H.B. Mitchell

Multi-Sensor Data Fusion

AN INTRODUCTION

Springer

Multi-Sensor Data Fusion

H.B. Mitchell

Multi-Sensor Data Fusion

An Introduction

With 81 Figures and 59 Tables

Springer

Dr. H.B. Mitchell
homepage: www.hbmitchell.com

Library of Congress Control Number: 2007926176

ISBN 978-3-540-71463-7 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Typesetting: by the author and Integra, India using a Springer LaTeX macro package
Cover design: Erich Kirchner, Heidelberg

Printed on acid-free paper SPIN: 11911104 5 4 3 2 1 0

This book is dedicated to the memory of my parents

WOOLF AND BELLA CISSIE MITCHELL

They surrounded me with love and gently encouraged me in my studies. They were never discouraged nor lost faith in me or my ability. This book is a small token of my love for them.

May their memory be a blessing

Preface

The purpose of this book is to provide an introduction to the theories and techniques of multi-sensor data fusion. The book has been designed as a text for a one-semester graduate course in multi-sensor data fusion. It should also be useful to advanced undergraduates in electrical engineering or computer science who are studying data fusion for the first time and to practising engineers who wish to apply the concepts of data fusion to practical applications.

The book is intended to be largely self-contained in so far as the subject of multi-sensor data fusion is concerned, although some prior exposure to the subject may be helpful to the reader. A clear understanding of multi-sensor data fusion can only be achieved with the use of a certain minimum level of mathematics. It is therefore assumed that the reader has a reasonable working knowledge of the basic tools of linear algebra, calculus and simple probability theory. More specific results and techniques which are required are explained in the body of the book or in appendices which are appended to the end of the book.

Although conceptually simple, the study of multi-sensor data fusion presents challenges that are fairly unique within the education of the electrical engineer or computer scientist. Unlike other areas encounted by a student of these subjects, multi-sensor data fusion draws on, and brings together, theories and techniques that are typically taught separately in the traditional curricula. To become competent in the field the student must be familiar with tools taken from a wide range of diverse subjects, including: neural networks, signal processing, statistical estimation, tracking algorithms, computer vision, and control theory. All too often the student views multi-sensor data fusion as a miscellaneous assortment of processes and techniques which bear no relationship to each other. We have attempted to overcome this problem by presenting the material using a common statistical Bayesian framework. In this way the different theories and techniques are clearly integrated and the underlying pattern of relationships that exist between the different methodologies are emphasized. Furthermore, by adopting a single framework, we have kept the book at a reasonable size while treating many new and important

topics in great depth. We should point out that we have not, however, ignored other frameworks when this seemed appropriate.

As with any other branch of engineering, multi-sensor data fusion is a pragmatic activity which is driven by practicalities. It is therefore important that the student is able to experiment with the different techniques presented in the book. For this purpose software code, written in Matlab, is particularly convenient and we have included details of relevant Matlab code which may be downloaded from the worldwide web. For the professional engineer we have both illustrated the theory with many real-life applications and have provided him with an extensive list of up-to-date references. Additional information, including material for course instructors, is available from the author's homepage: www.hbmitchell.com.

The book is based on seminars, lectures and courses on multi-sensor data fusion which have been taught over several years. The structure and content of the book is based on material gathered and ideas exchanged with my colleagues. Particular thanks are extended to Dr. Paul Schaefer and Mr. Michael Avraham with whom I have discussed most topics in the book and to Ms. Ruth Rotman, Prof. Brian Cowan and Prof. Stanley Rotman who have kindly read and commented on the various drafts of the book. I am also indebted to my wife and children for the support and patience they have shown me while the book was being written.

Finally, to the reader. We hope you will enjoy reading this book and that it will prove to be an useful addition to the increasingly important and expanding field of multi-sensor data fusion.

H.B. Mitchell

Contents

Part I

Basics

1

Introduction

1.1 Definition

The subject of this book is *multi-sensor data fusion* which we define as "the theory, techniques and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format". In performing sensor fusion our aim is to improve the quality of the information, so that it is, in some sense, *better* than would be possible if the data sources were used individually.

The above definition implies that the sensor data, or the data derived from the sensory data, consists of multiple measurements which have to be combined. The multiple measurements may, of course, be produced by multiple sensors. However, the definition also includes multiple measurements, produced at different time instants, by a single sensor.

The general concept of multi-sensor data fusion is analogous to the manner in which humans and animals use a combination of multiple senses, experience and the ability to reason to improve their chances of survival.

The basic problem of multi-sensor data fusion is one of determining the best procedure for combining the multi-sensor data inputs. The view adopted in this book is that combining multiple sources of information with *a priori* information is best handled within a statistical framework. The main advantage of a statistical approach is that explicit probabilistic models are employed to describe the various relationships between sensors and sources of information taking into account the underlying uncertainties. In particular we restrict ourselves to the Bayesian methodology which provides us with a useful way to formulate the multi-sensor data fusion problem in mathematical terms and which yields an assessment of the uncertainty in all unknowns in the problem.

1.2 Synergy

The principal motivation for multi-sensor data fusion is to improve the quality of the information output in a process known as *synergy*. Strictly speaking, synergy does not require the use of multiple sensors. The reason being that the synergistic effect may be obtained on a temporal sequence of data generated by a single sensor. However, employing more than one sensor may enhance the synergistic effect in several ways, including: increased spatial and temporal coverage, increased robustness to sensor and algorithmic failures, better noise suppression and increased estimation accuracy.

