_i) = p(\boldsymbol{M}_i \mid \boldsymbol{X}_1^{(i)}, \ldots, \boldsymbol{X}_t^{(i)}, \boldsymbol{X}_{t+1}^{(i)}) =$$

$$N_X(\hat{\boldsymbol{M}}_i \frac{1}{t+1} + \boldsymbol{X}_{t+1}^{(i)} \frac{t}{t+1}, \mathbf{I}\frac{1}{t+1}\sigma_x^2). \tag{2.29}$$

Thus, utilisation of the maximal value of the *a posteriori* density as the estimate of $\boldsymbol{M}_i$ after t +1 training vectors results in the following iterative procedure

$$(\hat{\boldsymbol{M}}_i)^{t+1} = (\hat{\boldsymbol{M}}_i)^t(1-\eta) + \boldsymbol{X}_{t+1}^{(i)}\eta, \tag{2.30}$$

where $\eta = \frac{1}{t+1}$ and $(\hat{\boldsymbol{M}}_i)^t = \hat{\boldsymbol{M}}_i$.

The reader should pay attention to the above procedure, where the parameter η is the "learning step". This parameter is derived strictly from using the statistical approach. In this training scheme, the optimal value should diminish as the number of training vectors increases. This η value is optimal for the given assumptions and performance criterion.

2.4.3 Learning the Mean Vector and CM

An iterative way to recompute the estimate of the mean (as well as the variance) can be applied to construct the iterative procedure to adapt other parameters of the classification rule. Similar equations have been derived for estimation of the covariance matrix and for the *a poster*iori distribution of the multivariate Gaussian vector $\boldsymbol{X}$. Again, in this approach, $\boldsymbol{M}_1$, $\boldsymbol{M}_2$, and $\overline{\boldsymbol{\Sigma}}$ are supposed to be random variables and the prior distribution $p^{prior}(\boldsymbol{M}_1, \boldsymbol{M}_2, \overline{\boldsymbol{\Sigma}})$ to be known. Then we seek for the *a posteriori* (predictive) density of vector $\boldsymbol{X}$, and, for this density estimate, we design the classifier. Such a classifier is optimal in the sense that it yields the minimal classification error for a *set of classification problems* defined by the prior distribution $p^{prior}(\boldsymbol{M}_1, \boldsymbol{M}_2, \overline{\boldsymbol{\Sigma}})$. For the uniform prior distribution of $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\overline{\boldsymbol{\Sigma}}$, the predictive density is

$$p_i^{aposteriori}(\boldsymbol{X} \mid \boldsymbol{X}_1^{(i)}, \boldsymbol{X}_2^{(i)}, ..., \boldsymbol{X}_{N_i}^{(i)}) \propto$$

$$\left(\frac{N_i}{N_i+1}\right)^{\frac{n}{2}} \left(1+\frac{N_i(\boldsymbol{X}-\hat{\boldsymbol{M}}_i)^T \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{X}-\hat{\boldsymbol{M}}_i)}{(N_i+1)(N-2)}\right)^{\frac{N-3}{2}}. \tag{2.31}$$

If one uses the above density to obtain the classification rule, a quadratic discriminant function results. When $N_2 = N_1 = \overline{N}$ and the prior probabilities of the classes are equal, the above optimal Bayes classifier is linear and becomes equivalent to the Fisher linear discriminant. Thus, the Fisher classifier is an optimal sample-based classifier in the predictive Bayes sense when the prior distribution of the mean vector and the covariance matrix are uniform and we have the same number of training vectors from both pattern classes ($N_2 = N_1$).

2.4.4 Qualities and Shortcomings

The Bayes predictive approach is the only method that formulates optimal sample based classification rules. This method allows us to analyse the optimality of the empirical plug-in approach. We have seen that EDC becomes the optimal classification rule for a uniform prior distribution of the mean vectors $\boldsymbol{M}_1$, $\boldsymbol{M}_2$. For the case $N_2=N_1$, the standard Fisher linear DF becomes the optimal classification rule for uniform prior distribution of the mean vectors $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and the covariance matrix $\overline{\boldsymbol{\Sigma}}$. This classifier, however, is no longer optimal when $N_2 \neq N_1$. The use of the Bayes approach allows us to avoid painful problems associated with inversion of ill-conditioned sample covariance matrix in the high dimensional case.

The plug-in approach is computationally simple and intuitively understandable. The Bayes predictive approach, however, requires much more complex analytical expressions. A main drawback of the predictive Bayes approach consists of the definition of the prior distribution density of unknown parameters. The flat and almost uniform *a priori* distribution of *n*-variate mean vectors as well as $n(n+1)$ coefficients of an inverse of the covariance matrix, $\overline{\boldsymbol{\Sigma}}^{-1}$, imposes that certain unrealistic sets of parameters $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\overline{\boldsymbol{\Sigma}}$ can occur. Among these occurrences, one can meet situations that have almost singular distributions of pattern classes.

The almost uniform prior distribution is a suggested mathematical manoeuvre to overcome numerical difficulties. It is not based on the analysis of real-world situations. In Chapter 3, we will see that the expected PMC of such "optimal" solutions are at times even higher than that of the plug-in classifier. Therefore, the optimism that arose after suggesting this approach in the mid-sixties soon died away. Nevertheless, it is a powerful theoretical tool to analyse and design decision making rules. No doubt, in the future, this method will be elaborated further with practical models to define prior distributions of unknown parameters and should find a wider range of applications.

2.5 Modifications of the Standard Linear and Quadratic DF

If N_1, N_2 are small in comparison with the number of dimensions *n*, i.e. $N < n + 2$, then there arises problems associated with the inversion of the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$. In Section 2.3, we described a number of structures of the covariance matrix which can be useful to overcome this kind of difficulty. In this section, we will present four other techniques to deal with the dimensionality problem.

2.5.1 A Pseudo-Inversion of the Covariance Matrix

One of possibilities for dealing with singular matrices is to use a pseudo-inversion. A sense of the pseudo-inversion of the symmetric matrix $\hat{\boldsymbol{\Sigma}}$ consists of a singular value decomposition of this matrix:

$$\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T,$$

where $\boldsymbol{\Phi}$ is an orthogonal $n \times n$ matrix such that

$$\boldsymbol{\Phi}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi} = \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}, \tag{2.32}$$

and $\boldsymbol{\Phi}$ is called an eigenvectors matrix and $\boldsymbol{\Lambda}$ is called an eigenvalues matrix of $\hat{\boldsymbol{\Sigma}}$.

Note that $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{\Phi}\,\mathbf{\Phi}^T = \mathbf{I}$ and $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^T$. In order to grasp a better understanding of the pseudo-inversion, suppose for a moment that $\hat{\mathbf{\Sigma}}$ is a positive defined matrix and can be inverted in a standard way. Following (2.32), the inverse can be written as

$$\hat{\mathbf{\Sigma}}^{-1} = (\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T)^{-1} = (\mathbf{\Phi}^T)^{-1}\mathbf{\Lambda}^{-1}\mathbf{\Phi}^{-1} = \mathbf{\Phi}\,\mathbf{\Lambda}^{-1}\mathbf{\Phi}^T. \tag{2.33}$$

Now, let the sample covariance matrix be singular. Some of the eigenvalues are equal to zero. Suppose $\lambda_1, \lambda_2, \ldots, \lambda_r > 0$, and $\lambda_{r+1}, \ldots, \lambda_n = 0$. When $N_i < n+1$, then $r = N_i - 1$. Denote

$$\lambda = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \lambda_r \end{bmatrix}.$$

Thus, we can rewrite equation (2.32) in a block way

$$\mathbf{\Phi}^T\,\hat{\mathbf{\Sigma}}\,\mathbf{\Phi} = [\mathbf{\Phi}_1\,\mathbf{\Phi}_2]^T\,\hat{\mathbf{\Sigma}}\,[\mathbf{\Phi}_1\,\mathbf{\Phi}_2] = \begin{bmatrix} \lambda & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.34}$$

where we have split the orthogonal matrix $\mathbf{\Phi} = [\mathbf{\Phi}_1\,\mathbf{\Phi}_2]$ into $n \times r$ matrix $\mathbf{\Phi}_1$ and $n \times (n\text{-}r)$ matrix $\mathbf{\Phi}_2$. Then, in analogy with the conventional inverse (2.33), the pseudo-inverse of matrix $\hat{\mathbf{\Sigma}}$

$$\hat{\mathbf{\Sigma}}^{+} = \mathbf{\Phi}\begin{bmatrix} \lambda^{-1} & 0 \\ 0 & 0 \end{bmatrix}\mathbf{\Phi}^T = [\,\mathbf{\Phi}_1\,\mathbf{\Phi}_2]\begin{bmatrix} \lambda^{-1} & 0 \\ 0 & 0 \end{bmatrix}[\,\mathbf{\Phi}_1\,\mathbf{\Phi}_2]^T = \mathbf{\Phi}_1\,\lambda^{-1}\,\mathbf{\Phi}_1{}^T. \tag{2.35}$$

In a number of mathematical program packages, there exist standard matrix pseudo-inversion subroutines. Use of (2.35) in (1.3) results in the Fisher linear DF with pseudo-inversion, or, simply, the **Pseudo-Fisher classifier PF**. Equation (2.35) shows that we invert only the positive defined eigenvalues matrix λ. Instead of classifying the vectors $\boldsymbol{X}$ in the original n-dimensional space, we classify new vectors $\boldsymbol{X}_{new} = \mathbf{\Phi}_1{}^T\boldsymbol{X}$ in an r-dimensional subspace of non zero eigenvalues of the covariance matrix. The resulting decision hyperplane is at *equal distances* from all N training vectors.

In the analysis of adaptive classification algorithms (Chapter 4), we will also meet pseudo-inversion. There, instead of the standard sample covariance matrix $\hat{\mathbf{\Sigma}}$, the following sample covariance matrix of non-centred training vectors is sometimes used:

$$\hat{\mathbf{\Sigma}}_{NC} = \frac{1}{N_1 + N_2 - 2}\sum_{i=1}^{2}\sum_{j=1}^{N_i}\boldsymbol{X}_j^{(i)}(\boldsymbol{X}_j^{(i)})^T. \tag{2.36}$$

2.5.2 Regularised Discriminant Analysis (RDA)

Another approach to the overwhelming difficulties associated with the matrix inversion in regression and discriminant analysis, apart from the standard CM estimates (1.4) or (2.10*b*), is to use a "ridge" CM estimate

$$\hat{\boldsymbol{\Sigma}}_{\text{RDA}} = \hat{\boldsymbol{\Sigma}} + \lambda \mathbf{I}, \tag{2.37}$$

where λ is a positive regularisation constant.

The use of representation (2.32) indicates that $\hat{\boldsymbol{\Sigma}} + \lambda \mathbf{I} = \boldsymbol{\Phi}\,(\boldsymbol{\Lambda} + \lambda \mathbf{I})\boldsymbol{\Phi}^T$. Now, it is easier to invert the matrix $\hat{\boldsymbol{\Sigma}}_{\text{RDA}}$: we add positive constants λ to all eigenvalues of the matrix $\hat{\boldsymbol{\Sigma}}$. The new matrix $\hat{\boldsymbol{\Sigma}} + \lambda \mathbf{I}$ becomes positive defined. Inserting (2.37) into Equation (1.3) gives

$$\boldsymbol{V}^{\text{RDA}} = (\hat{\boldsymbol{\Sigma}} + \lambda \mathbf{I})^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2) \text{ and } v_0 = -\tfrac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)^T \boldsymbol{V}^{RDA}. \tag{2.38a}$$

If the prior probabilities of the classes are equal among themselves, we perform the classification according to the sign of the discriminant function. For further analysis and for practical applications it is convenient to rewrite Equation (2.38*a*) in a following way

$$\boldsymbol{V}^{\text{RDA}} = k_\lambda\,((1-\lambda_1)\hat{\boldsymbol{\Sigma}} + \lambda_1 \mathbf{I})^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2), \tag{2.38b}$$

where $0 \le \lambda_1 = \lambda/(1+\lambda) \le 1$, and $k_\lambda = 1/(1+\lambda)$ can be ignored in classification according the sign of the discriminant function.

