

Ma 84

ACTA POLYTECHNICA SCANDINAVICA

MATHEMATICS, COMPUTING AND MANAGEMENT IN ENGINEERING SERIES No. 84

Subspace Classifiers in Recognition of Handwritten Digits

JORMA LAAKSONEN

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science
P.O. Box 2200
FIN-02015 HUT
Finland

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Auditorium F1 of Helsinki University of Technology on the 7th of May 1997, at 12 o'clock noon.

ESPOO 1997

Laaksonen, J., **Subspace Classifiers in Recognition of Handwritten Digits.**

Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 84, Espoo 1997, 152 pp. Published by the Finnish Academy of Technology. ISBN 952-5148-20-3. ISSN 1238-9803. UDC 681.327.12:519.2

Keywords

pattern recognition, adaptive systems, neural networks, statistical classification, subspace methods, prototype-based classification, feature extraction, optical character recognition, handwritten digits, classifier comparison, benchmarking study

Abstract

This thesis consists of two parts. The first part reviews the general structure of a pattern recognition system and, in particular, various statistical and neural classification algorithms. The presentation then focuses on subspace classification methods that form a family of semiparametric methods. Several improvements on the traditional subspace classification rule are presented. Most importantly, two new classification techniques, here named the Local Subspace Classifier (LSC) and the Convex Local Subspace Classifier (LSC+), are introduced. These new methods connect the subspace principle to the family of nonparametric prototype-based classifiers and, thus, seek to combine the benefits of both approaches.

The second part addresses the recognition of handwritten digits, which is the case study of this thesis. Special attention is given to feature extraction methods in optical character recognition systems. As a novel contribution, a new method, here named the error-corrective feature extraction, is presented. The prototype recognition system developed for the experiments is described and various options in the implementation are discussed.

For the background of the experiments, thirteen well-known statistical and neural classification algorithms were tested. The results obtained with two traditional subspace methods and ten novel techniques presented in this thesis are compared with them. The results show that the Convex Local Subspace Classifier performs better than any other classification algorithm in the comparison.

The conclusions of this thesis state that the suggested enhancements make the subspace methods very useful for tasks like the recognition of handwritten digits. This result is expected to be applicable in other similar cases of recognizing two-dimensional isolated visual objects.

To my parents

Acknowledgments

Most of all, I wish to acknowledge the guidance of my supervisor, Professor Erkki Oja, in the preparation of this work. He was the first to hear about, and comment on, the innovations that constitute the substance of this thesis. I have been able to rely with confidence on his extensive expertise in the field.

Secondly, I would like to express my gratitude to Academy Professor Teuvo Kohonen, who earlier was my supervisor and who introduced me to the key concepts of the problems involved in this thesis. He has addressed many of the questions discussed in this work, and, in general, his contributions to scientific research in the fields of pattern recognition and neural networks have been immense.

In addition to Professor Erkki Oja, Docent Lasse Holmström, Dr. Petri Koistinen, and Professor Jouko Lampinen have coauthored publications used as the foundation of some parts of this thesis. This collaboration has not only been successful but, also, most pleasant.

Of my colleagues at the Laboratory of Computer and Information Science and at the Neural Networks Research Centre of Helsinki University of Technology, I wish to first thank Lic. Tech. Kimmo Valkealahti. His careful proofreading of my manuscript uncovered many weaknesses which I was then able to correct. Also, I'm very grateful to my long-time colleague and first instructor, Docent Jari Kangas, for many enlightening discussions during the eight years we worked together. For providing an inspiring atmosphere for scientific research, I thank all my co-workers at the laboratory and at the research centre.

This manuscript was reviewed by Associate Professor Pauli Kuosmanen, Tampere University of Technology, and Docent Visa Koivunen, University of Oulu. I hereby express my thanks to them. The text was also read and revised by Dr. Atro Voutilainen, Dr. Pauliina Raento, and Ms Kathleen Tipton. Their outstanding contributions to the readability of the thesis are herewith acknowledged.

Finally, I wish to thank my dear wife Tuula Nissinen, who has shared in the delights and griefs of this research and writing process with me.

Otaniemi, April 18, 1997

A handwritten signature in black ink, appearing to read 'Jouni Lampinen', written in a cursive style.

Contents

Abstract	2
Acknowledgments	5
Contents	7
List of Symbols	11
List of Acronyms	13
1 Introduction	15
1.1 Summary of Novelties and Contributions	16
2 Parts of a Pattern Recognition System	19
2.1 Data Collection	20
2.2 Registration	21
2.3 Preprocessing	21
2.4 Segmentation	22
2.5 Normalization	22
2.6 Feature Extraction	23
2.7 Classification and Clustering	24
2.8 Postprocessing	24
2.9 Feedback Between Stages	25
2.10 Trainable Parts in a System	26

3	Classification Methods in Pattern Recognition	27
3.1	Mathematical Preliminaries	29
3.2	Density Estimation Methods	30
3.2.1	Discriminant Analysis Methods	31
3.2.2	Kernel Discriminant Analysis, Probabilistic Neural Net	32
3.2.3	Reduced Kernel Density Analysis, Radial Basis Functions	33
3.3	Regression Methods	33
3.3.1	Multi-Layer Perceptron	35
3.3.2	Local Linear Regression	35
3.3.3	Tree classifier, MARS and FDA	36
3.4	Prototype Classifiers	37
3.4.1	k -Nearest Neighbor Classifiers	37
3.4.2	Learning Vector Quantization	38
3.4.3	Learning k -NN Classifier	39
3.5	Special Properties of Neural Methods	40
3.6	Cross-Validation in Classifier Design	42
3.7	Rejection	42
3.8	Committees	43
3.9	On Comparing Classifiers	44
3.10	Classifiers: Theory and Practice	45
4	Subspace Classification Methods	47
4.1	Classical Subspace Methods	48
4.1.1	Subspace Basics	48
4.1.2	Classification Rule	50
4.1.3	CLAFIC	50
4.1.4	Multiple Similarity Method	51
4.1.5	Method of Orthogonal Subspaces	51
4.1.6	Generalized Fukunaga-Koontz Method	52

4.2	Basic Learning Subspace Methods	53
4.2.1	Category Feature Subspace Method	53
4.2.2	Learning Subspace Method	54
4.2.3	Averaged Learning Subspace Method	55
4.2.4	Neural Network Interpretation of Subspace Learning	56
4.3	Modifications on Subspace Classifiers	57
4.3.1	Using Class-Specific Means	57
4.3.2	Selection of Subspace Dimensions	59
4.3.3	Weighted Projection Measures	60
4.3.4	Multiple Subspaces per Class	61
4.3.5	Treatment of Neighborhood of Origin	62
4.3.6	Density Function Interpretation of Subspace Methods	62
4.3.7	Computational Aspects	67
4.4	Prospects of Subspace Classifiers	68
5	Local Subspace Classifier	71
5.1	Basic Local Subspace Classifier	71
5.2	LSC+ Classifier	75
5.3	Combining LSC with Prototype Classifiers	76
5.4	Iterative Solution for LSC	78
5.5	Neural Network Interpretation of LSC	79
6	Survey of Off-line Recognition of Handwriting	83
6.1	History	83
6.2	Application Areas	84
6.3	Fields of Character Recognition	85
6.4	Feature Extraction in Handwriting Recognition	87
6.4.1	Reconstruction from Features	90
6.4.2	Template Matching	90

6.4.3	Volume Features	91
6.4.4	Outline Features	99
6.4.5	Skeleton Features	102
6.4.6	Discrete Features	104
6.4.7	Zoning: Combination of Low-Level Features	105
6.4.8	Error-Corrective Feature Extraction in OCR	106
6.5	OCR: From Parts to a Whole	108
7	Prototype Recognition System	111
7.1	Image Acquisition	111
7.2	Registration of Images	113
7.3	Digit Segmentation	113
7.4	Normalization of Digits	114
7.5	Feature Extraction	115
7.6	Data Sets	115
8	Comparison of Classification Methods	117
8.1	CLAFIC- μ and ALSM- μ	118
8.2	Selection of Subspace Dimensions	118
8.3	Weighted Subspace Projection Measures	120
8.4	Probability Density Function Interpretation	121
8.5	Local Subspace Classifier	122
8.6	Error-Corrective Feature Extraction	123
8.7	Learning k -NN Classifier	125
8.8	Summary of Classification Results	127
9	Conclusions	131
	Bibliography	133

List of Symbols

$g(\cdot)$	classification function	29
d	dimensionality of feature data	29
$\mathbf{x}$	input feature vector	29
c	number of classes	29
j	class index of vector $\mathbf{x}$ in $1, \dots, c$	29
n	number of vectors in the training set	29
n_j	number of vectors belonging to the class j in the training set	29
$P_j, \hat{P}_j$	<i>a priori</i> probability of class j , its sample estimate	29
$f_j(\mathbf{x})$	probability density function of class j	29
$f(\mathbf{x})$	probability density function of the pooled data	29
$q_j(\mathbf{x})$	<i>a posteriori</i> probability of class j given vector $\mathbf{x}$	29
$\epsilon(\mathbf{x})$	probability of vector $\mathbf{x}$ to be misclassified	30
ϵ	overall misclassification rate	30
$\boldsymbol{\mu}_j, \hat{\boldsymbol{\mu}}_j$	mean of vectors $\mathbf{x}$ in class j , its sample estimate	31
$\hat{\boldsymbol{\mu}}$	sample estimate of the mean of training data	51
$\hat{\boldsymbol{\mu}}_{\mathbf{f}}, \hat{\boldsymbol{\Sigma}}_{\mathbf{f}}$	sample mean and autocovariance of raw image data	93
$\boldsymbol{\Sigma}_j, \hat{\boldsymbol{\Sigma}}_j$	autocovariance matrix of vectors $\mathbf{x}$ in class j , its sample estimate	31
$\boldsymbol{\Sigma}, \hat{\boldsymbol{\Sigma}}$	autocovariance matrix of vectors $\mathbf{x}$, its sample estimate	31
λ, γ	regularization parameters in RDA	31
λ	weight decay parameter in MLP+WD	35
λ_{ij}	i th eigenvalue of $\hat{\mathbf{R}}_j$	50
h_j, γ	smoothing parameters in KDA	32
$\gamma(t)$	correction coefficient in error-corrective feature extraction	107
$\mathbf{I}$	identity matrix	31
$K(\mathbf{x})$	kernel probability density function	32
$\mathbf{m}_i$	prototype vector in a SOM	33

$\mathbf{m}_{ij}, \mathbf{M}_j$	used prototypes of class j in LSC, matrix of used prototypes	74
ℓ	number of vectors in a SOM	33
ℓ	number of hidden units in MLP	35
ℓ	number of prototypes in an LVQ codebook	38
ℓ	number of vectors in L- k -NN	39
ℓ	subspace dimension	48
w_i	kernel weight in RKDA	33
w	“window” width in LVQ	38
w, h	width and height of normalized input image	90
$\mathbf{y}_i$	desired output for pattern $\mathbf{x}_i$ in regression methods	34
$\mathbf{r}(\mathbf{x})$	response to pattern $\mathbf{x}$ in regression methods	34
α	learning rate parameter in delta rule	38
α, β	scatter matrix update coefficients in ALSM	55
α, β	multipliers of matrices in error-corrective feature extraction	107
ρ	overall rejection rate	43
θ	rejection parameter	43
ρ, θ	polar coordinates	94
$\mathcal{L}$	subspace	48
$\mathbf{u}_i, \mathbf{U}$	subspace basis vector, matrix of subspace basis vectors	48
$\zeta_i, \mathbf{z}$	subspace coefficient, vector of subspace coefficients	48
$\mathbf{P}$	projection matrix	49
$\mathbf{R}_j, \widehat{\mathbf{R}}_j$	autocorrelation matrix of vectors $\mathbf{x}$ in class j , its sample estimate .	50
$\mathbf{0}$	zero vector or matrix	51
$\mathbf{G}_j$	generating matrix	52
D, D_j	subspace dimension and subspace dimension of class j in LSC . . .	72
$\widehat{\mathbf{S}}_j(t)$	scatter matrix in ALSM	55
$\mathbf{S}(t)$	scatter matrix in error-corrective feature extraction	107
$c_{ij}, \mathbf{c}_j$	prototype coefficient of class j in LSC, vector of coefficients	74
$f(x, y), \mathbf{f}$	normalized input image function and vector	90
$t_j(x, y), \mathbf{t}_j$	image template function and vector	90
$k_i(x, y), \mathbf{k}_i$	image mask function and vector	92
$\mathbf{K}$	image mask matrix	92
$\widehat{\mathbf{A}}_{\mathbf{f}}, \widehat{\mathbf{B}}_{\mathbf{f}}$	covariances of interclass direction and misclassification direction in error-corrective feature extraction	106

List of Acronyms

AIC	Akaike Information Criteria	59
ALSM	Averaged Learning Subspace Method	55
CAFESSM	Category Feature Subspace Method	53
CART	Classification and Regression Trees	34
CCA	Curvilinear Component Analysis	77
CLAFIC	Class-Featuring Information Compression	50
CLT	Central Limit Theorem	64
DCT	Discrete Cosine Transform	94
DFT	Discrete Fourier Transform	94
EM	Expectation-Maximization	30
FDA	Flexible Discriminant Analysis	37
GAM	Generalized Additive Models	34
GFK	Generalized Fukunaga-Koontz Method	52
GRNN	General Regression Neural Network	34
HMM	Hidden Markov Model	86
ICA	Independent Component Analysis	61
KDA	Kernel Discriminant Analysis	32
KLT	Discrete Karhunen-Loève Transform	92
k -NN	k -Nearest Neighbor	37
LDA	Linear Discriminant Analysis	31
L- k -NN	Learning k -NN Classifier	39
LLR	Local Linear Regression	35

LPT	Log-Polar Transform	94
LSC	Local Subspace Classifier	71
LSC+	Convex Local Subspace Classifier	75
LSM	Learning Subspace Method	54
LSSOM	Local Subspace SOM	76
LVQ	Learning Vector Quantization	38
MARS	Multivariate Adaptive Regression Splines	36
MAT	Medial Axis Transform	102
MDL	Minimum Description Length	59
MLP	Multi-Layer Perceptron	35
MLP+WD	Multi-Layer Perceptron with weight decay regularization	35
MOSS	Method of Orthogonal Subspaces	51
MSM	Multiple Similarity Method	51
OCR	Optical Character Recognition	85
PCA	Principal Component Analysis	50
PNN	Probabilistic Neural Network	32
PP	Projection Pursuit	34
QDA	Quadratic Discriminant Analysis	31
RBF	Radial Basis Function	33
RDA	Regularized Discriminant Analysis	31
RKDA	Reduced Kernel Discriminant Analysis	33
SDF	Similar Discriminant Function	98
SIMCA	Soft Independent Modelling of Class Analogy	58
SIMD	Single Instruction Multiple Data	67
SOM	Self-Organizing Map	76
SVD	Singular Value Decomposition	74
WVQ	Weighted Vector Quantization	77

Chapter 1

Introduction

Pattern recognition – or, more generally, artificial perception – is a research field with an increasing number of application areas and enduring scientific attention. In the process of pattern recognition, a technical device assigns the visual or other input patterns it receives to predefined classes. The science and art of pattern recognition is in developing the automatic systems capable of performing this challenging task.

An operational pattern recognition system consists of a series of processing modules, of which the feature extraction and classification stages are the most crucial for its overall performance. The feature extraction methods are generally specific to each particular pattern recognition problem, whereas the same classification algorithms can be utilized in various applications. Numerous classification algorithms have been proposed during the past four decades of development in automated recognition. Moreover, different taxonomies have been presented to explain the similarities and differences of the classification methods.

As the efficiency of data processors used in implementing pattern classification algorithms has increased, new, computationally more demanding methods have been studied. In addition, it is now practicable to store larger models or more prototypes in the memory of the recognition system. Therefore, some old classification algorithms, previously considered too complex, may have turned out to be realizable due to technological development. During the last few years, neural network methods have been applied successfully in many fields of pattern recognition. In some cases, major improvements in classification accuracy have been achieved. The neural classification algorithms are not, however, ready-made solutions for all the problems of artificial perception. In all applications, the preprocessing and feature extraction stages must also be carefully designed.

The purpose of this thesis is to show that the subspace classification methods are very suitable for the recognition of handwritten digits. The text offers some new modifications to the traditional subspace classification rule. In addition, some new classification techniques are introduced. The usefulness of these modifications and new techniques is demonstrated with experiments.

The organization of this thesis is as follows. An overview of the parts forming a general pattern recognition system is presented in Chapter 2. The importance of feature extraction and classifying stages is emphasized. Attention is focused on the classification methods. Various types of neural and traditional statistical classifiers are reviewed in Chapter 3. A novel taxonomy of classification methods is suggested. The subspace methods of classification are examined intensively in Chapter 4. Many improvements upon the original subspace classification algorithm are offered. Most importantly, a new method, here named the Local Subspace Classifier (LSC), and its variant, named the Convex Local Subspace Classifier (LSC+), are introduced in Chapter 5.

The recognition of handwritten digits serves as the case study of this thesis. Chapter 6 describes this research field as a subfield of optical character recognition. An extensive review of applicable feature extraction methods is presented in Section 6.4. The implementation of the prototype of the handwritten digit recognition system is described in Chapter 7. A large-scale comparison of classification methods has been carried out, and its results are examined in Chapter 8. The newly introduced Convex Local Subspace Classifier shows superior classification performance in these experiments. Chapter 9 concludes the central questions and observations of the previous chapters. It argues that the proposed modifications on the subspace classification principle are beneficial and that, in general, the subspace methods should be applied more extensively to various classification problems.

1.1 Summary of Novelties and Contributions

Due to the monographic form of this thesis, novel ideas are scattered throughout the text. Therefore, this section emphasizes the novelties and lists the forums where the ideas have been or will be presented. My own contributions in the coauthored publications used as the basis of this study are explained. The publications are numbered [1]-[8] at the end of this section.

The taxonomy used in describing the classification methods in Chapter 3 was presented in Publications [1] and [2]. These publications also contained the results of a large comparison of statistical and neural classification methods. I contributed to the formulation of the taxonomy and to the design of the experiments. As well, I developed the prototype recognition system for obtaining the data and performed the experiments with the subspace and prototype classifiers. Publication [2] appeared as the leading article in a special issue of *IEEE Transactions on Neural Networks* concentrating on pattern recognition applications of neural networks. Of the two case studies, only the results of the handwritten digit recognition are repeated in this text.

The novel idea of the Learning k -Nearest Neighbors classification method described in Section 3.4.3 and Publication [3] was introduced by me. The method is a simple but powerful modification of the classical technique and offers improved classification accuracy in situations where the number of prototypes that can be stored is limited. Although this

method does not seem to belong under the title of this thesis, it is closely related to the Local Subspace Classifier scheme introduced in this thesis, and is therefore included.

The novel improvements to subspace classifiers described in Section 4.3.2 and Section 4.3.3 were developed by me and were presented in Publication [4]. These modifications change the subspace classification rule slightly and produce better classification accuracy and increased robustness to small variations in the subspace dimensions. The original experiments were extended for this thesis, and cross-validation was now used in selecting the appropriate parameter values.

The text used in the general overview of a pattern recognition system in Chapter 2 was solely written by me. It will appear as a part of Publication [5] as an invited contribution to a volume in *Control and Dynamic Systems: Advances in Theory and Applications*, edited by Prof. Cornelius T. Leondes and published by the Academic Press. The text serves as a tutorial and contains no specific scientific novelties.

The concept of the error-corrective feature extraction described in Sections 2.9 and 6.4.8 was introduced by me. It will be published as Publication [6]. The proposed scheme makes the feature extraction stage adaptive and autonomously error-corrective. The classification accuracy is therefore increased.

The novel probability density function interpretation of the subspace classification rule in Section 4.3.6 was given by me. It will be published as Publication [7]. The interpretation explains the operation of a subspace classifier in a new way. Also, it provides a statistically justified rejection mechanism.

The Local Subspace Classifier, its convex version, and extension to the Local Subspace Self-Organizing Map in Chapter 5 were solely presented by me. These innovations will be published as Publication [8]. The proposed classification methods are in this thesis shown to perform better than any other classifier in recognition of handwritten digits.

List of Publications

- [1] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja (1996). Neural Network and Statistical Perspectives of Classification. In *Proceedings of 13th International Conference on Pattern Recognition*, Vienna, Austria, pp. 286–290.
- [2] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja (1997). Neural and Statistical Classifiers – Taxonomy and Two Case Studies. *IEEE Transactions on Neural Networks* 8(1), 5–17.
- [3] J. Laaksonen and E. Oja (1996). Classification with Learning k -Nearest Neighbors. In *Proceedings of the International Conference on Neural Networks*, Volume 3, Washington D.C., pp. 1480–1483.
- [4] J. Laaksonen and E. Oja (1996). Subspace Dimension Selection and Averaged Learning Subspace Method in Handwritten Digit Classification. In *Proceedings of*

the International Conference on Artificial Neural Networks, Bochum, Germany, pp. 227–232.

- [5] J. Lampinen, J. Laaksonen and E. Oja (1997). Neural Network Systems, Techniques and Applications in Pattern Recognition. To appear in *Control and Dynamic Systems: Advances in Theory and Applications*, edited by Cornelius T. Leondes, Academic Press, New York.
- [6] J. Laaksonen and E. Oja (1997). Error-Corrective Feature Extraction in Handwritten Digit Recognition. To appear in *Proceedings of the International Conference on Engineering Applications of Neural Networks*, Stockholm, Sweden.
- [7] J. Laaksonen and E. Oja (1997). Density Function Interpretation of Subspace Classification Methods. To appear in *Proceedings of the Scandinavian Conference on Image Analysis*, Lappeenranta, Finland.
- [8] J. Laaksonen (1997). Local Subspace Classifier and Local Subspace SOM. To appear in *Proceedings of the Workshop on Self-Organizing Maps*, Otaniemi, Finland.

Chapter 2

Parts of a Pattern Recognition System

This thesis focuses on issues of *pattern recognition*. This key term can be defined in many ways, including

Pattern recognition is an information-reduction process: the assignment of visual or logical patterns to classes based on the features of these patterns and their relationships. (Britannica Online 1996)

The basic setting, observed from the point of view of classification, is as follows. There is one unknown object presented as a set of signals or measurements at the input of a black box called a pattern recognition system. At the output of the system, there is a set of predefined classes. The task of the system is to assign the object to one of the classes. In a more general setting, there is more than one object to be recognized. In this case, the classification of the nearby objects may or may not be interdependent. The list of classes may also contain a special reject class for the objects the system is unable to classify.

Depending on the measurements and the classes, we are led to divergent fields of pattern recognition. These include speech or speaker recognition, detection of clinical malformations in medical images or time-signals, document analysis and recognition, etc. All these disciplines call for expertise both in the subject matter and in the general theory and practice of pattern recognition. There exists extensive literature on both the overall and specific questions of pattern recognition systems and applications. The classical textbook sources include Andrews (1972), Duda and Hart (1973), Tou and Gonzalez (1974), Young and Calvert (1974), Gonzalez and Thomason (1978), Sklansky and Wassel (1981), De-
vijver and Kittler (1982), Fu (1982), Fukunaga (1990), and Bow (1992), some of which are actually revised versions of earlier editions. Recent developments, such as the use of neural methods, are examined, for instance, in books by Pao (1989), Therrien (1989), Schalkoff (1992), Pavel (1993), Bishop (1995), and Ripley (1996). During the past 30 years, many valuable article collections have been edited. These include Kittler et al. (1982), Kanal (1981), Kanal (1985), Dasarathy (1991), and Chen et al. (1993).

Technical systems are often considered as being comprised of consecutive modules, each performing its precisely defined task in the process. The whole system can then be modeled in bottom-up fashion as a block diagram. In the simplest case, the flow of the data stream is one-directional from left to right as shown in Figure 2.1, presenting a general pattern recognition system. The diagram shown is naturally only one intuition of how to depict a view, and alternative structures are given, for example, by Fu (1982), Fu and Rosenfeld (1984), Bow (1992), and by Schalkoff (1992).

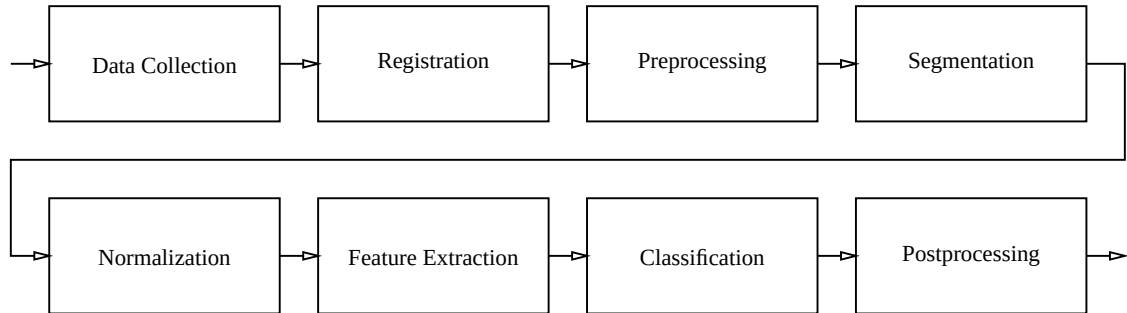


Figure 2.1: A block diagram of a generic pattern recognition system.

The following sections shortly describe each of the stages from data collection to post-processing. Some issues are clarified with examples principally from optical character recognition, which will be represented as the primary case study of this thesis in Chapters 6-8, and from speech recognition. Thus, some of the described stages may not have obvious counterparts in other types of pattern recognition systems.

2.1 Data Collection

The first stage in any pattern recognition system is data collection. Before a pattern vector is formed out of a set of measurements, these measurements need to be performed using some sensors and be converted to a numerical form. In the cases of image analysis or character recognition, such equipment include video cameras and scanners; in the case of speech recognition, microphones, etc. The input data is sampled in time and/or space domain and digitized to be represented by a preset number of bits per measurement. The data collection devices should record the objects with adequate fidelity and reasonable price, as any additional noise may impair the operation of the system. The data collection phase should also be designed in such a manner that the system is robust enough to adapt to the variations in operation of individual signal measurement devices. If the measured phenomenon itself is time-varying, the data collection should tolerate that.

The data collection stage may include auxiliary storage for the collected data. The use of temporary storage is necessary if recognition cannot be performed simultaneously with data acquisition. Larger mass storage for training data is needed while a pattern recognition system is constructed. On some occasions, the size of data storage needed may

turn out to be a prohibitive factor in the development or use of a pattern recognition system. This discrepancy can be somewhat eased by compressing the stored data, but, in the worst case, the fidelity of data representation has to be sacrificed for the sake of storage shortage. The difficulty is often solved either by reducing the spatial or temporal resolution of the data sampling or by representing the measurements with a lower precision using fewer bits per sample. Similar problems and solutions arise if the channel used in transferring the data forms a bottleneck for the requirements of on-line processing.

2.2 Registration

In the registration of data, rudimentary model fitting is performed. The internal coordinates of the recognition system are mapped to the actual data acquired. At least some *a priori* knowledge about the world surrounding the system is used in designing the registration stage. This external information mainly answers questions such as: “How has the data been produced?” and “Where or when does sensible input begin and end?” The registration process thus defines the operative framework for the system. In this way, the system knows what to expect as valid input.

In the registration block of a speech recognition system, the epochs during which input is comprised solely of noise and/or silence are detected and discarded from further processing. Then, the beginnings and endings of the utterances are located and marked. In optical character recognition, the system must locate the area of interest in the input image. In the case of fill-in forms, the area may be registered with some special printed marks, but in document analysis the system has to locate it automatically, based upon the overall layout of the page image.

2.3 Preprocessing

Real-world input data always contains some amount of noise, and certain preprocessing is needed to reduce its effect. The term “noise” is to be understood in the wide sense: anything that hinders a pattern recognition system from fulfilling its task may be regarded as noise, no matter how inherent it is in the data. Some desirable properties of the data may also be enhanced with preprocessing before the data is passed on further in the recognition system.

Preprocessing is normally accomplished by some simple filtering of the data. In the case of speech recognition, this may mean linear high-pass filtering aimed at removing the base frequency and enhancing the higher frequencies. In image recognition, the image may be median filtered to remove spurious impulsive noise which might hamper the segmentation process. Decorrelation of the color components is a usually advantageous preprocessing step for color images. Such a process transforms an image from the RGB (red-green-blue) coordinates, for instance, linearly to the YIQ (luminosity-inphase-quadrature) system.

2.4 Segmentation

The registered and preprocessed input data has to be subdivided into parts to create meaningful entities for classification. This stage of processing is called segmentation. It may either be a clearly separate process or tightly coupled with the previous or following processes. In either case, after the pattern recognition system has completed the processing of a block of data, the resulting segmentation of the data into its subparts can be revealed. Depending on the application, the segmentation block may either add the information regarding the segment boundaries to the data flow, or, alternatively, copy all the segments in separate buffers and forward them to the following stage one by one.

In speech recognition, a meaningful entity is most likely a single phoneme or a syllable that contains a small but varying number of phonemes. In optical character recognition, the basic units for classification are single characters or some of the few composite glyphs, such as ‘fi’ and ‘fl’.

Some pattern recognition applications would be described better if segmentation were placed after the classification stage in Figure 2.1. In such systems, the input data is partitioned with fixed-sized windows at fixed spatial or temporal intervals. The actual segmentation can take place only after the subparts have been labeled in the classification stage. The cooperation between the stages of a pattern recognition system is discussed further in Section 2.9.

2.5 Normalization

A profound common characteristic of the environments where automated pattern recognition systems are used is the inherent variance of the objects to be recognized. Without this variance, the pattern recognition problem would not exist at all. Instead, we would be concerned with deterministic algorithms, such as for sorting, searching, computer language compiling, Fourier transform, etc. Therefore, the central question in pattern recognition is how these variances can be accounted for. One possibility is to use feature extraction or classification algorithms which are *invariant* to variations in the outcomes of objects. For example, image features invariant to rotation are easy to define. But inevitably, some types of natural variance will always evade the invariant feature extraction. Therefore, a separate normalization step is called for in almost all pattern recognition systems.

We may think that the objects we perceive pre-exist in a Platonic world of ideas before they materialize as patterns we are bound to measure and deal with. In this process, the patterns are somehow disturbed. In a technical setting, normalization is then a process which tries to revert the measured object back to its ideal form. Unfortunately, the actual form and amount of the disturbances are not known. Therefore, there is no direct way to restore the undisturbed object. We may, however, suppose that the ideal objects follow some *a priori* rules we have established and, then, make the actual patterns obey the same rules wishing that the amount of distortion will somewhat diminish.

As a side effect, normalization always causes loss of degrees of freedom when discrete presentation of the objects is employed. This is reflected as dimension reduction in the intrinsic dimensionality of the data. If the normalization could be performed ideally, only the dimensionality increase caused by the disturbances and other noise would be canceled out. This is unfortunately not true, but, as will be explained in the following section, the dimensionality of the data has to be reduced nevertheless. Therefore, the reduction of the intrinsic dimensionality of the data during the normalization process is not a serious problem if the normalization is successful.

For example, depending on individual habit, handwriting may not stand straight upwards but is somewhat slanted to the left or to the right. Ideal, or at least normalized, characters can be achieved by estimating the slant and reverting it. In speech recognition, the loudness of speech can be normalized to a constant level by calculating the energy of an utterance and then scaling the waveform accordingly.

2.6 Feature Extraction

The meaning of the feature extraction phase is most conveniently defined by referring to its purpose:

Feature extraction problem ... is that of extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the *within-class* pattern variability while enhancing the *between-class* pattern variability. (Devijver and Kittler 1982)

During the feature extraction process, the dimensionality of data is reduced. This is almost always necessary, due to the technical limits in memory and computation time. A good feature extraction scheme should maintain and enhance those features of the input data which make distinct pattern classes separate from each other. At the same time, the system should be immune to variations produced both by the users of the system and by the technical devices used in the data acquisition stage.

Besides savings in memory and time consumption, there exists another important reason for proper dimensionality reduction in the feature extraction phase. It is due to the phenomenon known as the *curse of dimensionality* (Bellman 1961). The curse of dimensionality states that increasing the dimensionality of the feature space first enhances classification accuracy, but rapidly leads to sparseness of the training data and to poor representation of the vector densities, and thereby decreases classification performance. This happens even though the amount of information present in the data is enriched while its dimensionality is increased. The system designer is therefore forced to seek a balance between the dimensionality of the data with the number of training vectors per unit cube in the feature vector space. A classical rule-of-thumb says that the number of training vectors per class should be at least 5–10 times the dimensionality (Jain and Chandrasekaran 1982).

An issue connected to feature extraction is the *choice of metric*. The most natural and commonly-used metric in pattern recognition algorithms is the Euclidean metric. Other metrics have occasionally also been used for more or less intuitively-driven motivations. The variances of individual features may in some applications differ by orders of magnitude, which is likely to impair the classifier. With the Euclidean metric, the situation can be eased by applying a suitable linear transform to the components of the feature vector.

The diverse possibilities for feature extraction in recognition of handwritten characters are discussed in-depth in Section 6.4. In speech recognition, the features are usually based on the assumption that the speech waveform is momentarily stable. In that case, spectral, cepstral, or linear prediction coefficients can be used as descriptors.

2.7 Classification and Clustering

Together with feature extraction, the most crucial step in the process of pattern recognition is classification. All the preceding stages should be designed and tuned for improving its success. The operation of the classification phase can be simplified as being a transform of quantitative input data to qualitative output information. The output of the classifier may either be a discrete selection of one of the predefined classes, or a real-valued vector expressing the likelihood values for the assumption that the pattern originated from the corresponding class.

The primary division of the various classification algorithms used is that of *syntactic* and *statistical* methods. The statistical methods and neural networks are related in the sense that both can, in general, use the same features. Due to the centrality of the topic to this thesis, classification is not covered in this introductory chapter but analyzed comprehensively in Chapter 3.

A topic closely related to classification but outside the scope of this thesis is clustering. In clustering, either the existence of predefined pattern classes is not assumed, the actual number of classes is unknown, or the class memberships of the vectors are generally unknown. Therefore, the purpose of the clustering process is to group the feature vectors to clusters in which the resemblance of the patterns is stronger than between the clusters (Hartigan 1975). The processing blocks surrounding the classification stage in Figure 2.1 are, in general, also applicable to clustering problems.

2.8 Postprocessing

In most pattern recognition systems, some data processing is also performed after the classification stage. These postprocessing subroutines, like the normalization processes, bring some *a priori* information about the surrounding world into the system. This additional expertise can be utilized in improving the overall classification accuracy. A complete postprocessing block may itself be a hybrid of successive and/or cooperative

modules. In the context of this representation, however, it is sufficient to regard the postprocessor as an atomic operator.

The postprocessing phase is generally feasible if the individual objects or segments together form meaningful entities, such as bank account numbers, words, or sentences. The soundness or existence of these higher level objects can be examined and, if an error is found, further steps can be taken to correct the misclassification. The postprocessing phase thus resolves interdependencies between individual classifications. This is either practicable by the operation of the postprocessing stage alone or in conjunction with the segmentation and classification blocks, as the following section explains.

2.9 Feedback Between Stages

Figure 2.1 depicted a block diagram of an idealized pattern recognition application. Such systems, where the data flows exclusively from the left to the right, can hardly ever be optimal in recognition accuracy. By making the successive blocks interact, the overall performance of the system can be considerably enhanced. The system, of course, becomes much more complex, but, generally, it is the only way to increase classification accuracy.

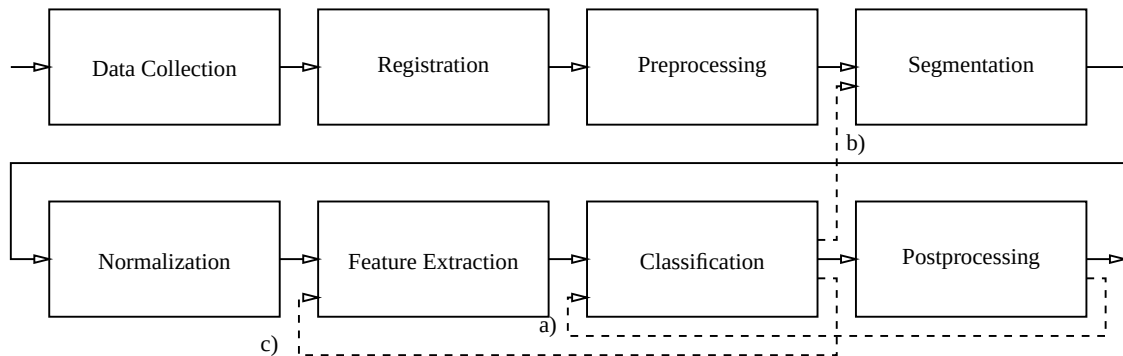


Figure 2.2: A block diagram of a pattern recognition system with some potential feedback routes.

Figure 2.2 displays three potential routes for the backward links with dashed arrows and labels a), b), and c). The motivations for these three configurations are

- a) Information is fed back from postprocessing to classification. When the postprocessor detects an impossible or highly improbable combination of outputs from the classifier, it notifies the classifier. Either the postprocessor itself is able to correct the fault, or it asks the classifier to try again. In either case, the classifier ought to be able to revise its behavior and not repeat similar errors. The classifier may also mediate this feedback information back to the segmentation block, as will be explained below.

- b) The classifier revises the segmentation phase. In this case, the classifier or the postprocessor has detected one or more successive patterns that are hard to classify. This might be an indication of erroneous segmentation, which should be located and corrected. This scheme can also be viewed as a segmentation algorithm that probes the succeeding stages with tentative segments. It is then the responsibility of the classifier to select the most probable combination.

In this scheme, it is also possible that segmentation is performed after classification. The data flows unmodified in its first pass through the segmentation block. After classification, the data is fed back to the segmenter for the actual segmentation and then unmodified through the system to the postprocessing stage.

- c) The correctness of the classifications is used to revise the feature extractor. This kind of operation is usually practicable only during the training phase, and generally necessitates the re-design of the classifier. This kind of scheme, exemplified in Section 6.4.8, may be called *error-corrective feature extraction*.

2.10 Trainable Parts in a System

All the stages of a pattern recognition system contain parameters or variables which need to be given appropriate values. Some of these parameters are so delicate that they have to be selected by an expert in the application field and kept constant thereafter. Others may be tunable by trial and error or cross-validation process in cooperation with an expert observing the overall performance of the system top-down. Profoundly more interesting, however, are parameters which the system is able to learn autonomously from training with available data. Neural networks provide a whole new family of diverse formalisms for adaptive systems. Error-corrective neural training can be used in various parts of a pattern recognition system to improve the overall performance.

In most cases, the adaptive nature of the neural networks is only utilized during the training phase and the values of the free parameters are fixed at its end. A long-term goal, however, is to develop neural systems which retain their ability to adapt to a slowly evolving operation environment. In such automata, the learning of the system would continue automatically and by itself endlessly. However, the stability of such systems is in doubt.

In many systems claimed to be neural, a traditional classifier has only been replaced by a neural solution. This is, of course, reasonable if it improves the performance of the system. Nevertheless, a more principled shift to a completely neural solution might be feasible and motivated. At least the normalization and feature extraction stages, together with classification, could be replaced with neural counterparts in many systems. Only then would the full potential of neural systems be utilized.

Chapter 3

Classification Methods in Pattern Recognition

Numerous taxonomies for classification methods in pattern recognition have been presented, but none has gained an uncontested status. The most fundamental dichotomy, however, is quite undisputed and divides *statistical* and *syntactic* classifiers. The attention of this thesis is limited to the former, whereas the latter – also known as *linguistic* or *structural* approaches – are treated in many textbooks, including the works of Gonzalez and Thomason (1978), Fu (1982), and Pavel (1993).

The statistical, or *decision theoretic*, methods can be divided further in many ways depending on the properties one wants to emphasize. The contradiction of *parametric* versus *nonparametric* methods is an often-used dichotomy. In parametric methods, a specific functional form is assumed for the feature vector densities, whereas nonparametric methods refer directly to the available exemplary data. Somewhere between these extremes, there are *semiparametric* methods which try to achieve the best of both worlds by using adaptable parameters the number of which is restricted and depends on the inherent complexity of the data (Bishop 1995). The distinction between the nonparametric and semiparametric methods is quite vague. In general, the computational and memory requirements of the latter increase more slowly when the amount of training material is increased.

One commonly-stated division (e.g., Schalkoff 1992) separates *neural* and classical *statistical* methods. It is useful only if one wants to regard these two approaches as totally disjointed competing alternatives. At the opposite extreme, neural methods are seen only as iterative ways to arrive at the classical results of the traditional statistical methods (e.g., Ripley 1996). Better still, both methods can be described by using common terms, as has been done by Holmström et al. (1996a) and summarized in this text.

Neural methods may additionally be characterized by their learning process: *supervised* learning algorithms require all the exemplary data to be classified before the training phase begins, whereas *unsupervised* algorithms may use unlabeled data as well. Due to

the general nature of classification, primarily only supervised methods are applicable. For clustering, data mining, and feature extraction, the unsupervised methods can be beneficial as well, or even the only choice.

Above, the pattern recognition problem has been approached from the viewpoint of mathematical theory. Instead, the perspective of the user of a hypothetical pattern recognition system produces a totally different series of dichotomies. Figure 3.1 depicts one such taxonomy, originally presented by Jain and Mao (1994).

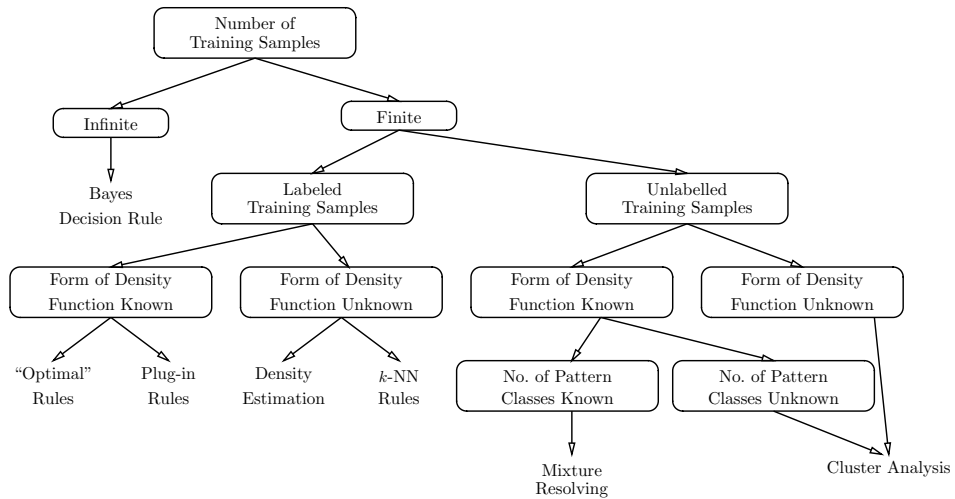


Figure 3.1: Dichotomies in the design of a statistical pattern recognition system, adapted from Jain and Mao (1994).

The following sections describe shortly some classification algorithms according to the taxonomy presented in Table 3.1. The methods are, in the first place, divided to density estimators, regression methods, and others. Each of these groups is examined in a dedicated section. The parametric, semiparametric, or nonparametric nature of each method is discussed in the text. Section 3.5 addresses the neural characteristics of the various classification methods. In Table 3.1, the algorithms considered neural are printed in italics.

	density estimators	regression methods	others
parametric	QDA LDA RDA		
semiparametric	<i>RKDA</i>	<i>MLP RBF</i>	CLAFIC <i>ALSM</i>
nonparametric	KDA <i>PNN</i>	LLR <i>MARS</i>	<i>k</i> -NN <i>LVQ</i> <i>L-k-NN</i>

Table 3.1: Taxonomy of classification algorithms presented in the following sections and used in experiments. Algorithms considered neural are printed in italics.

After the introduction of the central mathematical notations in the next section, the classification methods included in the first column of Table 3.1 will be described in Section 3.2. Some regression methods will then be presented in Section 3.3. The category of “others” in Table 3.1 is composed of nonparametric prototype classifiers and semiparametric subspace classifiers. The former are described in Section 3.4, and the latter are considered in-depth in Chapter 4.

