# Classification, Parameter Estimation and State Estimation

## AN ENGINEERING APPROACH USING MATLAB

SECOND EDITION

BANGJUN LEI • GUANGZHU XU • MING FENG • YAOBIN ZOU
FERDINAND VAN DER HEIJDEN • DICK DE RIDDER • DAVID M. J. TAX

WILEY

# Classification, Parameter Estimation and State Estimation

# Classification, Parameter Estimation and State Estimation

An Engineering Approach Using MATLAB

Second Edition

*Bangjun Lei*

*Guangzhu Xu*

*Ming Feng*

*Yaobin Zou*

*Ferdinand van der Heijden*

*Dick de Ridder*

*David M. J. Tax*

WILEY

This edition first published 2017


*Registered Offices*
John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA
John Wiley & Sons, Ltd., The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

*Editorial Office*
The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.



*Library of Congress Cataloging-in-Publication Data*

Names: Heijden, Ferdinand van der. | Lei, Bangjun, 1973– author. | Xu, Guangzhu, 1979– author. | Ming, Feng, 1957– author. | Zou, Yaobin, 1978– author. | Ridder, Dick de, 1971– author. | Tax, David M. J., 1973– author.
Title: Classification, parameter estimation, and state estimation : an engineering approach using MATLAB / Bangjun Lei, Guangzhu Xu, Ming Feng, Yaobin Zou, Ferdinand van der Heijden, Dick de Ridder, David M. J. Tax.
Description: Second edition. | Hoboken, NJ, USA : John Wiley & Sons, Inc., 2017. | Revised edition of: Classification, parameter estimation, and state estimation : an engineering approach using MATLAB / F. van der Heijden … [et al.]. 2004. | Includes bibliographical references and index.
Identifiers: LCCN 2016059294 (print) | LCCN 2016059809 (ebook) | ISBN 9781119152439 (cloth) | ISBN 9781119152446 (pdf) | ISBN 9781119152453 (epub)
Subjects: LCSH: Engineering mathematics–Data processing. | MATLAB. | Measurement–Data processing. | Estimation theory–Data processing.
Classification: LCC TA331 .C53 2017 (print) | LCC TA331 (ebook) | DDC 681/.2–dc23
LC record available at https://lccn.loc.gov/2016059294

Cover Design: Wiley
Cover Images: neural network © maxuser/Shutterstock; digital circuit board © Powderblue/Shutterstock

Set in 10/12pt WarnockPro by Aptara Inc., New Delhi, India

Printed in Great Britain by TJ International Ltd, Padstow, Cornwall

10 9 8 7 6 5 4 3 2 1

# Contents

# Preface

Information processing has always been an important factor in the development of human society and its role is still increasing. The inventions of advanced information devices paved the way for achievements in a diversity of fields like trade, navigation, agriculture, industry, transportation and communication. The term 'information device' refers here to systems for the sensing, acquisition, processing and outputting of information from the real world. Usually, they are measurement systems. Sensing and acquisition provide us with signals that bear a direct relation to some of the physical properties of the sensed object or process. Often, the information of interest is hidden in these signals. Signal processing is needed to reveal the information and to transform it into an explicit form. Further, in the past 10 years image processing (together with intelligent computer vision) has gone through rapid developments. There are substantial new developments on, for example, machine learning methods (such as Adaboost and it's varieties, Deep learning etc.) and particle filtering like parameter estimation methods.

The three topics discussed in this book, classification, parameter estimation and state estimation, share a common factor in the sense that each topic provides the theory and methodology for the functional design of the signal processing part of an information device. The major distinction between the topics is the type of information that is outputted. In classification problems the output is discrete, that is a class, a label or a category. In estimation problems, it is a real-valued scalar or vector. Since these problems occur either in a static or in a dynamic setting, actually four different topics can be distinguished. The term state

estimation refers to the dynamic setting. It covers both discrete and real-valued cases (and sometimes even mixed cases).

The similarity between the topics allows one to use a generic methodology, that is Bayesian decision theory. Our aim is to present this material concisely and efficiently by an integrated treatment of similar topics. We present an overview of the core mathematical constructs and the many resulting techniques. By doing so, we hope that the reader recognizes the connections and the similarities between these constructs, but also becomes aware of the differences. For instance, the phenomenon of overfitting is a threat that ambushes all four cases. In a static classification problem it introduces large classification errors, but in the case of a dynamic state estimation it may be the cause of instable behaviour. Further, in this edition, we made some modifications to accommodate engineering requests on intelligent computer vision.

Our goal is to emphasize the engineering aspects of the matter. Instead of a purely theoretical and rigorous treatment, we aim for the acquirement of skills to bring theoretical solutions to practice. The models that are needed for the application of the Bayesian framework are often not available in practice. This brings in the paradigm of statistical inference, that is learning from examples. MATLAB®* is used as a vehicle to implement and to evaluate design concepts.

As alluded to above, the range of application areas is broad. Application fields are found within computer vision, mechanical engineering, electrical engineering, civil engineering, environmental engineering, process engineering, geo-informatics, bio-informatics, information technology, mechatronics, applied physics, and so on. The book is of interest to a range of users, from the first-year graduate-level student up to the experienced professional. The reader should have some background knowledge with respect to linear algebra, dynamic systems and probability theory. Most educational programmes offer courses on these topics as part of undergraduate education. The appendices contain reviews of the relevant material. Another target group

* MATLAB® is a registered trademark of The MathWorks, Inc. (http://www.mathworks.com).

is formed by the experienced engineers working in industrial development laboratories. The numerous examples of MATLAB® code allow these engineers to quickly prototype their designs.

The book roughly consists of three parts. The first part, Chapter 2, presents an introduction to the PRTools used throughout this book. The second part, Chapters 3, 4 and 5, covers the theory with respect to classification and estimation problems in the static case, as well as the dynamic case. This part handles problems where it is assumed that accurate models, describing the physical processes, are available. The third part, Chapters 6 up to 8, deals with the more practical situation in which these models are not or only partly available. Either these models must be built using experimental data or these data must be used directly to train methods for estimation and classification. The final chapter presents three worked out problems. The selected bibliography has been kept short in order not to overwhelm the reader with an enormous list of references.

The material of the book can be covered by two semester courses. A possibility is to use Chapters 3, 4, 6, 7 and 8 for a one-semester course on Classification and Estimation. This course deals with the static case. An additional one-semester course handles the dynamic case, that is Optimal Dynamic Estimation, and would use Chapter 5. The prerequisites for Chapter 5 are mainly concentrated in Chapter 4. Therefore, it is recommended to include a review of Chapter 4 in the second course. Such a review will make the second course independent from the first one.

