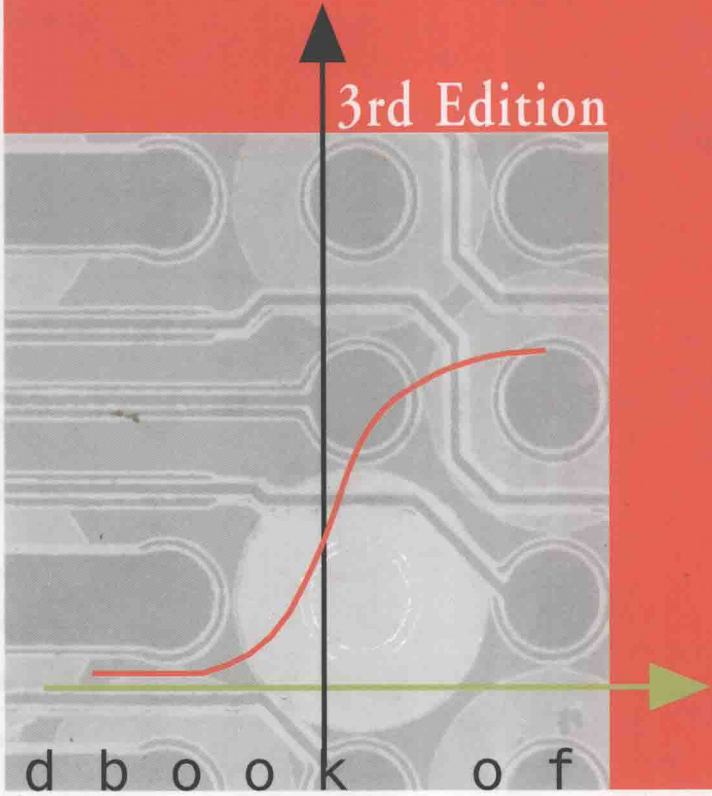
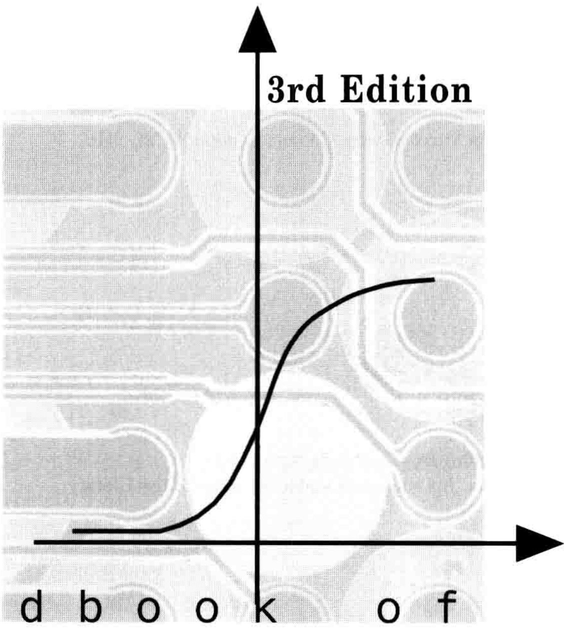


3rd Edition

Handbook of Pattern Recognition and Computer Vision

editors C H Chen & P S P Wang



Handbook of Pattern Recognition and Computer Vision

editors

C H Chen
University of Massachusetts Dartmouth, USA

P S P Wang
Northeastern University, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN 981-256-105-6

Printed in Singapore by Mainland Press

Preface to the Third Edition

Dedicated to the memory of the late Professor King Sun Fu (1930-1985), the handbook series, with first edition (1993), second edition (1999) and third edition (2005), provides a comprehensive, concise and balanced coverage of the progress and achievements in the field of pattern recognition and computer vision in the last twenty years. This is a highly dynamic field which has been expanding greatly over the last thirty years. No handbook can cover the essence of all aspects of the field and we have not attempted to do that. The carefully selected 33 chapters in the current edition were written by leaders in the field and we believe that the book and its sister volumes, the first and second editions, will provide the growing pattern recognition and computer vision community a set of valuable resource books that can last for a long time. Each chapter will speak for itself the importance of the subject area covered.