3.1 Mathematical Preliminaries

A central mathematical notation in the theory of classifiers is the *classification function* $g : \mathbb{R}^d \mapsto \{1, \dots, c\}$. Thus, for each real-valued d -dimensional *input feature vector* $\mathbf{x}$ to be classified, the value of $g(\mathbf{x})$ is an integer in the range of $1, \dots, c$, c being the number of classes. The classes are indexed with j when appropriate. The training set used in designing a classifier consists of n column vectors $\mathbf{x}_i$, $i = 1, \dots, n$, of which n_j vectors $\mathbf{x}_{ij}$, $i = 1, \dots, n_j$ belong to class j . The ordered pair $(\mathbf{x}_i, j_i)$ is, stochastically speaking, one realization of $(\mathbf{X}, J)$, an ordered pair of a vector random variable $\mathbf{X}$ and a discrete-valued random variable J . By assuming the realizations $(\mathbf{x}_i, j_i)$ in both the training and testing samples to be independent and identically distributed, many considerations are notably simplified. Taking into account context-dependent information, however, might be beneficial in many applications.

The *a priori* probability of class j is denoted by P_j , its probability density function by $f_j(\mathbf{x})$, and that of the pooled data, with all the classes combined, by $f(\mathbf{x}) = \sum_{j=1}^c f_j(\mathbf{x})$. Naturally, the priors have to meet the condition $\sum_{j=1}^c P_j = 1$. When using this notation, the Bayes classifier that minimizes the non-weighted misclassification error (see Devijver and Kittler 1982) is defined as the one returning the index j of the largest $P_j f_j(\mathbf{x})$,

$$g_{\text{BAYES}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} P_j f_j(\mathbf{x}) . \quad (3.1)$$

An equivalent formulation is to consider the *a posteriori* probability $q_j(\mathbf{x}) = P(J = j \mid \mathbf{X} = \mathbf{x})$ of class j , given $\mathbf{x}$, and use the rule

$$g_{\text{BAYES}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} q_j(\mathbf{x}) . \quad (3.2)$$

The rules (3.1) and (3.2) are equivalent since

$$q_j(\mathbf{x}) = P(J = j \mid \mathbf{X} = \mathbf{x}) = \frac{P_j f_j(\mathbf{x})}{f(\mathbf{x})} . \quad (3.3)$$

In practice, however, the classifiers (3.1) and (3.2) have to be estimated from the training data $(\mathbf{x}_1, j_1), \dots, (\mathbf{x}_n, j_n)$ of pattern vectors with known classes. Then, two distinct approaches emerge. The use of rule (3.1) requires explicit estimation of the class-conditional probability density functions f_j . For (3.2), some regression technique can be used to

estimate the posterior probabilities q_j directly without separately considering the class-conditional densities.

The probability of a vector $\mathbf{x}$ to be misclassified is notated $\epsilon(\mathbf{x})$. Using the Bayes rule, it is $\epsilon_{\text{BAYES}}(\mathbf{x}) = 1 - \max_{j=1,\dots,c} q_j(\mathbf{x})$. Thus, the *overall misclassification rate* ϵ of the Bayes classifier is

$$\epsilon_{\text{BAYES}} = 1 - \int_{\substack{\mathbf{x} \in \mathbb{R}^d \\ j = g_{\text{BAYES}}(\mathbf{x})}} f_j(\mathbf{x}) d\mathbf{x} . \quad (3.4)$$

3.2 Density Estimation Methods

In the density estimation approach, estimates for both the prior probabilities P_j and the class-conditional densities $f_j(\mathbf{x})$ are needed in (3.1). The estimation of the former is quite straightforward. The more difficult and vague task is to estimate the class-conditional densities. A classical *parametric* approach is to model them as multivariate Gaussians. Depending on whether equal or unequal class covariances are assumed, the logarithm of $P_j f_j(\mathbf{x})$ is then either a linear or quadratic function of $\mathbf{x}$, giving rise to *Linear Discriminant Analysis* (LDA) and *Quadratic Discriminant Analysis* (QDA) (see McLachlan 1992). A recent development is *Regularized Discriminant Analysis* (RDA) (Friedman 1989), which interpolates between LDA and QDA.

The success of these methods depends heavily on the validity of the normality assumption. If the class-conditional densities truly are normal, a near-Bayesian classification error level can be achieved. On the other hand, if the densities are neither unimodal nor continuous, disastrous performance may follow. The critical areas for classification accuracy are, however, those where the distributions of the classes overlap. If the normality assumption holds there, the classification accuracy may be good even though the overall performance of the density estimation was poor.

In *nonparametric* density estimation, no fixed parametrically-defined form for the estimated density is assumed. Kernel or Parzen estimates, as well as k -nearest neighbor methods with large values of k , are examples of popular nonparametric density estimation methods. They give rise to *Kernel Discriminant Analysis* (KDA) (Hand 1982) and *k-Nearest Neighbor* (k -NN) classification rules (see Dasarathy 1991, and Section 3.4.1).

In another approach, the densities are estimated as finite mixtures of some standard probability densities by using the *Expectation-Maximization* (EM) algorithm or some other method (Dempster et al. 1977; Redner and Walker 1984; Tråvén 1991; Priebe and Marchette 1991; Priebe and Marchette 1993; Hastie and Tibshirani 1996). Such an approach can be viewed as an economized KDA or as an instance of the *Radial Basis Function* (RBF) approach (Broomhead and Lowe 1988; Moody and Darken 1989). The Self-organizing Reduced Kernel Density Estimator introduced by Holmström and Hämmäläinen (1993) estimates densities in the spirit of radial basis functions. The corre-

sponding classification method is here referred to as *Reduced Kernel Discriminant Analysis* (RKDA).

3.2.1 Discriminant Analysis Methods

Quadratic Discriminant Analysis (QDA) (see McLachlan 1992) is based on the assumption that pattern vectors from class j are normally distributed with mean vector $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$. The density estimation approach leads to the rule

$$g_{\text{QDA}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmax}} \left[\log \hat{P}_j - \frac{1}{2} \log \det \hat{\boldsymbol{\Sigma}}_j - \frac{1}{2} (\mathbf{x} - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}_j) \right]. \quad (3.5)$$

Here $\hat{\boldsymbol{\mu}}_j$ and $\hat{\boldsymbol{\Sigma}}_j$ denote the class-wise sample mean and the sample covariance estimates. Likewise, $\hat{P}_j$ is the sample estimate for the *a priori* of class j . The operator “det” stands for the determinant of a matrix.

If it is assumed that the classes are normally distributed with different mean vectors but with a common covariance matrix $\boldsymbol{\Sigma}$, then the previous formula is simplified to the *Linear Discriminant Analysis* (LDA) (see McLachlan 1992) rule

$$g_{\text{LDA}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmax}} \left[\log \hat{P}_j + \hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \frac{1}{2} \hat{\boldsymbol{\mu}}_j) \right], \quad (3.6)$$

where a natural estimate for $\boldsymbol{\Sigma}$ is the pooled covariance matrix estimate $\hat{\boldsymbol{\Sigma}} = \sum_{j=1}^c \hat{P}_j \hat{\boldsymbol{\Sigma}}_j$.

Regularized Discriminant Analysis (RDA) by Friedman (1989) is a compromise between LDA and QDA. The decision rule is otherwise the same as (3.5), but instead of $\hat{\boldsymbol{\Sigma}}_j$, regularized covariance estimates $\hat{\boldsymbol{\Sigma}}_j(\lambda, \gamma)$ with two regularizing parameters are used. Parameter λ controls the shrinkage of the class-conditional covariance estimates toward the pooled estimate and γ regulates the shrinkage toward a multiple of the identity matrix $\mathbf{I}$. Let us denote by $\mathbf{K}_j$ the matrix $\sum_{i=1}^{n_j} (\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_j)^T$, and let $\mathbf{K} = \sum_{j=1}^c \mathbf{K}_j$. Then

$$\hat{\boldsymbol{\Sigma}}_j(\lambda, \gamma) = (1 - \gamma) \hat{\boldsymbol{\Sigma}}_j(\lambda) + \frac{\gamma}{d} \operatorname{tr} \left(\hat{\boldsymbol{\Sigma}}_j(\lambda) \right) \mathbf{I}, \quad \text{where} \quad (3.7)$$

$$\hat{\boldsymbol{\Sigma}}_j(\lambda) = \frac{(1 - \lambda) \mathbf{K}_j + \lambda \mathbf{K}}{(1 - \lambda) n_j + \lambda n} \quad (3.8)$$

and the operator “tr” stands for the trace of a matrix. The constant d is again the dimensionality of the data. QDA is obtained when $\lambda = 0, \gamma = 0$, and LDA when $\lambda = 1, \gamma = 0$, provided that the estimates $\hat{\boldsymbol{\Sigma}}_j = \mathbf{K}_j/n_j$ and $\hat{P}_j = n_j/n$ are used.

3.2.2 Kernel Discriminant Analysis, Probabilistic Neural Net

In *Kernel Discriminant Analysis* (KDA) (Hand 1982; Silverman and Jones 1989), kernel estimates $\hat{f}_j(\mathbf{x})$ of the class-conditional densities are formed. Then, the classification rule (3.1) is applied. The estimate of the class-conditional density of class j is

$$\hat{f}_j(\mathbf{x}) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_{h_j}(\mathbf{x} - \mathbf{x}_{ij}), \quad j = 1, \dots, c, \quad (3.9)$$

where, given a fixed probability density function $K(\mathbf{x})$, called the kernel, $h_j > 0$ is the smoothing parameter of class j , and K_h denotes the scaled kernel $K_h(\mathbf{x}) = h^{-d}K(\mathbf{x}/h)$. This scaling ensures that K_h and, hence, each $\hat{f}_j$ is a probability density. A popular choice is the symmetric Gaussian kernel $K(\mathbf{x}) = (2\pi)^{-d/2} \exp(-\|\mathbf{x}\|^2/2)$. The choice of suitable values for the smoothing parameters is crucial, and several approaches have been proposed, among others, by Silverman (1986), Scott (1992), McLachlan (1992), and by Wand and Jones (1995).

The selection of the smoothing parameters can be based on a cross-validated error count. Two methods, here denoted by KDA1 and KDA2, are considered here. In the first method, all the smoothing parameters h_j are fixed to be equal to a parameter h . The optimal value for h is then selected by cross-validation (see Section 3.6) as the value which minimizes the cross-validated error count. In the second method, the smoothing parameters are let to vary separately starting from a common value selected in KDA1. Both methods lead to optimization problems in which the objective function is piecewise constant. In KDA1, the search space is one-dimensional, and the optimization problem can be solved simply by evaluating the objective function on a suitable grid of values of h .

In KDA2, the lack of smoothness of the objective function is a problem. Instead of minimizing the error count directly, it is advantageous to minimize a smoothed version of it. In a smoothing method described by Tutz (1986), the class-conditional posterior-probability estimates $\hat{q}_j(\mathbf{x})$ that correspond to the current smoothing parameters are used to define the functions u_j ,

$$u_j(\mathbf{x}) = \exp(\gamma \hat{q}_j(\mathbf{x})) / \sum_{k=1}^c \exp(\gamma \hat{q}_k(\mathbf{x})), \quad \gamma > 0. \quad (3.10)$$

The smoothed error count is given by $n - \sum_{i=1}^n u_{j_i}(\mathbf{x}_i)$ which converges toward the true error count as $\gamma \rightarrow \infty$. Since the smoothed error count is a differentiable function of the smoothing parameters, a gradient-based minimization method can be used for the optimization.

The *Probabilistic Neural Network* (PNN) of Specht (1990) is the neural network counterpart of KDA. All training vectors are stored and used as a set of Gaussian densities. In practice, only a subset of the kernels is actually evaluated when the probability values are calculated.

3.2.3 Reduced Kernel Density Analysis, Radial Basis Functions

The standard kernel density estimate suffers from the curse of dimensionality: as the dimension d of data increases, the size of a sample $\mathbf{x}_1, \dots, \mathbf{x}_n$ required for an accurate estimate of an unknown density f grows quickly. On the other hand, even if there are enough data for accurate density estimation, the application may limit the complexity of the classifier applicable in practice. A kernel estimate with a large number of terms may be computationally too expensive to use. One solution is to *reduce* the estimate, i.e., to use fewer kernels but to place them at optimal locations. It is also possible to introduce kernel-dependent weights and smoothing parameters. Various reduction approaches have been described by Fukunaga and Mantock (1984), Fukunaga and Hayes (1989), Grabec (1990), Smyth and Mellstrom (1992), and Wu and Fallside (1991). Some of these methods are essentially the same as the *Radial Basis Function* (RBF) approach of classification (Broomhead and Lowe 1988; Moody and Darken 1989; Bishop 1995).

The Self-organizing Reduced Kernel Density Estimate (Holmström and Hämmäläinen 1993) has the form

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{\ell} w_i K_{h_i}(\mathbf{x} - \mathbf{m}_i), \quad (3.11)$$

where $\mathbf{m}_1, \dots, \mathbf{m}_{\ell}$ are the reference vectors of a Self-Organizing Map (Kohonen 1995), $w_1, \dots, w_{\ell}$ are nonnegative weights with $\sum_{i=1}^{\ell} w_i = 1$, and h_i is a smoothing parameter associated with the i th kernel. In order to achieve substantial reduction, $\ell \ll n$ should be selected. The kernel locations $\mathbf{m}_i$ are obtained by training the Self-Organizing Map by using the whole available sample $\mathbf{x}_1, \dots, \mathbf{x}_n$ from f . The weights w_i are computed iteratively and reflect the amount of training data in the Voronoi regions of the corresponding reference vectors. The smoothing parameters are optimized via stochastic gradient descent that attempts to minimize a Monte Carlo estimate of the integrated squared error $\int (\hat{f} - f)^2$. Simulations have shown that when the underlying density f is multimodal, the use of the feature map algorithm gives better density estimates than k -means clustering, an approach proposed by MacQueen (1967). *Reduced Kernel Discriminant Analysis* (RKDA) (Holmström and Hämmäläinen 1993) uses estimates (3.11) for the class-conditional densities in the classifier (3.1). A drawback of RKDA in pattern classification applications is that the smoothing parameters of the class-conditional density estimates used in the approximate Bayes classifier are optimized from the point of view of integrated squared error. Instead, the optimization ought to be based on the discrimination performance, which is the true focus of interest.

3.3 Regression Methods

In the second approach to classification, the class-posterior probabilities $q_j(\mathbf{x}) = P(J = j \mid \mathbf{X} = \mathbf{x})$ are directly estimated by using some regression technique. *Parametric* methods include linear and logistic regression. Examples of the *nonparametric* methods are

Projection Pursuit (PP) (Friedman and Stuetzle 1981; Flick et al. 1990), *Generalized Additive Models* (GAM) (Hastie and Tibshirani 1990), *Multivariate Adaptive Regression Splines* (MARS) (Friedman 1991), *Local Linear Regression* (LLR) (Cleveland and Devlin 1988), and the Nadaraya-Watson kernel regression estimator (Nadaraya 1964; Watson 1964) (see Wand and Jones 1995; Koistinen and Holmström 1992), which is also called the *General Regression Neural Network* (GRNN) (Specht 1991). Other neural network approaches regarded as *semiparametric* include (see Haykin 1994; Bishop 1995) *Multi-Layer Perceptrons* (MLP) and *Radial Basis Function* (RBF) expansions.

“One-of- c ” coding can be used to define the desired output $\mathbf{y}_i$ for the pattern $\mathbf{x}_i$ from class j_i to be the unit vector $[0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^c$, with 1 in the j_i th place. In the *least-squares* approach, one then tries to solve the minimization

$$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c (\mathbf{y}_i^{(j)} - \mathbf{r}^{(j)}(\mathbf{x}_i))^2 = \min_{\mathbf{r} \in \mathcal{R}} \quad (3.12)$$

over a family $\mathcal{R}$ of $\mathbb{R}^c$ -valued response functions $\mathbf{r}$, where we denote the j th component of a vector $\mathbf{z}$ by $\mathbf{z}^{(j)}$. The corresponding mathematical expectation is minimized by the vector of class-posterior probabilities, $\mathbf{q} = [q_1, \dots, q_c]^T$. Of course, this ideal solution may or may not belong to the family $\mathcal{R}$, and, besides, sampling variation will nevertheless prevent the exact estimation of $\mathbf{q}$ even when it does belong to $\mathcal{R}$ (White 1989; Richard and Lippman 1991).

The least-squares fitting criterion (3.12) can be understood as emerging from the use of the maximum likelihood principle for estimating a regression model where errors are distributed normally. The applicability of the least-squares method is, however, not limited to the normality assumption. If no parametric model is assumed, the properties of the estimate may be difficult to establish. The *logistic* approach (see McLachlan 1992) uses binomially distributed error, which is the statistically correct model if independent and identically distributed vectors are assumed. One natural multivariate logistic regression approach is to model the posterior probabilities as the *softmax* (Bridle 1990) of the components of $\mathbf{r}$,

$$P(J = j \mid \mathbf{X} = \mathbf{x}) = q_j(\mathbf{x}) = \frac{\exp(\mathbf{r}^{(j)}(\mathbf{x}))}{\sum_{k=1}^c \exp(\mathbf{r}^{(k)}(\mathbf{x}))} . \quad (3.13)$$

Note that this also satisfies the natural condition $\sum_{k=1}^c q_k = 1$. A suitable fitting criterion is to maximize the conditional log-likelihood of $\mathbf{y}_1, \dots, \mathbf{y}_n$ given that $\mathbf{X}_1 = \mathbf{x}_1, \dots, \mathbf{X}_n = \mathbf{x}_n$. In the case of two classes, this approach is equivalent to the use of the cross-entropy fitting criterion (Bishop 1995).

A very natural approach would be a regression technique that uses the error rate as the fitting criterion to be minimized (Highleyman 1962). *Classification and Regression Trees* (CART) are an example of a nonparametric technique that estimates the posterior probabilities directly but uses neither the least-squares nor the logistic regression approach (Breiman et al. 1984).

3.3.1 Multi-Layer Perceptron

In the standard *Multi-Layer Perceptron* (MLP), there are d inputs, ℓ hidden units and c output units. All the feed-forward connections between adjacent layers are included, and the logistic activation function is used in the hidden and output layers (see Haykin 1994; Bishop 1995). Such a network has $(d+1)\ell + (\ell+1)c$ adaptable weights, which are determined by minimizing the sum-of-squares errors criterion (3.12).

To scale the response vectors better within the range of the logistic function, a modified desired output for input $\mathbf{x}_i$ can be used. For example, a vector $\tilde{\mathbf{y}}_i$, with the components $\tilde{\mathbf{y}}_i^{(j)} = 0.1 + 0.8\mathbf{y}_i^{(j)}$, can replace the original $\mathbf{y}_i$. Then the scaled outputs $1.25(\mathbf{r}^{(j)}(\mathbf{x}) - 0.1)$ of the optimized network can be regarded as estimating the posterior probabilities $P(J = j \mid \mathbf{X} = \mathbf{x})$. A good heuristic is to start the local optimizations from a variety of random initial points and to keep the weights yielding the minimum value for the sum-of-squares error to prevent the network from converging to a shallow local minimum. It is advisable to scale the random initial weights so that the inputs into the logistic activation functions are of the order unity (Bishop 1995).

In weight decay regularization (see Bishop 1995), a penalty for weights that have a large absolute value is introduced in order to encourage smooth network mappings. The training of MLPs with weight decay (MLP+WD) tries to minimize the criterion

$$\frac{1}{n} \left[\sum_{i=1}^n \sum_{j=1}^c (\tilde{\mathbf{y}}_i^{(j)} - \mathbf{r}^{(j)}(\mathbf{x}_i, \mathbf{w}))^2 + \lambda \sum_{w \in W} w^2 \right]. \quad (3.14)$$

Here, $\mathbf{w}$ comprises all the weights and biases of the network, W is the set of weights between adjacent layers excluding the biases, and λ is the weight decay parameter. The network inputs and the outputs of the hidden units should be roughly comparable before the weight decay penalty in the form given above makes sense. To achieve this, it may be necessary to rescale the inputs.

3.3.2 Local Linear Regression

Local Linear Regression (LLR) (Cleveland and Devlin 1988; Wand and Jones 1995) is a nonparametric regression method which has its roots in classical methods proposed for the smoothing of time series data, (see Cleveland and Loader 1995). Such estimators have received more attention recently, (see Hastie and Loader 1993). The particular version described below is also called LOESS. Local Linear Regression models the regression function in the neighborhood of each point $\mathbf{x}$ by means of a linear function $\mathbf{z} \mapsto \mathbf{a} + \mathbf{B}(\mathbf{z} - \mathbf{x})$. Given training data $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$, the fit at point $\mathbf{x}$ is calculated as follows. First, a weighted linear least-squares problem involving an unknown matrix, $\mathbf{B}$, and an unknown vector, $\mathbf{a}$, is solved,

$$\sum_{i=1}^n \|\mathbf{y}_i - \mathbf{a} - \mathbf{B}(\mathbf{x}_i - \mathbf{x})\|^2 w(\|\mathbf{x}_i - \mathbf{x}\|/h(\mathbf{x})) = \min_{\mathbf{a}, \mathbf{B}} \quad (3.15)$$

Then, the coefficient vector $\mathbf{a}$ gives the fit at $\mathbf{x}$. A reasonable choice for the function w is the tricube weight function (Cleveland and Devlin 1988), $w(u) = \max((1 - |u|^3)^3, 0)$. The local bandwidth $h(\mathbf{x})$ is controlled by a neighborhood size parameter $0 < \alpha \leq 1$. A variable k is selected to be equal to αn rounded to the nearest integer and $h(\mathbf{x})$ is made equal to the distance to the k th closest neighbor of $\mathbf{x}$ among the vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$. If the components of $\mathbf{x}$ are measured in different scales, it is advisable to select the metric for the nearest neighbor calculation carefully. At a given $\mathbf{x}$, the weighted linear least-squares problem can be reduced to inverting a $(d + 1) \times (d + 1)$ matrix, where d is the dimensionality of $\mathbf{x}$, (see Wand and Jones 1995).

3.3.3 Tree classifier, MARS and FDA

The introduction of tree-based models in statistics dates back to Morgan and Sonquist (1963), although their current popularity is largely due to the seminal book by Breiman et al. (1984). For Euclidean pattern vectors $\mathbf{x} = [x_1, \dots, x_d]^T$, a classification tree is a binary tree in which, at each node, the decision to branch either to the left or right is based on a test of the form $x_i \geq \lambda$. The cut-off values λ are chosen to optimize a suitable fitting criterion. The tree growing algorithm recursively splits the pattern space $\mathbb{R}^d$ into hyperrectangles while trying to form maximally pure nodes, that is, subdivision rectangles that ideally contain training vectors from one class only. Stopping criteria are used to keep the trees reasonable in size, although a commonly-employed strategy is to first grow a large tree that overfits the data and then use a separate pruning stage to improve its generalization performance. A terminal node is labeled according to the class with the largest number of training vectors in the associated hyperrectangle. The tree classifier therefore uses the Bayes rule with the class-posterior probabilities estimated by locally constant functions. The particular tree classifier described here is available as a part of the S-Plus statistical software package (see Becker et al. 1988; Chambers and Hastie 1992; Venables and Ripley 1994). This implementation uses a likelihood function to select the optimal splits (Clark and Pregibon 1992). Pruning is performed by the minimal cost-complexity method. The cost of a subtree T is taken to be

$$R_\alpha(T) = \epsilon(T) + \alpha \cdot \text{size}(T) , \quad (3.16)$$

where $\epsilon(T)$ is an estimate of the classification error of T , the size of T is measured by the number of its terminal nodes, and $\alpha > 0$ is a cost parameter. An overfitted tree is pruned by giving increasingly large values to α and by selecting nested subtrees that minimize R_α .

Multivariate Adaptive Regression Splines (MARS) (Friedman 1991) is a regression method that shares features with tree-based modeling. MARS estimates an unknown function r using an expansion

$$\hat{r}(\mathbf{x}) = a_0 + \sum_{k=1}^M a_k B_k(\mathbf{x}) , \quad (3.17)$$

where the functions B_k are multivariate splines. The algorithm is a two-stage procedure. It begins with a forward stepwise phase which adds basis functions to the model in a deliberate attempt to overfit the data. The second stage of the algorithm is the standard linear regression backward elimination. The maximum order of variable interactions (products of variables) allowed in the functions B_k , as well as the maximum value of M allowed in the forward stage, are parameters that need to be tuned experimentally. Backward model selection uses the generalized cross-validation criterion introduced by Craven and Wahba (1979).

The original MARS algorithm only fits scalar-valued functions and, therefore, is not well-suited to discriminatory tasks with more than two classes. A recent proposal called *Flexible Discriminant Analysis* (FDA) (Hastie et al. 1994), with its publicly available S-Plus implementation in the StatLib program library, contains vector-valued MARS as one of its ingredients. FDA is not, however, limited to just MARS since it allows other regression techniques as its building blocks as well. In FDA, c separate MARS models $\mathbf{r}^{(j)}$ with equal basis function sets but different coefficients a_k can first be trained to map training vectors $\mathbf{x}_i$ to the corresponding unit vectors $\mathbf{y}_i$. Then, a linear map A is constructed to map the regression function output space $\mathbb{R}^c$ onto a lower dimensional feature space $\mathbb{R}^\ell$ in a manner that optimally facilitates prototype classification based on the transformed class means $A(\mathbf{r}(\hat{\boldsymbol{\mu}}_j))$ and a weighted Euclidean distance function.

3.4 Prototype Classifiers

One distinct branch of classifiers under the title “others” in Table 3.1 on page 28 are the prototype classifiers LVQ, k -NN, and L- k -NN. They share the principle of keeping copies of training vectors in memory. The classification decision $g(\mathbf{x})$ is then based on the distances between the stored prototypes and the input vector $\mathbf{x}$. Either the training vectors are retained as such, or some sort of training phase is used to extract properties of a multitude of training vectors to each of the prototypes. In either case, these classifiers are typical representatives of the *nonparametric* classification methods.

3.4.1 k -Nearest Neighbor Classifiers

In a *k-Nearest Neighbor* (k -NN) classifier, each class is represented by a set of prototype vectors (see Dasarathy 1991). The k closest neighbors of an input pattern vector $\mathbf{x}$ are found among all the prototypes, and the majority rule determines the class label. A potential tie of two or more classes can be broken, for example, by decreasing k by one and re-voting.

In classical pattern recognition, the nonparametric k -NN classification method has been very popular since the first publication by Fix and Hodges (1951) and an important limiting accuracy proof by Cover and Hart (1967). The k -NN rule should still be regarded as a sort of a reference classifier, against which other statistical and neural classifiers

should be compared (Toussaint et al. 1982). Its advantage is that no time is needed in training the classifier. On the other hand, a huge amount of memory and time are needed during the classification phase. An important improvement in memory consumption, while still keeping the classification accuracy moderate, may be achieved by using some *editing* method. Perhaps the oldest editing rule, the *Condensing* algorithm, was presented by Hart (1968). The classifier is initialized with no prototypes in it. Then, the training vectors are tentatively classified, and if the classifier is unable to correctly classify an input vector, that vector is added as a new prototype. In an editing rule by Wilson (1972), each vector in the training set is classified using all the training vectors other than itself. After all vectors have been tentatively classified, those yielding an error are deleted from the prototype set. Devijver and Kittler (1982) have suggested a more advanced *Multiedit* algorithm in which the training set is partitioned to smaller vector sets which are then used in classifying one another. The misclassified vectors are deleted after each iteration of the algorithm, and the remaining prototypes are re-pooled. This iteration continues until no editing has taken place for a preset number of epochs. In all the described editing methods, a vector set originally used as a k -NN classifier is converted to a smaller edited prototype set which is employed as a 1-NN classifier.

3.4.2 Learning Vector Quantization

The *Learning Vector Quantization* (LVQ) algorithm (Kohonen 1995) produces a set of prototype or *codebook* pattern vectors $\mathbf{m}_i$ that are applicable in a 1-NN classifier. The training consists of moving a fixed number ℓ of codebook vectors iteratively toward, or away from, the training vectors $\mathbf{x}_i$. The variations of the LVQ algorithm differ in the rules determining which of the codebook vectors are updated, and how. In the learning process, all the modifications to the codebook vectors are made according to the *delta rule*. This means that each additive correction to a codebook vector value is a fraction of the difference between the input vector value and the current codebook vector value, i.e.,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] . \quad (3.18)$$

The α parameter controls the learning rate and is usually made to decrease monotonically with time. Positive values of α cause the movement of $\mathbf{m}_i$ toward $\mathbf{x}$ and negative values away from it. In LVQ2 and LVQ3, an additional parameter w is used to control the relative width of a “window” around the midplane of the two nearest codebook vectors. The vector updates take place only if the input vector $\mathbf{x}$ falls into this window.

The LVQ learning process can be interpreted either as an iterative movement of the decision boundaries between neighboring classes, or as a way to generate a set of codebook vectors whose density reflects the shape of the function s defined as

$$s(\mathbf{x}) = P_j f_j(\mathbf{x}) - \max_{k \neq j} P_k f_k(\mathbf{x}) , \quad (3.19)$$

where $j = g_{\text{BAYES}}(\mathbf{x})$. Note that the zero set of s consists of the Bayes optimal decision boundaries.

3.4.3 Learning k -NN Classifier

Besides editing, iterative learning algorithms can also be applied to k -NN classifiers. The adaptation rules of the *Learning k -NN Classifier* (L- k -NN) presented by Laaksonen and Oja (1996a) resemble those of LVQ, but, unlike LVQ, it is still able to utilize the improved classification accuracy provided by majority voting. As in LVQ, the Learning k -NN rules use a fixed number ℓ of prototype vectors with predetermined class labels j for classification. Another common characteristic is that the prototypes are moved iteratively according to the delta rule (3.18) during the training period. After the adaptation, the classification function for a new unknown input vector is based on the majority label among its k closest code vectors exactly as in the standard k -NN.

Three slightly different training algorithms have been proposed for the L- k -NN Classifier (Laaksonen and Oja 1996a). The objective of all these rules is to make the correct classification of the input vectors more probable by moving some of the prototype vectors $\mathbf{m}_{ij}$ in the neighborhood of an input vector $\mathbf{x}$ toward and some away from it. Figure 3.2 illustrates these three rules in a simple two-dimensional two-class (white and gray) case. In the figure, the value of the parameter k is 3. The area that contains the training vector and its k nearest prototype vectors is shaded. The training vector is depicted with a small white circle. According to the 3-NN rule, the vector has originally been classified erroneously as gray.

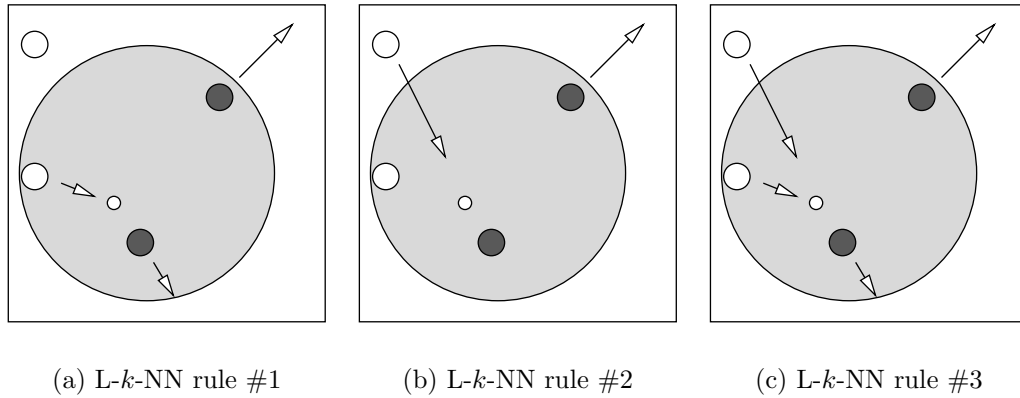


Figure 3.2: The L- k -NN training rules #1, #2, and #3. The small circle depicts the training vector, the larger ones the prototype vectors. The arrows indicate the directions in which the vectors are moved.

L- k -NN rule #1. All the k nearest prototypes are moved. If the class of the prototype is the same as the class of the training vector, the prototype is moved toward the training vector; otherwise, away from it. For $k = 1$, this is equivalent to the LVQ1 algorithm.

L- k -NN rule #2. Only the k th and the $(k + 1)$ th nearest prototypes are moved if the interchange of their order would change the classification of the training vector

from incorrect to correct. If the class of the prototype is the same as the class of the training vector, the prototype is moved toward the training vector; otherwise, away from it.

L- k -NN rule #3. All the $k + 1$ nearest prototypes are moved. If the class of the prototype is the same as the class of the training vector, the prototype is moved toward the training vector; otherwise, away from it.

Compared to the standard k -NN classifier, L- k -NN needs less memory to store the prototype vectors, because each trained prototype represents a multitude of training vectors. Therefore, both L- k -NN and LVQ are somewhat more of a semiparametric model than the classical k -NN classifier.

3.5 Special Properties of Neural Methods

The previous discussion characterized some popular classification techniques in terms of their underlying mathematical principles. In this general context, many neural networks can be seen as representatives of certain larger families of statistical techniques. This abstract point of view, however, fails to identify some of those key features of neural networks that characterize them as a distinct methodology.

From the very beginning of neural network research by McCulloch and Pitts (1943) and Rosenblatt (1958 and 1961), the goal was to demonstrate problem-solving without explicit programming. The neurons and networks were supposed to learn from examples and store this knowledge in a distributed way among the connection weights.

The original methodology was exactly the opposite to the goal-driven, or top-down, design of statistical classifiers in terms of explicit error functions. In neural networks, the approach has been bottom-up; to start from a very simple linear neuron that computes a weighted sum of its inputs, to add a saturating smooth nonlinearity, and to construct layers of similar parallel units. It turned out that “intelligent” behavior like speech synthesis (Sejnowski and Rosenberg 1987) emerged through simple learning rules. The computational aspect has always been central in neural networks. At least in principle, everything that a neural network does should be accomplished by a large number of simple local computations which use the available input and output signals, as in real neurons. Unlike heavy numerical algorithms, no such operations as matrix inversions are required by, or permitted for, neural methods. Perhaps the best example of a clean-cut neural network classifier is the LeNet system for handwritten digit recognition (LeCun et al. 1989; Bottou et al. 1994). Computational models like this support well the implementation in regular VLSI circuits.

In the current neural networks research, these original views are clearly becoming vague as some of the most fundamental neural networks, such as the one-hidden-layer MLP or RBF networks, have been shown to have very close connections to statistical techniques.

The goal remains, however, to build much more complex artificial neural systems for demanding tasks like speech recognition (Kohonen 1988) or computer vision (Lampinen and Oja 1995). In such applications, it is difficult, or even impossible, to state the exact optimization criteria for all the consequent processing stages.

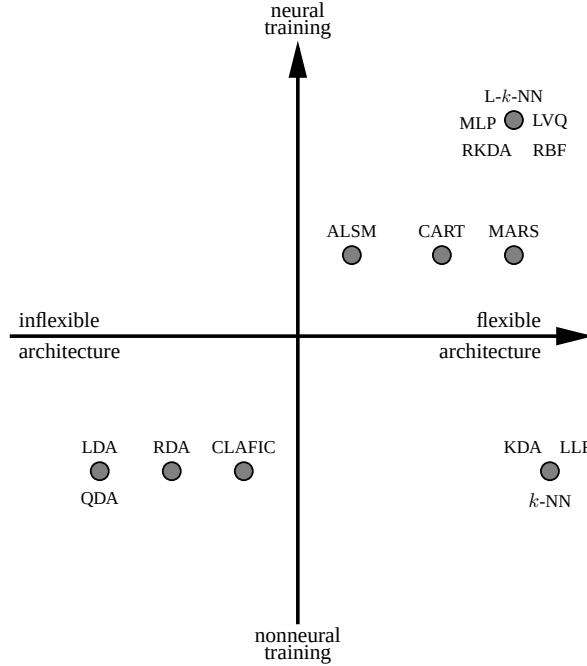


Figure 3.3: Neural characteristics of some classifiers according to Holmström et al. (1997).

Figure 3.3 assesses the neural characteristics of some of the discussed classification methods. The horizontal axis describes the flexibility of a classifier architecture in terms of the versatility of the discriminant function family encompassed by a particular method. A high flexibility in the architecture is a property often associated with neural networks. In some cases (MLP, RBF, CART, MARS), the flexibility can also include algorithmic model selection during learning.

In the vertical dimension, the various classifiers are categorized on the basis of how they are designed from a training sample. Training is considered nonneural if the training vectors are used as such in classification (e.g., k -NN, KDA), or if some statistics are first estimated in batch mode and the discriminant functions are computed from them (e.g., QDA, CLAFIC). Neural learning is characterized by simple local computations in a number of real, or virtual, processing elements. Neural learning algorithms are typically of the error correction type; for some such algorithms, not even an explicit cost function exists. Typically, the training set is used several times (epochs) in an on-line mode. Note, however, that for some neural networks (MLP, RBF), the current implementations, in fact, often employ sophisticated optimization techniques which would justify moving them downwards in the map to the lower half plane.

In this schematic representation, the classical LDA and QDA methods are seen as the least neural with the RDA and CLAFIC that possess at least some degree of flexibility in their architecture. The architecture of KDA, k -NN, and LLR is extremely flexible. In comparison to CLAFIC, the ALSM method allows for both incremental learning and flexibility in the form of subspace dimensions that can change during learning. In this view, such methods as MLP, RBF, RKDA, LVQ, and L- k -NN are good examples of neural classifiers. ALSM, CART, and MARS also exhibit neural characteristics to some degree.

3.6 Cross-Validation in Classifier Design

In order to get reliable estimates of classifier performance, the available data should first be divided into two separate parts: the training sample and the testing sample. The whole process of classifier design should then be based solely on the training sample. In addition to parameter estimation, the design of some classifiers involves the choice of various tuning parameters and model, or architecture, selection. To utilize the training sample efficiently, cross-validation (Stone 1974), or “rotation” (Devijver and Kittler 1982), can be used. In v -fold cross-validation, the training sample is first divided into v disjoint subsets. One subset at a time is then put aside, a classifier is designed on the basis of the union of the remaining $v - 1$ subsets and tested for the held-out subset. Cross-validation approximates the design of a classifier which uses all the training data and which is then tested with an independent set of data. The cross-validation process enables defining a reasonable objective function to be optimized in classifier design. For instance, for a fixed classifier, the dimension of the pattern vector can be selected so that it minimizes the cross-validated error count. After the optimization, an unbiased estimate of the performance of the optimized classifier can be obtained by means of a separate testing sample. Notice that the performance estimates may become biased if the testing sample is in any way used during the training of the classifier.

3.7 Rejection

Other criteria than minimum classification error in the sense of the Bayesian misclassification rate (3.4) can be important in practice. These criteria include the use of class-dependent misclassification costs and Neyman-Pearson style classification (Young and Calvert 1974; Holmström et al. 1995). The use of a *reject class* can help to reduce the misclassification rate ϵ in tasks where exceptional handling, for instance, by a human expert, of particularly ambiguous cases is feasible. The decision to reject a pattern $\mathbf{x}$ and to handle it separately can be based on its probability to be misclassified, which for the Bayes rule is $\epsilon(\mathbf{x}) = 1 - \max_{j=1,\dots,c} q_j(\mathbf{x})$. The highest misclassification probability occurs when the posterior probabilities $q_j(\mathbf{x})$ are equal and then $\epsilon(\mathbf{x}) = 1 - 1/c$. Consequently, a rejection threshold $0 \leq \theta \leq 1 - 1/c$ can be selected, and $\mathbf{x}$ rejected if

$$\epsilon(\mathbf{x}) > \theta. \quad (3.20)$$

The notation $g(\mathbf{x})$ used for the classification function can be extended to include the rejection case by denoting with $g(\mathbf{x}) = 0$ all the rejected vectors $\mathbf{x}$. When the overall rejection rate of a classifier is denoted by ρ , the rejection-error balance can be depicted as a curve in the $\rho\epsilon$ -plane, parameterized with the θ value. For example, in the recognition of handwritten digits, the rejection-error curve is found to be generally linear in the $\rho \log \epsilon$ -plane (Geist et al. 1992), as can be observed in Figure 8.6 on page 130.

3.8 Committees

In practice, one is usually able to classify a pattern by using more than one classifier. It is then quite possible that the combination of the opinions of several parallel systems results in improved classification performance. Such hybrid classifiers, classifier ensembles, or *committees*, have been studied intensively in recent years (Perrone 1994). A sample committee is displayed in Figure 3.4.

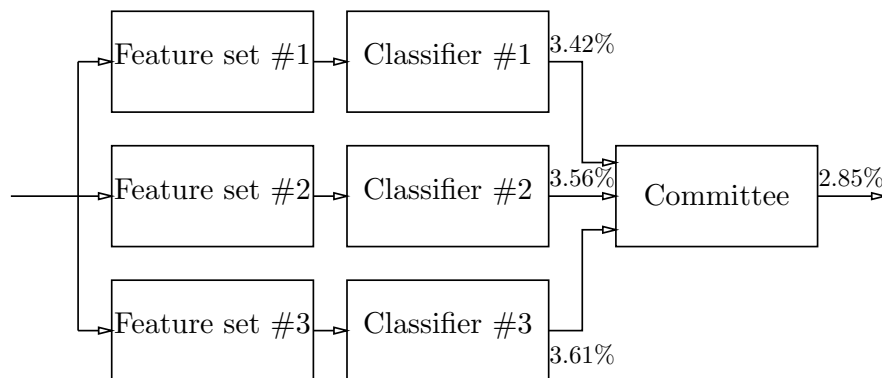


Figure 3.4: A committee classifier. In this semifictional example, the resulting error rate of the committee of three classifiers, each of which has a dedicated feature extraction block, is clearly lower than any of the error rates of its members.

In addition to improved classification performance, there are other reasons for using a committee classifier. The feature vector may be constructed of components that originate from very diverse domains. Some may be statistical quantities, such as moments, and others discrete structural descriptors, such as numbers of endpoints, loops, and so on. As a consequence, it may not be reasonable at all to concatenate all the features into a single feature vector and to use a single classifier. In some other situations, the computational burden can be reduced either during training or in the recognition phase if the classification is performed in several stages.

Various methods exist for forming a committee of classifiers, even if their output information is of different types. In the simplest case, a classifier only outputs its decision concerning the class of an input pattern, but, sometimes, some measure of the certainty

of the decision is also provided. The classifier may propose a set of classes in the order of decreasing certainty, or a measure of decision certainty may be given for all the classes. Various ways to combine classifiers with such types of output information are analyzed by Xu et al. (1992), Ho et al. (1992 and 1994), and by Huang et al. (1995).

The simplest decision rule is to use a majority rule among the classifiers in the committee, possibly ignoring the opinion of some of the classifiers (Xu et al. 1991). Two or more classifiers that use different sets of features may be combined to implement rejection of ambiguous patterns (Nadal et al. 1990; Kimura and Shridhar 1991; Suen et al. 1992; Lam and Suen 1994). A genetic algorithm can be applied in searching for optimal weights to combine the classifier outputs (Lam and Suen 1995). Theoretically more advanced methods may be derived from the EM-algorithm (Xu and Jordan 1993; Ho et al. 1994; Jordan and Jacobs 1994; Xu et al. 1995) or from the Dempster-Shafer theory of evidence (Franke and Mandler 1992; Xu et al. 1992).

The outputs of several regression-type classifiers may be combined linearly (Jacobs 1995) or nonlinearly (Tresp and Taniguchi 1995) to reduce the variance of the posterior-probability estimates. A more general case is the reduction of variance in continuous function estimation. Here, a set of MLPs can be combined in a committee classifier which has reduced output variance and thus smaller expected classification error (Hansen and Salamon 1990; Wolpert 1992; Perrone and Cooper 1993; Krogh and Vedelsby 1995). A separate confidence function may also be incorporated in each of the MLPs (Śmieja 1994).

Given a fixed feature extraction method, either a common training set can be used to design a number of different types of classifiers (Idan and Auger 1992) or, alternatively, different training sets can be used to design several versions of one classifier type (Drucker et al. 1993; Drucker et al. 1994; Hinton et al. 1995; Schwenk and Milgram 1995; Sollich and Krogh 1995).

3.9 On Comparing Classifiers

Some classification accuracies attained by using the classification algorithms described in the previous sections will be presented in Chapter 8. Such comparisons need, however, to be considered with utmost caution.

During the last few years, a large number of papers have described and analyzed various neural and other classification algorithms. The results of such experiments cannot generally be compared, due to the use of different raw data material, preprocessing, and testing policies. Prechelt (1996) analyzed articles published in two major neural networks journals in 1993 and 1994. In these articles, neural algorithms were employed in experimental evaluations. The bare conclusion was that the quality of the quantitative results, if presented at all, was poor. For example, the famous NETtalk experiments by Sejnowski and Rosenberg (1987) were replicated by Schmidt et al. (1994) and compared to the performance of a k -NN classifier. Their conclusion was that the original results were hard to reproduce and the regenerated MLP results were outperformed by the k -NN classifier.