Representation (2.38*a*) shows that, when $\lambda_1 = 0$, we have the standard Fisher linear DF and, when $\lambda_1 = 1$, we have the Euclidean distance classifier. This classification method is known as a **regularized linear discriminant analysis** (RDA). The parameter λ controls a similarity of RDA to EDC and to the Fisher linear DF. The following modification of the linear RDA is used to design the linear classifier when the covariance matrices are different

$$\boldsymbol{V}^{\text{RDA*}} = (\gamma(1-\lambda_1)\hat{\boldsymbol{\Sigma}}_1 + (1-\gamma)(1-\lambda_1)\hat{\boldsymbol{\Sigma}}_2 + \lambda_1 \mathbf{I})^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2), \tag{2.39}$$

where we have two regularisation parameters, λ_1 and γ.

The weight vector (2.39) resembles that of the Anderson-Bahadur procedure (Equation (2.13)). Some authors claim it to be the most powerful classification method among other parametric linear statistical classifiers. Here, the optimal values of two parameters, λ_1 and γ, should be determined in order to obtain the best generalisation. The optimal values of λ_1 and γ depend on the training-set size. It is important to know that the optimal values fluctuate with each random training-set.

The configuration of the real multivariate data is usually unknown to the designer. There is a small chance that optimal values of λ_1 and γ can be determined theoretically. It is recommended that one uses a cross-validation error rate estimation method to find the optimal values.

2.5.3 Scaled Rotation Regularisation

One more generalisation of the linear RDA is a scaled rotation (SR). In addition to the regularisation of the eigenvalues Λ in the singular value decomposition $\hat{\Sigma} = \Phi\Lambda\Phi^T$, here we regularise the orthogonal $n \times n$ eigenvectors matrix Φ and estimate the covariance matrix as $\hat{\Sigma}_{SR} = \Phi^{\alpha} (\Lambda + \lambda \mathbf{I})(\Phi^{\alpha})^T$. The parameter α, a real valued variable, controls the degree of rotation determined by matrix Φ^{α} from the identity matrix $\mathbf{I}$ (when $\alpha=0$) to the eigenvectors matrix Φ (when $\alpha=1$) and beyond (when $\alpha>1$). The regularisation is performed by means of two parameters, α and λ. This technique is especially effective when used with a linear data transformation, $\boldsymbol{X}_{new} = \Lambda^{-1/2} \Phi^T \boldsymbol{X}$, performed prior to the training of the non-linear single layer perceptron (see Chapter 5).

2.5.4 Non-Gaussian Densities

The multivariate Gaussian density is the simplest statistical model for multivariate vectors with continuous components. A more general family of the distributions is

$$f_i(\boldsymbol{X}) = k_e \Psi(\boldsymbol{X})\, J\{H(\boldsymbol{X}, \mathbf{m}_i, \mathbf{K}_i)\}, \tag{2.40}$$

where $\mathbf{m}_i$ is a n-variate vector, $H(\boldsymbol{X}, \mathbf{m}_i, \mathbf{K}_i)$ is a distance measure between the two vectors, $\boldsymbol{X}$ and $\mathbf{m}_i$, in a space transformed by matrix $\mathbf{K}_i$, $\Psi(\boldsymbol{X})$ is an arbitrary *scalar function* of $\boldsymbol{X}$, $J\{\cdot\}$ is an arbitrary monotone decreasing uni-variate function and k_e is a normalising constant.

One of the possible metrics is $H(\boldsymbol{X}, \mathbf{K}_i, \mathbf{m}_i) = (\boldsymbol{X} - \mathbf{m}_i)^T \mathbf{K}_i^{-1} (\boldsymbol{X} - \mathbf{m}_i)$. The function $\Psi(\boldsymbol{X})$ does not depend on the class index. Thus, using (2.40) in (2.6) results in a *quadratic* discriminant function. If matrix $\mathbf{K}_i$ does not depend on the class index i, then we have the *linear* classifier. A method of obtaining maximum likelihood estimates of the resulting discriminant function was proposed by Day and Kerridge (1967). For linear classification we have to maximise the product

$$L = \prod_{i=1}^{2} \prod_{j=1}^{N_i} \frac{[\exp(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0)]^{2-i}}{1 + \exp(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0)} \tag{2.41}$$

with respect to the weights $\boldsymbol{V}$, v_0. Note that the same result can be obtained without any assumptions on the type of the distribution density functions of the pattern classes. In the probabilistic interpretation, a ratio

$$\frac{\exp(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0)}{1 + \exp(\boldsymbol{V}^T \boldsymbol{X}_j^{(i)} + v_0)}$$

expresses *a posteriori* probability that vector $\boldsymbol{X}_j^{(i)}$ belongs to class ω_i. Like the sum (1.7), the product L is in fact a cost function to be minimised. In discriminant analysis literature, this decision-making procedure is known as a **logistic regression.** It has received considerable attention in solving medical diagnostic problems where it is important to estimate *a posteriori* probabilities that the observation vector $\boldsymbol{X}$ belongs to a certain pattern class.

Note that in the probabilistic model (2.40), the vector $\mathbf{m}_i$ is not the mean vector and the matrix $\mathbf{K}_i$ is not the covariance matrix. The model (2.40) is more general than the model of multivariate Gaussian distribution (2.9). This model contains a favourable peculiarity, having weaker assumptions on the class conditional densities. The weaker assumptions, however, require more training vectors to estimate the density and to design the classification rule. This is the unfavourable characteristic of the model.

Resemblance between the product L and the cost function utilised in the single layer perceptron training (see Equation (1.7) and Chapter 4) indicates that the logistic regression can be analysed as one of the modifications of the artificial neural networks.

2.5.5 Robust Discriminant Analysis

In many applications, we have data contaminated by "noise", i.e. the measurement vectors have many atypical observations, called *outliers*. In this case, the maximum likelihood estimates derived for the Gaussian data become ineffective. There is a branch of statistics which deals with such data and derives the parameter estimates that are robust to atypical observations: *robust statistics.* Most popular are Huber *M*-estimates. As an example, we present an iterative procedure to obtain a robust estimate of the mean vector $\boldsymbol{M}$ and the covariance matrix $\boldsymbol{\Sigma}$.

To obtain robust estimates $\hat{\boldsymbol{M}}_M$ and $\hat{\boldsymbol{\Sigma}}_M$ (here we omit the index of class i), one first computes conventional sample maximum-likelihood estimates $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{\Sigma}}$. Then, these estimates are used to compute initial distances H_j of each training vector $\boldsymbol{X}_j$ to a centre $\hat{\boldsymbol{M}}$ of the training-set $H_j = (\mathbf{X}_j - \hat{\boldsymbol{M}})^T \hat{\boldsymbol{\Sigma}}_M^{-1} (\mathbf{X}_j - \hat{\boldsymbol{M}})$. In the end, one replaces sample estimates by the following weighted estimates

$$\hat{\boldsymbol{M}}_M = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{X}_j w_j \Big/ \sum_{j=1}^{N} w_j,$$

$$\hat{\boldsymbol{\Sigma}}_M = \frac{1}{N-1} \sum_{j=1}^{N} (\boldsymbol{X}_j - \hat{\boldsymbol{M}}_M)(\boldsymbol{X}_j - \hat{\boldsymbol{M}}_M)^T w_j^2 \Big/ \sum_{j=1}^{N} w_j^2,$$

where w_j is a weight (an "importance") of the observation vector $\boldsymbol{X}_j$, e.g.

$$w_j = \begin{cases} 2/H_j & \text{if} \quad H_j > 2 \\ 1 & \text{if} \quad H_j \le 2. \end{cases}$$

In the second iteration, one uses the previously obtained estimates $\hat{\boldsymbol{M}}_M$ and $\hat{\boldsymbol{\Sigma}}_M$ to find new values for H_j, and then again finds new estimates $\hat{\boldsymbol{M}}_M$ and $\hat{\boldsymbol{\Sigma}}_M$. The iterative procedure is repeated several times and the influence of the distant atypical training vectors is diminished. It is suggested that the robust estimates $\hat{\boldsymbol{M}}_M$ and $\hat{\boldsymbol{\Sigma}}_M$ can be used in the plug-in sample linear and quadratic discriminant functions.

One more way to get the robust discriminant function is obtained from a generalisation of the Fisher criterion (2.12). The generalisation of the Fisher criterion (2.12) can be written in following way:

$$T_{gen} = \frac{1}{N_1{}^+ N_2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} \varphi\left(\alpha \frac{t_j^i - \boldsymbol{V}^T \boldsymbol{X}_j^{(i)} - v_0}{\sqrt{\boldsymbol{V}^T \mathbf{S}^{-1} \boldsymbol{V}}}\right), \tag{2.42}$$

where $t_j^{(i)}$ is the class index of the training pattern vector $\boldsymbol{X}_j^{(i)}$: $t_j^{(1)} = +1$, $t_j^{(2)} = -1$, $\varphi(c)$ is a non-decreasing odd and non-constant function, e.g. $\varphi(c) = (-1)^{i+1} \tanh(c)$ (c is a positive constant). Parameter α controls a robustness and should be found in trial and error.

Fisher used the linear function $\varphi(c) = c$. The utilisation of the non-linear function $\varphi(c)$ generalises the Fisher criterion. It has an essential similarity with the adaptive SLP design discussed in Chapter 1. Due to saturation of the non-linear function $\varphi(c)$, the solution becomes more robust to outliers than the standard Fisher discriminant. For numerical optimisation of the above criterion, the function $\varphi(c)$ should be smooth and differentiable. It should be similar to the soft-limiting activation function (1.8) used in the ANN design.

2.6 Nonparametric Local Statistical Classifiers

The assumptions on the structure class conditional densities considered in the previous sections are based on the global structure of multivariate density functions. A large group of statistical pattern classification algorithms is founded on the utilisation of nonparametric estimates of the multivariate density functions. Here, no assumptions about the global structure of the populations are made. This group of density estimation methods allow us to consider complex multimodal distributions.

2.6.1 Methods Based on Mixtures of Densities

If we are confronted with multimodal densities, it is reasonable to assume that each class consists of several subclasses. The density function can be written as a sum

$$p_i(\boldsymbol{X}) = \sum_{j=1}^{m_i} P_{ij}\, p(\boldsymbol{X} \mid \mathbf{Y}_{ij}), \tag{2.43}$$

where m_i is a number of subclasses in the population (class) ω_i, P_{ij} is the *a priori* probability of the j^{th} subclass from ω_i ($j = 1, \ldots, m_i$; $\sum_{j=1}^{m_i} P_{ij} = 1$), $p(\boldsymbol{X} \mid \mathbf{Y}_{ij})$ is a probability density function of the j-th subclass, and $\mathbf{Y}_{ij}$ is a vector of parameters of the ij-th component (subclass) of the mixture.

Representation (2.43) allows one to approximate complex unimodal and multimodal distribution density functions that can be found in practical pattern recognition problems. An illustration of two univariate densities, $f_1(x)$ and $f_2(x)$, with $m_1=3$ and $m_1=2$ is presented in Figure 2.3.

In the two pattern classes case, the discriminant function (DF) obtained from representation (2.43) has the following form

$$h(\boldsymbol{X}) = P_1 \sum_{j=1}^{m_1} P_{1j}\, p_1(\boldsymbol{X} \mid \mathbf{Y}_{1j}) - P_2 \sum_{j=1}^{m_2} P_{2j}\, p_2(\boldsymbol{X} \mid \mathbf{Y}_{2j}). \tag{2.44}$$

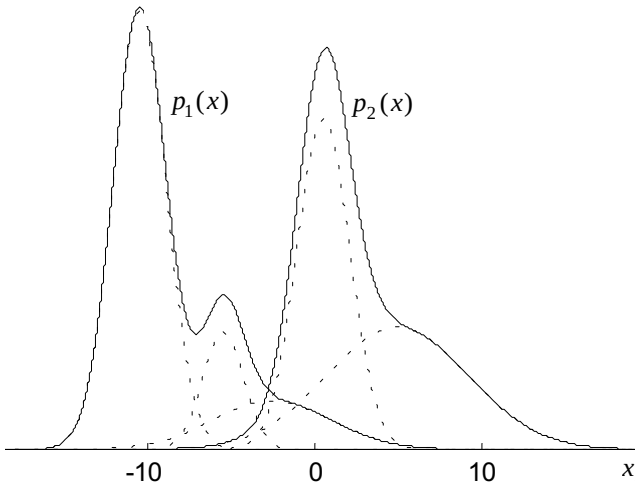

Fig. 2.3. Two multimodal uni-variate density functions represented by the mixture model: $p_1(x)$ is composed of three components and $p_2(x)$ of two.