The book continues to contain five parts. Part 1 is on the basic methods of pattern recognition. Though there are only five chapters, the readers may find other coverage of basic methods in the first and second editions. Part 2 is on basic methods in computer vision. Again readers may find that Part 2 complements well what were offered in the first and second editions. Part 3 on recognition applications continues to emphasize on character recognition and document processing. It also presents new applications in digital mammograms, remote sensing images and functional magnetic resonance imaging data. Currently one intensively explored area of pattern recognition applications is the personal identification problem, also called biometrics, though the problem has been around for a number of years. Part 4 is especially devoted to this topic area. Indeed chapters in both Part 3 and Part 4 represent the growing importance of applications in pattern recognition. In fact Prof. Fu had envisioned the growth of pattern recognition applications in the early 60's He and his group at Purdue had worked on the character recognition, speech recognition, fingerprint recognition, seismic pattern recognition, biomedical and remote sensing recognition problems, etc. Part 5 on system and technology presents other important aspects of pattern recognition and computer vision.

Our sincere thanks go to all contributors of this volume for their outstanding technical contributions. We would like to mention specially Dr. Quang-Tuan Luong, Dr. Giovanni Garibotto and Prof. Ching Y. Suen for their original contributions to all three volumes. Other authors who have contributed to all three volumes are: Prof. Thomas S. Huang, Prof. J.K. Aggarwal, Prof. Yun Y. Tang, Prof. C.C. Li, Prof. R. Chellappa and Prof. P.S.P. Wang. We are pleased to mention that Prof. Thomas Huang and Prof. Jake Aggarwal are the recipients respectively in 2002 and 2004, of the prestigious K.S. Fu Prize sponsored by the International Association of Pattern Recognition (IAPR). Among Prof. Fu's Ph.D. graduates at Purdue who have contributed to the handbook series are: C.H. Chen (1965), M.H. Loew (1972), S.M. Hsu (1975), S.Y. Lu (1977), K.Y. Huang (1983) and H.D. Cheng (1985). Finally we would like to pay tribute to the late Prof. Azirel Rosenfeld (1931-2004) who, as one IAPR member put it, is a true scientist and a great giant in the field. He was awarded the K.S. Fu Prize by IAPR in 1988. Readers are reminded to read Prof. Rosenfeld's inspirational article on "Vision – Some Speculations" that appeared as Foreword of the second edition of the handbook series. Prof. Rosenfeld's profound influence in the field will be felt in the many years to come.

The camera ready manuscript production requires certain amount of additional efforts, as compared to typeset printing, on the part of editors and authors. We like to thank all contributors for their patience in making the necessary revisions to comply with the format requirements during this long process of manuscript preparation. Our special thanks go to Steven Patt, in-house editor of World Scientific Publishing, for his efficient effort to make a timely publication of the book possible.

September, 2004 The Editors

Contents

Part 1

Basic Methods in Pattern Recognition

CHAPTER 1.1

STATISTICAL PATTERN RECOGNITION

R.P.W. Duin, D.M.J. Tax

Information and Communication Theory Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
P.O.Box 5031, 2600 GA, Delft, The Netherlands
E-mail: {R.P.W.Duin,D.M.J.Tax}@ewi.tudelft.nl

A review is given of the area of statistical pattern recognition: the representation of objects and the design and evaluation of trainable systems for generalization. Traditional as well as more recently studied procedures are reviewed like the classical Bayes classifiers, neural networks, support vector machines, one-class classifiers and combining classifiers. Further we introduce methods for feature reduction and error evaluation. New developments in statistical pattern recognition are briefly discussed.

1. Introduction

Statistical pattern recognition is the research area that studies statistical tools for the generalization of sets of real world objects or phenomena. It thereby aims to find procedures that answer questions like: does this new object fit into the pattern of a given set of objects, or: to which of the patterns defined in a given set does it fit best? The first question is related to cluster analysis, but is also discussed from some perspective in this chapter. The second question is on pattern classification and that is what will be the main concern here.

The overall structure of a pattern recognition system may be summarized as in Figure 1. Objects have first to be appropriately represented before a generalization can be derived. Depending on the demands of the procedures used for this the representation has to be adapted, e.g. transformed, scaled or simplified.

The procedures discussed in this chapter are partially also studied in areas like statistical learning theory [32], machine learning [25] and neural networks [14]. As the emphasis in pattern recognition is close to application areas, questions related to the representation of the objects are important here: how are objects described (e.g. features, distances to prototypes), how extensive may this description be, what are the ways to incorporate knowledge from the application domain? Representations have to be adapted to fit the tools that are used later. Simplifications of representations like feature reduction and prototype selection should thereby be considered.