Example 1.1. Multi-Modal Biometric Systems [226]. Biometric systems that rely on a *single* biometric trait for recognition are often characterized by high error rates. This is due to the lack of completeness or *universality*[1] in most biometric traits. For example, fingerprints are not truly universal since it is not possible to obtain a good quality fingerprint from people with hand-related disabilities, manual workers with many cuts and bruises on their fingertips or people with very oily or very dry fingers. Multi-modal biometric sensor systems solve the problem of non-universality by fusing the evidence obtained from multiple traits. □

Example 1.2. Multiple Camera Surveillance Systems [142]. The increasing demand for security by society has led to a growing need for surveillance activities in many environments. For example, the surveillance of a wide-area urban site may be provided by periodically scanning the area with a single narrow field-of-view camera. The temporal coverage is, however, limited by the time required for the camera to execute one scan. By using multiple cameras we reduce the mean time between scans and thereby increase the temporal coverage. □

Broadly speaking, multi-sensor data fusion may improve the performance of the system in four different ways [18]:

Representation. The information obtained during, or at the end, of the fusion process has an abstract level, or a granuality, higher than each input data set. The new abstract level or the new granuality provides a richer semantic on the data than each initial source of information

Certainty. If V is the sensor data before fusion and $p(V)$ is the *a priori* probability of the data before fusion, then the gain in certainty is the growth in $p(V)$ after fusion. If V_F denotes data after fusion, then we expect $p(V_F) > p(V)$.

Accuracy. The standard deviation on the data after the fusion process is smaller than the standard deviation provided directly by the sources. If data is noisy or erroneous, the fusion process tries to reduce or eliminate

[1] A biometric trait is said to be universal if every individual in the target population is able to present the trait for recognition.

noise and errors. In general, the gain in accuracy and the gain in certainty are correlated.

Completeness. Bringing new information to the current knowledge on an environment allows a more complete the view on this environment. In general, if the information is redundant and concordant, we could also have a gain in accuracy

Example 1.3. Multi-Modal Medical Imaging: Gain in Completeness. We consider images obtained through Magnetic Resonance Imaging (MRI), Computed Tomography (CT) and Positron Emission Tomography (PET). The multi-sensor data fusion of all three images allows a surgeon to view "soft tissue" information (MRI) in the context of "bone" (CT) and in the context of "functional" or "physiological information" (PET). □

1.3 Multi-Sensor Data Fusion Strategies

As the above examples show, multi-sensor data fusion is a wide-ranging subject with many different facets. In order to understand it better, and to become familiar with its terminology, we shall consider it from three different points of view as suggested by Boudjemaa and Forbes [27], Durrant-Whyte [69] and Dasarathy [56]. Other points of view will be developed in succeeding chapters of the book.

1.3.1 Fusion Type

Boudjemaa and Forbes [27] classify a multi-sensor data fusion system according to what aspect of the system is fused:

Fusion across sensors. In this situation, a number of sensors nominally measure the same property, as, for example, a number of temperature sensors measuring the temperature of an object.

Fusion across attributes. In this situation, a number of sensors measure different quantities associated with the same experimental situation, as, for example, in the measurement of air temperature, pressure and humidity to determine air refractive index.

Fusion across domains. In this situation, a number of sensors measure the same attribute over a number of different ranges or domains. This arises, for example, in the definition of a temperature scale.

Fusion across time. In this situation, current measurements are fused with historical information, for example, from an earlier calibration. Often the current information is not sufficient to determine the system accurately and historical information has to be incorporated to determine the system accurately.

Example 1.4. Flood Forecasting [302]. Water companies are under constant pressure to reduce the frequency of combined sewer overflows to natural water courses from urban drainage systems (UDS). The management of storm-water through the UDS and similar applications require accurate real-time estimates of the rainfall. One way water companies have done this is to fuse measurements of the rainfall made by ground-based rain gauges and a weather radar system. This is "fusion across sensors" since, nominally, the two sensors measure the same property. □

1.3.2 Sensor Configuration

Durrant-Whyte [69] classifies a multi-sensor data fusion system according to its sensor configuration. There are three basic configurations:

Complementary. A sensor configuration is called complementary if the sensors do not directly depend on each other, but can be combined in order to give a more complete image of the phenomenom under observation. Complementary sensors help resolve the problem of *incompleteness*.

Competitive. A sensor configuration is competitive if each sensor delivers an independent measurement of the same property. The aim of competitive fusion is to reduce the effects of *uncertain* and *erroneous* measurements.

Cooperative. A cooperative sensor configuration network uses the information provided by two, or more, independent sensors to derive information that would not be available from the single sensors.

Example 1.5. Triangulation: Cooperative Fusion [272]. The location (x, y) of an object O is found by cooperatively fusing two bearings, θ_1 and θ_2, as measured by the direction-finding sensors, S_1 and S_2. Let (X_1, Y_1) and (X_2, Y_2) denote the locations of the two sensors S_1 and S_2, then by simple geometry (see Fig. 1.1), we find the location (x, y) of O, where

$$x = \frac{Y_2 - Y_1 + (X_1 \tan\theta_1 - X_2 \tan\theta_2)}{\tan\theta_1 - \tan\theta_2} ,$$
$$y = \frac{Y_2 \tan\theta_1 - Y_1 \tan\theta_2 + \tan\theta_1 \tan\theta_2 (X_1 - X_2)}{\tan\theta_1 - \tan\theta_2} .$$

Note: See Ex. 4.2 for a Bayesian analysis of this problem. □

1.3.3 Input/Output Characteristics

Dasarathy (1994) [56] classifies a multi-sensor data fusion system according to its joint input/output characteristics. Table 1.1 illustrates the types of inputs/outputs considered in this scheme.

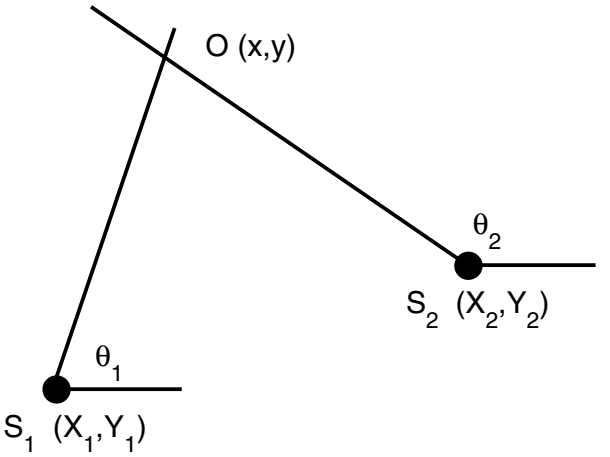

Fig. 1.1. Shows the calculation of the location (x, y) of the object O by triangulation of the angles θ_1 and θ_2.