In principle, considering representation (2.43), one can use any known parametric density function, e.g. the general multivariate density (2.40). Equation (2.43), however, requires a great number of parameters to be estimated. Therefore, in the literature, only analysis of simple cases with a small number of parameters can be found, i.e. Gaussian densities with the identity or the diagonal covariance matrices:

$$h(\boldsymbol{X}) = \sum_{i=1}^{2} (-1)^{i+1} P_i \sum_{j=1}^{m_i} P_{ij}\, (2\pi)^{-n/2}\, e^{-1/2(\boldsymbol{X}-\boldsymbol{C}_{ij})^T \boldsymbol{D}^{-1}(\boldsymbol{X}-\boldsymbol{C}_{ij})}, \tag{2.45}$$

where $\boldsymbol{C}_{ij}$ is a "centre", the mean vector of the j-th subclass of the class ω_i, and $\boldsymbol{D}$ is a diagonal matrix. In a simplified model, $\boldsymbol{D} = \mathbf{I}\sigma^2$.

There exist methods that use the training vectors in order to estimate unknown parameters of the mixture density. A typical example is the maximum likelihood method. An alternative is to use cluster analysis techniques, which are much faster.

The mixture model (2.43) and the discriminant function (2.44) constitute a basic model for a special type of artificial neural networks, known as radial basis functions (RBF). In the RBF approach, the statistical parameters of the mixtures $\mathbf{Y}_{1j}$, $\mathbf{Y}_{2j}$ serve as initial values of the weights in the iterative ANN training. Typically, the cost function of the sum of the squares is used in order to estimate the network weights by means of the adaptive training process.

2.6.2 Piecewise-Linear Classifiers

A significant simplification of the calculation process during the recognition phase follows if, instead of the sum in (2.44), one analyses each of the m_i components separately as an independent new subclass. Then a piecewise-linear discriminant function can be constructed:

$$h(\boldsymbol{X}) = \min_j \{ (\boldsymbol{X} - \boldsymbol{C}_{2j})^T \boldsymbol{D}^{-1} (\boldsymbol{X} - \boldsymbol{C}_{2j}) \} - \min_j \{ (\boldsymbol{X} - \boldsymbol{C}_{1j})^T \boldsymbol{D}^{-1} (\boldsymbol{X} - \boldsymbol{C}_{1j}) \} =$$

$$\min_j \{ -2\, \boldsymbol{X}^T \boldsymbol{D}^{-1} \boldsymbol{C}_{2j} + (\boldsymbol{C}_{2j})^T \boldsymbol{C}_{2j} \} - \min_j \{ -2\, \boldsymbol{X}^T \boldsymbol{D}^{-1} \boldsymbol{C}_{1j} + (\boldsymbol{C}_{1j})^T \boldsymbol{C}_{1j} \}. \qquad (2.46)$$

A decision making scheme which performs the above calculations is depicted below.

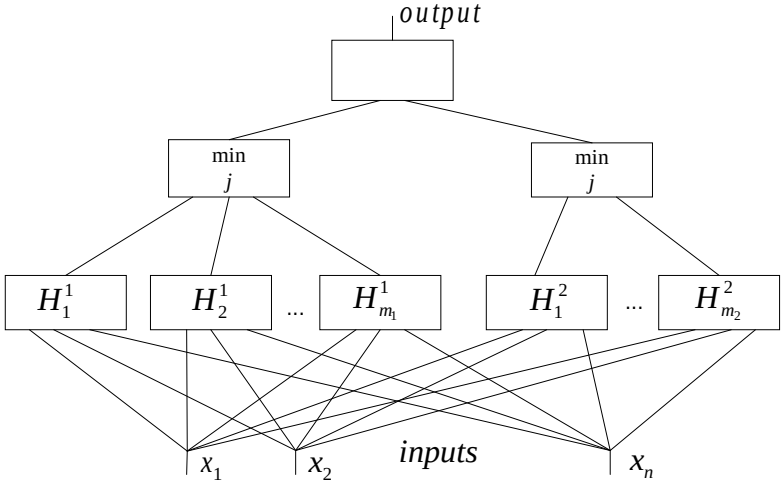

Fig. 2.4. A version of the piecewise-linear classifier designed for a mixture of spherical densities.

In the ANN literature, this kind of decision-making scheme and the discriminant function (2.46) are known as the learning vector quantisation (LVQ) network. In addition to the cluster analysis and the mixture decomposition methods, statistical analysis is traditionally used to evaluate the centres $\mathbf{C}_{ij}$. In the ANN approach, these parameters are used for initialisation of the network.

2.6.3 The Parzen Window Classifier

The mixture statistical model contains a structurisation of the multivariate densities only in local areas of the multivariate feature space, Ω. In some applications, we have no information on either the global structure or the local structure of the multivariate densities. However, we can assume:

1) the objects of one class are similar among themselves, i.e. distances between the vectors $\boldsymbol{C}_{i1}$, $\boldsymbol{C}_{i2}$, …, $\boldsymbol{C}_{ir}$ and $\boldsymbol{X}_s^{(i)}$ can be small;

2) each physical measurement $\boldsymbol{X}_j^{(i)}$ (j-th training vector from the class ω_i) is performed with some error, i.e.

$$\boldsymbol{X}_j^{(i)} = \boldsymbol{X}_{0j}^{(i)} + \xi_j^{(i)}, \tag{2.47}$$

where $\xi_j^{(i)}$ is a measurement error (noise) vector: e.g., $\xi_j^{(i)} \sim N_X(\mathbf{0}, \boldsymbol{D}\lambda)$, and $\boldsymbol{X}_{0j}^{(i)}$ is a measured value of the n-variate feature vector. In principle, it might be obtained only in a case when the physical measurements are performed exactly (without errors).

The above two assumptions are not very restrictive. They impose that, in a small distance from each training vector, an other observation from the same pattern class can be found. Essentially, it is *additional prior information*. This information can be profitable if used inventively.

Assume now that to each training vector $\boldsymbol{X}_j^{(i)}$, we add an infinite number of new random vectors $\boldsymbol{Z}_1$, $\boldsymbol{Z}_2$, $\boldsymbol{Z}_3$, … having Gaussian distribution with zero mean and the diagonal covariance matrix $\boldsymbol{D}\lambda$:

$$p(\boldsymbol{Z}) = N_z(\mathbf{0}, \boldsymbol{D}\lambda) ,$$

Resulting density estimates of the new vectors $\boldsymbol{X}_{sj}^{(i)} = \boldsymbol{X}_j^{(i)} + \boldsymbol{Z}_s$ can be analysed by superposition of N_i random vectors. The probability density function of a set of "noisy" vectors can be represented as

$$\hat{p}_i(\boldsymbol{X}) = \frac{1}{N_i} \sum_{j=1}^{N_i} N_X(\boldsymbol{X}_j^{(i)}, \boldsymbol{D}\lambda). \tag{2.48}$$

In the statistical literature, this estimate of the probability density function is known as a nonparametric Parzen window (PW) estimate. Use of (2.48) in (2.5),

the equation for the optimal Bayes classifier, results in the following plug-in discriminant function

$$h^{PW}(\boldsymbol{X}) = P_1 \hat{p}_1(\boldsymbol{X}) - P_2 \hat{p}_2(\boldsymbol{X}) =$$

$$\frac{P_1}{N_1} \sum_{j=1}^{N_1} N_X(\boldsymbol{X}_j^{(1)}, \boldsymbol{D}\lambda) - \frac{P_2}{N_2} \sum_{j=1}^{N_2} N_X(\boldsymbol{X}_j^{(2)}, \boldsymbol{D}\lambda). \qquad (2.49)$$

In many practical applications reported in the literature, the PW classifier appears to be one of the best classifiers and often outperforms the ANN. To illustrate the Parzen window approach, we depict the non-parametric PW estimates of the univariate densities of the two pattern classes in Figure 2.5.

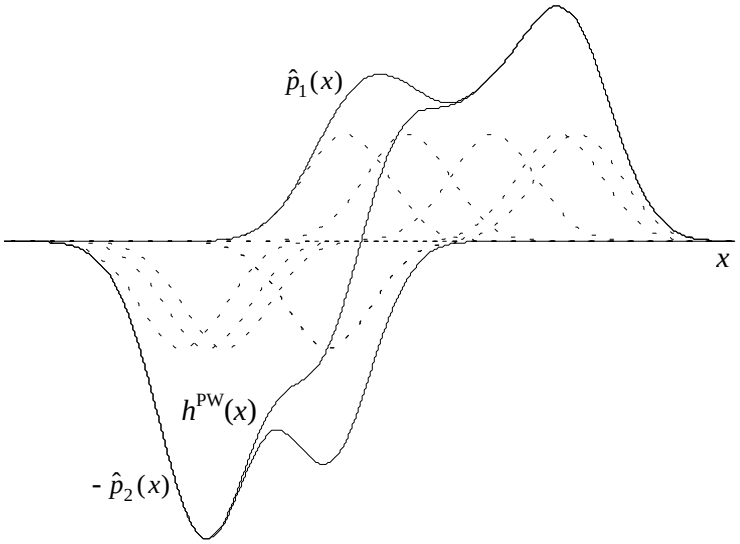

Fig. 2.5. The nonparametric Parzen window estimates of PDF of the two pattern classes and their difference – the value of the discriminant function.

A comparison of (2.49) with (2.45) shows that the two equations differ:

- in the number of elements to be summed (N_i versus m_i);
- in definition of the weighting coefficients ($1/N_i$ versus P_{ij}); and
- in the centres of Gaussian densities ($\boldsymbol{X}_j^{(i)}$ versus $\boldsymbol{C}_j^{(i)}$).

This means that the density estimators in (2.48) are in fact the mixture estimators when the centres of the subclasses are all vectors of the training-set. A generalisation of the Parzen window estimate with the Gaussian kernels (2.48) is:

$$\hat{p}_i(\boldsymbol{X}) = \frac{1}{N_i} \sum_{j=1}^{N_i} \kappa\{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})\}, \tag{2.50}$$

where $H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})$ is a distance between vectors $\boldsymbol{X}$ and $\boldsymbol{X}_j^{(i)}$; for continuous variables usually $H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)}) = (\boldsymbol{X} - \boldsymbol{X}_j^{(i)})^T \mathrm{A}_i (\boldsymbol{X} - \boldsymbol{X}_j^{(i)})$ and A_i is an $n \times n$ positive definite symmetric matrix, $\kappa(H)$ is an arbitrary monotonic decreasing function of argument H called a *kernel* or a *smoothing function*. It can be a bell-shaped function like the Gaussian function in Equation (2.48), a logistic function

$$\kappa\{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})\} = \frac{\lambda}{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)}) + \lambda}$$

uniform, trapezoidal, etc. In principle, it can even be asymmetric. In the case of continuous variables, the matrix A can be the full (not constrained) covariance matrix, the diagonal matrix, $\mathrm{A} = \boldsymbol{D}\lambda$, or an identity matrix, $\mathrm{A} = \mathbf{I}\lambda$ (Euclidean distance). In principle, the matrix A can be common or different for both classes. It can be dependent on each training vector, $\boldsymbol{X}_j^{(i)}$, too.

The parameter λ is called a *smoothing parameter*. When λ is small, then we have weak smoothing. When λ is large, the smoothing is significant and "fulfils an empty space between the multivariate training vectors".

Peculiarities of the decision boundary. Very often the spherical Gaussian kernel $\kappa\{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})\} = c_{\mathrm{PW}} \exp(-H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})/\lambda)$ with $\mathrm{A} = \mathbf{I}\lambda$ and normalising constant c_{PW} is used. By using the first two terms of Taylor series expansion, $\exp(H) = 1 + H/1! + H^2/2! + \ldots$, it is easy to show that when λ is very large, function (2.49) leads to

$$h^{\mathrm{PW}}(\boldsymbol{X}) \to \frac{c_{\mathrm{PW}}}{\lambda} \left[\frac{P_2}{N_2} \sum_{j=1}^{N_2} (\boldsymbol{X} - \hat{\boldsymbol{M}}_2)^T (\boldsymbol{X} - \hat{\boldsymbol{M}}_2) - \frac{P_1}{N_1} \sum_{j=1}^{N_1} (\boldsymbol{X} - \hat{\boldsymbol{M}}_1)^T (\boldsymbol{X} - \hat{\boldsymbol{M}}_1) \right].$$

Let $N_2 = N_1 = \overline{N}$ and $P_2 = P_1$. Then we have a linear discriminant function with weight vector $\boldsymbol{V}^{\mathrm{PW}} \equiv \boldsymbol{V}^{\mathrm{E}}$:

$$h^{\mathrm{PW}}(\boldsymbol{X}) \to \frac{c_{\mathrm{PW}}}{\lambda^2} \left[\boldsymbol{X}^T \boldsymbol{V}^{(\mathrm{E})} - \frac{1}{2} (\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)^T \boldsymbol{V}^{(\mathrm{E})} + \frac{1}{2} \left(1 - \frac{1}{\overline{N}}\right) (\mathrm{tr}\,(\hat{\boldsymbol{\Sigma}}_2 - \hat{\boldsymbol{\Sigma}}_1)) \right],$$

as $\lambda \to \infty$. (2.51)

Comparison of (2.51) with the weight vector of EDC (1.2) indicates that only the threshold value v_0 is different from that of the Euclidean distance classifier, i.e. the decision boundary is parallel to the discriminant hyperplane of the EDC. A distance between the two hyperplanes depends on a difference of traces (tr) of two sample covariance matrices, $\hat{\boldsymbol{\Sigma}}_1$ and $\hat{\boldsymbol{\Sigma}}_2$. Thus, this distance is a random variable which varies with each new training-set.

