5. Integration of Statistical and Neural Approaches

The sheer number of statistical and neural network-based classification methods induces the important and practical question: "Which classification method or methods are superior?" In this chapter we address the similarities and differences between statistical pattern classification methods and nonparametric (structural) neural network classification methods. Rather than focusing on the confrontation between these two classification paradigms, we focus on integrating them in order to utilise their particular strengths and properties and to diminish their shortcomings.

5.1 Statistical Methods or Neural Nets?

For a number of years a scientific discussion concerning the preference of parametric or nonparametric classification rules has been ongoing. A number of empirical comparisons have been made (for comments and additional references see e.g. Raudys *et al*., 1975; Michie *et al*., 1994; Duin, 1996; Cherkassky and Mulier, 1998; Sohn, 1999; Lim *et al*., 2000 and Ho and Basu, 2000). Advocates of statistical pattern recognition have insisted that a vector $\boldsymbol{X}$ is a random vector and its properties can be described by a multivariate distribution density. In the parametric classification approach, advocates assume that the density function of vector $\boldsymbol{X}$ is known. *Assumptions concerning the type of pattern-class density function* is prior information that is infused into the classifier design process. There are only a few examples of potentially useful prior information that can help improve a classification rule in the finite training-set size case: the assumption that the data is Gaussian with common covariance matrix, the assumption that 70% of the eigenvalues of the covariance matrix are equal to zero, the assumption that the features are independent, and the assumption that components of the input vector constitute a stationary time series described by the 4-th order auto-regression process. When additional (prior) information is at least approximately correct, one can use this information to reduce the classification error.

Some scientists advocate that making assumptions about the type of pattern-class multivariate distribution densities and their parameters in order to construct a classifier is unwise. They stress the fact that in real-world pattern classification and prediction problems, theoretical assumptions about the distribution density

functions are frequently violated. Instead of making assumptions about the distribution density functions of the feature vector ***X***, these scientists recommend that one make *assumptions about the structure of the classification rule.* For example, one could assume a linear discriminant function in the space of original or transformed features and then estimate the unknown coefficients (weights) of the decision rule directly from the training data. For this classification paradigm, one could choose to minimise the empirical error (empirical risk). This is a typical approach to the classifier formulation problem utilised in the artificial neural network approach. Using this approach, we do not parameterise the distribution densities. Therefore, this methodology for designing linear classifiers is often viewed as a nonparametric classifier design approach. In order to distinguish this methodology from the nonparametric density function estimation method used in statistics, we refer to this methodology as a structural approach.

In spite of the very useful features associated with including prior information in the design of a statistical parametric classifier, followers of the structural (neural net) approach continually emphasise their main argument concerning the inconsistency of prior assumptions in real-world problems. They continue to criticise statistical classifier design and the application of parametric-based classification algorithms. Following Vapnik (1995,) they "stress that for estimation with finite samples it is always better to solve a specific estimation problem (i.e., classification, regression) rather than attempt a general joint (input, output) density estimation". According to several authors "this point while obvious, has not been clearly stated in the classical texts on statistical estimation and pattern recognition" (Cherkassky and Mulier, 1998; page 55). We diametrically oppose the statement that "this point is obvious" and seek configuration and training sample size conditions where the structural classifier design approach is not flawless.

Advocates of statistical theory maintain that, when solving real-world problems, one needs to take into account the problem complexity and the training-set size. A number of case studies have been published that compares the structural and statistical approaches. The discussion between followers of these different classifier design paradigms is ongoing.

5.2 Positive and Negative Attributes of Statistical Pattern Recognition

In the statistical classifier design approach, we make assumptions about the distribution density functions of the pattern classes and we choose the performance criterion. We then utilise the training data to design an optimal decision rule.

Positive attributes of the statistical approach are:

SP1. The possibility of obtaining strictly optimal algorithms.

SP2. The possibility of using the information contained in the statistical hypothesis about type of pattern-class distribution density functions. In the case of correct strong assumptions, the training-set size can be very small.

SP3. The performance – complexity – the training-set size relationships for certain popular statistical classification algorithms is known.

Negative attributes of the statistical approach are:

SN1. In many practical cases, we do not know the correct statistical pattern-class model. The model selected can be far from reality and lead to a significant increase in the classification error.

SN2. The plug-in technique to derive sample based classification rules is not optimal in the unequal training-set size case.

SN3. The Bayes predictive technique for deriving sample based classification rules requires one to know the prior distribution densities of unknown parameters. Utilisation of "wide", almost flat prior distributions leads to classification rules that are optimal for a variety of potentially possible (realisable) pattern classification problems. These rules, however, are not optimal for any particular problem.

SN4. In many multivariate data models, both the analytical expressions for the decision rule and for the performance – complexity – the training-set size relationships are very complex and cannot be practically applied.

SN5. Complex statistical classification algorithms require one to estimate a large number of parameters from the training data and, therefore, suffer from the curse of dimensionality.

SN6. For each particular pattern-recognition problem we need to choose the best algorithm from a large number of possible candidates.

5.3 Positive and Negative Attributes of Artificial Neural Networks

In the neural network classifier design approach, we make assumptions about the type of the decision boundary and utilise the training data directly in order to determine the unknown coefficients (the weights) of the classification rule.

Positive attributes of the neural network approach are:

NP1. We do not need to make assumptions about the distribution densities and can utilise the training-set data directly in order to determine unknown coefficients (weights) of the decision rule.

NP2. The neural networks are universal approximators.

NP3. While training the SLP classifier, we can obtain several statistical classifiers of differing complexity.

NP4. In some cases, SLP, MLP and RBF based classifiers can have very good small training-set size properties.

NP5. The information contained in almost correct initial weights of the SLP classifier can be saved if we optimally stop the training process.

Negative attributes of the neural network approach are:

NN1. We must choose the architecture of the network and fix the training parameters.

NN2. We need to somehow stop training optimally.

NN3. Except for some cases associated with different stages of the SLP training and for the unrealistic error bounds, the performance – complexity – the training-set size relationships are unknown.

NN4. The training process is affected by the singularity of the data, the weights initialisation, local minima and the plateau phenomenon. Often the training process is very slow and some authors are almost correct in saying that "training the neural network is not a science, it's an art".

5. 4 Merging Statistical Classifiers and Neural Networks

5.4.1 Three Key Points in the Solution

Both the statistical approach and artificial neural networks have their positive and negative attributes. Rather than bickering about the attributes of each classifier design approach, one would be wise to exploit the positive aspects of each of them in order to create an integrated approach.

Understanding the dependence of the neural network's performance on the initial conditions is the first key to solving this problem. We can design a linear classifier and use its weights $\hat{\boldsymbol{V}}_0$, $\hat{v}_0$ as a starting point for training of the single layer perceptron. Then in further SLP training, we can obtain several classification rules of differing complexity. The second key to solving the problem is understanding of this fact and that some of the classifiers obtained in the non-linear SLP training are insensitive to the traditional normality assumptions practised in parametric statistical classification.

To design the linear statistical classifier

$$g(\boldsymbol{X}) = (\boldsymbol{X} - \tfrac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2))^T \hat{\boldsymbol{\Sigma}}_e^{-1} (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2). \tag{5.1}$$

we use the inverse of a certain sample CM estimator, say $\hat{\Sigma}_e$. In the simplest case, we assume $\hat{\Sigma}_e$ is the identity matrix and the resulting classifier is the EDC. In another instance, we use the traditional maximum likelihood estimate and obtain the standard Fisher classifier. Many other choices have been considered in Chapter 2 such as structured covariance matrix based classifiers and regularised discriminant analysis and its modifications.

Instead of using $\hat{\Sigma}_e$ to design the linear statistical classifier (5.1), we can use the matrix $\hat{\Sigma}_e$ to transform the data. After the first iteration in the new feature space one can obtain EDC, which in the original feature space corresponds to classifier (5.1). The third key is understanding that this classifier serves as good initialisation weights for further SLP training. We present more details below.

5.4.2 Data Transformation or Statistical Classifier?

Let Φ be an $n \times n$ eigenvectors matrix of $\hat{\Sigma}_e$, let Λ be its $n \times n$ diagonal eigenvector matrix, i.e. $\hat{\Sigma}_e = \Phi \Lambda \Phi^T$, and let us perform the linear data transformation

$$\boldsymbol{Y} = \mathbf{F}\, \boldsymbol{X}, \tag{5.2}$$

where $\mathbf{F} = \Lambda^{-1/2} \Phi^T$, and train the SLP afterwards satisfying the conditions E1 – E4 presented in Section 4.1. We repeat these conditions once more:

E1) the centre of the data $\hat{\boldsymbol{M}}$ is moved to the zero point,
E2) we start training from zero weights,
E3) the target is $t_2 = - t_1 N_1/N_2$ (for tanh activation function),
E4) we use total gradient training (batch mode).

Then after the first iteration we obtain the Euclidean distance classifier in the new transformed feature space

$$g^{(t=1)}(\boldsymbol{Y}) = (\boldsymbol{Y} - ½(\hat{\boldsymbol{M}}_{y1} + \hat{\boldsymbol{M}}_{y2}))^T (\hat{\boldsymbol{M}}_{y1} - \hat{\boldsymbol{M}}_{y2})) k_E, \tag{5.3}$$

where $\hat{\boldsymbol{M}}_{yi} = \mathbf{F}\, \hat{\boldsymbol{M}}_i$ (i = 1, 2) are the sample mean vectors in the transformed space and k_E is a constant.

Utilisation of $\hat{\Sigma}_e^{-1} = \Phi \Lambda^{-1} \Phi^T$, $\hat{\boldsymbol{M}}_{yi} = \Lambda^{-1/2} \Phi^T \hat{\boldsymbol{M}}_i$ and $\boldsymbol{Y} = \Lambda^{-1/2} \Phi^T \boldsymbol{X}$ allows one to represent the discriminant function (5.3) as

$$g^{(t=1)}(\boldsymbol{Y}) = (\boldsymbol{X} - ½(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2))^T \Phi\, \Lambda^{-1/2} \Lambda^{-1/2} \Phi^T (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)\, k_E =$$

$$(\boldsymbol{X} - ½(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2))^T \hat{\Sigma}_e^{-1} (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)\, k_E. \tag{5.4}$$

The above representation shows that after the first iteration we obtain the classifier that in the original feature space, $\boldsymbol{X}$, is equivalent to the standard linear Fisher classifier with the CM estimator $\hat{\boldsymbol{\Sigma}}_e$ (Equation (5.1)). Thus, after the data transformation and after the first iteration performed such that conditions E1 – E4 are satisfied, we obtain a good "natural" initial weight vector.

5.4.3 The Training Speed and Data Whitening Transformation

Let to determine the transformation matrix $\mathbf{F} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T$ we use the true population covariance matrix $\boldsymbol{\Sigma}$. The covariance matrix of the vector $\boldsymbol{Y}$ is

$$\boldsymbol{\Sigma}_{\boldsymbol{Y}} = E\,\boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T E\,\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\Phi}\,\boldsymbol{\Lambda}^{-1/2}$$

$$= \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}\boldsymbol{\Phi}\,\boldsymbol{\Lambda}^{-1/2} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\lambda}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\,\boldsymbol{\Lambda}^{-1/2} = \mathbf{I},$$

where, in this case, $\boldsymbol{\Phi}$ and $\boldsymbol{\Lambda}$ are determined by the representation $\boldsymbol{\Sigma} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\,\boldsymbol{\Phi}^T$.

This transformation is known as a *data whitening transformation*.

Let to determine the transformation matrix $\mathbf{F}$ we use the sample population covariance matrix, $\hat{\boldsymbol{\Sigma}}$, and we note that $\boldsymbol{\Sigma}_{\boldsymbol{Y}} \neq \mathbf{I}$. In this case $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}}$, the sample covariance matrix of $\boldsymbol{Y}$, becomes the identity matrix. Consequently, all n eigenvalues of the matrix $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}}$ are equal to 1. In Section 4.3.3 we have demonstrated that in situations where the data and the sample covariance matrix are almost singular, i.e. when we have an immense difference between smallest and largest eigenvalues of the covariance matrix, it is difficult, if not impossible, to ensure fast BP convergence of the training algorithm. Therefore, training the perceptron in the transformed feature space helps one to overcome the singularity problem and to speed up the convergence rate.

Suppose now that to determine the transformation matrix $\mathbf{F}$ we use a certain *sample estimator*, $\hat{\boldsymbol{\Sigma}}_e$, of the true population covariance matrix $\boldsymbol{\Sigma}$. Thus, both $\boldsymbol{\Sigma}_{\boldsymbol{Y}}$, the pattern-class covariance matrix and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}}$, the estimated covariance matrix of $\boldsymbol{Y}$, will differ from the identity matrix. If the matrix $\hat{\boldsymbol{\Sigma}}_e$ is similar to $\hat{\boldsymbol{\Sigma}}$, then $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}}$ is similar to the identity matrix. Thus, after the data transformation, the components of the new vectors $\boldsymbol{Y}$ will be almost uncorrelated and all new features will have approximately the same variances. Thus, the training rate of the SLP should be fast if training is performed in the new transformed feature space.

If, however, $\hat{\boldsymbol{\Sigma}}_e$ is an inaccurate estimator, then the following *negative consequenc*e can occur: we can obtain a large dispersion of the eigenvalues of $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{Y}}$. In such case, the training process can become slow.

5.4.4. Dynamics of the Classifier after the Data Whitening Transformation

Analysis of the SLP similar to that performed in Section 4.1.2.2 but done in the transformed feature space, $\boldsymbol{Y} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T\boldsymbol{X}$, indicates that after t iterations we have the classifier

$$g^{(t)}(\boldsymbol{Y}) = (\boldsymbol{X} - \tfrac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2))^T(\hat{\boldsymbol{\Sigma}}_e \frac{2}{(t-1)\eta}\frac{\overline{N}}{\overline{N}-1} + \hat{\boldsymbol{\Sigma}})^{-1}(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2))\,k_t \qquad (5.5)$$

Equation (5.5) shows that as the number of training iterations t increases, the covariance matrix's structurisation or regularisation effects vanish.

Thus, as t increases, the resulting decision rule approaches the standard Fisher linear discriminant function. This conclusion is precisely correct at the beginning of the training process while the perceptron's weights are small and the activation function behaves as a linear function. The fact that we are approaching the Fisher classifier means that further training of SLP can diminish negative effects of wrong assumptions used to obtain $\hat{\boldsymbol{\Sigma}}_e$. At the same time, it means that further increase in t diminishes the positive effects of the correct assumptions. It creates a need to stop training optimally.

In perceptron training, as the weights are increasing, the activation function begins to act as a non-linear function. Then we have both a robust and regularised classification rule. This rule pays less attention to outliers, e.g. atypical training patterns. The classifier progression to the robust classifier algorithms depends both on the learning step η and on the number of iterations t. Thus, in order to find the best balance between saving the information contained in the covariance matrix structure assumptions and diminishing outlier effects, we need to control both the learning step and the number of iterations.

Further training can realise a significant increase in the magnitude of the weights. Then we obtain a classification rule that minimises the empirical error. In the case that all training patterns are classified correctly, we can obtain the maximal margin (support vector) classifier in the space of transformed variables, $\boldsymbol{Y}$. In the case of non-Gaussian training data, when the SLP training is stopped optimally, the last three linear decision rules can outperform standard parametric statistical classifiers.

A very favourable attribute of the integrated approach is that we can approach the maximal margin classifier from a good standpoint. Information from correct statistical guesses about the data structure (type of the distribution density function, etc.) can be utilised. It is a result of successful initialisation.

5.5 Data Transformations for the Integrated Approach

5.5.1 Linear Transformations

In the data whitening transformation, we perform the linear transformation

$$\boldsymbol{Y} = \mathbf{F}(\boldsymbol{X} - \hat{\boldsymbol{M}}), \tag{5.6}$$

where $\hat{\boldsymbol{M}} = \frac{1}{2}(\hat{\boldsymbol{M}}_1 + \hat{\boldsymbol{M}}_2)$, $\mathbf{F} = \Lambda^{-1/2}\Phi^T$, and Λ, Φ^T are eigenvalues and eigenvectors of the sample CM estimator $\hat{\Sigma}_e$.

In Chapter 2, we reviewed several constrained CM models: the diagonal, the block diagonal, the Toeplitz model and its modifications, the first order tree dependence model, etc. In addition to the utilisation of structured matrices, one can exploit different regularised CM estimators such as:

a) "classical" ridge estimators: $\hat{\Sigma}_{\text{RDA}}{}^{(1)} = \hat{\Sigma} + \lambda\mathbf{I}$, $\hat{\Sigma}_{\text{RDA}}{}^{(2)} = \hat{\Sigma} + \lambda\,\text{diag}(\hat{\Sigma})$;

b) the scaled rotation CM estimator: $\hat{\Sigma}_{\text{SR}} = \Phi^{\alpha}(\Lambda + \lambda\mathbf{I})\Phi^{\alpha T}$; and

c) the Friedman's CM estimator for the two populations case:

$$\hat{\Sigma}_{\text{FRIEDMAN}}{}^{(1)} = \lambda_1\hat{\Sigma}_1 + \lambda_2\hat{\Sigma}_2 + (1-\lambda_1-\lambda_2)\mathbf{I}, \text{ or its modification}$$

$$\hat{\Sigma}_{\text{FRIEDMAN}}{}^{(\text{MOD})} = \Phi(\lambda_1\Lambda_1 + \lambda_2\Lambda_2 + (1-\lambda_1-\lambda_2)\mathbf{I})\Phi^T,$$

where Λ_1, Λ_2 are eigenvalue matrices of the covariance matrices $\hat{\Sigma}_1, \hat{\Sigma}_2$. In the last model, the eigenvectors matrix Φ is assumed to be common to both populations.

The number of models proposed in the multivariate statistical analysis and pattern recognition literatures is much larger. The reader can combine known models and invent new ones quite easily. Unfortunately, many good models have not been applied to practical real-world problems. One of the reasons for this is the negative attribute SN1 in Section 5.2: the model can be very inaccurate and can, therefore, lead to a significant increase in the classification error. The integrated approach, however, has the potential to reduce the effect of this shortcoming and to "rescue" these theoretical models.

After training the SLP in the transformed feature space, $\boldsymbol{Y} = \mathbf{F}(\boldsymbol{X} - \hat{\boldsymbol{M}})$, we obtain the linear discriminant function

$$g(\boldsymbol{Y}) = \hat{\boldsymbol{W}}^T\boldsymbol{Y} + \hat{w}_0, \tag{5.7}$$

where $\hat{w}_0$, $\hat{\boldsymbol{W}}$ are the perceptron's weights, $\hat{\boldsymbol{M}}$ is a vector and $\mathbf{F}$ is an $r \times n$ matrix.

To classify the unlabeled vector $\boldsymbol{x}^*$ we have two possibilities:

a) we perform the linear transformation $\boldsymbol{y}^* = \mathbf{F}(\boldsymbol{x}^* - \hat{\boldsymbol{M}})$, and use the discriminant function (5.7) to classify $\boldsymbol{y}^*$ afterwards;

b) we return the discriminant function into the original feature space and use the modified discriminant function

$$g(\boldsymbol{X}) = \hat{\boldsymbol{V}}^T \boldsymbol{X} + \hat{v}_0, \tag{5.8}$$

where $\hat{\boldsymbol{V}} = \mathbf{F}^T \hat{\boldsymbol{W}}$, and $\hat{v}_0 = \hat{w}_0 - \hat{\boldsymbol{W}}^T \mathbf{F} \hat{\boldsymbol{M}}$.

This expression for the classifier weights follows from the representation

$$g(\boldsymbol{Y}) = \hat{\boldsymbol{W}}^T \mathbf{F}(\boldsymbol{X} - \hat{\boldsymbol{M}}) + \hat{w}_0 = (\mathbf{F}^T \hat{\boldsymbol{W}})^T \boldsymbol{X} + (\hat{w}_0 - \hat{\boldsymbol{W}}^T \mathbf{F} \hat{\boldsymbol{M}}).$$

5.5.2 Non-Linear Transformations

The idea of transforming the data in order to obtain the best sample based classifier immediately following the first iteration can be extended to non-Gaussian data models. In the theory of statistics, it is well known that the non-linear transformation $x = \Psi^{-1}(F(y))$ can convert a distribution $F(y)$ of any non-Gaussian univariate random variable y into the Gaussian random variable.