Each chapter is closed with a number of exercises. The mark at the end of each exercise indicates whether the exercise is considered easy ('0'), moderately difficult ('*') or difficult ('**'). Another possibility to acquire practical skills is offered by the projects that accompany the text. These projects are available at the companion website. A project is an extensive task to be undertaken by a group of students. The task is situated within a given theme, for instance, classification using supervised learning, unsupervised learning, parameter estimation, dynamic labelling and dynamic estimation. Each project consists of a set of instructions together with data that should be used to solve the problem.

The use of MATLAB® tools is an integrated part of the book. MATLAB® offers a number of standard toolboxes that are useful for parameter estimation, state estimation and data analysis. The standard software for classification and unsupervised learning is not complete and not well structured. This motivated us to develop the PRTools software for all classification tasks and related items. PRTools is a MATLAB® toolbox for pattern recognition. It is freely available for non-commercial purposes. The version used in the text is compatible with MATLAB® Version 5 and higher. It is available from http://37steps.com.

The authors keep an open mind for any suggestions and comments (which should be addressed to cpese@wiley.com). A list of errata and any other additional comments will be made available at the companion website.

## Acknowledgements

We thank everyone who has made this book possible. Special thanks are given to Dr. Robert P. W. Duin for his contribution to the first version of this book and for allowing us to use PRTools and all materials on 37steps.com throughout this book. Thanks are also extended to Dr. Ela Pekalska for the courtesy of sharing documents of 37steps.com with us.

## About the Companion Website

This book is accompanied by a companion website:

**www.wiley.com/go/vanderheijden/classification_parameterestimation_stateestimation/**

The website includes:

- Code and Datasets
- Tutorials
- Pictures
- Errata

# 1

# Introduction

Engineering disciplines are those fields of research and development that attempt to create products and systems operating in, and dealing with, the real world. The number of disciplines is large, as is the range of scales that they typically operate in: from the very small scale of nanotechnology up to very large scales that span whole regions, for example water management systems, electric power distribution systems or even global systems (e.g. the global positioning system, GPS). The level of advancement in the fields also varies wildly, from emerging techniques (again, nanotechnology) to trusted techniques that have been applied for centuries (architecture, hydraulic works). Nonetheless, the disciplines share one important aspect: engineering aims at designing and manufacturing systems that interface with the world around them.

Systems designed by engineers are often meant to influence their environment: to manipulate it, to move it, to stabilize it, to please it, and so on. To enable such actuation, these systems need information, for example values of physical quantities describing their environments and possibly also describing themselves. Two types of information sources are available: *prior* knowledge and *empirical* knowledge. The latter is knowledge obtained by sensorial observation. Prior knowledge is the knowledge that was already there before a given observation became available (this does not imply that prior knowledge is obtained without any observation). The combination of prior knowledge and empirical knowledge leads to *posterior* knowledge.

*Classification, Parameter Estimation and State Estimation: An Engineering Approach Using* MATLAB,
Second Edition. Bangjun Lei, Guangzhu Xu, Ming Feng, Yaobin Zou, Ferdinand van der Heijden,
Dick de Ridder, and David M. J. Tax.

Companion Website: www.wiley.com/go/vanderheijden/classification_parameterestimation_stateestimation/

The sensory subsystem of a system produces measurement signals. These signals carry the empirical knowledge. Often, the direct usage of these signals is not possible, or is inefficient. This can have several causes:

- The information in the signals is not represented in an explicit way. It is often hidden and only available in an indirect, encoded, form.
- Measurement signals always come with noise and other hard-to-predict disturbances.
- The information brought forth by posterior knowledge is more accurate and more complete than information brought forth by empirical knowledge alone. Hence, measurement signals should be used in combination with prior knowledge.

Measurement signals need processing in order to suppress the noise and to disclose the information required for the task at hand.

## 1.1 The Scope of the Book

In a sense, classification and estimation deal with the same problem: given the measurement signals from the environment, how can the information that is needed for a system to operate in the real world be inferred? In other words, how should the measurements from a sensory system be processed in order to bring maximal information in an explicit and usable form? This is the main topic of this book.

Good processing of the measurement signals is possible only if some knowledge and understanding of the environment and the sensory system is present. Modelling certain aspects of that environment – like objects, physical processes or events – is a necessary task for the engineer. However, straightforward modelling is not always possible. Although the physical sciences provide ever deeper insight into nature, some systems are still only partially understood; just think of the weather. Even if systems are well understood, modelling them exhaustively may be beyond our current capabilities (i.e. computer power) or beyond the scope of the application. In such cases, approximate general models,


**Figure 1.1** Licence plate recognition: a classification problem with noisy measurements.

but adapted to the system at hand, can be applied. The development of such models is also a topic of this book.

### 1.1.1 Classification

The title of the book already indicates the three main subtopics it will cover: classification, parameter estimation and state estimation. In classification, one tries to assign a class label to an object, a physical process or an event. Figure 1.1 illustrates the concept. In a speeding detector, the sensors are a radar speed detector and a high-resolution camera, placed in a box beside a road. When the radar detects a car approaching at too high a velocity (a parameter estimation problem), the camera is signalled to acquire an image of the car. The system should then recognize the licence plate, so that the driver of the car can be fined for the speeding violation. The system should be robust to differences in car model, illumination, weather circumstances, etc., so some pre-processing is necessary: locating the licence plate in the image, segmenting the individual characters and converting it into a binary image. The problem then breaks down to a number of individual classification problems. For each of the locations on the license plate, the input consists of a binary image of a character, normalized for size, skew/rotation and intensity. The desired output is the label of the true character, that is one of 'A', 'B',…, 'Z', '0',…, '9'.

Detection is a special case of classification. Here, only two class labels are available, for example 'yes' and 'no'. An example is a quality control system that approves the products of a manufacturer or refuses them. A second problem closely related to classification is identification: the act of proving that an object-under-test and a second object that is previously seen are the same. Usually, there is a large database of previously seen objects to choose from. An example is biometric identification, for example fingerprint recognition or face recognition. A third problem that can be solved by classification-like techniques is retrieval from a database, for example finding an image in an image database by specifying image features.

### 1.1.2 Parameter Estimation

In parameter estimation, one tries to derive a parametric description for an object, a physical process or an event. For example, in a beacon-based position measurement system (Figure 1.2), the goal is to find the position of an object, for example a ship or a mobile robot. In the two-dimensional case, two beacons with known reference positions suffice. The sensory system provides two measurements: the distances from the beacons to the object, $r_1$ and $r_2$. Since the position of the object involves two parameters, the estimation seems to boil down to solving two equations with two unknowns. However,

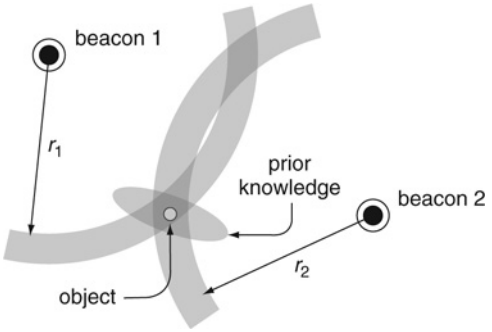


**Figure 1.2** Position measurement: a parameter estimation problem handling uncertainties.

the situation is more complex because measurements always come with uncertainties. Usually, the application not only requires an estimate of the parameters but also an assessment of the uncertainty of that estimate. The situation is even more complicated because some prior knowledge about the position must be used to resolve the ambiguity of the solution. The prior knowledge can also be used to reduce the uncertainty of the final estimate.