Table 1.1. Dasarathy's Input/Output Data Fusion Model.

Symbol	Name	Description/Example
DaI-DaO	Data Input/Data Output	Input data is smoothed/filtered.
DaI-FeO	Data Input/Feature Output	Features are generated from the input data e.g. edge detection in an image.
FeI-FeO	Feature Input/Feature Output	Input features are reduced in number or new features are generated by fusing input features.
FeI-DeO	Feature Input/Decision Output	Input features are fused together to give output decision.
DeI-DeO	Decision Input/Decision Output	Multiple input decisions are fused together to give a final output decision.

It is sometimes useful to divide the DeI-DeO fusion model into two sub-models: a "soft" decision-input model (denoted as DsI-DeO) in which each input decision is accompanied by a degree-of-support value and a "hard" decision-input model (denoted as DhI-DeO) in which the input decisions are not accompanied by any degree-of-support values.

1.4 Formal Framework

Multi-sensor data fusion systems are often of a size and a complexity that requires the use of a formal framework [234] around which we may organize our knowledge. The framework adopted in this book is that of a distributed network of autonomous modules, in which each module represents a separate function in the data fusion system.

Apart from providing us with a structured framework, the modular design decreases the complexity involved in designing a data fusion system by compartmentalizing the design problem. Modularity also helps in the construction and maintenance of an operational multi-sensor data fusion system.

In analyzing a multi-sensor data fusion system it is useful to divide the system into three parts: the physical, information and cognitive domains and to determine the flow of data between these parts [72, 73, 76, 234] (Fig. 1.2).

Physical Domain: Hardware. The physical domain contains the sensor modules each of which represents a sensor which physically interacts with the external environment. Each module contains a *sensor model* which provides us with a description of the measurements made by the sensor and of the local environment within which the sensor operates. In some applications we may wish to physically change the external environment. In this case the physical domain will also contain actuators which are able to modify the external environment.

Information Domain: Software. The information domain constitutes the heart of a multi-sensor data fusion system. It contains three blocks which are responsible for data fusion, control application/resource management and human-machine interface (HMI). The data fusion block is constructed as an autonomous network of "fusion" modules. This network is responsible for combining all the sensor data into a unified view of the environment in the form of an "environmental image". The control application/resource management block is constructed as autonomous networks of "control"

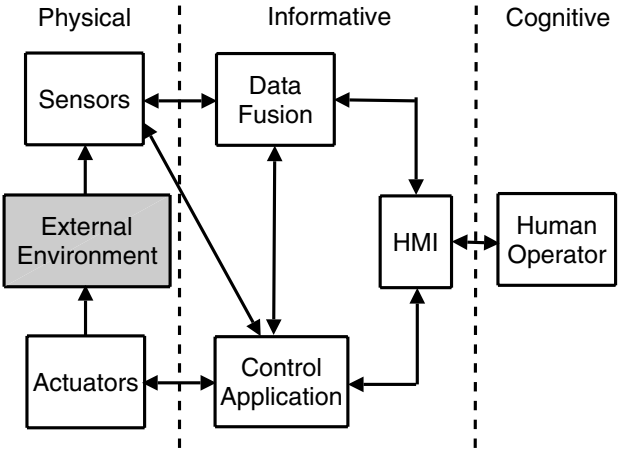

Fig. 1.2. Shows the division of a generic multi-sensor data fusion system into three parts or domains: physical, informative and cognitive and the flow between the parts. The figure is divided into three panels which correspond to the physical domain (leftmost panel), information domain (middle panel) and cognitive domain (rightmost panel).

modules. This network is responsible for all decisions which are made on the basis of the environmental image. In many applications the decisions are fed back to the sensor block. In this case the process is known as "sensor management".

Cognitive Domain: Human User. In many multi-sensor data fusion applications the human user is the final arbiter or decision maker. In this case it is important to design the system in such a way that all the information which is transmitted to the human user is transformed into a form which is intuitively usable by the user for his decision-making process.

Example 1.6. Control Loop [72, 73, 76]. Fig. 1.3 shows a control loop containing four blocks: sensor, actuator, data fusion and control application. The environment is observed with one or more sensors. The corresponding sensor observations are then passed to the data fusion block where they are combined to form a unified view of the environment ("environmental image"). The environmental image is, in turn, passed to the control application block. The loop is closed by feeding the decisions made by the control application block back to the environment. □

1.4.1 Multi-Sensor Integration

In the data fusion block only a few of the autonomous modules perform "multi-sensor data fusion" as defined in Sect. 1.1, the remaining modules perform auxiliary functions. The modules which perform the data fusion will receive input data from the physical sensors $S_1, S_2, \ldots, S_N$ and from other

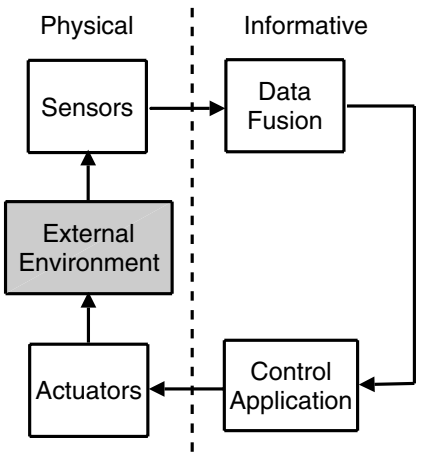

Fig. 1.3. Shows a simple control loop built from four blocks: sensors, actuators, data fusion and control application. A feedback mechanism is included by allowing the control application to act on the external environment via an actuator block.

modules. In this case, we sometimes find it convenient to differentiate between multi-sensor data fusion performed by the individual modules and multi-sensor data integration performed by the entire data fusion block [187] (see Fig. 1.4).