The notation $\Psi^{-1}(c)$ indicates the inverse distribution function of the standard Gaussian $N(0,1)$ variable. $F(y)$ is a cumulative distribution function of y. We can use this idea to transform multivariate non-Gaussian data with independent components. We can also try to exploit the above ideas in order to transform non-Gaussian asymmetrical multivariate data with dependent components into the spherical Gaussian. We present two examples of the transformation of bi-variate non-Gausian data into the spherical Gaussian data.

Example 1. In Figure 5.1*a* we depict 500 bi-variate data points after rotation and the subsequent transformation with the non-linear transformation $y_i = \exp(x_i)$ of correlated Gaussian vectors. This is "the original non-Gaussian data" that we need to normalise and to decorrelate.

We perform this task by means of an iterative application of the non-linear transformation $x_i = \Psi_i^{-1}(\hat{\boldsymbol{F}}_i(y_i))$ to each component y_i of the vector $\mathbf{y} = (y_1, y_2)^T$ and subsequent linear transformation $\boldsymbol{Y} = \mathbf{T}\boldsymbol{X}$. $\hat{\boldsymbol{F}}_i(y_i)$ is the sample estimate of the cumulative distribution function of the component y_i. $\mathbf{T}$ is a 2×2 orthonormal matrix composed from two bi-variate eigenvectors of the sample covariance matrix for the vector $\boldsymbol{X}$ after each step of the transformation $x_i = \Psi_i^{-1}(\hat{\boldsymbol{F}}_i(y_i))$. Typically, after several cycles of the application of both transformations, we obtain distributions very similar to the spherical Gaussian distribution. Figure 5.1*b* shows the 500 data points after the first iteration of the above two-stage procedure and *c* shows them after the fourth iteration.

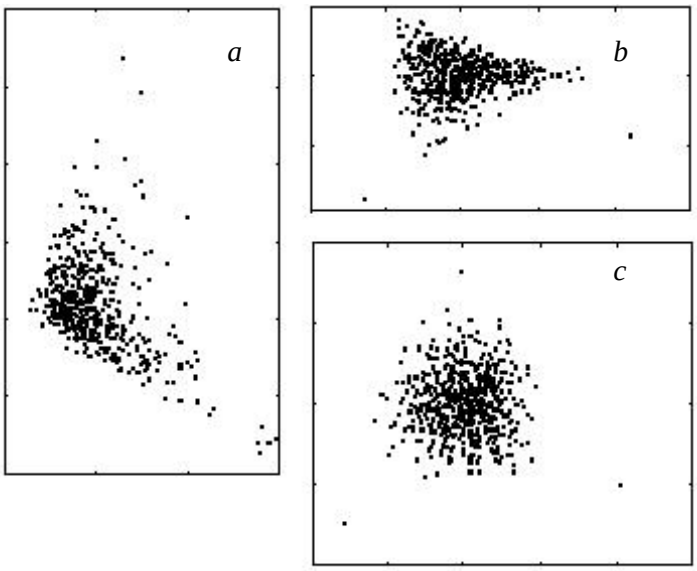

Fig. 5.1. Distributions of 500 bi-variate vectors: *a* – original data, *b* – after the first iteration consisting of non-linear "normalising" transformation and subsequent linear decorrelation, *c* – after the fourth iteration.

Example 2. We illustrate the non-linear feature transformations to solve the classification task. In Figure 5.2 we have 50 + 50 bi-variate training vectors from two pattern classes. This artificial non-Gaussian bi-variate data is obtained after application of the non-linear $x_i = \exp(\xi_i)$ transformation of two GCCM classes (i =1, 2). After an ideal (known *a priori*) normalising transformation, $y_i = \ln(x_i)$ the generalisation error is 8.5%. This is the Bayes error rate. The generalisation error of the standard linear Fisher DF is 16% and that of the non-linear SLP is 15%. In this experiment, we forget the data generation method and test two approaches to transforming the data in order to improve the classification performance.

In the first approach, we attempt to find some non-linear transformation $\boldsymbol{Y} = \Psi(\mathbf{T}\boldsymbol{X})$ that normalises and decorrelates the vector $\boldsymbol{X}$. For this we apply the iterative two-stage procedure described above. This procedure makes use of the non-linear transformation $y_i = \Psi^{-1}(\hat{\boldsymbol{F}}(z_i))$ of each single component z_i of vector $\boldsymbol{Z}$ obtained after the linear decorrelation transformation of vector $\boldsymbol{X}$: $\boldsymbol{Z} = \mathbf{T}\boldsymbol{X}$. In the two-category case, we applied this sequence of transformations only to the vectors of one pattern class. After converting vectors of the first class into approximately spherically Gaussian data, we have learned the new transformation, $\boldsymbol{Y} = \Psi^{-1}(\mathbf{T}\boldsymbol{X})$, by the MLP composed from two inputs, 12 hidden *non-linear* neurones and two *linear* neurones in the outputs. The training-set composed from the first pattern-class vectors was used to learn the non-linear feature transformation operation.

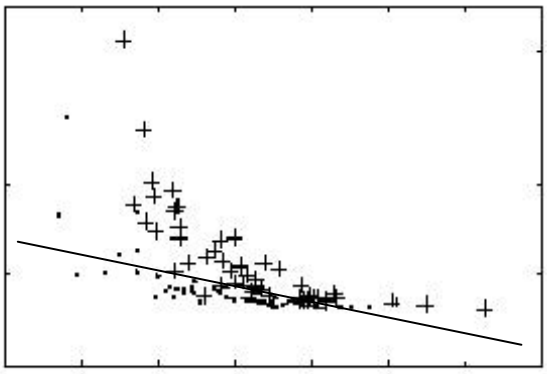

Fig. 5.2. Distribution of two highly asymmetrical bi-variate pattern classes and the decision boundary of the non-linear SLP classifier.

Later, we applied this non-linear data transformation both to the training and to the test sets. After training the SLP in the transformed (***Y***) space, we obtained only 3% errors while classifying the training-set (the resubstitution error) and 10% errors while classifying the test set. This is only slightly higher than the Bayes error (8.5% errors) and notably smaller than 15%, the generalisation error of the SLP classifier in the original (***X***) space.

The two-stage decision-making scheme just discussed consists of an MLP used for the feature transformation and of the non-linear SLP used for the classification. In classifying new vectors, the two-stage solution can be simplified. In order to bring together both stages we remark that the MLP feature transformer's outputs are *linear*. Therefore the output SLP classifier can use the 12 outputs of the hidden layer neurones of the feature transformation MLP directly as its inputs. As a result, we can devise a new simpler, single layer MLP classifier with two inputs, 12 hidden neurones, and one non-linear output. An interesting perspective arises when ones uses weights of this MLP classifier as the starting weights for further training.

The second data transformation method consists of introducing new derivative features, x_1^2, x_1x_2, and x_1^2, in addition to the original features, x_1 and x_2. We have already described this technique in Section 4.2.1. We applied the SLP to solve the above pattern classification problem in the new five-variate space and obtained the test-set error of 10%. This is the same accuracy as that found in applying the previous two-stage technique composed of the SLP and the MLP.

We have tested the "derivative features" approach on a number of other non-linear pattern classification problems and have found that in the finite training-set case, this approach often outperforms the conventional back propagation MLP classifier and the standard quadratic discriminant function.

The theoretical and experimental demonstrations presented above confirm that the linear and non-linear data transformation techniques can become powerful tools in designing simple neural classifiers with good generalisation properties. As in the feature extraction approach, we shift the most difficult portion of the pattern recognition problem into the feature definition stage.

5.5.3 Performance of the Integrated Classifiers in Solving Real-World Problems

We have attempted to evaluate the efficacy of the integration of the statistical and neural net approaches for solving real world pattern classification problems.

Example 3. In Figure 5.3 we present a characteristic example with real-world data: the vowel classification problem, with two pattern classes, 20 speakers, 28 spectral features, and the training set sizes $N_1 = N_2 = 20$. To estimate the generalisation error we used Nt = 800 vectors. We trained the SLP classifier in the original feature space (FS), curve 1, and in the transformed FS, curve 2. We used the standard linear transformation $\boldsymbol{Y} = \mathbf{F}(\boldsymbol{X} - \hat{\boldsymbol{M}})$, where $\mathbf{F}$ was obtained from the regularised sample covariance matrix. In both experiments, we shifted the training-set mean vector to the centre of the co-ordinates, used the standard sum of squares cost function with the sigmoid activation function with targets of either 1 or 0. For training we applied the total gradient descent training procedure with the zero initial weight vector. In order to obtain the maximal margin quickly, we gradually increased the learning step η according to the rule $\eta = 0.001 \times 1.03^t$, where t is the current iteration number. On the right of Figure 5.3, we have the generalisation error $\hat{\varepsilon}_N^{\mathrm{SLP}}$ versus the number of iterations and on the left we have $\hat{\varepsilon}_N^{\mathrm{SLP}}$ versus the margin.

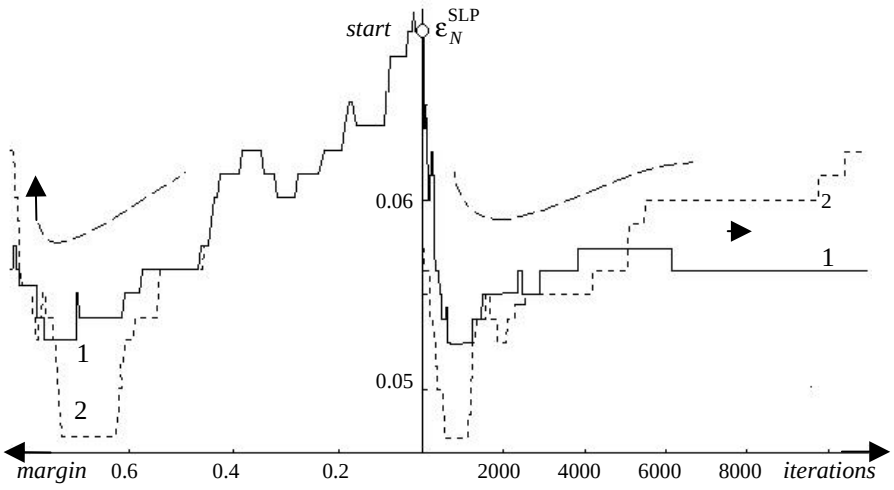

Fig. 5.3. Utilisation of the SLP to classify the vowel data: the generalisation error $\hat{\varepsilon}_N^{\mathrm{SLP}}$ versus the number of iterations and versus the margin in the original (curves 1) and in the transformed (curves 2) feature spaces (Reprinted from *Neural Networks*, 13: 9–11, Raudys, How good are support vector machines? 2000, with permission from Elsevier Science).

The graphs show that after the first iteration in the original FS, the EDC gives 7% generalisation errors. The regularised discriminant analysis (SLP after the first iteration in the transformed FS) gives 5.7% errors and the maximal margin classifier (SLP in the transformed FS after 110,000 iterations) gives 6.2% errors. The optimally stopped SLP in the original FS yields an even better result: 5.3% errors. The best result, however, was obtained by an optimally stopped SLP trained in the transformed FS, which gives a classification error of 4.7%.

This experiment supports the previous chapter's conclusions that excessive growth in the margin width increases the generalisation errors. The minimal generalisation error (4.7%) is obtained much earlier before we encounter the SV classifier (6.2%). Thus, for this classification problem, the integrated approach merging the statistical and neural net classifier design methods was useful.

While analysing the effectiveness of the integrated approach we performed several thousand training experiments with a dozen 18 - to 166-variate real-world data sets. We used various data rotation and scaling methods, two dozen covariance matrix structurisation methods and different, randomly-chosen large and small training sets. In most of the experiments, we found the integrated approach yielded excellent results. In majority of the experiments, conventional structurisation of the covariance matrices was inferior to regularised discriminant analysis. Typically, the well-trained SLP classifier outperformed the RDA classifier. I

In complicated cases with high dimensional data sets and small training sets, however, we have almost singular estimated CM with several extremely small or even zero eigenvalues. Then the SLP trains very slowly and is inferior to RDA.

Example 4. As a summarising example, we present results with 24 different data structures performed on ten real-world data sets. In Figure 5.4, we have a histogram of distributions of the relative efficacy of different classifiers (211 sequences of experiments with the 24 CM structures; each sequence of the experiments was performed 25 times with different randomly-chosen training sets). The efficacy was measured as a ratio – the generalisation error of the optimised standard RDA (a benchmark method) divided by the generalisation error of the classifier under consideration: $\gamma = \varepsilon_N^{\text{RDA}} / \varepsilon_N^{\text{classifier}}$. We see the black histogram (SLP after the data transformations) is shifted noticeably to the right.

Utilisation of statistical methods to whiten the data and subsequent intelligent training (the conditions E1 – E4) very often improves the classification performance. Exceptions constitute cases where an extremely small size of the training-set was insufficient to perform a useful data transformation. It was noticed that in cases where the training sets were extremely small, it was difficult to discover and to utilise the CM structure properly. The experiments show that typically, a gain in classification performance is obtained in situations where the statistical methods without any data transformation perform poorly.

Thus, the SLP classifier training improves when *approximately correct statistical assumptions* are utilised. In order to obtain the best results a number of different CM structurisation or CM regularisation methods for each particular data type should be examined.

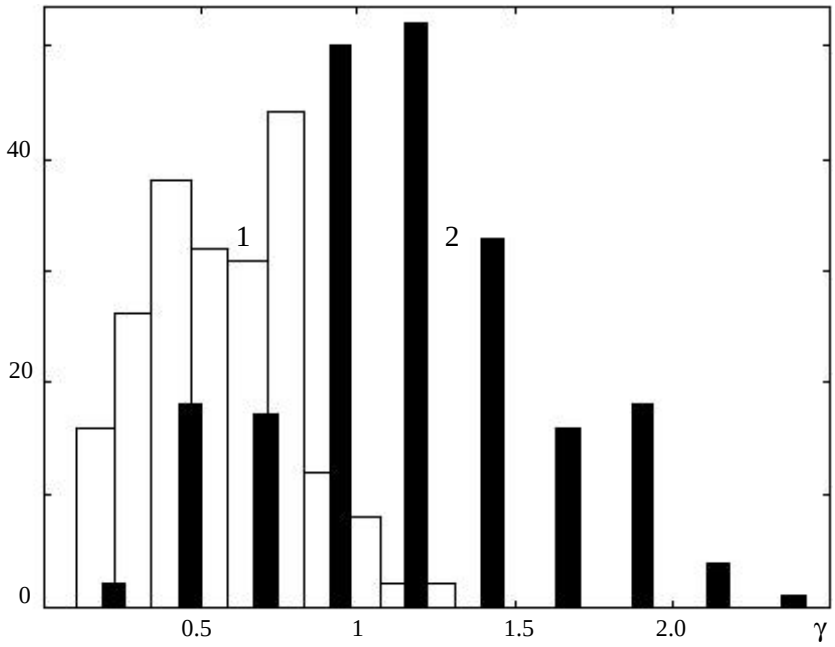

Fig. 5.4. The efficacy of structurisation in 211 series of the experiments with the stuctured CM used to design the statistical classifiers: 1 – a histogram of the gain γ for the linear discriminant analysis with differently structured sample CM, 2 – a histogram for optimally trained SLP after the linear data transformation obtained from structured sample CM (each series consists of 25 independent experiments).

5.6 The Statistical Approach in Multilayer Feedforward Networks

In SLP classifier training, the data whitening transformation plays a doubly beneficial role:

- it gives a good start and reduces the generalisation error,
- it improves the convergence rate.

No doubt the data whitening transformation is useful for the multilayer feedforward networks training. In this case, however, we cannot begin training from zero initial weights and obtain a good start: in order to become different hidden units, one has to start training from different initial weight vectors. Nevertheless, a number of ways may be suggested to utilise statistical methods. While training feedforward neural networks, one can use the statistical methods to choose good starting weights. Three options can be suggested:

1. Use weights of the statistical classifier directly to determine the starting multilayer network's weights. In MLP design one can use the weights of a statistically designed piecewise-linear classifier to determine the weights of the hidden layer neurones (Raudys and Skurichina, 1992), use prototypes and other statistical techniques (Denoeux and Langelle, 1993). To fix the initial weights of the output layer neurone one should carefully train them for a while by the standard methods. In RBF and LVQ neural networks, a standard approach is to determine the initial weights by cluster analysis techniques. The decision tree classifiers can be used, too (Kubat, 1998).

2. Use a well-trained good statistical classifier in order to edit the training-set and exclude incorrectly classified training vectors. In a modification of this approach, one can "increase" the training-set by injecting noise and then edit the new larger training-set before training the perceptron. It was noticed that the edited training-set improves the training speed and assists in reducing the possibility of being trapped into bad local minima. In this technique, the edited training-set is utilised for MLP weights initialisation.

3. Map the original design set vectors into a low-dimensional space where it is easier to avoid trapping local minima. As an extreme case, the two-dimensional mapping can be applied. Here an operator can inspect solutions (decision boundaries) and interfere in the training process interactively (A. Raudys, 2000). After training in the low-dimensional space, remaining directions (transformed features) are added sequentially in order to return to a higher-dimensional space.

A certain additional improvement can be obtained when after training the feedforward network, we train the output layer neurones using the integrated approach. Clearly, more techniques to integrating the statistical methods into complex feedforward network design can be and should be developed to improve classification performance.

5.7 Concluding and Bibliographical Remarks

In the integrated approach to classifier design, we employ all positive attributes of the neural network approach (NP1 – NP5) as well as the positive attributes SP1 and SP2 of the statistical approach. Moreover, we diminish the harmful influence of the negative attributes SN1, SN2, NN1 and NN4.

The dependence of the generalisation error of the perceptron on the initial conditions, earlier referred as a negative attribute, now becomes an extremely positive property that plays a central role in the integrated classifier approach. In the case that one starts with almost correct initial conditions, the perceptron weights will, at some point, closely approximate the weight vector where the true minimum of the generalisation error occurs. Then optimal stopping can be very effective. In statistical classification and prediction algorithm design, the designer utilises all his skill and knowledge (prior information and the training-set vectors) to design the best possible rule. In the integrated classifier design approach, the

designer should utilise this skill and knowledge in order to transform the data. Then the classifier designer obtains the best statistical rule after the first iteration.

There are five basic reasons for the success of the integrated approach:

1. Statistical methods can be utilised to select good initial weights. The good start can help obtain a smaller generalisation error.

2. The employment of prior guesses about probability density functions allows one to improve the small sample properties of statistical classification and prediction algorithms. E.g., the common covariance matrix characterised by a small number of parameters does not affect the generalisation error if both the dimensionality and the training-set size are large.

3. The use of the SLP allows one to obtain robust, minimum empirical error and maximal margin classifiers and to classify non-Gaussian patterns well.

4. For many theoretical data models the SLP can have very good small sample properties.

5. In order to obtain a full range of statistical classification and prediction algorithms, one needs to satisfy conditions E1 – E4 and apply the gradient descent training that performs better in the transformed feature space.

Many facets concerning the integration of statistical and neural network classification paradigms remain unresolved. As an example we mention the necessity of developing a wider scale of statistical models to characterise the covariance matrix by a few parameters, analysis of situations where the pattern classes have different covariance matrices, developing methods for the case when training-set sizes in each pattern class are different, designing simple and reliable CM estimation algorithms, fast, reliable and simple non-linear data transformation methods, finding ways to utilise the integrated approach in the many pattern-classes case, designing MLP and RBF classifiers, and obtaining better model choice algorithms.

The first hints that one should decorrelate the inputs were from visual cortex investigators in biological systems. In an analysis of retinal ganglion cells, Atick and Redlich (1990) proposed that the purpose of retinal processing is to improve the efficiency of visual representation by recording the input into a spatially-decorrelated form. A bit later, the decorrelation dynamics of lateral interaction between orientation selective cells in the early visual cortex was shown to yield quantitative predictions about orientation contrast, adaptation, and the development of orientation selectivity that are in agreement with experiments (Dong, 1993, 1994). Similar explanation was suggested in Buchsbaum and Gottshalk (1983) in analysis of the colour perception.