In order to improve the accuracy of the estimate the engineer can increase the number of (independent) measurements to obtain an overdetermined system of equations. In order to reduce the cost of the sensory system, the engineer can also decrease the number of measurements, leaving us with fewer measurements than parameters. The system of equations is then underdetermined, but estimation is still possible if enough prior knowledge exists or if the parameters are related to each other (possibly in a statistical sense). In either case, the engineer is interested in the uncertainty of the estimate.

### 1.1.3 State Estimation

In state estimation, one tries to do either of the following – either assigning a class label or deriving a parametric (real-valued) description – but for processes that vary in time or space. There is a fundamental difference between the problems of classification and parameter estimation, on the one hand, and state estimation, on the other hand. This is the ordering in time (or space) in state estimation, which is absent from classification and parameter estimation. When no ordering in the data is assumed, the data can be processed in any order. In time series, ordering in time is essential for the process. This results in a fundamental difference in the treatment of the data.

In the discrete case, the states have discrete values (classes or labels) that are usually drawn from a finite set. An example of such a set is the alarm stages in a safety system (e.g. 'safe', 'pre-alarm', 'red alert', etc.). Other examples of discrete state estimation are speech recognition, printed or handwritten text recognition and the recognition of the operating modes of a machine.

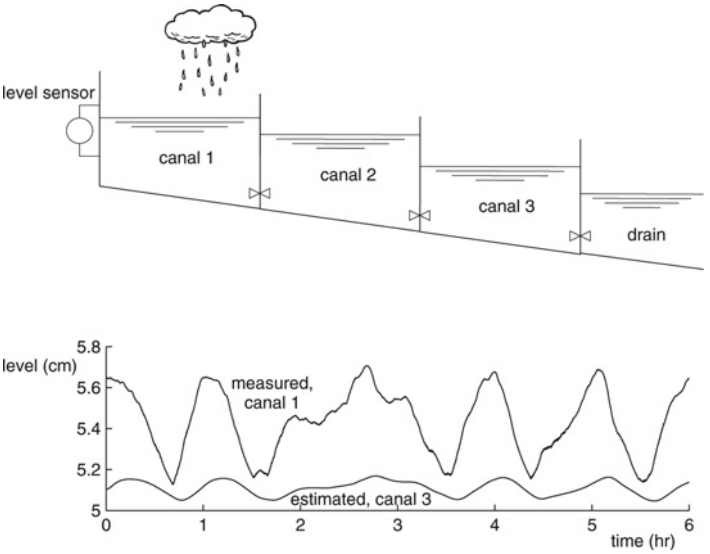


**Figure 1.3** Assessment of water levels in a water management system: a state estimation problem (the data are obtained from a scale model).

An example of real-valued state estimation is the water management system of a region. Using a few level sensors and an adequate dynamical model of the water system, a state estimator is able to assess the water levels even at locations without level sensors. Short-term prediction of the levels is also possible. Figure 1.3 gives a view of a simple water management system of a single canal consisting of three linearly connected compartments. The compartments are filled by the precipitation in the surroundings of the canal. This occurs randomly but with a seasonal influence. The canal drains its water into a river. The measurement of the level in one compartment enables the estimation of the levels in all three compartments. For that, a dynamic model is used that describes the relations between flows and levels. Figure 1.3 shows an estimate of the level of the third compartment using measurements of the level in the first compartment. Prediction of the level in the third compartment is possible due to the causality of the process and the delay between the levels in the compartments.

### 1.1.4 Relations between the Subjects

The reader who is familiar with one or more of the three subjects might wonder why they are treated in one book. The three subjects share the following factors:

- In all cases, the engineer designs an instrument, that is a system whose task is to extract information about a real-world object, a physical process or an event.
- For that purpose, the instrument will be provided with a sensory subsystem that produces measurement signals. In all cases, these signals are represented by vectors (with fixed dimension) or sequences of vectors.
- The measurement vectors must be processed to reveal the information that is required for the task at hand.
- All three subjects rely on the availability of models describing the object/physical process/event and of models describing the sensory system.
- Modelling is an important part of the design stage. The suitability of the applied model is directly related to the performance of the resulting classifier/estimator.

Since the nature of the questions raised in the three subjects is similar, the analysis of all three cases can be done using the same framework. This allows an economical treatment of the subjects. The framework that will be used is a probabilistic one. In all three cases, the strategy will be to formulate the posterior knowledge in terms of a conditional probability (density) function:

$$P(\text{quantities of interest} \mid \text{measurements available})$$

This so-called posterior probability combines the prior knowledge with the empirical knowledge by using Bayes' theorem for conditional probabilities. As discussed above, the framework is generic for all three cases. Of course, the elaboration of this principle for the three cases leads to different solutions because the nature of the 'quantities of interest' differs.

The second similarity between the topics is their reliance on models. It is assumed that the constitution of the object/physical process/event (including the sensory system) can be captured by a mathematical model. Unfortunately, the physical structures responsible for generating the objects/process/events are often

unknown, or at least partly unknown. Consequently, the model is also, at least partly, unknown. Sometimes, some functional form of the model is assumed, but the free parameters still have to be determined. In any case, empirical data are needed in order to establish the model, to tune the classifier/estimator-under-development and also to evaluate the design. Obviously, the training/evaluation data should be obtained from the process we are interested in.

In fact, all three subjects share the same key issue related to modelling, namely the selection of the appropriate generalization level. The empirical data are only an example of a set of possible measurements. If too much weight is given to the data at hand, the risk of overfitting occurs. The resulting model will depend too much on the accidental peculiarities (or noise) of the data. On the other hand, if too little weight is given, nothing will be learned and the model completely relies on the prior knowledge. The right balance between these opposite sides depends on the statistical significance of the data. Obviously, the size of the data is an important factor. However, the statistical significance also holds a relation with dimensionality.

Many of the mathematical techniques for modelling, tuning, training and evaluation can be shared between the three subjects. Estimation procedures used in classification can also be used in parameter estimation or state estimation, with just minor modifications. For instance, probability density estimation can be used for classification purposes and also for estimation. Data-fitting techniques are applied in both classification and estimation problems. Techniques for statistical inference can also be shared. Of course, there are also differences between the three subjects. For instance, the modelling of dynamic systems, usually called *system identification*, involves aspects that are typical for dynamic systems (i.e. determination of the order of the system, finding an appropriate functional structure of the model). However, when it finally comes to finding the right parameters of the dynamic model, the techniques from parameter estimation apply again.