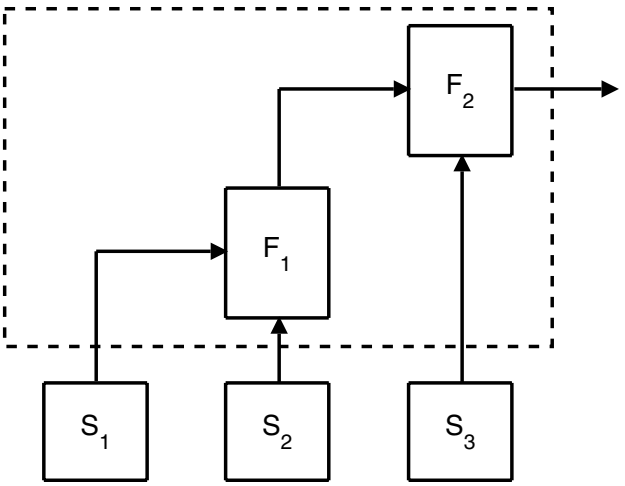

Fig. 1.4. Shows two multi-sensor data fusion blocks F_1 and F_2. F_1 performs data fusion on the output of sensors S_1 and S_2 and F_2 performs data fusion on the output of F_1 and the sensor S_3. Together F_1 and F_2 perform "multi-sensor integration".

1.5 Catastrophic Fusion

[2] The unsuspecting reader may conclude, on the basis of what has been presented so far, that a multi-sensor fusion system is *always* better than a single sensor system. This conclusion is, however, mistaken: Often the performance of a multi-sensor data fusion system is below that of the individual sensors. This phenomenom is known as *catastrophic fusion* and clearly should be avoided at all times.

Formally, catastrophic fusion [218] is said to occur when the performance of a multi-sensor data fusion system F is significantly lower than the performance of one, or more, of the individual sensors S_m. In general, each sensor $S_m, m \in \{1, 2, \ldots, M\}$, is designed to operate correctly only under specific conditions, or environment, C_m. Let C_F denote the corresponding condition

[2] This section contains advanced material. It assumes the reader has some familiarity with Bayesian statistics (see Chapts. 8-10). The section is not, however, essential for what follows and it may be left for a second reading.

for the correct operation of all the sensors in system F, then C_F is the "intersection" of the C_m which we may write symbolically as

$$C_F = C_1 \wedge C_2 \wedge \ldots C_M \ . \tag{1.1}$$

Sometimes, however, F is used in an environment C which is inconsistent with one of the C_m, say C_{m^*}. When this happens, the signal from the corresponding sensor, S_{m^*}, may dominant the fused output with catastrophic results. To prevent this happening, multi-sensor fusion systems often employ *secondary classifiers* which monitor the performance of each sensor S_m.

Example 1.7. Automatic Speech Recognition: Preventing Catastrophic Fusion [218]. In ideal, or clean, conditions automatic speech recognition systems perform very well using a single audio sensor S_A. However, we often observe a substantial reduction in performance when background noise, channel distortion or reverberation are present. This has led to the development of automatic speech recognition systems which employ an audio sensor S_A *and* a visual sensor S_V (see Ex. 3.11). The two sensors are *complementary*: speech characteristics that are visually confusable are often acoustically distinct, and characteristics that are acoustically confusable are often visually distinct. The audio-visual system work as follows. Given a finite set of utterances $U_i, i \in \{1, 2, \ldots, N\}$, we identify an unknown utterance, U, as

$$U^* = \arg\max_i \big(P(U_i|y_A, y_V)\big) \ , \tag{1.2}$$

where $P(U_i|y_A, y_V)$ is the conditional probability of U_i given an audio-visual observation (y_A, y_V). To a first approximation, the audio and visual features are conditionally independent. In this case, we may write $P(U_i|y_A, y_V)$ as

$$P(U_i|y_A, y_V) \propto P(U_i|y_A) \times P(U_i|y_V) \ , \tag{1.3}$$

where the conditional probabilities $P(U_i|y_A)$ and $P(U_i|y_V)$ are calculated offline using a set of training samples D.

If the set of training samples and test samples are well matched, the solution represented by (1.2) and (1.3) is theoretically optimal: we automatically compensate for any noise or distortion and assign more importance to the classification which is more "certain". The underlying assumption is, however, that the conditional probabilities $P(U_i|y_A)$ and $P(U_i|y_V)$ generated during training, match the speech data that is being tested. When the speech data is contaminated with noise this assumption is no longer valid and the probability estimates are incorrect. In such cases, it is common to use a weighted integration scheme, where the influence of the noisy channel is attenuated. This weighting can be a function of the signal-to-noise ratio (SNR) in each channel, and can be implemented as follows:

$$P(U_i|y_A, y_V) \propto P^{\alpha_A}(U_i|y_A) P^{\alpha_V}(U_i|y_V) \ ,$$

where

$$\alpha_A + \alpha_V = 1 \, .$$

The purpose of the weights $\alpha_m, m \in \{A, V\}$, is to "neutralize" a sensor S_m, whenever the secondary classifiers determine that S_m is operating outside its specified operating conditions C_m. In the worst case, when environmental conditions make the sensor completely unreliable we set the corresponding weight to zero. □

1.6 Organization

Although we shall discuss the physical, information and cognitive domains in a multi-sensor data fusion system, the emphasis will be on the data fusion block. The book is organized into five parts as follows.

Part I: Introduction. This consists of Chapts. 1-3. Chapt. 1 is a general introduction to the subject of multi-sensor data fusion. In this chapter we provide an overview of the basic concepts used in multi-sensor data fusion. In Chapt. 2 we consider the sensors, which ultimately, are the source of all input data into a multi-sensor data fusion system. In Chapt. 3 we consider the different system architectures which may be employed in constructing a data fusion system.

Part II: Common Representational Format. This consists of Chapts. 4-7. In Chapt. 4 we provide an overview of the basic concept of a common representational format and the different techniques used to create such a representation. The techniques may be broadly divided into three different types which are then discussed in turn in Chapts. 5, 6 and 7.

Part III: Data Fusion. This consists of Chapts. 8-13. In Chapt. 8 we give an overview of the Bayesian approach to multi-sensor data fusion and the different techniques involved. This is followed by Chapts. 9-11 in which we deal with parameter estimation theory including sequential estimation theory. In Chapts. 12-13 we consider decision fusion.