A contribution of each single training vector $\boldsymbol{X}_j^{(i)}$ to the value of DF (2.49) depends on distance $H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})$. While using the Gaussian kernel $\exp(-H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})/\lambda)$, the contribution of vector $\boldsymbol{X}_j^{(i)}$ diminishes exponentially with $H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})/\lambda$. Therefore, for very small λ, we have one nearest neighbour rule.

Example 4. In Figure 2.6, we plot a scatter diagram of $N = 100$ bi-variate vectors of two non-overlapping pattern classes. An algorithm that generates the classes is presented in subsection 2.8.3 (see Figure 2.12*a*). The straight line (1), in Figure 2.6, corresponds to the decision boundary of the Euclidean distance classifier. The straight line (2) is the decision boundary of the Parzen window classifier with Gaussian kernel and great smoothing ($\lambda = 1000$). It is parallel to line (1). For $\lambda = 2$, we have a smooth non-linear decision boundary (curve 3) and for $\lambda = 0.001$ the decision boundary is close to that of the 1-*NN* (one nearest neighbour) rule: we see the decision boundary is placed approximately in the middle of the closest training vectors of opposite classes.

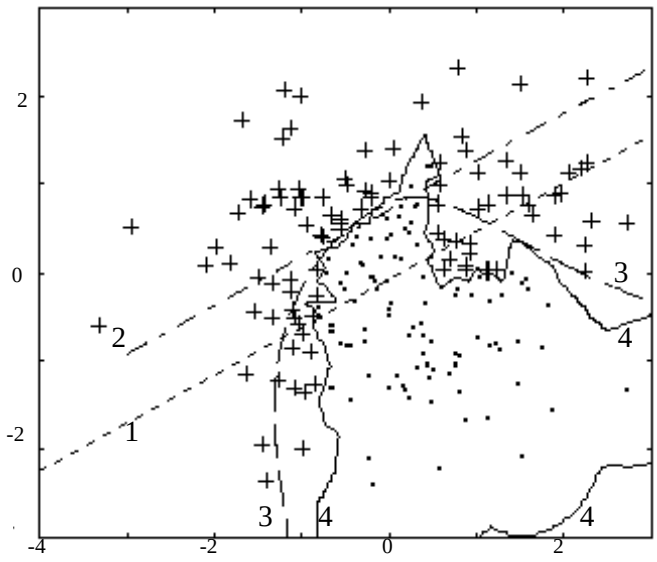

Fig. 2.6. Effect of the smoothing parameter λ on a complexity of the PW classifier: 1 is the decision boundary of the EDC; 2 is PW with $\lambda = 1000$; 3 is $\lambda = 2$; and 4 is $\lambda = 0.001$.

We see that the classification rule and complexity of the nonparametric Parzen window estimator depends severely on the value of the smoothing parameter λ. For large values of λ, we have a simple decision boundary and for small values of λ, we have a complex one. One of the main problems while designing a classifier

which uses the Parzen window approach, is to choose the proper complexity of the classifier, i.e. the value of the smoothing parameter λ.

In other variants of the Parzen window classifier, a value of the parameter λ can be constant in the entire multivariate feature space Ω or may vary in it. The parameter λ can be determined *a priori*, or it can be found from the information contained in the training-set.

2.6.4 The *k-NN* Rule and a Calculation Speed

Suppose now, that in an *n*-dimensional sphere, S_r, the output of the kernel function $\kappa(\alpha)$ is a constant value and, outside the sphere, it equals zero. In other words, we suppose that the "measurement noise" is uniformly distributed in an *n*-dimensional sphere in the *n*-variate feature space Ω. Let a radius of this sphere vary in Ω, so that exactly k training vectors are contained in the sphere. Then $\hat{p}_1(\boldsymbol{X}) / \hat{p}_2(\boldsymbol{X})$, a ratio of the two PDF estimates at the fixed point $\boldsymbol{X}$ of the space, becomes proportional to the ratio of n_1, a number of the first class training vectors in the sphere, against n_2, a number of the second class training vectors in this sphere. Thus, allocation of an unknown vector $\boldsymbol{X}$ to one of the classes should be accomplished according to class membership of the majority of training vectors in the sphere containing exactly k vectors. Such a classification rule is called a *k-nearest neighbour rule* and its shorthand definition is the *k-NN* rule.

The parameter, k, is very important: it determines the complexity of the classifier i.e. the amount of smoothing in the estimation of the multivariate density functions. Determination of an optimal number of neighbours is a very weighty problem while designing the *k-NN* classifier. When k =1, we have a simple nearest neighbour classifier.

The *k-NN* rules can differ in k i.e. the number of nearest neighbours. Different modifications of this classifier use different types of metrics in which the distances between the vector $\boldsymbol{X}$ and its neighbours from the training-set are measured. When training-set vectors that are classified incorrectly by the *k-NN* rule are deleted from the set, an increase in the performance of the new rule has sometimes been noticed. Various heuristics to delete "unnecessary" training observations exist. In order to reduce the number of calculations necessary to make classifications, some authors propose using only typical training vectors from each pattern class. Other authors propose excluding training vectors that are far from a decision boundary. Similar ideas are also used in the ANN design.

The *k-NN* classifier requires the calculation of distances between $\boldsymbol{X}$, the vector to be classified, and all $N = N_1 + N_2$ training vectors. Therefore, it is slow in the classification phase. Several special calculation schemes for a fast search for the nearest neighbours have been proposed in the literature. The ideas for reducing the classification time of the *k-NN* classifier can also be used for the PW classifier.

One of the possibilities for reducing the calculation time required to make the classification is to use *multivariate histograms*. The main idea of designing the multivariate histograms is to split the multivariate space Ω into "hyper-rectangular" spaces, i.e. cells $\Omega_1, \Omega_2, \ldots, \Omega_m$, and for each class to calculate the numbers of training observations that fall in each cell. An allocation of an

unknown object is performed according to class membership of the majority of training vectors of one class in the cell that contains the object to be classified. The simplest way to design the multivariate histogram classifier is to determine one or several threshold values for each variable and to remember the number of training vectors in each cell. Then the class indexes $\gamma_1, \gamma_1, ..., \gamma_m$, should be determined for all m cells. Such a classifier is in fact a classifier for categorical variables. We analyse this category of classification methods in Section 2.9.

We emphasise once more that the nonparametric PW and the *k-NN* classifiers are closely related among themselves, to the radial basic function and to the learning vector quantisation classifiers used in modern ANN technology.

2.6.5 Polynomial and Potential Function Classifiers

In order to reduce the classification time of the PW classifier, we can use different expansions of the kernel function. These expansions lead to *polynomial* and *potential function classifiers* where we do not need to store all of the training vectors in the computer memory at one time.

In the **polynomial classifiers**, we utilise the Taylor series expansion of the kernel function. For the kernel $\kappa\{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})\} = c_{\text{PW}} \exp(-H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})/\lambda)$ with the Euclidean metric $H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)}) = (\boldsymbol{X} - \boldsymbol{X}_j^{(i)})^T(\boldsymbol{X} - \boldsymbol{X}_j^{(i)})$, we can write

$$\hat{p}_i(\boldsymbol{X}) = \frac{c_{\text{PW}}}{N_i} \sum_{j=1}^{N_i} \exp\{(-\frac{1}{2}(\boldsymbol{X} - \boldsymbol{X}_j^{(i)})^T(\boldsymbol{X} - \boldsymbol{X}_j^{(i)})/\lambda)\} =$$

$$\frac{c_{\text{PW}}}{N_i} \exp\{-\frac{1}{2}\boldsymbol{X}^T\boldsymbol{X}/\lambda\} \sum_{j=1}^{N_i} \exp\{-\frac{1}{2}\boldsymbol{X}_j^{(i)T}\boldsymbol{X}_j^{(i)}/\lambda\} \exp\{\boldsymbol{X}^T\boldsymbol{X}_j^{(i)}/\lambda\}.$$

Having applied the Taylor series expansion $\exp\{c\} = 1 + c + c^2/2 + \ldots$ for the last term of the above representation, we obtain

$$\hat{p}_i(\boldsymbol{X}) = \frac{c_{\text{PW}}}{N_i} \exp\{-\frac{1}{2}\boldsymbol{X}^T\boldsymbol{X}/\lambda)\} \{1 + \frac{1}{\lambda}\mathbf{A}^T\boldsymbol{X} + \frac{1}{\lambda^2}\boldsymbol{X}^T\mathbf{B}\boldsymbol{X} + \ldots\}, \tag{2.52}$$

where vector column **A** and matrix **B** are certain functions of the training vectors and do not depend on $\boldsymbol{X}$, the vector to be classified.

A complexity of the decision boundary depends on a number of terms utilised in the expansion. From the other side, representation (2.52) can be used only when values $\boldsymbol{X}^T\boldsymbol{X}/\lambda$ are small, i.e., for large λ.

In the **potential functions** approach, we utilise following expansion

$$\kappa\{H(\boldsymbol{X}, \boldsymbol{X}_j^{(i)})\} = \sum_{s=1}^{r} a_s(\boldsymbol{X}_j^{(i)}) b_s(\boldsymbol{X}), \tag{2.53}$$

where $a_s(\boldsymbol{X}_j^{(i)})$ and $b_s(\boldsymbol{X})$, $(s = 1, 2, \ldots, r)$ are *a priori* selected sets of functions.

Then

$$\hat{p}_i(\boldsymbol{X}) = \sum_{s=1}^{r} v_{is}\, b_s(\boldsymbol{X}).$$

In a case where the representation (2.53) is exact, $v_{is} = \frac{1}{N_i} \sum_{j=1}^{N_i} a_s(\boldsymbol{X}_j^{(i)})$.

In a case where the representation (2.53) is approximate, we can find unknown coefficients $v_{i1}, v_{i1}, \ldots, v_{ir}$ by minimising a leaving-one-out approximation error

$$\frac{1}{N_i} \sum_{l=1}^{N_i} \Big(\sum_{s=1}^{r} v_{is}\, b_s(\boldsymbol{X}_l^{(i)}) - \frac{1}{N_i - 1} \sum_{j=1\ j\neq l}^{N_i} \kappa\{H(\boldsymbol{X}_l^{(i)}, \boldsymbol{X}_j^{(i)})\} \Big)^2.$$

The potential function approach is in fact the mapping of the input vectors $\boldsymbol{X}$ into the new r-variate space by means of an *a priori* selected set of functions. Therefore, minimisation of the above sum of squares error leads to a linear discriminant function in the new feature space. Therefore, in this classifier only new sample means and sample covariance matrices are utilised to find the coefficients of the decision rule. The artificial neural networks and the support vector approaches can give one many ideas and can lead to new algorithms which can improve solutions.

2.7 Minimum Empirical Error and Maximal Margin Linear Classifiers

Often an objective of the classifier design is to obtain a rule which minimises the probability of misclassification. For known class conditional densities, the representation (2.6) can be used to obtain the classifier with the minimal probability of misclassification. The type of the classifier depends on the assumptions about the conditional densities.

In some cases, special efforts should be made in order to obtain a simple linear classifier instead of a more complex non-linear decision rule that minimises the probability of classification. As an example, we have mentioned the Anderson–Bahadur linear DF (2.13). In the multivariate Gaussian case, this rule is asymptotically the best one among all linear classifiers.