Figure 1.4 shows an overview of the relations between the topics. Classification and parameter estimation share a common foundation indicated by 'Bayes'. In combination with models for dynamic systems (with random inputs), the techniques for

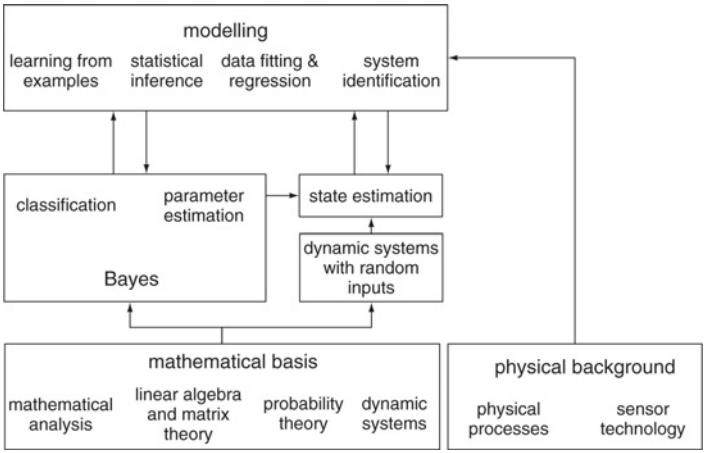


**Figure 1.4** Relations between the subjects.

classification and parameter estimation find their application in processes that proceed in time, that is state estimation. All this is built on a mathematical basis with selected topics from mathematical analysis (dealing with abstract vector spaces, metric spaces and operators), linear algebra and probability theory. As such, classification and estimation are not tied to a specific application. The engineer, who is involved in a specific application, should add the individual characteristics of that application by means of the models and prior knowledge. Thus, apart from the ability to handle empirical data, the engineer must also have some knowledge of the physical background related to the application at hand and to the sensor technology being used.

All three subjects are mature research areas and many overview books have been written. Naturally, by combining the three subjects into one book, it cannot be avoided that some details are left out. However, the discussion above shows that the three subjects are close enough to justify one integrated book covering these areas.

The combination of the three topics into one book also introduces some additional challenges if only because of the differences in terminology used in the three fields. This is, for instance, reflected in the difference in the term used for 'measurements'.

In classification theory, the term 'features' is frequently used as a replacement for 'measurements'. The number of measurements is called the 'dimension', but in classification theory the term 'dimensionality' is often used.[1] The same remark holds true for notations. For instance, in classification theory the measurements are often denoted by **x**. In state estimation, two notations are in vogue: either **y** or **z** (MATLAB® uses **y**, but we chose **z**). In all cases we tried to be as consistent as possible.

## 1.2 Engineering

The top-down design of an instrument always starts with some primary need. Before starting with the design, the engineer has only a global view of the system of interest. The actual need is known only at a high and abstract level. The design process then proceeds through a number of stages during which progressively more detailed knowledge becomes available and the system parts of the instrument are described at lower and more concrete levels. At each stage, the engineer has to make design decisions. Such decisions must be based on explicitly defined evaluation criteria. The procedure, the elementary design step, is shown in Figure 1.5. It is used iteratively at the different levels and for the different system parts.

An elementary design step typically consists of collecting and organizing knowledge about the design issue of that stage, followed by an explicit formulation of the involved task. The next step is to associate the design issue with an evaluation criterion. The criterion expresses the suitability of a design concept related to the given task, but also other aspects can be involved, such as cost of manufacturing, computational cost or throughput.

1 Our definition complies with the mathematical definition of 'dimension', i.e. the maximal number of independent vectors in a vector space. In MATLAB® the term 'dimension' refers to an index of a multidimensional array as in phrases like: 'the first dimension of a matrix is the row index' and 'the number of dimensions of a matrix is two'. The number of elements along a row is the 'row dimension' or 'row length'. In MATLAB® the term 'dimensionality' is the same as the 'number of dimensions'.

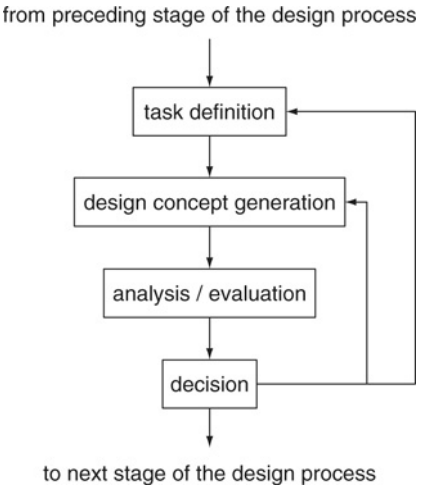


**Figure 1.5** An elementary step in the design process (Finkelstein and Finkelstein, 1994).

Usually, there are a number of possible design concepts to select from. Each concept is subjected to an analysis and an evaluation, possibly based on some experimentation. Next, the engineer decides which design concept is most appropriate. If none of the possible concepts are acceptable, the designer steps back to an earlier stage to alter the selections that have been made there.

One of the first tasks of the engineer is to identify the actual need that the instrument must fulfil. The outcome of this design step is a description of the functionality, for example a list of preliminary specifications, operating characteristics, environmental conditions, wishes with respect to user interface and exterior design. The next steps deal with the principles and methods that are appropriate to fulfil the needs, that is the *internal functional structure* of the instrument. At this level, the system under design is broken down into a number of functional components. Each component is considered as a subsystem whose input/output relations are mathematically defined. Questions related to the actual construction, realization of the functions, housing, etc., are later concerns.

The functional structure of an instrument can be divided roughly into sensing, processing and outputting (displaying,

recording). This book focuses entirely on the design steps related to *processing*. It provides:

- Knowledge about various methods to fulfil the processing tasks of the instrument. This is needed in order to generate a number of different design concepts.
- Knowledge about how to evaluate the various methods. This is needed in order to select the best design concept.
- A tool for the experimental evaluation of the design concepts.

The book does not address the topic 'sensor technology'. For this, many good textbooks already exist, for instance see Regtien *et al.* (2004) and Brignell and White (1996). Nevertheless, the sensory system does have a large impact on the required processing. For our purpose, it suffices to consider the sensory subsystem at an abstract functional level such that it can be described by a mathematical model.

## 1.3 The Organization of the Book

Chapter 2 focuses on the introduction of PRTools designed by Robert P.W.Duin. PRTools is a pattern recognition toolbox for MATLAB® freely available for non-commercial use. The pattern recognition routines and support functions offered by PRTools represent a basic set covering largely the area of statistical pattern recognition. In this book, except for additional notes, all examples are based on PRTools5.