Le Cun (1987) and Le Cun, Kanter, and Solla (1991) have shown that convergence properties of the perceptron training process depend on the ratio of the largest and smallest eigenvalues of the training data covariance matrix. In

order to improve convergence properties of the gradient descent algorithm, researchers have suggested that one transform the input data by removing the mean and decreasing correlations across input variables (Halkaaer and Winter, 1996).

There are a number of contributed papers, survey papers and books that review and analyse artificial neural networks from the statistical perspective. Many of them have been mentioned in previous chapters. Three additional sources of interest are survey papers by Cheng and Titterington (1994), Ripley (1994) and Jain, Duin and Mao (2000).

The first two keys to successfully consolidating the statistical and neural net approaches and using both techniques simultaneously are understanding the effect of initial values (Raudys and Amari, 1998) and understanding the dynamic theory of progressive development of the SLP during its training (Raudys, 1998*b*). First, the integrated approach of utilising the positive qualities of statistical pattern recognition in the adaptive neural classifier design was applied to save the useful information obtained in the scaled rotation regularisation (Section 2.5.3). Then the integrated approach was used to improve the efficacy of the stucturisation of the covariance matrices. The first publications on the integrated approach appeared in 1998 (Raudys 1998*ab*). Large-scale experimental investigations with dozens of artificial and real world data sets were published in Saudargiene (1999), Raudys (2000*c*), and Raudys and Saudargiene (1998, 2001).

6. Model Selection

In the previous chapters, it was shown that the number of algorithms for designing the linear and non-linear classification rules is very large and there exist a great variety of strategies for developing the classification algorithm from the design set. There are a number of parametric statistical algorithms suggested for the standard, Gaussian or exponential families of multivariate distribution densities. A great variety of algorithms arise from differing assumptions about the covariance matrices and their estimation methods. Apart from the mixture models, there exist nonparametric density estimation methods that lead to different modifications of the Parzen window and *k-NN* classification rules. Piecewise-linear and MLP-based models increase the number of potentially applicable algorithms further. The fact that many algorithms depend on continuous parameters such as smoothing, regularisation constants and the number of iterations, increases the field of potential possibilities. The abundance of algorithms can be increased many times by feature selection, feature extraction and ANN architecture selection techniques. The integrated approach to designing the classification rule also increases the number of possible alternatives.

Therefore, a natural problem arises: the selection of the algorithm that best fits a given pattern-recognition problem. One of the most popular criteria is to select an algorithm similar to that which was utilised successfully in previous analogical applications. In fact, it is the unformalised, rough Bayes approach. For this purpose a number of empirical comparisons have been performed (see e.g. Raudys *et al*., 1975; Michie *et al*., 1994; Duin, 1996; Cherkassky and Mulier, 1998; Sohn, 1999; Lim *et al*., 2000 and Ho and Basu, 2000). Another approach is to estimate performances of each competing model precisely and select the best algorithm.

In many applications, one of the principal characteristics in designing the pattern-recognition system is the classification error. Therefore, the model selection should be based on this parameter. Unfortunately, in real situations, we often have to compare a tremendous number of competing models and the error estimation becomes too expensive a task. As a result, a great number of approximate and easy to calculate performance measures have appeared in the literature. Utilisation of approximate measures, however, leads to error in the model selection. The validation set begins to serve as an additional training-set. As a result, a model selection bias arises.

A few remarks concerning the terminology should be restated. The validation set is a part of the design set and is used to evaluate the performance of the

competing variants of the pattern-recognition system. In fact, the validation set is also used for training, however, in a different way. So, an important problem is to split the design set into the training and validation sets. The answer depends on the data, the classifier used, on the number of variants compared and on the performance estimation method. Unfortunately, the problem of optimal splitting has not yet been solved. The test set is used to evaluate the performance of the final variant and should be used only once. The performance evaluation techniques applied to the validation set and to the test set are the same.

6.1 Classification Errors and their Estimation Methods

6.1.1 Types of Classification Error

In the previous chapters, we have met the Bayes, asymptotic, conditional and expected classification errors.

The *optimal* or *Bayes PMC*, denoted by ε_B, is the probability of misclassification when one has a complete knowledge of the prior probabilities $P_1, P_2, \ldots, P_L$ of L classes, the class-conditional density functions $p_i(X)$. The optimal or Bayes PMC is given in (3.1) by

$$\varepsilon_B = \int_{\Omega} min\{P_1 p_1(\boldsymbol{X}), \ldots, P_L p_L(\boldsymbol{X})\} d\boldsymbol{X}.$$

The *conditional PMC* (the *generalisation error*), denoted by $\varepsilon_N^{\mathsf{A}}$, is the probability of misclassification that is obtained when one uses a fixed training-set and a fixed classifier design method, say A. The value of the conditional PMC depends directly upon the classifier design method A, the training-set size, the statistical characteristics of the data and the particular training-set.

The *expected PMC* (the *mean generalisation error*), denoted by $\bar{\varepsilon}_N^{\mathsf{A}}$, is the expectation of the conditional PMC, $\varepsilon_N^{\mathsf{A}}$, over all theoretically possible training sets of the given size ($N_1, N_2, \ldots, N_L$ vectors from each pattern class), $\bar{\varepsilon}_N^{\mathsf{A}} = E\,\varepsilon_N^{\mathsf{A}}$.

The *asymptotic PMC*, denoted by $\varepsilon_\infty^{\mathsf{A}}$, is a limit, i.e. the probability of misclassification of the classifier design method A when one has a complete knowledge of the class conditional density functions $p_i(X)$. The asymptotic PMC may be expressed as

$$\varepsilon_\infty^{\mathsf{A}} = \lim_{N_1 \to \infty \ \ldots \ N_L \to \infty} \bar{\varepsilon}_N^{\mathsf{A}}.$$

These four probabilities of misclassification have different interpretations and different applications. For instance, if we wish to evaluate the classification

performance of a particular pattern-recognition system that we have just designed, then the conditional PMC, ε_N^A, (the conditional generalisation error) is an appropriate measure of classification accuracy. If we wish to evaluate a dissimilarity of several pattern classes, then the Bayes, PMC, ε_B, is a suitable performance measure. One may be interested in evaluating the classification performance of some pattern-recognition system to be designed in the future by using some unknown training-set composed of $N_1, N_2, \ldots, N_L$, vectors from each class. Then, the expected error rate (the expected generalisation error), $\bar{\varepsilon}_N^A$, can be fitted best to evaluate the algorithm's performance and select the best model.

In the previous chapters, the training-set error (empirical or apparent error) has also been discussed. It is the frequency of classification errors obtained while classifying the training-set vectors. Sometimes this estimate is used to solve the model selection problem and it will be discussed in subsequent sections. While solving the model selection task, one uses sample-based approximate performance estimates. Therefore, three new error estimates arise: an apparent error in the selection, an ideal error and a true error in the model selection (Section 6.5). These errors should not be confused with other classification error types.

6.1.2 Taxonomy of Error Rate Estimation Methods

A large number of the error rate estimation methods proposed in the pattern recognition literature can be categorised into two types:

- parametric based estimators and
- nonparametric based estimators.

Furthermore, each of these two groups can be partitioned according to following two properties:

- the way in which the multivariate observations are used to train and to test (validate) the classifiers.
- the type of pattern error function which determines the contribution of each observation of the test (validation) set to the estimate of the appropriate PMC.

We consider several widely utilised error rate estimation methods in each of the two major groups: parametric based and nonparametric based error rate estimators.

6.1.2.1 Methods for Splitting the Design Set into Training and Validation Sets

1. The hold-out method (HM) is the simplest and, possibly, the most often used approach to estimating the classification error rate. In this method, the design set observation vectors are split into two separate sets: the training set (*TS*) and the validation set (*VS*).

If the validation set is used only once, then it can be called the test set. If the same validation set is used many times, a separate independent test set should be provided.

The classification rule is trained by using the observations belonging to the training-set. The performance of the classification rule is estimated as a proportion of misclassified observations from *VS*. For the two category case, it is

$$\hat{\varepsilon}_{HM} = \frac{K_{HO}}{N_{v1} + N_{v2}}, \tag{6.1}$$

where: N_{vi} is a number of the test set's (or the validation set's) vectors from the *i*-th pattern class;

$$K_{HO} = \sum_{j=1}^{N_{v1}} K_1(g(\boldsymbol{X}_{1j})) + \sum_{j=1}^{N_{v2}} K_2(g(\boldsymbol{X}_{2j})) \tag{6.2}$$

$K_i(g)$ is an indicator (a pattern error) function.

In an *error counting estimate*, function $K_i(g)$ expresses the classification errors:

$K_1(g(\boldsymbol{X}_{1j})) = 1$ if $g(\boldsymbol{X}_{1j}) < 0$, and $\boldsymbol{X}_{1j}$ belongs to ω_1; $K_1(g(\boldsymbol{X}_{1j})) = 0$ otherwise,

$K_2(g(\boldsymbol{X}_{2j})) = 1$ if $g(\boldsymbol{X}_{2j}) > 0$, and $\boldsymbol{X}_{2j}$ belongs to ω_2; $K_0(g(\boldsymbol{X}_{2j})) = 0$ otherwise;

$g(\boldsymbol{X})$ is the discriminant function score. In an error counting estimate, the sum K_{HO} expresses a number of misclassifications. In Section 6.1.2.3, we will introduce more pattern error functions.

In the hold-out method, we split the training observations of the design set into two mutually exclusive sets. A shortcoming of this approach is that neither the training-set nor the validation set can utilise all information contained in the design set. Suppose we have a large design set, which is sufficient for training the classifier and for obtaining an accurate estimate of its error. Then the major problems in error rate estimation are the amount of available computer memory and the time required to perform the necessary computations. However, in the small design set case, many problems can occur. One problem is to find an optimal partition of the design set into the training-set and the validation set. This method is sometimes called a cross-validation or one-fold cross-validation.

2. The resubstitution method uses all design set observations as a training-set and then reuses the complete design set as a test (or the validation set). This is the *empirical error*, $\hat{\varepsilon}_{RM}$. Some people call it an *apparent error*. For the two category case, it is

$$\hat{\varepsilon}_{RM} = \frac{K_R}{N_{t1} + N_{t2}}, \tag{6.3}$$

where N_{ti} is a number of vectors from the i –th class in the design (training) set, and

$K_R = \sum_{j=1}^{N_{t1}} K_1(g(\boldsymbol{X}_{1j})) + \sum_{j=1}^{N_{t2}} K_2(g(\boldsymbol{X}_{2j}))$ is a sum of the pattern error functions calculated for all design (training) set vectors.

3. The leaving-one-out method (LOOM) also utilizes all the design set vectors for training. In this method, we split the design set, consisting of $N_d = N_{d2} + N_{d2}$ observation vectors, into the training-set, consisting of N_d - 1 observations, and a test (validation) set consisting of a single observation. The procedure of training the classifiers and testing their performance on the single test vector is repeated N_d times with different observation vectors behaving as the test set at each new iteration. The LOO estimate is a proportion of misclassified observations over all N_d splits

$$\hat{\varepsilon}_{LOOM} = \frac{K_{LOO}}{N_{d1} + N_{d2}}, \tag{6.4}$$

where K_{LOO} is a sum of the pattern error functions.

There are many other modifications of the LOO method where more than one observation vector comprises the test sample. For instance, a Π (rotation) method is a compromise between the LOO and the hold-out estimation methods. This method consists of partitioning the total design set into k separate sets of size N_k. The training of the classifier is performed on a set consisting of $(k\text{-}1)N_k$ vectors and tested on the remaining N_k vectors. This procedure is repeated k times and the resulting error rate is an average of the hold-out error rates for the k separate sets. Frequently this method is called the *k-fold cross-validation*.

4. The bootstrap method (BM) uses the bootstrap sampling concept popularised by Efron (1982). In this method, the original design set, *DS*, is treated as the population of interest and a bootstrap training set, *BS*, is formed by sampling with replacement from the initial design (training) set. A brief description of the bootstrap error-rate estimation method is as follows.

A bootstrap training set, *BS*, is selected randomly (with replacement) from the original training observations, *DS*, and the classification rule is trained using the bootstrap set. Then, the resubstitution classification error estimate $\hat{\varepsilon}_{RM}^{(B)}$ is found using the bootstrap set, *BS*. Afterwards, an estimate $\hat{\varepsilon}_{N}^{(B)}$ of the conditional error rate is found by calculating the estimated error rate using the vectors of the data set *DS* which were not selected into *BS*. These bootstrap calculations are repeated r times in order to evaluate the bias of the resubstitution estimator $\hat{\varepsilon}_{RM}$. The bias is evaluated as

$$Bias(\hat{\varepsilon}_{RM}) = \overline{\hat{\varepsilon}_{N}^{(B)} - \hat{\varepsilon}_{RM}^{(B)}},$$

where the upper bar means that the average is calculated from r randomly chosen boostrap sets.

Then the bootstrap estimator of the error rate is

$$\hat{\varepsilon}_{BM} = \hat{\varepsilon}_{RM}^{(B)} + Bias(\hat{\varepsilon}_{RM}). \tag{6.5}$$

The bootstrap method results in the almost unbiased estimates of the conditional PMC. The estimated bias term $Bias(\hat{\varepsilon}_{RM})$ is a double random variable. It is influenced by the random character of the design set vectors, *DS*. In addition, it is affected by the r randomly-formed bootstrap sets. With an increase in r, the effect of the second factor decreases. How large must r be? As a rule of thumb, we recommend $r \geq 20$. This suggestion is derived from a requirement: a variance initiated by the limited number of r constitutes 5% of a total variance of the bootstrap estimate. It means an increase in r can reduce the variance by only 5%.

6.1.2.2 Practical Aspects of Using of the Leave-One-Out Method

The LOO estimator is computationally intensive. Therefore, for the parametric linear and quadratic classifiers, special recursive formulae have been derived to diminish the computational effort. In this approach, we need to invert the covariance matrices only once. The amount of calculation can be reduced by employing the Bartlett equality, often used in multivariate statistical and neural net analysis:

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(1 + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}, \tag{6.6}$$

where $\mathbf{A}$ is the $n \times n$ positive defined matrix and $\mathbf{U}$ and $\mathbf{V}$ are n-variate vectors.

In the L pattern class case, let $D_S(\boldsymbol{X}_j^{(i)}) = (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)^T \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)$ be the standard linear DF corresponding to the s-th pattern class (here we perform the classification according to a minimum of $\{D_1(\boldsymbol{X}), D_2(\boldsymbol{X}), \ldots, D_L(\boldsymbol{X})\}$). Then the linear DF trained by *DS*, with the exception of the test observation, $\boldsymbol{X}_j^{(i)}$, is

$$g_S(\boldsymbol{X}_j^{(i)}) = -k_{1i}\{D_S(\boldsymbol{X}_j^{(i)}) + \frac{[D_s(\boldsymbol{X}_j^{(i)})]^2}{k_{2i} - D_s(\boldsymbol{X}_j^{(i)})}\}, \tag{6.7}$$

where $k_{1i} = \begin{cases} 1 & \text{if } i \neq s, \\ (\frac{N_i}{N_i - 1})^2 & \text{if } i = s, \end{cases}$, $k_{2i} = (N_i - 1)(\sum_{j=1}^{L} N_j - L)/N_i$, and

N_i is a number of vectors from the i-th class in the design (training) set.

For the quadratic discriminant we have $D_S(\boldsymbol{X}_j^{(i)}) = (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)^T \hat{\boldsymbol{\Sigma}}_s^{-1} (\boldsymbol{X}_j^{(i)} - \hat{\boldsymbol{M}}_i)$,

$$g_s(\boldsymbol{X}_j^{(i)}) = -k_{3i}\{D_s(\boldsymbol{X}_j^{(i)}) + \frac{[D_S(\boldsymbol{X}_j^{(s)})]^2}{k_{4S} - D_S(\boldsymbol{X}_j^{(s)})}\} - \ln|\hat{\boldsymbol{\Sigma}}_i| - \ln[1 - \frac{D_S(\boldsymbol{X}_j^{(i)})}{k_{4i}}] - k_{5i}, \text{ if } i = s,$$

$$g_s(\boldsymbol{X}_j^{(i)}) = -D_s(\boldsymbol{X}_j^{(i)}) - \ln|\hat{\boldsymbol{\Sigma}}_s|, \qquad \text{if } i \neq s, \tag{6.8}$$

where $k_{3i} = N_i^2 (N_i-2)/(N_i-1)^3$, $k_{4i} = (N_i-1)^2 / N_i$, $k_{5i} = n \ln((N_i-1)/(N_i-2))$.

For the nonparametric *k*-nearest neighbour and the Parzen window classifiers, the leaving-one-out method does not require additional computational effort. In the maximum margin (support vector) classifier (for details see Chapter 2), the linear discriminant boundary is defined as the maximal distance from the training-set vectors. The vectors on the boundary are called the supporting patterns. They determine the position of the discriminant boundary and can be straightforwardly utilised to speed up the leaving-one-out estimate of the classification error.

6.1.2.3 Pattern Error Functions

1. Error counting. The simplest way to estimate the classification error is to find the frequency of incorrectly classified observations (Equation (6.2)) and use it as the error estimate $\hat{\varepsilon}_{EC}$. In this method, only the number of misclassifications is counted. All misclassified pattern vectors have the same contribution to the classification error estimate. A contribution of the vector $\boldsymbol{X}_j$ to the error estimate does not depend on the distance of the vector $\boldsymbol{X}_j$ from the decision boundary. This method is called error counting and can be used with all four design set splitting methods discussed in Section 6.1.2.1.

2. The smoothed error rate estimator is similar to the error counting error rate estimator. In the sum (6.2), instead of a threshold function, one uses a more general pattern error function $K(g(\boldsymbol{X}_j))$. A small perturbation in the value of the discriminant $g(\boldsymbol{X}_j)$ corresponds to a small change in contribution to the error estimate. In general, it is required that (we consider a case, where $\boldsymbol{X}_j$ belongs to ω_1).

$$K(g(\boldsymbol{X}_j)) \to 1 \text{ as } g(\boldsymbol{X}_j) \to -\infty \text{ and } K(g(\boldsymbol{X}_j)) \to 0 \text{ as } g(\boldsymbol{X}_j) \to \infty.$$

The logistic sigmoid function $K(g(\boldsymbol{X}_j)) = 1 - 1/(1+e^{-g(\boldsymbol{X}_j)})$ can serve as an example of the kernel (curve 1 in Figure 6.1). Another choice of the kernel function is the Gaussian cumulative function with variance λ^2. The kernel function can be piecewise-linear, like line 2 in Figure 6.1. Change of λ or the slope of the piecewise-linear function controls a contribution of each correctly and incorrectly classified pattern vector to the estimate of the classification error rate. With the increase in λ or in the slope, more pattern vectors contribute to sum (6.2). An extreme case arises when we use the threshold function as the kernel, i.e.

$K(g(\boldsymbol{X}_j)) = 0$ if $g(\boldsymbol{X}_j) > 0$, and $K(g(\boldsymbol{X}_j)) = 1$ otherwise (piecewise-linear line3 in Figure 6.1). Then we have the error counting estimate.

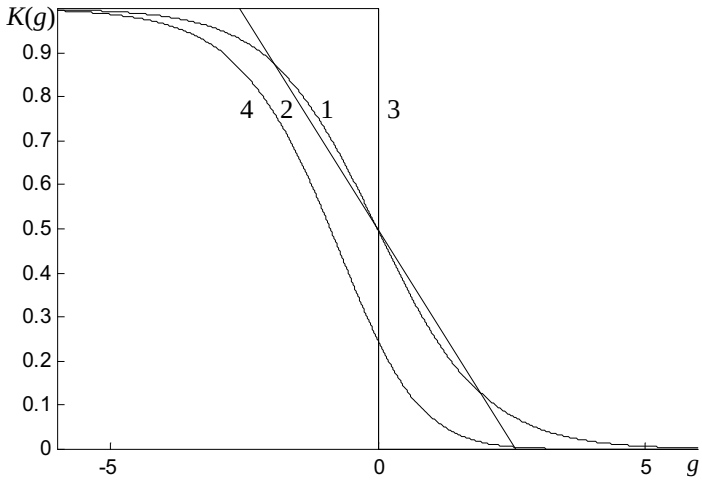

Fig. 6.1. The pattern error functions: 1 – logistic function $K(g)=1-1/(1+ e^{-g})$, 2 – piecewise-linear function, 3 – threshold function, 4 – pattern error function of the sum of squares (4.2).