2.7.1 The Minimum Empirical Error Classifier

In practice, the class conditional densities are unknown, however, the designer has only the training-set available. Thus, the designer should like to obtain the classifier with the minimum number of errors while classifying the training vectors. We call such types of the classification rule the minimum empirical error (MEE) classifier. To design the MEE classifier, one needs to introduce a sample based cost function and to minimise it. One of the possible cost functions which characterises the number of misclassifications among the training-set vectors is

$$Cost = \frac{1}{N_1{}^+ N_2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} \varphi (t_j^{(i)} - \boldsymbol{V}^T \boldsymbol{X}_j^{(1)} - v_0), \tag{2.54}$$

where $t_j^{(i)}$ is the class index of the training pattern vector $\boldsymbol{X}_j^{(i)}$, $t_j^{(1)} = +1$, $t_j^{(i)} = -1$, and $\varphi(\varepsilon)$ is a threshold function, e.g.

$$\varphi(\varepsilon) = \begin{cases} 1 \text{ if } \boldsymbol{V}^T \boldsymbol{X}_j^{(1)} + v_0 < 0 \text{ or } \boldsymbol{V}^T \boldsymbol{X}_j^{(2)} + v_0 > 0 \\ 0 \text{ if } \boldsymbol{V}^T \boldsymbol{X}_j^{(1)} + v_0 > 0 \text{ or } \boldsymbol{V}^T \boldsymbol{X}_j^{(2)} + v_0 < 0 \end{cases} \tag{2.55}$$

The threshold function is not continuous. Therefore, it impossible to minimise the cost function (2.54) with (2.55) using ordinary numerical methods. To minimise the empirical error, instead of minimising (2.54) with (2.55), Do-Tu and Installe (1978) proposed using a smooth analogue of the threshold function

$$\varphi(\varepsilon,\sigma) = (-1)^{i-1} \int_{-\infty}^{\varepsilon} h(\sigma, t)dt \text{ , with } \quad 0 \le h(\sigma, t) \le 1 \text{ ,} \tag{2.56}$$

where $h(\sigma, t)$ is a window function, e.g., a zero mean Gaussian density with variance σ^2.

The minimisation of the cost (2.54) with (2.56) can be performed in an iterative way. The parameter σ is decreasing during the training process. Then, with an increase in the number of iterations, the function $\varphi(\varepsilon,\sigma)$ gradually approaches the threshold function (2.55) and the cost function (2.54) eventually minimises the empirical error more and more precisely. In Chapter 4, we will show that in the non-linear SLP training, we can obtain a similar effect automatically.

A number of other authors have tried to find the coefficients of the linear discriminant function that minimised the empirical classification error**.** Their algorithms are based on the use of a sequential random search procedure, the Kiefer–Wolfowitz stochastic approximation, linear programming, the algebraic method, linear transformations of the co-ordinate system, and a number of other bright heuristic ideas. Ad hoc principles are used to overcome numerical difficulties. Thus, in one of their algorithms, Vapnik and Chervonenkis (1974), after applying a conventional algorithm to find the weights of the linear discriminant function, sequentially excluded training pattern vectors that were at the farthest distances from a current discrimination boundary $\boldsymbol{V}^T\boldsymbol{X} + v_0 = 0$ and were classified incorrectly. At the end, one obtains the classifier that minimises the empirical error approximately.

2.7.2 The Maximal Margin Classifier

When the number of empirical errors is zero, the resulting discriminant function is not unique. Numerous hyperplanes can classify the training vectors without error.

Some additional criteria should be introduced in order to obtain a unique solution. Typically, a minimal distance between the discriminant hyperplane and the training set vectors is maximised. In order to find the optimal weight vector $\boldsymbol{V}$ in their "generalised portrait" linear classification method, Vapnik and Chervonenkis (1974) suggested minimising the quadratic form

$$\boldsymbol{V}^T\boldsymbol{V} \tag{2.57}$$

under the constraints $\boldsymbol{V}^T\boldsymbol{X}_j^{(1)} \geq \Delta$ and $\boldsymbol{V}^T\boldsymbol{X}_j^{(2)} < -\Delta$, (j = 1,2, ... , N_i ; i = 1,2). Δ is a positive constant, a bound for the margin.

In reaching a unique solution, they used the quadratic optimisation techniques. This classification algorithm is called a *maximal margin classifier*. There are a fixed number of training vectors equally distant from the discriminating hyperplane. These vectors are called *support vectors* and a fixed set of the support vectors determines a position of this hyperplane (Figure 2.7).

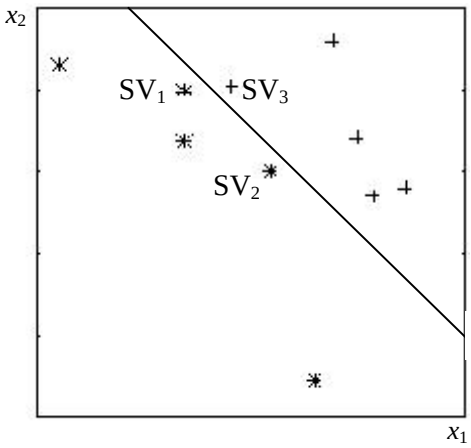

Fig. 2.7. Ten bi-variate training vectors. Only three of them (support vectors SV_1, SV_2 and SV_3) determine the position of the optimal (maximum margin) decision boundary (the dotted line).

2.7.3 The Support Vector Machine

A generalisation of the maximal margin classifier is a support vector (SV) machine that implements the following idea: it maps the input vectors $\boldsymbol{X}$ into a high dimensional feature space through non-linear mapping via a chosen *a priori*. In this space, the maximal margin separating hyperplane is constructed. To map the input vectors $\boldsymbol{X}$ into the high dimensional feature space, one can use polynomial functions, radial basic functions, outputs of the hidden layer of the multilayer perceptrons, etc.

Example 5. In Figure 2.8*a*, we depict a distribution of 15 + 15 bi-variate training vectors from two artificially generated pattern classes and the decision boundary of the SV machine designed in a five-variate space of the features $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_1^2$, $y_4 = x_2^2$, $y_5 = x_1x_2$. The two-stage decision-making scheme is presented in Figure 2.8*b*. Here the first stage maps bi-variate input vectors $\boldsymbol{X}$ into the five-dimensional feature space through the five polynomial functions. In the second stage, the linear maximal margin classifier is realised.

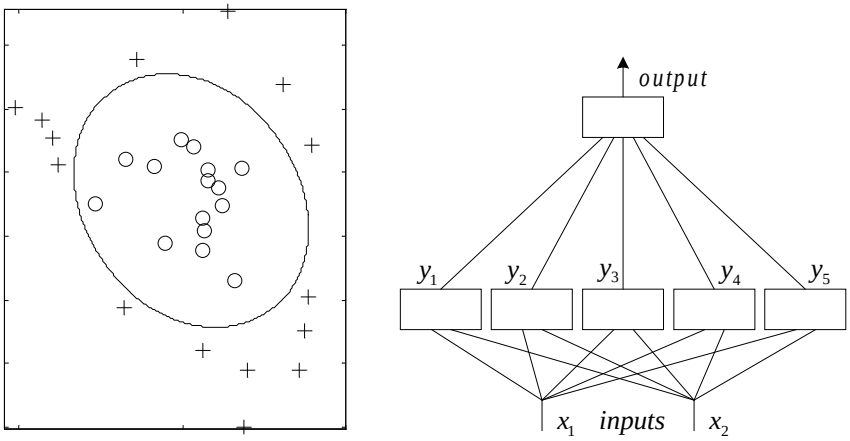

Fig. 2.8. A distribution of 15 + 15 bi-variate training vectors from two artificially generated pattern classes and the decision boundary of the support vector machine designed in five-variate space of the new features $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_1^2$, $y_4 = x_2^2$, $y_5 = x_1x_2$ (an optimal boundary is an ellipsis) and the decision making scheme (right).

The maximal margin classifier results in maximal distances of the training vectors from the resulting hyperplane. In the new high dimensional space, we also have the maximal margin. However, we cannot say that in the original *n*-variate space, the distances of a few of the closest training vectors to a non-linear decision boundary are equal among themselves and cannot be increased further. In order to find the optimal weight vector Cortes and Vapnik (1965) suggested a smart method to minimise the quadratic form in the original feature space. Instead of using the quadratic optimization technique to find the weights of the linear classifier in the second stage of the decision making algorithm, in principle, one can use other statistical methods, train the SLP classifier, etc.

2.8 Piecewise-Linear Classifiers

Like the nonparametric classifiers, the piecewise-linear (PWL) decision boundaries allow one to discriminate classes having complex multimodal distribution densities. The piecewise linear classifiers have gained great attention in pattern recognition theory and practice. A very large number of methods based

on formal principles and bright heuristic ideas have been proposed during the last three decades. A decision making scheme of the piecewise-linear classifiers resembles that of the multilayer perceptron with one hidden layer. The learning vector quantisation neural networks also realise the piecewise-linear decision boundaries. The weights that determine the piecewise-linear classifiers can be used to initialise the weights of the hidden layer units of MLP. It is the reason why we pay special attention to these algorithms in this book.

In principle, there are two approaches to designing the piecewise-linear classifiers. In the first approach, the designer begins from the *description of the classes*. In the second, he begins from the *description of hypothetical decision boundaries* (the structural or architectural approach).

2.8.1 Multimodal Density Based Classifiers

In the first approach to designing the piecewise-linear classifiers, it has been traditional to suppose that each class is composed from a mixture of the components. Each component can be represented by the sub-population or by a single multivariate pattern vector that is close to a border between the pattern classes. Then one needs to find the linear decision sub-boundaries that classify the sub-populations, the single pattern vectors or the groups of single pattern vectors of the opposite classes. It is not difficult to do this if one chooses *exact definitions* of the sub-populations, single pattern vectors or groups of single pattern vectors of different classes. The methods used to design the linear classifiers, described already in this chapter, can be used for this purpose. In accordance with the selection of the sub-populations, single pattern vectors or groups of single pattern vectors of different classes and the classifier design method used, one can acquire a piecewise-linear or even piecewise-quadratic decision boundary.

Example 6. We have had the PWL classifier designed according to the first approach in Section 2.6.2. There we assumed that each of the m_i components of the mixture is an independent subclass. In Figure 2.9 we depict two pattern classes, one having a three-modal density with components ω_{11}, ω_{12}, ω_{13} and the other having a bi-modal density with components ω_{21}, ω_{22}. A piecewise-linear decision boundary is composed from three piecewise-linear boundaries: h_{11}, h_{22}, and h_{32}. Each of the elementary first level discriminant functions $h_{ij}(\boldsymbol{X})$ allocates an unlabeled vector $\boldsymbol{X}$ into one of subregions Ω_{ij} or $\overline{\Omega}_{ij}$. A final decision is made on the basis of elementary decisions by a special output stage classifier.

A hierarchical decision-making scheme for the "mixture" model of the piecewise-linear classifier is displayed in Figure 2.10. Decision element H_j performs a linear operation to calculate a "pseudo-distance" y_j between an unlabeled input vector $\boldsymbol{X}$ and the sub-population ω_j^* with the mean vector $\boldsymbol{M}_j^*$

$$y_j = -2\,\boldsymbol{X}^T\boldsymbol{M}_j^* + \boldsymbol{M}_j^{*T}\boldsymbol{M}_j^* . \tag{2.58}$$

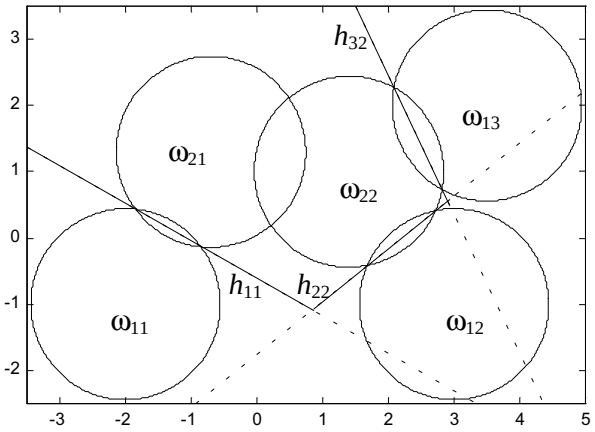

Fig. 2.9. Two bi-variate populations, one having a bi-modal density and the other having a three-modal density, and a piecewise-linear decision boundary.