The second part of the book, containing Chapters 3, 4 and 5, considers each of the three topics – classification, parameter estimation and state estimation – at a theoretical level. Assuming that appropriate models of the objects, physical process or events, and of the sensory system are available, these three tasks are well defined and can be discussed rigorously. This facilitates the development of a mathematical theory for these topics.

The third part of the book, Chapters 6 to 9, discusses all kinds of issues related to the deployment of the theory. As mentioned in Section 1.1, a key issue is modelling. Empirical data should be combined with prior knowledge about the physical process underlying the problem at hand, and about the sensory system used. For classification problems, the empirical data are often

represented by labelled training and evaluation sets, that is sets consisting of measurement vectors of objects together with the true classes to which these objects belong. Chapters 6 and 7 discuss several methods to deal with these sets. Some of these techniques – probability density estimation, statistical inference, data fitting – are also applicable to modelling in parameter estimation. Chapter 8 is devoted to unlabelled training sets. The purpose is to find structures underlying these sets that explain the data in a statistical sense. This is useful for both classification and parameter estimation problems. In the last chapter all the topics are applied in some fully worked out examples. Four appendices are added in order to refresh the required mathematical background knowledge.

The subtitle of the book, 'An Engineering Approach using MATLAB®', indicates that its focus is not just on the formal description of classification, parameter estimation and state estimation methods. It also aims to provide practical implementations of the given algorithms. These implementations are given in MATLAB®, which is a commercial software package for matrix manipulation. Over the past decade it has become the *de facto* standard for development and research in data-processing applications. MATLAB® combines an easy-to-learn user interface with a simple, yet powerful, language syntax and a wealth of functions organized in toolboxes. We use MATLAB® as a vehicle for experimentation, the purpose of which is to find out which method is the most appropriate for a given task. The final construction of the instrument can also be implemented by means of MATLAB®, but this is not strictly necessary. In the end, when it comes to realization, the engineer may decide to transform his or her design of the functional structure from MATLAB® to other platforms using, for instance, dedicated hardware, software in embedded systems or virtual instrumentation such as LabView.

MATLAB® itself has many standard functions that are useful for parameter estimation and state estimation problems. These functions are scattered over a number of toolboxes. The toolboxes are accompanied with a clear and crisp documentation, and for details of the functions we refer to that.

Most chapters are followed by a few exercises on the theory provided. However, we believe that only working with the actual algorithms will provide the reader with the necessary insight to

fully understand the matter. Therefore, a large number of small code examples are provided throughout the text. Furthermore, a number of data sets to experiment with are made available through the accompanying website.

## 1.4 Changes from First Edition

This edition attempts to put the book's emphasis more on image and video processing to cope with increasing interests on intelligent computer vision. More contents of most recent technological advancements are included. PRTools is updated to the newest version and all relevant examples are rewritten. Several practical systems are further implemented as showcase examples.

Chapter 1 is slightly modified to accommodate new changes in this Second Edition.

Chapter 2 is an expansion of Appendix E of the First Edition to accommodate the new changes of PRTools. Besides updating each subsection, the PRTools organization structure and implementation are also introduced.

Chapters 3 and 4 are, Chapters 2 and 3 in the First Edition, respectively.

Chapter 5 has now explicitly established the state space model and measurement model. A new example of motion tracking has been added. A new section on genetic station estimation has been written as Section 5.5. Further, an abbreviation of Chapter 8 of the First Edition has been formed as a new Section 5.6. The concept of 'continuous state variables' has been adjusted to 'infinite discrete-time state variables' and the concept of 'discrete state variables' to 'finite discrete-time state variables'. Several examples including 'special state space models' including "random constants', 'first-order autoregressive models', 'random walk' and 'second- order autoregressive models' have been removed.

In Chapter 6, Adaboost algorithm theory and its implementation with PRTools are added in Section 6.4 and convolutional neural networks (CNNs) are presented in Section 6.5.

In Chapter 7, several new methods of feature selection have been added in Section 7.2.3 to reflect the newest advancements on feature selection.

In Chapter 8, kernel principal component analysis is additionally described with several examples in Section 8.1.3.

In Chapter 9, three image recognition (objects recognition, shape recognition and face recognition) examples with PRTools routines are added.

## 1.5 References

Brignell, J. and White, N., *Intelligent Sensor Systems*, Revised edition, IOP Publishing, London, UK, 1996.

Finkelstein, L. and Finkelstein A.C.W., *Design Principles for Instrument Systems in Measurement and Instrumentation* (eds L. Finkelstein and K.T.V. Grattan), Pergamon Press, Oxford, UK, 1994.

Regtien, P.P.L., van der Heijden, F., Korsten, M.J. and Olthuis, W., *Measurement Science for Engineers*, Kogan Page Science, London, UK, 2004.

# 2

# PRTools Introduction

## 2.1 Motivation

Scientists should build their own instruments, or at least be able to open, investigate and understand the tools they are using. If, however, the tools are provided as a black box there should be a manual or literature available that fully explains the ins and outs. In principle, scientists should be able to create their measurement devices from scratch; otherwise the progress in science has no foundations.

In statistical pattern recognition one studies techniques for the generalization of examples to decision rules to be used for the detection and recognition of patterns in experimental data. This research area has a strong computational character, demanding a flexible use of numerical programs for data analysis as well as for the evaluation of the procedures. As still new methods are being proposed in the literature a programming platform is needed that enables a fast and flexible implementation.

MATLAB® is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, and testing and system evaluation. Pattern recognition is studied in almost all areas of applied science. Thereby the use of a widely available numerical toolset like MATLAB® may be profitable for both the use of existing techniques as well as for the study of new algorithms. Moreover, because of its general nature in comparison with more specialized statistical environments, it offers an easy

*Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*, Second Edition. Bangjun Lei, Guangzhu Xu, Ming Feng, Yaobin Zou, Ferdinand van der Heijden, Dick de Ridder, and David M. J. Tax.

Published 2017 by John Wiley & Sons, Ltd.
Companion Website: www.wiley.com/go/vanderheijden/classification_parameterestimation_stateestimation/

**Table 2.1** Notation differences between this book and the PRTools documentation

| Mathematical notation | Notation in pseudo-code | PRTools notation | Meaning |
|---|---|---|---|
| $T$ | `x, z` | `a, b` | *data set* |
| $n$ | `n` | `m` | *number of objects* |
| $N, D$ | `N, D` | `k, n` | *number of features, dimensions* |
| $K$ | `K` | `c` | *number of classes* |

integration with the pre-processing of data of any nature. This may certainly be facilitated by the large set of toolboxes available in MATLAB®.

PRTools is a MATLAB® toolbox designed by Robert P.W. Duin at first for pattern recognition research. The pattern recognition routines and support functions offered by PRTools represent a basic set covering largely the area of statistical pattern recognition. With the help of researchers in many areas, PRTools has updated to version 5 and can work well with the simultaneous use of the MATLAB® Statistical Toolbox Stats and integrates a number of its classifiers. In this book, except for additional notes, all examples are based on PRTools5.