Some larger evaluations, or *benchmarking* studies, have also been published. Each benchmark has tried to assess a set of classification algorithms in a fair and impartial setting. Some of the latest in this category include the studies by Blue et al. (1994), Cheng and Titterton (1994), Michie et al. (1994), Blayo et al. (1995), and by Holmström et al. (1997). Duin (1996) addressed the fundamental philosophical question involved in the classifier comparisons by asking whether the experiments are merely testing the skills of those experts who use the algorithms or whether they provide information on the applicability of the classifiers for the needs of the non-experts. Devroye (1988) calculated the distribution-free bounds for the difference between the achieved and the achievable error levels for a set of classification algorithms, both in the cases of finite and infinite training sets.

3.10 Classifiers: Theory and Practice

This chapter has presented a collection of different statistical and neural classification algorithms. The methods have been assessed with respect to their theoretical characteristics, including the way they model the underlying probability density functions of the data, and the parametric/nonparametric nature of their internal representations. Knowing all these facts, however, does not as such help in selecting the best available classifier for a particular pattern recognition application.

First of all, there cannot be such a thing as the best all-around classifier. Each real-world application calls for different characteristics of classifiers. For example, in some applications, it may be impossible to collect a comprehensive training sample of data. Yet, the system should be robust and able to reliably extrapolate from the available data. On the other hand, in some situations, different kinds of errors have different degrees of severity. Therefore, the classifier should somehow reflect the asymmetrical nature of the classification problem. It should thus be evident that no single classifier can be a sovereign solution in all situations.

One important aspect, which has not been discussed in detail in this chapter, is the computational complexity of the classification methods. From the point of view of the system designer, the time available for each classification decision, the cost of the system, and the obtainable overall error rate form a vicious circle to be solved. In general, the cost and the time can be regarded as given. Therefore, the system designer, in practice, has a fixed number of processor cycles to be divided between the stages of the pattern recognition application. Besides the limited number of computational operations available, the amount of system memory may, in some classifiers, turn out to be a limiting factor as well. Even though the memory devices are nowadays cheap, any limit in the size and cost of the system will eventually be reached.

The scalability of a classifier is an important design factor. The parametric methods are not scalable at all. Their accuracy may get better when the size of the training sample is increased but this improvement is likely to saturate very soon. After the estimation of the

model parameters is accurate enough, the limitations of the parametric model itself will become evident. The nonparametric methods, on the contrary, scale well. Their inherent problem is, therefore, whether the classification accuracy is adequate before some system resource has been exhausted. Another weakness of the nonparametric classifiers is that they, in general, are not able to extrapolate very reliably. The semiparametric methods try to combine the valuable properties of both the parametric and nonparametric approaches. Nothing, however, really guarantees that only the desirable characteristics are included and all undesirable properties excluded.

The designer of a pattern recognition application should thus be aware of a wide spectrum of different classification techniques. If she has in her toolbox a bunch of classifiers, she can actually evaluate the achievable recognition accuracies by using methods of different kinds. This evaluation process should be combined with the selection of the features to be used in the system. The feature extraction and classification stages form a tightly coupled pair whose cooperation is crucial to the overall performance of the system. Therefore, when a pattern recognition system is being designed, expertise in both the classification methods and the characteristics of the particular application area is required.

Chapter 4

Subspace Classification Methods

The motivation for subspace classifiers originates from compression and optimal reconstruction of multidimensional data with linear *principal components*. The use of linear subspaces as class models is based on the assumption that the vector distribution in each class lies approximately on a lower-dimensional subspace of the feature space. An input vector from an unknown class is then classified according to the shortest distance to the subspaces, each of which represents one class. Even though this linearity assumption is seldom valid, acceptable classification accuracies can be achieved if the input vector dimensionality is large enough.

In Table 3.1 on page 28, the subspace classifiers CLAFIC and ALSM were placed under the title “others”. This was done because, so far, there has not been a density function estimation or regression interpretation of the subspace methods. Such an interpretation is now given in Section 4.3.6. The subspace methods are generally regarded as semiparametric classifiers. This means that they are based on parametric models with effective techniques for controlling the number of free parameters. The traditional subspace classifiers are mostly straightforward statistical methods, but the newer learning variants can be formulated and interpreted in the neural computation framework.

The history of the subspace methods in data analysis was begun in the 1930s by Hotelling (1933). The value of the subspace methods in data compression and optimal reproduction was observed in the 1950s by Kramer and Mathews (1956). Ten years later, Watanabe et al. (1967) published the first application in pattern classification. Learning subspace methods emerged from the mid-1970s, after the pioneering work of Kohonen et al. (1978). From the beginning, these methods aimed for classification instead of optimal compression or reproduction. The guiding idea in the learning methods is to modify the bases of the subspaces in order to diminish the number of misclassifications. The nature of the modifications varies in different learning algorithms.

This chapter is organized as follows. Section 4.1 examines the classical subspace methods. Basic learning variants of the subspace classifier are described in Section 4.2. Section 4.3 focuses on some general considerations and improvements of the subspace classifiers. The prospects of the subspace methods are addressed in Section 4.4.

4.1 Classical Subspace Methods

First, this section introduces the basic mathematical notations and operations needed when dealing with subspaces in classification tasks. Second, classical subspace classification algorithms are presented. Some of the details are postponed until later sections of this chapter. The style of the notations and illustrations is adopted from Oja (1983).

4.1.1 Subspace Basics

In this presentation, all the operations take place in a d -dimensional real-valued vector space $\mathbb{R}^d$. The most common way to define an ℓ -dimensional subspace $\mathcal{L}$ uses a set of linearly independent basis vectors, $\{\mathbf{u}_1, \dots, \mathbf{u}_\ell\}$, which can be combined into a $d \times \ell$ matrix $\mathbf{U}$ which, thus, has rank ℓ . This notation leads easily to the definition of the subspace as the set of linear combinations of the basis vectors

$$\mathcal{L} = \mathcal{L}_{\mathbf{U}} = \left\{ \mathbf{x} \mid \mathbf{x} = \sum_{i=1}^{\ell} \zeta_i \mathbf{u}_i, \zeta_i \in \mathbb{R} \right\} = \left\{ \mathbf{x} \mid \mathbf{x} = \mathbf{U}\mathbf{z}, \mathbf{z} \in \mathbb{R}^{\ell} \right\}. \quad (4.1)$$

In the last form, the coefficient scalars $\{\zeta_1, \dots, \zeta_\ell\}$ were combined into a multiplier vector $\mathbf{z}$. In the notation, the basis vectors are not uniquely defined even though the subspace is. Thus, the set of vectors $\{\mathbf{u}_1, \dots, \mathbf{u}_\ell\}$ may be orthonormalized without loss of generality by using, for example, the well-known *Gram-Schmidt orthonormalization process* (see Golub and van Loan 1989), if the vectors are not already orthonormal. Figure 4.1 illustrates these concepts with a two-dimensional subspace in a three-dimensional space.

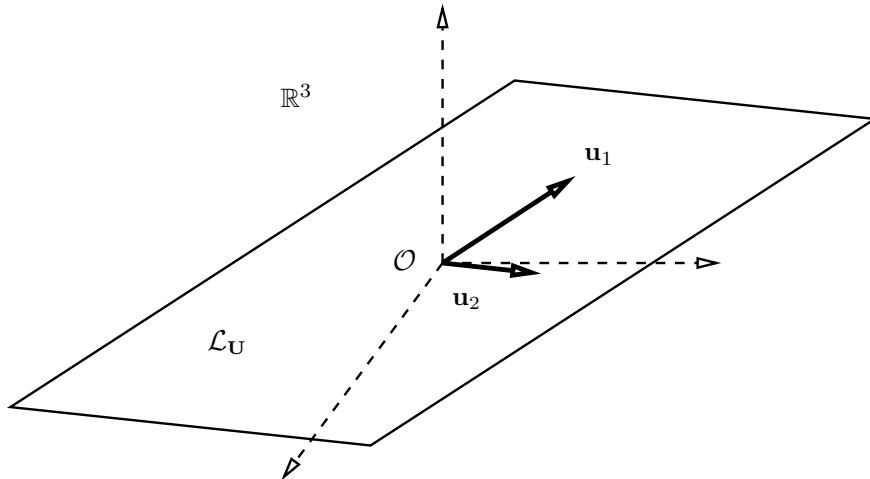


Figure 4.1: A two-dimensional subspace $\mathcal{L}$ in a three-dimensional space. $\mathcal{L}$ is defined by its basis vectors $\mathbf{u}_1$ and $\mathbf{u}_2$.

A central operation of the subspace methods is the *projection* of a vector $\mathbf{x}$ on a subspace $\mathcal{L}$. In the linear case, this operation can be expressed using a *projection matrix* $\mathbf{P}$ which has two distinctive characteristics. First, it associates each vector $\mathbf{x}$ of $\mathbb{R}^d$ with a projection vector $\mathbf{P}\mathbf{x} = \hat{\mathbf{x}} \in \mathcal{L} \subset \mathbb{R}^d$. Second, every vector of the subspace $\mathcal{L}$ is projected onto itself, i.e., $\mathbf{P}\hat{\mathbf{x}} = \hat{\mathbf{x}}$. This is equivalent to $\mathbf{P}^2 = \mathbf{P}$, which means that $\mathbf{P}$ is idempotent. The vector $\mathbf{x}$ can be presented as the sum of two vectors, $\hat{\mathbf{x}} = \mathbf{P}\mathbf{x}$ which belongs to the subspace $\mathcal{L}$, and $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}} = (\mathbf{I} - \mathbf{P})\mathbf{x}$ which is the residual. Figure 4.2 schematically displays these relations. The projection matrix turns out to be another useful way

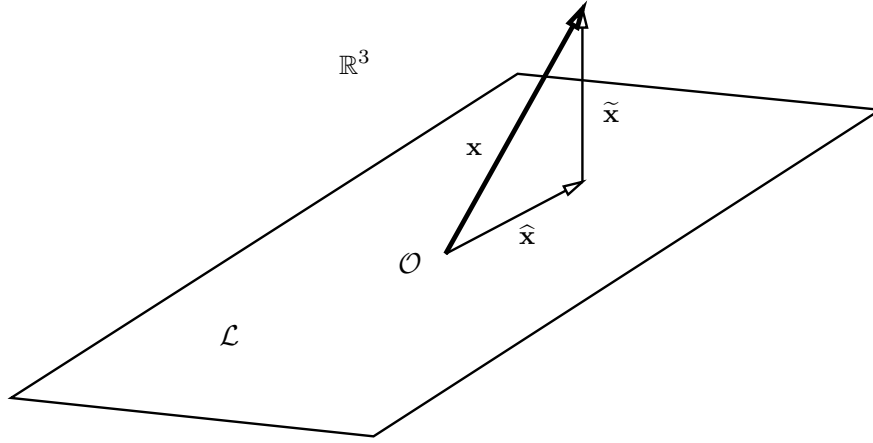


Figure 4.2: The orthogonal projection of a three-dimensional vector $\mathbf{x}$ on a two-dimensional subspace $\mathcal{L}$. $\hat{\mathbf{x}}$ is the projection and $\tilde{\mathbf{x}}$ the residual.

of defining a subspace. Unlike the set of basis vectors, $\mathbf{U}$, the projection matrix $\mathbf{P}$ is uniquely defined. The projection matrix definition of the subspace relies on the fact that the subspace is not affected by the projection onto itself, i.e.,

$$\mathcal{L} = \{ \mathbf{x} \mid \mathbf{P}\mathbf{x} = \mathbf{x} \} . \quad (4.2)$$

The combination of (4.1) and (4.2) shows that $\mathbf{P}\mathbf{U}\mathbf{z} = \mathbf{U}\mathbf{z}$, which leads to the relation between the $\mathbf{P}$ and $\mathbf{U}$ matrices in the general case. That is, the columns $\mathbf{u}_i$ of $\mathbf{U}$ are those eigenvectors of $\mathbf{P}$ the corresponding eigenvalues of which are equal to one.

If the projection $\mathbf{P}$ is required to be such that the norm of the residual component, $\|\tilde{\mathbf{x}}\|$, is minimized, orthogonal projection, for which the projection vector $\hat{\mathbf{x}}$ and the residual vector $\tilde{\mathbf{x}}$ are mutually perpendicular, results. Due to many desirable properties provided by the orthogonal projection, it is the most widely used in subspace algorithms. Most importantly, the multipliers of the basis vectors in (4.1) are obtained as inner products $\zeta_i = \mathbf{x}^T \mathbf{u}_i$. Moreover, the change of the multiplication order leads to the determination of the projection matrix $\mathbf{P}$ of the subspace $\mathcal{L}_{\mathbf{U}}$ from

$$\hat{\mathbf{x}} = \sum_{i=1}^{\ell} \zeta_i \mathbf{u}_i = \sum_{i=1}^{\ell} (\mathbf{x}^T \mathbf{u}_i) \mathbf{u}_i = \sum_{i=1}^{\ell} (\mathbf{u}_i \mathbf{u}_i^T) \mathbf{x} = \mathbf{U} \mathbf{U}^T \mathbf{x} = \mathbf{P} \mathbf{x} . \quad (4.3)$$

Thus, in the case of an orthonormal basis $\mathbf{U}$, the orthogonal projection matrix turns out to be $\mathbf{P} = \mathbf{U}\mathbf{U}^T$ and $\mathbf{P}^T = \mathbf{P}$. This shows that the projection matrix is not only idempotent but also symmetric. An important quantity in the subspace methods is the length of the projection vector $\hat{\mathbf{x}}$. In some situations, only the norm $\|\hat{\mathbf{x}}\|$ is needed and the calculation of the actual projection vector $\hat{\mathbf{x}}$ can be omitted. The squared norm $\|\hat{\mathbf{x}}\|^2$ may be calculated from many different formulations, but most likely from

$$\|\hat{\mathbf{x}}\|^2 = \|\mathbf{U}^T \mathbf{x}\|^2 = \sum_{i=1}^{\ell} (\mathbf{x}^T \mathbf{u}_i)^2 = \sum_{i=1}^{\ell} \zeta_i^2. \quad (4.4)$$

4.1.2 Classification Rule

In this section, the classification function $g(\mathbf{x})$ introduced in Chapter 3 is specialized for subspace methods. In the classifier formulation, the subspaces and associated matrices are subscripted with the class index j . The length of the projection $\hat{\mathbf{x}}_j$ on the subspace $\mathcal{L}_j$ is used as a similarity measure between the input vector $\mathbf{x}$ and the class j . The input vector is then classified according to the maximal similarity value

$$g_{\text{ss}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmax}} \|\hat{\mathbf{x}}_j\|^2 = \underset{j=1,\dots,c}{\operatorname{argmax}} \|\mathbf{U}_j^T \mathbf{x}\|^2. \quad (4.5)$$

The subscript ‘SS’ acts as an abstract placeholder for the still unspecified way of how to select for each class j the dimension ℓ_j and the actual basis vectors $\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_j j}$ which form the subspace basis $\mathbf{U}$. (4.5) shows the fundamental linearity of the subspace classification methods: the length of the input vector $\mathbf{x}$ does not contribute to the classification decision. In other words, the subspace classifier is *invariant* to the input vector length in the sense discussed in Section 2.5.

Two potential weaknesses of the subspace classification rule are evident in (4.5). First, the information on the *a priori* probabilities of the classes cannot be utilized even if it were available. Second, the classification of vectors residing near the origin of the coordinate system may be arbitrary in the presence of additive noise. These questions are addressed in Sections 4.3.6 and 4.3.5, respectively.

4.1.3 CLAFIC

The employment of the *Principal Component Analysis* (PCA), or the *Karhunen-Loève Transform* (KLT), in classification tasks leads to the *Class-Featuring Information Compression* (CLAFIC) algorithm introduced by Watanabe et al. (1967). CLAFIC simply forms the base matrices for the classifier subspaces from the eigenvectors of the class-conditional correlation matrices. For each class j , the correlation matrix $\mathbf{R}_j = E[\mathbf{x}\mathbf{x}^T \mid \mathbf{x} \in j]$ is estimated with $\hat{\mathbf{R}}_j = n_j^{-1} \sum_{i=1}^{n_j} \mathbf{x}_{ij} \mathbf{x}_{ij}^T$. The first ℓ_j eigenvectors of $\hat{\mathbf{R}}_j$, $\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_j j}$, in the order of decreasing eigenvalue λ_{ij} , are then used as columns of the basis matrix $\mathbf{U}_j$,

$$\mathbf{U}_j = \left(\mathbf{u}_{ij} \mid (\hat{\mathbf{R}}_j - \lambda_{ij} \mathbf{I}) \mathbf{u}_{ij} = \mathbf{0}, \lambda_{ij} \geq \lambda_{(i+1)j}, i = 1, \dots, \ell_j \right), \quad (4.6)$$

where $\mathbf{0}$ is the zero vector. The sample mean $\hat{\boldsymbol{\mu}}$ of the pooled training set is normally subtracted from the pattern vectors before they are classified or used in initializing the CLAFIC classifier. Because the class-conditional correlations $\mathbf{R}_j$ of the input vectors $\mathbf{x}$ differ from the corresponding class-wise covariances $\boldsymbol{\Sigma}_j$, the first eigendirection in each class merely reflects the direction of the class mean from the pooled mean translated to the origin. The calculation of the eigenvalues and eigenvectors of a symmetric positive definite matrix, such as $\hat{\mathbf{R}}_j$, is described, for instance, by Golub and van Loan (1989). The selection of the subspace dimensions $\ell_1 \dots, \ell_c$ is left open in the basic formulation of CLAFIC. This important question is addressed in Section 4.3.2. Until then, it can be simply assumed that the dimensions are somehow fixed to appropriate values.

4.1.4 Multiple Similarity Method

One way to generalize the classification function (4.5) is to introduce individual weights for all the basis vectors. Iijima et al. (1973) have selected to weight each basis vector with the corresponding eigenvalue in their *Multiple Similarity Method* (MSM)

$$g_{\text{MSM}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} \sum_{i=1}^{\ell_j} \frac{\lambda_{ij}}{\lambda_{1j}} (\mathbf{x}^T \mathbf{u}_{ij})^2. \quad (4.7)$$

This emphasizes the effect of the most prominent directions, for which $\lambda_{ij}/\lambda_{1j} \lesssim 1$. The selection of the subspace dimension ℓ_j is, therefore, less essential because the influence of the less prominent eigenvectors, which have multipliers $\lambda_{ij}/\lambda_{1j} \gtrsim 0$, diminishes gradually. The influence of the phantom eigendirections which have small eigenvalues and are created by additive noise is thus canceled out. Section 4.3.3 puts forward a new improved version of this scheme.

4.1.5 Method of Orthogonal Subspaces

The subspaces that represent two different pattern classes may have a large common subspace. This is problematic because the discrimination between these classes weakens if the subspace dimensions ℓ_j are small. On the other hand, if the subspace dimensions are increased, the classification decisions become dominated by the less robust principal directions. This problem may be avoided if the subspaces are made mutually orthogonal. This leads to a variant of the CLAFIC known as the *Method of Orthogonal Subspaces* (MOSS) by Kulikowski and Watanabe (1970) and Watanabe and Pakvasa (1973).

Pairwise orthogonalization of two subspaces is possible whenever their dimensions satisfy the obvious condition $\ell_i + \ell_j \leq d$. In that case, two subspaces are said to be mutually orthogonal if any vector of one of the subspaces has zero projection on the other, and *vice versa*. This is equal to the condition that the basis vectors are orthogonal not only within, but also between, the subspaces. Thus, the projection matrices $\mathbf{P}_i$ and $\mathbf{P}_j$ of two

orthogonal subspaces fulfill the condition

$$\mathbf{P}_i \mathbf{P}_j = \mathbf{P}_j \mathbf{P}_i = \mathbf{0} , \quad (4.8)$$

where $\mathbf{0}$ is the zero matrix. The orthogonalization process of MOSS is accomplished by removing the intersections of the subspaces as described, for instance, by (Therrien 1975). In short, the projection operators $\mathbf{P}_j$ are replaced with mutually orthogonal operators $\mathbf{P}'_j$, which are formed by using the *generating matrix* $\mathbf{G}_j$,

$$\mathbf{G}_j = a_j \mathbf{P}_j + \sum_{i=1, i \neq j}^c a_i (\mathbf{I} - \mathbf{P}_i) . \quad (4.9)$$

The otherwise arbitrary positive multipliers a_j must satisfy the condition $\sum_{j=1}^c a_j = 1$. The eigenvalues and eigenvectors are now calculated from $\mathbf{G}_j$, and the orthogonal projection operators $\mathbf{P}'_j$ are formed from the ℓ'_j eigenvectors $\mathbf{v}_{ij}$ which have eigenvalues equal to one,

$$\mathbf{P}'_j = \sum_{i=1}^{\ell'_j} \mathbf{v}_{ij} \mathbf{v}_{ij}^T . \quad (4.10)$$

Naturally, $\forall j : \ell'_j \leq \ell_j$. In some cases, the procedure, however, leads to an unacceptable situation where, for some j , $\ell'_j = 0$, and the corresponding subspace vanishes (Karhunen and Oja 1980).

4.1.6 Generalized Fukunaga-Koontz Method

Fukunaga and Koontz (1970) reasoned that it was necessary to select such basis vectors that the projections on rival subspaces were minimized. Their original formulation of the problem and the criticism against it, presented by Foley and Sammon (1975), considered only the two-class case. Instead, the *Generalized Fukunaga-Koontz Method* (GFK) of Kittler (1978) handles an arbitrary number of classes. In the two-class case, the correlation matrices of both classes are first estimated. The Karhunen-Loève Transform is then applied to their sum $\mathbf{Q} = \mathbf{R}_1 + \mathbf{R}_2$ and the eigenvalues λ_i and eigenvectors $\mathbf{u}_i$ are used in defining a transformation matrix $\mathbf{S}$, which is used to transform the original vector $\mathbf{x}$ to $\mathbf{x}'$,

$$\mathbf{S} = \left(\frac{\mathbf{u}_1}{\sqrt{\lambda_1}} \dots \frac{\mathbf{u}_d}{\sqrt{\lambda_d}} \right) . \quad (4.11)$$

For the correlation matrix $\mathbf{R}'_j$ of the transformed vector $\mathbf{x}' = \mathbf{S}^T \mathbf{x}$, it holds that $\mathbf{R}'_j = \mathbf{S}^T \mathbf{R}_j \mathbf{S}$, and further $\mathbf{R}'_1 + \mathbf{R}'_2 = \mathbf{I}$. Thus, $\mathbf{R}'_1$ and $\mathbf{R}'_2$ have the same eigenvectors, and the corresponding eigenvalues are positive and sum up to unity. This leads to the following interpretation of the nature of the eigenvectors: When ordered according to the descending eigenvalues, the first few eigenvectors of $\mathbf{R}'_1$ are optimal for describing the distribution of the transformed vectors $\mathbf{x}'$ which belong to the first class. On the other hand, the

eigenvectors with the smallest eigenvalues describe the second class. The method was primarily developed for feature extraction and clustering, but it also lends itself directly to classification.

The multi-class version of the algorithm minimizes simultaneously the length of the residual component $\tilde{\mathbf{x}}$ and the sum of the projection lengths on the other subspaces. The minimization may be carried out individually for each class j and its basis vectors $\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_j j}$

$$e_{\text{GFK}j} = \sum_{i=\ell_j+1}^d \mathbf{u}_{ij}^T \left[\mathbf{R}_j + \sum_{\substack{k=1 \\ k \neq j}}^c (\mathbf{I} - \mathbf{R}_k) \right] = \min_{\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_j j}} !. \quad (4.12)$$

Thus, an optimal basis for the class j may be formed from the eigenvectors that correspond to the largest eigenvalues of the generating matrix

$$\mathbf{G}_j = \mathbf{R}_j + \sum_{\substack{i=1 \\ i \neq j}}^c (\mathbf{I} - \mathbf{R}_i). \quad (4.13)$$

The Generalized Fukunaga-Koonz Method may be considered to first model the pooled distribution of all the classes and then to subtract the effect of rival classes from that of the correct one. Acting on the covariance matrices, it is also a kind of a predecessor of the *Averaged Learning Subspace Method* (ALSM), to be described in Section 4.2.3.

4.2 Basic Learning Subspace Methods

The subspace classifiers introduced so far have all been parametric models in the sense that the subspaces are selected in a manner which best fits to the assumed linear model for the classes. The three algorithms investigated in this section, on the contrary, are capable of learning in decision-directed fashion. Their ability to enhance the classification accuracy during a training period makes them better-suited for pattern recognition tasks than the previous methods, whose origins lie more in optimal reconstruction of input vectors.

4.2.1 Category Feature Subspace Method

One of the oldest decision-directed subspace classification algorithms is the *Category Feature Subspace Method* (CAFESSM), described by Noguchi (1976 and 1978). In CAFESSM, some vectors are selected from each class distribution to represent the whole class as a generally nonorthonormal subspace basis. Only linear independence of the basis is now assumed, and, therefore, the projection matrix $\mathbf{P}_j$ needs to be restated employing the matrix pseudo-inverse,

$$\mathbf{P}_j = \mathbf{U}_j \mathbf{U}_j^\dagger = \mathbf{U}_j (\mathbf{U}_j^T \mathbf{U}_j)^{-1} \mathbf{U}_j^T. \quad (4.14)$$

The CAFESSM algorithm actually does not describe how to add and remove vectors to, and from, the basis in order to increase the explaining power of the base. Various iterative algorithms that use some sort of heuristic reasoning may be applied. The virtue of the CAFESSM algorithm is in the possibility of iterative refinement of the vector set that determines the subspaces. The selection of the basis does not need to reflect the actual distribution of the class. Instead, the accuracy of classification is the ultimate optimization goal.

4.2.2 Learning Subspace Method

The *Learning Subspace Method* (LSM) was introduced by Kohonen et al. (1978 and 1979). The basic idea of LSM is to initially set up the classifier with CLAFIC and then *rotate* the subspaces during the training phase in order to diminish the number of misclassified vectors. The decision-directed rotations make the projections longer on the correct subspaces and shorter on the others. This process leads to an optimized selection of the classification subspaces in terms of classification error.

For any input vector $\mathbf{x}$, there are always two subspaces of interest. The first is its *own* subspace, i.e., the one that $\mathbf{x}$ belongs to. The second is the *rival* subspace, i.e., the best runner-up subspace. Subscripts o and r are used to denote these two subspaces, $\mathcal{L}_o$ and $\mathcal{L}_r$. According to the classification rule of (4.5), $\mathbf{x}$ is misclassified when the rival projection length $\|\hat{\mathbf{x}}\|_r$ exceeds the length $\|\hat{\mathbf{x}}\|_o$ of the projection on the own subspace. This can be corrected either by forcing $\|\hat{\mathbf{x}}\|_o$ to be larger, or by making $\|\hat{\mathbf{x}}\|_r$ smaller, or both. The rotation operator used in changing the projection length is applied to the basis vectors of the subspace and is solely determined by the input vector $\mathbf{x}$. A general form for such a rotation transforming the subspace basis $\mathbf{U}$ to a new basis $\mathbf{U}'$ is

$$\mathbf{U}' = (\mathbf{I} + \mu \mathbf{x} \mathbf{x}^T) \mathbf{U} . \quad (4.15)$$

Positive values of μ turn the basis vectors toward $\mathbf{x}$, making the projection vector longer, whereas small negative μ shorten the projection length. In the special case where $\mu = -(\mathbf{x}^T \mathbf{x})^{-1}$, the new basis is orthogonal to $\mathbf{x}$, and the projection vanishes. Kohonen et al. (1978) derived a closed form solution for μ in the case where the desired projection length $\|\hat{\mathbf{x}}'\|$ is given. Omitting the details, the result is

$$\mu = \frac{1}{\|\mathbf{x}\|^2} \left(\frac{\|\hat{\mathbf{x}}'\|}{\|\hat{\mathbf{x}}\|} \sqrt{\frac{\|\mathbf{x}\|^2 - \|\hat{\mathbf{x}}\|^2}{\|\mathbf{x}\|^2 - \|\hat{\mathbf{x}}'\|^2}} - 1 \right) . \quad (4.16)$$

Various learning functions have been proposed for LSM. Most of them can be formulated as monotonically increasing and piecewise defined functions in a $\xi\xi'$ -plane, where $\xi = \|\hat{\mathbf{x}}\|_o - \|\hat{\mathbf{x}}\|_r$ and $\xi' = \|\hat{\mathbf{x}}'\|_o - \|\hat{\mathbf{x}}'\|_r$ (Riittinen 1986). Note that the rotated bases $\mathbf{U}'_o$ and $\mathbf{U}'_r$ are no longer orthonormal and need to be re-orthonormalized through the Gram-Schmidt process.

4.2.3 Averaged Learning Subspace Method

The *Averaged Learning Subspace Method* (ALSM) was introduced by Kuusela and Oja (1982) and described comprehensively by Oja (1983). It is an iterative learning version of CLAFIC – or a batch version of LSM, depending on how one wants to look at it. In ALSM, the class-conditional *scatter matrices*, i.e., unnormalized correlation matrices, are modified after errors. The subspaces are then recomputed after every training epoch. The algorithm was originally derived from LSM by replacing the rotations with their expected values, and noting that the rotations change the basis vectors toward the dominant eigenvectors of the expected rotation matrices. Such an iteration can then be speeded up by performing the corrective learning operations on the class-conditional scatter matrices instead of on the subspaces, and recalculating the eigenvectors of the scatter matrices after each epoch. Thus, the entire ALSM algorithm is

1. The initial scatter matrices $\widehat{\mathbf{S}}_j$ for the epoch $t = 0$ are computed for all the classes j according to

$$\widehat{\mathbf{S}}_j(0) = \sum_{i=1}^{n_j} \mathbf{x}_{ij} \mathbf{x}_{ij}^T = n_j \widehat{\mathbf{R}}_j. \quad (4.17)$$

2. The matrices $\mathbf{U}_j(t)$ of basis vectors for the subspaces $\mathcal{L}_j(t)$ are produced from the eigenvectors of the $\widehat{\mathbf{S}}_j(t)$ matrices similarly to CLAFIC in (4.6).
3. All the training vectors are tentatively classified, and corrections are applied to the scatter matrices by the rule

$$\widehat{\mathbf{S}}_j(t+1) = \widehat{\mathbf{S}}_j(t) + \alpha \sum_{\mathbf{x}_{ij} \in A_j(t)} \mathbf{x}_{ij} \mathbf{x}_{ij}^T - \beta \sum_{\mathbf{x}_{ik} \in B_j(t)} \mathbf{x}_{ik} \mathbf{x}_{ik}^T, \quad (4.18)$$

where $A_j(t)$ is the set of vectors $\mathbf{x}_{ij}$ of class j which during the training epoch t are classified using (4.5) erroneously to some other class. The set $B_j(t)$ consists of vectors $\mathbf{x}_{ik}$ which during that epoch are classified erroneously as j . The small positive multipliers α and β control the size of the scatter matrix correction.

4. The iteration is continued from Step 2 with t incremented by one, unless a predefined number of training epochs has been reached, or the classification accuracy starts to decrease.

The corrections of the subspaces are carried out in batch fashion, which makes the final result independent of the presentation order of the vectors. This property provides increased statistical stability compared to the LSM algorithm. The speed and stability of the ALSM learning depends on the selection of the parameters α and β . In general, they can be kept equal and allowed to decay with the iteration step. The selection of the dimensions ℓ_j can either be done once and for all before starting the training, or they can be reselected after each training epoch. The training of the classifier can be continued with the LSM algorithm after the ALSM phase. If the recognition accuracy decreases, the original ALSM subspaces can be used in the application. The ALSM method has been used, for example, in classification of visual textures and colors by Parkkinen (1989).

4.2.4 Neural Network Interpretation of Subspace Learning

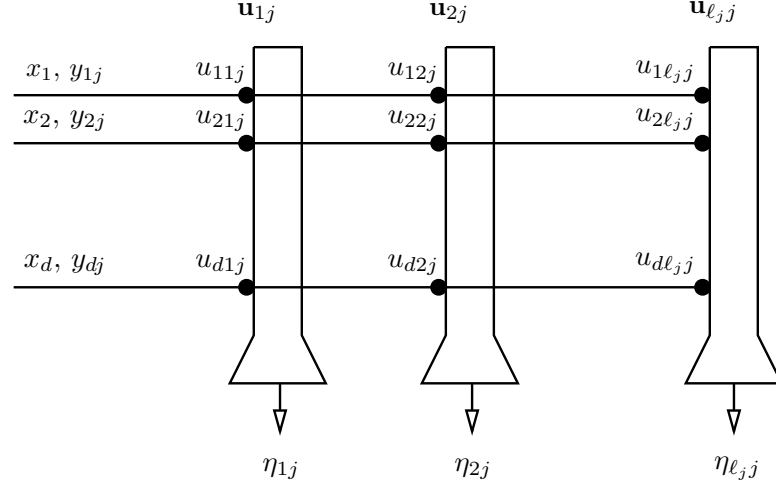


Figure 4.3: The Subspace Network according to Oja (1989).

Figure 4.3 displays the *Subspace Network*, a neural interpretation of learning subspace methods by Oja (1989). In the Subspace Network model for a subspace classifier, each class j is represented as a cluster of ℓ_j basic neurons $\mathbf{u}_{ij}$ that have d input connections with individual synaptic weights u_{kij} . The activation ζ_{ij} , the differentials of the neuron outputs η_{ij} , the cluster feedback y_{kj} , and the differentials of the weights are expressed with a set of equations

$$\zeta_{ij} = \sum_{k=1}^d u_{kij} x_k = \mathbf{u}_{ij}^T \mathbf{x}, \quad (4.19)$$

$$\frac{d\eta_{ij}}{dt} = \zeta_{ij} - \sigma^{-1}(\eta_{ij}), \quad (4.20)$$

$$y_{kj} = \sum_{i=1}^{\ell_j} u_{kij} \zeta_{ij}, \text{ and} \quad (4.21)$$

$$\frac{du_{kij}}{dt} = (x_k - y_{kj}) \alpha \zeta_{ij}. \quad (4.22)$$

In (4.20), $\sigma^{-1}(\cdot)$ is the inverse of the sigmoid function, and α in (4.22) is a positive coefficient that controls the strength of the feedback and, thus, the speed of the learning. The time-constant of the neuron-wise feedback in the output differential (4.20) is small and η_{ij} rapidly settles to its steady state $\eta_{ij} = \sigma(\zeta_{ij})$. On the contrary, the changes in the synaptic weights u_{kij} take place very slowly.

Comparison of (4.19) and (4.21) to the previous matrix notation in (4.3) shows that the feedbacks $y_{1j}, \dots, y_{dj}$ form the projection vector $\hat{\mathbf{x}}_j$, provided that the vector set $\{\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_jj}\}$ that forms the matrix $\mathbf{U}_j$ is orthonormal. Actually, the orthonormality

condition is fulfilled. If the vectors $\mathbf{x}_{ij}$ of class j are assumed to be stationary, then the average of the differential equation (4.22) can be expressed in matrix form by using the class-conditional autocorrelations $\mathbf{R}_j$:

$$\frac{d\mathbf{U}_j}{dt} = \alpha(\mathbf{R}_j\mathbf{U}_j - \mathbf{U}_j(\mathbf{U}_j^T\mathbf{R}_j\mathbf{U}_j)) . \quad (4.23)$$

The analysis of such differential systems has shown, according to Oja and Karhunen (1985), Oja (1989), Yan et al. (1994), and Wyatt and Elfadel (1995), that the columns of $\mathbf{U}_j$ approach orthonormality as t grows. Furthermore, in the limiting case, they span the same subspace of $\mathbb{R}^d$ spanned by the ℓ_j dominant eigenvectors of $\mathbf{R}_j$. Thus, the formation of the CLAFIC type classifier subspaces can be motivated neurally. The ALSM learning can be accommodated in (4.23) by setting the coefficient α to a negative value in the case of misclassification. This, of course, needs external supervision in training, which is not an inherent property of the Subspace Network.

4.3 Modifications on Subspace Classifiers

This section addresses some general questions considering the actual use and implementation of the subspace methods. These include the use of class-specific means in classification, the selection of the dimensions for the subspaces, the use of more than one subspace to model each class, and the unfortunate special case where one of the classes occupies a region around the origin. Then, a novel interpretation of the subspace methods in terms of gamma-distributed probability density functions is put forward. Finally, the computational requirements and possibilities of implementing the subspace classifiers are analyzed.

4.3.1 Using Class-Specific Means

Page 51 stated that the sample mean $\hat{\boldsymbol{\mu}}$ of the entire training sample is normally first subtracted from the training vectors $\mathbf{x}_i$ before the class-conditional correlations $\hat{\mathbf{R}}_j$ are formed. Similar subtraction is then performed on all the vectors of the testing set before classification. Thus, the origin of the coordinate system is in every situation first transformed to $\hat{\boldsymbol{\mu}}$ of the training sample. This practice implies quite a strong assumption about the linearity of the class. A more flexible architecture can be obtained by modeling each class individually as a *linear manifold*, centered at the estimated mean $\boldsymbol{\mu}_j$ of the class. For this modification, some notations need to be restated. The classification rule (4.5) can no longer be based on maximizing the projection length. Instead, the minimum of the residual length $\|\tilde{\mathbf{x}}_j\|$ is searched for. Actually, these two procedures are equal when class-conditional means are not used.

In the general form, and further in the special case of orthogonal projection matrix $\mathbf{P}_j = \mathbf{U}_j \mathbf{U}_j^T$, the modified classification function is

$$g_{\text{SS}-\boldsymbol{\mu}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmin}} \|\tilde{\mathbf{x}}_j\|^2 = \underset{j=1,\dots,c}{\operatorname{argmin}} \|(\mathbf{I} - \mathbf{P}_j)(\mathbf{x} - \boldsymbol{\mu}_j)\|^2 \quad (4.24)$$

$$= \underset{j=1,\dots,c}{\operatorname{argmin}} (\|(\mathbf{x} - \boldsymbol{\mu}_j)\|^2 - \|\mathbf{U}_j^T(\mathbf{x} - \boldsymbol{\mu}_j)\|^2) . \quad (4.25)$$

The suffix $-\boldsymbol{\mu}$ in the classifier name specifies that the individual class-means are subtracted. In the case of CLAFIC, the basis vectors $\mathbf{u}_{ij}$ are now the eigenvectors of the class-conditional covariance matrix $\boldsymbol{\Sigma}_j$. The mean subtraction principle is generally combinable with any subspace classification method. To some extent, it resembles the *Soft Independent Modelling of Class Analogy* (SIMCA) method by Wold (1976).

The intersections of the class regions produced by (4.24) are no longer cone-shaped, as are the ones produced by the original classification function (4.5). Instead, the classes intersect in unanticipated regions of the feature space. In the vicinity of the intersections, the classification decision becomes more or less arbitrary. In general, however, provided that the pattern dimensionality d is large enough in relation to the sum of the subspace dimensions ℓ_j , the intersections will most likely occur in regions where the input density is sparse.

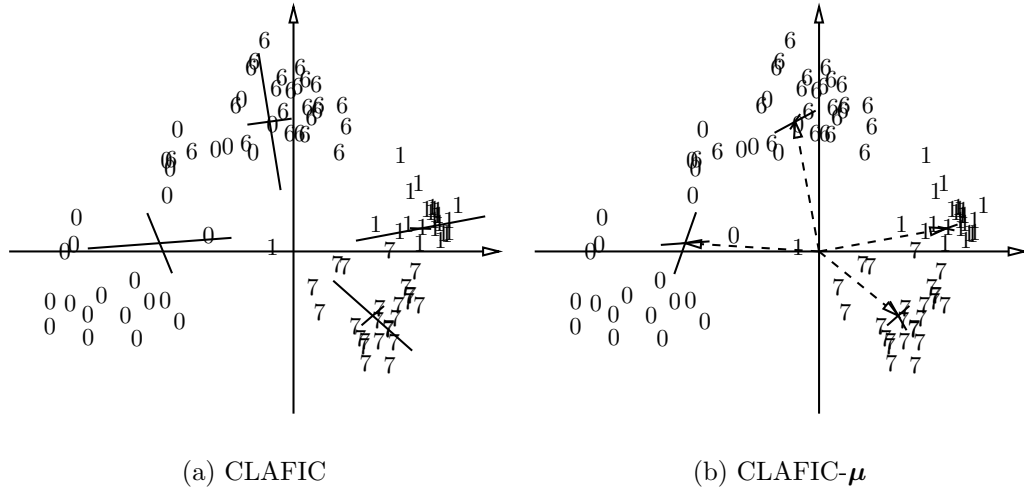


Figure 4.4: Handwritten digit classes projected on a plane. In case (a), the subspace basis vectors are translated to the center of the corresponding class for visualization only. In (b), the subspaces are actually formed around the centers.

Figure 4.4 illustrates how three classes of handwritten digits are located in the first two dimensions of the Karhunen-Loève Transformed data. In case (a), the class-conditional means are not subtracted. Therefore, the first eigendirection of each class quite clearly points toward the origin of the coordinate system. In (b), the class means plotted with dashed arrows are subtracted, and the eigendirections now model the two-dimensional class shapes more precisely.

4.3.2 Selection of Subspace Dimensions

The important question of how the subspace dimensions ℓ_j can be selected was deferred in Section 4.1.3. Various general criteria exist for model order selection, including *Minimum Description Length* (MDL) by (Rissanen 1978) and *Akaike Information Criteria* (AIC) by (Akaike 1974).

In this section, two approaches are investigated. Both methods need one parameter value to be fixed, for example, by using training set cross-validation. Firstly, the simplest possibility is to set all the ℓ_j s to be equal to a value ℓ . Then, the optimal value for this integer scalar is searched for. Typically, a U-shaped error rate curve $\epsilon(\ell)$, such as those seen in Figure 8.1 on page 121, follows. The selection of ℓ is then straightforward.

Secondly, the selection can be based on the non-increasing series $\{\lambda_{1j}, \dots, \lambda_{dj}\}$ of the eigenvalues of the autocorrelation matrix $\hat{\mathbf{R}}_j$ in (4.6). There are two alternatives. The first is to select ℓ_j so that the corresponding eigenvalue of the first eigenvector left out from each of the bases, $\lambda_{\ell_j+1,j}$, is less than a preset proportion $\kappa_1 \in (0, 1)$ of the sum of all the eigenvalues,

$$\frac{\lambda_{\ell_j j}}{\sum_{i=1}^d \lambda_{ij}} > \kappa_1 \geq \frac{\lambda_{\ell_j+1,j}}{\sum_{i=1}^d \lambda_{ij}}. \quad (4.26)$$

The second option is to set the residual sum of the left-out eigenvalues, i.e., the expected squared residual length $E\|\tilde{\mathbf{x}}\|^2$, to be less than a preset proportion κ_2 common to all the classes,

$$\frac{\sum_{i=\ell_j}^d \lambda_{ij}}{\sum_{i=1}^d \lambda_{ij}} > \kappa_2 \geq \frac{\sum_{i=\ell_j+1}^d \lambda_{ij}}{\sum_{i=1}^d \lambda_{ij}}. \quad (4.27)$$

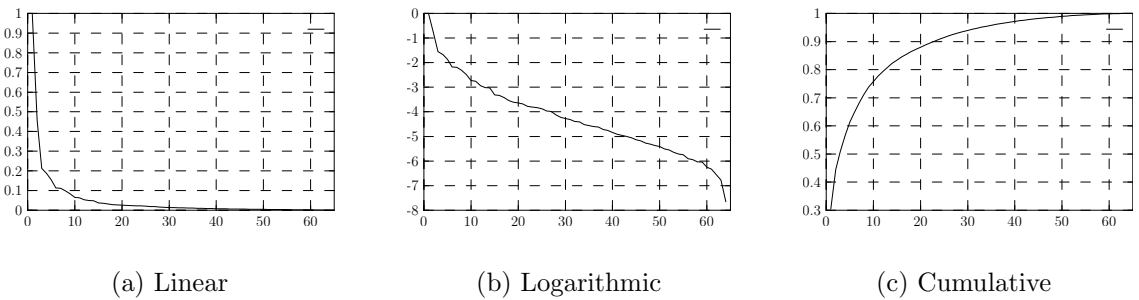


Figure 4.5: The eigenvalues of the correlation matrix of handwritten digits ‘7’. The ordinates from left to right are $\lambda_{i,7}/\lambda_{1,7}$, $\log(\lambda_{i,7}/\lambda_{1,7})$, and $\sum_{k=1}^i \lambda_{k,7} / \sum_{k=1}^d \lambda_{k,7}$, while index i is shown on the abscissa.