The cost function of the squares (1.7) can be utilised to estimate the classification error. For the targets $t_1 = 1$ and sigmoid activation function, $K(g) = (1+ e^{g})^{-2}$. In Figure 6.1 (curve 4), we depict this function. In Chapter 4, we have seen that the cost function begins to calculate the classification error as the weights of the single layer perceptron increase. Hence, in estimation the classification error, the cost function of the sum of squares with large weights can be sufficiently exact.

While estimating the classification error rates, the smoothed error rate estimators can invoke a bias. However, a variance of the estimator $\hat{\varepsilon}_{SM}$ is smaller than that of the error counting estimate $\hat{\varepsilon}_{EC}$. Both the bias and the variance, depend on smoothing. The bias increases with λ, whereas the variance decreases with λ. One problem with formulating the estimator $\hat{\varepsilon}_{SM}$ is the choice of the best smoothing parameter λ.

3. Parametric estimators for Gaussian patterns. In Section 3.4.1, for the two pattern class case, we had the following formula for the asymptotic probability of misclassification of the standard Fisher linear discriminant function:

$$\varepsilon_{\infty}^{F} = \Phi\{-\delta/2\}, \tag{6.9}$$

where δ^2 is the squared Mahalanobis distance.

This formula gives the exact values of the asymptotic error for the GCCM data model when the mean vectors $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and the common matrix $\overline{\Sigma}$ are known. When these parameters are unknown, they can be replaced by their sample estimates $\hat{\boldsymbol{M}}_1$, $\hat{\boldsymbol{M}}_2$ and $\hat{\boldsymbol{\Sigma}}$. Then the sample squared Mahalanobis distance is

$$\hat{\delta}^2 = (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2), \tag{6.10}$$

and the following simple parametric estimate (called *D* estimate) of the classification error has been suggested in the literature

$$\hat{\varepsilon}_D = \Phi\{-\hat{\delta}/2\}. \tag{6.11}$$

The estimate (6.11) depends on two random vectors, $\hat{\boldsymbol{M}}_1$ and $\hat{\boldsymbol{M}}_2$ and random matrix $\hat{\boldsymbol{\Sigma}}$. Therefore it is a random variable too. By means of the standard orthogonal transformations traditionally used in multivariate statistical analysis, it can be shown that $\hat{\delta}^2$ is distributed as a scaled ratio of the non-central and central chi-square random variables (the scaled non-central *F* random variable):

$$\hat{\delta}^2 \sim \frac{(N_1+N_2)(N_1+N_2-2)}{N_1 N_2} \frac{\chi^2_{\frac{N_1N_2}{N_1+N_2}\delta^2, n}}{\chi^2_{N_1+N_2-n-1}}. \tag{6.12}$$

Its expectation is:

$$E\,\hat{\delta}^2 = \frac{(N_1+N_2)(N_1+N_2-2)(\frac{N_1N_2}{N_1+N_2}\delta^2+n)}{N_1N_2(N_1+N_2-n-3)} > \delta^2. \tag{6.13}$$

We see that $\hat{\delta}^2$ is a biased estimate of the Mahalanobis distance δ^2. The use of (6.10) in estimator (6.9) results in an optimistically biased estimate of the Bayes classification error. An unbiased estimator of the distance δ^2 is given by

$$\hat{\delta}^2_{DS} = \frac{(N_1+N_2-n-3)}{N_1+N_2-2}\hat{\delta}^2 - \frac{N_1+N_2}{N_1N_2}n, \tag{6.14}$$

An expected value $E\,\hat{\delta}^2_{\mathrm{DS}} = \delta^2$. The estimator (6.14) can be used in (6.9) in order to reduce the bias of the asymptotic classification error estimate:

$$\hat{\varepsilon}_{DS} = \Phi\{-\delta^2_{\mathrm{DS}}/2\}. \tag{6.15}$$

In spite of the fact that (6.14) is unbiased, the estimator *DS* (6.15) is a slightly biased estimator of the asymptotic error $\hat{\varepsilon}_\infty^{(\mathrm{F})}$. This bias is caused by the non-linear characteristic of the Laplace function $\Phi(c)$.

The unbiased estimate (6.14) also is a random variable. Therefore, in the small training sample size N case, the estimator (6.14) may sometimes be negative. In order to avoid negative values, it is proposed that one skip the last negative term in (6.14). The parametric error estimates $\hat{\varepsilon}_D$ and $\hat{\varepsilon}_{DS}$ are appropriate for evaluating the classification performance if the data can be described by the GCCM data model. The parametric estimates, however, do not work well if the pattern classes differ from this Gaussian model.

4. A pseudo-parametric error rate estimator has been suggested as a compromise between the parametric and non-parametric approaches. It combines the LOO method with an assumption that the discriminant function scores $g(\boldsymbol{X}_j^{(i)})$ (i = 1, 2; j =1, 2, ..., N_i) are conditionally Gaussian distributed. This assumption is based on a central limit theorem: each value of the discriminant $g(\boldsymbol{X}_j^{(i)})$ is a sum of a large number of weakly dependent random components. Consequently, it is approximately Gaussian distributed. This assumption is especially appropriate when the discriminant function is linear, i.e. $g(\boldsymbol{X}_j^{(i)}) = v_0 + \sum_{s=1}^{n} v_s x_{js}$, the number of features and the sizes of the training sets are large. Letting $\hat{E}[g_1(\boldsymbol{X})]$ and $\hat{E}[g_2(\boldsymbol{X})]$ represent the group means of the discriminant scores corresponding to both pattern classes and letting $\hat{V}[g_1(\boldsymbol{X})]$ and $\hat{V}[g_2(\boldsymbol{X})]$ be the sample variances of the respective discriminant scores, the pseudo-parametric error rate estimator can be written as

$$\hat{\varepsilon}_{PP} = \Phi\{-\hat{\delta}_{PP}/2\}. \tag{6.16}$$

where $\hat{\delta}_{PP}^2 = \dfrac{[\hat{E}[g_1(\boldsymbol{X})] - \hat{E}[g_2(\boldsymbol{X})]]^2}{(N_1\hat{V}[g_1(\boldsymbol{X})] + N_2\hat{V}[g_2(\boldsymbol{X})])/(N_1+N_2)}$.

The error counting and smoothed pattern error functions; as well as the pseudo-parametric error rate estimators, can be used with any of the four methods of splitting the design set into the training-set and test (validation) set (the hold-out, resubstitution, leaving-one-out and the bootstrap sample splitting methods). In principle, one can acquire a great number of different classification error estimation methods.

6.2 Simplified Performance Measures

A large number of competing pattern classification models arise in the feature definition phase. Here the designer needs to select a small number of most informative features from a pool of candidates. In other cases, the designer

performs a linear or non-linear transformation of the input measurement vectors in order to extract a small number of informative features. To extract the new features, one introduces a certain performance measure and optimises it. A characteristic requirement for feature extraction measures is that these criteria should be differentiable in order to apply simple optimisation algorithms.

Feature selection and feature extraction are computationally intensive procedures. In real situations, we have to compare an immense number of competing models and the error estimations become computationally very expensive. Similar problems arise in ANN architecture selection. Therefore, simple and fast feature evaluation criteria are necessary. As a result, numerous approximate and easy to calculate performance measures have appeared in the literature. Almost every performance measure from that listed below can be used to solve all tasks: feature selection, feature extraction, and ANN architecture adjustment.

6.2.1 Performance Criteria for Feature Extraction

6.2.1.1 Unsupervised Feature Extraction

A standard and most popular linear mapping technique, used in both multivariate data analysis and pattern recognition, is the *principal component* method often known as a discrete Karhunen-Loev expansion. The feature mapping criterion to be minimised is

$$Cost_{PC} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{X}_i - \boldsymbol{X}^*_i)^T (\boldsymbol{X}_i - \boldsymbol{X}^*_i) = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{X}_i - \mathbf{B}^T\boldsymbol{Y}_i)^T (\boldsymbol{X}_i - \mathbf{B}^T\boldsymbol{Y}_i), \qquad (6.17)$$

where

$\boldsymbol{X}_i$, (i =1, 2, ..., N) are the n-variate vectors of the design (training) set,

$\boldsymbol{Y}_i = \mathbf{B}\boldsymbol{X}_i$, ($i$ =1,2, ..., N) are new vectors in the r-variate new feature space Ω_r,

$\boldsymbol{X}^*_i = \mathbf{B}^T\boldsymbol{Y}_i$, is an inverse transformation of vector $\boldsymbol{Y}_i$ into the n-variate space,

$$\mathbf{B} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \\ \cdots \\ \mathbf{C}_r \end{bmatrix}$$ is an $r \times n$ transformation matrix,

$\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_r$ are $1\times n$ orthonormal vectors, eigenvectors of the sample covariance matrix $\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{X}_i - \hat{\boldsymbol{M}})(\boldsymbol{X}_i - \hat{\boldsymbol{M}})^T$, $\hat{\boldsymbol{M}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{X}_i$.

The distance $(\boldsymbol{X}_i - \mathbf{B}^T\boldsymbol{Y}_i)^T(\boldsymbol{X}_i - \mathbf{B}^T\boldsymbol{Y}_i)$ characterises the mapping error for one observation vector. The principal component method does not use any information

about the class membership of the vectors $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$. Therefore, in spite of a large number of advantageous features that are characteristic of the principal component method (maximal accuracy of the representation of the pattern vectors and the covariance matrix, maximal entropy function if the pattern vectors are Gaussian; see, e.g., Young and Calvert, 1974, Chapter 6), the application of this method to pattern classification tasks at times destroys the separability between the pattern classes.

An *inter-distances method* uses a criterion

$$Cost_{ID} = \frac{1}{N(N-1)} \sum_{j=i+1}^{N} \sum_{i=1}^{N} (H_{ij}^{o} - H_{ij}^{tr})^2, \tag{6.18}$$

where $H_{ij}^{o} = (\boldsymbol{X}_i - \boldsymbol{X}_j)^T (\boldsymbol{X}_i - \boldsymbol{X}_j)$ is a distance between the pattern vectors $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ in the original n-dimensional space Ω_n and $H_{ij}^{tr} = (\boldsymbol{Y}_i - \boldsymbol{Y}_j)^T (\boldsymbol{Y}_i - \boldsymbol{Y}_j)$ is a distance between the pattern vectors $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$ in the transformed r-dimensional space Ω_r.

An unknown $r \times n$ data transformation matrix $\mathbf{B}$ is found by the iterative search technique. At instant t, the matrix $\mathbf{B}_t$ is updated according to the following formula

$$\mathbf{B}_{t+1} = \mathbf{B}_t \mathbf{T}_{\mathbf{B}_t} \mathbf{R}^{-1},$$

where $$\mathbf{R} = \frac{1}{N(N-1)} \sum_{j=i+1}^{N} \sum_{i=1}^{N} (\boldsymbol{X}_i - \boldsymbol{X}_j)(\boldsymbol{X}_i - \boldsymbol{X}_j)^T$$ and

$$\mathbf{T}_{\mathbf{B}_t} = \frac{1}{N(N-1)} \sum_{j=i+1}^{N} \sum_{i=1}^{N} \frac{(\boldsymbol{X}_i - \boldsymbol{X}_j)^T (\boldsymbol{X}_i - \boldsymbol{X}_j)}{(\boldsymbol{X}_i - \boldsymbol{X}_j)^T \mathbf{B}_t^T \mathbf{B}_t (\boldsymbol{X}_i - \boldsymbol{X}_j)} (\boldsymbol{X}_i - \boldsymbol{X}_j)(\boldsymbol{X}_i - \boldsymbol{X}_j)^T.$$

A start matrix $\mathbf{B}_0$ is obtained by the principal component method: $\mathbf{B}_0 = \mathbf{B}_{PC}$. The principal component method preserves a *global structure* of the data. The inter-distances method tries to preserve *local structures* of the data.

There are several methods for a *non-linear mapping* into the low-dimensional feature space. In the non-linear mapping, one searches for $r \times N$ coordinates of r-variate vectors $\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_N$ directly. Here we cannot use such a criterion function as in Equation (6.18). One has to introduce a normalisation of the distances. One of the first non-linear mapping algorithms proposed in the literature (Sammon, 1969; Niemann, 1980) were procedures based on the following criteria

$$Cost_S = \frac{1}{\sum_{i<j} H_{ij}^{O}} \sum_{j=i+1}^{N} \sum_{i=1}^{N} \frac{(H_{ij}^{O} - H_{ij}^{tr})^2}{H_{ij}^{o}}, \text{ and} \tag{6.19}$$

$$Cost_N = \frac{1}{\sum_{i<j} (H_{ij}^O)^2} \sum_{j=i+1}^{N} \sum_{i=1}^{N} (H_{ij}^O - H_{ij}^{tr})^2 \ . \tag{6.20}$$

Minimisation of the above criteria may be interpreted as the non-linear transformation of vectors from the *n*-dimensional space Ω_n into the *r*-dimensional space Ω_r such that all inter-distances between the vectors are preserved as much as possible. Both cost functions are multi-extremal. The minimisation is performed by means of iterative procedures.

The non-linear mapping procedures preserve the local structures of a complex pattern space, however, they do not provide an explicit algorithm to process the forthcoming pattern vectors. One of the possibilities for obtaining the explicit mapping algorithm is to approximate the function $\boldsymbol{Y} = f(\boldsymbol{X})$ by MLP with *n* inputs and *r* outputs. In MLP training, we utilise the *n*-dimensional vectors $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N$ as inputs and the *n*-dimensional vectors $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_N$ obtained by the non-linear mapping, as desired outputs of the network. We already utilised this technique in Section 5.5.2.

6.2.1.2 Supervised Feature Extraction

In *pattern classification*, however, one is interested in a transformation such that the *class separability* is maximally preserved. One of such criteria used for the two-class problem is Fisher's criterion

$$Cost_F\ (\boldsymbol{V}_1) = \frac{\boldsymbol{V}_1^T \hat{\boldsymbol{\Sigma}}_B \boldsymbol{V}_1}{\boldsymbol{V}_1^T \hat{\bar{\boldsymbol{\Sigma}}}\ \boldsymbol{V}_1}, \tag{6.21}$$

where $\boldsymbol{V}_1$ is the *n*-variate transformation $y_1 = \boldsymbol{V}_1^T \boldsymbol{X}$ vector to be determined during the minimisation process, $\hat{\boldsymbol{\Sigma}}_B$ is the "between the classes covariance matrix": $\hat{\bar{\boldsymbol{\Sigma}}}_B = (\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)(\hat{\boldsymbol{M}}_1 - \hat{\boldsymbol{M}}_2)^T$ and $\hat{\bar{\boldsymbol{\Sigma}}}$ is the sample covariance matrix pooled over two pattern classes $\hat{\bar{\boldsymbol{\Sigma}}} = ½ (\hat{\boldsymbol{\Sigma}}_1 + \hat{\boldsymbol{\Sigma}}_2)$.

Minimisation of (6.21) results in a new feature $y_1 = \boldsymbol{V}_1^T \boldsymbol{X}$ with the maximal discriminative power. At the second step, we may maximise $Cost_F\ (\boldsymbol{V}_2)$ under the constraint that $\boldsymbol{V}_2$ is orthogonal to the weight vector $\boldsymbol{V}_1$ determined in the first step, i.e. $\boldsymbol{V}_1^T \boldsymbol{V}_2 = 0$. The vectors $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ determine the optimal discriminant plane. This concept may be expanded to yield the "optimal discriminant set of vectors".

In a generalisation of this approach, one can use any of the discriminant function techniques to obtain the new features. The new features may be postulated to be orthogonal to the previously determined vectors; they may also be

determined on the condition of obtaining the best discrimination only from some part of the training vectors.

There are a couple of methods which minimise the classification error directly. The first of them minimises a decrease in the probability of misclassification in the classification problem of the L multivariate normal populations (Tubbs, Coberley and Young, 1982) In the linear feature reduction procedure $\boldsymbol{Y} = \mathbf{B}\boldsymbol{X}$, the $r \times n$ transformation matrix $\mathbf{B}$ is expressed as

$$\mathbf{B} = \mathbf{A}\mathbf{U}_1,$$

where $\mathbf{A}$ is an arbitrary $r \times r$ non-singular matrix, $\mathbf{U}_1$ is an $r \times n$ partition of the $n \times n$ matrix $\mathbf{U}$ that consists of the orthonormalised eigenvectors of $n \times n$ matrix $\hat{\mathbf{C}}\,\hat{\mathbf{C}}^T$ and $\hat{\mathbf{C}} = [\hat{\boldsymbol{M}}_2,..., \hat{\boldsymbol{M}}_L, \hat{\boldsymbol{\Sigma}}_2 - \mathbf{I},..., \hat{\boldsymbol{\Sigma}}_L - \mathbf{I}]$ is an $n \times ((n + 1)(L - 1))$ matrix composed of sample estimates of the means $\hat{\boldsymbol{M}}_i$ and covariance matrices $\hat{\boldsymbol{\Sigma}}_i$ under the standard linear transformation such that $\hat{\boldsymbol{M}}_1 = \mathbf{0}$ and $\hat{\boldsymbol{\Sigma}}_1 = \mathbf{I}$ (the identity matrix).

Unfortunately, this method can become useless if the data is multimodal or asymmetric. To extract the linear features for more complex non-linearly separable data, the MLP with one hidden layer can be utilised (A. Raudys and Long, 2000). In this approach, a number of hidden units h in the first hidden layer is equal to the number of new features. Thus, the number h determines a degree of "non-linear separablility of the data". The weighted sums $y_i = \boldsymbol{X}^T\boldsymbol{V}_i + v_{i0}$ (before submitting them to the activation function) are calculated in h hidden units and serve as the new features. The technique is based on the fact that with an increase in the magnitude of the weights, the cost function of the non-linear single layer perceptron (in the output layer) begins to minimise the empirical classification error (see section 4.1.2.7). A complexity of a non-linear decision boundary utilised to evaluate the separability of the pattern classes in the new feature space depends on h, the number of new features.

6.2.2 Performance Criteria for Feature Selection

The simplest performance measure is the *Mahalanobis distance*. It is a perfect separability measure for the GCCM classes. When the covariance matrices are different, this measure can result in errors in the feature selection and extraction.

Another widely used separability measure is a *divergence*. The divergence between the classes ω_i and ω_j is given by

$$Div(i,j) = \int (p_i(\boldsymbol{X}) - p_j(\boldsymbol{X})) \ln(p_i(\boldsymbol{X}) / p_j(\boldsymbol{X}))d\boldsymbol{X}.$$

For the Gaussian classes the divergence can be written as

$$Div(i, j) = \tfrac{1}{2} \operatorname{tr} [(\hat{\boldsymbol{\Sigma}}_i - \hat{\boldsymbol{\Sigma}}_j)(\hat{\boldsymbol{\Sigma}}_i^{-1} - \hat{\boldsymbol{\Sigma}}_j^{-1})] + \tfrac{1}{2} \operatorname{tr} [(\hat{\boldsymbol{\Sigma}}_i^{-1} + \hat{\boldsymbol{\Sigma}}_j^{-1})(\boldsymbol{M}_i - \boldsymbol{M}_j)(\boldsymbol{M}_i - \boldsymbol{M}_j)^T],$$

where $\boldsymbol{M}_i$ is the mean, and $\boldsymbol{\Sigma}_i$ is the covariance matrix of the i-th class.

In the L category case, an average interclass divergence is presented as

$$Div = \sum_{i=1}^{L} \sum_{j=1}^{L} Div(i, j). \tag{6.23}$$

Many quality measures are based on the *information theory*. Examples are the Shannon information and entropy, the conditional entropy, and Vajda's average conditional quadratic entropy. There are more distance measures that enable one to obtain simplified estimates of the quality of the feature subsets. Devijver's Bayesian distance, Minkovski measures of nonuniformity, Bhattacharyya bound, Chernoff bound, Kolmogorov variational distance, Devijver's and Lissack and Fu's generalisation of variational distance, Ito's approximating functions, Jeffreys–Matusita distance are examples of separability measures.