PRTools has been used in many courses and PhD projects and received hundreds of citations. It is especially useful for researchers and engineers who need a complete package for prototyping recognition systems as it includes tools for representation. It offers most traditional and state-of-the-art off-the-shelf procedures for transformations and classification and evaluation. Thereby, it is well suited for comparative studies.

The notation used in PRTools manual documentation and code differs slightly from that used in the code throughout this book. In this chapter we try to follow the notation in the book. In Table 2.1 notation differences between this book and the PRTools documentation are given.

## 2.2 Essential Concepts

For the automatic recognition of the classes of objects, first some measurements have to be collected, for example using sensors,

then they have to be represented, for example in a feature space, and after some possible feature reduction steps they can be finally mapped by a classifier on the set of class labels. Between the initial representation in the feature space and this final mapping on the set of class labels the representation may be changed several times: simplified feature spaces (feature selection), normalization of features (e.g. by scaling), linear or non-linear mappings (feature extraction) and classification by a possible set of classifiers, combining classifiers and the final labelling. In each of these steps the data are transformed by some mapping. Based on this observation the following two basic concepts of PRTools are defined:

*Datasets*: matrices in which the rows represent the objects and the columns the features, class memberships or other fixed sets of properties (e.g. distances to a fixed set of other objects). In PRTools4 and the later version an extension of the dataset concept has been defined as *Datafiles*, which refer to datasets to be created from directories of files.

*Mappings*: transformations operating on datasets. As pattern recognition has two stages, *training* and *execution*, mappings have also two types, *untrained* and *trained*.

An *untrained mapping* refers just to the concept of a method, for example forward feature selection, PCA (refer to Chapter 7 of this book). It may have some parameters that are needed for training, for example the desired number of features or some regularization parameters. If an untrained mapping is applied to a dataset it will be trained (training).

A trained mapping is specific for the training set used to train the mapping. This dataset thereby determines the input dimensionality (e.g. the number of input features) as well as the output dimensionality (e.g. the number of output features or the number of classes). When a trained mapping is applied to a dataset it will transform the dataset according to its definition (execution).

In addition fixed mappings are used. They are almost identical to trained mappings, except that they do not result from a training step, but are directly defined by the user: for example the transformation of distances by a sigmoid function to the [0, 1] interval. PRTools deals with sets of *labelled* or *unlabelled*

*objects* and offers routines for the generalization of such sets into functions for *mapping* and *classification*. A classifier is thereby a special case of a mapping as it maps objects on class labels or on [0, 1] intervals that may be interpreted as *class memberships*, *soft labels* or *posterior probabilities*. An *object* is a $k$-dimensional vector of *feature values, distances, (dis)similarities* or *class memberships*. Within PRTools they are usually just called features. It is assumed that for all objects in a problem all values of the same set of features are given. The space defined by the actual set of features is called the feature space. Objects are represented as points or vectors in this space. New objects in a feature space are usually gradually converted to labels by a series of *mappings* followed by a final *classifier*.

Sets of *objects* may be given externally or may be generated by one of the data generation routines of PRTools. Their *labels* may also be given externally or may be the result of a *cluster analysis*. By these technique similar objects within a larger set are grouped (clustered). The similarity measure is defined by the cluster technique in combination with the object representation in the feature space. Some clustering procedures do not just generate labels but also a classifier that classifies new objects in the same way. A fundamental problem is to find a good *distance measure* that agrees with the dissimilarity of the objects represented by the feature vectors. Throughout PRTools the Euclidean distance is used as a default. However, scaling the features and transforming the feature spaces by different types of mappings effectively changes the distance measure.

The *dimensionality of the feature space* may be reduced by the selection of subsets of good features. Several strategies and criteria are possible for searching good subsets. *Feature selection* is important because it decreases the amount of features that have to be measured and processed. In addition to the improved computational speed in lower dimensional feature spaces there might also be an increase in the accuracy of the classification algorithms. Another way to *reduce the dimensionality* is to *map* the data on a linear or non-linear subspace. This is called linear or non-linear *feature extraction*. It does not necessarily reduce the number of features to be measured, but the advantage of an increased accuracy may still be gained. Moreover, as lower

dimensional representations yield less complex classifiers better generalizations can be obtained.

Using a *training set* a classifier can be trained such that it generalizes this set of examples of labelled objects into a *classification rule*. Such a classifier can be linear or non-linear and can be based on two different kinds of strategies. The first strategy minimizes the expected classification error by using estimates of the *probability density functions*. In the second strategy this error is minimized directly by *optimizing the classification function* of its performance over the learning set or a separate evaluation set. In this approach it has to be avoided because the classifier becomes entirely adapted to the training set, including its noise. This decreases its generalization capability. This '*overtraining*' can be circumvented by several types of *regularization*(often used in neural network training). Another technique is to simplify the classification function afterwards (e.g. the pruning of decision trees).

In PRTools4 and the later version the possibility of an automatic optimization has been introduced for parameters controlling the complexity or the regularization of the training procedures of mappings and classifiers. This is based on a *cross validation* (see below) over the training set and roughly increases the time needed for training by a factor of 100. Constructed classification functions may be evaluated by *independent test sets* of labelled objects. These objects have to be excluded from the training set, otherwise the evaluation becomes optimistically biased. If they are added to the training set, however, better classification functions can be expected. A solution to this dilemma is the use of *cross validation* and *rotation* methods by which a small fraction of objects is excluded from training and used for testing. This fraction is rotated over the available set of objects and results are averaged. The extreme case is the *leave-one-out* method for which the excluded fraction is as large as one object.

The performance of classification functions can be improved by the following methods:

1. A *reject* option in which the objects close to the decision boundary are not classified. They are rejected and might be classified by hand or by another classifier.

2. The selection or averaging of classifiers.
3. A multistage classifier for *combining* classification results of several other classifiers.

For all these methods it is profitable or necessary that a classifier yields some distance measure, confidence or posterior probability in addition to the hard, unambiguous assignment of labels.

## 2.3 PRTools Organization Structure and Implementation

PRTools makes use of the possibility offered by MATLAB® to define 'Classes' and 'Objects'. These programming concepts should not be confused with the *classes* and *objects* as defined in pattern recognition. The two main 'Classes' defined in PRTools are: dataset and mapping. As a child of dataset datafile has also been defined, inheriting most properties of dataset. A large number of operators (like * or []) and MATLAB® commands have been overloaded and have thereby a special meaning when applied to a dataset and/or a mapping.