Figure 4.5 displays some curves obtained from the actual eigenvalues of a correlation matrix of a class of handwritten digits. The selection of κ_1 in (4.26) corresponds to setting a horizontal threshold level in Figure 4.5(a), or its logarithm counterpart in Figure 4.5(b).

The decay of the eigenvalues can be seen to be so smooth that, by visual inspection, it is not obvious how to set a decisive value $\ell_j = i$, above which the eigenvalues would suddenly be approximately equal to zero and the corresponding eigenvectors could be discarded. Similarly, the selection of κ_2 in (4.27) corresponds to setting a threshold on level $1 - \kappa_2$ in Figure 4.5(c). This seems to be a problematic task as well. Using either of these schemes leads again to a U-shaped error rate curve $\epsilon(\kappa)$, this time with a real-valued abscissa. The smaller the selected value κ is, the larger are the subspace dimensions ℓ_j for all the classes. The κ_2 -type selection was presented together with CLAFIC by Watanabe et al. (1967), whereas the κ_1 -scheme was introduced by Kohonen et al. (1980).

Any criterion used to select the subspace dimensions simultaneously to all the classes can be considered to produce only an initial guess. The subspace dimensions can then be modified further by observing how classification errors are distributed among the classes. Using the notations introduced on page 55, an iteration rule suggested by Laaksonen and Oja (1996b) can be formulated as follows:

$$j = \operatorname{argmax}_{i=1,\dots,c} |\#A_i(t) - \#B_i(t)|, \quad (4.28)$$

$$\ell_j(t+1) = \ell_j(t) + \frac{\#A_j(t) - \#B_j(t)}{|\#A_j(t) - \#B_j(t)|}. \quad (4.29)$$

The operator “#” stands for the cardinality of a set. Again, the sets $A_j(t)$ and $B_j(t)$ are the sets of vectors misclassified from, and to, the class j during the epoch t , as described above with ALSM. Thus, in each iteration step t , one of the ℓ_j s is either increased or decreased by one, and all the other subspace dimensions are preserved. An account needs to be established for all the used combinations of the ℓ_j s. If the system is about to select a combination which has already been used, the second-best selection for j in (4.28) is employed. The iteration is stopped after a fixed number of trials, and the combination which gave the lowest overall error rate is re-established. The usefulness of the iteration may be insignificant if either the number of classes, c , or the input vector dimensionality, d , is small.

4.3.3 Weighted Projection Measures

Section 4.1.4, which introduced the MSM classifier, already presented one form of non-linearity added to the subspace classification rule (4.5): fixed coefficients $\lambda_{ij}/\lambda_{1j}$ were included in the summation of the inner products. In a more general setting, weights w_{ij} can be selected by using other criteria. In this section, the contents of the previous section are first reinterpreted by using the general weighting scheme. Two ways of selecting the w_{ij} s are suggested.

Adding individual weights to all the terms in the classification rule (4.5) leads to

$$g_{w\text{-SS}}(\mathbf{x}) = \operatorname{argmax}_{j=1,\dots,c} \sum_{i=1}^d w_{ij} (\mathbf{u}_{ij}^T \mathbf{x})^2. \quad (4.30)$$

In (4.30), the last summation index, which in (4.5) was ℓ_j , is replaced with d . This notational change allows the reformulation of the selection of the ℓ_j s. The basic CLAFIC algorithm can be reproduced from (4.30), if the weights w_{ij} are defined

$$w_{ij} = \begin{cases} 1, & \text{iff } i \leq \ell_j, \\ 0, & \text{iff } i > \ell_j. \end{cases} \quad (4.31)$$

The problem of selecting the ℓ_j s still remains in the following two schemes of setting the weights. The optimization rules used in selecting the weights are based on iteration, and these iterations should occur alternately with the ℓ_j -optimization in order to produce the best results.

The first scheme (Kohonen 1996) originates from an observation that if some of the subspaces produce systematically projections that are too long or too short, they benefit or suffer in the classification decision. This leads to considering adding a unique multiplier to each class. Thus, setting $w_{1j} = \dots = w_{\ell_j j} = w_j$ reduces the problem to c constants w_j to be settled, for example, by using standard optimization techniques.

In the second scheme, the w_{ij} s are computed from the correlation matrix eigenvalues $w_{ij} = \lambda_{ij}$ in a manner inspired by MSM. A somewhat more general formulation than the original MSM is to select a parameter γ and to set an exponential form

$$w_{ij} = \begin{cases} \left(\frac{\lambda_{ij}}{\lambda_{1j}} \right)^\gamma, & \text{iff } i \leq \ell_j, \\ 0, & \text{iff } i > \ell_j. \end{cases} \quad (4.32)$$

Setting $\gamma = 0$ reduces (4.30) to the conventional CLAFIC and $\gamma = 1$ makes it equal to MSM, whereas selections $\gamma < 0$ lead to some degree of variance normalization or data-whitening within each subspace. The value for γ can, thus, be experimentally selected from the range of $(-1, 1)$. The kind of nonlinearity injected into the subspace classification in this section can be considered, because the inner products $\mathbf{u}_{ij}^T \mathbf{x}$ are only multiplied, and no functional form of nonlinearity is applied to them. More general forms of strong nonlinearity are an important topic of ongoing research, such as in the field of *Independent Component Analysis* (ICA), (Jutten and Hérault 1991; Comon 1994; Oja 1995; Cardoso and Laheld 1996; Karhunen et al. 1997).

4.3.4 Multiple Subspaces per Class

More than one subspace can be used in representing each class. This may be necessary if the distribution of a class cannot be modeled accurately enough with one subspace. For the easy use of this variation, we should know, in advance, how the vectors of each class are divided among the subspaces. This information is normally unavailable, and some heuristic needs to be employed. The overall classification accuracy may be enhanced in this way. The need for additional subspaces, however, runs against the core assumption regarding the linear nature of the pattern classes.

In CLAFIC and ALSM, a modification of the iterative k -means, i.e., the LBG algorithm (Gersho and Gray 1992), can be used to split the training set of each class among its subspaces (Oja and Parkkinen 1984). In LSM, the subspaces can be initialized using different combinations of the class-correlation eigenvectors. Thus, the training sets are not decisively divided. Instead, each vector may traverse from one subspace of its own class to another. In the case of a speaker-independent speech recognition system, the initial division of the training vectors obtained from pronounced phonemes can be based on dividing the speakers into arbitrary groups (Riittinen et al. 1980).

4.3.5 Treatment of Neighborhood of Origin

As a direct consequence of the inherent linearity assumption of the subspace methods, a problem arises when one of the classes resides around, or near, the origin. The linearity assumption states that the lengths of the vectors do not correlate with their classification. Therefore, the classification regions of all the classes extend to the vicinity of the origin. In that region, however, any additive noise in input vectors affects tremendously the direction cosines of the vectors. Consequently, the classification of such vectors becomes unreliable. At the origin itself, the distribution of the noise fully determines the classifications.

Depending on the nature of the data, there may be no acceptable subspace solution to the classification problem. Thus, either the subspace classifier has to be abandoned completely, or, a two-stage classification system has to be formed. In the latter case, some other classifier, such as QDA of Section 3.2.1, can be used first to test the hypothesis of whether the input vector belongs to the class around origin. If that test fails at a predefined risk level, a subspace classifier is used. If the problematic class resides only partly near the origin, it should also have an associated subspace which participates in the second classification stage. The training vector set of the class needs to be split into two subsets. One contains the vectors from the region of the origin and is used in estimating the covariance $\hat{\mathbf{R}}$ in (3.5). The other is used in the initialization of the associated subspace. Some iterative relaxation may be needed in performing this split adequately.

4.3.6 Density Function Interpretation of Subspace Methods

Chapter 3 stated, in general terms, that there is no way to interpret subspace classifiers as density estimation methods. Nevertheless, such an interpretation is sought in this section. The final result is a new classification function which replaces the traditional subspace classification rule (4.5) with one similar to the classification function (3.1) used with density estimation methods of classification. The density function interpretation solves two well-known shortcomings of the standard subspace methods as a by-product. First, it solves the problem of unequal priors, which occurs with standard subspace methods when the *a priori* probabilities of classes are unequal. Second, the new formalism provides a statistically justified rejection mechanism.

Section 3.2 showed that most of the density-function-based classification methods assume the class distributions to be Gaussian. As such, this assumption is not transferable to subspace formalism. Hence, no probability density function interpretation of the subspace classification rule has been presented so far. The approach in this section, however, explains the subspace formalism in terms of the gamma probability density function. First, it is demonstrated that the coefficients ζ_i in (4.1) tend to be zero-mean normally distributed. Second, it is shown that they may be regarded as mutually independent. Third, it is argued that the sum of their squares can be approximated with gamma distribution, the underlying form of the chi-square distribution. Fourth, a reformulated classification rule is presented.

The derivation of the new classification rule is illustrated with real-world data from the prototype system for handwritten digit recognition, which is described in detail in Chapter 7. The derivation is, however, independent from that particular case study, which is used only to visualize the soundness of some steps on the path.

Distribution of Coefficients In (4.1), the ζ_i coefficients were introduced as general multipliers in the subspace formalism. Later, in CLAFIC, their role became that of principal components in the Karhunen-Loève expansion of the class distribution. This section considers the actual distributions of the ζ_i coefficients. The objective of the analysis is to show that the distributions can be regarded as Gaussian.

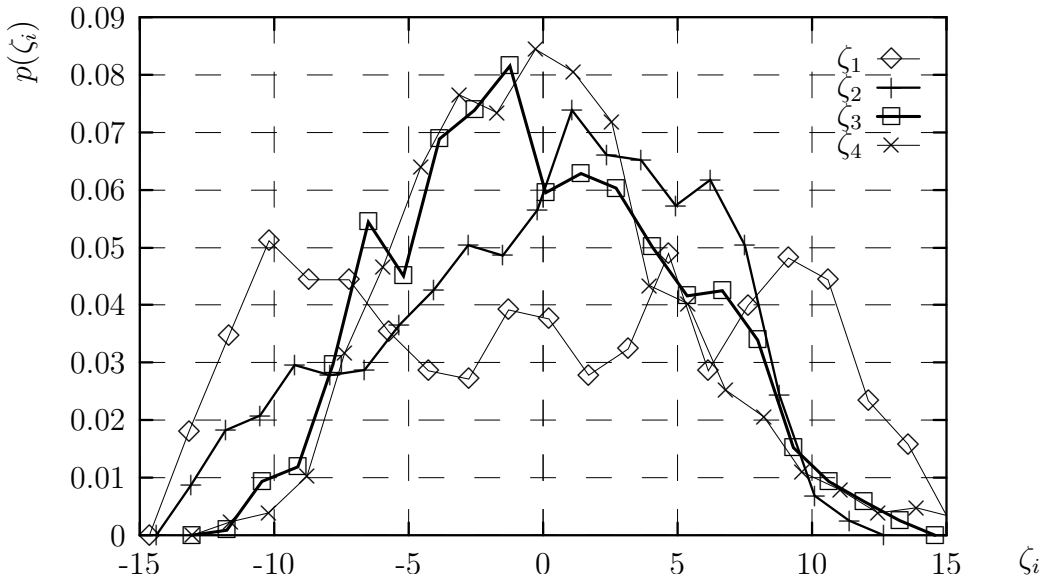


Figure 4.6: Measured distributions of single-variable marginal probability densities of $\zeta_1, \dots, \zeta_4$ of a sample of handwritten digits ‘7’.

For illustration, Figure 4.6 displays histograms of the marginal probability densities of the coefficients ζ_i , obtained from handwritten digits in class ‘7’ on their own subspace $\mathcal{L}_7$. The class-conditional mean has been subtracted from the vectors in the fashion

of (4.24) before the coefficients were calculated. It can be seen that the distribution of ζ_1 is somewhat flat, and the distributions of ζ_2 and ζ_3 are positively and negatively slanted, respectively. Otherwise, the larger the i , the more the distribution of ζ_i resembles zero-mean normal distribution with decreasing variance. This observation leads to the first simplification in the derivation of the new subspace classification rule based on probability density functions: it is assumed that the distribution of the ζ_i coefficients, with i large enough, is zero-mean Gaussian. It can be assumed that many large-dimensional natural data samples exhibit this characteristic to some extent.

Whether the ζ_i coefficients in a particular application really are normally distributed or not, can be tested statistically. Applicable tests include such as the parametric χ^2 -test (Feller 1971) and nonparametric tests of the Kolmogov-Smirnov type, such as the Lilliefors test (Conover 1980). In this demonstration case of the handwritten digits ‘7’, both tests show, in general terms, that the first half-dozen coefficients are non-Gaussian, the succeeding coefficients, up to index $i \approx 40$, are normally distributed, and the remaining ones are, again, non-Gaussian.

In this particular case, the Gaussianity of the distributions can be explained by the process of how the data was produced. In the prototype recognition system described in Chapter 7, the 64-dimensional feature vectors $\mathbf{x}$ were formed using the Karhunen-Loève Transform. Each component of $\mathbf{x}$ is therefore an inner product between a 1024-dimensional normalized input image $\mathbf{f}$ and one estimated principal mask $\mathbf{k}$ of the training sample. Thus, because the coefficients ζ_i are further inner products of the $\mathbf{u}_i$ basis vectors and the feature vector $\mathbf{x}$, they are actually sums of 65536 products of to some extent independent but not identically distributed values. The *Central Limit Theorem* (CLT) (see Mendel 1995) may be interpreted to state that these kinds of series are asymptotically normally distributed. Section 6.4.3 addresses feature extraction with the Karhunen-Loève Transform.

Independence of Coefficients Here, the potential mutual independence of the ζ_i coefficients is examined. Again, the projections of the handwritten digits ‘7’ serve as an illustration. Some two-dimensional marginal distributions of the ζ_i values are displayed in Figure 4.7. Each subfigure shows the dependence of two coefficients ζ_i and ζ_j for some small indices i and j . The distributions in Figure 4.6 were in fact one-dimensional marginal distributions of the distributions of Figure 4.7. It is observable that when, say, $i \geq 4$ and $j > i$, the distributions in the $\zeta_i\zeta_j$ -plane are unimodal and symmetric around the origin, thus resembling the bivariate Gaussian distribution.

In this exemplary case, the coefficient pairs can truly be shown to be uncorrelated by calculating the correlation coefficient for each pair. The resulting t -statistics justifies that the ζ_i variables are uncorrelated. From the uncorrelatedness and the Gaussianity of the coefficients, it follows that they are independent as wished. From the Gaussianity it also follows that the coefficients are not only pairwise independent but mutually independent in all combinations. For the derivation of the new classification rule, it may thus be assumed that other natural multi-dimensional data sets also exhibit similar independence of subspace projection coefficients.

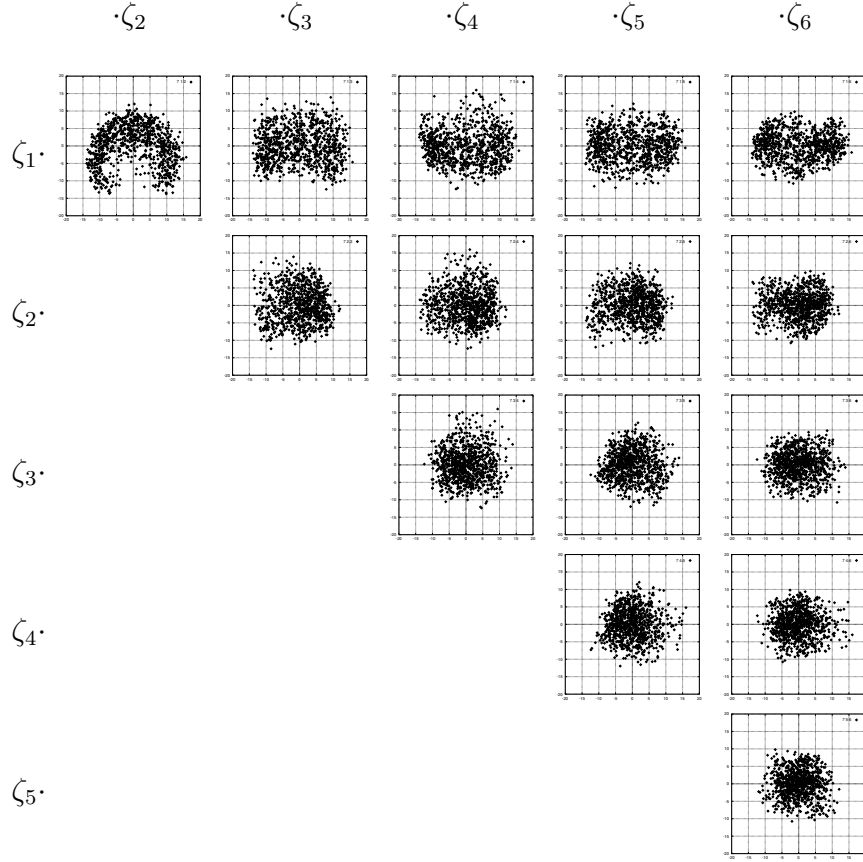


Figure 4.7: Projections of some ‘7’s on subspace $\mathcal{L}_7$. Each subfigure displays the marginal distribution in the plane of the coefficients ζ_i and ζ_j , shown on the left and on the top, respectively. The ranges of the variables are $[-20, 20]$ in all images.

Distribution of Squared Residual Length The length of the residual vector $\tilde{\mathbf{x}}$ is analyzed next. The squared residual length can be expressed as a sum of a truncated series, similarly to (4.5):

$$\|\tilde{\mathbf{x}}\|^2 = \sum_{i=\ell+1}^d (\mathbf{x}^T \mathbf{u}_i)^2 = \sum_{i=\ell+1}^d \zeta_i^2. \quad (4.33)$$

If the ζ_i coefficients really were independent and Gaussian and, additionally, had equal variance, i.e., $\sigma_{\zeta_i}^2 = \sigma^2$, then $\sigma^{-2}\|\tilde{\mathbf{x}}\|^2$ would be distributed as $\chi_{d-\ell}^2$. Also, if Mahalanobis distance (see Bishop 1995) were used instead of Euclidean, the squared residual length would be χ^2 distributed. In reality, of course, the $\sigma_{\zeta_i}^2$ s decrease continuously as i increases and cannot be replaced with a common σ^2 . Nevertheless, it can be assumed here that the distribution of $\|\tilde{\mathbf{x}}\|^2$ follows approximately the gamma probability density function $f_\gamma(x; \alpha, \nu)$, which is the underlying form of the χ^2 distribution. The gamma density is

expressed as

$$f_\gamma(x; \alpha, \nu) = \frac{\alpha^\nu}{\Gamma(\nu)} x^{\nu-1} e^{-\alpha x}, \quad x, \alpha, \nu > 0, \quad (4.34)$$

$$\Gamma(\nu) = \int_0^\infty t^{\nu-1} e^{-t} dt, \quad \nu \geq 0. \quad (4.35)$$

The fitting of the gamma distribution to the observed values may be executed either by maximum likelihood estimation or – like here with the continuing example of handwritten digits ‘7’ – using the method of moments (see Sorenson 1980). Figure 4.8 shows the distribution histogram of $\|\tilde{\mathbf{x}}\|^2$ with parameter values $\ell = 24$ and $d = 64$ in (4.33). The gamma probability density function with parameters estimated from the data is plotted aside.

Thus, a model which is able to estimate the value of the probability density function of a pattern class at the location of the input vector $\mathbf{x}$ has been formulated. At the same time, it has been demonstrated that at least the real-world data used for illustration follows that model with some degree of fidelity. These observations can, again, be extrapolated to say that also other natural data may be modeled similarly and their probability function be approximated with the gamma probability density function.

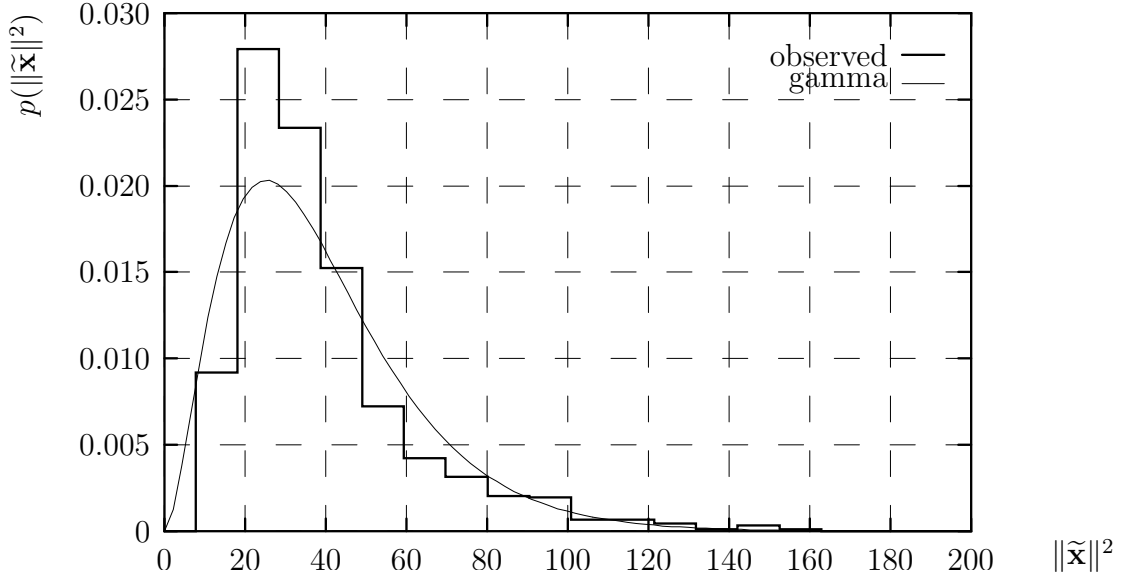


Figure 4.8: Empirical $\|\tilde{\mathbf{x}}\|^2$ distribution of handwritten ‘7’s plotted as a histogram. The corresponding estimated gamma distribution model is drawn as a continuous function.

Reformulated Classification Rule The estimated probability density functions for each pattern class j ,

$$\hat{f}_j(\mathbf{x}) = f_\gamma(\|\tilde{\mathbf{x}}_j\|^2; \hat{\alpha}_j, \hat{\nu}_j) \quad (4.36)$$

may now be inserted, together with the estimated *a priori* class probabilities $\hat{P}$, to the Bayes classifier (3.1) to obtain

$$g_{\text{SS-PDF}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmax}} \hat{P}_j f_\gamma(\|\tilde{\mathbf{x}}_j\|^2; \hat{\alpha}_j, \hat{\nu}_j) . \quad (4.37)$$

Consequently, the *a posteriori* probabilities q_j can be calculated using the relation (3.3). The $\|\tilde{\mathbf{x}}_j\|$ values and the gamma distribution model can either be calculated from the projection of (4.3), or from (4.24), in which the class-conditional means are first subtracted. In either case, the classification is no longer based on maximizing the projection length nor on minimizing the residual length, but on maximizing the *a posteriori* probability of the observed squared residual length. The intrinsic invariance of the subspace classifier concerning the norm of the pattern vector $\mathbf{x}$ has now disappeared. This may be regarded as either a pro or con, depending on the nature of the input data.

The involvement of the estimated *a priori* probability $\hat{P}_j$ of the class j in 4.37 allows for the presented classifier to be used in cases in which the classes are not equally represented in the mixture. Additionally, a rejection mechanism similar to (3.20) can be employed.

According to the Bayesian theory, the misclassifications take place in areas where the class distributions overlap. Generally, if automated pattern classification is feasible at all, the distributions are also sparse in those regions. Therefore, the accuracy of the density function estimation is crucial, in particular, in the areas far from the class means, i.e., in the rightmost tail of the distribution in Figure 4.8. Thus, the performance of the proposed classification rule mainly depends on its success in modeling the tails of the squared residual-length distributions. The proposed algorithm still tries to model the distributions of the $\|\tilde{\mathbf{x}}\|$ values globally. Another approach might be that of fitting the functional model only with the extreme tail distributions.

4.3.7 Computational Aspects

The computational complexity of an algorithm has to be considered when it is used in a real-world application. The various subspace methods have different phases and the computational complexity varies accordingly. Table 4.1 lists the principal procedures of the adaptive LSM and ALSM subspace classification methods. The combined numbers of multiplication and addition operations needed are shown. The parameter n stands for the number of training vectors, c for the number of subspaces, d for feature vector dimensionality, ℓ for the average of all the ℓ_j s, and ϵ for the average classification error rate during the training.

Most of the operations are needed in computing inner products like $\mathbf{u}_{ij}^T \mathbf{x}$. These calculations can be effectively performed in parallel if a *Single Instruction Multiple Data* (SIMD) computer system is available. With SIMD, the computation time can be reduced easily by the factor $1/N$, where N is the number of processors in the system.

<i>operation</i>	<i>LSM</i>	<i>ALSM</i>
correlation	$2nd^2$	$2nd^2$
eigenvectorization	$30cd^2 + 4cd^3$	$30cd^2 + 4cd^3$ *
classification	$2ncd\ell$ *	$2ncd\ell$ *
+weighted projection	$+ncd\ell$ *	$+ncd\ell$ *
rotation	$4nd^2\ell$ *	
re-orthonormalization	$20nd\ell$ *	
scatter matrix update		$4nd^2\epsilon$ *

Table 4.1: Computational needs of subspace methods in numbers of simple mathematical operations. The stages marked with an asterisk (*) are performed in every training epoch, the others only once. The estimate for the complexity of the eigenvectorization phase is from Press et al. (1988).

4.4 Prospects of Subspace Classifiers

This chapter has examined the subspace classification methods starting from their mathematical preliminaries and traversing through the classical algorithms and their learning versions to some novel modifications. The subspace classification methods have not been popular in pattern recognition tasks in the last few years. One explanation for this may be that, in this period, many new neural classification algorithms have been introduced and they have captured the attention of the researchers of adaptive classification systems. This thesis serves on its behalf the purpose of re-enhancing the status of the subspace methods. The novel improvements presented aim to bring the subspaces back to the assortment of the generally acknowledged classification techniques.

The subspace classifiers are not suitable for all classification tasks. The mathematical assumptions behind the subspace formalism demands that the pattern classes are distributed as low-dimensional subspaces in a higher-dimensional feature space. In some applications, the dimensionality of the input is inherently small. In such situations, it is most probable that some other classifiers will produce better classification accuracy than the subspace methods. Even if the input data is linear by nature, not all feature extraction methods preserve this characteristic. Therefore, only a subset of available feature extraction techniques may be combinable with the subspace classification methods.

In addition to the nature of the input data, its quantity also has crucial influence on the applicability of the subspace methods. The estimation of the eigenvectors may become unreliable if the number of vectors in the design sample is small compared to the dimensionality of the data. The semiparametric nature of the subspace classification methods allows for moderate scaling of the model when the number of training vectors is increased. The maximum size is, however, hard-limited by the fact that the subspace dimensionality has to be less than the feature vector dimensionality. Therefore, the performance improvement of the subspace methods when the amount of training data is increased, is

saturated after the eigenvector estimation stage is accurate enough.

The computational needs of the subspace methods, within the framework of modern computers, can be regarded as moderate. The training of the models requires a large memory capacity in the computer if the feature vector dimensionality is large. Also, the eigenvectorization process is computationally demanding and ready-made library functions for that purpose may not be available for all computer platforms. During the operation of the classifier, however, only a small fraction of that memory is needed and the computational complexity is also small. This makes the subspace classifiers well-suited to embedded classification applications in which the classifiers are shipped as a built-in component of a larger data processing system.

Chapter 5

Local Subspace Classifier

This chapter introduces a new classification technique. The proposed method is named the *Local Subspace Classifier* (LSC), which indicates the kinship of the algorithm to the subspace classification methods. On the other hand, the Local Subspace Classifier is an heir of the pure prototype classification methods like the k -NN rule. Therefore, it is argued that, in a way, LSC fills the gap between the subspace and prototype methods of classification. From the domain of the prototype-based classifiers, LSC offers the benefits related to the local nature of classification, while it simultaneously utilizes the capability of the subspace classifiers to produce generalizations from the training sample.

The LSC technique is first explained and visualized in the context of recognition of handwritten digits in Section 5.1. In Section 5.2, a modified formulation of LSC, named the *Convex Local Subspace Classifier* (LSC+), is presented. In Section 5.3, combining the LSC principle with other prototype-based classification methods is studied. The most interesting of these combinations is the approach here named the *Local Subspace Self-Organizing Map* (LSSOM). Section 5.4 presents an iterative replacement for the matrix inversion needed in the original formulation of LSC. The neural interpretations of the LSC methods are discussed in Section 5.5. The performance of the two classifiers is demonstrated in Section 8.5 of Chapter 8 with experiments that use handwritten digit data.

5.1 Basic Local Subspace Classifier

The major strength and weakness of the prototype-based classifiers, such as the k -NN rule, is the local nature of the classification decision. The classification of any input vector is based entirely on the distances to at most k prototypes in each pattern class. In theory, increasing k increases the presentation accuracy of the classifier, but with sparse design sets encountered in practice, the classification accuracy decreases instead. The problem is that the prototypes can be so sparsely distributed in the delicate areas of the class borders that the classification becomes unreliable, because the locality of the prototypes

is lost. Thus, the empty space between the prototypes should somehow be filled.

The strength of semiparametric subspace methods lies in their ability to generalize from the training sample. On the other hand, they rely very strongly on the assumption that the classes are globally linear in form. Thus, the subspace methods are not equipped to describe the mostly nonlinear form of the class boundaries, which normally are poorly represented in the training material.

In general terms, when the Local Subspace Classifier processes an input vector, it first searches for the prototypes closest to that vector in all classes. A local subspace – or, more precisely, a *linear manifold* – is then spanned by these prototype vectors in each class. The classification is then based on the shortest distance from the input vector to these subspaces. Figure 5.1 illustrates the process in a two-class case.

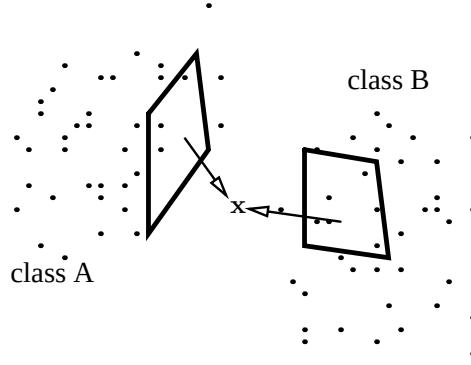


Figure 5.1: The principle of the Local Subspace Classifier in a two-class case. The input vector marked “ $\mathbf{x}$ ” is classified according to the distances shown by arrows to the local subspaces spanned by a few prototypes in the classes “A” and “B”.

Following the notation used earlier with the traditional subspace methods, a D -dimensional linear manifold $\mathcal{L}$ of the d -dimensional space is defined by a matrix $\mathbf{U} \in \mathbb{R}^{d \times D}$ with rank D , and an offset vector $\boldsymbol{\mu} \in \mathbb{R}^d$, provided that $D \leq d$,

$$\mathcal{L}_{\mathbf{U}, \boldsymbol{\mu}} = \{\mathbf{x} \mid \mathbf{x} = \mathbf{U}\mathbf{z} + \boldsymbol{\mu} ; \mathbf{z} \in \mathbb{R}^D\} . \quad (5.1)$$

Note that the offset $\boldsymbol{\mu}$ in (5.1) is not uniquely defined but can be replaced by any $\boldsymbol{\mu}' \in \mathcal{L}_{\mathbf{U}, \boldsymbol{\mu}}$. Among the choices for $\boldsymbol{\mu}'$, however, is a unique selection which minimizes the norm $\|\boldsymbol{\mu}'\|$ and is orthogonal to the basis $\mathbf{U}$. The matrix $\mathbf{U}$ itself may, in general, be assumed orthonormal.

A D -dimensional linear manifold can also be determined by $D + 1$ prototypes, provided that the set of prototypes is not degenerate. Figure 5.2 illustrates a two-dimensional linear manifold defined by the three handwritten ‘4’s in the corners of the triangle. Each of the three corners represents a pure instant of human-printed glyph, whereas the images in between are linear combinations thereof. The original images are 32×32 binary images

after size normalization and slant removal. Due to the sparsity of the prototypes in the 1024-dimensional space, the interpolation produces intermediate images containing holes, such as in the middle of the bottom line. Despite this shortcoming, it may be assumed that any such interpolated image is as good a representative of the class as the original prototypes. Therefore, if the input vector resembles any of such artificial images, it can be considered to have originated from that class.

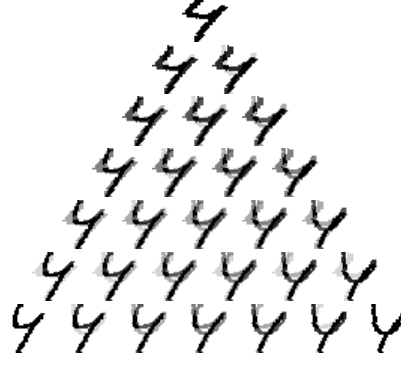


Figure 5.2: A piece of the two-dimensional linear manifold spanned by three examples of handwritten digits ‘4’ in the corners. The gray areas in the interpolated virtual digit images reflect the gradual pixel-wise change from one image to another.

In the following, a procedural formulation of the LSC algorithm is given. The prototypes forming the classifier are denoted by $\mathbf{m}_{ij}$ where $j = 1, \dots, c$ is the index of the class and $i = 1, \dots, n_j$ indexes the prototypes in that class. The manifold dimension in class j is denoted D_j . The distance between two d -dimensional vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as the Euclidean norm of their difference, i.e., $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. When classifying an input vector $\mathbf{x}$, the following is done for each class $j = 1, \dots, c$:

1. Find the $D_j + 1$ prototype vectors closest to $\mathbf{x}$ and denote them $\mathbf{m}_{0j}, \dots, \mathbf{m}_{D_j j}$.
2. Form a $d \times D_j$ -dimensional basis of the vectors $\{\mathbf{m}_{1j} - \mathbf{m}_{0j}, \dots, \mathbf{m}_{D_j j} - \mathbf{m}_{0j}\}$.
3. Orthonormalize the basis by using the Gram-Schmidt process and denote the resulting orthonormal matrix $\mathbf{U}_j = (\mathbf{u}_{1j} \dots \mathbf{u}_{D_j j})$.
4. Find the projection of $\mathbf{x} - \mathbf{m}_{0j}$ on the manifold $\mathcal{L}_{\mathbf{U}_j, \mathbf{m}_{0j}}$:

$$\hat{\mathbf{x}}'_j = \mathbf{U}_j \mathbf{U}_j^T (\mathbf{x} - \mathbf{m}_{0j}) . \quad (5.2)$$

5. Calculate the residual of $\mathbf{x}$ relative to the manifold $\mathcal{L}_{\mathbf{U}_j, \mathbf{m}_{0j}}$:

$$\tilde{\mathbf{x}}_j = \mathbf{x} - \hat{\mathbf{x}}_j = \mathbf{x} - (\mathbf{m}_{0j} + \hat{\mathbf{x}}'_j) = (\mathbf{I} - \mathbf{U}_j \mathbf{U}_j^T) (\mathbf{x} - \mathbf{m}_{0j}) . \quad (5.3)$$

Input vector $\mathbf{x}$ is then classified according to the minimal $\|\tilde{\mathbf{x}}_j\|$ to the class j , i.e.,

$$g_{\text{LSC}}(\mathbf{x}) = \underset{j=1,\dots,c}{\operatorname{argmin}} \|\tilde{\mathbf{x}}_j\| . \quad (5.4)$$

The classification rule is the same as the subspace classification rule (4.24) that utilizes class-specific means with the exceptions that the “mean” vector is now the prototype closest to the input vector, and that the basis matrix of the subspace is now formed from only a small fraction of the training vectors in the class. In any case, the residual length from the input vector $\mathbf{x}$ to the linear manifold is equal to or smaller than to the nearest prototype, i.e., $\|\tilde{\mathbf{x}}_j\| \leq d(\mathbf{x}, \mathbf{m}_{0j})$. These entities are sketched in Figure 5.3.

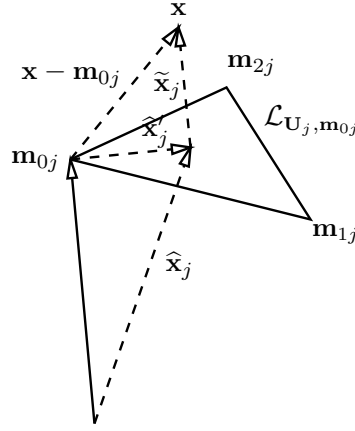


Figure 5.3: The d -dimensional entities involved in the classification by the Local Subspace Classifier in the case $D = 2$.

By introducing the multipliers $\{c_{0j}, \dots, c_{D_jj}\}$ that form the coefficient vector $\mathbf{c}_j = (c_{0j} \dots c_{D_jj})^T$, and by using the matrix $\mathbf{M}_j = (\mathbf{m}_{0j} \dots \mathbf{m}_{D_jj})$, the projection vector $\hat{\mathbf{x}}_j$ can be expressed as

$$\hat{\mathbf{x}}_j = \mathbf{m}_{0j} + \hat{\mathbf{x}}'_j = \sum_{i=0}^{D_j} c_{ij} \mathbf{m}_{ij} = \mathbf{M}_j \mathbf{c}_j , \quad \sum_{i=0}^{D_j} c_{ij} = 1 . \quad (5.5)$$

If the explicit values for the coefficients c_{ij} are needed, they can be solved with matrix pseudo-inversion, which results either from the least-squares normal form or the *Singular Value Decomposition* (SVD) of $\mathbf{M}_j$, i.e., $\mathbf{S}_j = \mathbf{A}_j^T \mathbf{M}_j \mathbf{B}_j$ (see Golub and van Loan 1989),

$$\mathbf{c}_j = \mathbf{M}_j^\dagger \hat{\mathbf{x}}_j = (\mathbf{M}_j^T \mathbf{M}_j)^{-1} \mathbf{M}_j^T \hat{\mathbf{x}}_j = \mathbf{B}_j \mathbf{S}_j^\dagger \mathbf{A}_j^T \hat{\mathbf{x}}_j . \quad (5.6)$$

It can be seen that the LSC method degenerates to the 1-NN rule for $D = 0$. In that case, the projection matrix $\mathbf{U}\mathbf{U}^T$ equals the zero matrix and $\tilde{\mathbf{x}}_j = \mathbf{x} - \mathbf{m}_{0j}$. The two leftmost images in Figure 5.4 illustrate the difference of the 1-NN classifier and the LSC method when $D = 1$. In the figure, the input vector displayed as “ $\mathbf{x}$ ” is classified differently in the two cases, even though the prototype vectors in the classifier remain the same.

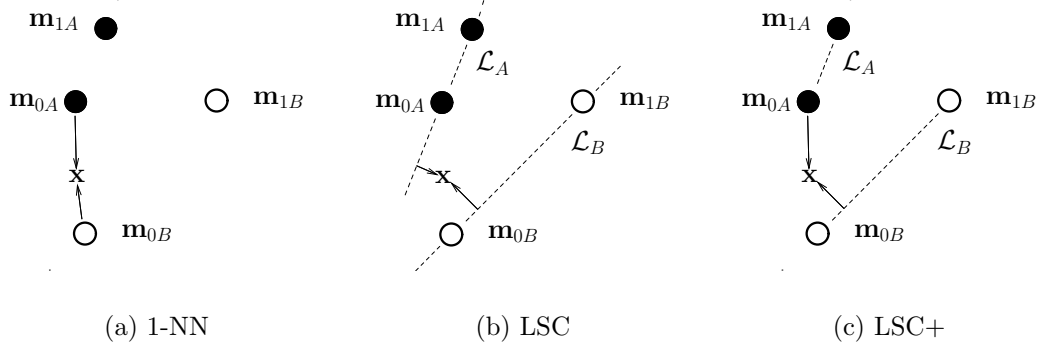


Figure 5.4: Comparison of the 1-NN rule and the basic and convex versions of the Local Subspace Classifier. Using the 1-NN and LSC+ classifiers, $\mathbf{x}$ is classified as “B” but as “A” with the LSC method.

5.2 LSC+ Classifier

The center image in Figure 5.4 shows that the input vector $\mathbf{x}$ may occasionally be projected outside the convex region determined by the prototypes $\mathbf{m}_{0j}, \dots, \mathbf{m}_{D_jj}$. Whether this is beneficial or disadvantageous depends on the particular data sets involved. If extrapolation from the original prototypes is not desired, the orthogonal projection can be replaced with a convex projection. Here, the modified method is called the *Convex Local Subspace Classifier* (LSC+). The appended plus-sign arises from the observation that the coefficients $\{c_{0j}, \dots, c_{D_jj}\}$ in (5.5) are constrained to be positive in the case where the projection vector $\hat{\mathbf{x}}_j$ is not allowed to exceed the convex region determined by the prototypes.

In LSC+, the projection vector $\hat{\mathbf{x}}_j$ is thus not the one that produces the shortest residual $\tilde{\mathbf{x}}_j$. Primarily, it lies within the convex region bound by the nearest prototypes, and only secondarily minimizes the residual length. Obviously, if the orthogonal and convex projections differ, the convex projection lies on the boundary of the convex area and the residual length $\|\tilde{\mathbf{x}}\|$ is longer in the convex than in the orthogonal case. The images (b) and (c) in Figure 5.4 show how a vector $\mathbf{x}$ is classified differently between LSC and LSC+.

The algebraic solution for the convex projection $\hat{\mathbf{x}}_j$ belongs to the class of constrained nonlinear optimization problems. The convex projection fulfills the Kuhn-Tucker conditions (see Taha 1982) and, therefore, has a unique global minimum solution which can be obtained by iterative application of nonlinear programming. In the experiments presented in this thesis, however, another intuitively plausible algorithm has been selected. In the employed method, the convexity condition is achieved iteratively by removing from the basis $\mathbf{M}_j$ the vector $\mathbf{m}_{ij}$ the coefficient of which, c_{ij} , is the smallest. This is effectively the same as forcing the corresponding coefficient to be equal to zero. Thus, the dimensionality D_j of the linear manifold is, in the process, decreased by one. After each removal, the coefficients for the remaining vectors are solved again. The iteration is continued until none of the coefficients is negative and a convex solution has thus been obtained.

5.3 Combining LSC with Prototype Classifiers

The Local Subspace Classifier has so far been portrayed as a method for making more effective use of the information inherent in the prototype vectors of a k -NN classifier. The technique may also be gracefully combined with other types of prototype classifiers, such as the Learning k -Nearest Neighbor classifier (L- k -NN) of Section 3.4.3 and the Learning Vector Quantization (LVQ) of Section 3.4.2, and with the topology-preserving vector quantization produced by the *Self-Organizing Map* (SOM) (Kohonen 1995).

In the basic LVQ1, LVQ2, and LVQ3 formulations (Kohonen 1995), the classification is always based on one nearest codebook vector in each class. The usefulness of the Local Subspace Classifier can be tested after the training of the codebook has ended, and if no advantage is gained, the original 1-NN classification can be retained. For the other adaptive prototype classifiers mentioned, the LSC principle may be applied either only after the training of the classifier has ended, or, more plausibly, already during the adaptation phase. In the latter case, the training algorithms need to be modified to some extent. The formulation of the Learning k -Nearest Neighbor classifier (see Section 3.4.3) is such that more than one vector from each class is moved simultaneously. As such, the principle of adaptation in L- k -NN is compatible with the LSC technique and the three learning rules of L- k -NN can easily be extended to conform with it.

A central operation in the adaptive prototype-based vector classification algorithms is the *delta rule*, described on page 38. When the delta rule is applied to the Local Subspace Classifier, it is natural not to modify only one prototype vector but to extend the modification to all D_j prototype vectors involved in the classification. Due to the equiproportional nature of the linear vector movements in the delta rule, the hyperplanes determined by the prototypes are moved in a fashion that preserves the directions of the hyperplanes. Therefore, the prototype hyperplane after the delta movements is parallel to the original hyperplane. Figure 5.5 illustrates the movements of the prototype vectors in a simple two-dimensional $D = 1$ case. The center image shows both the original and modified vector locations. Due to the similar triangles, the lines joining the prototypes really are parallel. Furthermore, the directions of the residual vectors $\tilde{\mathbf{x}}_A$ and $\tilde{\mathbf{x}}_B$ are not altered during the application of the delta rule but rather the lengths of the residuals are increased or decreased by the factor α . As a result of the prototype movements, the classification of the input vector $\mathbf{x}$ changes. This is indicated in Figure 5.5 by the shift of the dotted classification border line over the $\mathbf{x}$ vector.