Part IV. Sensor Management. This consists of a single chapter (Chapt. 14) in which we consider sensor management. Specifically we consider how the decisions made by the data fusion block may, if required, be fed back to the sensors.

Part V. Appendices. This consists of Appendices A and B. Appendix A is a list of relevant software written in matlab which is available on the world wide web. Sufficient information is provided in the table so that the software may be easily found on the world wide web using a simple search machine [3]. Appendix B is a summary of elementary results in probability theory, linear algebra and matrix theory with which the reader should be familiar.

[3] The internet addresses themselves are not given for the simple reason that internet adresses tend to have a very short timelife.

1.7 Further Reading

General overviews on multi-sensor data fusion are [3, 6, 7, 108, 109, 131, 154, 187, 318]. For an extended discussion regarding the issues involved in defining multi-sensor data fusion and related terms, see [316, 236].

2

Sensors

2.1 Introduction

Sensors are devices which interact *directly* with the environment and which are ultimately the source of all the input data in a multi-sensor data fusion system [87]. The physical element which interacts with the environment is known as the *sensor element* and may be any device which is capable of perceiving a physical property, or environmental attribute, such as heat, light, sound, pressure, magnetism or motion. To be useful, the sensor must map the value of the property or attribute to a quantitative measurement in a *consistent* and *predictable* manner.

In Chapt. 1 we introduced a formal framework in which we represented a sensor fusion system as a distributed system of autonomous modules. To support such a scheme, the sensors must not only measure a physical property, but must also perform additional functions. These functions can be described in terms of compensation, information processing, communication and integration:

Compensation. This refers to the ability of a sensor to detect and respond to changes in the environment through self-diagnostic tests, self-calibration and adaption.

Information Processing. This refers to processes such as signal conditioning, data reduction, event detection and decision-making, which enhance the information content of the raw sensor measurements.

Communications. This refers to the use of a standardized interface and a standardized communication protocol for the transmission of information between the sensor and the outside world.

Integration. This refers to the coupling of the sensing and computation processes on the same silicon chip. Often this is implemented using microelectro-mechanical systems (MEMS) technology.

A practical implementation of such a sensor is known as a smart, or intelligent, sensor [77].

2.2 Smart Sensor

A *smart sensor* is a hardware/software device that comprises in a compact small unit a sensor element, a micro-controller, a communication controller and the associated software for signal conditioning, calibration, diagnostic tests and communication. The smart sensor transforms the raw sensor signal to a standardized digital representation, checks and calibrates the signal, and transmits this digital signal to the outside world via a standardized interface using a standardized communication protocol.

Fig. 2.1 shows the measurement of a physical property by a smart sensor.

The transfer of information between a smart sensor and the outside world is achieved by reading (writing) the information from (to) an interface-file system (IFS) which is encapsulated in the smart sensor as shown in Fig. 2.2 [75].

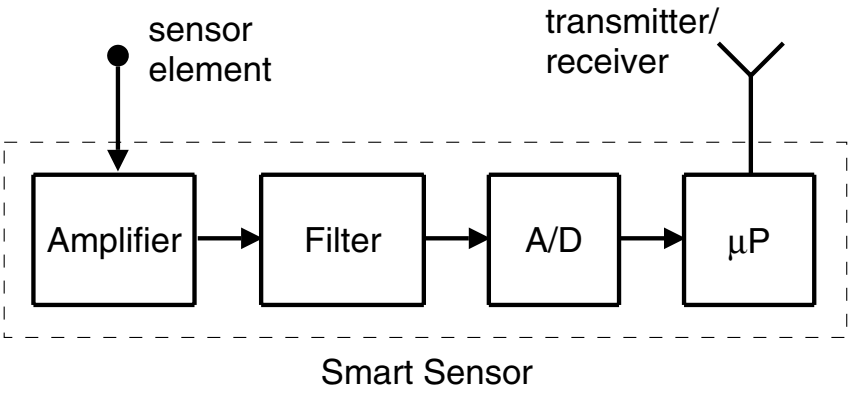

Fig. 2.1. The sensor element measures the physical property and outputs an analog signal which is amplified, filtered and then converted to a digital signal by the analog-to-digital, or A/D, unit. The digital signal is processed by the microprocessor, μP, where it is temporally stored before being transmitted by the transmitter/receiver. *Note*: The smart sensor may also receive command instructions via the transmitter/receiver unit.

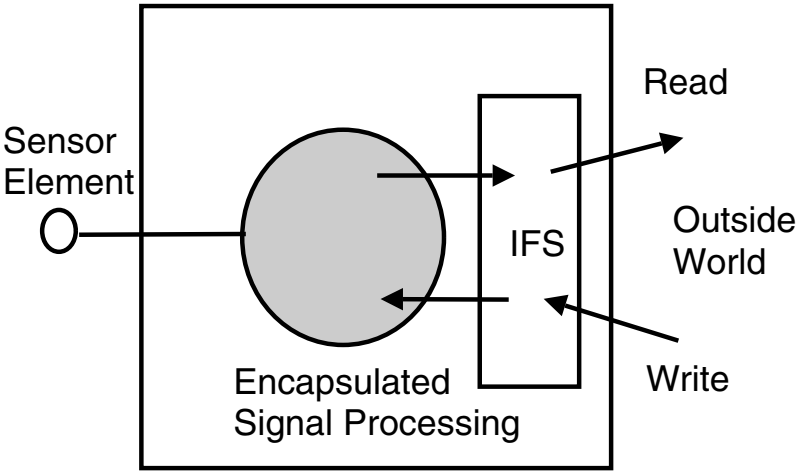

Fig. 2.2. Shows a smart sensor with a sensor element and the encapsulated signal processing functions and the encapsulated Interface file system (IFS).

2.3 Logical Sensors

A *logical sensor* [116] is defined as any device which functions as a source of information for a multi-sensor data fusion node. Thus a logical sensor encompass both physical sensors and any fusion node whose output is subsequently fed into another fusion node. Unless stated otherwise, from now on the term "sensor" will refer to a logical sensor or a source-of-information.