In the Euclidean distance measure $H^{E}(\boldsymbol{X}, \boldsymbol{M}_j^*) = (\boldsymbol{X} - \boldsymbol{M}_j^*)^T(\boldsymbol{X} - \boldsymbol{M}_j^*)$, term $\boldsymbol{X}^T\boldsymbol{X}$ is common for all first level elements H_j. Therefore, Equation (2.58) ignores this term. Hence, the first level elements are linear. Elements "$\min_j$" in the Figure 2.10, choose a minima from each group of the first level elements. The output decision element "min" determines which minimum is the smallest one and allocates the vector $\boldsymbol{X}$ to the corresponding class.

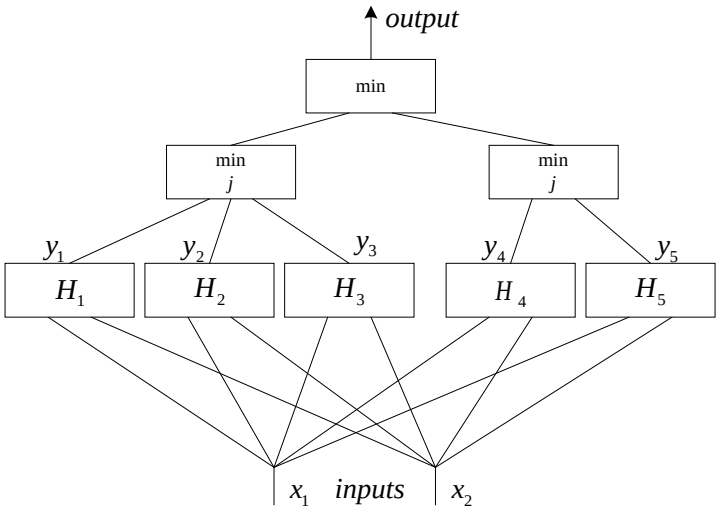

Fig. 2.10. Decision-making scheme for the "mixture model" piecewise-linear classifier.

The only training required to design the classifier is a determination of the mean vectors of the subclasses. This can be done, for example, by using one from a great amount of known cluster analysis methods. Instead of the Euclidean distance , one can use more sophisticated distance measures, e.g. the metric $H^{\mathrm{A}}(\boldsymbol{X}, \mu_{ij}) = (\boldsymbol{X} - \boldsymbol{M}_j^*)^T \mathrm{A} (\boldsymbol{X} - \boldsymbol{M}_j^*)$, where A stands for an $n \times n$ matrix. In the learning vector quantisation (LVQ) network approach, unknown parameters are learned in an iterative procedure where a certain cost function is minimised.

2.8.2 Architectural Approach to Design of the Classifiers

In the multimodal density based approach, an architecture of the decision making algorithm follows from the description of the pattern classes. In the architectural approach, it is supposed that the functional form (architecture) of the piecewise-linear decision-making scheme is known *a priori.* Usually it is supposed that the piecewise-linear classification is performed in two stages. In the first stage, there is a certain number of the linear classifiers which perform classification in the *n*-variate feature space. In the second stage, there is an additional decision-making element that groups decisions already made in the first stage. Let the first stage's linear classifiers H_i generate outputs

$$y_i = \begin{cases} 0 & \text{if} \quad v_o + \sum_{i=1}^{n} v_i x_i > 0 \\ 1 & \text{otherwise} \end{cases}.$$

In the second stage, a final classification is performed. In principle, one can use any known parametric or nonparametric classifiers for the categorical data. In the next section, we will consider algorithms that can be utilised while making the final decision. The simplest example for binary solutions (0 or 1) made by the second stage element can serve the linear or quadratic classifiers, e.g.

$$output = \begin{cases} 0 & \text{if} \quad w_o + \sum_{i=1}^{h} w_i y_i > 0 \\ 1 & \text{otherwise} \end{cases}.$$

A hierarchical decision making scheme which performs such classification is presented in Figure 2.11.

2.8.3 Decision Tree Classifiers

Nowadays, decision tree classifiers are among the most powerful and most popular classification techniques used to work with complex continuous, discrete and mixed data. This technique's decision-making scheme is very similar to the feed-forward multilayer perceptron. In order to choose a proper architecture of the multilayer perceptron it is important to know the ideas used to design the decision tree classifiers. Solutions of the MLP classifier can be represented as a decision

tree, and *vice versa*, i.e., the decision tree can be utilised to fix MLP architecture and its initial weights.

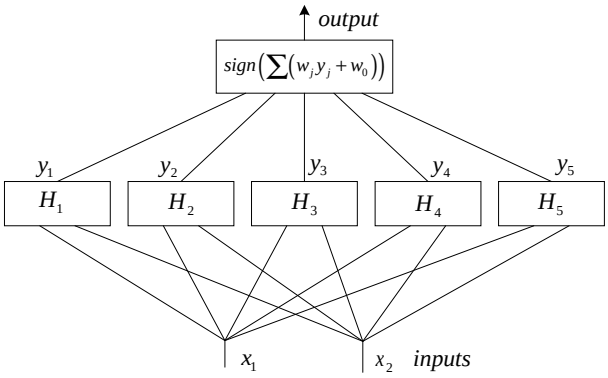

Fig. 2.11. Decision-making scheme in the "architectural" piecewise-linear classifier.

To make a solution concerning the class membership of the vector ***X*,** the decision tree classifier most often uses only one feature at a time and allocates the vector ***X*** to one of the branches of the tree. Then it is sent further to the next branch, etc., until a final decision is made. In more complicated algorithms, more than one feature takes part in decision-making. Typically linear classifiers are used in each decision, however, decision tree design algorithms have been proposed where quadratic classifiers are utilised.

Example 7. In Figure 2.12, we present a scatter diagram of the artificial data and the decision tree classifier. The decision is performed by feature x_1 at first. In this stage, conditions $x_1 < a$ or $x_1 > b$ are verified. If $x_1 < a$, we have a final (terminal) node: we allocate vector ***X*** to the class "×". If $a \le x_1 \le b$, we go on to the second decision stage, and verify a condition: if $h(\boldsymbol{X}) = v_1 x_1 + v_2 x_2 = < d$ ($v_1 = -1$, $v_2 = 1$, $d = 1$), we allocate the object ***X*** to the class "+" if $v_1 x_1 + v_2 x_2 \ge d$, we allocate object ***X*** to the class "×". If, at the very beginning, we have had an answer $x_1 > b$, in the second stage we verify: if $x_2 < c$, , we decide on class "+", and if $x_2 \ge c$, then class "×".

The decision tree classifier can easily be used to classify objects characterised by the features, some of which are continuous and the rest are discrete. Such variables are called *mixed* and such situations often occur in applications. There is a great number of algorithms proposed to design the decision tree classifiers. In order to design the decision tree, at each node one needs to specify the set of features, the number of classes, the type of the classifier, and its parameters. In addition, one needs to specify the complexity and the architecture of the tree: the geometry, the number of nodes, and the number of final leaves. There is a tremendous number of ways to obtain the decision tree classifiers of different configurations and different complexity. The decision trees can differ according to:

– strategies utilised to make the decisions at each single node (the type of decision made at each node; in our example, we had two types of decision made by the single features and by the linear discriminant function);
– metrics (criteria) to determine the "most powerful" feature or the subset of features in order to make a current decision in each node (e.g., the Mahalanobis distance, the empirical probability of misclassification, a divergence, etc., see Chapter 6);
– a metric used to determine parameters of the single classifiers in the nodes;
– a way to design the architecture of the tree and its complexity;
– a criterion to determine the classification performance in order to determine the complexity of the decision tree, a rule to stop splitting;
– a type of procedure to optimise the decision tree just designed.

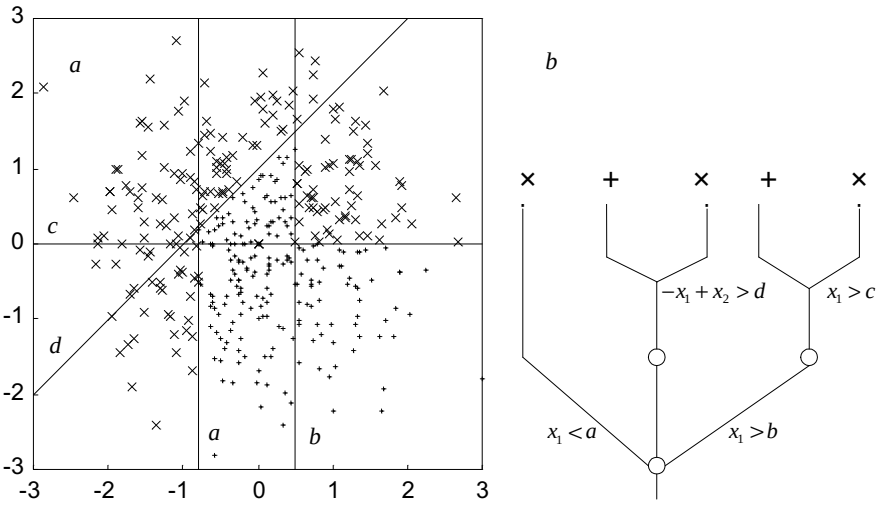

Fig. 2.12. *a*) distribution of two class training vectors in the bi-variate feature space; *b*) the decision tree classifier.

We see that the decision tree classifier is not a *unique* decision-making scheme. For the same training-set one can design an enormous number of the decision tree classifiers of different complexity. A knowledge about decision tree design strategies is very important for the ANN designers. We remind the reader that the decision-making scheme of the multi-layer perceptron can be approximated by the decision tree classifier and *vice versa*, i.e. the decision tree classifier can be represented as the MLP.

Example 8. In Figure 2.13*a*, we present a scheme of the MLP with *two hidden layers* that classify the data presented in Figure 2.12*a*. Four neurones in the first hidden layer are verifying the following conditions: $x_1 < a$ (A), $x_1 < b$ (B), $-x_1+x_2<d$ (C), $x_2 < c$ (D). Five neurones in the second hidden layer perform an operation "AND" and two output neurones, an operation "OR". At first, this MLP

was composed without training, directly from the decision tree architecture and its parameters. The coefficients that describe the decision tree solution were used as the *initial values* of the MLP classifier. Then MLP was trained by using the back-propagation technique. As a result, we obtain a smooth decision boundary (curve in Figure 2.13*b*). After multiplication of all weights of the hidden layer by factor 10000 we obtain a piecewise-linear decision boundary that can be utilised to form a new decision tree where both features, x_1 and x_2, are utilised in each node. Four dotted straight lines represent the single linear decision boundaries of four hidden layer neurones.

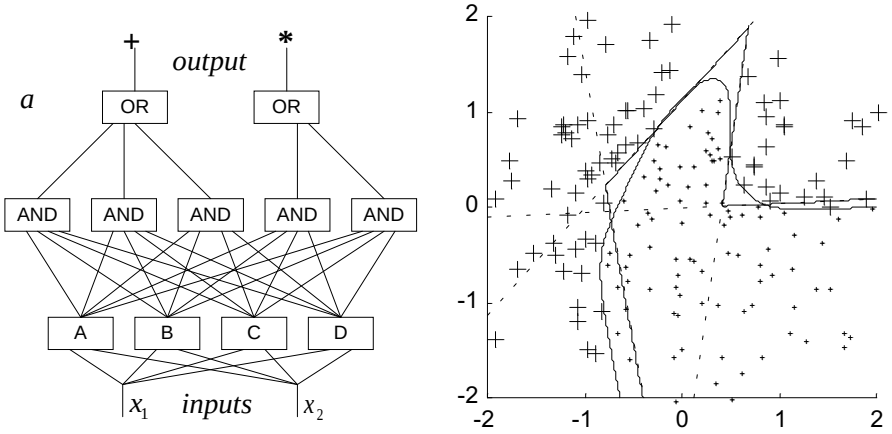

Fig. 2.13. The MLP with two hidden layers which reproduces the decision boundary of the decision tree classifier in Figure 2.12*b* (*a*) and the decision boundary of the MLP with one hidden layer, four hidden units, and the hard-limiting and smooth activation functions.

2.9 Classifiers for Categorical Data

In many applications, the features are not continuous but discrete: each measurement can assume only several possible values (states, cells or bins), such as, a profession, a sex, the type of an engine, etc. In ANN design, we are faced with categorical variables when we design a co-operative neural network composed from several expert networks and one "boss" network. The boss (a gating rule) should analyse categorical judgements of the expert networks (indexes of the pattern classes to which an unknown input vector $\boldsymbol{X}$ has been assigned by each expert network) and make a final decision.