The most interesting extension of the LSC technique is based on the topology-preserving Self-Organizing Map. In the course of the training of a SOM, a set of vectors located in the neighborhood of the best-matching unit are moved toward the training vector. Again, this movement, provided that all these vectors are moved with an equal $\alpha(t)$ coefficient in the delta rule (3.18), is compatible with the LSC delta rule principle. The locating of the best-matching unit, on the contrary, is not. In the basic SOM, each map unit is individually matched against the input vector, and the neighborhood is formed around the winning neuron thereafter. In the *Local Subspace SOM* (LSSOM), linear manifolds are formed by using the vectors in the neighborhoods of *all* map units. The best-matching

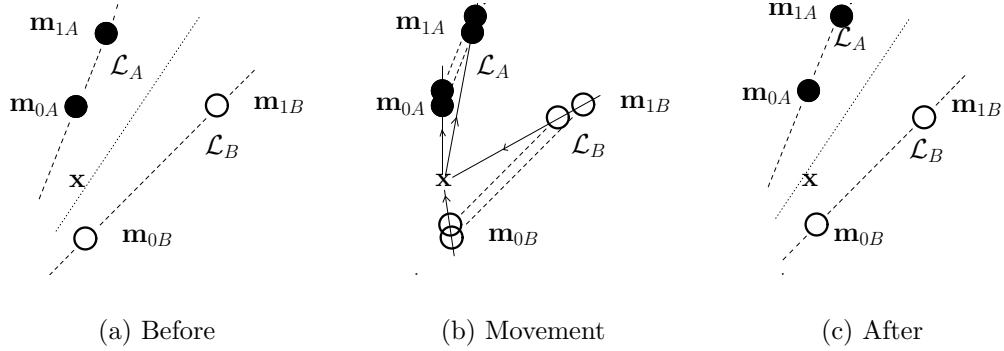


Figure 5.5: Delta rule in the LSC training. The locations of the prototype vectors are shown before, during and after the application of the delta rule. The movement of the dotted line indicates that the classification of $\mathbf{x}$ is changed from ‘A’ to ‘B’ in the process.

local subspace is sought as the one with the shortest distance to the input vector similarly to the classification rule (5.4). Therefore, each unit of the LSSOM participates in a multitude of local subspaces.

Due to the unsupervised nature of LSSOM training, the prototype vectors $\mathbf{m}_i$ are moved only toward the input vectors. The radius of the neighborhood in LSSOM is decreased during the training just as in the normal SOM. The size D of the neighborhood is naturally smaller at the borders and corners of the map than in its inner parts. If desired, and labeled data are available for supervised training, it is possible to have a separate LSSOM map for all the classes involved. In that case, the units representing the class of the input vectors are moved toward the input vector, and the units of the nearest competing class are moved further from it by the delta rule. The principle of the LSC+ may also be optionally applied with the LSSOM principle.

As the result of the Local Subspace SOM, an elastic net is formed from the nodes of the SOM. The difference compared to the basic SOM is that, in the LSSOM, the space between the nodes is included in the locally linear manifolds. Where in the SOM there are discrete points defined by the node vectors in the feature space, in the LSSOM there are linear manifolds spanned by the same vectors. Therefore, there are no “holes” or “jumps” in the mapping from the multidimensional input space to the two-dimensional grid space. Figure 5.6 depicts how various neighboring units in a Self-Organizing Map form local subspaces.

The principle of the LSSOM is related to other approaches to nonlinear modeling of high-dimensional data by using lower-dimensional functional forms. As such, the LSSOM resembles the *Principal Curves* by Hastie and Stuetzle (1989), the *Weighted Vector Quantization* (WVQ) modification of SOM by De Haan and Egecioğlu (1991), the *Principal Manifolds* by Ritter et al. (1992), the *Adaptive Principal Surfaces* by LeBlanc and Tibshirani (1994), the *Continuous Mapping* interpretation of SOM by Mulier and Cherkassky (1995), and the *Curvilinear Component Analysis* (CCA) by Demartines and Hérault (1997).

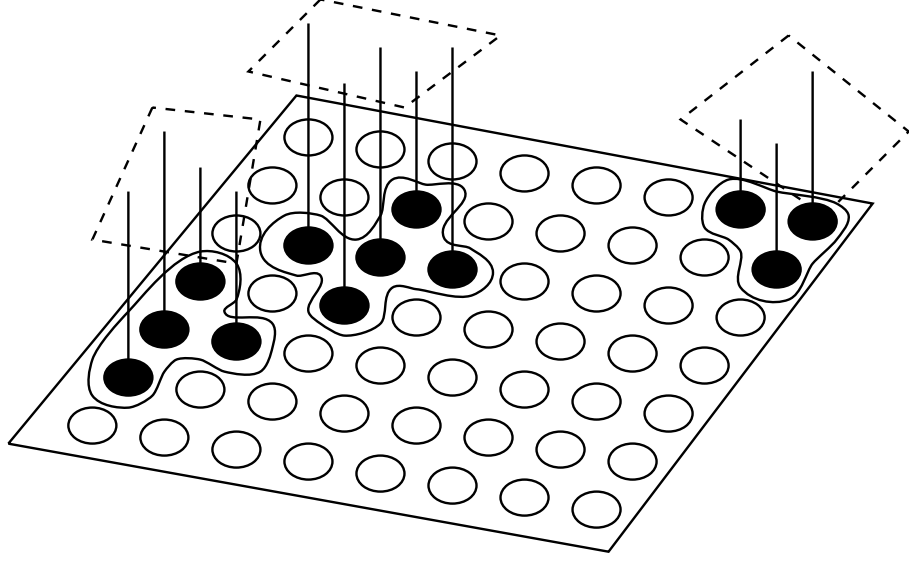


Figure 5.6: Local linear manifolds formed from the prototype units of a Self-Organizing Map. Here, each manifold is formed of three to five prototype vectors.

5.4 Iterative Solution for LSC

An iterative algorithm may be applied in order to avoid the explicit use of the Gram-Schmidt orthonormalization process or the matrix pseudoinversion in the solution for $\hat{\mathbf{x}}_j$ and the multipliers c_{ij} . In each iteration step, the projection vector $\hat{\mathbf{x}}_j(t)$ is moved toward one of the $\mathbf{m}_{ij}$ vectors. The amount of the movement is selected to give the residual minimum along that direction. The iteration can be formulated:

1. Initialize $\hat{\mathbf{x}}_j(0) = \mathbf{m}_{0j}$, $\mathbf{c}_j(0) = (1 \ 0 \ \dots \ 0)^T$, $t = 1$.

2. Set $i = t \bmod (D_j + 1)$.

3. Calculate
$$b_j(t) = \frac{(\mathbf{x} - \hat{\mathbf{x}}_j(t-1))^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))}{(\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))} . \quad (5.7)$$

4. Update

$$\hat{\mathbf{x}}_j(t) = (1 - b_j(t))\hat{\mathbf{x}}_j(t-1) + b_j(t)\mathbf{m}_{ij} , \quad (5.8)$$

$$c_{ij}(t) = (1 - b_j(t))c_{ij}(t-1) + b_j(t) , \quad (5.9)$$

$$\forall k \neq i : c_{kj}(t) = (1 - b_j(t))c_{kj}(t-1) . \quad (5.10)$$

5. End iteration if $t > D_j$ and $\frac{\|\hat{\mathbf{x}}_j(t) - \hat{\mathbf{x}}_j(t-D_j-1)\|}{\|\hat{\mathbf{x}}_j(t) + \hat{\mathbf{x}}_j(t-D_j-1)\|} \leq \varepsilon$, where ε is a small positive limit.

6. Increment t by one and go to Step 2.

The process is illustrated in Figure 5.7, which shows that the new location of $\hat{\mathbf{x}}$ is always such that $\mathbf{x} - \hat{\mathbf{x}}_j(t)$ is perpendicular to $\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1)$. Therefore, the length of $\mathbf{x} - \hat{\mathbf{x}}_j(t)$ forms a non-increasing series until $\hat{\mathbf{x}}_j(t)$ converges to the position in which $\mathbf{x} - \hat{\mathbf{x}}_j(t)$ is orthogonal to all $\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t)$, $i = 0, \dots, D_j$. The speed of the convergence depends on the order in which the vectors $\mathbf{m}_{ij}$ are presented. At each step of the iteration, (5.9) and (5.10) guarantee that the sum of the multipliers c_{ij} remains equal to one.

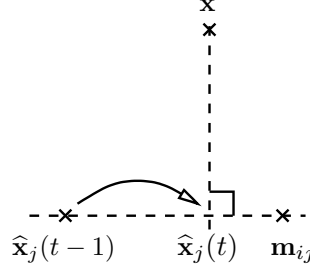


Figure 5.7: One step of the iterative solution for $\hat{\mathbf{x}}_j = \lim_{t \rightarrow \infty} \hat{\mathbf{x}}_j(t)$. The approximate solution $\hat{\mathbf{x}}_j$ is changed from $\hat{\mathbf{x}}_j(t-1)$ to $\hat{\mathbf{x}}_j(t)$ by moving a fraction determined by $b_j(t)$ in (5.7) toward $\mathbf{m}_{ij}$.

In the case of the Convex LSC (LSC+), the requirement for positive multipliers c_{ij} can be incorporated in the above algorithm by replacing Step 3 with

3. Calculate

$$b'_j(t) = \frac{(\mathbf{x} - \hat{\mathbf{x}}_j(t-1))^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))}{(\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1))}, \quad (5.11)$$

$$b_j(t) = \max_{\substack{k=0, \dots, D_j \\ k \neq i}} \left\{ \frac{c_{kj}(t-1) - 1}{c_{kj}(t-1)}, \frac{c_{ij}(t-1)}{c_{ij}(t-1) - 1}, \min\{b'_j(t), 1\} \right\}, \quad (5.12)$$

which guarantees that all the multipliers c_{ij} remain within $0 \leq c_{ij} \leq 1$ at every step of the iteration. If a denominator in (5.12) equals zero, the corresponding term is interpreted to have the value of minus infinity. Thus, it is neglected in the selection of the maximum value.

5.5 Neural Network Interpretation of LSC

Another iterative solution for the projection vector $\hat{\mathbf{x}}_j$ can be derived. The principle of this iteration stems from a system of continuous-time differential equations leading to the solution of a set of error functions. These error functions originate from the observation that the residual vector $\mathbf{x} - \hat{\mathbf{x}}_j$ should be orthogonal to all the vectors $\mathbf{m}_{ij} - \hat{\mathbf{x}}_j$, due to the orthogonality of the projection operator. An iterative algorithm of the gradient-descent

type can be derived by starting from the notion that the inner product of two orthogonal vectors equals zero. Thus, the error functions $e_{0j}, \dots, e_{D_jj}$ are defined as

$$e_{ij} = (\mathbf{x} - \hat{\mathbf{x}}_j)^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j). \quad (5.13)$$

The gradient of e_{ij} can then be written

$$\nabla_{\hat{\mathbf{x}}_j} e_{ij} = 2(\hat{\mathbf{x}}_j - \frac{\mathbf{m}_{ij} + \mathbf{x}}{2}), \quad (5.14)$$

which may be interpreted as a vector starting from the midpoint of $\mathbf{m}_{ij}$ and $\mathbf{x}$ and going through $\hat{\mathbf{x}}_j$. Viewing from the point $\hat{\mathbf{x}}_j$, a zero value of e_{ij} can then be found by following the gradient line because the zero set of e_{ij} is the surface of a hyperball centered at $\frac{\mathbf{m}_{ij} + \mathbf{x}}{2}$. The surface passes through both $\mathbf{x}$ and $\mathbf{m}_{ij}$, as illustrated in Figure 5.8. In the interior of the ball, e_{ij} is negative, and in its exterior, positive.

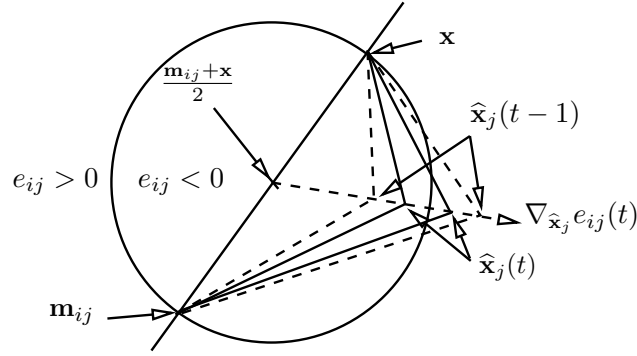


Figure 5.8: Neural iterative solution for $\hat{\mathbf{x}}_j = \lim_{t \rightarrow \infty} \hat{\mathbf{x}}_j(t)$. The dashed arrow displays the direction of the gradient $\nabla_{\hat{\mathbf{x}}_j} e_{ij}$. If $\hat{\mathbf{x}}_j(t-1)$ lies within the hyperball, the error value e_{ij} is negative, and in the exterior, positive. In both cases, $\hat{\mathbf{x}}_j$ is moved toward the surface of the ball.

A gradient-descent algorithm for a discrete-time iterative solution of the equation $e_{ij} = 0$ can be formulated by multiplying the gradient $\nabla_{\hat{\mathbf{x}}_j} e_{ij}$ with the value of the error function e_{ij} . The update of $\hat{\mathbf{x}}_j(t)$ is then made in the opposite direction. Depending on the sign of e_{ij} , the direction of the change of $\hat{\mathbf{x}}$ is either toward or away from the center of the hyperball. In either case, the movement is directed toward the surface. These two cases are illustrated in Figure 5.8 in which the vectors $\mathbf{x} - \hat{\mathbf{x}}_j$ and $\mathbf{m}_{ij} - \hat{\mathbf{x}}_j$ are drawn with dashed and solid lines corresponding to the location of $\hat{\mathbf{x}}$ before and after its movement, respectively. Note that all the error equations $e_{ij} = 0$, $i = 0, \dots, D_j$, can be solved simultaneously. This gives rise to the iterative updating rule for $\hat{\mathbf{x}}(t)$:

1. Initialize $\hat{\mathbf{x}}_j(0)$ with any random value in $\mathbb{R}^d$. Set $t = 1$.

2. For each i in $0, \dots, D_j$, calculate

$$e_{ij}(t) = (\mathbf{x} - \hat{\mathbf{x}}_j(t-1))^T (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1)), \quad (5.15)$$

$$\Delta \hat{\mathbf{x}}_{ij}(t) = -\alpha(t) e_{ij}(t) \left(\hat{\mathbf{x}}_j(t-1) - \frac{\mathbf{m}_{ij} + \mathbf{x}}{2} \right) + \beta(t) (\mathbf{m}_{ij} - \hat{\mathbf{x}}_j(t-1)). \quad (5.16)$$

3. Update $\hat{\mathbf{x}}_j(t) = \hat{\mathbf{x}}_j(t-1) + \sum_{i=0}^{D_j} \Delta \hat{\mathbf{x}}_{ij}(t)$.
4. End iteration, if $\frac{\|\hat{\mathbf{x}}_j(t) - \hat{\mathbf{x}}_j(t-1)\|}{\|\hat{\mathbf{x}}_j(t) + \hat{\mathbf{x}}_j(t-1)\|} \leq \varepsilon$, where ε is a small positive limit.
5. Increase t by one and go to Step 2.

The small positive multipliers $\alpha(t)$ and $\beta(t)$ control the rate and stability of the convergence of the iteration. They should decrease monotonically in time. The $\beta(t)$ term forces the projection $\hat{\mathbf{x}}_j$ to move toward the mean value of the prototype set $\{\mathbf{m}_{0j}, \dots, \mathbf{m}_{D_jj}\}$ and, thus, to the subspace $\mathcal{L}_j$. In the beginning, its value should be near $1/(D_j + 1)$, and it may decrease rapidly. If $\mathbf{m}_{0j}$ were used as $\hat{\mathbf{x}}_j(0)$ instead of a random initial value, the β term would be unnecessary because the iteration would already start within the subspace $\mathcal{L}_j$.

This process can be given a neural interpretation as follows. Each neural unit of the system stores one prototype vector $\mathbf{m}_{ij}$ in its synaptic weights. The unit receives as input both the vector $\mathbf{x}$ and the current value of its projection $\hat{\mathbf{x}}_j(t)$. It first calculates its error function e_{ij} and, then, produces its contribution to the change of $\hat{\mathbf{x}}_j(t)$. These modifications are integrated by a neural unit on the second layer, which, in turn, changes the value of $\hat{\mathbf{x}}_j(t)$ stored dynamically as the state of one neural unit. All the factors of the update term $\Delta \hat{\mathbf{x}}_j(t)$ can be calculated in parallel with simple mathematical operations on the stored and input vectorial quantities. Therefore, the production of the projection vector $\hat{\mathbf{x}}_j$ can be considered neural and depicted as in Figure 5.9.

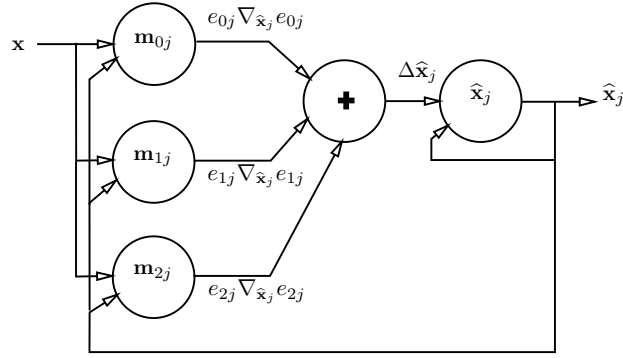


Figure 5.9: The neural interpretation of the projection operation in the Local Subspace Classifier.

Chapter 6

Survey of Off-line Recognition of Handwriting

This chapter presents an overview of methods used in the recognition of handwriting. Section 6.1 reviews the history of the research interest. Some potential application areas are listed in Section 6.2. Section 6.3 presents various subtopics of the character recognition discipline and its neighboring research interests. A detailed survey of feature extraction methods is presented in Section 6.4. Finally, Section 6.5 addresses some of the questions related to the implementation of real-world character recognition systems. In general questions of digital image processing, the interested reader is referenced to the books by Rosenfeld and Kak (1982), Jain (1989), Schalkoff (1989), Pratt (1991) and Gonzalez and Woods (1992).

6.1 History

Automatic recognition of printed and handwritten text was regarded as an important and challenging goal even before the advent of modern electronic computers. Similar to the recognition of speech, it is a task in which it is easy to compare the performance of a machine to that of a human. The first technical breakthrough on the path leading to modern systems was made by Carey in 1870, when he invented what he called the retina scanner. In 1890, Nipkow developed the first sequential scanning device, the cornerstone of both television and reading machines. After the collection of visual data was made feasible, the first attempts to interpret the contents of visual information were made by Tyurin in 1900, deAlbe in 1912, and Thomas in 1926. The first patents in the field were issued in Germany to Goldberg in 1927 and to Tauschek in 1929, and in the United States to Handel in 1933. For a review of the historical development, see Mantas (1986) and Mori et al. (1992).

The eve of the era of practical handwriting recognition systems dates back to the 1950s. In 1946, John Mauchly and J.P. Eckert developed the first operational electronic digital

computer, ENIAC. Soon after this, the new machinery was applied to automated pattern recognition, including recognition of printed characters studied, among others, by Glauberman (1956), ERA (1957) and Chow (1957).

During the 1960s and 70s, the recognition systems for printed text were developed to an operational level, for example, by Balm (1970), but reliable system performance required that special fonts were used in printing. The first systems capable of recognizing cursive script were also introduced, for instance by Lindgren (1965). Mostly syntactic pattern recognition methods were used in classification during these decades.

In the 1980s and 90s, neural networks have been applied to character recognition problems. They are used primarily as classifiers, but some bottom-up neural solutions have also emerged, such as that by LeCun et al. (1989). Most of the current systems use statistical features and either neural networks or statistical classification algorithms.

During the past few years, many excellent overviews of the development and the state-of-the-art recognition systems and methods have been presented, for instance, by Rabinow (1969), Harmon (1972), Freedman (1974), Suen et al. (1980), Mantas (1986), Govindan and Shivaprasad (1990), Tappert et al. (1990), Suen et al. (1992), Mori et al. (1992), and by Suen et al. (1993). The problems of character recognition have been addressed in numerous scientific conferences. The most important series of conferences have been organized by the International Association for Pattern Recognition (IAPR), the Institute of Electrical and Electronics Engineers (IEEE), and the International Society for Optical Engineers (SPIE). It has been a great benefit to the development of handwritten character recognition methods that many benchmarking studies on classification methods have used handwritten character data. Some of the most recent comparisons include Wilkinson et al. (1991), Geist et al. (1992), Blue et al. (1994), Michie et al. (1994), and Blayo et al. (1995).

6.2 Application Areas

One of the most tempting applications for automated processing of machine-printed and handwritten text is a system which can locate, recognize, and interpret any written text in any paper document, such as hand-drawn maps and casual memos. Such a machine could facilitate the transfer of old handwritten manuscripts and registers to computer databases. In offices, a lot of secretarial work could be handed over to such a machine. These kinds of systems are, however, still far beyond the current level of development.

Technical problems can be drastically reduced if the task is restricted to processing such forms where the users of the system have filled in a certain type of data. In such a case, the system knows beforehand for what and where to search. Such forms include tax forms and letters in postal sorting centers. Operational systems have been described, among others, by Gilloux (1993), Srihari (1993), and Burges et al. (1993).

The most recent platform for recognition of human-written characters are palmtop computers, personal assistants, and electronic notebooks. From a technical point of view,

such systems are just another form of table digitizers used for years in the coding of technical drawings. Nevertheless, they have brought on-line character recognition into the spotlight. The striking benefit of such systems over the traditional scanning of paper documents is interactivity; the system receives an immediate response and acceptance from the user and even learns to avoid errors.

Potential users for systems that recognize handwriting are found mainly among the governmental authorities, such as postal and tax departments, or among large private companies, such as banks, which have a considerable amount of written communication with their customers. On the other hand, if such systems were small and economic enough, they might provide the visually handicapped with valuable aid.

6.3 Fields of Character Recognition

The application area in this case study is off-line recognition of isolated handwritten digits, which is a highly specialized area of pattern recognition research. Various realms in character recognition overlap. In order to fully comprehend any particular case, some overall view of the entire field has to be reached. This section provides such an understanding in accordance with the dichotomies presented in Figure 6.1. The frequently used term *Optical Character Recognition* (OCR) is slightly off from the taxonomy. The word *optical* was earlier used to distinguish an optical recognizer from systems which recognized characters that were printed using special magnetic ink. Nowadays, on-line handwritten character recognition should thus not be considered optical in this classical sense. Consequently, the term OCR is somewhat unfortunate and misleading.

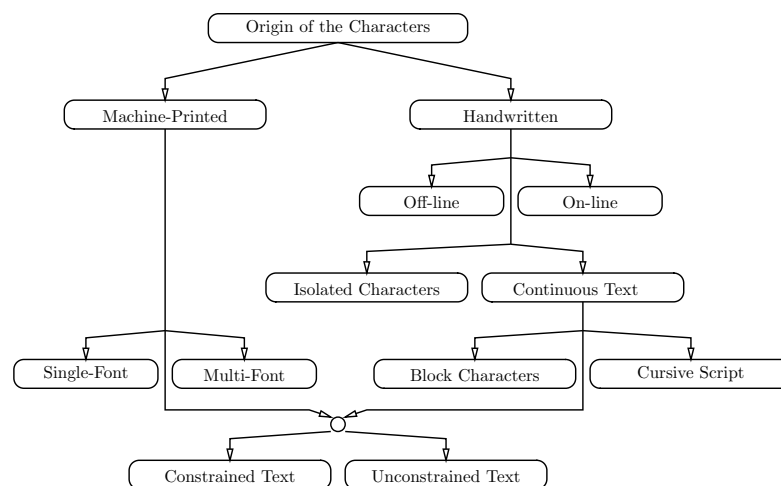


Figure 6.1: Fields of character recognition. In each branch, the more demanding alternative is placed on the right.

Character recognition, in general, means recognition of any form of printed or written

textual or symbolic information including, for example, mathematical formulas. From this perspective, the way the data is produced and processed is meaningless. Therefore, the characters may or may not be visual. The latter is the case, for instance, in pen-driven palmtop computers. The division between a *machine-printed* and *handwritten* origin of characters is quite obvious, though not categorical; a system that recognizes handwriting should also be able to handle machine-printing. The distinction is further blurred by fonts that imitate handwriting. Due to larger variations in character images, recognition of handwritten text is generally regarded as a more demanding task than recognition of printed text for which there have long been operative commercial products (Andersson 1969 and 1971).

In the case of handwriting recognition, there is a crucial dichotomy between *on-line* and *off-line* applications. Compared to off-line systems, on-line systems have additional information concerning the order and timing of the movements the user makes with the virtual pen. By utilizing the on-line information, the classification accuracy can be enhanced, but the algorithms needed in processing the data are much more involved. The research of on-line algorithms has increased lately, and a common data format has been proposed for making comparative benchmark studies practicable (Guyon et al. 1994).

Another independent division of handwritten text concerns *isolated characters* and *continuous text*. The former occur mostly as simple one-digit or one-character responses to queries in fill-in forms and in some on-line recognition applications where the writing area is limited. The latter incorporates the need to segment the input image before the classification phase. Systems that recognize machine-printed text always assume continuous input. Characters in continuous text form some larger entities, such as words or numbers. A special case of continuous text recognition is *cursive script* recognition. The segmentation is in that case much more difficult and methods like the *Hidden Markov Models* (HMM) are often used (Dunn and Wang 1992). An easier case is *block letters*. They may touch each other but their form is not so much affected by neighboring characters as in script writing. Thus, the internal variance of each block letter class is smaller. A corresponding pair can also be found for machine-printing: the earliest systems were able to recognize only fonts developed especially for OCR purposes. Slightly more evolved systems accepted any *single font*. Modern systems for machine-printed OCR are able to recognize *many fonts* or even to adapt to any number of new font styles. Figure 6.2 shows real-world examples of all these three types.

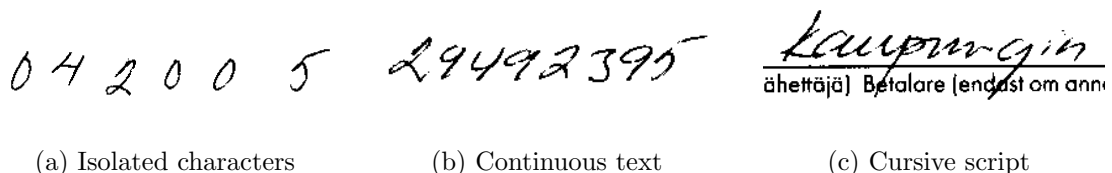


Figure 6.2: Various types of handwriting extracted from one fill-in form.

The last dichotomy in Figure 6.1 is common to both machine-printed and handwritten systems and divides the recognition of *constrained* and *unconstrained* text. The former

includes situations where the user of the recognition system is advised to write or type the input to the system by filling in predefined boxes of a form. Thus, the system has plenty of *a priori* information of what to look for and where. Recognition of isolated characters may be seen as an extreme of this scheme. In the most unconstrained case, the user has a blank page which she can fill in with any information in any style and even in any direction. This approach directly leads to *document analysis*. It is an important research branch of its own. In document analysis, various types of information included in paper documents are located without any prior information. Besides, in some cases, rudimentary document analysis is an inevitable registration step before handwritten text can be segmented and recognized. In practice, a text entry can be considered unconstrained if there is more than one line of input and if the number of words in each line is not known in advance. Examples of both constrained and unconstrained texts are shown in Figure 6.3.

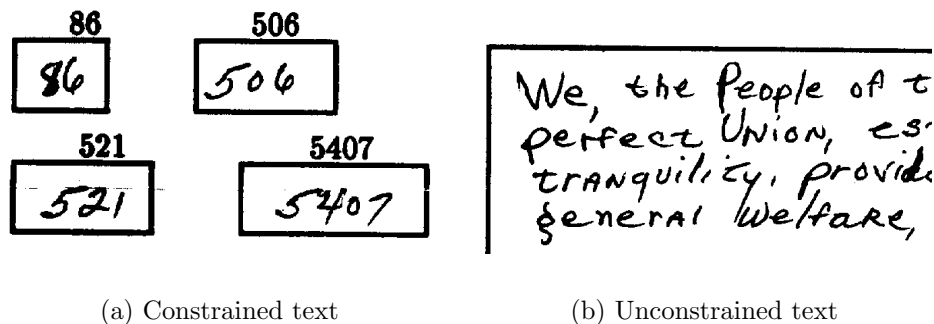


Figure 6.3: Examples of constrained and unconstrained handwritten text.

One constraint on the recognition system is the set of recognized symbols. The simplest case only contains ten digits and possibly some additional symbols, such as punctuation marks and plus and minus signs. Then, either uppercase alone or both uppercase and lowercase letters can be admitted as input. The most versatile systems should accept Greek and Cyrillic alphabets, or even Asian character sets.

6.4 Feature Extraction in Handwriting Recognition

This section describes two taxonomies of feature extraction methods for handwriting. Section 6.4.1 discusses the general issue of reconstructibility from features. In the six succeeding sections, some feature extraction methods are described in detail. Finally, Section 6.4.8 introduces a new scheme, here called the *error-corrective feature extraction*.

In addition to the taxonomy presented in the previous section, the field of character recognition can also be described on the basis of data collection methods, feature extraction principles, classification methods, or the data presentation form used. The latter was selected by Trier et al. (1996), who presented an excellent overview of the feature extraction methods for character recognition. This taxonomy is reproduced in Table 6.1. The four image representation forms, gray-scale, solid binary, binary contour, and vector skeleton,

feature extraction	representation form				page
	<i>Gray-scale subimage</i>	<i>Binary solid symbol</i>	<i>Binary outer contour</i>	<i>Vector skeleton</i>	
Template matching	X	X		X	90
Deformable templates	X			X	91, 102
Unitary transforms	X	X			92, 94
Log-polar transform *	X	X			94
Geometric moments	X	X			95
Zernike moments	X	X			96
Wavelets *	X	X			98
Algebraic features *	X	X			98
Projection histograms		X			99
Fitted Masks *	X	X			99
Contour profiles			X		100
Chain codes			X		100
Spline curve			X		101
Fourier descriptors			X	X	101, 102
Graph description				X	103
Discrete features				X	104
Zoning	X	X	X	X	105

Table 6.1: Feature extraction methods tabulated by the representation form following the taxonomy by Trier et al. (1996) but rearranged and grouped, and methods marked with an asterisk have been added. The rightmost column shows the page where the method is described.

are shown as columns. Reasonable feature extraction methods are indicated for each. Here, the methods are grouped and sorted differently than in the original presentation. The methods not examined by Trier et al. (1996) are marked with an asterisk. As can be seen, the selection of the representation form and features are not totally interdependent nor independent. After the former has been fixed, the latter still has some options. This weak interdependency is carried on to the selection of classification methods.

Another taxonomy motivated by feature extraction and classification principles is presented in Table 6.2. All the entries of Table 6.1, except for template matching, deformable templates, and zoning, are placed in the taxonomy. Transforms, moments, wavelets, algebraic features, and histograms are grouped under the title *volume features*. Now, the primary divisions of feature extraction methods fall between heuristic and systematic methods, on the one hand, and, according to the classical division between structural and statistical pattern recognition on the other.

	structural	statistical
heuristic	• Discrete features	• Fitted masks
systematic	• Chain codes • Spline curve • Graph description	• Volume features • Contour profiles • Fourier descriptors

Table 6.2: A taxonomy of feature types for character recognition. The dichotomy “structural” versus “statistical” reflects the corresponding principles of classification. The words “heuristic” and “systematic” refer to the way the features are selected.

The heuristic methods are most often characterized by strict confinement to some properties found useful in the classification of some particular data. Another motivation for their use can be found from psychological observations concerning human perception, among others, by Blesser et al. (1976). In contrast, systematic methods try to model the input data more rigorously and independently from a specific application. The structural methods work with a small set of discrete variables, such as the number of endpoints in the character or the locations and the types of the junctions in the image. Such indicators can be handled with tree-structured classifiers or with sets of rules that associate a digit class to various combinations of the indicators. The statistical methods are most suitable for handling continuous-valued vectors which can be naturally obtained with some mathematical transformations of the input image. Such feature vectors can be classified either with classical statistical methods or with neural networks. Some researchers, including Weideman et al. (1995) and Heutte et al. (1996), have tried to combine the benefits of both structural and statistical approaches. Because this thesis concentrates mainly on statistical and neural methods, the systematic statistical feature extraction methods receive most of the attention.

6.4.1 Reconstruction from Features

In pattern recognition, the ability to reconstruct the original input pattern from the features extracted from it is not necessary. In the particular case of character recognition, however, reconstruction gives essential feedback to the designer of the classifier system about the loss of precision of data in the feature extraction phase. The reconstruction is easiest if unitary transforms of the image are used to obtain volume features, as the following section demonstrates. In the reconstruction figures, it should be observed how many feature terms are required to obtain a certain quality in the reconstruction. A large number of terms required indicates that the resulting classifier may suffer from the curse of dimensionality. Or, that the selected feature extraction scheme does not fit together with the data. The objective of feature extraction is, however, **not** good reconstruction or effective dimension reduction, but sufficient classification accuracy.

6.4.2 Template Matching

The template classification methods are the oldest and the most straightforward methods of character recognition. Actually, they are not feature extraction methods in the strict sense. Still, the template matching approach is worth exploring in this context because many feature extraction methods are based on it.

An input image is modeled as a discrete spatial function $f(x, y)$ where x and y are the coordinates in a normalized $w \times h$ image. The mean-square distance of an image f and a template t_j of equal size is a commonly-used dissimilarity measure,

$$d^2(f, t_j) = \sum_{x=1}^w \sum_{y=1}^h (f(x, y) - t_j(x, y))^2 \quad (6.1)$$

$$= (\mathbf{f} - \mathbf{t}_j)^T (\mathbf{f} - \mathbf{t}_j) \quad (6.2)$$

$$= \mathbf{f}^T \mathbf{f} + \mathbf{t}_j^T \mathbf{t}_j - 2\mathbf{t}_j^T \mathbf{f} . \quad (6.3)$$

In the second and third forms, the vectors $\mathbf{f}$ and $\mathbf{t}_j$ are formed by concatenating the pixels of the image and the mask, respectively. The input image is then classified directly according to the minimum distance to a template. If the method is used in an application in which the speed of operation is important, then the distance calculation would be done with the third, inner product, form. Only the correlation term $\mathbf{t}_j^T \mathbf{f}$ needs to be calculated for every input image and template, while the $\mathbf{t}_j^T \mathbf{t}_j$ term can be calculated once for each template and stored. The value of the term $\mathbf{f}^T \mathbf{f}$ is not needed at all in matching.

More application-oriented similarity measures than that of (6.3) ought to be used with binary-valued images. In order to formulate the notation, values n_{bb} , n_{bw} , n_{wb} , and n_{ww} are first defined. They represent the number of pixels that are black in both the image and the template, black only in the image, etc. Various similarity measures of a merely logical rather than algebraic nature can be derived from these quantities. The similarity value one indicates a perfect match between the image and the template, whereas a zero

or negative similarity value results from a complete mismatch. Tubbs (1989) and Gader et al. (1991) found the Jaccard and Yule similarities the most suitable for recognizing handwritten digits

$$d_{\text{Jaccard}} = \frac{n_{\text{bb}}}{n_{\text{bb}} + n_{\text{bw}} + n_{\text{wb}}} , \quad (6.4)$$

$$d_{\text{Yule}} = \frac{n_{\text{bb}}n_{\text{ww}} - n_{\text{bw}}n_{\text{wb}}}{n_{\text{bb}}n_{\text{ww}} + n_{\text{bw}}n_{\text{wb}}} . \quad (6.5)$$

Template matching is an optimal classification method in the sense that if an infinite number of templates and infinite time were available, classification accuracy equal to human performance would be obtainable. This is generally not possible if dimension-reducing feature extraction is used. In practical applications, however, the method suffers severely from the curse of dimensionality. Therefore, feature extraction methods are used.

Deformable Templates Deformable templates are an evolutionary form of the genuine template matching approach. Character images as such form a high-dimensional feature space. On the other hand, small variations in any particular image span a low-dimensional manifold around the image. These deformations can be modeled with the Taylor expansion or other similar methods. The distance measure (6.3) also needs to be generalized to express the minimal distance between the input image and a manifold “centered” at a template. Because the modeled deformations are expressed with continuous values, gray-scale images are especially suited to be recognized using deformation approximation. Like template matching, deformation does not produce actual features in the strict sense of feature extraction. Rather, the method can be interpreted as defining a new non-Euclidean distance measure between the input image and the template.

For example, Simard et al. (1993) introduced a transformation distance metric for the classification of handwritten characters and analyzed it further in (Simard et al. 1994) and (Hastie et al. 1995). They defined the distance between two character images as the minimum distance between the tangent planes originating from the normalized images. The tangent plane is generated from a combination of simulated translation, rotation, scaling, and two hyperbolic transformations, and, thus, parameterized with a six-dimensional vector $\mathbf{w}$. The deformed image $\hat{\mathbf{f}}(\mathbf{w})$ is expressed as an expansion around the original image $\mathbf{f}$,

$$\hat{\mathbf{f}}(\mathbf{w}) = \mathbf{f} + \frac{\partial \mathbf{f}(\mathbf{w})}{\partial \mathbf{w}} \mathbf{w} . \quad (6.6)$$

Then, the distance between two images $\mathbf{f}_{\mathbf{x}}$ and $\mathbf{f}_{\mathbf{y}}$ is

$$d(\mathbf{f}_{\mathbf{x}}, \mathbf{f}_{\mathbf{y}}) = \min_{\mathbf{w}_{\mathbf{x}}, \mathbf{w}_{\mathbf{y}}} \|\hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{w}_{\mathbf{x}}) - \hat{\mathbf{f}}_{\mathbf{y}}(\mathbf{w}_{\mathbf{y}})\| . \quad (6.7)$$

6.4.3 Volume Features

In a way, volume features are descendants of straightforward template matching. Here, the templates are presented as a set of weighted *masks*, or *kernels*, superimposed on

the image. The use of masks (i) tries to compensate for small spatial variations, and (ii) produces dimension reduction beneficial in the sense of the curse of dimensionality. A general discrete spatial mask is defined by a function $k_i(x, y)$, or by $k_{pq}(x, y)$ in the case of a two-dimensional transform domain. Each component $x_1, \dots, x_d$ of a d -dimensional feature vector $\mathbf{x}$ is calculated as the inner product of the mask k_i and the image f ,

$$x_i = \sum_{x=1}^w \sum_{y=1}^h k_i(x, y) f(x, y) = \mathbf{k}_i^T \mathbf{f} . \quad (6.8)$$

The mask vectors $\mathbf{k}_i$ are formed by concatenating the pixels in a manner similar to the formation of the image vector $\mathbf{f}$ in (6.3). When the mask vectors are combined into the matrix $\mathbf{K} = (\mathbf{k}_1, \dots, \mathbf{k}_d) \in \mathbb{R}^{wh \times d}$ we are led to the notation

$$\mathbf{x} = \mathbf{K}^T \mathbf{f} . \quad (6.9)$$

The squared Euclidean distance between two feature vectors $\mathbf{x}$ and $\mathbf{y}$ may then be expressed by using the mask matrix and the original images $\mathbf{f}_\mathbf{x}$ and $\mathbf{f}_\mathbf{y}$:

$$\|\mathbf{x} - \mathbf{y}\|^2 = (\mathbf{K}^T \mathbf{f}_\mathbf{x} - \mathbf{K}^T \mathbf{f}_\mathbf{y})^T (\mathbf{K}^T \mathbf{f}_\mathbf{x} - \mathbf{K}^T \mathbf{f}_\mathbf{y}) \quad (6.10)$$

$$= (\mathbf{f}_\mathbf{x} - \mathbf{f}_\mathbf{y})^T \mathbf{K} \mathbf{K}^T (\mathbf{f}_\mathbf{x} - \mathbf{f}_\mathbf{y}) \quad (6.11)$$

How faithfully the original metrics of the input space are transformed to the feature space depends on the eigendirections of the $\mathbf{K} \mathbf{K}^T$ matrix. An important subclass of volume features is formed by those which can be produced using *unitary transforms*. A unitary transform is a reversible linear transform whose kernel $\mathbf{K}$ forms a complete, orthonormal base. In the unitary case, the mask matrix obeys $\mathbf{K}^T \mathbf{K} = \mathbf{I}$, and the matrix $\mathbf{K} \mathbf{K}^T$ in (6.11) can be interpreted as an orthonormal projection operator, as in Section 4.1. The image $\hat{\mathbf{f}} = \mathbf{K} \mathbf{K}^T \mathbf{f}$ is thus the reconstruction of the original image $\mathbf{f}$. A “virtual template” $t(x, y)$ can also be formed from any feature vector $\mathbf{x}$ and the mask matrix $\mathbf{K}$,

$$t(x, y) = \sum_{i=1}^d k_i(x, y) x_i = \mathbf{K} \mathbf{x} . \quad (6.12)$$

The dimensionality d is typically some orders of magnitude smaller than the dimensionality of the image space, i.e., $w \times h$. Therefore, neither the original templates nor the input images can be presented accurately with the masks. As a result, decreased quality of reconstruction is obtained.

Discrete Karhunen-Loève Transform Various unitary transforms have been suggested and used in feature extraction in the hope that all essential information of the shapes of the objects is concentrated in a few transform coefficients. If that were the case, the identity of the object could be revealed by using these coefficients as statistical features. The most commonly-used unitary transform in OCR applications is the *Discrete Karhunen-Loève Transform* (KLT), which was originally a method for optimal coding of

data (Watanabe 1965). The kernel masks are formed by calculating first the covariance matrix of the training data and solving the eigenvalues and eigenvectors of that matrix. The first eigenvectors in the order of decreasing eigenvalue are then used as the masks. Due to the optimality property, maximal amount of variance is preserved in the transformation. The process thus resembles the formation of the subspace basis matrix for CLAFIC in (4.6), i.e.,

$$\widehat{\boldsymbol{\mu}}_{\mathbf{f}} = \frac{1}{n} \sum_{i=1}^n \mathbf{f}_i \quad (6.13)$$

$$\widehat{\boldsymbol{\Sigma}}_{\mathbf{f}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{f}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{f}})(\mathbf{f}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{f}})^T \quad (6.14)$$

$$\mathbf{K} = \left(\mathbf{k}_i \mid (\widehat{\boldsymbol{\Sigma}}_{\mathbf{f}} - \lambda_i \mathbf{I}) \mathbf{k}_i = \mathbf{0}, \lambda_i \geq \lambda_{i+1}, i = 1, \dots, d \right) \quad (6.15)$$

Figure 6.4 displays how increasing the feature vector dimensionality improves the reconstruction accuracy. At the same time, the overall amplitude of the reconstruction deviates more and more from the zero level shown as the background gray.

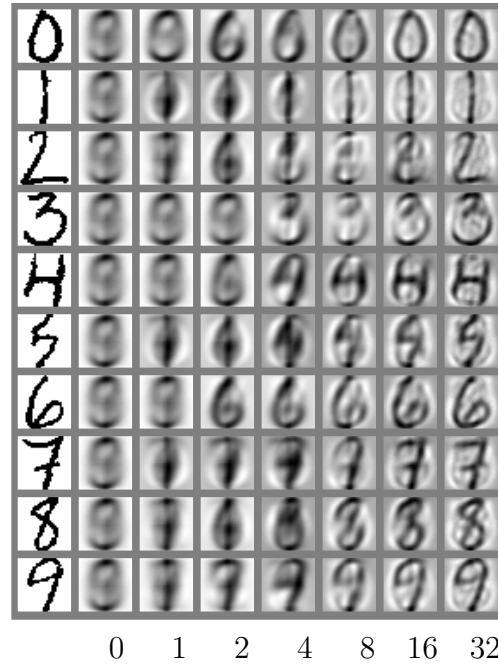


Figure 6.4: Reconstruction of 32×32 -sized handwritten digits from their Discrete Karhunen-Loève Transform. The feature space dimensionality d is shown below the images. The mean of the data was subtracted from the images before the transformation and added back afterwards.

The use of KLT in handwritten character recognition was suggested by Andrews (1971) in a comparison in which he conjectured (!) it to be superior to other unitary transforms including Fourier, Walsh-Hadamard, and Haar Transforms. The computational complexity,

which inhibited the actual use of KLT in those days, is a less important issue nowadays. Consequently, it is at least a feature extraction method worth starting the experiments with. KLT is a data-dependent transform, which may be interpreted either as a strength or a weakness, depending on the stationarity of the input data during the period beginning with the collection of the training material and ending in the actual use of the application. The major drawback of KLT is that the training vectors from all the classes are summed up before the eigenvectors are solved. Therefore, the special properties of the distributions of the classes may overlap in the feature extraction process.