2.4 Interface File System (IFS)

The IFS provides a structured (name) space which is used for communicating information between a smart sensor and the outside world [74]. The following example neatly illustrates the concept of an IFS.

Example 2.1. Brake Pedal in a Car: An Interface File System [160]. We consider a driver in a car. For the purposes of braking, the *brake pedal* acts as an IFS between the driver and the brakes. At this interface there are two relevant variables. The first variable is P: the pressure applied to the brake pedal by the driver and the second variable is R: the resistance provided by the brakes back to the driver. The relative position of the brake pedal uniquely identifies the values of P and R to both the driver and the brakes. The temporal association between sensing the information (e. g. by the driver) and receiving the information (e. g. by the brakes) is implicit, because of the mechanical connection between the driver and the brakes. □

A record of the IFS can be accessed, both from the sensor and from the outside world (Fig. 2.2). Whenever one of the internal processes of a smart

sensor requires information from the outside world or produces information for the outside world, it accesses the appropriate records of the IFS and reads (writes) the information to (from) this record. The internal processes of a smart sensor are thus not visible to the outside world.

Often we implement the IFS so that it acts as a temporal firewall between the sensors and the outside world. In this case, the IFS uses *local* interface file systems that can be written by the sensor and read by the outside world, without having direct communication between the sensor and the outside world (Fig. 2.3).

2.4.1 Interface Types

In the smart sensor model we distinguish between three interface types: the real-time service interface, the diagnostic and maintenance interface and the configuration and planning interface. All information that is exchanged across these interfaces is stored in files of the IFS.

Real-time Service (RS) Interface. This interface provides time sensitive information to the outside world. The communicated data is sensor observations made by a sensor on real-time entities in the environment. The RS accesses the real-time IFS files within the smart sensor which is usually time-critical.

Diagnostic and Management (DM) Interface. This interface establishes a connection to each smart sensor. Most sensors need both parameterization and calibration at start-up and the periodic collection of diagnostic information to support maintenance activities. The DM interface accesses the diagnostic IFS files within the smart sensor which is not usually time-critical.

Configuration and Planning (CP) Interface. This interface is used to configure a smart sensor for a given application. This includes the integration

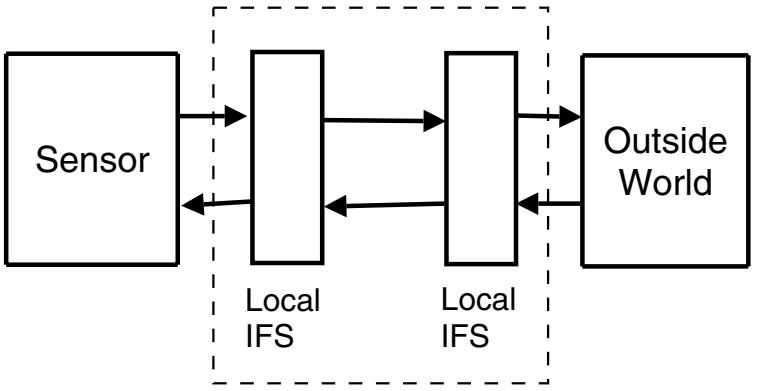

Fig. 2.3. Shows the Interface file system (IFS) built as a temporal firewall. The IFS contains two separate local interface file systems: one is accessed by the sensor and the other is accessed by the outside world.

and set-up of newly connected sensor modules. The CP interface accesses the configuration IFS files within the smart sensor which is not usually time-critical.

2.4.2 Timing

For reading (writing) information from (to) the different interfaces we could use any possible communication and computational schedule [70]. In distributed systems two different types of scheduling are used:

Event Triggered. In this schedule all communication and processing activities are initiated whenever a significant change of state occurs.

Time Triggered. In this schedule all communication and processing activities are initiated at pre-determined times.

For many safety-critical real-time applications the *time-triggered architecture* (TTA) [159] is preferred because its schedule is deterministic. The TTA relies on the synchronization [238] of all local clocks [1] to a global clock: in which all measurements, communication and processing activities are initiated at *pre-determined* time instants which are *a priori* known to all nodes [2].

The use of a global clock does not in and of itself ensure that in a distributed system we are able to *consistently* order events according to their global time. This is illustrated in the following example.

Example 2.2. Ordering Events in a Distributed Network [159]. Consider two local clocks c and C. Suppose the following sequence of events occurs: local clock c ticks, event e occurs, local clock C ticks. In such a situation the event e is time-stamped by the two clocks with a difference of one tick. The finite precision of the global time-base make it impossible in a distributed system to order events consistently on the basis of their global time stamps. □

The TTA solves this problem by using a *sparse time base* in which we partition the continuum of time into an infinite sequence of alternating durations of *activity* (duration π) and *silence* (duration Δ) (Fig. 2.4). The architecture must ensure that significant events, such as the sending of a message or the observation of an event, occur only during an interval of activity. Events occuring during the same segment of activity are considered to have happened at the same time. Events that are separated by at least one segment of silence can be consistently assigned to different time-stamps for all clocks in the system. This ensures that temporal ordering of events is always maintained.

[1] A clock is defined as a physical device for measuring time. It contains a counter and a physical oscillation mechanism that periodically generates an event which increases the counter by one. This event is the "tick" of the clock. Whenever the sensor perceives the occurrence of an event e it will instantaneously record the current tick number as the time of the occurrence of the event e.

[2] The process of synchronizing the local clocks performed using a process known as temporal registration. A brief introduction to some of the techniques used in temporal registration is given in Chapt. 6.

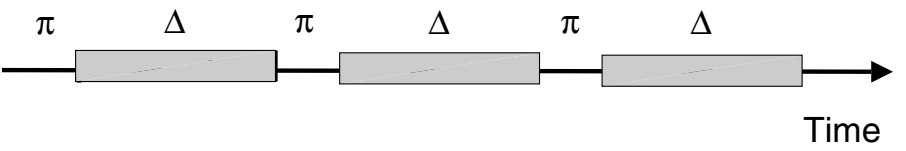

Fig. 2.4. Shows the sparse time based used in the time-triggered architecture. π denotes the intervals of activity and Δ denotes the intervals of silence. If δ is the precision of the clock synchronization, then we require $\Delta > \pi > \delta$.