2.9.1 Multinomial Classifiers

Let the *j*-th variable assume m_j possible values (states, bins). Then a total number of theoretically possible bins of *n*-variate vector $\boldsymbol{X}$ will be $m = \prod_{j=1}^{n} m_j$. Each

particular n-variate vector $\boldsymbol{X}$ can assume only one bin from m possible ones, say, s_j -th bin (j = 1, 2, …, m). In order to design the classification rule, we assume that n-variate vector $\boldsymbol{X}$ is a random one and use the theory of statistical decisions. Denote a probability that vector $\boldsymbol{X}$ from ω_i has assumed state s_j by $P_j^{(i)}$ (j=1, 2, …, m). In the two category case, the conditional distribution of vector $\boldsymbol{X}$ from class ω_i is characterised by m probabilities

$$P_1^{(i)}, P_2^{(i)}, \ldots, P_{m-1}^{(i)}, P_m^{(i)}, \quad \text{with} \quad \sum_{j=1}^{m} P_j^{(i)} = 1, \quad (i = 1, 2).$$

Let vector $\boldsymbol{X}$ be assumed to be the state s_j. Then Equation (2.5) results in the following optimal Bayes discriminant function

$$h(X) = P_1 P_j^{(1)} - P_2 P_j^{(2)}. \tag{2.59}$$

Probabilities of the states s_1, s_2, … , s_m can be described by a multinomial distribution. Therefore, the classifier in which a decision is made according to probabilities of all m cells is called a *multinomial* classifier. We index it by the letter M.

Example 9. To illustrate the above decision-making scheme, let us analyse a two category classification problem with three variables (features). Let the first and second features take only two values, 0 and 1. The third feature can take three values: 0, 1 and 2. Then a total number of possible states (bins) of the 3-variate vector $\boldsymbol{X}$ is 12 (m = 2×2×3):

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}
000	001	002	010	011	012	100	101	102	110	111	112

Let the conditional probabilities of each of these 12 states (bins) be:

s_1	s_2	s_3	s_4	s_5	S_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}
0.01	0.02	**0.07**	**0.10**	**0.17**	**0.11**	0.08	**0.17**	0.01	0.02	**0.23**	0.01
0.16	0.02	0.03	0.02	0.02	0.05	**0.16**	0.03	**0.21**	**0.13**	0.10	**0.07**

In this problem, we must allocate vector $\boldsymbol{X}$ to the first class if it will take one of the following states: s_3, s_4, s_5, s_6, s_8, s_{11} (the probabilities of these bins are in bold). If vector $\boldsymbol{X}$ takes one of states s_1, s_7, s_9, s_{10}, s_{12}, then one needs to allocate the vector $\boldsymbol{X}$ to the second class (the probabilities of these bins are also in bold). In a case where the vector $\boldsymbol{X}$ takes the second state, s_2, and the prior probabilities of the

classes are equal among themselves ($P_2 = P_1 = ½$), it is unimportant to which class it is allocated.

The asymptotic and the Bayes probabilities of misclassification are

$$\varepsilon_{\infty}^{M} = \varepsilon_B = 0.2 =$$

½ (0.01 + 0.02 + 0.03 + 0.02 + 0.02 + 0.05 + 0.08 + 0.03 + 0.01 + 0.02 + 0.10 + 0.02).

In a general case, the asymptotic and Bayes probabilities of misclassification of the multinomial classifier are

$$\varepsilon_{\infty}^{M} = \varepsilon_B = \sum_{j=1}^{m} \min \{ P_1 P_j^{(1)}, P_2 P_j^{(2)} \}. \tag{2.60}$$

2.9.2 Estimation of Parameters

In the L category case, the multinomial classifier is determined by $L \times m$ probabilities $P_j^{(i)}$ ($j = 1, ..., m, \ i = 1, ..., L$). To evaluate the unknown parameters one can use the maximum likelihood estimates:

$$\hat{P}_j^{(i)} = n_j^{(i)} / N_i , \tag{2.61}$$

where $n_j^{(i)}$ is a number of observations from the i-th class training set falling into the state s_j.

Utilisation of (2.61) in (2.59) results in the plug-in rule. In order to get the Bayes predictive rule, let us assume that the prior distribution of the probabilities $P_1^{(i)}, P_2^{(i)}, ..., P_{m-1}^{(i)}, P_m^{(i)}$ follow a Dirichlet distribution

$$P_{prior}(P_1^{(i)}, P_2^{(i)}, ..., P_{m-1}^{(i)}, P_m^{(i)}) = k_{mi} (P_1^{(i)})^{\gamma_1 - 1} (P_2^{(i)})^{\gamma_2 - 1} ... (P_{m-1}^{(i)})^{\gamma_{m-1} - 1} (P_m^{(i)})^{\gamma_m - 1},$$

where k_{mi} is a normalising constant and $\gamma_1, \gamma_2, .., \gamma_m$ are parameters of the prior distribution.

Then the Bayes estimate of $P_j^{(i)}$ is

$$\hat{P}_j^{(i)} = (n_j^{(i)} + \gamma_i) / (N_i + \sum_{j=1}^{m} \gamma_i), \tag{2.62}$$

If we do not give preference to any particular bin, then $\gamma_1 = \gamma_2 = ... \gamma_m = \gamma$. Then instead of (2.62) we will have

$$\hat{P}_j^{(i)} = (n_j^{(i)} + \gamma) / (N_i + m\gamma), \tag{2.63}$$

If γ =1, then we have a uniform distribution of the bin's (cell's) probabilities $P_1^{(i)}$, $P_2^{(i)}$, ..., $P_{m-1}^{(i)}$, $P_m^{(i)}$. If $N_2 \neq N_1$, the estimates (2.63) differ from estimates (2.61). Thus, we notice once more that the plug-in and the Bayes predictive classification rules are different.

2.9.3 Decision Tree and the Multinomial Classifiers

In the two category case, making a decision by the multinomial classifier one needs to remember all 2m conditional probabilities (essentially one needs to remember only the class memberships of each state). It is easy to do, when we have a 3-variate vector and only 12 states. In many practical cases, one can have tens and hundreds of features. When the number of features, n, is moderate or large, the number of states (bins) becomes enormously large. For example, when we have 10 variables and only one threshold (each feature can have then only two states), the number of cells $m = 2^{10} = 1024$. When we have 20 variables, then $m > 10^6$. The necessity of compressing the information arises: we need to represent a decision-making scheme by a small number of parameters.

Different algorithms for compressing this information exist. Two principal assumptions used in these algorithms are:

- many states belonging to one pattern class can be joined together, then separate states can be described by a smaller number of features;
- class memberships of the bins with zero or very small probabilities $P_j^{(i)}$ can be arbitrary.

One can also use the Boolean algebra approach and present a solution in conjunctive disjunctive form. Special algorithms to design multivariate histograms and decision tree classifiers can be used here. The multivariate histogram that splits multivariate space Ω into the "hyper-rectangular" spaces i.e., cells $\Omega_1, \Omega_2, \ldots, \Omega_m$, is in fact, the multinomial classifier. We have met the decision tree solutions in Section 2.8.3.

Example 10. In Figure 2.14, we have the decision tree classifier for the above classification problem with three categorical variables. At the beginning, the first feature x_1 makes a decision and allocates the object $\boldsymbol{X}$ to one of the branches. If the object was allocated to the right branch, a final decision is made by the third variable x_3. If it was allocated to the left branch, then the second feature x_2 makes a subsequent decision. If x_2 =1, then the object is allocated to the first class, otherwise one uses the third feature x_3. The final number of branches, i.e. leaves, of the decision tree is 5. It is significantly less than the number of bins m = 12. To make a decision, one needs to remember class memberships of the leaves and the architecture of the tree. It requires much less computer memory than the multinomial classifier. Moreover, assessments of the decision tree can be interpreted easily. The decision tree classifier design from the training data or from a set of probabilities $P_1^{(i)}$, $P_2^{(i)}$, ..., $P_{m-1}^{(i)}$, $P_m^{(i)}$, (i = 1,..., L) is a combinatorial

problem which, in practical cases, can be solved only with a certain accuracy (deviation from an optimal solution).

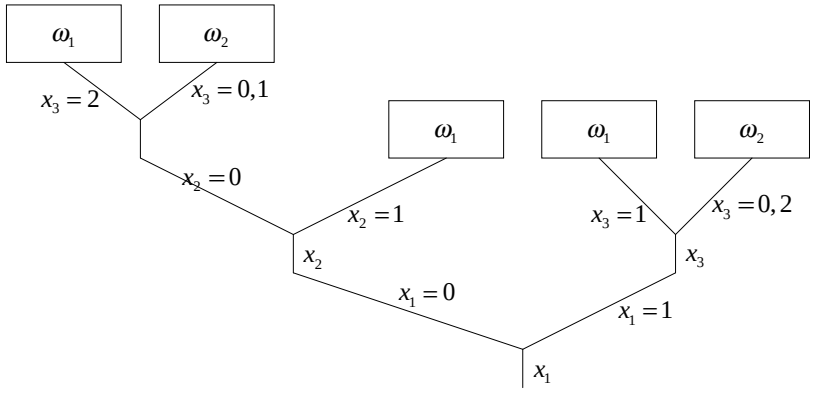

Fig. 2.14. The decision tree classifier for categorical two class data with three features.

2.9.4 Linear Classifiers

In order to overcome difficulties associated with the tremendous number of bins, the parametric methods to describe the multivariate distribution can also be used. In the statistical inference, a popular class of decision-making algorithms, when dealing with categorical data, are the latent class model and contingency tables. We mention here only one simple method.

The simplest assumption most often used to design parametric classifiers is an assumption that variables are mutually independent. Then

$$P\{x_1 = s_1, x_2 = s_2, \ldots, x_n = s_n \mid \omega_i\} = \prod_{j=1}^{n} P_j^{(i)}(s_g^j), \tag{2.64}$$

where $P_j^{(i)}(s_g^j)$ is a probability that the j-th component of the multivariate vector $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)^T$ will take the state s_g^j, $g = 1, 2, \ldots, m_j$.

The number of probabilities $P_j^{(i)}(s_g^j)$ to be estimated from the training data is $\sum_{j=1}^{n} m_j$ - n. Applying the general theory to design the classification rule, we have the following discriminant function:

$$h(\boldsymbol{X}) = \sum_{j=1}^{n} \ln \frac{P_j^{(1)}(s_g^j)}{P_j^{(2)}(s_g^j)} + \ln \frac{P_1}{P_2}. \tag{2.65}$$

To estimate the probabilities $P_j^{(i)}(s_g)$, one can use the maximum likelihood or Bayes approach (essentially it will be the equations (2.61) and (2.63) applied for each variable separately). For independent binary (with two states only) variables, we have a linear discriminant function and, for independent ternary variables, it is quadratic. In the general case, for *m*-valued conditionally independent variables we have the (*m*-1)-th degree polynomial discriminant function (Toussaint, 1972).

The binary variables can be considered as continuous. Therefore, in this case, all classifier design techniques suitable for the continuous variables can be utilised.

2.9.5 Nonparametric Local Classifiers

To estimate the distribution of a group of dependent categorical variables and design the classification rule, one can use *the non-parametric approach*: the *k-NN* and Parzen window rules. For the categorical variables, we cannot use the conventional kernels used for the continuous variables. There is a short list of distance measures useful for this purpose:

$$\varphi(H) = (\lambda)^{n - H(\boldsymbol{X}, \boldsymbol{X}_j)} (1 - \lambda)^{H(\boldsymbol{X}, \boldsymbol{X}_j)}, \qquad (2.66a)$$

where $H(\boldsymbol{X}, \boldsymbol{X}_j)$ is a distance between $\boldsymbol{X}$ and $\boldsymbol{X}_j$, e.g. a number of disagreements between these two vectors, and λ is the smoothing parameter incorporated in the distance metric already,

$$\varphi(H) = \sum_{j=1}^{n} H(x_\alpha, x_{\alpha j}), \qquad (2.66b)$$

$$\varphi(H) = \prod_{j=1}^{n} H(x_\alpha, x_{\alpha j}), \qquad (2.66c)$$

where $H(x_\alpha, x_{\alpha j})$ is a distance between the α-th components of vectors $\boldsymbol{X}$ and $\boldsymbol{X}_j$, e.g., $H(x_\alpha, x_{\alpha j}) = 1$ if $|x_\alpha - x_{\alpha j}| < \delta$ and $H(x_\alpha, x_{\alpha j}) = 0$ otherwise.