Discrete Cosine Transform A distinctive family of unitary transforms is the *Discrete Fourier Transform* (DFT) and its relatives. The *Discrete Cosine Transform* (DCT) (see Gonzalez and Woods 1992) is the most commonly used of them in image coding and analysis. One of its uses is the JPEG standard of still picture compression (Wallace 1991). An advantage of DCT over DFT is that its calculations are not complex but real-valued. Each of the DCT coefficients is calculated:

$$x_{pq} = \frac{c_p c_q}{N} \sum_{x=1}^w \sum_{y=1}^h \cos \frac{(2x-1)p\pi}{2w} \cos \frac{(2y-1)q\pi}{2h} f(x, y), \text{ where} \quad (6.16)$$

$$c_i = \begin{cases} 1, & \text{for } i = 0, \\ \sqrt{2}, & \text{for } i = 1, 2, \dots \end{cases} \quad (6.17)$$

The feature vector $\mathbf{x}$ is formed from the coefficients in the order: $x_{00}, x_{10}, x_{01}, x_{20}, x_{11}, \dots$. If the input data is assumed to be produced by a first-order Markov process, then the eigenvectors of the process covariance matrix approach the DCT kernel vectors as the correlation coefficient approaches one (Ahmed et al. 1974; Unser 1984). Therefore, it can be argued that DCT approximates KLT for highly correlated sequences and shares its valuable properties.

Log-Polar Transform One possibility for non-unitary image feature extraction is the *Log-Polar Transform* (LPT), which is invariant to scale and rotation. Actually, LPT is not a feature extraction method at all, but, instead, a biologically motivated way of resampling an image (Schwartz 1977). In LPT, the Cartesian (x, y) coordinates are transformed to polar coordinates with a logarithmic radius,

$$f(x, y) = f(\log \rho, \theta), \quad \log \rho = \frac{\log(x^2 + y^2)}{2}, \quad \theta = \tan^{-1} \frac{y}{x}. \quad (6.18)$$

When using LPT, the origin of the coordinate system is translated to the center of the mass of the image before the change of coordinates. Similar translation is performed also after the coordinate change. The invariance to scale is due to the log operator which transforms a scaling factor to a translation in the $\log \rho$ direction. Likewise, rotation in the input image is reflected as a translation in the θ direction of the resulting image. Thus, both variations are canceled out by the two translations of the coordinate system.

Due to the effects of sampling, LPT creates concentric masks with increasing separation between the rings. Another quite similar approach is to use sampling centered along a logarithmic spiral which extends from the center of the image (Weiman and Chaikin 1979). The use of LPT is, however, somewhat inconvenient because the new coordinates are heterogeneous and, therefore, two separate sampling parameters need to be selected. The Log-Polar Transform has been used mostly as a preprocessing step prior to the actual feature extraction, for instance, by Wechsler and Zimmerman (1988). Kageyu et al. (1991) used LPT with the Fourier transform in the recognition of handwritten digits. Figure 6.5 shows the Log-Polar transformed image of a handwritten '4' in $s \times s$ discretization of the log-polar plane, where $s = 5, 10, 15, 20, 25$. Below, corresponding reconstructed 32×32 images are shown.

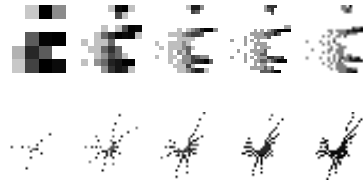


Figure 6.5: Log-Polar transformed digit '4' and reconstructed images using 25, 100, 225, 400, and 625 coefficients.

Moments Moments and invariant features based on them have been used widely in visual pattern recognition ever since they were introduced by Hu (1961). The basic moment of the order (p, q) – p and q being non-negative integers – is defined as:

$$m_{pq} = \sum_{x=1}^w \sum_{y=1}^h x^p y^q f(x, y) . \quad (6.19)$$

More suitable than the plain moments for pattern recognition purposes are the corresponding *central moments* μ_{pq} for $p + q > 1$

$$\mu_{pq} = \sum_{x=1}^w \sum_{y=1}^h \left(x - \frac{m_{10}}{m_{00}}\right)^p \left(y - \frac{m_{01}}{m_{00}}\right)^q f(x, y) . \quad (6.20)$$

Hu (1962) also presented the *normalized central moments*, or *absolute similitude moment invariants*, as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} , \quad \gamma = \frac{p+q}{2} + 1 . \quad (6.21)$$

A set of moment invariants known as Hu's *absolute orthogonal moment invariants* may be calculated by using the normalized central moments (Haralick and Shapiro 1992):

$$\phi_1 = \eta_{20} + \eta_{02} \quad (6.22)$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (6.23)$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (6.24)$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (6.25)$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\ + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (6.26)$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (6.27)$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\ - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (6.28)$$

A selection of these ϕ_i s or other similar constructs can be used to form the feature vector $\mathbf{x}$. Alt (1962) defined another form of *normalized central moments* m_{pq}^* , which are invariant under translation, scale, and any general affine transform:

$$m_{pq}^* = \frac{\sum_{x=1}^w \sum_{y=1}^h x^p y^q f(x, y)}{m_{00}}, \quad (6.29)$$

$$x^* = (x - \frac{m_{10}}{m_{00}}) / \sqrt{\mu_{20}/m_{00}}, \quad (6.30)$$

$$y^* = (y - \frac{m_{01}}{m_{00}}) / \sqrt{\mu_{02}/m_{00}}. \quad (6.31)$$

The normalized central moments can still be presented as a set of image masks in which the scaling and positioning of the higher-order masks depend on the lower-order moments.

Teague (1980) presented a thorough review of moment invariants and related themes. Maitra (1979) and Hsia (1981) have contributed significantly to the renewed interest in the theory of moment invariants. Reddi (1981) derived radial and angular moments as an alternative representation for Hu's invariants. Belkasim et al. (1991) presented a survey and comparison of various moment invariants. New theoretical formulations include those by Reiss (1991) and Li (1992). Recently, Flusser and Suk (1993) proposed a set of *affine moment invariants*, which are invariant under affine transformation, and demonstrated their use in recognition of handwritten characters (Flusser and Suk 1994).

The invariance properties of moments have proven to be useful, for example, in the recognition of aircraft images. In character recognition, the rotation, mirroring, and reflection invariance may be a disadvantage, however. Therefore, only some of the proposed moment invariants are applicable as such to character recognition.

Zernike Moments Previously, moments were defined in the Cartesian coordinate system. In a somewhat corresponding manner, Teague (1980) defined the Zernike moments in polar coordinates. The Zernike moments form a set of complex harmonics in a manner similar to the Fourier transform in the Cartesian system. They were originally defined

for continuous-valued variables, for which they form an orthonormal set of kernels. For discrete variables this is not true, so the transform is not unitary. A Zernike moment A_{pq} is defined for all $p \geq 0$, $|q| \leq p$, and even $p - |q|$, by using the polar kernels $V_{pq}(\rho, \theta)$ and the radial polynomial $R_{pq}(\rho)$,

$$A_{pq} = \frac{p+1}{\pi} \sum_x \sum_y f(x, y) V_{pq}^*(\rho, \theta), \quad (6.32)$$

$$V_{pq}(\rho, \theta) = R_{pq}(\rho) e^{iq\theta}, \quad (6.33)$$

$$R_{pq}(\rho) = \sum_{s=0}^{\frac{p-|q|}{2}} (-1)^s \frac{(p-s)!}{s! \left(\frac{p+|q|}{2} - s\right)! \left(\frac{p-|q|}{2} - s\right)!} \rho^{p-2s}. \quad (6.34)$$

The polar coordinates (ρ, θ) are defined in a unit circle embracing the centered and properly scaled image to be transformed in the usual way:

$$\rho = \sqrt{x^2 + y^2} \leq 1, \quad \theta = \tan^{-1} \frac{y}{x}. \quad (6.35)$$

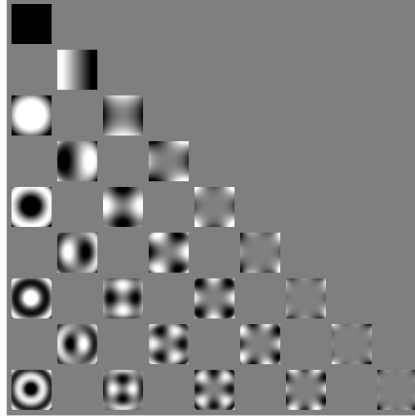


Figure 6.6: Shapes of the first 25 Zernike kernels with fixed horizontal orientation. $V_{0,0}$ at top, $V_{8,0}$ in bottom left and $V_{8,8}$ in right corners.

Shapes of the Zernike masks up to $p = q = 8$ are plotted in Figure 6.6. Figure 6.7 displays reconstruction from the Zernike coefficients. The complex Zernike moments may be used as such or combined further to produce invariant features. Teague (1980) suggested the formation of invariant features as products of individual Zernike moments. His approach produces features invariant to reflection and rotation. Khotanzad and Hong (1990a and 1990b) used the magnitudes of the complex Zernike moments as features in recognition of handwritten characters. Some additional formulations of Zernike invariants were studied by Belkasim et al. (1991). Their survey and benchmarking study on moment invariants used handwritten digit data. Just as with regular moments, some forms of invariance in Zernike moments may be a disadvantage in character recognition since, for instance, digits ‘6’ and ‘9’ are distinguishable only by direction.

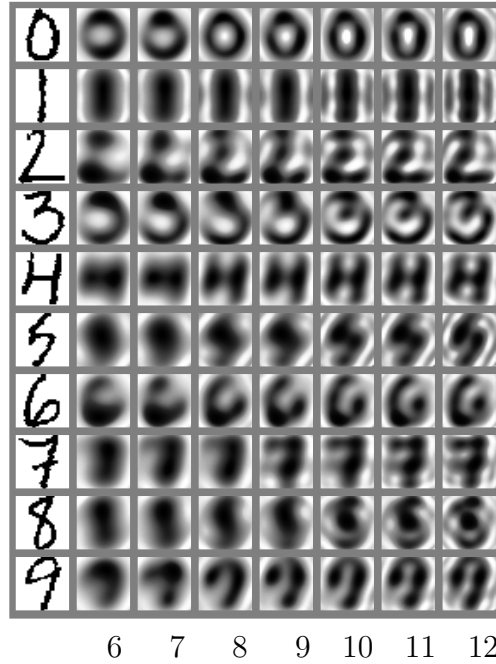


Figure 6.7: Reconstruction of digits from their Zernike coefficients. The maximum of order p of the coefficients used is shown below.

Wavelets Lately, wavelets have been used intensively in image analysis and coding (see Vetterli and Kovačević 1995; Strang and Nguyen 1996), and there exist both unitary and non-unitary versions of wavelet kernels. Compared to the Discrete Cosine Transform, wavelets are more localized and, thus, able to describe the image with fewer coefficients. Of various sets of wavelets, the kernel of the complex two-dimensional *Gabor filters* is written (Daugman 1988):

$$k_{u_0, v_0, \alpha, \beta}(x, y) = e^{-\pi(x-x_0)^2\alpha^2 + (y-y_0)^2\beta^2} e^{-2\pi i u_0(x-x_0) + v_0(y-y_0)} . \quad (6.36)$$

(x_0, y_0) is the center point of the function in the image, i.e., normally, the center of the image $(\frac{w+1}{2}, \frac{h+1}{2})$. The scaling parameters α and β are inversely proportional to the variances of the Gaussian frequency band in x and y -directions. The modulation parameters (u_0, v_0) define the mean frequency of the template in the Fourier-domain.

Algebraic Features Algebraic features are constructed from an image considered as a matrix. Various algebraic transforms or matrix decompositions can be used for feature extraction from an image. The Karhunen-Loève Transform, presented on page 92, is one such example.

Hong (1991), who introduced the term of algebraic features, stated that they represent intrinsic attributions of an image. He used the *Singular Value Decomposition* (SVD) of a matrix as the classification feature for face images. Cheng et al. (1993) and Liu et al. (1993) applied the *Similar Discriminant Function* (SDF) to the same problem. Finally,

Liu et al. (1994) applied the algebraic feature extraction technique to the recognition of handwritten characters.

Projection Histograms Projection histograms are mainly used for segmenting lines, words, and characters in document images. Glauberman (1956), however, used them as features in his early hardware character reader. Projection histograms can be interpreted as horizontal and vertical bar masks. They transform a two-dimensional image into two one-dimensional histograms. Tang et al. (1996) used ring projection, in which the images were projected onto concentric rings of equal width. The resulting histograms were then decomposed with one-dimensional wavelets.

Due to their sensitivity, for instance, to small rotations in the images, projection histograms cannot be used as the only features in classification. If used in combination with other features, histograms may offer useful additional information.

Fitted Masks All the volume feature extraction methods described up to this point have had a statistical or other form of mathematical background. Therefore, in Table 6.2, they were characterized as systematic features. As a counter-example, there are classification systems that use heuristic fitted masks designed by recognition system developers. Thus, they are very application-specific and typically non-unitary. Simply put, each mask indicates whether there exists a certain meaningful line or curve in the input image. The way the masks are used may be more logical than arithmetical by nature. As an example of heuristic masks, Burr (1988) presented his “shadow codes”, later used by Weideman et al. (1995).

6.4.4 Outline Features

Outline features are based on modeling the outline of the character to be recognized. Therefore, it is necessary to trace the closed outer contour curve. For binary images this edge detection task is easy, but gray-scale images need first to be changed into binary images using either global or local thresholding. Alternatively, an algorithm capable of tracking the contour of a gray-level image has to be used, such as the one presented by Herman and Liu (1978). In another approach, described by Lam et al. (1992), the outline contour is first located roughly by comparing the gray-values of neighboring pixels. The resulting outline is then thinned to the width of one pixel and used in character recognition. In Figure 6.8, the image (b) displays the 8-connected outline of the handwritten digit ‘4’ in image (a).

The motivation for the use of the outline features in classification is grounded in the observation that characters can be distinguished by their outer contours. The outer contours of such letters as the uppercase ‘B’ and ‘D’ may, however, be very similar. Therefore, it is possible that the features derived from the outline are very sensitive to noise or otherwise unsuitable for classification. The shortcomings of the outline features may, at least

partially, be overcome by using the inner contours or other additional features. Another serious drawback of the outline features follows if each character is assumed to be a single-connected region or classifiable by using only the largest of the regions. This assumption is valid, for example, in the case of the letters 'i' and 'j'. Most diacritical characters, such as 'ä' and 'ö', however, cannot be distinguished from their non-diacritical counterparts by relying on the single-connected component hypothesis.

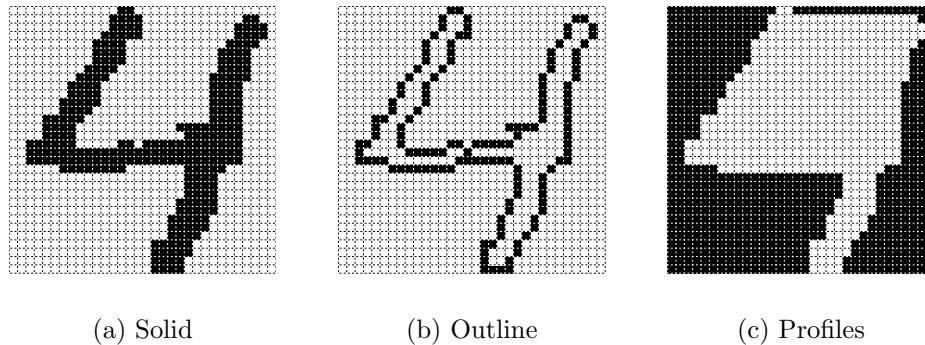


Figure 6.8: A handwritten digit '4' in (a), its outline in (b), and left and right contour profiles in (c).

Contour Profiles Contour profiles are the simplest features extractable from character outlines. In the case of binary images, the outline does not need to be explicitly traced before extracting the profiles. The left contour profile measures the distance from the left edge of the image to the left contour of the character on each row. The right contour profile is formed correspondingly. This results in two integer-valued vectors which have a dimensionality equal to the height of the image. The profiles can be used as feature vectors in statistical classification or processed further to form heuristic features. Both approaches were tested by Shridhar and Badreldin (1985 and 1986) and Kimura and Shridhar (1991). Figure 6.8 (c) shows the contour profiles of a handwritten digit '4' of Figure 6.8 (a).

An advantage of contour profiles is their immunity to small gaps in character images. On the other hand, they are extremely sensitive to rotation and slant. Therefore, proper preprocessing has to be used. Even then, the profiles may be useless as classification features. Figure 6.8 illustrates how difficult it might be to distinguish handwritten digits '4' and '9' by their contour profiles.

Chain Code Histograms Chain codes are formed by vectorizing the closed outline of a character. The outline chain is then transformed to a string in which each element indicates the direction of the contour at that particular point. The number of quantization levels used in representing the directions may vary. Most often the four basic directions that result from an eight-connected neighborhood model will suffice. A direction histogram can be generated from the entire image or any part of it. Thus, this

method can be combined well with the zoning approach, discussed in Section 6.4.7. Chain code histograms were successfully used in character recognition by Kimura and Shridhar (1991).

Splines The general concept of splines comprises any linear, polynomial, or rational approximations of the outer contour of a character. In general, the resulting representations are suitable for syntactical pattern classification. For example, Ali and Pavlidis (1977) presented a recognition system in which the boundary of a digit is described by using four types of entities extracted from a linear approximation of the boundary: curves, protrusions, lines, and breaks. The resulting entity strings are then classified by using two successive grammar parsers.

When a spline description is formed, the chain code has to be first broken into subparts to be modeled with individual spline curves. The breaking points are located along the closed curve in places with high curvature (Sekita et al. 1988). In general, a p th-order spline is continuous up to the $(p-1)$ th derivative in its breaking points. Taxt et al. (1990) used an algorithm in which the distance from the center of the image to the B-spline-approximated outer boundary is measured with regular intervals along the spline curve. A statistical feature vector is then formed from these measurements and from the mean curvatures calculated in the segments between the measurement points. Thus, the splines are used only to smoothen the extracted outline.

Fourier Descriptors Fourier descriptors are used for characterizing the outer boundary of a character. The Fourier coefficients capable of representing the form of a handwritten digit can be used as statistical features and classified in a manner similar to that of statistical volume features. The first coefficients describe the most coarse shape of the character. The amount of detail is increased as more coefficients are included. In this sense, the Fourier descriptors resemble the coefficients that result from the Cosine and Karhunen-Loève Transforms.

Granlund (1972) represented a closed and, thus, periodic complex-valued contour function $u(t)$ using complex Fourier coefficients a_n :

$$a_n = \frac{1}{2\pi} \int_0^{2\pi} u(t) e^{-int} dt, \quad (6.37)$$

where t is an angular parameter with a fixed starting direction. The coefficients a_n for $n \neq 0$ are invariant with respect to position. The scaling of the image corresponds to a constant multiplier in the coefficients. Rotation invariance may also be produced, but it is seldom needed in digit recognition where the orientation of images is fixed by normalization. Shridhar and Badreldin (1984), who used 15 real-valued descriptors $x_i = |a_i|/|a_1|$ in digit classification, needed less than three averaged models per digit class in 1-NN classification to obtain adequate classification accuracy.

6.4.5 Skeleton Features

A skeleton of an image can be used as a starting point for feature extraction. The skeletonization process, also known as *thinning* or the *Medial Axis Transform* (MAT), is by itself problematic regarding both the computational aspects and the potential uniqueness of the result. These issues have been addressed, among others, by Lam et al. (1992) and Jang and Chin (1992). Figure 6.9 shows a handwritten digit ‘7’ and its skeleton in the images (a) and (b), respectively. In the context of handwritten character recognition, the goal of the thinning process is quite obvious: the trace of a pen spreading ink while moving on a paper should be recovered from the final image. The average line width can be estimated prior to thinning and that information used in skeletonization. Alternatively, the average line width can be estimated after thinning if the information is needed in further processing. *Vectorization* is a step that commonly follows skeletonization. In vectorization, the pen trace is represented with a small number of linear line segments known as *strokes*. Many feature extraction methods use strokes as input data.

In recognition of handwritten digits, Taxt and Bjerde (1994) used a feature extraction scheme based on the *Elliptic Fourier Descriptors*, originally presented by Kuhl and Giardina (1982). In that case, the resulting features were statistical. A general view is, however, that features extracted from the image skeleton lead to syntactical classification.

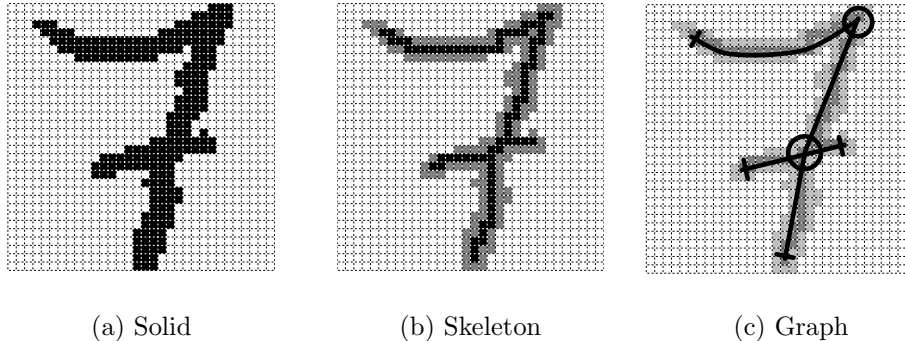


Figure 6.9: A handwritten digit ‘7’ in (a), its 4-connected skeleton in (b), and a hypothetical graph description in (c).

Vector Templates Small variations in the shape and relative location of the parts of a handwritten symbol cause extensive changes in the extracted skeleton of the image. Therefore, direct template matching of skeletonized characters is seldom feasible. Instead, the skeleton templates are transformed to deformable vector models which describe the input images and the stored samples of the classes. The handwritten characters are modeled as “rubber bands” which are deformed to match the actual images as accurately as possible. The needed amount of deformation is used as a dissimilarity measure between each model and the image. In many cases, vector template matching may be seen as a special case of the general setting of *dynamic programming* (see Hu 1982).

Williams (1994) and Revow et al. (1996) described a handwritten digit recognition method in which trainable deformable B-splines were interpreted as Gaussian “ink generators” along the length of the spline. The measured distribution of ink, i.e., the input image, was fitted to the stored models of pen movement and spreading of ink. The fitting was performed by using a modification of the Expectation-Maximization (EM) algorithm. Parker (1994) presented a digit recognition scheme in which the line width of the input image was estimated and used in the thickening of the skeleton templates. The final comparison was made bit-wise between the thickened templates and the actual input image. Thus, the algorithm was quite near the classical template-matching scheme described in Section 6.4.2, but was considerably more elaborate due to the skeletonization stage.

Wakahara (1993 and 1994) described an iterative technique that used local affine transformation for distortion-tolerant matching. In the method, skeletonized images were deformed by linear transforms applied to each skeleton pixel in the input image and in the models. Each transform was also applied to the neighboring pixels, which made the deformation of the image more continuous. As well, local structural information was stored into all pixels of the skeletons and used as a similarity measure between corresponding points in the input and model images. The optimal local deformations were finally resolved as a set of simultaneous linear equations.

In the method developed by Hastie and Tibshirani (1994), only the model templates were represented in the skeleton form. The sum of the smallest squared distances from all the black pixels in the input image to the skeleton was used as a dissimilarity measure. Using this measure, the best global affine transformation for the skeleton was selected. Additional dissimilarity values were introduced for the parts of the prototype not covered by the input image, for differing aspect ratio, for excessive rotation and shear, and for the cases where the mapping between the input image and the model was not continuous. A statistical feature vector was finally formed from these values. Del Bimbo et al. (1994) presented a method for matching a one-dimensional deformable template to a printed digit image of poor binarization and segmentation quality. They combined with linear discriminant functions four different similarity measures between the input image and each model template. In the experiments, the input images were not actually skeletonized prior to matching, because they were originally machine-printed and thin.

Graph Description A graph description of an image defines the subparts of the image and their spatial relationships. In character recognition, the image is most often represented as a skeleton. Its legitimate subparts are typically strokes, line ends, arcs, holes, etc. The recognized spatial relations may be coded as a simple grammar (e.g., Baptista and Kulkarni 1988), or as a binary decision tree (e.g., Brown et al. 1988). Another possibility is to store a set of representative graph templates in the memory of the system. Figure 6.9 (c) shows for illustration a handwritten digit ‘7’ as a graph representation which consists of three straight lines, one arc, one junction, one corner, and four endpoints.

Recognition schemes based on graph description have been used widely in recognition of Chinese characters. For example, Lu et al. (1991) formed a graph of the branches of the

character and performed graph matching. Lee and Chen (1992) first extracted line segments and then used their centers, slopes and corresponding mutual relations as features in matching with dynamic programming between the input character and the reference models. A somewhat similar approach was proposed by Rocha and Pavlidis (1994) for Western alphabets. Cheng et al. (1993) utilized *relaxation* (see Hummel and Zucker 1983) when calculating the similarity of two images. The characters were represented as sets of strokes, each of which also contained information about the neighboring strokes. A similar approach was applied to Latin characters by Lam and Suen (1988). In general, more elaborate preprocessing is needed in the processing of Asian character sets than of European alphabets. Various techniques for the former were reviewed and their effectiveness evaluated by Lee and Park (1994).

Even though most of the graph-description systems fall into the category of structural methods, statistical classification is still also feasible. Kahan et al. (1987) parameterized the linear strokes of the printed Roman alphabet by their location, orientation and length, and, then, clustered the resulting points in the parameter space. After pruning, the total number of clusters needed for presenting all the line segments that appeared in the training set was about 100. When holes, arcs, and other parts were also parameterized and clustered in a similar manner, the total number of clusters was 300, which then was the dimensionality of the statistical binary feature vector.

6.4.6 Discrete Features

Discrete features are a set of non-systematic heuristic features aimed rather to distinguish between the character classes than to describe the shapes of the characters. The selection of the features for a particular classification task is highly knowledge-driven, and, therefore, learning algorithms are scarcely practicable. Discrete features are mostly used with decision trees or in syntactic pattern recognition. Or *vice versa* – syntactic recognition of characters is almost always based on some form of discrete features.

The extraction of discrete structural features begins either from the original or skeletonized input image which can easily be converted to a collection of branches, ending points and junctions. The enumeration of all heuristic structural features used in recognition of handwritten digits is a formidable task. The following examples should give a sufficient comprehension of the vast variety of possibilities. First, the features may be isolated non-systematic measurements of the image. These include such as maximum width, aspect ratio, total stroke length, and the proportion of black pixels in the upper part of the image. Second, the features may be Boolean values that indicate the presence or absence of a certain characteristic, such as a dot or diacritics on the top of the image body. Third, some small integer values can be extracted, such as the numbers of intersections with straight lines at specific places, holes, connected components, strokes, junctions, and ending and crossing points. Also, the coordinates and other additional descriptions of these entities and extreme points may be included. The number of descriptive features may vary from an input image to another, which makes statistical treatment of the heuristic features difficult.

Purely heuristic features in recognition of handwritten digits were used by Pavlidis and Ali (1975). In their system, 14 integer-valued and 8 Boolean-valued features were used to create a classification tree with the total of 45 final nodes and 10 edges in the longest path. Mai and Suen (1990) developed a scheme in which the set of heuristic Boolean features was iteratively increased under human supervision. New features were added to the feature vector in order to resolve the misclassifications that occurred during the tentative classification of the training set.

It is possible to use heuristic features only in one stage of a multi-stage classification system. For example, Huang and Chuang (1986) first extracted some integer-valued heuristic features and then fitted the vector templates of those models which matched exactly the input image in the first stage. Shridhar and Badreldin (1984) developed a system in which the digits '0', '1', '3', and '8' were classified statistically, using Fourier descriptors. Of the remaining digit classes, '2's and '5's, on the one hand, and '4's, '6's, '7's, and '9's, on the other, were distinguished by heuristic features which were based on transitions from white to black pixels on horizontal and vertical scan lines in the four quadrants of the digit image. In another publication, the same authors described a syntactic recognition system (Shridhar and Badreldin 1985). Forty-eight feature predicates were derived from the contour profile presentation of the input digits. Twenty-seven digit models, each corresponding to its own production rule, were then used to represent the ten classes of digits.

6.4.7 Zoning: Combination of Low-Level Features

Zoning has already been mentioned in connection with other feature extraction methods. Actually, it is not a feature extraction method but an auxiliary procedure used in combination with actual feature extractors. In zoning, the area of the input image is split typically into four or nine subimages by using a rectangular grid. One, or some, of the above-described feature extraction methods is then applied to each of these regions separately. The final feature vector is formed by concatenating the features from the subimages. Both the benefits and shortcomings of the zoning method are quite obvious. Generally, it is advantageous that the characteristics of the individual parts of the image are not mixed, but processed separately. On the other hand, the fixed artificial borders that divide the image to its subparts increase variation, because small changes in the locations and relative sizes of the parts of a symbol easily lead to drastic changes in the contents of the subimages. This further leads to discontinuous changes in the resulting feature vector.

Veelaert (1994) presented an interesting algorithm related to zoning. His technique combines multiple low-level feature detectors into larger ones. Each of these lower-level features are described separately with a mask. On some occasions, it is also possible to decompose a set of large masks to another set of small masks which are able to detect exactly the same set of features as the original set. These ideas could be applied to feature extraction in recognition of handwritten symbols.

6.4.8 Error-Corrective Feature Extraction in OCR

The traditional view of feature extraction in pattern recognition applications has been that the system designer, using her expertise, selects the features to be used. Thus, from the viewpoint of classification, the features have been fixed before any actual classification experiments have been performed. Also, any information loss during the feature extraction phase has been definitive and irreversible. Therefore, it has been the task of the classifier design to overcome these problems. As neural classification algorithms have replaced traditional statistical and knowledge-based classifiers in many applications, the question has been raised as to whether the feature extraction phase might also be made adaptive and autonomously error-corrective. This chapter describes one such scheme called the *error-corrective feature extraction*. The goal is to reformulate the classical Karhunen-Loève Transform (KLT) in a manner which enables feedback from the classifier in a manner depicted with the arrow labeled “c)” in Figure 2.2 on page 25.

The error-corrective feature extraction scheme is presented in two formulations which complement each other. In the first, the features are not adaptive, but the feature selection is controlled by the resulting overall classification accuracy. In the second formulation, the feature extraction stage is revised with each individual misclassified vector in a manner familiar from the learning subspace classifiers. Thus, the feature extraction phase may be regarded as genuinely neural because it is adaptive and able to learn from examples.

Both forms of the error-corrective feature extraction originate from KLT. In the first formulation, the covariance matrix used in KLT is replaced with a sum matrix that enhances the directions between the classes and across the class borders. We begin with defining three matrices formed from the original normalized $w \times h$ -dimensional input vectors $\mathbf{f}$ and their estimated mean $\widehat{\boldsymbol{\mu}}_{\mathbf{f}}$ prior to any feature extraction:

$$\widehat{\boldsymbol{\Sigma}}_{\mathbf{f}} = \frac{\sum_{i=1}^n (\mathbf{f}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{f}})(\mathbf{f}_i - \widehat{\boldsymbol{\mu}}_{\mathbf{f}})^T}{n}, \quad (6.38)$$

$$\widehat{\mathbf{A}}_{\mathbf{f}} = \frac{\sum_{j=1}^c \sum_{i=1, i \neq j}^c \sum_{k=1}^{n_j} \sum_{l=1}^{n_i} (\mathbf{f}_{kj} - \mathbf{f}_{li})(\mathbf{f}_{kj} - \mathbf{f}_{li})^T}{\sum_{j=1}^c \sum_{i=1, i \neq j}^c n_j n_i}, \quad (6.39)$$

$$\widehat{\mathbf{B}}_{\mathbf{f}} = \frac{\sum_{j=1}^c \sum_{i=1, i \neq j}^c \sum_{k=1}^{n_j} \sum_{l=1}^{n_i} s_p(\mathbf{f}_{kj}, \mathbf{f}_{li}) (\mathbf{f}_{kj} - \mathbf{f}_{li})(\mathbf{f}_{kj} - \mathbf{f}_{li})^T}{\sum_{j=1}^c \sum_{i=1, i \neq j}^c \sum_{k=1}^{n_j} \sum_{l=1}^{n_i} s_p(\mathbf{f}_{kj}, \mathbf{f}_{li})}, \quad (6.40)$$

$$s_p(\mathbf{f}_{kj}, \mathbf{f}_{li}) = \begin{cases} 1, & \text{iff } li = \underset{tu, tu \neq kj}{\operatorname{argmin}} |\mathbf{f}_{kj} - \mathbf{f}_{tu}|, \\ 0, & \text{otherwise.} \end{cases} \quad (6.41)$$

In the notation, the normalized images which belong to the training sample and originate from the class j are denoted $\{\mathbf{f}_{1j}, \dots, \mathbf{f}_{n_jj}\}$. Thus, the $\widehat{\Sigma}_{\mathbf{f}}$ matrix is the covariance matrix used in KLT. The matrix $\widehat{\mathbf{A}}_{\mathbf{f}}$ models the average directions from one class to another. Every pair of vectors which belong to separate classes contributes to the sum. If the number of training vectors is large, only a fraction of the total number of terms is sufficient to estimate the $\widehat{\mathbf{A}}_{\mathbf{f}}$ matrix. $\widehat{\mathbf{B}}_{\mathbf{f}}$ goes even further; only the vector pairs which really are located in the areas of class borders are used. The pattern vector misclassification function $s_p(\mathbf{f}_{kj}, \mathbf{f}_{li})$ attains a non-zero value only when the vector $\mathbf{f}_{kj}$ is erroneously classified by using the leave-one-out 1-NN classification in the normalized input pattern space. In the limiting case of an infinite training sample, $\widehat{\mathbf{B}}_{\mathbf{f}}$ is formed from vectors perpendicular to the Bayesian class border in $\mathbb{R}^{w \times h}$.

A compound covariance-type matrix $\mathbf{S}$ is expressed as a linear combination of the three matrices above:

$$\mathbf{S} = \alpha \widehat{\mathbf{A}}_{\mathbf{f}} + \beta \widehat{\mathbf{B}}_{\mathbf{f}} + (1 - \alpha - \beta) \widehat{\Sigma}_{\mathbf{f}}. \quad (6.42)$$

In this feature extraction scheme, the kernel matrix $\mathbf{K} = \mathbf{K}(\alpha, \beta) \in \mathbb{R}^{wh \times d}$ is formed from the eigenvectors of the matrix $\mathbf{S}$ similarly to KLT in (6.15). Again, as mentioned in the general case of volume features and (6.11), the matrix $\mathbf{K}\mathbf{K}^T$ determines how faithfully the original metrics of the input space $\mathbb{R}^{w \times h}$ is transformed to the feature space $\mathbb{R}^d$. This is the rationale behind the use of the $\widehat{\mathbf{A}}_{\mathbf{f}}$ and $\widehat{\mathbf{B}}_{\mathbf{f}}$ matrices. It guarantees that the feature masks $\mathbf{k}_i$ describe more the differences between the classes than the overall shape of the input vector distribution. Optimal values for the multipliers α and β need to be found experimentally, for example by using error cross-validation and by observing the resulting classification performance. If the value of β is too large, poorly-representative eigenvectors $\mathbf{k}$ may emerge. This results from the observation that the estimation of $\widehat{\mathbf{B}}_{\mathbf{f}}$ may not be robust if the number of misclassified vectors is small, compared to the input vector dimensionality.

In the second formulation of the error-corrective feature extraction approach, the feedback from the classifier block in Figure 2.2 to the feature extraction block is not limited to the overall classification accuracy. Instead, each individual misclassified vector is used to revise the feature extraction process. The misclassifications are observed in the feature vector space $\mathbf{x} = \mathbf{K}\mathbf{f}$, but the corrections are made using the original normalized input images $\mathbf{f}$. Therefore, the process is able to improve the feature extraction stage. The matrix $\mathbf{S}$ is now made adaptive and its initial value can be obtained by using any of the matrices $\widehat{\Sigma}_{\mathbf{f}}$, $\widehat{\mathbf{A}}_{\mathbf{f}}$, and $\widehat{\mathbf{B}}_{\mathbf{f}}$, or their linear combination that results from the optimization process. The iteration formula is

$$\mathbf{S}(0) = n (\alpha \widehat{\mathbf{A}}_{\mathbf{f}} + \beta \widehat{\mathbf{B}}_{\mathbf{f}} + (1 - \alpha - \beta) \widehat{\Sigma}_{\mathbf{f}}), \quad (6.43)$$

$$\mathbf{S}(t+1) = \mathbf{S}(t) + \gamma(t) \sum_{j=1}^c \sum_{i=1, i \neq j}^c \sum_{k=1}^{n_j} \sum_{l=1}^{n_i} s_f(\mathbf{f}_{kj}, \mathbf{f}_{li}) (\mathbf{f}_{kj} - \mathbf{f}_{li})(\mathbf{f}_{kj} - \mathbf{f}_{li})^T, \quad (6.44)$$

$$s_f(\mathbf{f}_{kj}, \mathbf{f}_{li}) = \begin{cases} 1, & \text{iff } g(\mathbf{K}(t)^T \mathbf{f}_{kj}) \neq j, \text{ i.e., } \mathbf{K}(t)^T \mathbf{f}_{kj} \text{ is misclassified} \\ & \text{and } il = \underset{tu, tu \neq kj}{\operatorname{argmin}} |\mathbf{K}(t)^T \mathbf{f}_{kj} - \mathbf{K}(t)^T \mathbf{f}_{tu}|, \\ 0, & \text{otherwise.} \end{cases} \quad (6.45)$$

Again, the kernel matrix $\mathbf{K}(t+1)$ is formed from the principal eigenvectors of the $\mathbf{S}(t+1)$ matrix after the training period t . During each training epoch, all the n feature vectors, formed from the normalized input images $\mathbf{f}$ that use the transform $\mathbf{K}(t)$, are tentatively classified. The difference between the pattern vector misclassification function $s_p(\cdot, \cdot)$, in (6.41), and the feature vector misclassification function $s_f(\cdot, \cdot)$, in (6.45), is that $s_p(\cdot, \cdot)$ indicates the incorrectness of 1-NN classification in the high-dimensional $\mathbb{R}^{w \times h}$ pattern space, whereas $s_f(\cdot, \cdot)$ reports the misclassifications in the lower-dimensional feature vector space. Thus, the latter error-corrective feature extraction version is not simply an iterative version of the former. Instead, it models more accurately the actual operation of the combination of a feature extractor and a classifier. The coefficient $\gamma(t)$ in (6.44) needs to be given constant or decreasing positive values similarly to other uses of delta-rule correction (3.18). Any classification function $g(\mathbf{x})$ can be used in implementing (6.45), but a k -NN classifier is an apparent choice.

The optimal feature vector dimensionality d may vary when the $\mathbf{S}(t)$ matrix is evolving. Therefore, various values of d should be tested during each training epoch. Obviously, the training of the feature extraction block has to be stopped eventually. The overall classification accuracy can be monitored by using error cross-validation. The adaptation process can then be terminated when the classification accuracy stops improving. The feature extraction matrix $\mathbf{K}(t)$ of the lowest error rate can then be used. The relationship between the principles and formalisms of the latter error-corrective feature extraction method and the ALSM classification method, presented in Section 4.2.3, should be noticed. In both approaches, the outer products of all erroneously classified vectors are added to a scatter matrix the eigenvectors of which are recomputed after each training epoch. Therefore, it may be stated that the proposed feature extraction scheme is a successor of KLT in a manner similar to that of ALSM being a successor of CLAFIC.

6.5 OCR: From Parts to a Whole

This chapter has presented an overview of various aspects of character recognition, in particular, off-line recognition of handwritten digits. The focus of the presentation has mostly been on potential feature extraction methods. The options related to the normalization and other preprocessing stages have almost been ignored. As discussed in Sections 2.3 and 2.5, an efficient implementation of the stages preceding the feature extraction block is as essential as the choice of proper features. Median filtering (see Pratt 1991) and morphological smoothing (see Serra 1982) are, among others, feasible preprocessing techniques for optical character recognition systems. The knowledge of preprocessing and feature extraction methods is an important topic in the design of any pattern recognition application. In addition, this knowledge is, in general, specific to that particular application. On the contrary, the classification methods are more universal and one classification technique can, in general, be applied to a large variety of different applications.

The importance of the connection between the feature extraction and classification stages was addressed in the last few pages where the error-corrective feature extraction technique

was introduced. The error-correction technique demonstrated that a pattern recognition system can be more than the mere sum of its parts. If individual processing stages of a system are allowed or forced to interact, the overall performance can be expected to increase. The more the data processing can be made error-corrective and adaptive, the more flexible the system will be. Naturally, all this means increased computational complexity during the system design as the cost of advantages in recognition accuracy.

The aspects of the various classification techniques presented in Chapter 3 need to be considered when the combination of feature extraction and classification methods is selected. Due to the statistical nature of the classifiers of Chapter 3, they are easily combinable with the statistical feature extraction methods presented in this chapter. The structural features of this chapter call for the use of syntactic classification methods which have been omitted from this presentation. In any case, some design factors, such as the type of input and the available computer capacity, dictate the choices in the design of all the parts of a pattern recognition system. In this respect, feature extraction methods, such as the Karhunen-Loève Transform, which allow the selection of the feature vector dimensionality depending on the amount of data and computer capacity available, are the most scalable to any specific application. The computational resources need, therefore, to be carefully divided between the stages of a recognition system in order to obtain the best achievable overall performance.

Optical character recognition applications are almost always embedded systems in which the aspect of real-time computing is essential. Therefore, the balance between the accuracy of recognition and the time and computational resources available for each classification task has to be tuned carefully. Fortunately, many statistical feature extraction and classification methods are quite simple and can be implemented with special hardware. The realization of an all-purpose OCR system on a single adaptive VLSI chip can therefore be regarded as a long-term research and development goal in the field of character recognition research.

Chapter 7

Prototype Recognition System

This chapter describes the prototype recognition system implemented for the case study of classification of handwritten digits. The stages of the system are explained in the order of the processing. Some general issues and options related to the implementation are addressed. Section 7.6 describes the data produced with the prototype system. These data samples were originally produced for the benchmarking study reported by Holmström et al. (1996b and 1997). Here, they are used in the experiments described in the next chapter.

7.1 Image Acquisition

The first stage in optical character recognition is the data acquisition. There are primarily two ways of acquiring images: scanners and video cameras. Depending on the needs and facilities, both of them can produce either binary or gray-scale images with a varying number of bits per pixel. The acquisition of color images is also practicable, but seldom used, due to the large increase in the amount of data and only moderate gain in recognition accuracy. The scanning resolution may vary but it is typically between 200 and 400 dots per inch in both spatial directions. The resolution needed is inversely proportional to the expected size of the characters. The recognition system should be made independent of the original image size and resolution by image normalization.

In the prototype system, the images were scanned by using automatic feed for A4-sized paper documents. The resolution was 300 dots per inch in both directions. Binary images were produced by setting an appropriate threshold. The images were stored in binary form only. The fill-in form used with the prototype system was designed to imitate a tax form where each character has a predefined location where it is to be written. Lightly-colored boxes that guided the entry disappeared successfully as a result of the thresholding. All the participants of the data collection filled out identical forms, where the desired symbol was printed above each intended location of handwritten entry. One such image is shown on page 112.

OPTISEN LUVUN TESTILOMAKE

Täytä alla oleviin ruutuihin ruudun yläpuolella oleva merkki. Kirjaimet kirjoitetaan isoina ja tekstaten. Käytä lyijykynää, hyvälaatuista kuulakärkikynää tai ohutta tussikynää.

Esimerkiksi A B C

A B C

1	J	U	L	I	U	S		C	E	A	S	A	R	2	V	Ä	I	N	Ö
1	J	U	L	I	U	S		C	E	A	S	A	R	2	V	Ä	I	N	Ö
3	X	A	N	T	I	P	P	A	4	G	A	B	R	I	E	L			
3	X	A	N	T	I	P	P	A	4	G	A	B	R	I	E	L			
5	W	I	L	L	E	6	H	Å	K	A	N	7	D	A	V	I	D		
5	W	I	L	L	E	6	H	Å	K	A	N	7	D	A	V	I	D		
8	Z	E	U	S	9	F	A	N	N	Y	M	A	T	S	Q	U	O		
8	Z	E	U	S	9	F	A	N	N	Y	M	A	T	S	Q	U	O		
O	T	T	O	S	Ä	D	E	Å	K	E	P	Ä	I	V	Ö				
O	T	T	O	S	Ä	D	E	Å	K	E	P	Ä	I	V	Ö				
1	2	3	4	5	6	7	8	9	0	.	,	+	-	*	/				
1	2	3	4	5	6	7	8	9	0	.	,	+	-	*	/				

Figure 7.1: An example of the input data used in the experiments. The actual width of the input image is 206 and the height 293 millimeters.