There are a number of criteria for **choosing the ANN architecture**. Weights sensitivity calculation methods are used to evaluate the usefulness of the single features and hidden nodes. One of the popular measures is an *equivocation* called the *entropic loss*, log loss or stochastic complexity – the expectation of the logarithm of *the predictive (a posteriori) probability*

$$e = - \sum_{i=1}^{L} P_i \int p_i(\boldsymbol{X}) \log P(\omega_i | \boldsymbol{X}) \, d\boldsymbol{X}. \tag{6.24}$$

This measure does not contain a threshold function and leads to comparatively simple formulae in analytical investigations.

A *saliency* is one more popular measure in the feature and weights selection (pruning) in the ANN design. The saliency characterises an importance of the weight. It is defined by the second derivatives $H_{ij} = \dfrac{\partial^2 cost(\boldsymbol{X})}{\partial v_{ij}^2}$ of the cost function. If one uses the cost function of the sum of squares and approximates matrix $((H_{ij}))$ by a diagonal matrix composed from diagonal elements of $((H_{ij}))$, the saliency of the ij-th connection weight v_{ij} is approximated around a minimum v_{ij}^*, by:

$$s_{ij} = \frac{\partial^2 cost(\boldsymbol{X})}{\partial v_{ij}^2} v_{ij}^2. \tag{6.25}$$

In the feature selection, one uses sums of the saliencies corresponding to each single feature. A simple and reliable measure to determine the partial contribution of the ij-th weight is

$$\pi_{ij} = \frac{|v_{ij}|}{\sum_k |v_{ik}|}, \tag{6.26}$$

where the summation over all weights of the i-th neurone should be performed.

6.2.3 Feature Selection Strategies

The feature selection methods are specified by:

- a procedure (strategy) used to select a subset of r features out of a list of n total features and
- a criterion used to evaluate the quality of each subset.

Each of the criteria discussed in Section 6.2.2 can be used with different feature selection algorithms. Suppose one wishes to select the subset of r features from a list of n original features. An entire number of subsets of the size r is given by

$$C_n^r = \frac{n!}{(n-r)!r!}.$$

It is clear, that the number of subsets C_n^r to be evaluated can become very large even for moderate feature selection problems. For example, to select the best subset of 10 features out of a possible 30, one requires 30,045,015 subset evaluations. In order to reduce the number of subsets evaluated, one should use *suboptimal search procedures*.

The simplest technique ranks variables according to an estimate of the performance when the features are used separately. Such ranking of individual performance measures of the single features is based on an assumption that separate features are statistically independent. Justification for this strategy lies in the fact that the feature with a low error rate when used alone should be a valuable member of the subset of the features. For correlated features, this strategy can sometimes reject very good features. Two examples are presented in Figure 6.2. Here, the combinations of two individually almost uninformative features can discriminate two pattern classes perfectly.

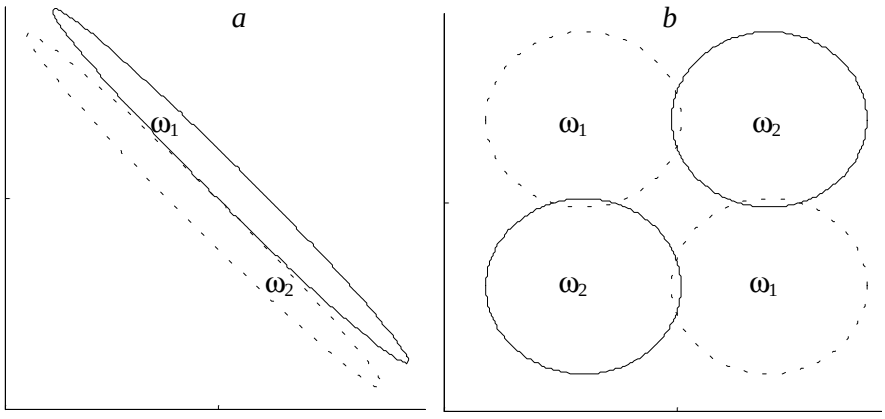

Fig. 6.2. Examples of two bi-variate distributions when a pair of individually uninformative features can discriminate two classes, ω_1 and ω_2, well.

In the *successive addition* procedure AD, the first step is to select the best single feature x_b. Then the second step is to select the best pair $x_b\ x_s$, which contains the best single feature x_b obtained in the previous step. This process continues by selecting a single feature that appears to be the best when combined with the previously selected subset of features. Often this procedure is called a *forward selection* and has much in common with successive augmentation of the ANN architecture. The number of subsets investigated in order to find the subset of r features from n possible features is given by

$$\sum_{i=1}^{n}(n - i + 1) = r(n - r - 1)/2) .$$

To select 10 features out of 30 we need to evaluate 255 subsets.

It is an annoying fact that individually the top variable or the best subset of 2, 3, 4, etc. variables sometimes can not participate in the best subset selection of higher dimensionality. Thus, in some forward selection procedures, one begins from the best subset of variables determined by a full exploration of all C_n^k possible combinations of k variables ($k << r$).

In the *successive elimination* (*backward selection*) procedure EL, at first we determine the worst single variable x_w, which provides the smallest decrease in the classification accuracy. In certain cases, the rejection of the worst features causes an increase instead of a decrease in the performance. In the second step, variable x_w is eliminated and the worst single variable x_{ws} is again omitted from the remaining n - 1 variables. We eliminate the variable x_{ws} in the second step, etc. The variables are eliminated iteratively until optimal classification performance subsets are obtained. The number of subsets investigated in order to find the subset of r features from n possible ones is much smaller than C_n^r.

A *combined procedure*, named ADEL, consists of successive applications of two previous procedures, the AD and EL procedures. Let us, at the t-th step of the algorithm, have k variables included into the feature set. We successively add one from the remaining n - k variables and estimate the quality of the obtained feature subset. We repeat this procedure n - k times. The best subset composed from k + 1 variables is selected. In the next step, we use the successive elimination procedure to delete one of the variables. After two such substeps (addition and subsequent elimination) we return to k-variate space. If we succeed in obtaining a better subset of k features, then we continue the procedure of successive addition and elimination with the new subset composed of k variables. Otherwise, we continue the procedure with the subset composed of k + 1 variables.

This procedure may operate in reverse. At first, two variables are deleted as in the backward selection EL, then one feature is added successively from the remaining ones, etc. This procedure is called ELAD. Both procedures are primary variants of a training technology, nowadays called *evolutionary computation*.

In the *random search* feature selection, a number, say m^*, of subsets is generated: each subset consists of r variables selected from n original ones in a random way. Then performances of all m^* subsets are estimated and the best one

is selected. In spite of the simplicity, the effectiveness of this procedure is similar to that of the other methods.

In a *random search with adaptation* procedure (Lbov, 1965) several random subsets, say m^*, are generated in a random way. Each subset consists of r variables. At the beginning, the probability of occurrence of all variables are equal: $q_1 = q_2 = ... = q_n$. Afterwards the performance criterion is estimated for all m^* subsets of the variables and the prior probabilities $q_1, q_2, ..., q_n$ are changed. Variables, which are often included in worthy subsets, are prompted by increasing their prior probabilities. Variables which often fall into unworthy subsets are suppressed by decreasing their prior probabilities. Several repetitions of the procedure allow one to obtain a subset near to the optimal one.

In the *synthesis* method, all C_n^k possible combinations consisting of r variables are explored (k = 2, 3, 4 or 5). Then, the best r variables that most often fall into the worthy subsets are selected.

There are many more heuristic feature selection procedures proposed in the literature. Branch and bound methods, heuristic search algorithms, dynamic programming methods, genetic algorithms and simulated annealing algorithms are suggested for this purpose. In Section 6.5 we will present arguments, that if one uses inexact performance measures in the feature selections, there is no need to utilise complex, expensive feature selection strategies.

In many pattern recognition problems, there are an excessively large number of variables. Even the procedures of the simplest variable selection, such as forward or backward selection, become troublesome. As a side note, the features are often known to be redundant. To overcome the difficulty arising in the feature selection problem with an excessively large number of initial variables, Jain and Dubes (1978) proposed a strategy for *feature definition*. The strategy involves two phases:

- clustering of the features based on a correlation measure of feature similarity;
- compressing each cluster into a single feature.

The authors *suggest an interaction* between the analyst and the data. The analyst investigates clustering for interpretability, the number of features and separation among the pattern classes.

6.3 Accuracy of Performance Estimates

6.3.1 Error Counting Estimates

6.3.1.1 The Hold-Out Method

The hold-out (cross-validation) method is the simplest and most often used error rate estimator for pattern classifiers. In the hold-out error estimation method (HM), the design set is split into the training set (TS) and the validation set (VS).

Let $N_d = N_t + N_v$ be the total number of observations from all classes in the design set, N_t be the number of observations from all classes in the training set, and N_v be the number of observations from all classes in the validation set. Let all observation vectors in the design set be statistically independent. The proportion, $\hat{\varepsilon}_{HM}$, of misclassified observations from the validation set (the test set in certain applications) is the hold-out estimate of the error rate. The estimate $\hat{\varepsilon}_{HM}$ is a *doubly random* variable. Its distribution is conditioned on the TS and VS. For fixed TS, the estimate $\hat{\varepsilon}_{HM}$ is a binomial random variable. Its mean value is equal to $\hat{\varepsilon}_{N_t}$, the conditional PMC (the generalisation error) of the classifier trained on N_t particular observations of the training-set. A variance of $\hat{\varepsilon}_{HM}$ is

$$V[\hat{\varepsilon}_{HM}] = \frac{\varepsilon_{N_t}(1-\varepsilon_{N_t})}{N_v}. \tag{6.27}$$

Here, A, the index of the classifier type, has been omitted. To search for an optimal split of the design set, let us compare $V[\hat{\varepsilon}_{HM}]$ with the variance of the conditional PMC of the linear Fisher DF (Equation (3.12) in Section 3.4.4)

$$V[\varepsilon_N^F] \approx \frac{2(\phi(\delta/2))^2}{\delta^2 N^2}\left(\frac{\delta^4}{16} + \left(1+\frac{\delta^2}{4}\right)^2 (n-1)\right), \tag{6.28}$$

Numerical analysis of (6.27) and (6.28) shows that, for moderate values of the dimensionality, training and validation sets, $V[\hat{\varepsilon}_{HM}] >> V[\varepsilon_N^F]$. One possible explanation for this phenomenon is that the hold-out error counting estimator is a nonparametric method and the linear Fisher discriminant is a parametric one. In the nonparametric hold-out error counting estimate, no prior information concerning the class-conditional densities or concerning the discriminant function $g(\boldsymbol{X})$ is used. To design the Fisher discriminant, one utilises the important additional information which proclaim that the pattern classes are Gaussian and that the covariance matrices of separate classes are equal among themselves. This information reduces the variance.

The values $V[\hat{\varepsilon}_{HM}]$ and $V[\varepsilon_N^F]$ can be utilised to find *the best split* of the design set into the training and validation sets for the standard Fisher linear classifier. Equations (6.27) and (6.28) along with the equation for the expected error (3.10) indicate that the optimal split should depend on the dimensionality n, the distance between the pattern classes δ and the design set size N_d. Note that formula (6.28) is only valid for the standard Fisher linear discriminant function and the GCCM data model. The optimal split found for this classifier can be ineffective when applied to other classifiers.

6.3.1.2 The Resubstitution Estimator

In this method, all the N_d design set observations are used to train the classifier. Hence, the training-set can be called the design set, i.e. $N_t = N_d$. After training, the same vectors are used to estimate the classification performance. Therefore, the classifier adapts to the peculiarities of the design set and results in an optimistic (biased) estimate of the error rate. For example, for the Fisher linear DF a mean value of the resubstitution error counting estimate is

$$E[\hat{\varepsilon}^{\mathrm{F}}_{RM}] \approx \Phi\{ -\frac{\delta}{2}\sqrt{T_M T_{\bar{\Sigma}}} \}, \tag{6.29}$$

where $T_M = 1+ \dfrac{2n}{\delta^2 \bar{N}}$ and $T_{\bar{\Sigma}} = 1+\dfrac{n}{2\bar{N} - n}$.

In Equation (6.29) $\bar{N} = N/2 = N_v/2 = N_t/2 = N_d/2$. The terms T_M and $T_{\bar{\Sigma}}$ arise from the approximate sample estimation of the mean vectors and the covariance matrix, respectively. The expected classification error of the Fisher classifier is

$$E[\varepsilon^{\mathrm{F}}_{N}] \approx \Phi\{ -\frac{\delta}{2}\frac{1}{\sqrt{T_M T_{\bar{\Sigma}}}} \}. \tag{3.10}$$

With an increase in N, the number of design set vectors, both expected values, $E[\hat{\varepsilon}^{\mathrm{F}}_{RM}]$ and $E[\varepsilon^{\mathrm{F}}_{N}]$, converge on each other and to the asymptotic error $\varepsilon^{\mathrm{F}}_{\infty}$ The symmetry is obvious between Equations (6.29) and (3.10).

Example 1. In Figure 6.3, we present graphs of dependence of the expected PMC and the expectation of the resubstitution estimator on N, the training-set size. The graphs are calculated for data with the Mahalanobis distance $\delta = 3.76$, $n = 10$ and $n = 30$. For a large number of design samples, the both graphs become symmetrical with respect to the asymptotic classification error $\varepsilon^{\mathrm{F}}_{\infty}$.

This symmetry is observed for a number of other statistical classifiers. Unfortunately, there are no similar analytical expressions for many of them. Nevertheless, in Chapter 3 we have had a number of the theoretical equations of the expected PMC. These equations characterise the increase in the mean generalisation error $\Delta_N = \bar{\varepsilon}^{\mathrm{A}}_{N} - \bar{\varepsilon}^{\mathrm{A}}_{N}$. The symmetry suggests that they can be used to evaluate the bias of the resubstitution estimate. Thus, these theoretical results may help to determine conditions when the resubstitution estimator can be used to estimate the asymptotic PMC. From the other hand, the theoretical difference $\Delta_N = \bar{\varepsilon}^{\mathrm{A}}_{N} - \varepsilon^{\mathrm{A}}_{\infty}$ can be utilised to improve the resubstitution estimate

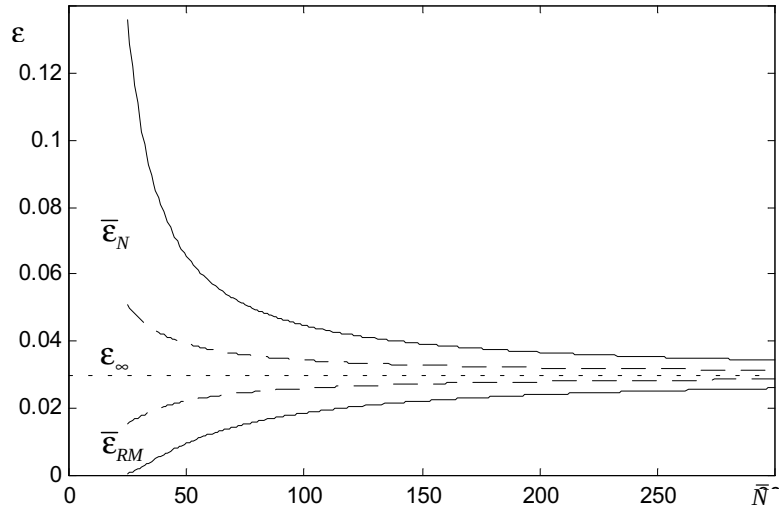

Fig. 6.3. The expected PMC $\bar{\varepsilon}_N$ and resubstitution error $\bar{\hat{\varepsilon}}_{RM}$ of the Fisher linear DF versus the number of training vectors N. The GCCM data model with the Mahalanobis distance $\delta = 3.76$ and dimensionality $n = 30$ (solid lines) and $n = 10$ (dashed lines).

For example, in order to estimate the expected PMC of the Fisher linear classifier instead of $\hat{\varepsilon}_{RM}^{(F)}$ one can use:

$$\hat{\varepsilon}_{RM}^{F*} = \hat{\varepsilon}_{RM}^{F} + \left(\Phi\{-\frac{\delta}{2}\frac{1}{\sqrt{T_M T_{\bar{\Sigma}}}}\}-\Phi\{-\frac{\delta}{2}\sqrt{T_M T_{\bar{\Sigma}}}\}\right) \text{ or}$$

$$\hat{\varepsilon}_{RM}^{F+} = \hat{\varepsilon}_{RM}^{F} + 2\,\Delta_N = \hat{\varepsilon}_{RM}^{F} + 2\left(\Phi\{-\frac{\delta}{2}\frac{1}{\sqrt{T_M T_{\bar{\Sigma}}}}\}-\Phi\{-\frac{\delta}{2}\}\right), \qquad (6.30)$$

where Δ_N denotes an absolute increase in the mean generalisation error.

The resubstitution estimate is a *random variable*. The distribution of the error counting resubstitution estimate is conditioned on $E\hat{\varepsilon}_{RM}$, its expected value. A variance of $\hat{\varepsilon}_{RM}$ is

$$V[\hat{\varepsilon}_{RM}] \approx \frac{E\hat{\varepsilon}_{RM}(1-E\hat{\varepsilon}_{RM})}{N_d}, \qquad (6.31)$$

where A, an index of the classifier's type, has been omitted.

Comparison of (6.31) with (6.27), the variance of the hold-out estimate, indicates that for $N_d = N_v$ the variance of the resubstitution error is smaller. The difference is caused by the fact that the expected value of the resubstitution

estimate, $\bar{\hat{\varepsilon}}_{RM}$, is smaller than the expected value of the hold-out estimate, $\bar{\hat{\varepsilon}}_{HM}$. *Relative variances* $V[\hat{\varepsilon}_{RM}]/(\bar{\hat{\varepsilon}}_{RM})^2$, and $V[\hat{\varepsilon}_{HM}]/(\bar{\hat{\varepsilon}}_{HM})^2$, however, are virtually the same.

6.3.1.3 The Leaving-One-Out Estimator

When the observations in the design set are statistically independent, the leaving-one-out method (N_d-fold cross-validation) is practically the unbiased estimator of the conditional PMC e_N^{A}. The variance of the leaving-one-out error counting estimate is expressed by the following equation

$$V[\hat{\varepsilon}_{LOOM}] = \frac{\varepsilon_{N_d}(1-\varepsilon_{N_d})}{N_d}. \tag{6.32}$$

In Equation (6.32), N_d is used to emphasise that one does not split the design set into the training and validation sets. All vectors are used, instead of a portion.

We emphasise once more that the leaving-one-out estimate is unbiased only if all design set vectors are statistically independent. Statistically dependent vectors more or less duplicate each other. Each omitted vector can be correlated with other vectors used for training. Thus, the leaving-one-out estimate becomes similar to resubstitution estimate which is optimistically biased.

For large N (when $\hat{\varepsilon}_{LOOM}^{\mathsf{A}} / \hat{\varepsilon}_{RM}^{\mathsf{A}} < 2$) the symmetry of the learning curves of the resubstitution estimate and the expected probability of misclassification (the expected value of the leaving-one-out estimate) suggests the following simple, almost unbiased estimate of the asymptotic error

$$\hat{\varepsilon}_{\infty}^{(\mathrm{A})} = \tfrac{1}{2}(\hat{\varepsilon}_{LOOM}^{\mathsf{A}} + \hat{\varepsilon}_{RM}^{\mathsf{A}}). \tag{6.33}$$

In the above estimate, index A emphasises that the asymptotic error is specific for each classifier design method. Note, the difference $\hat{\varepsilon}_{LOOM}^{\mathsf{A}} - \hat{\varepsilon}_{RM}^{\mathsf{A}}$ (or the ratio $\hat{\varepsilon}_{LOOM}^{\mathsf{A}} / \hat{\varepsilon}_{RM}^{\mathsf{A}}$ can be utilised to evaluate the training-set size necessary for sufficient training of the classifier A.

6.3.1.4 The Bootstrap Estimator

In the small design set case, estimate (6.5) is a slightly biased estimate of the conditional PMC $\varepsilon_N^{\mathsf{A}}$. To overcome the bias, a number of modifications of the bootstrap method have been proposed in the literature. The variance of the first term in the right side of Equation (6.5) is the variance of the bootstrap resubstitution estimate. The expectation of the resubstitution estimate is smaller than the expected probability of misclassification. Therefore, the variance of the resubstitution estimate is smaller than the variance of the leaving-one-out estimate.