In order to derive, from a training set, a classifier that is valid for new objects (i.e. that it is able to generalize) the representation should fulfill an important condition: representations of similar real world objects have to be similar as well. The representations should be close. This is the so-called compactness hypothesis[2] on which the generalization from examples to new, unseen objects is built. It enables the estimation of their class labels on the basis of distances to examples or on class densities derived from examples.

Objects are traditionally represented by vectors in a feature space. An important recent development to incorporate domain knowledge is the representation of objects by their relation to other objects. This may be done by a so called kernel method[29], derived from features, or directly on dissimilarities computed from the raw data[26].

We will assume that, after processing the raw measurements, objects are given in a p-dimensional vector space Ω. Traditionally this space is spanned by p features, but also the dissimilarities with p prototype objects may be used. To simplify the discussion we will use the term feature space for both. If K is the number of classes to be distinguished, a pattern classification system, or shortly classifier $C(x)$ is a function or a procedure that assigns to each object $\mathbf{x}$ in Ω a class ω_c, with $c = 1, ..., K$. Such a classifier has to be derived from a set of examples $\mathcal{X}^{\text{tr}} = \{\mathbf{x}_i, i = 1...N\}$ of known classes y_i. $\mathcal{X}^{\text{tr}}$ will be called the training set and $y_i \in \omega_c, c = 1...K$ a label. Unless otherwise stated it is assumed that y_i is unique (objects belong to just a single class) and is known for all objects in $\mathcal{X}^{\text{tr}}$.

In section 2 training procedures will be discussed to derive classifiers $C(x)$ from training sets. The performance of these classifiers is usually not just related the quality of the features (their ability to show class differences) but also to their number, i.e. the dimensionality of the feature space. A growing number of features

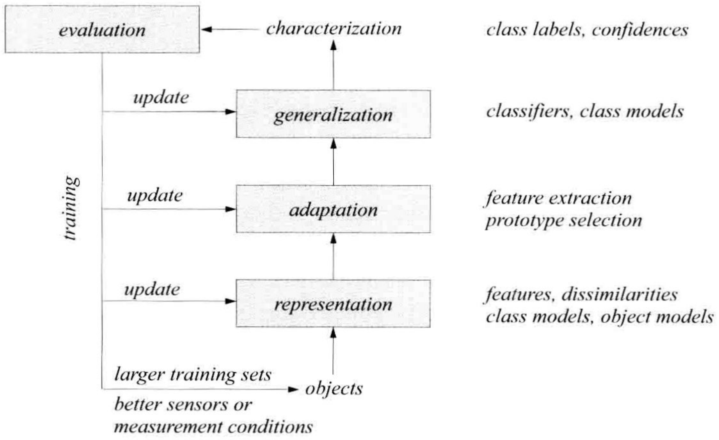


Fig. 1. The pattern recognition system

may increase the class separability, but, may also decrease the statistical accuracy of the training procedure. It is thereby important to have a small number of good features. In section 3 a review is given of ways to reduce the number of features by selection or by combination (so called feature extraction). The evaluation of classifiers, discussed in section 4, is an important topic. As the characteristics of new applications are often unknown before, the best algorithms for feature reduction and classification have to be found iteratively on the basis of unbiased and accurate testing procedures.

This chapter builds further on earlier reviews of the area of statistical pattern recognition by Fukunaga [12] and by Jain et al [16]. It is inevitable to repeat and summarize them partly. We will, however, also discuss some new directions like one-class classifiers, combining classifiers, dissimilarity representations and techniques for building good classifiers and reducing the feature space simultaneously. In the last section of this chapter, the discussion, we will return to these new developments.

2. Classifiers

For the development of classifiers, we have to consider two main aspects: the basic assumptions that the classifier makes about the data (which results in a functional form of the classifier), and the optimization procedure to fit the model to the training data. It is possible to consider very complex classifiers, but without efficient methods to fit these classifiers to the data, they are not useful. Therefore, in many cases the functional form of the classifier is restricted by the available optimization routines.

We will start discussing the two-class classification problem. In the first three sections, 2.1, 2.2 and 2.3, the three basic approaches with their assumptions are given: first, modeling the class posteriors, second, modeling class conditional probabilities and finally modeling the classification boundary. In section 2.4 we discuss how these approaches can be extended to work for more than two classes. In the next section, the special case is considered where just one of the classes is reliably sampled. The last section, 2.6, discusses the possibilities to combine several (non-optimal) classifiers.