The central data structure of PRTools is the dataset. It primarily consists of a set of objects represented by a matrix of feature vectors. Attached to this matrix is a set of labels, one for each object and a set of feature names, also called feature labels. Labels can be integer numbers or character strings. Moreover, a set of prior probabilities, one for each class, is stored. In most help files of PRTools, a dataset is denoted by A. In almost any routine this is one of the inputs. Almost all routines can handle multiclass object sets. It is possible that for some objects no label is specified (an NaN is used, or an empty string). Such objects are, unless otherwise mentioned, skipped during training. It is possible to define more than one set of labels in a dataset. For instance, when the objects are pixels in an image, then they may be labelled according to their image segment, but also according to the image, or to the sensor used, or the place the image has been measured.

Data structures of the 'Classes' mapping store data transformations ('mappings'), classifiers, feature extracting results, data

scaling definitions, non-linear projections, etc. They are usually denoted by W.

The easiest way to apply a mapping W to a dataset A is by A*W. The matrix multiplication symbol * is overloaded to this purpose. This operation may also be written as map (A, W). Like everywhere else in MATLAB®, concatenations of operations are possible, for example A*W1*W2*W3, and are executed from left to right.

In the beginning of the pattern recognition chain (see Figure 2.1), in the constitution of feature vectors, mappings can also be used. Raw data such as images may be stored on disk,

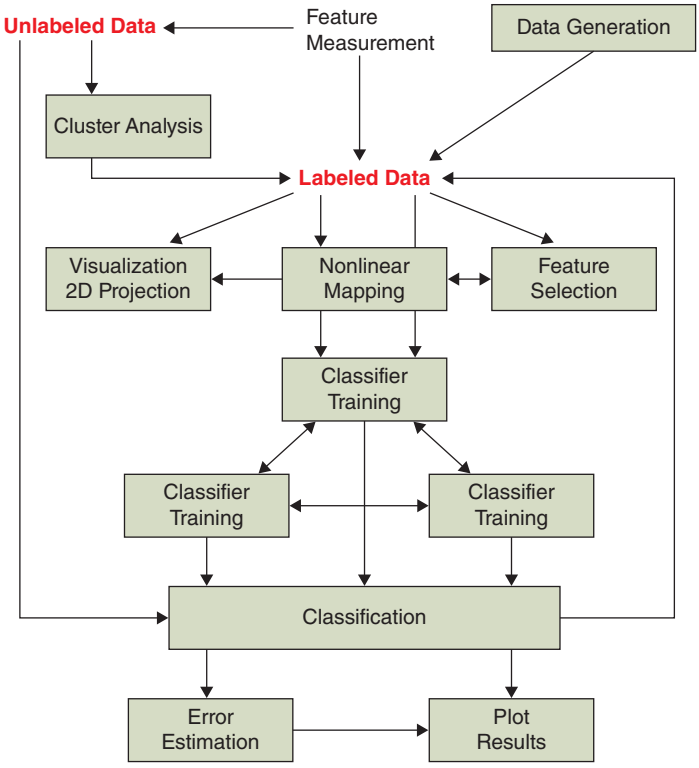


**Figure 2.1** The workflow of PRTools.

and by image processing and image analysis properties may be measured that can be used as features. The definition of raw data items is enabled in *PRTools* by the programming class called *datafile*. Datafiles are a type of a pre-stage of datasets. By mappings defining the proper pre-processing a dataset may be constructed. By following mappings, classifiers can be trained and applied, resulting in an overall recognition system.

Figure 2.1 is the workflow to use PRTools for pattern recognition problems. The total recognition workflow in PRTools terms consists of the following steps:

```
1. Collect raw data on disk
2. Define a datafile such as A pointing to the raw data
3. Define a mapping such as W_prepproc for an appropriate
   preprocessing and analyzing the datafile
4. Apply the mapping to the datafile, resulting in a dataset,
   B = A*W_preproc
5. Define a suited conversion of the feature space, e.g. by PCA:
   W_featred
6. Apply this mapping on B : C = B*W_featred
7. Train a classifier in this space: W_classf
8. Apply the dataset to this classifier:
   labels = C*W_classf
          = B*W_featred*W_classf
          = A*W_preproc*W_featred*W_classf.
```

As the mappings `W_preproc`, `W_featred` and `W_classf` are stored in variables and as the concatenations of a sequence of mappings is defined in PRTools, the entire recognition system can be stored in a single variable: `W_recsys=W_preproc*W_featred*W_classf`. New objects, for example images stored on disk as a datafile A, can now be classified by `labels=A*W_recsys`.

In this example three mappings have to be specified by the user. The first, `W_preproc`, is usually entirely based on the background knowledge of the user of the type of images he or she wants to classify. The other two, the feature reduction and the classifier, have to be derived from data based on an optimization of a cost function or an estimation of parameters given a model assumption. In pattern recognition terms, these mappings are thereby the result from training. Datasets are needed for this, based on the same pre-processing and representation of the data to be classified later. There are many routines in PRTools available for training mappings and classifiers. It is in fact the core of the toolbox.

Consequently, we distinguish two sets of objects: a training set with given labels (class memberships) to be used for designing the system and an unlabelled set for which the class memberships have to be found. The first step of the program is the definition of these sets such that they can be handled by PRTools. Let us assume that the raw data has been stored in two directories, 'directory_1' and 'directory_2':

```
A_labeled = datafile('directory_1');
A_unlabeled = datafile('directory_2');
```

It will be described later how the labels of A_labeled have to be supplied and how they are stored. The first mapping has to define features for objects. A simple command is the use of histograms, which can be specified by the following mapping:

```
W_preproc = histm([],[1:256]);
```

The pre-processing of the two datafiles and their conversion to datasets is performed by

```
B_labeled = dataset(A_labeled*W_preproc);
B_unlabeled = dataset(A_unlabeled*W_preproc);
```

Let us assume that a feature reduction by PCA is demanded to five features. It has to be derived from the pre-processed data, of course:

```
W_featred = pca(B_labeled,5);
```

Suppose that finally the Fisher classifier is used. It has to be found in the reduced feature space:

```
W_classf == fisherc(B_labeled*W_preproc*W_featred);
```

The labels for B_unlabeled can now be estimated by

```
labels = B_unlabeled*W_preproc*W_featred*W_classf*labeld;
```

in which `labeld` is a standard PRTools mapping that maps classifier outcomes to labels. The classification system can also be

stored in a single variable `W_class_sys`:

```
W_class_sys = W_preproc*W_featred*W_classf*labeld;
labels = B_unlabeled*W_class_sys;
```