7.2 Registration of Images

Even if the entry of data has been strictly constrained to specific fill-in areas, the system has to match the actual coordinates in the scanned image with those of the stored model of the form. In the prototype system, this registration process is based on locating four easily distinguished registration marks in the image. These marks appear in the corners of the printed area in Figure 7.1.

After the location of the registration marks, the translation and scaling factors are estimated in the least-squares fashion. If any of the marks contributes to the total error more than a preset amount, it is rejected, and the model is re-fitted without it. If there were more than four registration marks, more than one corrupted mark could be ignored, and the mapping between the model and the actual image would still be possible.

In extensively rotated images, the amount of rotation needs to be estimated. The image can then be rotated accordingly to correct the distortion. Luckily, in the case of the data used in the experiments, this step was unnecessary. If machine-printed text or other disturbing material remain in the image, they should be masked out in the preprocessing stage. Fortunately, the fill-in form of Figure 7.1 was designed so that this step was not needed either.

7.3 Digit Segmentation

An image of a group of handwritten digits has to be segmented into separate areas of one single digit before any of them can be classified. Figure 7.1 shows that the handwritten symbols are quite far from each other in the material used with the prototype system. Hence, they can be segmented easily, but still reliably. In general, the segmentation task is simple if (i) each symbol consists of only one connected component, and if (ii) the symbols do not touch their neighbors or other objects in the image. If the latter condition is strictly and the former almost always fulfilled, horizontal and vertical projections can be used to segment the image. This segmentation can then be fine-tuned with, for example, the following algorithm, used in the prototype system.

The segmentation is launched by extracting a rectangular subimage large enough to contain a digit. The coordinates of this block are calculated from the corresponding coordinates in the stored image model, by using the affine transform solved in the registration step. Starting from the center of this block, a rectangular area is expanded until all four sides touch the outer contour of the digit and there are no black pixels outside the rectangle closer than five pixels from its sides. The purpose of this stopping condition is to ensure that small vertical and horizontal gaps in the image will not break the digit incorrectly.

If the characters were heavily slanted, the amount of slant should be estimated first and the corresponding non-perpendicular projection operation should be used. Because the

digits in the material were clearly separated, the removal of slant was not needed during segmentation.

7.4 Normalization of Digits

Before feature extraction, the segmented handwritten digit images have to be normalized. The normalization stage is aimed to cancel the effects of some particular sorts of variation in the way the individual digit images have been drawn. Schürmann et al. (1992) list the following variations in handwriting and ways to compensate for them:

- *Rotation*: If the baseline of the written text differs from the horizontal scanning axis by a certain angle, a rotation by this angle normalizes the characters.
- *Slant*: For each character, the parameter of the slant or the shear must be determined. The objective of the de-shearing process is to reduce the intraclass variability. A practicable solution is to determine the regression line of the image and use it as a de-shearing parameter.
- *Stroke width*: It is feasible to calculate the black pixel area and the circumference of a character, and estimate the total stroke length and width from these. A proper normalization of the image can then be performed by morphological erosion or dilation procedures on the image (see Serra 1982).
- *Scale*: The class membership of a digit does not normally depend on its size. Even the aspect ratio may vary. Therefore, it is reasonable to standardize both. In general, any binary image can be transformed into a standard-size gray-value image by averaging the corresponding areas of the binary image. In character recognition, the perception of 'C' and 'c' is size-dependent. Therefore, it is important to store the original image sizes for later use by postprocessing algorithms.

In the prototype system, the second and fourth of these normalization options have been implemented. First, the images are normalized to the size of 32×32 pixels. Most of the digits are higher than they are wide, so they are shrunk or stretched vertically to the correct height. The image pixels are expanded or compressed horizontally so that the original aspect ratio is maintained. Thereafter, the digits are centered in the horizontal direction. In rare cases, when the width of a digit is larger than its height, similar normalization is performed by interchanging the roles of the horizontal and vertical directions. During the process, black pixels are assigned the value one and white pixels the value minus one.

In the second and last normalization stage, the slant of the image is removed. This is accomplished by first calculating the magnitude of the centered horizontal difference of the image. In other words, from the original image $f(x, y)$, $x, y \in \{1, \dots, 32\}$, a new spatial function is formed:

$$f_x(x', y') = |f(x' + 17, y' + 16) - f(x' + 16, y' + 16)|, \quad (7.1)$$

where $x' = -15, \dots, 15$ and $y' = -15, \dots, 16$. Then, a linear regression model $x' = ay'$ is estimated by using the minimum squared error method from the x' and y' -coordinates of the pixels of $f_x(x', y')$ with non-zero values. A positive value of the estimated parameter $\hat{a}$ indicates a forward slant and a negative value backward slant of the digit. The final normalized image $f_N(x, y)$ is then formed by sliding the rows of the original image horizontally by the amount of pixel positions indicated by the $\hat{a}$ parameter

$$f_N(x, y) = f\left(x + \frac{\hat{a}(y - 16.5)}{16}, y\right), \text{ for } x, y \in \{1, \dots, 32\}. \quad (7.2)$$

Finally, the pattern vector $\mathbf{f}$ is constructed by concatenating the pixel values of $f_N(x, y)$ row-wise.

7.5 Feature Extraction

When the prototype recognition system was planned, the first decisions concerned the choice of algorithms to be used in the classification phase. The selection favored statistical and neural classification methods. This choice then dictated that the feature extraction method should belong to the group of systematic statistical methods of Table 6.2. In preliminary experiments, poor classification accuracy was obtained with moments and contour profiles, whereas good accuracy was achieved with the Zernike moments and the Karhunen-Loève Transform. The KLT was used exclusively in the experiments.

The KLT features have also been used by other research groups, including many participants of the First Census Conference (Wilkinson et al. 1991) and the developers of the public domain recognition system of the National Institute of Standards and Technology (NIST) (Garris et al. 1994). The results of the classifier comparison made by Blue et al. (1994) were also obtained with KLT features. Based on the above experiences and on local preliminary tests, it was decided that the length of the feature vectors used would be 64 components maximum.

7.6 Data Sets

A set of 17 880 binary images of handwritten digits were extracted from 894 fill-in forms similar to the one shown in Figure 7.1. Every form contained exactly two instances of each digit class. Some partially or poorly filled, or erroneously scanned or binarized forms were completely rejected, even though some of their digits might still have been useful.

The material was divided into training and testing samples of equal size. No one contributed to both of the sets. Therefore, the two samples could be regarded as independent. All the *a priori* probabilities of the digit classes were equal to 0.1 in both sets. The number of images in both was 8 940.

The mean of the training set was calculated and subtracted from both sets. The covariance matrix of the training sample was estimated and its first 64 eigenvectors with the largest eigenvalues were calculated. The kernel matrix $\mathbf{K}$ of (6.15) was formed from the vectors and the final 64-dimensional KLT feature vectors were calculated as in (6.9).

Chapter 8

Comparison of Classification Methods

The results of the experiments performed with various classifiers are presented in this chapter. In all experiments, the same training and testing data sets have been used. The production of the data sets with the prototype of a handwritten digit recognition system was described in the previous chapter. It should be remembered that the results presented and the conclusions drawn only apply to the particular data sample and feature extraction method used.

First, Section 8.1 studies the use of class-conditional means with subspace classification methods. Section 8.2 reviews the selection of subspace dimensions. Section 8.3 investigates the proposed principle of weighting the projection measures in subspace methods. Section 8.4 verifies experimentally the validity of the introduced gamma probability density function interpretation of subspace classification. Section 8.5 demonstrates the employment of the Local Subspace Classifier introduced in this thesis. Section 8.6 examines the principle of error-corrective feature extraction. Section 8.7 studies the performance of the introduced Learning k -NN Classifier. Finally, Section 8.8 combines the classification accuracies achieved with different classifiers in the preceding sections with a large comparison of neural and statistical classification algorithms. In addition, results of two simple experiments, the first with a committee classifier, and the second with rejection option are presented.

If not otherwise stated, tenfold error cross-validation (see Section 3.6) has been used in finding the optimal values for all the parameters used by each algorithm. Likewise, the reported error percentages have been calculated by using the independent test set. In general, the methods studied are independent of the order in which the training sample is presented. Also, the methods do not depend on any random initial values. For these reasons, the classification accuracy of such methods can be given as a single scalar value. In the case of the Learning k -NN Classifier, however, both the order of presentation and the initialization matter. Therefore, the final classification accuracy is evaluated ten times and the mean and standard deviation of these outcomes are presented.

8.1 CLAFIC- μ and ALSM- μ

In Section 4.1.2, the basic subspace classification rule was described. Later, in Section 4.3.1, a replacement rule which takes into account class-specific means of the feature vector distributions was introduced. The experiments of this section demonstrate the effect of the suggested modified classification rule on the overall accuracy of the subspace classifier.

Two CLAFIC classifiers were created, one to be used with the original classification rule (4.5), and the other with the class-conditional variation (4.24). 64-dimensional feature vectors were first found to be the best for both. Using the cross-validation procedure, optimal values for the feature vector dimensionality d and the subspace dimension ℓ common to all the classes were found. For the former, the full-length 64-dimensional feature vectors produced the smallest cross-validated error count with both variations. The resulting percentages of classification error were evaluated by using the independent test sample. The results are shown in Table 8.1. The CLAFIC- μ variant, i.e., the class-conditional means, is clearly better. Also, the resulting subspace dimension common to all classes is smaller in that case. This means smaller computational complexity.

	CLAFIC	ALSM	parameters
SS	4.3	3.2	$d = 64, \ell = 29$, ALSM: $\alpha = \beta = 3.0$, 7 epochs
SS- μ	3.9	2.7	$d = 64, \ell = 25$, ALSM: $\alpha = \beta = 3.7$, 8 epochs

Table 8.1: Percentages of test set classification error with the standard (SS) and the class-conditional-mean (SS- μ) versions of the CLAFIC and ALSM classifiers.

After recording the CLAFIC and CLAFIC- μ performances, ALSM training was applied to both variations of the classifier. The subspace dimensions were kept fixed during the training. The number of ALSM training epochs and a common value for α and β in (4.18) were selected by cross-validation. Table 8.1 shows the resulting parameter values and the evaluated classification error percentages. The numbers confirm the previous judgment concerning the superiority of the method of class-conditional means. Comparison of the results of the table horizontally shows that ALSM training proves to be advantageous in both cases.

8.2 Selection of Subspace Dimensions

Section 4.3.2 presented three principles for selecting initial values for the subspace dimensions ℓ_j in a subspace classifier, and an iterative algorithm for the refinement of the dimensions. Preliminary experiments with these techniques were published in (Laaksonen and Oja 1996b). However, those results were obtained without cross-validation. This sec-

tion presents more extensive experiments which evaluate the initialization and the refining algorithms.

The results of the experiments of this section are presented in Table 8.2. All the tests were performed with 64-dimensional feature vectors which were found to be best in the experiments of the previous section. The tests were made twice, once for the traditional subspace classification rule (4.5), shown on the left side of the table, and once for the individual class means variation (4.24), shown on the right side of the table. In the first set of results, all the subspaces were given an equal dimension ℓ which was selected through cross-validation. Thus, the first results shown on the top line of the table are the same as those presented in the previous section. Next, the iterative refinement rule (4.28) was applied to the subspace dimensions used in CLAFIC. The process was repeated for 200 epochs. The combination of dimensions yielding the smallest cross-validated error count was then reestablished. The resulting error percentages for the test set were evaluated both before and after employing the ALSM training. These numbers are shown below the title “Iterated” and below the original ALSM result in Table 8.2.

CLAFIC & ALSM				CLAFIC- μ & ALSM- μ			
Selection	Initial	Iterated	ALSM	Initial	Iterated	ALSM	Selection
$\ell = 29$	4.3	—	3.2	3.9	—	2.7	$\ell = 25$
	4.3	3.9	3.4	3.9	3.6	2.9	
$\kappa_1 = 0.0088$	4.2	—	3.5	3.6	—	3.0	$\kappa_1 = 0.012$
	4.2	4.1	3.5	3.6	3.6	3.2	
$\kappa_2 = 0.102$	5.0	—	3.2	4.0	—	2.8	$\kappa_2 = 0.175$
	5.0	4.1	3.0	4.0	3.6	3.3	

Table 8.2: Percentages of test set classification error with the three ways of selecting the initial subspace dimensions, with and without intermediate iterative refinement of the dimensions.

In addition to making the initial subspace dimensions equal for all classes, two other approaches were described in Section 4.3.2. These were: (4.26), selecting a value κ_1 , which relates the decay of the eigenvalues to their sum, and (4.27), selecting a value κ_2 , which compares the amount of residual variance to the total variance. These parameters were, again, cross-validated. The resulting parameter values are shown in the leftmost and rightmost columns of the table. Similarly, the iteration (4.28) was applied to these initial combinations, and the ALSM training was employed to both the initial and the iterated classifiers. The results are shown in Table 8.2.

The κ -parameters were cross-validated by using the following procedure. The summed subspace dimension, i.e., $\sum_{j=1}^c \ell_j$, was varied from 10 to 640 with an increment of 10. In the first case, the value for κ_1 , which was common to all the classes and produced the desired sum of subspace dimensions by (4.26), was determined and used. In the second case, a similar procedure was performed with κ_2 and the rule (4.27). Thus, the κ -values were actually functions of the summed subspace dimension, which made the

process somewhat easier due to the integer nature of the optimization argument. The ALSM parameter α (and, thus, the equal β) was selected by using a two-stage procedure. First, the range of $[0, 10]$ was tested with an increment of 0.5. Second, the subrange, such as $[2, 4]$, which gave the best cross-validated final accuracy of classification was re-evaluated with an increment of 0.1. The maximum number of ALSM epochs was 15 in both stages.

Three general conclusions can be drawn from the results of Table 8.2. First, the conclusions of the previous section claiming (i) the superiority of the classification rule with the subtraction of the individual class means, and (ii) the advantageousness of the ALSM training, are reinforced by the results. Second, the classification accuracies after the iteration do not seem to differ substantially among the three initialization methods. Third, the iterative selection of subspace dimensions is beneficial for the CLAFIC classification, but the advantage gained is lost when the ALSM training is applied. It can be concluded that the ALSM training is such a powerful method that the selection of the subspace dimensions need not be separately concerned.

8.3 Weighted Subspace Projection Measures

Weighting of the subspace projecting measures was presented in Section 4.3.3 as a way to enhance the classification performance of the subspace classification rule (4.5). The use of weighting was given a general formalism in (4.30) and a specific solution (4.32), in which one real-valued parameter γ needs to be chosen.

Experiments similar to those in Section 8.1 were performed with the exception that the dimensionality of the feature vectors was fixed to $d = 64$ and that the new weight-decay parameter γ was cross-validated together with the selection of the common subspace dimension ℓ . The value for γ was selected from the range of $[-0.1, 0.1]$ with an increment of 0.01. In the course of the ALSM training, the eigenvalues in (4.32) naturally changed from one epoch to another, but the values for ℓ and γ were kept constant. The results are displayed in Table 8.3.

<i>w</i> -CLAFIC & <i>w</i> -ALSM			<i>w</i> -CLAFIC- μ & <i>w</i> -ALSM- μ		
Selection	Initial	ALSM	Initial	ALSM	Selection
$\ell = 29, \gamma = 0$	4.3	3.2	3.9	2.7	$\ell = 25, \gamma = 0$
$\ell = 30, \gamma = 0.03$	3.7	3.0	3.4	2.5	$\ell = 27, \gamma = 0.05$

Table 8.3: Percentages of test set classification error using standard ($\gamma=0$) and weighted subspace projection measures.

The numbers in Table 8.3 suggest that modest positive values for the weight parameter γ produce a better classification accuracy than the referential value $\gamma = 0$ on the topmost line. Again, the ALSM training seems to be advantageous, but, at the same time, it

diminishes the relative benefit of the weighting procedure to some extent. The results indicate that the use of class-specific means in the classification rule (4.24) can successfully be combined with the use of weighted distance measures.

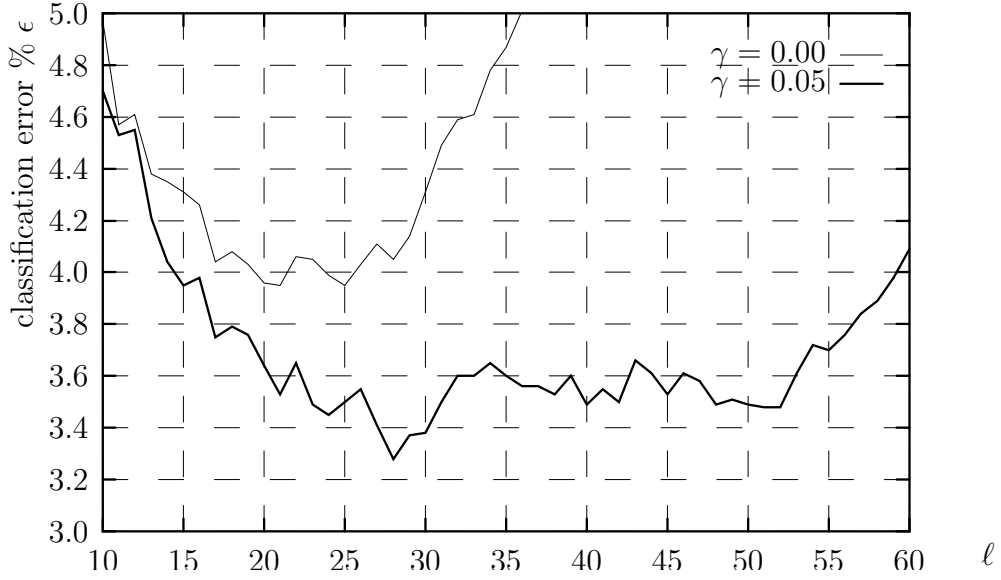


Figure 8.1: Test set classification errors $\epsilon(\ell)$ for unweighted CLAFIC- μ and for projection lengths weighted with $w_{ij} = (\lambda_{ij}/\lambda_{1j})^{0.05}$. Subspace dimension ℓ common to all classes has been used.

In addition to the enhanced accuracy of classification, the weighting also provides increased robustness in the selection of the subspace dimension parameter ℓ . This feature is illustrated in Figure 8.1. The upper curve shows the error percentages obtained with standard subspace projection operations in the case of CLAFIC- μ , and the lower curve with weighted projections and the weight parameter value $\gamma = 0.05$. In the latter case, the U-shape of the error curve $\epsilon(\ell)$ is both deeper and wider. Thus, it produces a better classification accuracy with diminished dependency on the selected parameter value ℓ . Due to the cross-validation process, in the latter case, the real minimum of the error curve, $\ell = 28, \epsilon = 3.3$, is missed in Table 8.3 where $\ell = 27$ yields 0.1 percentage points less accuracy.

8.4 Probability Density Function Interpretation

This section compares the traditional subspace classification rule (4.5) with the new rule based on the probability density function interpretation (4.37). In Table 8.4, the line labeled “SS” shows the results for the former, whereas the results for the latter are marked as “SS-PDF”. The tests were, again, performed by using the full-length 64-dimensional feature vectors. First, the classification errors were calculated by using the standard subspace classification rule. Then, the gamma probability density function interpretation was

applied, and the corresponding classification accuracies were evaluated. All the subspaces were given equal dimensionality ℓ . It was varied in order to find the best cross-validated selection individually for all the four classifiers. In Table 8.4, the values for ℓ are shown after the error percentages.

	CLAFIC	CLAFIC- μ
SS	4.3, $\ell = 29$	3.9, $\ell = 25$
SS-PDF	4.4, $\ell = 29$	3.9, $\ell = 27$

Table 8.4: Percentages of test set classification error with the traditional (SS) and with the classifier based on the gamma probability density function interpretation of the projection residual lengths (SS-PDF).

The performed classification experiments show that the classification accuracies of the conventional and the proposed subspace method do not differ significantly from one another. Therefore, the advantages of the proposed variation are, in this case, thus more of a theoretical than practical nature. The theoretical profits of the method are, (i) the new method facilitates a plausible rejection rule for input vectors that are difficult to classify, and, (ii) the new method is able to compensate for an asymmetric *a priori* distribution of the input vectors between the classes.

8.5 Local Subspace Classifier

Chapter 5 introduced a new classification technique named the Local Subspace Classifier (LSC) and its modification named LSC+. This section presents the experiments performed by employing these two novel methods. As usual, cross-validation was used in obtaining all the necessary parameters for the classifier. In the case of LSC and LSC+, these are d , the dimensionality of the input feature vectors, and D , the common dimensionality of the linear manifolds. The cross-validation process started with the configuration $d = 64, D = 0$. The value of D was incremented by one until the classification error had clearly started to increase. After that, the value of d was decremented by one and the process continued with $D = 0$. The two classification results obtained with the independent testing sample are shown in Table 8.5.

	ϵ -%	parameters
LSC	2.5	$d = 64, D = 12$
LSC+	2.1	$d = 64, D = 23$

Table 8.5: Percentages of test set classification error with the LSC and LSC+ classifiers.

The optimal value of the manifold dimension D is clearly larger in the LSC+ than in the LSC which does not require the convexity of the subspace projection. At the same time, the classification accuracy is significantly better in the former. A potential explanation for these two observations is that the convexity requirement allows more prototypes to be tentatively used in the classification decision. Due to the convexity condition, only that subset of the involved prototypes, which is the most compatible with the input vector, is then actually used. Figure 8.2 shows how the classification error rates measured with the independent testing set developed when the manifold dimension D was increased. Again, the real minima of the error curves are missed slightly in the cross-validation process, as can be seen by comparing Table 8.5 and Figure 8.2.

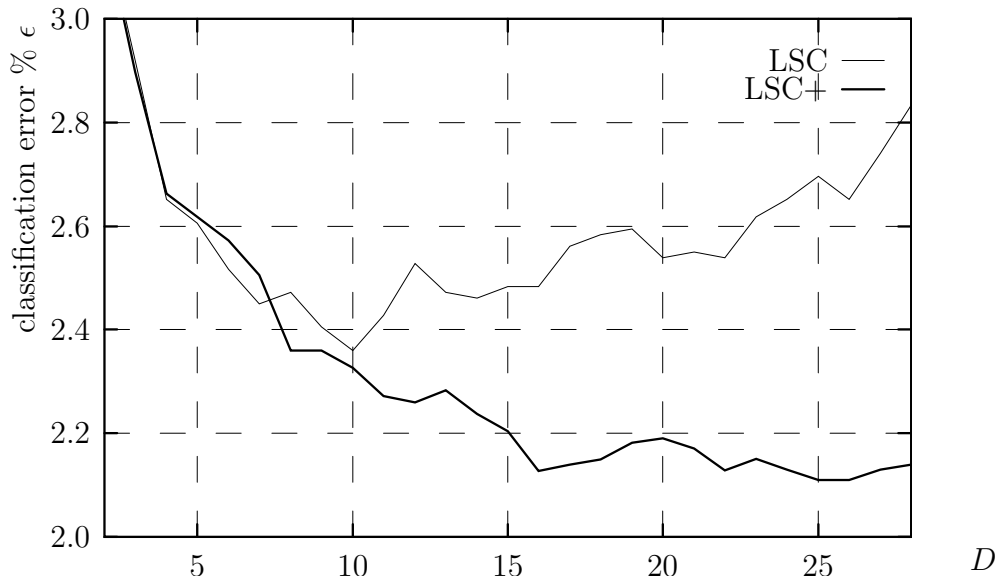


Figure 8.2: Test set classification error rates of the LSC and LSC+ classifiers as a function of the common manifold dimension D .

8.6 Error-Corrective Feature Extraction

The experiments in this section demonstrate the use of the error-corrective feature extraction method. Cross-validation was not used in performing the experiments, because the results would not be directly comparable to the other classification results, which will be summarized in Section 8.8. The error percentages given in this section can thus be compared only to each other.

Section 6.4.8 presented two ways to incorporate error-corrective actions in the feature extraction stage of a pattern recognition system. In the first technique, three correlation-type matrices, $\widehat{\Sigma}_{\mathbf{f}}$, $\widehat{\mathbf{A}}_{\mathbf{f}}$, and $\widehat{\mathbf{B}}_{\mathbf{f}}$, were formed using (6.38)–(6.41) from the training sample prior to feature extraction. At the first stage of the experiments, integer values were given

to α and β in (6.42). Feature extraction was performed accordingly. The classification error rate was evaluated using the 3-NN classification rule. The optimal dimension for the feature vector was selected “heretically” by observing the error rate obtained with the independent test sample. The error percentages and feature vector dimensionalities d are shown in Table 8.6.

α	β	ϵ -%	parameters
0	0	3.77	$d = 28$
1	0	3.75	$d = 28$
0	1	3.66	$d = 39$
0.1667	0.9167	3.45	$d = 33$

Table 8.6: Classification error percentages with different values of α and β in (6.42) and a 3-NN classifier.

At the second stage, the values of α and β were discretized to a grid with spacing of 1/12th of unity. The search concentrated on the most promising area around $(\alpha = 0, \beta = 1)$ in which error rates were computed. Some of them are displayed in Figure 8.3. The best attained error rate, with parameter values $\alpha = 0.1667$ and $\beta = 0.9167$, is also shown on the bottom line of Table 8.6.

In the second experiment, the matrix $\mathbf{S}(t)$ was modified iteratively by using (6.44) and (6.45), starting from the situation $\mathbf{S}(0) = \widehat{\Sigma}_{\mathbf{f}}$ in (6.43). The correction-rate parameter γ in (6.44) was given values in the range of $[0.5, 5]$ with an increment of 0.5, and the development of the classification accuracy was observed. After each iteration, the optimal feature vector dimensionality $d(t+1)$ was selected from the range of $[d(t) - 5, d(t) + 5]$. The evolution of the classification accuracy with $\gamma = 3$ is shown in Figure 8.4. The best attained results are tabulated in Table 8.7, together with the best results of Table 8.6.

	ϵ -%	parameters
KLT features	3.77	$d = 28$
1st method	3.45	$d = 33, \alpha = 0.1667, \beta = 0.9167$
2nd method	3.33	$d = 29, \gamma = 3$

Table 8.7: Final classification error percentages obtained with the two forms of error-corrective feature extraction. The classifier used in the experiments was a 3-NN classifier.

The classification performances in Table 8.7 show that both proposed error-corrective feature extraction methods lowered substantially the overall error rate in comparison to the original accuracy obtained with the Karhunen-Loève transformed features. But, as shown in Figure 8.4, the error-correction process of the second approach is somewhat unstable, oscillating with an amplitude of approximately 0.15 percentage points. In order to enforce

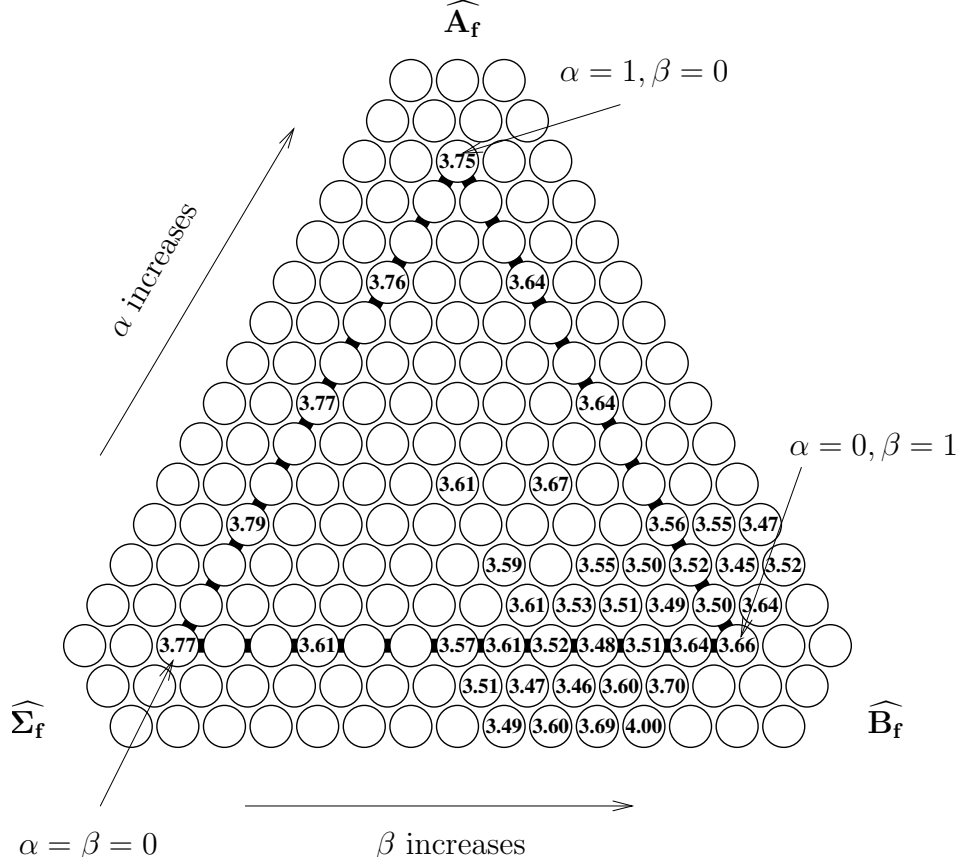


Figure 8.3: Classification accuracies with different combinations of the α and β multipliers in (6.42).

the convergence, the value of γ in (6.44) might be made monotonically decreasing with time.

8.7 Learning k -NN Classifier

Section 3.4.3 introduced the principle of the Learning k -NN Classifier (L- k -NN). In the experiments reported by Laaksonen and Oja (1996a), the learning rule #1 was found to be the best of the three proposed variants. Therefore, it was solely used in these experiments. In the experiments reported by Holmström et al. (1997), the k -NN classifier with values $k = 1$ and $k = 3$ was used, and the optimal feature vector dimensionality d was selected with cross-validation. Now, k was also selected with cross-validation, and the configuration $k = 3$ and $d = 38$ showed the best performance. These two parameters were used also with the L- k -NN and the absolute value of the adaptation strength parameter $\alpha(t)$ was fixed to be constantly equal to 0.1. Thus, ℓ , the number of prototypes used in the classifier, and the optimal number of training epochs remained to be solved. These two

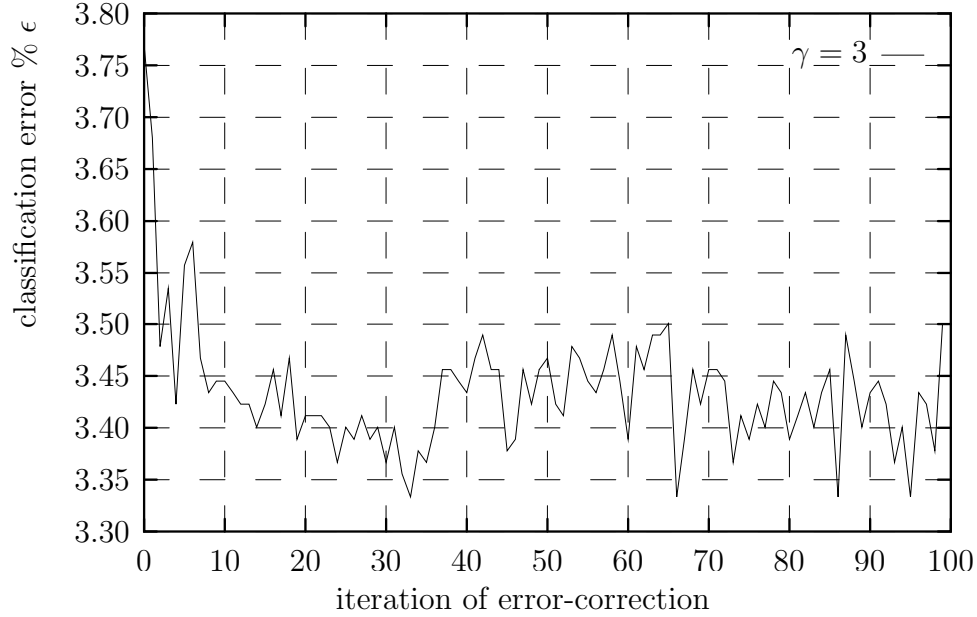


Figure 8.4: The evolution of the classification error rates in the error-corrective feature extraction using $\gamma = 3$ in (6.45). The best classification error rate, $\epsilon = 3.33\%$, is attained for the first time at iteration index 33.

were selected together by using the training set cross-validation procedure. The size of the codebook ℓ was varied from 100 to 8 925 with an increment of 25. The maximum number of training epochs was equal to ten. In the initialization of the prototype sets, the vectors were randomly picked from the central vectors of each class, i.e., three closest training vectors to each initial codebook vector belonged to the same class as the prototype itself.

	ϵ -%	σ_ϵ -%	parameters
k -NN	3.8		$k = 3, d = 28$
L- k -NN Rule #1	3.6	0.1	$\ell = 5\,750, \text{ \#epochs}=7$

Table 8.8: Percentages of test set classification error for the k -NN and L- k -NN classifiers.

The resulting parameter values and the obtained classification error rate with the independent testing set are shown in Table 8.8. The mean value ϵ and the standard deviation of the recognition error percentage σ_ϵ were estimated from ten different and independent initializations of the prototype set. The accuracy of the standard k -NN rule, with $k = 3$ and $d = 38$, is presented for comparison. The result shows that the learning version of the k -NN classification rule enhanced the classification accuracy only by 0.2 percentage points. The number of prototypes $\ell = 5\,750$ is, however, smaller than the training set size 8 940. This means less computation during the recognition phase.

8.8 Summary of Classification Results

This section summarizes all the cross-validated and mutually comparable classification results presented either by Holmström et al. (1997) or in the preceding sections of this chapter. The summary is given in Table 8.9. The tabulation of the classification error percentages contains 13 results obtained with classifiers of Chapter 3. Those figures are given merely for the purpose of comparison. There are also two results of the traditional subspace classification methods. More importantly, there are 10 results produced by employing the novel methods introduced in this thesis. These methods include the Learning k -NN classifier (L- k -NN), in Table 8.8; the version of the subspace rule that utilizes class-conditional subtraction of means (CLAFIC- μ) and its employment with the Averaged Learning Subspace Method (ALSM- μ), in Table 8.1; the weighted projection measure approach (w -CLAFIC- μ), (w -ALSM- μ), and (w -ALSM) in Table 8.3; the gamma probability density function interpretation of the subspace rule (CLAFIC-PDF) and (CLAFIC- μ -PDF), in Table 8.4; and, finally, the Local Subspace Classifier and its convex modification (LSC) and (LSC+), in Table 8.5. The experiments concerning the initial and iterated selection of the subspace dimensions presented in Table 8.2 did not show significant improvement over the initial accuracies. Therefore, none of those results are repeated here.

The RKDA, MLP, L- k -NN, and LVQ classifiers need initialization which makes their test-set performance depend either on a set of randomly generated real numbers or on the particular subset of training data used for initialization. In order to take this into account, each of these methods has been evaluated by computing the average performance and its standard deviation in ten independent trials.

To make the comparison of the percentages easy, all the classifiers with classification error smaller than 4.5% in Table 8.9 are also shown ordered on a line in Figure 8.5. Some general conclusions can be drawn from these accuracies. First, the discriminant analysis methods, QDA, RDA, KDA, performed relatively well. This can be interpreted as an indirect indication that the distribution of the data closely resembles the Gaussian distribution in the Bayesian class-border areas. Second, MLP performed quite badly without weight-decay regularization. The results obtained by the tree classifier and MARS were also disappointing. Third, the learning, or adaptive, algorithms, such as ALSM and L- k -NN, performed better than their non-adaptive counterparts, such as CLAFIC and k -NN.

Of the novel classification methods presented in this thesis, some conclusive remarks can be made. First, the performance of the Learning k -Nearest Neighbors algorithm was only marginally better than the performance of the standard k -NN rule. Thus, it was placed among the bulk of the tested methods. Second, those modified CLAFIC classifiers which are based on the gamma probability density function interpretation of the subspace rule, i.e., CLAFIC-PDF and CLAFIC- μ -PDF, did not exceed their traditional counterparts in accuracy. Consequently, the value of the density function interpretation is more in the new theoretical view it offers to the principle of subspace classification. Third, the approach in which the projection measures are weighted in the subspace classification function improved the classification performance substantially. For example, w -ALSM reduced

classifier	ϵ -%	σ_ϵ -%	parameters
LDA	9.8		$d = 64$
QDA	3.7		$d = 47$
RDA	3.4		$d = 61, \gamma = 0.25, \lambda = 0$
KDA1	3.7		$d = 32, h = 3.0$
KDA2	3.5		$d = 36, h_1, \dots, h_{10}$
RKDA	5.2	0.1	$d = 32, \ell = 35$
MLP	5.4	0.3	$d = 36, \ell = 40$
MLP+WD	3.5	0.1	$[d = 36, \ell = 40], \lambda = 0.05$
LLR	2.8		$d = 36, \alpha = 0.1$
Tree classifier	16.8		$d = 16, 849$ terminal nodes
FDA/MARS	6.3		$d = 32, 195$ second-order terms
k -NN	3.8		$k = 3, d = 38$
L- k -NN	3.6	0.1	$[k = 3, d = 38, \alpha = 0.1], \ell = 5750, \#epochs = 7$
LVQ	4.0	0.1	$[d = 38, \alpha(0) = 0.2, w = 0.5, 10 \text{ epochs LVQ1}], \ell = 8000, 1 \text{ epoch LVQ2}$
CLAFIC	4.3		$d = 64, \ell = 29$
ALSM	3.2		$[d = 64, \ell = 29], \alpha = \beta = 3.0, \#epochs = 7$
CLAFIC- μ	3.9		$d = 64, \ell = 25$
ALSM- μ	2.7		$[d = 64, \ell = 25], \alpha = \beta = 3.7, \#epochs = 8$
w -CLAFIC- μ	3.4		$\ell = 27, \gamma = 0.05$
w -ALSM- μ	2.5		$[\ell = 27, \gamma = 0.05], \alpha = \beta = 3.1 \#epochs = 6$
w -ALSM	3.0		$[\ell = 30, \gamma = 0.03], \alpha = \beta = 2.8 \#epochs = 8$
CLAFIC-PDF	4.4		$\ell = 29$
CLAFIC- μ -PDF	3.9		$\ell = 27$
LSC	2.5		$d = 64, D = 12$
LSC+	2.1		$d = 64, D = 23$
Committee	2.5		[LLR,ALSM,L- k -NN]

Table 8.9: Summary of testing set classification error percentages for various classification algorithms. For some methods, the estimated standard deviation in ten independent trials is also shown. The parameters given within square brackets were determined without cross-validation, for example, taken from the classifier on the previous line.

the error level of ALSM by 0.2 percentage points to 3.0 percent. Fourth, the proposed variation of the subspace classification rule that uses class-specific means performed very well. The result of $\text{ALSM-}\boldsymbol{\mu}$ was the fourth best obtained in the comparison. Fifth, the weighting of the projection measures and the use of class-conditional means can be successfully used together: the performance of the $w\text{-ALSM-}\boldsymbol{\mu}$ classifier was the second best. Sixth, and most importantly, the Local Subspace Classifier (LSC) did very well in handwritten digit classification. Its LSC+ variant was clearly superior to all the other classifiers in the comparison.

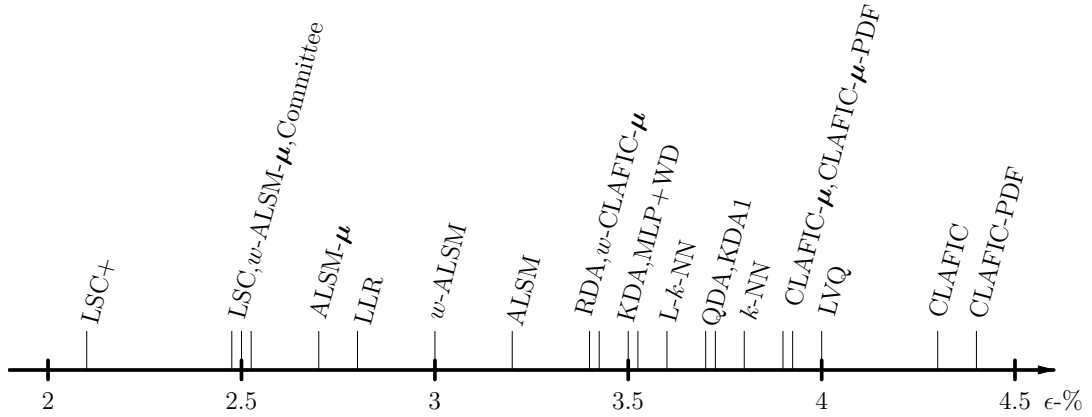


Figure 8.5: The classification error percentages of Table 8.9 on a line. The results of the LDA, RKDA, MLP, tree, and FDA/MARS classifiers were omitted because of poor performance.

As the final two experiments, a committee classifier and the rejection of digits that were difficult to classify were implemented. The committee classifier, the results of which are shown in the last line of Table 8.9, was formed utilizing the majority-voting principle from the LLR, ALSM, and $L\text{-}k\text{-NN}$ classifiers. In the article by Holmström et al. (1997), this set of classifiers was employed in the committee because LLR and ALSM were the two best classifiers in the original comparison and $L\text{-}k\text{-NN}$ was the best of the tested prototype-based classifiers. It was presumed that such a selection, in which the classification principles of the committee members were all different, would produce the best joint performance. Each member of the committee utilized the parameter values optimized earlier. Therefore, the feature vector dimensionality d was different for all three and varied from 36 to 64. As expected, this committee quite clearly outperformed all its individual members and was as good as the LSC and $w\text{-ALSM-}\boldsymbol{\mu}$ classifiers alone.

The rejection option was tested by using the LLR classifier. By varying the rejection threshold θ of (3.20), the reject-error curve shown in Figure 8.6 was obtained. The three diamonds in the figure display the reject-error trade-off points of the above described committee classifier. The first point was obtained when rejection was not allowed. The other two resulted from two voting strategies which rejected the input digit (i) in the case

of any disagreement, and (ii) only in the case of total disagreement between the committee members. As anticipated in Section 3.7, the error-rejection curve is nearly linear in the $\rho \log \epsilon$ -plane.

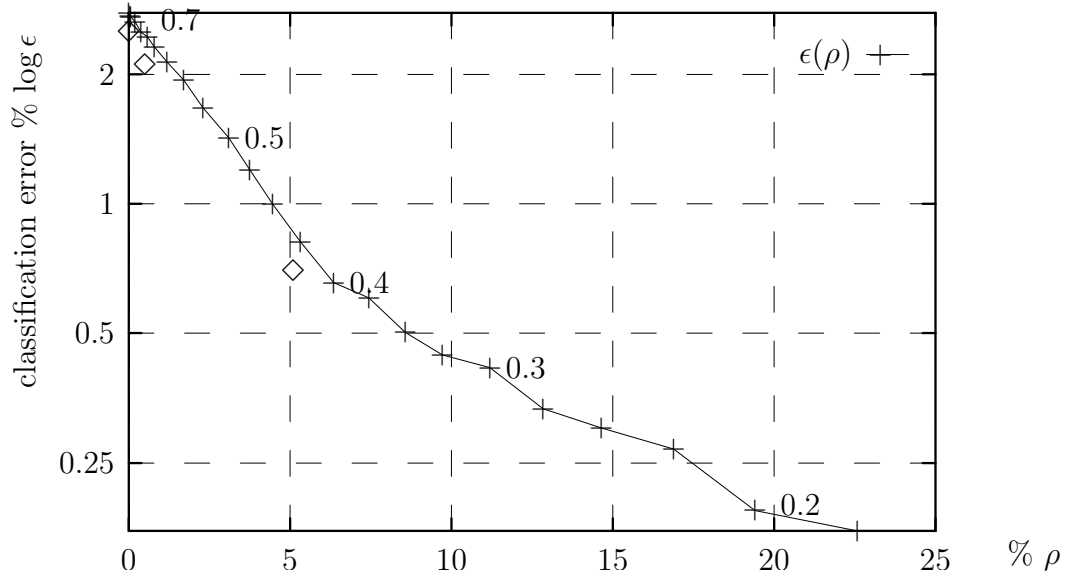


Figure 8.6: The error-reject curve for the LLR classifier. The rejection percentage is shown on the horizontal axis. The logarithmic vertical axis displays the error percentage for the non-rejected digits. The threshold parameter θ is given at selected points. The three diamonds indicate the results obtained with the committee classifier.