2.5 Sensor Observation

We may regard a sensor as a small "window" through which we are able to view a physical property which is characteristic of the outside world or environment. In general the physical property is evolving continuously in time and value. However, the sensor only provides us with a "snapshot" of the process: Often the output of a sensor is reduced to a single scalar value.

The output of a sensor is known as a *sensor observation* and includes the following terms:

Entity-Name E. This includes the name of the physical property which was measured by the sensor and the units in which it is measured. Often the units are defined implicitly in the way the system processes the measurement value.

Spatial Location $\boldsymbol{x}$. This is the position in space to which the measured physical property refers. In many cases the spatial location is not given explictly and is instead defined implicitly as the position of the sensor element [3].

Time Instant t. This is the time when the physical property was measured (see Sect. 2.4.2 above). In real-time systems the time of a measurement is often as important as the value itself.

Measurement $\boldsymbol{y}$. This is the value of the physical property as measured by the sensor element. The physical property may have more than one dimension and this is the reason we represent it as a vector $\boldsymbol{y}$. We often refer to it as the *sensor measurement* which may be discrete or continuous. We shall assume that all values are given in digital representation.

Uncertainty $\Delta\boldsymbol{y}$. This is a generic term and includes many different types of errors in $\boldsymbol{y}$, including measurement errors, calibration errors, loading errors and environmental errors. Some of these errors are defined *a priori* in the sensors data sheet and others may be calculated internally if the sensor is capable of validating its own measurements.

Symbolically we represent a sensor observation using the following 5-tuple

$$O = \langle E, \boldsymbol{x}, t, \boldsymbol{y}, \Delta\boldsymbol{y} \rangle \ . \tag{2.1}$$

[3] We shall often use $\boldsymbol{u}$ to represent a spatial location in place of $\boldsymbol{x}$.

Sometimes not all the fields in (2.1) are present. In this case we say the observation is *censored* and we represent the missing fields by an asterix ($*$). For example, if the spatial location $\boldsymbol{x}$ is missing from the sensor observation, then we write the corresponding censored observation as

$$O = \langle E, *, t, \boldsymbol{y}, \Delta\boldsymbol{y} \rangle \, . \tag{2.2}$$

Example 2.3. Time-of-Flight Ultrasonic Sensor [158]. In a time-of-flight ultrasonic sensor ToF, the sonar transducer emits a short pulse of sound and then listens for the echo. The measured quantity is the time between the transmission of the pulse and reception of the echo. This is converted to a *range reading* r which is one-half of the distance traveled by the sound pulse. In this case, $\boldsymbol{y}$ is the range reading and the corresponding sensor observation is

$$O = \langle ToF, *, t, r, \Delta r \rangle \, ,$$

where Δr is the uncertainty in r. □

Example 2.4. A Digital RGB (Color) Image. A digital RGB (color) image I is characterized by an array of pixel gray-levels. At any pixel x, the pixel gray-level $\boldsymbol{g}(x)$ is a vector of three values (R, G, B). Let $\boldsymbol{x}$ denote the relative position of the array of pixels, then the corresponding sensor observation is

$$O = \langle I, \boldsymbol{x}, t, \boldsymbol{G}, * \rangle \, ,$$

where $\boldsymbol{G}$ denotes the array of $\boldsymbol{g}(x)$ values. □

2.5.1 Sensor Uncertainty $\Delta\boldsymbol{y}$

Sensor measurements are uncertain, which means that they can only give an estimate of the measured physical property. The uncertainty can manifest itself in several different ways:

Random Errors are characterized by a lack of repeatability in the output of the sensor. They are caused by measurement noise. For example, they may be due to fluctuations in the capacity of the resistance of the electrical circuits in the sensor, or due to the limited resolution of the sensor.

Systematic Errors are characterized by being consistent and repeatable. There are many different types of systematic errors:

Calibration errors. These are a result of error in the calibration process, and are often due to linearization of the calibration process for devices exhibiting non-linear characteristics.

Loading errors. These arise if the sensor is *intrusive* which, through its operation, alters the measurand.

Environmental Errors. These arise from the sensor being affected by environmental factors which are not taken into account.

Common Representation Format Errors. These occur when we transform from the original sensor space to a common representational format (see Chapt 4).

Spurious Readings are non-systematic measurement errors. An example of such an error occurs when a sensor detects an obstacle at a given location $\boldsymbol{x}$ when, in fact, there is no obstacle at $\boldsymbol{x}$.

Example 2.5. Systematic Errors in a Tipping-Bucket Rain Gauge [302]. Rain-gauges, and in particular the most commonly used tipping-bucket rain-gauge (Fig. 2.5), are relatively accurate instruments. However, at high rain intensities, they suffer from "undercatchment": part of the water is not "weighed" by the tipping-bucket mechanism. This is a *systematic error* which, to a good approximation, may be corrected using the following calibration curve:

$$y = \alpha y_G^{\beta} ,$$

where y is the actual rain intensity, y_G is the rain-gauge measured intensity and α and β are two calibration parameters [302]. □

Example 2.6. Spurious Readings in Time-of-Flight Ultrasonic Sensor. In a time-of-flight ultrasonic sensor (see Ex. 2.3), the sensor calculates the distance r to the nearest obstacle from the measured time between the transmission of a pulse and the reception of its echo. Sometimes the sensor will detect an obstacle at a given range r when, in fact, there is no object at r. This can happen if the sensor receives a pulse emitted by second sensor and interprets the pulse as if it were its own pulse. This is a *spurious reading.* □

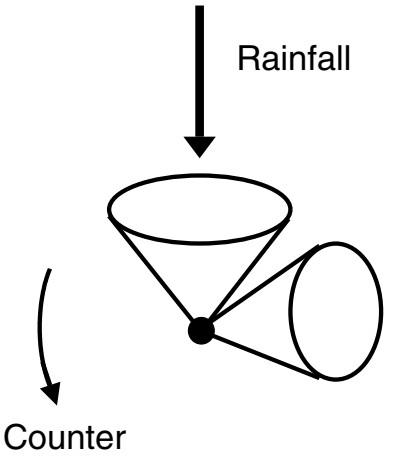

Fig. 2.5. Shows a tipping-bucket rain-gauge. This consists of two specially designed buckets which tip when the weight of 0.01 inches of rain falls into them. When one bucket tips, the other bucket quickly moves into place to catch the rain. Each time a bucket tips, an electronic signal is sent to a recorder. The tipping-bucket rain-gauge is especially good at measuring drizzle and very light rainfall. However it tends to underestimate the rainfall during very heavy rain events, such as thunderstorms.