2.1. *Bayes classifiers and approximations*

A classifier should assign a new object $\mathbf{x}$ to the most likely class. In a probabilistic setting this means that the label of the class with the highest posterior probability should be chosen. This class can be found when $p(\omega_1|\mathbf{x})$ and $p(\omega_2|\mathbf{x})$ (for a two class classification problem) are known. The classifier becomes:

$$\text{if} \quad p(\omega_1|\mathbf{x}) > p(\omega_2|\mathbf{x}) \quad \text{assign object } \mathbf{x} \text{ to } \omega_1, \text{ otherwise to } \omega_2. \tag{1}$$

When we assume that $p(\omega_1|\mathbf{x})$ and $p(\omega_2|\mathbf{x})$ are known, and further assume that misclassifying an object originating from ω_1 to ω_2 is as costly as vise versa, classifier (1) is the theoretical optimal classifier and will make the minimum error. This classifier is called the *Bayes optimal classifier.*

In practice $p(\omega_1|\mathbf{x})$ and $p(\omega_2|\mathbf{x})$ are not known, only samples $\mathbf{x}_i$ are available, and the misclassification costs might be only known in approximation. Therefore approximations to the Bayes optimal classifier have to be made. This classifier can be approximated in several different ways, depending on knowledge of the classification problem.

The first way is to approximate the class posterior probabilities $p(\omega_c|\mathbf{x})$. The *logistic classifier* assumes a particular model for the class posterior probabilities:

$$p(\omega_1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}, \qquad p(\omega_2|\mathbf{x}) = 1 - p(\omega_2|\mathbf{x}), \tag{2}$$

where $\mathbf{w}$ is a p-dimensional weight vector. This basically implements a linear classifier in the feature space.

An approach to fit this logistic classifier (2) to training data $\mathcal{X}^{\mathrm{tr}}$, is to maximize the data likelihood L:

$$L = \prod_i^N p(\omega_1|\mathbf{x}_i)^{n_1(\mathbf{x})} p(\omega_2|\mathbf{x}_i)^{n_2(\mathbf{x})}, \tag{3}$$

where $n_c(\mathbf{x})$ is 1 if object $\mathbf{x}$ belongs to class ω_c, and 0 otherwise. This can be done by, for instance, an iterative gradient ascent method. Weights are iteratively updated using:

$$\mathbf{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} + \eta \frac{\partial L}{\partial \mathbf{w}}, \tag{4}$$

where η is a suitably chosen learning rate parameter. In Ref. 1 the first (and second) derivative of L with respect to $\mathbf{w}$ are derived for this and can be plugged into (4).

2.2. *Class densities and Bayes rule*

Assumptions on $p(\omega|\mathbf{x})$ are often difficult to make. Sometimes it is more convenient to make assumptions on the class conditional probability densities $p(\mathbf{x}|\omega)$: they indicate the distribution of the objects which are drawn from one of the classes. When assumptions on these distributions can be made, classifier (1) can be derived using Bayes' decision rule:

$$p(\omega|\mathbf{x}) = \frac{p(\mathbf{x}|\omega)p(\omega)}{p(\mathbf{x})}. \tag{5}$$

This rule basically rewrites the class posterior probabilities in terms of the class conditional probabilities and the class priors $p(\omega)$. This result can be substituted into (1), resulting in the following form:

$$\text{if} \quad p(\mathbf{x}|\omega_1)p(\omega_1) > p(\mathbf{x}|\omega_2)p(\omega_2) \quad \text{assign } \mathbf{x} \text{ to } \omega_1, \text{ otherwise to } \omega_2. \tag{6}$$

The term $p(\mathbf{x})$ is ignored because this is constant for a given $\mathbf{x}$. Any monotonically increasing function can be applied to both sides without changing the final decision. In some cases, a suitable choice will simplify the notation significantly. In particular, using a logarithmic transformation can simplify the classifier when functions from the exponential family are used.