Chapter 9

Conclusions

The objective of this thesis has been twofold. First, statistical and neural classification methods were studied at large. Various aspects and characteristics of the methods were addressed in order to create a solid view of the different approaches of statistical classification. Also, a novel taxonomy of classification methods and a characterization of the properties that make a classification algorithm neural were presented. The taxonomy helps the designer of a pattern recognition system select a set of different classification techniques to be evaluated when selecting a classifier for a real-world pattern recognition application. Even though the quality of the classification stage is a key factor in the performance of the entire pattern recognition system, all the other blocks of the system need to be designed with similar care. Therefore, an overview of all the stages in a pattern recognition system was presented. In particular, the roles of the adaptive and mutually communicating parts were emphasized.

Among the wide variety of classification algorithms, the subspace methods were chosen for more extensive examination. Some new modifications of the subspace methods were suggested. Among them, the use of class-specific means in the classification rule is quite an obvious practice. The proposed weighting of projection measurements may be regarded as a generalization of an existing method. The iteration rule for the refinement of the subspace dimensions is a computationally demanding heuristic algorithm which works in an error-driven manner. All these modifications improved the overall classification accuracy of the subspace method. The novel gamma-function-based probability density function interpretation of the subspace rule is a new theoretical view to the principles of subspace classification. Most importantly, however, a new classification technique, the Local Subspace Classifier, was introduced together with its extension, the Convex Local Subspace Classifier.

The other main topic of this study was off-line optical recognition of handwritten digits. It was presented as the case study of the experiments. This particular branch of pattern recognition study was reviewed in an extensive introduction chapter. Special attention was directed at feature extraction methods applicable to character recognition. As a novel contribution to the feature extraction problem, an error-corrective feature extraction

algorithm was proposed. The method converts feature extraction to an adaptive and cooperating process which is able to learn to avoid classification errors. A prototype of a handwritten digit recognition system was developed and used for experiments.

A series of experiments was performed for the evaluation of the performance of the proposed algorithms. First, a careful comparison of 13 different statistical and neural classification methods was performed for the sake of comparison. Then, the new methods introduced in this study increased the number of comparable figures of merit to a total of 25. The results of these experiments showed that, of the proposed improvements on the subspace classification rule, the weighting of projection measures and the application of class-conditional means in the classification function improved significantly the accuracy attainable with the Averaged Learning Subspace Method, ALSM. These modifications diminished error to the level of the best other methods included in the extensive comparison.

The most striking improvement of classification accuracy was, however, attained with the novel Local Subspace Classifier, and, especially, with its Convex Local Subspace Classifier variant. These two methods were clearly superior in performance to all other classifiers in the comparison. This might be explained by the nature of the new methods which seek to combine the benefits of both the semiparametric subspace methods and the nonparametric prototype-based methods.

The performances of the remaining novel classifiers introduced in this thesis were modest in the experiments. From a theoretical point of view, however, the novel probability density function interpretation of the subspace classification rule, and the proposed scheme for the error-corrective feature extraction, are valuable. As such, they have brought new insight into the understanding of statistical classification principles.

The experiments described were carried out by employing training set cross-validation in the parameter selection. The reported classification accuracies were obtained with an independent testing sample. This procedure ensured an impartial treatment of the diverse algorithms. The applicability of the results and conclusions is limited only to the considered case study of recognition of handwritten digits and to the selected feature extraction. It can be anticipated, however, that similar results can be attained in other classification tasks where two-dimensional isolated visual objects are involved.

The argument forwarded in the introduction stated that subspace classification methods, when enhanced with the suggested modifications, are very useful in optical character recognition tasks. The experimental evidence presented in this study supports that statement.

Bibliography

- Ahmed, N., T. Natarajan, and K. R. Rao (1974). Discrete cosine transform. *IEEE Transactions on Computers* 23(1), 90–93.
- Akaike, H. (1974). Stochastic theory of minimal realization. *IEEE Transactions on Automatic Control* 19, 667–674.
- Ali, F. and T. Pavlidis (1977). Syntactic recognition of handwritten numerals. *IEEE Transactions on Systems, Man, and Cybernetics* 7(7), 537–541.
- Alt, F. L. (1962). Digital pattern recognition by moments. *Journal of the Association for Computing Machinery* 9(2), 240–258.
- Andersson, P. L. (1969). Optical character recognition—a survey. *Datamation* 15(7), 43–48.
- Andersson, P. L. (1971). OCR enters the practical stage. *Datamation* 17(23), 22–27.
- Andrews, H. C. (1971). Multidimensional rotations in feature selection. *IEEE Transactions on Computers* 20(9), 1045–1051.
- Andrews, H. C. (1972). *Introduction to mathematical techniques in pattern recognition*. John Wiley & Sons Inc.
- Balm, G. J. (1970). An introduction to optical character reader considerations. *Pattern Recognition* 2(3), 151–166.
- Baptista, G. and K. M. Kulkarni (1988). A high accuracy algorithm for recognition of handwritten numerals. *Pattern Recognition* 21(4), 287–291.
- Becker, R. A., J. M. Chambers, and A. R. Wilks (1988). *The NEW S Language*. New York: Chapman & Hall.
- Belkasim, S. O., M. Shridhar, and M. Ahmadi (1991). Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognition* 24(12), 1117–1138. Corrigendum in *Pattern Recognition* Vol. 26, No. 2, p. 377.
- Bellman, R. (1961). *Adaptive control processes: a guided tour*. Princeton, NJ: Princeton University Press.

- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.
- Blayo, F., Y. Cheneval, A. Guérin-Dugué, R. Chentouf, C. Aviles-Cruz, J. Madrenas, M. Moreno, and J. L. Voz (1995). Deliverable R3-B4-P Task B4: Benchmarks. Technical report, ESPRIT Basic Research Project Number 6891. Available as <<http://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases/Benchmarks.ps.Z>>.
- Blessner, B. A., T. T. Kuklinski, and R. J. Shillman (1976). Empirical tests for feature selection based on psychological theory of character recognition. *Pattern Recognition* 8(2), 77–85.
- Blue, J. L., G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson (1994). Evaluation of pattern classifiers for fingerprint and OCR applications. *Pattern Recognition* 27(4), 485–501.
- Bottou, L., C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Y. Simard, and V. Vapnik (1994). Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 77–82. IAPR.
- Bow, S.-T. (1992). *Pattern Recognition and Image Preprocessing*. Marcel Dekker, Inc.
- Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Chapman & Hall.
- Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, San Mateo, CA, pp. 211–217. Morgan Kaufmann Publishers.
- Britannica Online (1996). Encyclopædia Britannica on the Internet. <<http://www.eb.com/>>.
- Broomhead, D. S. and D. Lowe (1988). Multivariate functional interpolation and adaptive networks. *Complex Systems* 2, 321–355.
- Brown, R. M., T. H. Fay, and C. L. Walker (1988). Handprinted symbol recognition system. *Pattern Recognition* 21(2), 91–118.
- Burges, C. J. C., J. I. Ben, J. S. Denker, Y. LeCun, and C. R. Nohl (1993). Off line recognition of handwritten postal words using neural networks. *International Journal of Pattern Recognition and Artificial Intelligence* 7(4), 689–704.
- Burr, D. J. (1988). Experiments on neural net recognition of spoken and written text. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36(7), 1162–1168.
- Cardoso, J.-F. and B. H. Laheld (1996). Equivariant adaptive source separation. *IEEE Transactions on Signal Processing* 44(12), 3017–3030.

- Chambers, J. M. and T. J. Hastie (Eds.) (1992). *Statistical Models in S*. New York: Chapman & Hall.
- Chen, C. H., L. F. Pau, and P. S. P. Wang (1993). *Handbook of Pattern Recognition and Computer Vision*. World Scientific Publishing.
- Cheng, B. and D. Titterton (1994). Neural networks: A review from a statistical perspective. *Statistical Science* 9(1), 2–54. With comments.
- Cheng, F.-H., W.-H. Hsu, and M.-C. Kuo (1993). Recognition of handprinted Chinese characters via stroke relaxation. *Pattern Recognition* 26(4), 579–593.
- Cheng, Y.-Q., K. Liu, and J.-Y. Yang (1993). A novel feature extraction method for image recognition based on similar discriminant function (SDF). *Pattern Recognition* 26(1), 115–125.
- Chow, C. K. (1957). An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers* 6, 247–254.
- Clark, L. A. and D. Pregibon (1992). Tree-based models. In J. M. Chambers and T. J. Hastie (Eds.), *Statistical Models in S*, Chapter 9. New York: Chapman & Hall.
- Cleveland, W. and C. Loader (1995). Smoothing by local regression: Principles and methods. Technical report, AT&T Bell Laboratories. Available as <<http://netlib.att.com/netlib/att/stat/doc/95.3.ps>>.
- Cleveland, W. S. and S. J. Devlin (1988). Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association* 83, 596–610.
- Comon, P. (1994). Independent component analysis – a new concept? *Signal Processing* 36(3), 287–314.
- Conover, W. J. (1980). *Practical Nonparametric Statistics 2ed*. John Wiley & Sons Inc.
- Cover, T. M. and P. E. Hart (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13(1), 21–27.
- Craven, P. and G. Wahba (1979). Smoothing noisy data with spline functions. *Numerical Mathematics* 31, 317–403.
- Dasarathy, B. V. (1991). *Nearest Neighbor Pattern Classification Techniques*. IEEE Computer Society Press.
- Daugman, J. G. (1988). Complete discrete 2-D gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36(7), 1169–1179.
- De Haan, G. R. and Ö. Egecioglu (1991). Links between self-organizing feature maps and weighted vector quantization. In *Proceedings of 1991 International Joint Conference on Neural Networks*, Volume 1, Singapore, pp. 887–892. IEEE, INNS.

- Del Bimbo, A., S. Santini, and J. Sanz (1994). Ocr from poor quality images by deformation of elastic templates. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 433–435. IAPR.
- Demartines, P. and J. Hérault (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks* 8(1), 148–154.
- Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1), 1–38.
- Devijver, P. A. and J. Kittler (1982). *Pattern Recognition: a Statistical Approach*. London: Prentice Hall International.
- Devroye, L. (1988). Automatic pattern recognition: A study of the probability of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(4), 530–543.
- Drucker, H., C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik (1994). Boosting and other ensemble methods. *Neural Computation* 6(6), 1289–1301.
- Drucker, H., R. Schapire, and P. Simard (1993). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence* 7(4), 705–719.
- Duda, R. O. and P. E. Hart (1973). *Pattern Recognition and Scene Analysis*. New York: John Wiley & Sons Inc.
- Duin, R. P. W. (1996). A note on comparing classifiers. *Pattern Recognition Letters* 17, 529–536.
- Dunn, C. E. and P. S. P. Wang (1992). Character segmentation techniques for handwritten text – a survey. In *Proceedings of the 11th International Conference on Pattern Recognition*, Volume 2, Hague, pp. 577–580. IAPR.
- ERA (1957). An electronic reading automaton. *Electronic engineering* 29(4), 189–190.
- Feller, W. (1971). *An Introduction to Probability Theory and Its Applications, Volume II*. John Wiley & Sons Inc.
- Fix, E. and J. L. Hodges (1951). Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas.
- Flick, T. E., L. K. Jones, R. G. Priest, and C. Herman (1990). Pattern classification using projection pursuit. *Pattern Recognition* 23(12), 1367–1376.
- Flusser, J. and T. Suk (1993). Pattern recognition by affine moment invariants. *Pattern Recognition* 26(1), 167–174.

- Flusser, J. and T. Suk (1994). Affine moment invariants: a new tool for character recognition. *Pattern Recognition Letters* 15, 433–436.
- Foley, D. and J. Sammon (1975). An optimal set of discriminant vectors. *IEEE Transactions on Computers* C-24(3), 281–289.
- Franke, J. and E. Mandler (1992). A comparison of two approaches for combining the votes of cooperating classifiers. In *Proceedings of the 11th International Conference on Pattern Recognition*, Volume II, Hague, pp. 611–614. IAPR.
- Freedman, M. D. (1974). Optical character recognition. *IEEE Spectrum* 11(3), 44–52.
- Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association* 84(405), 165–175.
- Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics* 19, 1–141. with discussion.
- Friedman, J. H. and W. Stuetzle (1981). Projection pursuit regression. *Journal of the American Statistical Association* 76(376), 817–823.
- Fu, K. S. (1982). *Syntactic pattern recognition and applications*. Prentice-Hall.
- Fu, K. S. and A. Rosenfeld (1984). Pattern recognition and computer vision. *Computer* 17(10), 274–282.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition* (2nd ed.). Academic Press.
- Fukunaga, K. and R. R. Hayes (1989). The reduced Parzen classifier. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-11(4), 423–425.
- Fukunaga, K. and W. L. Koontz (1970). Application of the Karhunen-Loève expansion to feature selection and ordering. *IEEE Transactions on Computers* C-19(4), 311–318.
- Fukunaga, K. and J. M. Mantock (1984). Nonparametric data reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(1), 115–118.
- Gader, P., B. Forester, M. Ganzberger, A. Gillies, B. Mitchell, M. Whalen, and T. Yocum (1991). Recognition of handwritten digits using template and model matching. *Pattern Recognition* 24(5), 421–431.
- Garris, M. D., J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson (1994). NIST form-based handprint recognition system. Technical Report NISTIR 5469, National Institute of Standards and Technology.
- Geist, J., R. A. Wilkinson, S. Janet, P. J. Grother, B. Hammond, N. W. Larsen, R. M. Klear, M. J. Matsko, C. J. C. Burges, R. Creecy, J. J. Hull, T. P. Vogl, and C. L. Wilson (1992). The second census optical character recognition systems conference. Technical Report NISTIR 5452, National Institute of Standards and Technology.

- Gersho, A. and R. M. Gray (1992). *Vector quantization and signal compression*. Kluwer Academic Publishers.
- Gilloux, M. (1993). Research into the new generation of character and mailing address recognition systems at the French post office research center. *Pattern Recognition Letters* 14(4), 267–276.
- Glauber, M. H. (1956). Character recognition for business machines. *Electronics* 29(2), 132–136.
- Golub, G. H. and C. F. van Loan (1989). *Matrix computations* (2nd ed.). Johns Hopkins University Press.
- Gonzalez, R. and M. Thomason (1978). *Syntactic Pattern Recognition*. Addison-Wesley.
- Gonzalez, R. C. and R. E. Woods (1992). *Digital image processing*. Addison-Wesley.
- Govindan, V. K. and A. P. Shivaprasad (1990). Character recognition—a review. *Pattern Recognition* 23(7), 671–683.
- Grabec, I. (1990). Self-organization of neurons described by the maximum-entropy principle. *Biological Cybernetics* 63, 403–409.
- Granlund, G. H. (1972). Fourier preprocessing for hand print character recognition. *IEEE Transactions on Computers* 21(2), 195–201.
- Guyon, I., L. Schomaker, R. Plamondon, M. Liberman, and S. Janet (1994). UNIPEN project of on-line data exchange and recognition benchmarks. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 29–33. IAPR.
- Hand, D. J. (1982). *Kernel Discriminant Analysis*. Cichester: Research Studies Press.
- Hansen, L. K. and P. Salamon (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10), 993–1001.
- Haralick, R. M. and L. G. Shapiro (1992). *Computer and Robot Vision*. Addison-Wesley.
- Harmon, L. D. (1972). Automatic recognition of print and script. *Proceedings of the IEEE* 60(10), 1165–1176.
- Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* 14(3), 515–516.
- Hartigan, J. (1975). *Clustering Algorithms*. New York: John Wiley & Sons Inc.
- Hastie, T., P. Y. Simard, and E. Säcker (1995). Learning prototype models for tangent distance. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems* 7, Cambridge, MA, pp. 999–1006. MIT Press.
- Hastie, T. and W. Stuetzle (1989). Principal curves. *Journal of the American Statistical Association* 84(406), 502–516.

- Hastie, T. and R. Tibshirani (1996). Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society, Series B* 58(1), 155–176.
- Hastie, T., R. Tibshirani, and A. Buja (1994). Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association* 89, 1255–1270.
- Hastie, T. J. and C. Loader (1993). Local regression: Automatic kernel carpentry. *Statistical Science* 8, 120–143.
- Hastie, T. J. and R. Tibshirani (1994). Handwritten digit recognition via deformable prototypes. Technical report, AT&T Bell Laboratories. Available as <http://netlib.att.com/netlib/att/stat/doc/93.22.ps.Z>.
- Hastie, T. J. and R. J. Tibshirani (1990). *Generalized Additive Models*. Chapman & Hall.
- Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing Company, Inc.
- Herman, G. T. and H. K. Liu (1978). Dynamic boundary surface detection. *Computer Graphics and Image Processing* 7(1), 130–138.
- Heutte, L., J. V. Moreau, T. Paquet, Y. Lecourtier, and C. Olivier (1996). Combining structural and statistical features for the recognition of handwritten characters. In *Proceedings of 13th International Conference on Pattern Recognition*, Volume II, Vienna, pp. 210–214. IAPR.
- Highleyman, W. H. (1962). Linear decision functions with application to pattern recognition. *Proc. IRE* 50, 1501–1514.
- Hinton, G. E., M. Revow, and P. Dayan (1995). Recognizing handwritten digits using mixtures of linear models. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, Cambridge, MA, pp. 1015–1022. MIT Press.
- Ho, T. K., J. J. Hull, and S. N. Srihari (1992). A regression approach to combination of decisions by multiple character recognition algorithms. In D. P. D’Amato, W.-E. Blanz, B. E. Dom, and S. N. Srihari (Eds.), *Proceedings of SPIE Conference on Machine Vision Applications in Character Recognition and Industrial Inspection*, Number 1661 in SPIE, pp. 137–145.
- Ho, T. K., J. J. Hull, and S. N. Srihari (1994). Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(1), 66–75.
- Holmström, L. and A. Hämmäläinen (1993). The self-organizing reduced kernel density estimator. In *Proceedings of the 1993 IEEE International Conference on Neural Networks, San Francisco, California, March 28 - April 1*, Volume 1, pp. 417–421.

- Holmström, L., P. Koistinen, J. Laaksonen, and E. Oja (1996a). Comparison of neural and statistical classifiers – theory and practice. Technical Report A13, Rolf Nevanlinna Institute, Helsinki.
- Holmström, L., P. Koistinen, J. Laaksonen, and E. Oja (1996b). Neural network and statistical perspectives of classification. In *Proceedings of 13th International Conference on Pattern Recognition*, Volume IV, Vienna, pp. 286–290. IAPR.
- Holmström, L., P. Koistinen, J. Laaksonen, and E. Oja (1997). Neural and statistical classifiers – taxonomy and two case studies. *IEEE Transactions on Neural Networks* 8(1), 5–17.
- Holmström, L., S. Sain, and H. Miettinen (1995). A new multivariate technique for top quark search. *Computer Physics Communications* 88, 195–210.
- Hong, Z.-Q. (1991). Algebraic feature extraction of image for recognition. *Pattern Recognition* 24(3), 211–219.
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* 24, 498–520.
- Hsia, T. C. (1981). A note on invariant moments in image processing. *IEEE Transactions on Systems, Man, and Cybernetics* 11(12), 831–834.
- Hu, M.-K. (1961). Pattern recognition by moment invariants. *Proceedings of the IRE* 49, 1428.
- Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory* 8(2), 179–187.
- Hu, T. C. (1982). *Combinatorial algorithms*. Reading, MA: Addison-Wesley.
- Huang, J. S. and K. Chuang (1986). Heuristic approach to handwritten numeral recognition. *Pattern Recognition* 19(1), 15–19.
- Huang, Y. S., K. Liu, and C. Y. Suen (1995). The combination of multiple classifiers by a neural network approach. *International Journal of Pattern Recognition and Artificial Intelligence* 9(3), 579–597.
- Hummel, R. A. and S. W. Zucker (1983). On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(3), 267–287.
- Idan, Y. and J.-M. Auger (1992). Pattern recognition by cooperating neural networks. In S.-S. Chen (Ed.), *Proceedings of SPIE Conference on Neural and Stochastic Methods in Image and Signal Processing*, Number 1766 in SPIE, pp. 437–443.
- Iijima, T., H. Genchi, and K. Mori (1973). A theory of character recognition by pattern matching method. In *Proceedings of the 1st International Joint Conference on Pattern Recognition*, Washington, DC, pp. 50–56.

- Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation* 7(5), 867–888.
- Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice-Hall.
- Jain, A. K. and B. Chandrasekaran (1982). Dimensionality and sample size considerations in pattern recognition practice. In P. R. Krishnaiah and L. N. Kanal (Eds.), *Handbook of statistics*, Volume 2, pp. 835–855. North-Holland.
- Jain, A. K. and J. Mao (1994). Neural networks and pattern recognition. In J. M. Zurada, R. J. Marks II, and C. J. Robinson (Eds.), *Computational Intelligence Imitating Life*, Chapter IV-1, pp. 194–212. IEEE Press.
- Jang, B. K. and R. T. Chin (1992). One-pass parallel thinning: Analysis, properties, and quantitative evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(11), 1129–1140.
- Jordan, M. I. and R. A. Jacobs (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6(2), 181–214.
- Jutten, C. and J. Héroult (1991). Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture. *Signal Processing* 24(1), 1–10.
- Kageyu, S., N. Ohnishi, and N. Sugie (1991). Augmented multi-layer perceptron for rotation-and-scale invariant hand-written numeral recognition. In *Proceedings of 1991 International Joint Conference on Neural Networks*, Volume 1, Singapore, pp. 54–59. IEEE, INNS.
- Kahan, S., T. Pavlidis, and H. S. Baird (1987). On the recognition of printed characters of any font and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9(2), 274–288.
- Kanal, L. N. (Ed.) (1981). *Progress in pattern recognition 1*. North-Holland.
- Kanal, L. N. (Ed.) (1985). *Progress in pattern recognition 2*. North-Holland.
- Karhunen, J. and E. Oja (1980). Some comments on the subspace methods of classification. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pp. 1191–1194.
- Karhunen, J., E. Oja, L. Wang, R. Vigário, and J. Joutsensalo (1997). A class of neural networks for independent component analysis. *IEEE Transactions on Neural Networks* 8. To appear.
- Khotanzad, A. and Y. H. Hong (1990a). Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(5), 489–497.
- Khotanzad, A. and Y. H. Hong (1990b). Rotation invariant image recognition using features selected via a systematic method. *Pattern Recognition* 23(10), 1089–1101.

- Kimura, F. and M. Shridhar (1991). Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition* 24(10), 969–983.
- Kittler, J. (1978). The subspace approach to pattern recognition. In R. Trappl, G. J. Klir, and L. Ricciardi (Eds.), *Progress in cybernetics and systems research, vol. III*, pp. 92–97. Hemisphere Publishing.
- Kittler, J., K. S. Fu, and L. F. Pau (Eds.) (1982). *Pattern Recognition Theory and Applications; Proceedings of the NATO Advanced Study Institute*. D. Reidel Publishing Company.
- Kohonen, T. (1988). The 'neural' phonetic typewriter. *Computer* 21(3), 11–22.
- Kohonen, T. (1995). *Self-Organizing Maps*. Springer Series in Information Sciences 30. Springer-Verlag.
- Kohonen, T. (1996). Private communication.
- Kohonen, T., G. Németh, K.-J. Bry, M. Jalanko, and H. Riittinen (1978). Classification of phonemes by learning subspaces. Technical Report TKK-F-A348, Helsinki University of Technology, Espoo, Finland.
- Kohonen, T., G. Németh, K.-J. Bry, M. Jalanko, and H. Riittinen (1979). Spectral classification of phonemes by learning subspaces. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Washington, DC, pp. 97–100. IEEE.
- Kohonen, T., H. Riittinen, M. Jalanko, E. Reuhkala, and S. Haltsonen (1980). A thousand-word recognition system based on the Learning Subspace Method and Redundant Hash Addressing. In *Proceedings of the 5th International Conference on Pattern Recognition*, Volume 1, Miami Beach, FL, pp. 158–165. IAPR.
- Koistinen, P. and L. Holmström (1992). Kernel regression and backpropagation training with noise. In J. E. Moody, S. J. Hanson, and R. P. Lippman (Eds.), *Advances in Neural Information Processing Systems 4*, San Mateo, CA, pp. 1033–1039. Morgan Kaufmann Publishers.
- Kramer, H. and M. Mathews (1956). A linear coding for transmitting a set of correlated signals. *IRE Transactions on Information Theory IT-2*, 41–46.
- Krogh, A. and J. Vedelsby (1995). Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, Cambridge, MA, pp. 231–238. MIT Press.
- Kuhl, F. P. and C. R. Giardina (1982). Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing* 18(3), 236–258.
- Kulikowski, C. A. and S. Watanabe (1970). Multiclass subspace methods in pattern recognition. In *Proceedings of the National Electronics Conference*, Chicago, IL.

- Kuusela, M. and E. Oja (1982). The Averaged Learning Subspace Method for spectral pattern recognition. In *Proceedings of the 6th International Conference on Pattern Recognition*, München, pp. 134–137. IEEE.
- Laaksonen, J. and E. Oja (1996a). Classification with learning k -nearest neighbors. In *Proceedings of the International Conference on Neural Networks*, Volume 3, Washington D.C., pp. 1480–1483.
- Laaksonen, J. and E. Oja (1996b). Subspace dimension selection and averaged learning subspace method in handwritten digit classification. In *Proceedings of the International Conference on Artificial Neural Networks*, Bochum, Germany, pp. 227–232.
- Lam, L., S.-W. Lee, and C. Y. Suen (1992). Thinning methodologies – a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(9), 869–885.
- Lam, L. and C. Y. Suen (1988). Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition* 21(1), 19–31.
- Lam, L. and C. Y. Suen (1994). A theoretical analysis of the application of majority voting to pattern recognition. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 418–420. IAPR.
- Lam, L. and C. Y. Suen (1995). Optimal combinations of pattern classifiers. *Pattern Recognition Letters* 16, 945–954.
- Lam, S. W., A. C. Girardin, and S. N. Srihari (1992). Gray scale character recognition using boundary features. In D. P. D’Amato, W.-E. Banz, B. E. Dom, and S. N. Srihari (Eds.), *Proceedings of SPIE Conference on Machine Vision Applications in Character Recognition and Industrial Inspection*, Number 1661 in SPIE, pp. 98–105.
- Lampinen, J. and E. Oja (1995). Distortion tolerant pattern recognition based on self-organizing feature extraction. *IEEE Transactions on Neural Networks* 6(3), 539–547.
- LeBlanc, M. and R. Tibshirani (1994). Adaptive principal surfaces. *Journal of the American Statistical Association* 89(425), 53–64.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1, 541–551.
- Lee, H.-J. and B. Chen (1992). Recognition of handwritten Chinese characters via short line segments. *Pattern Recognition* 25(5), 543–552.
- Lee, S.-W. and J.-S. Park (1994). Nonlinear shape normalization methods for the recognition of large-set handwritten characters. *Pattern Recognition* 27(7), 895–902.
- Li, Y. (1992). Reforming the theory of invariant moments for pattern recognition. *Pattern Recognition* 25(7), 723–730.

- Lindgren, N. (1965). Machine recognition of human language: Part III—cursive script recognition. *IEEE Spectrum* 2(5), 104–116.
- Liu, K., Y.-Q. Cheng, and J.-Y. Yang (1993). Algebraic feature extraction for image recognition based on optimal discriminant criterion. *Pattern Recognition* 26(6), 903–911.
- Liu, K., Y.-S. Huang, C. Y. Suen, J.-Y. Yang, L.-J. Liu, and Y.-J. Liu (1994). Discriminance performance of the algebraic features of handwritten character images. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 426–428. IAPR.
- Lu, S. W., Y. Ren, and C. Y. Suen (1991). Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition* 24(7), 617–632.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman (Eds.), *Proc. Fifth Berkeley Symp. Math. Stat. and Prob.*, pp. 281–297. Berkeley, CA: U.C. Berkeley Press.
- Mai, T. A. and C. Y. Suen (1990). A generalized knowledge-based system for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Systems, Man, and Cybernetics* 20(4), 835–848.
- Maitra, S. (1979). Moment invariants. *Proceedings of the IEEE* 67(4), 697–699.
- Mantas, J. (1986). An overview of character recognition methodologies. *Pattern Recognition* 19(6), 425–430.
- McCulloch, W. S. and W. Pitts (1943). A logical calculus of the idea immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133.
- McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons Inc.
- Mendel, J. M. (1995). *Lessons in Estimation Theory for Signal Processing, Communications, and Control*. Prentice-Hall.
- Michie, D., D. J. Spiegelhalter, and C. C. Taylor (Eds.) (1994). *Machine learning, neural and statistical classification*. Ellis Horwood Limited.
- Moody, J. and C. J. Darken (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation* 1, 281–294.
- Morgan, J. N. and J. A. Sonquist (1963). Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association* 58, 415–434.
- Mori, S., C. Y. Suen, and K. Yamamoto (1992). Historical review of OCR research and development. *Proceedings of the IEEE* 80(7), 1029–1058.
- Mulier, F. and V. Cherkassky (1995). Self-organization as an iterative kernel smoothing process. *Neural Computation* 7(6), 1165–1177.

- Nadal, C., R. Legault, and C. Y. Suen (1990). Complementary algorithms for the recognition of totally unconstrained handwritten numerals. In *Proceedings of the 10th International Conference on Pattern Recognition*, Atlantic City, NJ, pp. 443–449. IAPR.
- Nadaraya, E. (1964). On estimating regression. *Theory of Probability and its Applications* 9, 141–142.
- Noguchi, Y. (1976). A construction method of category feature subspaces using orthogonal projection operators. *Bulletin of Electrotechnical laboratory* 40(8), 641–659.
- Noguchi, Y. (1978). Subspace method and projection operators. In *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, pp. 571–587.
- Oja, E. (1983). *Subspace Methods of Pattern Recognition*. Letchworth, England: Research Studies Press Ltd.
- Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems* 1(1), 61–68.
- Oja, E. (1995). The nonlinear PCA learning rule and signal separation – mathematical analysis. Technical Report A26, Helsinki University of Technology, Laboratory of Computer and Information Science.
- Oja, E. and J. Karhunen (1985). On stochastic approximation of eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications* 106, 69–84.
- Oja, E. and J. Parkkinen (1984). On subspace clustering. In *Proceedings of the 7th International Conference on Pattern Recognition*, Volume 2, Montreal, pp. 692–695. IAPR.
- Pao, Y.-H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Publishing Company.
- Parker, J. R. (1994). Vector templates and handprinted digit recognition. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 457–459. IAPR.
- Parkkinen, J. (1989). *Subspace methods in two machine vision problems*. Ph. D. thesis, University of Kuopio.
- Pavel, M. (1993). *Fundamentals of pattern recognition* (2nd ed.). Marcel Dekker, Inc.
- Pavlidis, T. and F. Ali (1975). Computer recognition of handwritten numerals by polygonal approximations. *IEEE Transactions on Systems, Man, and Cybernetics* 5(6), 610–614.
- Perrone, M. P. (1994). Pulling it all together: Methods for combining neural networks. In J. D. Cowan, G. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems* 6, San Francisco, CA, pp. 1188–1189. Morgan Kaufmann Publishers.

- Perrone, M. P. and L. N. Cooper (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone (Ed.), *Artificial Neural Networks for Speech and Vision*, pp. 126–142. Chapman & Hall.
- Pratt, W. K. (1991). *Digital image processing*. New York: John Wiley & Sons Inc.
- Prechelt, L. (1996). A quantitative study of experimental evaluations of neural network learning algorithms: Current research practice. *Neural Networks* 9(3), 457–462.
- Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1988). *Numerical Recipes in C*. Cambridge University Press.
- Priebe, C. E. and D. J. Marchette (1991). Adaptive mixtures: Recursive nonparametric pattern recognition. *Pattern Recognition* 24(12), 1197–1209.
- Priebe, C. E. and D. J. Marchette (1993). Adaptive mixture density estimation. *Pattern Recognition* 26(5), 771–785.
- Rabinow, J. C. (1969). Whither OCR. *Datamation* 15(7), 38–42.
- Reddi, S. S. (1981). Radial and angular moment invariants for image identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3(2), 240–242.
- Redner, R. A. and H. F. Walker (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review* 26(2).
- Reiss, T. H. (1991). The revised fundamental theorem of moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(8), 830–834.
- Revow, M., C. K. I. Williams, and G. E. Hinton (1996). Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(6), 592–606.
- Richard, M. D. and R. P. Lippman (1991). Neural network classifiers estimate bayesian *a posteriori* probabilities. *Neural Computation* 3(4), 461–483.
- Riittinen, H. (1986). *Recognition of phonemes in a speech recognition system using learning projective methods*. Ph. D. thesis, Helsinki University of Technology.
- Riittinen, H., S. Haltsonen, E. Reuhkala, and M. Jalanko (1980). Experiments on an isolated-word recognition system for multiple speakers. Technical Report TKK-F-A433, Helsinki University of Technology, Espoo, Finland.
- Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.
- Rissanen, J. (1978). Modelling by shortest data description. *Automatica* 14, 465–471.
- Ritter, H., T. Martinetz, and K. Schulten (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, Massachusetts: Addison-Wesley Publishing Company.

- Rocha, J. and T. Pavlidis (1994). A shape analysis model with applications to a character recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(4), 393–404.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in brain. *Psychological Review* 65, 386–408.
- Rosenblatt, F. (1961). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC.: Spartan Books.
- Rosenfeld, A. and A. C. Kak (1982). *Digital Picture Processing* (2nd ed.), Volume 1-2. Academic Press.
- Schalkoff, R. J. (1989). *Digital Image Processing and Computer Vision*. John Wiley & Sons Inc.
- Schalkoff, R. J. (1992). *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons Inc.
- Schmidt, W. F., D. F. Levelt, and R. P. W. Duin (1994). An experimental comparison of neural classifiers with 'traditional' classifiers. In E. S. Gelsema and L. S. Kanal (Eds.), *Pattern Recognition in Practice IV*, Volume 16 of *Machine Intelligence and Pattern Recognition*. Elsevier Science Publishers.
- Schürmann, J., N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberländer (1992). Document analysis – from pixels to contents. *Proceedings of the IEEE* 80(7), 1101–1119.
- Schwartz, E. L. (1977). Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics* 25(4), 181–194.
- Schwenk, H. and M. Milgram (1995). Transformation invariant autoassociation with application to handwritten character recognition. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, Cambridge, MA, pp. 991–998. MIT Press.
- Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons Inc.
- Sejnowski, T. J. and C. R. Rosenberg (1987). Parallel networks that learn to pronounce English text. *Complex Systems* 1(1), 145–168.
- Sekita, I., K. Toraichi, R. Mori, K. Yamamoto, and H. Yamada (1988). Feature extraction of handwritten Japanese characters by spline functions for relaxation matching. *Pattern Recognition* 21(1), 9–17.
- Serra, J. (1982). *Image analysis and mathematical morphology*. Academic Press.
- Shridhar, M. and A. Badreldin (1984). High accuracy character recognition algorithm using Fourier and topological descriptors. *Pattern Recognition* 17(5), 515–524.

- Shridhar, M. and A. Badreldin (1985). A high-accuracy syntactic recognition algorithm for handwritten numerals. *IEEE Transactions on Systems, Man, and Cybernetics* 15(1), 152–158.
- Shridhar, M. and A. Badreldin (1986). Recognition of isolated and simply connected handwritten numerals. *Pattern Recognition* 19(1), 1–12.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.
- Silverman, B. W. and M. C. Jones (1989). E. Fix and J.L. Hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation—commentary on Fix and Hodges (1951). *International Statistical Review* 57(3), 233–247.
- Simard, P. Y., Y. LeCun, and J. S. Denker (1993). Efficient pattern recognition using a new transformation distance. In *Neural Information Processing Systems 5*, pp. 50–58. Morgan Kaufmann Publishers.
- Simard, P. Y., Y. LeCun, and J. S. Denker (1994). Memory-based character recognition using a transformation invariant metric. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 262–267. IAPR.
- Sklansky, J. and G. N. Wassel (1981). *Pattern Classifiers and Trainable Machine*. Springer-Verlag.
- Śmieja, F. (1994). The Pandemonium system of reflective agents. Technical Report 1994/2, German National Research Center for Computer Science (GMD). Available at <<http://borneo.gmd.de/AS/janus/publi/publi.html>>.
- Smyth, P. and J. Mellstrom (1992). Fault diagnosis of antenna pointing systems using hybrid neural network and signal processing models. In J. Moody, S. Hanson, and R. Lippmann (Eds.), *Advances in Neural Information Processing Systems 4*, pp. 667–674. Morgan Kaufmann Publishers.
- Sollich, P. and A. Krogh (1995). Learning with ensembles: How over-fitting can be useful. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, Cambridge, MA. MIT Press.
- Sorenson, H. W. (1980). *Parameter estimation - principles and problems*. New York: Marcel Dekker.
- Specht, D. F. (1990). Probabilistic Neural Networks. *Neural Networks* 3, 109–118.
- Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks* 2(6), 568–576.
- Srihari, S. N. (1993). Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern Recognition Letters* 14(4), 291–302.

- Stone, C. J. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B* 36, 111–147.
- Strang, G. and T. Nguyen (1996). *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press.
- Suen, C. Y., M. Berthold, and S. Mori (1980). Automatic recognition of handprinted characters—the state of the art. *Proceedings of the IEEE* 68(4), 469–487.
- Suen, C. Y., R. Legault, C. Nadal, M. Cheriet, and L. Lam (1993). Building a new generation of handwriting recognition systems. *Pattern Recognition Letters* 14(4), 303–315.
- Suen, C. Y., C. Nadal, R. Legault, T. A. Mai, and L. Lam (1992). Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE* 80(7), 1162–1180.
- Taha, H. A. (1982). *Operations Research* (3rd ed.). New York: Macmillan Publishing.
- Tang, Y. Y., B. F. Li, H. Ma, J. Liu, C. H. Leung, and C. Y. Suen (1996). A novel approach to optical character recognition based on ring-projection-wavelet-fractal signatures. In *Proceedings of 13th International Conference on Pattern Recognition*, Volume II, Vienna, pp. 325–329. IAPR.
- Tappert, C. C., C. Y. Suen, and T. Wakahara (1990). The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(8), 787–808.
- Taxt, T. and K. W. Bjerde (1994). Classification of handwritten vector symbols using elliptic Fourier descriptors. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 123–128. IAPR.
- Taxt, T., J. B. Ólafsdóttir, and M. Dæhlen (1990). Recognition of handwritten symbols. *Pattern Recognition* 23(11), 1156–1166.
- Teague, M. R. (1980). Image analysis via the general theory of moments. *Journal of the Optical Society of America* 70(8), 920–930.
- Therrien, C. W. (1975). Eigenvalue properties of projection operators and their application to the subspace method of feature extraction. *IEEE Transactions on Computers* C-24(9), 944–948.
- Therrien, C. W. (1989). *Decision, Estimation, and Classification*. John Wiley and Sons.
- Tou, J. T. and R. C. Gonzalez (1974). *Pattern Recognition Principles*. Massachusetts: Addison-Wesley Publishing Company.
- Toussaint, G. T., E. Backer, P. Devijver, K. Fukunaga, and J. Kittler (1982). Summary of panel discussion on decision theoretic methods. In J. Kittler, K. S. Fu, and L. F. Pau (Eds.), *Pattern Recognition Theory and Applications; Proceedings of the NATO Advanced Study Institute*, pp. 569–572. D. Reidel Publishing Company.

- Tråvén, H. G. C. (1991). A neural network approach to statistical pattern classification by “semiparametric” estimation of probability density functions. *IEEE Transactions on Neural Networks* 2(3), 366–377.
- Tresp, V. and M. Taniguchi (1995). Combining estimators using non-constant weighting functions. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, Cambridge, MA, pp. 419–426. MIT Press.
- Trier, Ø. D., A. K. Jain, and T. Taxt (1996). Feature extraction methods for character recognition—a survey. *Pattern Recognition* 29(4), 641–662.
- Tubbs, J. D. (1989). A note on binary template matching. *Pattern Recognition* 22(4), 359–365.
- Tutz, G. E. (1986). An alternative choice of smoothing for kernel-based density estimates in discrete discriminant analysis. *Biometrika* 73, 405–411.
- Unser, M. (1984). On the approximation of the discrete Karhunen-Loeve transform for stationary processes. *Signal Processing* 7(3), 231–249.
- Veelaert, P. (1994). Arrays of low-level inequality based feature detecting cells. In *Proceedings of 12th International Conference on Pattern Recognition*, Volume II, Jerusalem, pp. 500–502. IAPR.
- Venables, W. N. and B. D. Ripley (1994). *Modern Applied Statistics with S-Plus*. New York: Springer-Verlag.
- Vetterli, M. and J. Kovačević (1995). *Wavelets and Subband Coding*. Prentice-Hall.
- Wakahara, T. (1993). Towards robust handwritten character recognition. *Pattern Recognition Letters* 14(4), 345–354.
- Wakahara, T. (1994). Shape matching using LAT and its application to handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(6), 618–629.
- Wallace, G. K. (1991). The JPEG still picture compression standard. *Communications of the ACM* 34(4), 31–44.
- Wand, M. P. and M. C. Jones (1995). *Kernel Smoothing*. Chapman & Hall.
- Watanabe, S. (1965). Karhunen-Loeve expansion and factor analysis. In *Transactions of the 4th Prague Conference on Information Theory, Statistical Decision Functions, and Random Processes*, Prague, pp. 635–660.
- Watanabe, S., P. F. Lambert, C. A. Kulikowski, J. L. Buxton, and R. Walker (1967). Evaluation and selection of variables in pattern recognition. In J. Tou (Ed.), *Computer and Information Sciences II*. New York: Academic Press.

- Watanabe, S. and N. Pakvasa (1973). Subspace method in pattern recognition. In *Proceedings of the 1st International Joint Conference on Pattern Recognition*, Washington, D.C.
- Watson, G. (1964). Smooth regression analysis. *Sankhyā Series A* 26, 359–372.
- Wechsler, H. and G. L. Zimmerman (1988). 2-D invariant object recognition using distributed associative memory. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(6), 811–821.
- Weideman, W. E., M. T. Manry, H.-C. Yau, and W. Gong (1995). Comparisons of a neural network and a nearest-neighbor classifier via the numeric handprint recognition problem. *IEEE Transactions on Neural Networks* 6(6), 1524–1530.
- Weiman, C. F. R. and G. Chaikin (1979). Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing* 11(3), 197–226.
- White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation* 1, 425–464.
- Wilkinson, R. A., J. Geist, S. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. W. Larsen, T. P. Vogl, and C. L. Wilson (1991). The first census optical character recognition systems conference. Technical Report NISTIR 4912, National Institute of Standards and Technology.
- Williams, C. K. I. (1994). *Combining deformable models and neural networks for hand-printed digit recognition*. Ph. D. thesis, University of Toronto.
- Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics* 2, 408–420.
- Wold, S. (1976). Pattern recognition by means of disjoint principal component models. *Pattern Recognition* 8, 127–139.
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks* 5, 241–259.
- Wu, L. and F. Fallside (1991). On the design of connectionist vector quantizers. *Computer Speech and Language* 5, 207–229.
- Wyatt, Jr, J. L. and I. M. Elfadel (1995). Time-domain solutions of Oja’s equations. *Neural Computation* 7, 915–922.
- Xu, L. and M. I. Jordan (1993). EM learning on a generalized finite mixture model for combining multiple classifiers. In *Proceedings of the World Congress on Neural Networks*, Volume IV, pp. 227–230.
- Xu, L., M. I. Jordan, and G. E. Hinton (1995). An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems* 7, Cambridge, MA, pp. 633–640. MIT Press.

Xu, L., A. Krzyżak, and C. Y. Suen (1991). Associative switch for combining multiple classifiers. In *Proceedings of 1991 International Joint Conference on Neural Networks*, Volume 1, Seattle, WA, pp. 43–48. IEEE, INNS.

Xu, L., A. Krzyżak, and C. Y. Suen (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics* 22(3), 418–435.

Yan, W.-Y., U. Helmke, and J. B. Moore (1994). Global analysis of Oja’s flow for neural networks. *IEEE Transactions on Neural Networks* 5(5), 674–683.

Young, T. Y. and T. W. Calvert (1974). *Classification, Estimation and Pattern Recognition*. New York: Elsevier Science Publishers.



This book was originally typeset with $\text{\LaTeX 2}_{\epsilon}$ and later PDFed by using pdf $\text{\LaTeX}$.