The bias term $Bias(\hat{\varepsilon}_{RM})$ in Equation (6.5) is also a random variable. Theoretical and experimental analysis has shown that the bootstrap estimate is more accurate for a small number of design set vectors N_d and for large values of asymptotic error ε_∞^A if compared with the leaving-one-out estimate. The variance of the bootstrap estimate approaches the variance of the leaving-one-out estimate (6.32) with an increase in N_d and a decrease in ε_∞^A

$$V[\hat{\varepsilon}_{BM}] \approx \frac{\varepsilon_{N_d}(1-\varepsilon_{N_d})}{N_d} \approx V[\hat{\varepsilon}_{LOOM}]. \tag{6.34}$$

6.3.2 Parametric Estimators for the Linear Fisher Classifier

The error counting estimators simply count the number of misclassifications and evaluate its proportion among the total number of vectors used. The estimate is nonparametric in nature. In this estimator, no assumptions concerning the distribution of either the pattern vectors or the discriminant function are made.

In order to construct the parametric estimates discussed in Section 6.1.2.3, we have assumed that the classes are multivariate Gaussian and the covariance matrices are common. This assumption provides an important additional piece of information. Suppose this assumption is correct. Obviously, if correctly used, this information can help improve the accuracy of the estimators.

Let us evaluate this gain in the accuracy for the GCCM data model. The sample estimate $\hat{\delta}^2$ of the Mahalanobis distance (6.11) can be represented as a scaled non-central F random variable (6.12). For large N, a variance of the sample estimate δ^2 is

$$V[\hat{\delta}^2] = \frac{\delta^2}{\bar{N}}\left(\frac{T_{\boldsymbol{M}}}{T_{\bar{\boldsymbol{\Sigma}}}}\right)^2\left\{\frac{\delta^2}{T_{\boldsymbol{\Sigma}}}+\frac{8}{T_{\boldsymbol{M}}}\right\}. \tag{6.35}$$

Utilisation of the first terms of the Taylor expansion of the parametric estimate (6.11) and the variance $V[\hat{\delta}^2]$ gives

$$V[\hat{\varepsilon}_D] = \frac{\phi\left(\frac{\delta}{2}\sqrt{T_{\boldsymbol{M}}T_{\bar{\boldsymbol{\Sigma}}}}\right)^2}{16\bar{N}} T_{\bar{\boldsymbol{\Sigma}}}\left(\delta^2 T_{\boldsymbol{M}}T_{\bar{\boldsymbol{\Sigma}}}+8\right). \tag{6.36}$$

Numerical calculations show that, for moderate values of δ, n and N, the variance (6.36) is lower than the variance of the resubstitution error counting estimate (6.31). A comparative difference between variances of both the classification error estimates increases with an increase in the distance δ (a decrease in the asymptotic

error ε_{∞}^{F}). It means that, for small error rates, the parametric estimate can become much more effective than the nonparametric one.

The estimate (6.12) is derived for multivariate Gaussian distributions with equal covariances. In real situations, the data differs from the theoretically postulated model. Therefore, undesired bias arises in estimating the classification error rates. In principle, the bias can be positive or negative. Its value depends on a difference between the idealised model (GCCM) and the real distributions.

There is one more important general remark concerning all types of estimates: the standard deviations of the estimates are directly related to their mean values. The smaller the mean value, the smaller will be the variance. Therefore, the optimistic bias of the estimate decreases the variance. Consequently, the resubstitution and the bootstrap estimates just seem to be more accurate than the leaving-one-out and the hold-out estimates.

6.3.3 Associations Between the Classification Performance Measures

In the model selection, one can use the error estimates as well as the simplified performance measures. The designer of a pattern recognition system should be aware that there is a great number of types of the classification errors. The simple, easy to calculate, measures often only approximately characterise the real quality of the feature subset or the network architecture. Moreover, a real rank of the feature subset depends on the type of classification rule used later. The features which provide good performance for the nonparametric Parzen window classifier can appear to be poor performers for the parametric linear Fisher classifier. And vice versa, in the small training-set case, uni-modal very highly-correlated features can be poor performers for the Parzen window classifier and very good for the linear Fisher classifier. Figure 6.2 shows two simple illustrative examples of this fact. The highly-correlated features are also poor performers for the Parzen window, the binary decision tree and the Radial basic function neural classifiers. Thus, the features which are good for the perceptron can be bad for the RBF network.

Example 2. The following examples are presented in order to demonstrate associations between different classification performance measures and the classification error. To avoid the finite sample size effects and reveal only similarities and differences between the measures, we utilise the resubstitution performance estimates.

Quadratic DF versus the Fisher linear DF. Figure 6.4 presents a scatter diagram of distribution of 126 values of the probability of misclassification (PMC) of the quadratic DF versus the PMC of the Fisher linear DF. Real-world lung data was used. We examined $C_9^5 = 126$ five-variate subsets of features selected from nine spectral and cepstral original features. For comparison of the two classification error evaluation methods, the resubstitution error counting method was used (the design set size was $N_1 = N_2 = 90$).

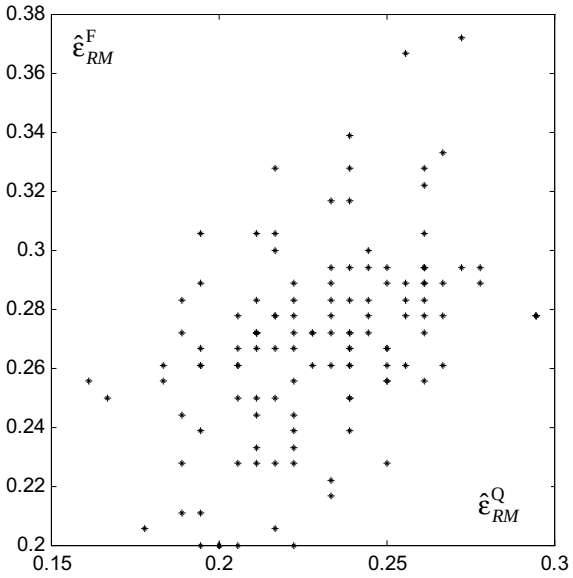

Fig. 6.4. The probability of misclassification of the linear Fisher $\hat{\varepsilon}_{RM}^{(\mathrm{F})}$ versus the probability of misclassification of the quadratic DF $\hat{\varepsilon}_{RM}^{(\mathrm{Q})}$.

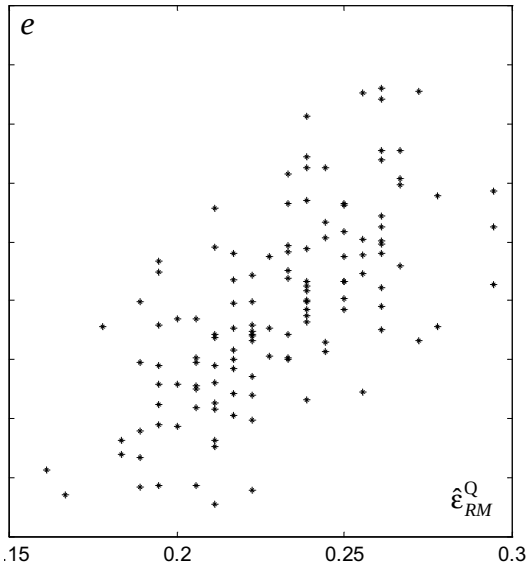

Fig. 6.5. The entropic loss e versus the PMC of the quadratic DF $\hat{\varepsilon}_{RM}^{(\mathrm{Q})}$.

One can see an obvious difference in the quality of the individual feature subsets evaluated by utilisation the standard linear and quadratic classifiers. The correlation coefficient between the performance criteria $\hat{\varepsilon}_{RM}^{(Q)}$ and $\hat{\varepsilon}_{RM}^{(F)}$ is rather small: $\rho = 0.47$. Note that for the GCCM pattern classes, theoretically one should obtain correlation $\rho = 1$. The scatter diagram characterises the differences in the classification errors which are functions of the mean vectors, covariance matrices and deviations of the data from the Gaussian distribution. Thus, the scatter diagrams can be useful as compact descriptions of the multivariate distribution of the pattern classes.

Quadratic classifier versus the entropic loss (6.24). In Figure 6.5, a scatter diagram is presented for 126 feature subsets evaluated from the lung data set. One can see that the performance evaluation measures rank the feature subsets in different ways: the correlation coefficient $\rho = 0.67$. It means that, for this particular lung diagnosis problem with cepstral features, the entropic loss is far from being a very good quality measure for the feature selection if, in future, the quadratic classifier is utilised.

Saliency versus PMC of the perceptron. The saliency (6.25), a popular measure in ANN architecture selection, is also an approximate quantity. Figure 6.6 contains a scatter diagram of the distribution of 54 values of the saliency $s_i = 1/1080 \Sigma s_i(\boldsymbol{X}_j)$ versus the resubstitution estimate of PMC of the non-linear SLP. Both measures are estimated for 53-variate vectors representing short segments of the lung noise.

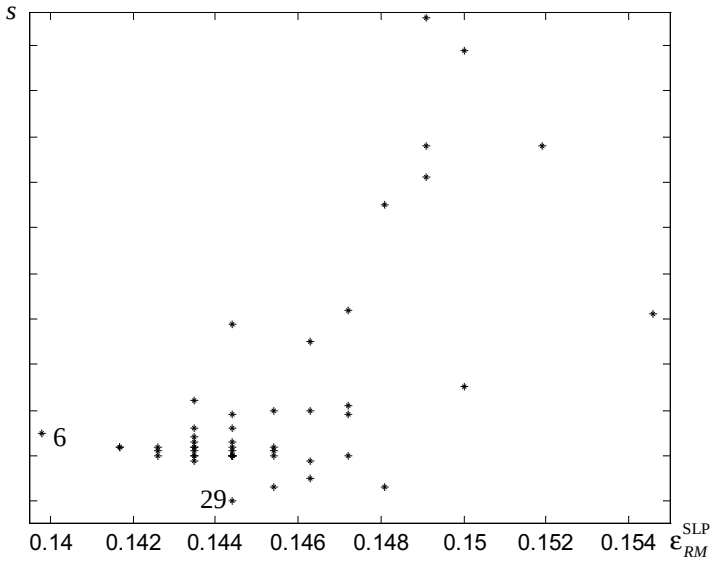

Fig. 6.6. Saliency versus the probability of misclassification $\varepsilon_N^{(SLP)}$ of the non-linear SLP.

The number of the vectors in the design set is 1080 (540 + 540). In this classification problem, we have used the nine-variate data which was mapped into 54-variate space of the new polynomial features. We considered the backward feature selection, where 53-variate feature subsets were selected from the 54 features. Thus, we calculated the saliency for m = 54 feature subsets. In this experiment, after training the SLP with 54 inputs for 1000 iterations in batch mode, the saliency was calculated for each feature. Then this feature was rejected and after 500 additional iterations, the PMC was evaluated.

It is much easier to calculate the saliency than to train the perceptron for the additional 500 iterations and to evaluate the classification error rate afterwards. Figure 6.6 illustrates that in the perceptron architecture selection, the classification error and the saliency suggest the rejection of different features (features 6 and 29). We have a notable difference between the classification error of the feature subsets without feature 6 or feature 29. In multiple application of this procedure, the selection errors accumulate. The difference allows one to assess the computation cost in terms of the classification error that is gained while utilising the proper feature selection criterion.

Low correlation between the simplified performance measures and the classification error is characteristic to most real-world problems. The indirect secondary performance measures are only approximately related to the classification accuracy. Their utilisation provides a foundation for finding new features but sometimes they give little information to minimise the classification error. Thus, if one utilises simple inexact performance measures, there is no need to apply a complex selection strategy to test a large number of models. We will return to this problem in Section 6.5.2.

6.4 Feature Ranking and the Optimal Number of Features

In Section 1.5, the peaking phenomenon was explained in the following situation:

- in the high-dimensional and finite training-set size case, the generalisation error can increase substantially;
- some of the features are almost worthless.

In this section, it will be shown that, in addition to the training-set size, two extra factors affect the peaking phenomenon and the optimal number of features:

- the classifiers' complexity;
- the order in which the features are presented.

6.4.1 The Complexity of the Classifiers

Example 3. In order to understand more clearly the peaking phenomenon in feature selection, let us return to the simple but instructive multivariate GCCM

data model, already discussed in Section 1.5. The means $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ are different and the covariance matrix $\overline{\Sigma}$ is common to both pattern classes. Let

$$\boldsymbol{M}_1 - \boldsymbol{M}_2 = (\delta_0, \delta_0, ..., \delta_0)^T \text{ and } \Sigma = \begin{bmatrix} 1 & \rho & ... & \rho \\ \rho & 1 & ... & \rho \\ ... & ... & ... & ... \\ \rho & \rho & ... & 1 \end{bmatrix}. \tag{6.37}$$

In this data model, all individual features have exactly the same discriminative power i.e. $\delta_i^2 = \delta_0^2$, because $m_{1i} - m_{2i} = \delta_0$. The Bayes error and asymptotic probability of misclassification of the Fisher linear DF are uniquely determined by the Mahalanobis distance $\delta^2 = (\boldsymbol{M}_1 - \boldsymbol{M}_2)^T \overline{\Sigma}^{-1} (\boldsymbol{M}_1 - \boldsymbol{M}_2)$. For the model analysed:

$$\overline{\Sigma}^{-1} = \begin{bmatrix} a & b & ... & b \\ b & a & ... & b \\ ... & ... & ... & ... \\ b & b & ... & a \end{bmatrix},$$

$$\text{where } a = \frac{n\rho - 2\rho + 1}{(1-\rho)(n\rho + 1 - \rho)}, \; b = -\frac{\rho}{(1-\rho)(n\rho + 1 - \rho)}, \; \delta^2 = \frac{n\,\delta_0^2}{n\rho + 1 - \rho}. \tag{6.38}$$

All features are equally informative; on the other hand, the features are correlated among themselves. Therefore, with an increase in n the Mahalanobis distance increases quickly at first but later this increase saturates. A limit value of the Mahalanobis distance is $\delta^2 \rightarrow \delta_0^2/\rho$, as the number of features $n \rightarrow \infty$.

As in Figure 1.8*a*, in Figure 6.7, we have the graph of dependence of the asymptotic classification error $\varepsilon_\infty^F = \varepsilon_B = \Phi\{-\tfrac{1}{2}\delta\}$ on the dimensionality n (the dotted curve). The data model (6.37) with $\delta_0^2 = 3.0$ and $\rho = 0.5$ was used. In Figure 6.7, we have the expected generalisation errors of the linear Fisher DF, F, and the quadratic DF, Q, versus n calculated from Equations (3.10) and (3.19). We see the optimal dimensionality for the quadratic classifier is smaller.

In Section 1.5, it was demonstrated that the optimal number of features depends on the number of training vectors. Now one can observe that it depends also on the complexity of the classifier. The optimal dimensionality increases with an increase in the training-set size and decreases with an increase in the complexity of the classifier (in strict terms, the complexity of the algorithm used to find the weights).

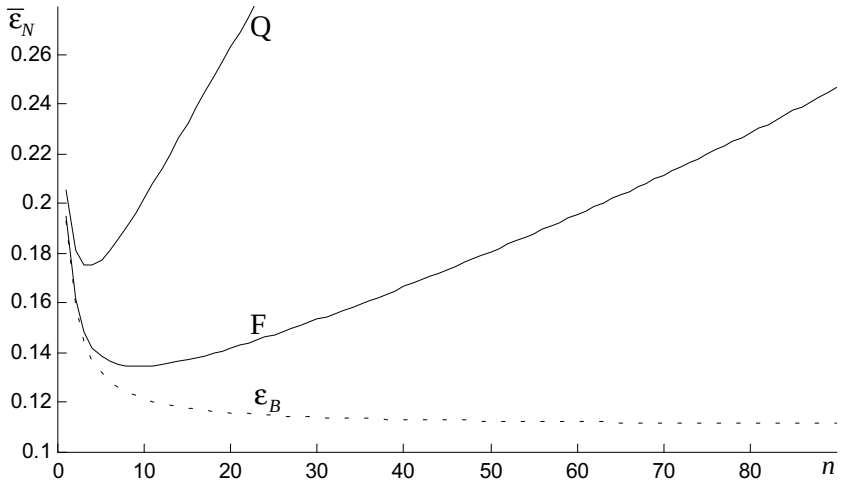

Fig. 6.7. The Bayes error ε_B and expected errors of the linear and quadratic classifiers versus dimensionality n; $N_2 = N_1 = 80$.

6.4.2 Feature Ranking

In Figure 6.7, the graphs were obtained for the model when all features are equally informative. The features have different levels of quality in most real-world problems. There are good and bad uninformative features. Thus, the researcher ranks and selects the best features first. After the feature ranking, the contribution of a single feature into the Mahalanobis distance depends on the feature number: the first features contribute the most and each following feature contributes less than the previous one.

Example 4. Consider a data model where individual features have different discriminative powers. All the features are statistically independent and the contribution of each single feature depends on its order number:

$$\delta_i^2 = (m_{1i} - m_{2i})^2 / \sigma_i^2 = \delta_0^2 \gamma^i \qquad (\gamma < 1).$$

The Mahalanobis distance of all n features is

$$\delta^2 = \delta_0^2 \sum_{j=1}^{n} \gamma^j = \delta_0^2 \gamma (1-\gamma^n)(1-\gamma)^{-1}. \tag{6.39}$$

Utilisation of data model (6.39) with parameters $\delta_0^2 = 0.8$, $\gamma = 0.9$ and the linear Fisher DF for the training-set size $\overline{N} = 80$ gives the two graphs marked 1 in Figure 6.8. In this figure, *dotted curves* represent the asymptotic PMC. *Solid curves*

represent the expected PMC. The two curves marked 1 correspond to the optimal and best possible feature ranking (ordering). Here, we use the best feature with the largest δ_i^2 at first, etc.

The curves marked by the number 2 correspond to the worst ordering when the worst features (with smallest δ_i^2) are presented at first. For the parameter values under consideration ($\gamma = 0.9$ and $\bar{N} = 80$) we have no minimum.

Let the features be presented at random. It is natural to assume $\delta_i^2 \approx \delta^2/n$. This situation is illustrated by the curves marked 3 in Figure 6.8. For the linear Fisher DF, $n_{opt} \approx \bar{N}$. It is a much higher value than n_{opt} obtained for the best ordering.

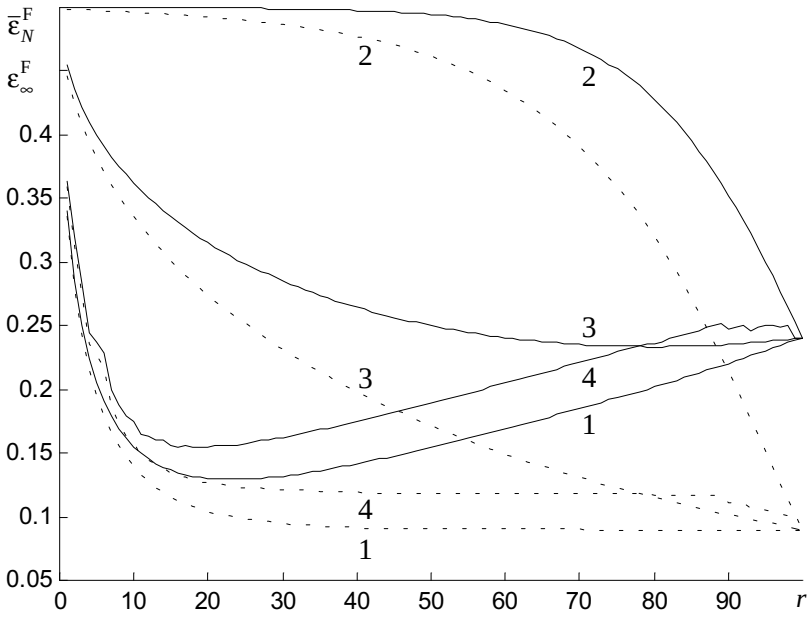

Fig. 6.8. Influence of feature ordering on the character of the curves; the generalisation (solid curves) and asymptotic errors (dotted curves) versus the number of features *r*: 1 – the best possible ordering, 2 – the worst possible ordering, 3 – random ordering, 4 – ordering according to inaccurate estimates of the individual quality of the features.