For the special case of a two-class problem the classifiers can be rewritten in terms of a single discriminant function $f(\mathbf{x})$ which is the difference between the left hand side and the right hand side. A few possibilities are:

$$f(\mathbf{x}) = p(\omega_1|\mathbf{x}) - p(\omega_2|\mathbf{x}), \tag{7}$$

$$f(\mathbf{x}) = p(\mathbf{x}|\omega_1)p(\omega_1) - p(\mathbf{x}|\omega_2)p(\omega_2), \tag{8}$$

$$f(\mathbf{x}) = \ln\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \ln\frac{p(\omega_1)}{p(\omega_2)}. \tag{9}$$

The classifier becomes:

$$\text{if} \quad f(\mathbf{x}) > 0 \quad \text{assign } \mathbf{x} \text{ to } \omega_1, \text{ otherwise to } \omega_2. \tag{10}$$

In many cases fitting $p(\mathbf{x}|\omega)$ on training data is relatively straightforward. It is the standard density estimation problem: fit a density on a data sample. To estimate each $p(\mathbf{x}|\omega)$ the objects from just one of the classes ω is used.

Depending on the functional form of the class densities, different classifiers are constructed. One of the most common approaches is to assume a Gaussian density for each of the classes:

$$p(\mathbf{x}|\omega) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{11}$$

where $\boldsymbol{\mu}$ is the (p-dimensional) mean of the class ω, and Σ is the covariance matrix. Further, $|\Sigma|$ indicates the determinant of Σ and Σ^{-1} its inverse. For the explicit values of the parameters $\boldsymbol{\mu}$ and Σ usually the maximum likelihood estimates are plugged in, therefore this classifier is called the *plug-in Bayes classifier.* Extra complications occur when the sample size N is insufficient to (in particular) compute Σ^{-1}. In these cases a standard solution is to regularize the covariance matrix such that the inverse can be computed:

$$\Sigma_\lambda = \Sigma + \lambda\mathcal{I}, \tag{12}$$

where $\mathcal{I}$ is the $p \times p$ identity matrix, and λ is the regularization parameter to set the trade off between the estimated covariance matrix and the regularizer $\mathcal{I}$.

Substituting (11) for each of the classes ω_1 and ω_2 (with their estimated $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and Σ_1, Σ_2) into (9) results in:

$$\begin{aligned} f(\mathbf{x}) = {} & \frac{1}{2}\mathbf{x}^T(\Sigma_2^{-1} - \Sigma_1^{-1})\mathbf{x} + \frac{1}{2}(\boldsymbol{\mu}_1\Sigma_1^{-1} - \boldsymbol{\mu}_2\Sigma_2^{-1})^T\mathbf{x} \\ & -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu}_2 - \frac{1}{2}\ln|\Sigma_1| + \frac{1}{2}\ln|\Sigma_2| + \ln\frac{p(\omega_1)}{p(\omega_2)}. \end{aligned} \tag{13}$$

This classifier rule is quadratic in terms of $\mathbf{x}$, and it is therefore called the *normal-based quadratic classifier.*

For the quadratic classifier a full covariance matrix has to be estimated for each of the classes. In high dimensional feature spaces it can happen that insufficient data is available to estimate these covariance matrices reliably. By restricting the covariance matrices to have less free variables, estimations can become more reliable. One approach the reduce the number of parameters, is to assume that both classes have an identical covariance structure: $\Sigma_1 = \Sigma_2 = \Sigma$. The classifier simplifies to:

$$f(\mathbf{x}) = \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\omega_1)}{p(\omega_2)} \tag{14}$$

Because this classifier is linear in terms of $\mathbf{x}$, this classifier is called the *normal-based linear classifier*.

For the linear and the quadratic classifier, strong class distributional assumptions are made: each class has a Gaussian distribution. In many applications this cannot be assumed, and more flexible class models have to be used. One possibility is to use a 'non-parametric' model. An example is the Parzen density model. Here the density is estimated by summing local kernels with a fixed size h which are centered on each of the training objects:

$$p(\mathbf{x}|\omega) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(\mathbf{x}; \mathbf{x}_i, h\mathcal{I}), \tag{15}$$

where $\mathcal{I}$ is the identity matrix and h is the width parameter which has to be optimized. By substituting (15) into (6), the *Parzen classifier* is defined. The only free parameter in this classifier is the size (or width) h of the kernel. Optimizing this parameter by maximizing the likelihood on the training data, will result in the solution $h = 0$. To avoid this, a leave-one-out procedure can be used [9].