## 2.4 Some Details about PRTools

### 2.4.1 Datasets

Datasets in PRTools are in the MATLAB® language defined as objects of the class DATASET. Below, the words 'object' and 'class' are used in the pattern recognition sense. A dataset is a set consisting of M objects, each described by K features. In PRTools, such a dataset is represented by an M × K matrix: M rows, each containing an object vector of K elements. Usually, a dataset is labelled. An example of a definition is

```
DATA = [RAND(3,2) ; RAND(3,2)+0.5];
LABS = ['A';'A';'A';'B';'B';'B'];
A = DATASET(DATA,LABS);
```

which defines a [6 × 2] dataset with 2 classes. The [6 × 2] data matrix (6 objects given by 2 features) is accompanied by labels, assigning each of the objects to one of the two classes A and B. Class labels can be numbers or strings and should always be given as rows in the label list. A label may also have the value NaN or may be an empty string, indicating an unlabelled object. If the label list is not given, all objects are marked as unlabelled. Various other types of information can be stored in a dataset. The simplest way to get an overview is by typing:

```
STRUCT(A)
which for the above example displays the following:
DATA: [6x2 double]
LABLIST: {2x4 cell}
NLAB: [6x1 double]
LABTYPE: 'crisp'
TARGETS: []
FEATLAB: [2x1 double]
FEATDOM: {1x2 cell }
PRIOR: []
COST: []
OBJSIZE: 6
FEATSIZE: 2
IDENT: [6x1 struct]
VERSION: {[1x1 struct] '21-Jul-2007 15:16:57'}
NAME: []
USER: []
```

These fields have the following meaning:

| Filed name | Description |
|---|---|
| DATA | An array containing the objects (the rows) represented by features (the columns). In the software and help-files, the number of objects is usually denoted by M and the number of features is denoted by K. Therefore, DATA has the size of [M,K]. This is also defined as the size of the entire dataset. |
| LABLIST | The names of the classes, can be strings stored in a character array. If they are numeric they are stored in a column vector. Mixtures of these are not supported. The LABLIST field is a structure in which more than a single label list and the corresponding priors and costs are stored. PRTools automatically keeps track of this. |
| NLAB | An [M x 1] vector of integers between 1 and C, defining for each of the M objects its class. |
| LABTYPE | 'CRISP', 'SOFT' or 'TARGETS' are the three possible label types. In case of 'CRISP' labels, a unique class, defined by NLAB, is assigned to each object, pointing to the class names given in LABLIST. For 'SOFT' labels, each object has a corresponding vector of C numbers between 0 and 1 indicating its membership (or confidence or posterior probability) of each of the C classes. These numbers are stored in TARGETS of the size M x C. They do not necessarily sum to one for individual row vectors. Labels of type 'TARGETS' are in fact no labels, but merely target vectors of length C. The values are again stored in TARGETS and are not restricted in value. |
| TARGETS | [M, C] array storing the values of the soft labels or targets. |
| FEATLAB | A label list (like LABLIST) of K rows storing the names of the features. |

(*continued*)

| Filed name | Description |
|---|---|
| FEATDOM | A cell array describing for each feature its domain. |
| PRIOR | Vector of length C storing the class prior probabilities. They should sum to one. If PRIOR is empty ([]) it is assumed that the class prior probabilities correspond to the class frequencies. |
| COST | Classification cost matrix. COST(I,J) are the costs of classifying an object from class I as class J. Column C+1 generates an alternative reject class and may be omitted, yielding a size of [C,C]. An empty cost matrix, COST=[] (default) is interpreted as COST=ONES(C)-EYE(C)(identical costs of misclassification). |
| OBJSIZE | The number of objects, M. In case the objects are related to an n-dimensional structure, OBJSIZE is a vector of length n, storing the size of this structure. For instance, if the objects are pixels in a [20 x 16] image, then OBJSIZE = [20, 16] and M = 320. |
| FEATSIZE | The number of features, K. In case the features are related to an n-dimensional structure, FEATSIZE is a vector of length n, storing the size of this structure. For instance, if the features are pixels in a [20 x 16] image, then FEATSIZE = [20, 16] and K = 320. |
| IDENT | A structure array of M elements storing user defined fields giving additional information on each of the objects. |
| VERSION | Some information related to the version of PRTools used for defining the dataset. |
| NAME | A character string naming the dataset, possibly used to annotate related graphics. |
| USER | A structure with user defined fields not used by PRTools. |

The fields can be set in the following ways:

1. In the DATASET construction command after DATA and LABELS using the form of {field name, value pairs}, for example

   A = DATASET (DATA, LABELS,'PRIOR', [0.4 0.6], 'FEATLIST',['AA';'BB']);

   Note that the elements in PRIOR refer to classes as they are ordered in LABLIST.
2. For a given dataset A, the fields may be changed similarly by the SET command:

   A = SET (A,'PRIOR',[0.4 0.6],'FEATLIST',['AA';'BB']);

3. By the commands

   SETDATA, SETFEATLAB, SETFEATDOM, SETFEATSIZE, SETIDENT, SETLABELS, SETLABLIST, SETLABTYPE, SETNAME, SETNLAB, SETOBJSIZE, SETPRIOR, SETTARGETS, SETUSER.

4. By using the dot extension as for structures, for example

   A.PRIOR = [0.4 0.6];
   A.FEATLIST = ['AA';'BB'];

   Note that there is no field LABELS in the DATASET definition. Labels are converted to NLAB and LABLIST. Commands like SETLABELS and A.LABELS, however, exist and take care of the conversion.

The data and information stored in a dataset can be retrieved as follows:

1. By DOUBLE(A) and by +A, the content of the A.DATA is returned.

   [N,LABLIST] = CLASSSIZES(A);

   It returns the numbers of objects per class and the class names stored in LABLIST. By DISPLAY(A), it writes the size of the dataset, the number of classes and the label type on the terminal screen. By SIZE(A), it returns the size of A.DATA: numbers of objects and features. By SCATTERD(A), it makes a

scatter plot of a dataset. By SHOW(A), it may be used to display images that are stored as features or as objects in a dataset.

2. By the GET command, for example

   [PRIOR, FEATLIST] = GET(A,'PRIOR','FEATLIST');

3. By the commands:

   GETDATA, GETFEATLAB, GETFEATSIZE, GETIDENT, GETLABELS, GETLABLIST, GETLABTYPE, GETNAME, GETNLAB, GETOBJSIZE, GETPRIOR, GETCOST, GETSIZE, GETTARGETS, GETTARGETS, GETUSER, GETVERSION.

   Note that GETSIZE(A) does not refer to a single field, but it returns [M,K,C]. The following commands do not return the data itself, instead they return indices to objects that have specific identifiers, labels or class indices:

   FINDIDENT, FINDLABELS, FINDNLAB.

4. Using the dot extension as for structures, for example

   PRIOR = A.PRIOR;
   FEATLIST = A.FEATLIST;

   Many standard MATLAB® operations and a number of general MATLAB® commands have been overloaded for variables of the DATASET type.

### 2.4.2 Datafiles

Datafiles are constructed to solve the memory problem connected with datasets. The latter are always in the core and their size is thereby restricted to the size of the computer memory. As in processing datasets copies are often (temporarily) created, it is in practice advisable to keep datasets under 10 million elements (objectsize × featuresize). A number of operations handle datasets sequentially or can be written like that, for example fixed mappings, testing and the training of some simple classifiers. Thus there is no problem to have such data stored on disk