In practice, however one never has the best, the worst or random ordering. One performs the feature selection first and then determines the optimal number of features. To illustrate the non-optimal feature ranking in Figure 6.8, there are two curves denoted by 4. The graphs are obtained for inaccurate individual feature ranking based on imprecise estimations of δ_i^2. In this virtual experiment, to each value of δ_i^2 we artificially added a random zero-mean noise, ξ_i, with variance

$V(\xi_i)= \delta_i^2 \, (\delta_i^2 +8) / \bar{N}$. Each single feature has a different noise variance. In this example, the noise variance depends on the feature's discriminative power: the noise variance was calculated using Equation (6.35) for a new dimensionality $n = 1$ (in feature ranking, we evaluated only one feature at a time) and $\delta = \delta_i = \delta_0^2 \, \gamma^i$. This data was used to imitate imprecise estimates for the feature quality. As a result of inaccurate feature ranking, the asymptotic PMC and expected PMC (curves 4 in Figure 6.8) are higher than in the case of the ideal feature ranking.

For non-optimal feature ranking, the optimal number of features is usually higher than for ideal ranking. However, exceptions are possible. The minimum of the expected PMC for non-optimal ranked features is *always* higher. Thus, the optimal dimensionality and the generalisation error depends on the feature ordering.

The spherical Gaussian distribution of the pattern classes is only one known data model where the training-set based feature ordering and subsequent feature selection do not allow the reduction of the generalisation error of EDC in the Bayes predictive sense (see Section 2.4.2, also Serdobolskij, 1983*b*). For most other models of the data and the classification rules the sample-based feature ranking can be effective.

The limited training-set size is only one factor that causes the peaking effect. Let the certain parametric model (say, the model A) of the distribution densities $p_i(\boldsymbol{X} \mid \Theta^{\mathsf{A}})$ be used to design the classification rule and the data model $p_i^{data}(\boldsymbol{X})$ be different. There exist situations when the asymptotic PMC $\varepsilon_\infty^{\mathsf{A}}$ (but not the Bayes error!) decreases at first and then begins to increase with a further increase in the number of features. This kind of peaking effect has already been illustrated in Chapter 1 (see Figure 1.8 *b*).

6.4.3 Determining the Optimal Number of Features

Summarising the above analysis, the main factors which affect the optimal number of features are:

- the pattern recognition problem to be solved and the order in which the features are presented;
- the assumptions used to design the classifier and a correspondence of the assumptions to the data where the algorithm should be applied;
- the size of the training-set and the individual characteristics of the vectors that compose this set.

The above factors are interrelated. Therefore, *a priori*, it is impossible to determine n_{opt} exactly. An enormous number of possible theoretical models of the true multivariate densities $p_i^{data}(\boldsymbol{X})$ suggests that the optimal number of features should be determined experimentally by trial and error. After the type of classifier is selected and the features (or the feature subsets) are ranked, one needs to obtain

the unbiased estimates $\hat{\varepsilon}_j$ of the generalisation error of the feature subsets containing 1, 2, ..., n-1 features. According to the estimates $\hat{\varepsilon}_1$, $\hat{\varepsilon}_2$, $\hat{\varepsilon}_3$, ..., the subset with the minimal error is selected.

In general, the feature subset selection based on imprecise estimates $\hat{\varepsilon}_1$, $\hat{\varepsilon}_2$, $\hat{\varepsilon}_3$, ..., is inaccurate. One of the possibilities for improving the selection accuracy is to smooth an empirical dependence of the estimates $\hat{\varepsilon}$ versus the number of features r, $\hat{\varepsilon} = f^{\mathsf{A}}(r)$, ($r$ = 1, 2, …, n) and to determine the optimal dimensionality, n_{opt}, according to a minimum of a smoothed curve $\hat{\varepsilon} = f^{\mathsf{A}}(r)$. Note that, for certain classifiers (A) and data types, the curve $\hat{\varepsilon} = f^{\mathsf{A}}(r)$ can have more than one minimum.

Similar considerations and procedures should be applied to solving the related problems: determination of the optimal smoothing parameter of the PW classifier, the regularisation constant in RDA, an optimal number of nearest neighbours, optimal number of iterations in ANN training, etc. In all these complexity control problems, one needs to obtain independent unbiased performance estimates and to find their minimum. The smoothing of empirical dependencies, accuracy versus the smoothing parameter, is also useful.

6.5 The Accuracy of the Model Selection

A priori, it is impossible to select the best sequence of the feature extraction and the classifier design procedures in order to get a good solution. It is a characteristic of real-world pattern recognition problems. Therefore, the designer is forced to test a number of variants.

There are two fundamental aspects of the accuracy problem:

- the accuracy of performance estimates utilised to evaluate quality of each competing variant,
- the accuracy of the selection process where we select one model on a basis of inaccurate estimates of a number of competing models.

6.5.1 True, Apparent and Ideal Classification Errors

In the selection process, the designer usually applies a certain performance evaluation method to estimate the classifier's accuracy and find the best solution. In addition to the four types of classification error discussed in Section 6.1, three new kinds of the classification error arise in the selection process. This problem is illustrated by the following example.

Example 5. To solve the two-category classification problem, five (m = 5) potential classification algorithms are tested:

A1 – the Fisher linear discriminant function;
A2 – the nonparametric three-nearest neighbour classifier;

A3 – the nonparametric five-nearest neighbour classifier;
A4 – the single layer perceptron with four hidden units;
A5 – the single layer perceptron with eight hidden units.

Suppose that the training-set is composed of $N_{t1} = 200$, $N_{t2} = 200$ vectors and an independent validation set of $N_{v1} = 250$, $N_{v2} = 250$ vectors. Let the validation set's estimates of the probability of misclassification ε_v be:

for classifier A1 $\hat{\varepsilon}_{v1} = 0.042$;
for classifier A2 $\hat{\varepsilon}_{v2} = 0.052$;
for classifier A3 $\hat{\varepsilon}_{v3} = 0.038$;
for classifier A4 $\hat{\varepsilon}_{v4} = 0.048$;
for classifier A5 $\hat{\varepsilon}_{v5} = 0.040$.

Then a natural choice would be to select the classifier A3 with $\hat{\varepsilon}_{v3} = 0.038$ and to recommend this classifier for further use, referring to its generalisation error $\varepsilon_N = 0.038$. The statement that the performance of the best classifier is 0.038, however, is not correct. To explain this, suppose we have an independent very large test set composed from $Nt_1 = 1000$ and $Nt_2 = 1000$ vectors in order to estimate the PMC of all five classifiers. Let the estimates of the probability of misclassification ε_t obtained using 2000 observation vectors be:

for classifier A1 $\varepsilon_{t1} = 0.0405$;
for classifier A2 $\varepsilon_{t2} = 0.0605$;
for classifier A3 $\varepsilon_{t3} = 0.0480$;
for classifier A4 $\varepsilon_{t4} = 0.0465$;
for classifier A5 $\varepsilon_{t5} = 0.0515$.

We assume these estimates are accurate as they are obtained from a very large test set. In Figure 6.9, all five pairs of estimates in a bi-variate ε_t - $\hat{\varepsilon}_v$ space are presented. From the test set estimates it follows that one should make use of the classifier A1 having an error rate $\varepsilon_{t1} = 0.0405$. However, the decision based on the validation set had suggested the use of the classifier A3. The actual (true) error rate of the classifier A3 is $\varepsilon_{t3} = 0.0480$. Thus, inexact model selection based on the small validation set caused an increase in the classification error:

$$\varepsilon_{t3} - \varepsilon_{t1} = 0.0480 - 0.0405 = 0.0075.$$

The true (test set) classification error of the algorithm selected according to the validation estimates is called the *true PMC in selection* and is denoted by ε_{true}. In our example, $\varepsilon_{true} = 0.0480 = \varepsilon_{t3}$.

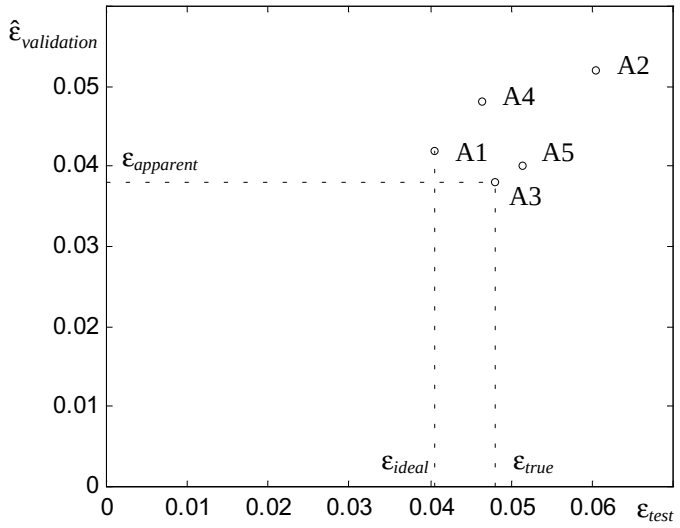

Fig. 6.9. A distribution of five pairs of estimates in the bi-variate ε_t - $\hat{\varepsilon}_v$ space and the apparent, true and ideal error rates in the selection of the best variant.

The validation set estimate of the classification error of the algorithm selected according to the validation estimates is called the *apparent PMC in selection*, and is denoted by $\varepsilon_{apparent}$. In our example, $\varepsilon_{apparent} = 0.0038 = \hat{\varepsilon}_{v3}$.

If the model selection was performed on the basis of the accurate estimates (in the above example, the estimates ε_{t1}, ε_{t2}, ε_{t3}, ε_{t4}, ε_{t5}), then one would select the first algorithm A1. The true (test set) classification error of the classification algorithm selected according to the true (exact) estimates is called an *ideal classification error in selection.* In our example, $\varepsilon_{ideal} = 0.0405 = \varepsilon_{t1}$.

It follows from the definition that $\varepsilon_{ideal} \leq \varepsilon_{true}$. In the case where the estimates are unbiased, most often $\varepsilon_{apparent} < \varepsilon_{ideal}$. For some randomly chosen validation sets, however, this inequality can be violated. The increase in the classification error due to non-ideal model selection

$$\Delta_{selection} = \varepsilon_{true} - \varepsilon_{ideal}$$

is called the *selection loss.*

6.5.2 An Effect of the Number of Variants

In classifier design, these three probabilities crop up all the time and are necessary in the analysis of the accuracy of feature selection, neural networks initialisation, architecture selection, etc. Therefore, in practice, one needs to know the relations between the errors considered and the parameters of a particular pattern-recognition problem to be solved.

Example 6. Figure 6.10 presents a scatter diagram of 200 pairs of the leaving-one-out estimate, $\hat{\varepsilon}$, of the generalisation error of the linear Fisher DF (design set size 25 + 25 = 50 vectors) and the estimate of the conditional PMC obtained on the independent test set (Nt = 500 vectors). The estimates were calculated using 200 independent randomly chosen four-variate subsets of features, selected from a pool of 19 features. Real-world medical data was used.

The leaving-one-out estimate, $\hat{\varepsilon}$, and the conditional PMC rank the feature subsets in different ways. In this example, the feature subset 63 is the best among all 200 subsets according to the leaving-one-out estimate: it results in the smallest error estimate $\hat{\varepsilon}_{63} = 0.14 = \varepsilon_{apparent}$. The test set estimate indicates differently: feature subset 152 is best, with the classification error $\varepsilon_{152} = 0.208 = \varepsilon_{ideal}$. If one selects the subset 63, then its conditional PMC would be $\varepsilon_{63} = 0.228 = \varepsilon_{true}$. The inaccurate feature subset selection increases the classification error from 0.208 to 0.228 ($\Delta_{selection} = 0.02$).

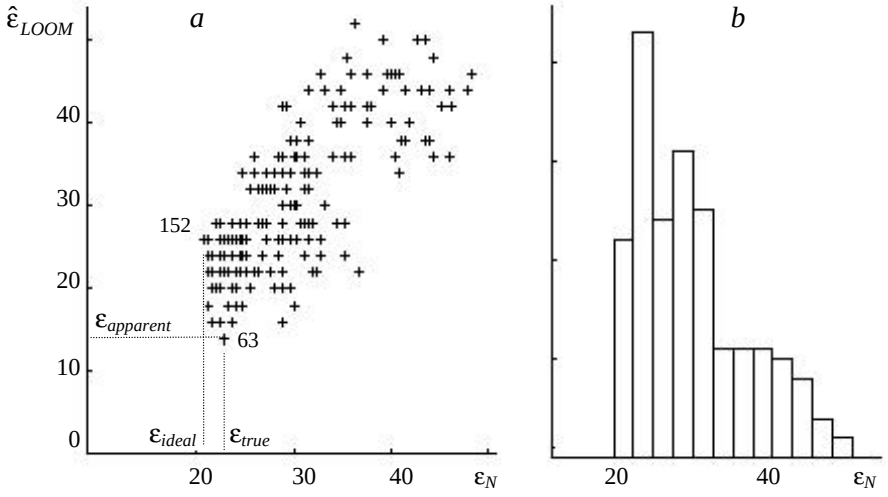

Fig. 6.10. The scatter diagram (*a*) and a histogram (*b*) of the distribution of 200 values of the conditional PMC, ε_N, and its leaving-one-out estimates, $\hat{\varepsilon}_{LOOM}$, evaluated for 200 randomly chosen subsets of features; $n = 19$, $r = 4$.

In order to analyse the behaviour of quantities $\Delta_{selection}$, ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$ analytically, one needs to make assumptions that the classifiers (the variants of the classification algorithm or the feature subsets) are chosen randomly. Furthermore, one can assume the estimates $\hat{\varepsilon}_1$, $\hat{\varepsilon}_2$, $\hat{\varepsilon}_3$, ... and the true probabilities of misclassification ε_1, ε_2, ε_3, ... to be *independent* random variables with a *known* joint distribution density function $f(\hat{\varepsilon}_j, \varepsilon_j)$. Then, from m randomly chosen pairs $(\varepsilon_{j1}, \hat{\varepsilon}_{j1})$, $(\varepsilon_{j2}, \hat{\varepsilon}_{j2})$, …, $(\varepsilon_{jm}, \hat{\varepsilon}_{jm})$, one selects two of the best variants (the first time according to $\hat{\varepsilon}_j$ and the second time according to ε_j) and analyses the expected behaviour of ε_{true}, ε_{idea}, $\varepsilon_{apparent}$ and $\Delta_{selection}$.

Let a distribution density $p(\hat{\varepsilon}_j, \varepsilon_j)$ of bi-variate vector $(\hat{\varepsilon}_j, \varepsilon_j)$ be known. This bi-variate density can be expressed as a product: $p(\hat{\varepsilon}_j, \varepsilon_j) = p(\hat{\varepsilon}_j | \varepsilon_j)\, p(\varepsilon_j)$. We will derive analytical expressions for distribution densities and the statistical moments of ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$. Utilisation of the *theory of the extremes* provides a distribution density of ε_{ideal} that can be expressed as

$$p_{ideal}(\varepsilon) = m(1-P_{\varepsilon}(\varepsilon))^{m-1}\, p(\varepsilon), \tag{6.40}$$

where $p(\varepsilon)$ is the unconditional distribution density of accurate performance values in different models and $P_{\varepsilon}(a) = \int_{-\infty}^{a} p(\varepsilon) d\varepsilon$ is a cumulative distribution function of ε.

Similarly, for the apparent error it can be written

$$p(\varepsilon_{apparent}) = m(1-P_{\hat{\varepsilon}}(\hat{\varepsilon}))^{m-1}\, p(\hat{\varepsilon}) = p_{apparent}(\hat{\varepsilon}), \tag{6.41}$$

where $p(\hat{\varepsilon})$ is the unconditional distribution density of estimates $p(\hat{\varepsilon}) = \int_{0}^{1} p(\hat{\varepsilon} | \varepsilon)\, p(\varepsilon) d\varepsilon$ and $P_{\hat{\varepsilon}}(a) = \int_{-\infty}^{a} p(\hat{\varepsilon}) d\hat{\varepsilon}$ is a cumulative distribution function of the estimate $\hat{\varepsilon}$.

The probability density of ε_{true} is

$$p_{true}(\varepsilon) = \int_{0}^{1} p(\varepsilon | \hat{\varepsilon})\, p_{apparent}(\hat{\varepsilon}) d\hat{\varepsilon}, \tag{6.42}$$

where $p(\varepsilon | \hat{\varepsilon}) = p(\hat{\varepsilon} | \varepsilon)\, p(\varepsilon) / p(\hat{\varepsilon})$.

In order to use the above equations to find the expected values of ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$ one needs to specify the joint distribution density function $f(\hat{\varepsilon}_j, \varepsilon_j)$.

The joint density will be defined as a product of conditional $p(\hat{\varepsilon}_j \mid \varepsilon_j)$ and unconditional $p(\varepsilon_j)$ densities:

$$p(\hat{\varepsilon}_j, \varepsilon_j) = p(\hat{\varepsilon}_j \mid \varepsilon_j)\, p(\varepsilon_j).$$

Analysis of artificially generated and real data indicates that, in the case where the original n variables are reasonably selected, often a distribution density $p(\varepsilon_j)$ of the PMC of the various subsets of variables is unimodal, bounded from the left and from the right. If good informative variables dominate over the bad ones the distribution density function has *positive asymmetry* (as in Figure 6.10*b*). If bad uninformative variables dominate over the good ones, the distribution of the values of PMC has *negative asymmetry*. Therefore, we will approximate the unconditional distribution density $p(\varepsilon_j)$ by that of the Generalised Beta distribution (Pearson I type)

$$p(\varepsilon_j) = k_\varepsilon (\varepsilon_j - \varepsilon_{min})^{\gamma-1} (\varepsilon_{max} - \varepsilon_j)^{\eta-1}, \tag{6.43}$$

where k_ε is a normalising constant, parameters ε_{min}, ε_{max} control the minimal and maximal values of ε_j, and γ, η control the shape of the density function $p(\varepsilon_j)$.

The sample based hold-out error counting estimate of the classification error is conditionally a binomial random variable. Let us consider a *hypothetical situation* where we assume that for the estimation of the classification performance of m variants one can have m *different* independent virtual validation sets of size N^*. Then, the conditional distribution $p(\hat{\varepsilon}_j \mid \varepsilon_j)$ is binomial, i.e. probability

$$P(\hat{\varepsilon}_j = \frac{i}{N^*} \mid \varepsilon_j) = C^i_{N^*} (\varepsilon_j)^i (1-\varepsilon_j)^{N^*-i}. \tag{6.44}$$

Note, in practice, one utilises only one validation set of size Nv vectors. For that reason, the parameter N^* characterises the evaluation accuracy of the models performance. It is an *effective validation set size*. In principle, the distribution model (6.44) can be applied when the error counting performance estimate or some other performance measure (e.g. divergence, saliency, entropy, etc) is used to evaluate quality of the competing models.

The use of Equations (6.43) and (6.44) in (6.40), (6.41) and (6.42) allows one to calculate the expected values of ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$ by numerical methods. In Table 6.1, we present integration results for the Generalised Beta – Binomial distribution model. We calculated the expected values of the true and apparent error rates for six m values (m is the number of models compared empirically). Here the distribution $p(\varepsilon_j)$ was supposed to be a symmetrical Generalised Beta distribution density with $\gamma = \eta = 4.0$ and $\varepsilon_{min} = 0.01$, $\varepsilon_{max} = 0.2$ (the upper part of Table 6.1), or $\varepsilon_{min} = 0.1$, $\varepsilon_{max} = 0.3$ (the lower part). The conditional distribution $p(\hat{\varepsilon}_j \mid \varepsilon_j)$ is Binomial with the effective virtual set size N^*. Values for $N^* = \infty$ correspond to ε_{ideal}, the expected value of the ideal error rate.

While inspecting Table 6.1, one can notice that the expected values $\bar{\varepsilon}_{true}$ and $\bar{\varepsilon}_{apparent}$ decrease with an increase in N^*. Both of the values approach ε_{ideal}. The expected difference $B_{selection} = \bar{\varepsilon}_{true} - \bar{\varepsilon}_{apparent}$, however, increases with N^*.