2.3. *Boundary methods*

Density estimation in high dimensional spaces is difficult. In order to have a reliable estimate, large amounts of training data should be available. Unfortunately, in many cases the number of training objects is limited. Therefore it is not always wise to estimate the class distributions completely. Looking at (1), (6) and (10), it is only of interest which class is to be preferred over the other. This problem is simpler than estimating $p(\mathbf{x}|\omega)$. For a two-class problem, we just a function $f(\mathbf{x})$ is needed which is positive for objects of ω_1 and negative otherwise. In this section we will list some classifiers which avoid estimating $p(\mathbf{x}|\omega)$ but try to obtain a suitable $f(\mathbf{x})$.

The *Fisher classifier* searches to find a direction $\mathbf{w}$ in the feature space, such that the two classes are separated as well as possible. The degree in which the two classes are separated, is measured by the so-called Fisher ratio, or Fisher criterion:

$$J = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}. \tag{16}$$

Here m_1 and m_2 are the means of the two classes, projected onto the direction $\mathbf{w}$: $m_1 = \mathbf{w}^T\boldsymbol{\mu}_1$ and $m_2 = \mathbf{w}^T\boldsymbol{\mu}_2$. The s_1 and s_2 are the variances of the two classes projected onto $\mathbf{w}$. The criterion therefore favors directions in which the means are far apart and the variances are small.

This Fisher ratio can be explicitly rewritten in terms of $\mathbf{w}$. First we rewrite $s_c^2 = \sum_{\mathbf{x}\in\omega_c}(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu}_c)^2 = \sum_{\mathbf{x}\in\omega_c}\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_c)(\mathbf{x}-\boldsymbol{\mu}_c)^T\mathbf{w} = \mathbf{w}^T S_c \mathbf{w}$. Second we write $(m_1 - m_2)^2 = (\mathbf{w}^T\boldsymbol{\mu}_1 - \mathbf{w}^T\boldsymbol{\mu}_2)^2 = \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{w} = \mathbf{w}^T S_B \mathbf{w}$. The term S_B is also called the *between scatter* matrix. J becomes:

$$J = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_1 \mathbf{w} + \mathbf{w}^T S_2 \mathbf{w}} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}, \tag{17}$$

where $S_W = S_1 + S_2$ is also called the *within scatter* matrix.

In order to optimize (17), we set the derivative of (17) to zero and obtain:

$$(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}. \tag{18}$$

We are interested in the direction of $\mathbf{w}$ and not in the length, so we drop the scalar terms between brackets. Further, from the definition of S_B it follows that $S_B\mathbf{w}$ is always in the direction $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$. Multiplying both sides of (18) by S_W^{-1} gives:

$$\mathbf{w} \sim S_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{19}$$

This classifier is known as the Fisher classifier. Note that the threshold b is not defined for this classifier. It is also linear and requires the inversion of the within-scatter S_W. This formulation yields an identical shape of $\mathbf{w}$ as the expression in (14), although the classifiers use very different starting assumptions!

Most classifiers which have been discussed so far, have a very restricted form of their decision boundary. In many cases these boundaries are not flexible enough to follow the true decision boundaries. A flexible method is the *k-nearest neighbor rule*. This classifier looks locally which labels are most dominant in the training set. First it finds the k nearest objects in the training set $NN(\mathbf{x})$, and then counts the number of these neighbors are from class ω_1 or ω_2:

$$\text{if} \quad n_1 > n_2 \quad \text{assign } \mathbf{x} \text{ to } \omega_1, \text{ otherwise to } \omega_2. \tag{20}$$

Although the training of the k-nearest neighbor classifier is trivial (it only has to store all training objects, k can simply be optimized by a leave-one-out estimation), it may become expensive to classify a new object $\mathbf{x}$. For this the distances to all training objects have to be computed, which may be prohibitive for large training sets and high dimensional feature spaces.

Another classifier which is flexible but does not require the storage of the full training set is the *multi-layered feed-forward neural network*[4]. A neural network is a collection of small processing units, called the neurons, which are interconnected by weights $\mathbf{w}$ and $\mathbf{v}$ to form a network. A schematic picture is shown in Figure 2. An input object $\mathbf{x}$ is processed through different layers of neurons, through the hidden layer to the output layer. The output of the j-th output neuron becomes:

$$o_j(\mathbf{x}) = h_j\left(\sum_{i=1}^{n} v_i h_i(\mathbf{w}_i^T\mathbf{x})\right) \tag{21}$$

(see Figure 2 for the meaning of the variables). The object $\mathbf{x}$ is now assigned to the class j for which the corresponding output neuron has the highest output o_j.