An appropriate choice of the kernel function allows the use of this non-parametric approach when we have a *mixture of continuous and categorical variables*. While constructing the classifier, we need to choose the optimal complexity of the rule, i.e. the value of the smoothing parameter λ. We will return to this question in the next chapter.

2.10 Bibliographical and Historical Remarks

Fisher (1936) was the first to propose a method of classifying multivariate vectors into several categories. Optimal discriminant functions in the case of known distribution density functions of the pattern classes were discussed by Welch

(1939) and a theory of statistical decisions is presented by Wald (1944, 1950). Anderson (1951) proposed the plug-in approach to design sample-based classification rules. He showed that the plug-in sample based classification rule is identical to the Fisher rule if $N_2 = N_1$ and the classes are multivariate Gaussian with common covariance matrix.

Anderson and Bahadur (1962) obtained the minimal classification error linear discriminant for a case when the classes are Gaussian but the covariance matrices are different. Their result was generalised by a wider class of distributions by Patterson and Mattson (1966) and Smith (1971). Later Zhezel (1975), Lipskaya (1981) and Bunke and Fisher (1983) have shown that, for two arbitrary multivariate populations, the weight vector of the discriminant function (optimal in a sense of a upper bound of the classification error) can be expressed as

$$\boldsymbol{V}_{\text{opt.}} = (\boldsymbol{\Sigma}_1 + \gamma\boldsymbol{\Sigma}_2)^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2),$$

where $\boldsymbol{M}_1, \boldsymbol{M}_2$ are mean vectors, $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ are covariance matrices, and a scalar parameter γ can be found from $\boldsymbol{M}_1 - \boldsymbol{M}_2$, and $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$.

Harley (1963), Hoerl and Kennard (1970) and DiPillo (1979) formulated the problem of regularisation of the covariance matrix in the small training-set situation. A scaled rotation regularisation was proposed by Raudys (2000*c*). For more details concerning RDA see Friedman (1989) and McLachlan (1992). A distinction of the complexity of five classification rules according to the structure of the covariance matrices was first analysed by Raudys (1972) and the block-diagonal type matrix structure was used by Goldin and Poplavskij (1970). Chow and Liu (1968) suggested the first order tree type model of the dependence between the variables for classification purposes. Zarudskij (1979, 1980) expressed the elements of the inverse of the covariance matrix as a function of elements of correlations and variances, and suggested using the Kruskal (1956) algorithm to find the graph. Different types of models of the Toeplitz matrix can be found in Anderson (1971) and Box and Jenkins (1976). Han (1970) considered the circular covariance matrix and suggested a simple way of its diagonalisation. Han (1968) and Pivoriunas (1978) considered models where all the variables are equally correlated and Kligiene (1977) has written an inverse of the covariance matrix of the autoregression stationary random process (Appendix 3).

The Bayes predictive approach to estimating the probability density function for designing the classification rule was first used by Abramson and Braverman (1962), who analysed the Gaussian case with unknown means and known covariances. Geisser (1964) and Keehn (1965) considered the case of the unknown covariance matrix. Gupta (1977) has shown that in the case where $N_2=N_1$, for uniform prior distributions of the components of the mean vector and the covariance matrix, the predictive Bayes approach results in the linear DF (1.3), in exactly the same way as the standard Fisher rule.

Schuermann (1977), Malinovskij (1979) and Duin (1995) suggested the use of the pseudo inverse of the covariance matrix while designing the sample-based classifier when $N < n$. Day (1969*b*) gave a general formula (2.40) for families of distribution densities for which the linear and quadratic classification are optimal and Day and Kerridge (1967) suggested a procedure to find unknown parameters.

Randless *et al*. (1978) proposed the use of the Huber (1981) robust M-estimates in discriminant analysis, as well as the generalised Fisher linear DF to classify contaminated populations. Other robust classification procedures for contaminated data have been considered in Kharin (1996).

Do-Tu and Installe (1978) invented the window function technique to design the classifier with smallest training-set (empirical) error. Other procedures of this group are based on the use of a sequential random search procedure (Wolf, 1966), the Kiefer-Wolfowitz stochastic approximation (Yau and Schumpert, 1968), linear programming (Ibaraki and Muroga, 1970), the algebraic method (Warmack and Gonzales, 1973), linear transformations of the co-ordinate system (Miyake, 1979) and heuristic ideas (Vapnik and Chervonenkis, 1974). When the zero empirical classification error is obtained, the resulting discriminant function is not unique. Some additional criteria have been introduced which favour an increase in the distance between the dicriminant hyperplane and the training vectors closest to it. Examples are the tolerance function (Babu and Chen, 1971), the margin function i.e. the Euclidean distance of training vector from the separating hyperplane (Glucksman, 1966; Vapnik and Chervonenkis, 1974; Boser *et al.*, 1992; Duin, 1995; and Cortes and Vapnik, 1995). The non-linear SLP classifier was first analysed as seven statistical classifiers by Raudys (1995*a*, 1996, 1998*b*).

Shlesinger (1968) and Day (1969*a*) suggested the procedure to find parameters of the mixture of multivariate Gaussian densities and proposed using this model for classification. Patrick and Shen (1971) suggested using *a priori* information on the means and covariance matrices of the subclasses. Parzen (1962) generalised the Rosenblatt (1956) histogram approach to estimate an unknown probability density function and found conditions of consistency of this estimate. Wolverton and Wagner (1969) showed that with increase in the training-set size and with decrease in the smoothing parameter, the Parzen window density estimate-based classifier tends to the optimal Bayes classifier. The nearest neighbour classification rule has been introduced by Fix and Hodges (1951) and subsequently generalised as the *k-NN* rule. Sebestyen (1962) suggested utilising the multivariate histograms for the classification. For more information about the kernel estimates see Hand (1982), Silverman (1986), Scott (1992) and Wand and Jones (1995). There are numerous references for the reduced kernel estimates, see e.g. Fukunaga and Hayes (1989) and Holmstroom and Hamalainen (1993).

The multinomial procedure to classify discrete valued vectors has been proposed in Linhart (1959) and Cochran and Hopkins (1961). Abend and Hartley (1969) suggested using the Bayes approach to estimate unknown probabilities of the states. A prototype of the decision tree classifier, i.e. a stepwise procedure to reduce the number of cells, has been proposed by Hills (1967). More information on the decision tree classifiers can be found in the monographs of Lbov (1981) and Breiman *et al.* (1984). Popular software for decision trees is prepared by Quinlan (1993). Taxonomy of the decision tree design algorithms considered in Section 2.8.3 is from Norusis (1991). A variety of different models used to design the classifiers for categorical variables and for the mixed ones can be found in Lachenbruch and Goldstein (1979).

In general, there exist several hundred algorithms to construct the classification rules. In the classical theory of statistical decision functions, an emphasis is made

on the description of the pattern classes: one estimates the conditional distribution densities $p_1(\boldsymbol{X}) = P(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_1)$ and $p_2(\boldsymbol{X}) = P(\boldsymbol{X} \mid \boldsymbol{X} \in \omega_2)$ at first and only then designs the discriminant function. We have titled this approach a *parametric* statistical approach.

A significant part of the statistical analysis performed in the field of the *structural* (artificial neural network) approach (Tsypkin (1966), Amari (1967), Levin *et al*, 1990; Amari, Fujita and Shinomoto, 1992 and Amari and Murata, 1993). In Chapter 3 we will present a short review of finite training-set size investigations obtained in a stream of this research direction. Therefore, here we present few details. In this approach, one makes assumptions on a structure of the discriminant function, i.e. *a posteriori* probability $P(o \mid \boldsymbol{V}, \boldsymbol{X})$, and on the unconditional distribution density $p(\boldsymbol{X}) = p_1(\boldsymbol{X})+p_2(\boldsymbol{X})$. We have named this a *constructional* statistical approach.

Typically $P(o=1 \mid \boldsymbol{V}, \boldsymbol{X})=1/(1+\exp(-\beta f(\boldsymbol{V}, \boldsymbol{X}))$, $P(o=2 \mid \boldsymbol{V}, \boldsymbol{X}) = 1- P(o=1|\boldsymbol{V}, \boldsymbol{X})$. The parameter β is called a "*temperature*" parameter. When parameter $\beta \to \infty$, the function $1/(1+\exp(-\beta f)$ turns out to be the threshold function (we denote this function by $\theta(f)$), and the network is *deterministic*, emitting an answer $y =1$ when $f(\boldsymbol{V}, \boldsymbol{X})>0$, and $y = -1$ otherwise. The threshold function $\theta(f)=1$, if $f > 0$ and $\theta(f)= -1$ otherwise. When β differs from infinity, then the output $P(o \mid \boldsymbol{V}, \boldsymbol{X})$ is smooth and the network is called *stochastic*.

For a further review of this approach, it is important to distinguish several cases. Suppose that there exists parameter $\boldsymbol{V}^*$ such that an ideal teacher network calculates $f_{\text{teacher}}(\boldsymbol{V}^*, \boldsymbol{X}) = f_{\text{network}}(\boldsymbol{V}^*, \boldsymbol{X})$, and generates the teacher signal t based on it. In some deterministic cases, there exists a set of the parameter vectors $\boldsymbol{V}$, all of which give the correct classification of the input vectors. In other words, there exists a neutral *empty zone* (the margin) between the vectors of two pattern classes. The teacher signal t is said to be *noiseless* if t is given by a sign of $f(\boldsymbol{V}^*, \boldsymbol{X})$ and *noisy* if t is stochastically produced depending on the value $f(\boldsymbol{V}^*, \boldsymbol{X})$, irrespective of the network itself being deterministic or stochastic. In many cases, $f_{\text{teacher}}(\boldsymbol{V}^*, \boldsymbol{X}) \neq f_{\text{network}}(\boldsymbol{V}^*, \boldsymbol{X})$ which means that there is no weight vector $\boldsymbol{V}^*$ such that function $f_{\text{network}}(\boldsymbol{V}^*, \boldsymbol{X})$ can reproduce the teacher's signal $t = \theta(f_{\text{teacher}}(\boldsymbol{V}^*, \boldsymbol{X}))$ correctly. The model $f_{\text{network}}(\boldsymbol{V}, \boldsymbol{X})$ is said to be *unfaithful* (*unrealisable*) in this case. In the classical discriminant analysis approach, we have a similar situation. For example, for Gaussian classes with different covariance matrices the quadratic DF is the optimal Bayes rule. Often the quadratic DF cannot classify the training-set without error. In fact, the teacher signal is noisy. In practice, however, in order to construct the linear classifier, the designer often assumes that the covariance matrices are common in all pattern classes. Thus, in this case, the optimal solution is unrealisable by the linear rule.

In the structural approach, one can make use of the Bayes predictive approach to find the weight $\boldsymbol{V}$. Let $p^{prior}(\boldsymbol{V})$ be a prior distribution of $\boldsymbol{V}$. Then, for a set of statistically independent input–outputs $\chi_N = [(\boldsymbol{X}_1, o_1),(\boldsymbol{X}_2, o_2), \ldots , (\boldsymbol{X}_N, o_N)]$, the joint probability density of n examples is

$$P(\boldsymbol{V}, \chi_n) = p^{prior}(\boldsymbol{V}) \prod_{i=1}^{n} P(o_i \mid \boldsymbol{V}, \boldsymbol{X}_i)\, f(\boldsymbol{X}_i).$$

By using the Bayes formula, the posterior probability density is given by

$$p^{porterior}(\boldsymbol{V} \mid \chi_n) = \frac{P(\boldsymbol{V}, \chi_N)}{\prod_{i=1}^{n} P(\boldsymbol{X}_i) \int p^{prior}(\boldsymbol{V}) \prod_{i=1}^{n} P(o_i \mid \boldsymbol{V}, \boldsymbol{X}_i) d\boldsymbol{V}},$$

and can easily be used to obtain the classification rule.

While defining the classification algorithms one ought to remember that *there is no single Bayes classification rule*. All plug-in rules are based on the Bayes formula, however they are different. The differences arise from assumptions about the structure of the distribution density function and the method utilised to estimate unknown parameters. The Parzen window classifier is also based on the Bayes formula. Asymptotically as $N \to \infty$ and $\lambda \to 0$, the PW approaches the Bayes decision rule. Consequently, in principle, it can be called the Bayes rule. One ought to distinguish the Bayes classification rule (plug-in rules) and the Bayes predictive approach where we have to make certain assumptions about the type and parameters of *a priori* distributions of the parameters.