Table 6.1. The expected values of the true (the left column for each sample size) and apparent (the right column) error rates in the best model selection (from Raudys, 1981).

m \ N^*	50		100		200		500		∞
5	.083	.047	.078	.056	.074	.062	.071	.065	.068
10	.076	.032	.069	.043	.064	.050	.060	.054	.057
50	.066	.010	.056	.022	.049	.030	.044	.036	.040
100	.063	.005	.052	.016	.044	.024	.039	.030	.034
500	.061	.000	.046	.006	.037	.014	.031	.021	.026
10^6	.061	.000	.042	.000	.042	.000	.029	.000	.010
5	.180	.127	.175	.141	.170	.150	.165	.156	.161
10	.174	.106	.168	.124	.161	.135	.155	.143	.150
50	.165	.069	.155	.094	.147	.110	.139	.122	.131
100	.162	.057	.152	.084	.143	.101	.135	.114	.126
500	.157	.035	.145	.065	.136	.085	.128	.101	.117
10^6	.149	.000	.132	.014	.123	.041	.115	.067	.100

Similar selection accuracy evaluations can be drawn from the experimental data. Let a number of models investigated experimentally be M. Let we have a set of M bi-variate vectors $(\varepsilon_{j1}, \hat{\varepsilon}_{j1})$, $(\varepsilon_{j2}, \hat{\varepsilon}_{j2})$, …, $(\varepsilon_{jm}, \hat{\varepsilon}_{jm})$ and want to evaluate accuracy characteristics when the best variant is selected from m randomly chosen models ($m << M$). We investigate all possible collections composed of m models selected from a pool of M models. A number of such collections is

$$J = C_M^m = \frac{M!}{(M-m)!m!}.$$

We consider the model selection characteristics ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$ for each collection. An original algorithm developed by Pikelis (Appendix 4) can be utilised to find mean values and variances of ε_{true}, ε_{ideal} and $\varepsilon_{apparent}$ calculated for J possible collections produced from M bi-variate vectors $(\varepsilon_{j1}, \hat{\varepsilon}_{j1})$, $(\varepsilon_{j2}, \hat{\varepsilon}_{j2})$, …, $(\varepsilon_{jm}, \hat{\varepsilon}_{jm})$ of the experimental data.

Example 7. In Figure. 6.11, graphs of dependencies of average values of $\varepsilon_{apparent}$, ε_{true} and ε_{ideal} on the size of the collection, m_i (m_i = 2, 3, 5, 10, 20, 50 and 100) are presented. The mean values were calculated from $M = 1000$ bi-variate vectors $(\varepsilon_{ti}, \hat{\varepsilon}_{vi})$ obtained for 1000 randomly selected four-variate subsets of the features

from the data considered in Example 6 at the start of Section 6.5.2 (in Figure 6.10*a* we only have 200 bi-variate vectors selected from 1000).

Theoretical values and the experimental graphs show that both the *selection loss,* $\Delta_{selection}$, and the *selection bias,* $B_{selection}$, become large if the effective validation set size, N^*, is small and the number of models, m, is large. Therefore, if the number of variants (models) m is large and N^* is small, the apparent error of selection (the error estimate of the best variant) can become particularly optimistically biased and can provide little information in determining the real performance of the classification system. It is important to realise that each single performance estimate, $\hat{\varepsilon}_j$, can be unbiased, however, the *estimate of the best variant is biased*.

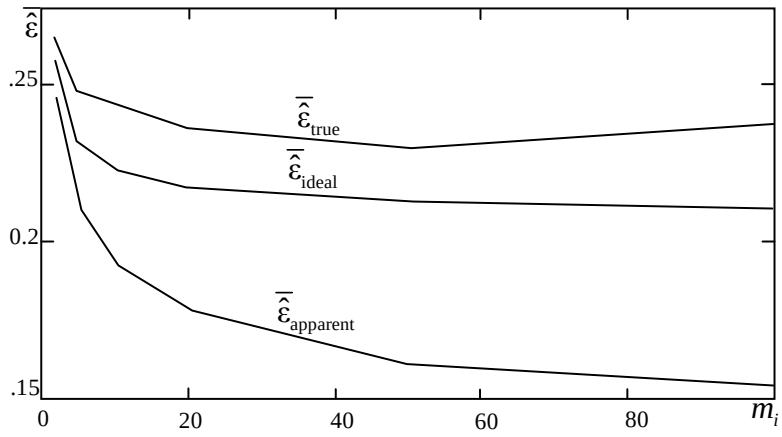

Fig. 6.11. The average values of the $\varepsilon_{apparent}$, ε_{ideal} and ε_{true} versus m_i, the number of the feature subsets.

The expected apparent error, $\bar{\varepsilon}_{apparent}$, diminishes continuously with the increase in m, the number of variants (models). The expected true error $\bar{\varepsilon}_{true}$ decreases with m at first, however, later it saturates and stops decreasing. An important conclusion follows: in finite validation set size case, it is not worth increasing the number of models investigated. It means one does not need to use sophisticated feature or ANN architecture selection methods, if the performance estimation accuracy is low.

Thus, it is reasonable to fix m_{max}, a maximal number of models, and economise on the computing time when the accuracy of estimates $\hat{\varepsilon}_1, \hat{\varepsilon}_2, \hat{\varepsilon}_3, \ldots$ is low. The last conclusion is based on the assumption that the distribution density $f(\varepsilon_j)$ is fixed *a priori* and does not change practically with an increase in m.

6.5.3 Evaluation of the Bias

Previous theoretical considerations about accuracy of the model selection process can help in understanding the accuracy problems in similar situations when one utilises certain inexact simplified performance evaluation methods instead of the exact classification error estimate. The theoretical considerations can be useful also when non-random model selection strategies (e.g., backward selection or pruning the network) are utilised instead of the random search.

In practice, the designer needs to know the difference $\Delta_{selection} = \varepsilon_{true} - \varepsilon_{ideal}$, the loss in the model selection, and $B_{selection} = \varepsilon_{true} - \varepsilon_{apparent}$, the selection bias, or to determine a rational estimate of the maximal number of models m_{max} to be compared experimentally. To solve these tasks one needs to know the distribution densities, $f(\hat{\varepsilon}_j|\varepsilon_j)$ and $f(\varepsilon_j)$.

Our notation "the effective validation size" is a conditional concept used in the previous subsection's formulation of the problem. In theoretical analysis, N^* is the size of m *independent virtual random validation sets* utilised to obtain estimates $\hat{\varepsilon}_1, \hat{\varepsilon}_2, \hat{\varepsilon}_3, ..., \hat{\varepsilon}_m$. While analysing theoretically it was supposed that different validation sets were utilised to obtain the estimates $\hat{\varepsilon}_1, \hat{\varepsilon}_2, \hat{\varepsilon}_3, ..., \hat{\varepsilon}_m$. In actual fact, we use a single validation set. Often other performance measures (such as the divergence, entropic loss or saliency) can be applied instead of the validation set error estimates. Thus, the size of the effective validation set size, N^*, depends on the method used to evaluate performance, the sample size used to obtain the estimates $\hat{\varepsilon}_1, \hat{\varepsilon}_2, \hat{\varepsilon}_3, ..., \hat{\varepsilon}_m$, the pattern recognition problem and a set of models compared experimentally. In practice this accuracy is also unknown.

Without some *additional information* it is impossible to know N^* before the experimentation. In order to evaluate $B_{selection}$ or the number m_{max} precisely, one needs to have a very large test set. Such a situation is unrealistic: one does not have to deal with problems of accuracy in very large test set size cases. We will consider a possible modus operandi when one does not have the test set and use some hypothetical assumptions (hypotheses) about the data.

Let us have M models to be compared. We have some "validation" set (it can be the design set if the leaving-one-out method is used) to estimate the performances of all M competing models.

Procedure 1.

1. Add spherical or coloured noise (see Section 4.6.4.3) to the validation set vectors and form a very large *pseudo-general population* (PGP). From this population, select a random *pseudo-validation set* (PVS) of the size Nv as before in model selection with the error counting estimates, and use both PVS and PGP to produce a set of pseudo-estimates $(\hat{\varepsilon}_1^*, \varepsilon_1^*), (\hat{\varepsilon}_2^*, \varepsilon_2^*), ..., (\hat{\varepsilon}_M^*, \varepsilon_M^*)$. While generating the PGP we assume that the bivariate distribution densities $f(\hat{\varepsilon}, \varepsilon)$ of the

true and pseudo-general populations are similar. This is in fact, our additional information about the data.

2. Subsequently use the pseudo-estimates $(\hat{\varepsilon}_1^*, \varepsilon_1^*), (\hat{\varepsilon}_2^*, \varepsilon_2^*), ..., (\hat{\varepsilon}_M^*, \varepsilon_M^*)$ and the Pikelis algorithm (Appendix 4) to evaluate the average values $\hat{\bar{\varepsilon}}_{apparent}$, $\hat{\bar{\varepsilon}}_{true}$, and $\hat{\bar{\varepsilon}}_{true} - \hat{\bar{\varepsilon}}_{apparent}$ for all $C_M^{m_i^*}$ possible collections of the vectors $(\hat{\varepsilon}_j^*, \varepsilon_j^*)$, say $m_i^* = 2, 5, 10, \ldots, M/5$.

3. Apply the standard statistical procedure to estimate the parameters ε_{min}, ε_{max}, γ, and η from the set $\varepsilon_1^*, \varepsilon_2^*, ..., \varepsilon_M^*$.

4. Select a set of the effective virtual validation set size N^*, say (Nv, $1.2Nv$, $1.4Nv$, …, $4.8Nv$, $5Nv$), and use Equations (6.43) and (6.44) in (6.40), (6.41) and (6.42) to calculate the expected values of ε_{true} and $\varepsilon_{apparent}$ for a number of selected values $m^*_i = 2, 5, 10, \ldots, M/5$ by numerical methods.

5. Select Nv which best fits the empirical differences $\hat{B}_{selection} = \hat{\bar{\varepsilon}}_{true} - \hat{\bar{\varepsilon}}_{apparent}$.

6. Calculate the difference of $B_{selection} = \varepsilon_{true} - \varepsilon_{apparent}$ for $m^*_i = M$ and Nv as determined in the previous step.

This procedure relies on an assumption that the PGP represents the designers problem sufficiently well.

Procedure 2.

Numerous simulation experiments show that it is reasonable to assume $3N < N^* < 5N$, when one uses error counting error estimates in the randomised best variant selection, where N is the number of vectors used to obtain the error estimates. For sketchy estimation of $B_{selection}$ and m_{max}, one can apply the data from Table 6.1 with $N^* = (3 \div 5)\, N$.

A rule of thumb.

For the error counting cross validation estimation use $B_{selection} = \alpha_s \sqrt{\hat{\varepsilon}(1-\hat{\varepsilon}) / Nv}$, where coefficient α_s depends on similarity of the models (the set of features, architecture of ANN, etc.) compared and m, a number of models. For $m = 10 \div 20$ different models with similar classification errors use $\alpha_s = 1$.

Example 8. In the problem of classifying the lung noise data set (66-variate vectors from each of two classes), the training-set was composed of $N = 70 + 70 = 140$ randomly selected vectors, the validation set was composed of $Nv = 40 + 40 = 80$ randomly selected vectors. We utilised the integrated approach to design the linear classification rule and tested 19 CM structurisation models:

Conventional representation	Block diagonal Representation
1. No structurisation	10. No structurisation
2. Circular	11. Circular
3. Toeplitz	12. Toeplitz
4. Tree	13. Tree
5. AR1	14. AR1
6. AR4	15. AR4
7. MA2	16. MA2
8. ARMA11	17. ARMA11
9. Additive noise	18. Additive noise
	19.Markov

Structured sample estimates were regularised using four values of regularisation parameter λ = 0, 0.05, 0.1, 0.2. Thus, M = 76 CM estimates were tested for the linear data transformation (decorrelation and scaling) and subsequent training of the SLP classifier. The validation set was used to stop the SLP training optimally. From 76 validation set estimates, we found that the best model is

the block diagonal representation + the tree dependence model and $\lambda = 0.05$.

with apparent error $\hat{\varepsilon} = 0.0875$. To evaluate the bias $B_{selection}$, the Procedure 1 was initially used. 2-*NN* directed noise was added to the validation set and PGP was obtained. The distribution of vectors $(\hat{\varepsilon}_1^*, \varepsilon_1^*), (\hat{\varepsilon}_2^*, \varepsilon_2^*), ..., (\hat{\varepsilon}_{76}^*, \varepsilon_{76}^*)$ is presented in Figure 6.12*a*.

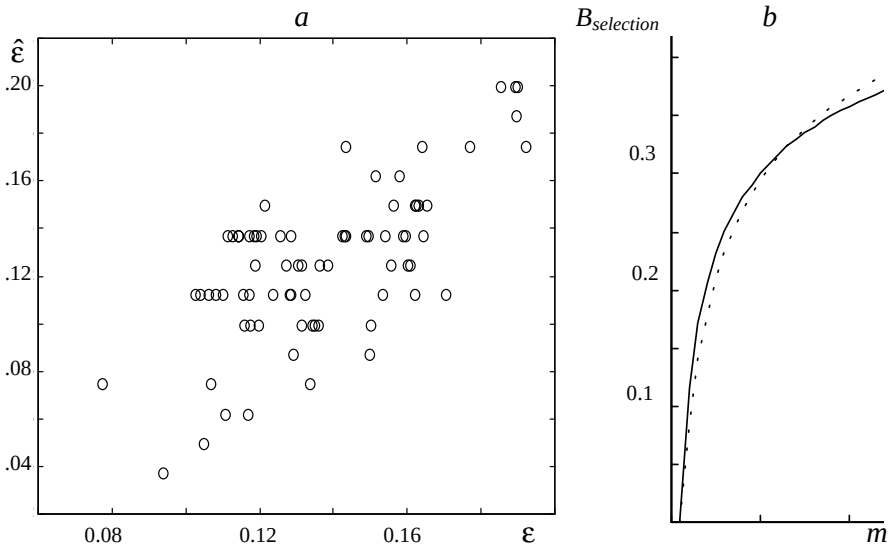

Fig. 6.12. *a*) distribution of the pseudo-validation and the pseudo-test errors, *b)* the bias $B_{selection} = \hat{\hat{\varepsilon}}_{true} - \hat{\hat{\varepsilon}}_{apparent}$ versus the number of models m_{max}: dots – the experimental estimate, solid – the theoretical estimate.

The model selection bias evaluated by utilising the Pikelis algorithm is presented in Figure 6.1*b* (the dotted curve) and the best theoretical fit was obtained for $N^* = 157$ and parameters $\varepsilon_{min} = 0.034$, $\varepsilon_{max} = 0.748$, $\gamma = 14.5$, and $\eta = 86$. As a result, the estimate of the bias was calculated analytically: for $M = 76$ models compared, $B_{selection} \approx 0.045$. Thus, the performance estimate of the model

"the block diagonal representation + the tree dependence model and $\lambda = 0.05$"

$$\hat{\bar{\varepsilon}}_{true} = \hat{\varepsilon} + B_{selection} = 0.0875 + 0.045 \approx 0.12.$$

The rough bias evaluation of Procedure 2 for $Nv = 200$, $\varepsilon_{min} = 0.1$ and the number of variants $M = 100$ resulted in $B_{selection} = 0.044 - 0.024 = 0.02$ (from Table 6.1). The rule of thumb gives $B_{selection} \approx \sqrt{0.12(1-0.12)/80} = 0.035$.

One needs to have in mind that both estimates rely on the assumptions made and can serve only as a *guide in evaluating the selection bias* $B_{selection}$. Nevertheless, the addition of term $B_{selection}$ to $\varepsilon_{apparent} = \min\{\hat{\varepsilon}_1, \hat{\varepsilon}_2, \hat{\varepsilon}_3, \ldots, \hat{\varepsilon}_M\}$ serves as an indicator of the adaptation to the validation set. At the same time one needs to remember that the new estimate $\varepsilon_{apparent} + B_{selection}$ is based on a random validation set. Therefore it remains a random variable. If the error counting estimates are used, its variance can be evaluated from Equations (6.27), (6.31), (6.32) or (6.34).

6.6 Additional Bibliographical Remarks

We have reviewed a number of tools and the performance measures which can be useful in selecting the best model, the feature system, and the architecture of the neural network. In order to select the best model correctly we need to use unbiased performance estimates. In practice, however, all performance estimates are inexact. Therefore, while selecting the best model we adapt to the validation set. Consequently, the validation set begins to serve as an additional training-set. In spite of the fact that we use unbiased estimates to evaluate performance of each competing variant, we obtain additional bias. We called it the bias in model selection.

That the partitioning of the classification error estimation methods depends on: the method used to split the design set into the training and validation sets and the type of pattern error function used to determine the contribution of each observation in the aggregate estimation function was suggested by Raudys and Vaitukaitis (1984). The leave-one-out method was suggested by Brailovskij (1964) and popularised by Lachenbruch and Mickey (1968). The fast calculation schemes for standard linear and quadratic classifiers were proposed by Lachenbruch and Mickey (1968) and Fukunaga and Kessel (1971). The first version of the bootstrap estimation approach belongs to Pinsker (1973) and the bootstrap error estimate was formalised by Efron (1982). The recommendation about choosing the number

of bootstrap samples $r \geq 20$ and the variance of the bootstrap estimate comes from Raudys (1988).

Glick (1978) introduced the smoothed modification of the error counting estimate. Fukunaga and Kessel (1973) suggested another modification known as the posterior probability estimate that allows one to use unlabeled test-set observation vectors. Lachenbruch and Mickey (1968) suggested parametric and the pseudo-parametric classification error estimates $\hat{\varepsilon}_D$, $\hat{\varepsilon}_{DS}$, PP.

Bryant and Guseman (1979) introduced the distance criterion (6.18) for the linear feature extraction. Two non-linear feature mapping techniques based on the criteria (6.19) and (6.20) are due to Sammon (1969) and Niemann (1980). The optimal discriminant set of vectors was suggested by Foley and Sammon (1975).

The saliency was introduced by Le Cun *et al*. (1990). The entropic loss (6.24) was considered in Vajda (1970) and Amari and Murata (1993), while the heuristic measure (6.26) by Yacoub and Bennani (1997). Other examples of the feature performance measures utilised in the ANN design are a relevance of single weights by Mozer and Smolensky (1989), an estimated sensitivity by Karnin (1990), a saliency by Cibas *et al*. (1996). Utilisation of the MLP for linear and non-linear feature extraction was suggested by Bourland and Kamp (1988).

The symmetry between expectations of the expected and resubstitution errors was shown in Raudys (1973), Pivoriunas and Raudys (1978) and Amari and Murata (1993). The estimate (6.33) of the asymptotic PMC based on a symmetry of the expected and resubstitution errors was recommended by Raudys (1973, 1976). For more information about error estimation methods see McLachlan (1992), Toussaint (1974) and Hand (1986). Useful reviews of feature extraction and selection from Young and Calvert (1974, Chapter 6), Gelsema and Eden (1980), McKay and Campbell (1982*ab*), Devijver and Kittler (1982), Siadliecki, Siadlecka and Sklansky (1988), Mao and Jain (1995), Jain and Zongker (1997), Somol *et al*. (1999), Jain, Duin and Mao (2000) are recommended reading.

The optimal dimensionality problem for pattern classification was first analysed by Allays (1966), Hughes (1965) and Lbov (1966). Relations between the optimal dimensionality and the training-set size for the additive noise model (6.37) were considered by Jain and Waller (1978). A procedure to smooth the empirical curve of the estimate of the generalisation error versus the number of features was proposed in Raudys (1979*b*).

Estes (1965) was the first to pay attention to the adaptation bias in feature selection. The model selection bias was analysed in Meshalkin (1977), Raudys (1979*a*, 1981, 1987), Raudys and Pikelis (1982) and Serdobolskij (1983*b*). The algorithm to estimate the bias in the best variant selection is from Pikelis (1991). The apparent, ideal and true errors in selection were introduced in Raudys (1981). For a specific model of distribution of $f(\hat{\varepsilon}, \varepsilon)$ expected values of these types of classification error were obtained and tabulated. The bias evaluation Procedure 1 is from Raudys, Saudargiene and Povilonis (2000).