2.6 Sensor Characteristics

In selecting an appropriate sensor for a single sensor application we need to consider all of the sensors individual characteristics. These characteristics may be grouped into the following five categories:

State. The sensors are classified as being internal or external. Internal sensors are devices used to measure "internal" system parameters such as position, velocity and acceleration. Examples of such sensors include potentiometers, tachometers, accelerometers and optical encoders. External sensors are devices which are used to monitor the system's geometric and/or dynamic relation to its tasks and environment. Examples of such sensors include proximity devices, strain gauges and ultrasonic, pressure and electromagnetic sensors.

Function. The sensors are classified in terms of their functions, i. e. in terms of the parameters, or *measurands*, which they measure. In the mechanical domain the measurands include displacement, velocity, acceleration, dimensional, mass and force.

Performance. The sensors are classified according to their performance measures. These measures include accuracy, repeatability, linearity, sensitivity, resolution, reliability and range.

Output. The sensors are classified according to the nature of their output signal: analog (a continuous output signal), digital (digital representation of measurand), frequency (use of output signal's frequency) and coded (modulation of output signal's frequency, amplitude or pulse).

Energy Type. The sensors are classified according to the type of energy transfered to the sensor. For example, thermal energy involves temperature effects in materials including thermal capacity, latent heat and phase change properties, or electrical energy involves electrical parameters such as current, voltage, resistance and capacitance.

2.7 Sensor-Sensor Properties

In selecting a set of sensors for a multi-sensor application we need to take into account not only the individual sensor characteristics (discussed above) but also the sensor-sensor (relational) properties [18]. We may classify the later under the following headings:

Distributed. Sensors which give information on the same environment but from different points of view or from different subsets of the environment.

Complementary. Sensors which together perceive the whole environment but which individually only perceive a subset of the environment.

Heterogeneous. Individual sensors which give data with completely different characteristics and types.

Redundant. Sensors which perceive the same environment or phenomenon, with little differences between them.

Contradictory. Sensors with the same definition space, but which provide very different information about the same entity.

Concordant. Sensors which provide compatible information concerning the environment, i. e. they corroborate each other.

Discordant. Sensors which provide incompatible information concerning the environment

Different Granuality. Sensors which provide redundant data but which observe the environment at different scales.

Synchronous/Asynchronous. Sensors which provide data which are temporally concordant or not.

A given set of sensors may, of course, possess more than one of the above relationships.

Example 2.7. Physiological Measurements [18]. We consider two physiological measurements made on a given patient: temperature and blood pressure. The measurements are provided by two sensors: a thermometer and a tensiometer. The two data sources are *distributed*, *complementary* and *heterogeneous* as defined on the physiological space of the patient. □

2.8 Sensor Model

[4] As we explained in Sect. 2.2, the smart sensor smoothes, checks and calibrates the sensor signal before it is transmitted to the outside world. In order to perform these functions, we require a sufficiently rich *sensor model* which will provide us with a coherent description of the sensors ability to extract information from its surroundings, i. e. to make meaningful sensor observations. This information will also be required when we consider the fusion of multi-sensor input data. In that case we often include the sensor model within the general "background information" I.

We begin our development of the sensor model by distinguishing between the variable $\boldsymbol{\Theta}$ in which we are interested, and a sensor measurement $\boldsymbol{y}$. We directly observe N *raw* sensor measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, while the variable of interest $\boldsymbol{\Theta}$ is not directly observed and must be *inferred*. In mathematical terms we interpret the task of inferring $\boldsymbol{\Theta}$ as estimating the *a posteriori* probability, $P(\boldsymbol{\Theta} = \boldsymbol{\theta}|\boldsymbol{y}, I)$, where $\boldsymbol{\theta}$ represents the true value of the variable of interest $\boldsymbol{\Theta}$ and $\boldsymbol{y} = (\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_N^T)^T$ denotes the vector of N sensor measurements [5]. This is in accordance with the Bayesian viewpoint

[4] This section contains advanced material. It assumes the reader has some familiarity with the concepts of Bayesian statistics (see Chapts. 8–11). The section is not, however, essential for what follows and it may be left for a second reading.

[5] We use bold letters for both the variable of interest $\boldsymbol{\Theta}$ and for the sensor measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. This is to emphasize that, in general, $\boldsymbol{\Theta}$ and $\boldsymbol{y}_i$

(see Chapt. 8) adopted in this book and which assumes that all the available information concerning $\boldsymbol{\Theta}$ is contained in $p(\boldsymbol{\Theta} = \boldsymbol{\theta}|\boldsymbol{y}, I)$ [6].

Figure (2.6) shows a simple, but useful sensor model [62]. Input to the model are three probability distributions:

A Priori pdf $\pi(\boldsymbol{\theta}|I)$. This is a continuous probability density function which describes our *a priori* beliefs about $\boldsymbol{\theta}$. In the absence of any further information we often model the distribution using a histogram of historical data, or we may construct it from *a priori* information we have concerning typical $\boldsymbol{\theta}$ values. For reasons of computational convenience we often assume that $\pi(\boldsymbol{\theta}|I)$ is a Gaussian distribution with a mean value $\boldsymbol{\mu}_0$ and a covariance matrix $\boldsymbol{\Sigma}_0$:

$$\pi(\boldsymbol{\theta}|I) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \;, \tag{2.3}$$

Sensor Reliability $P(\Lambda|I)$. This is a *discrete* probability distribution which specifies the *a priori* reliability of the sensor. In the simplest model, there

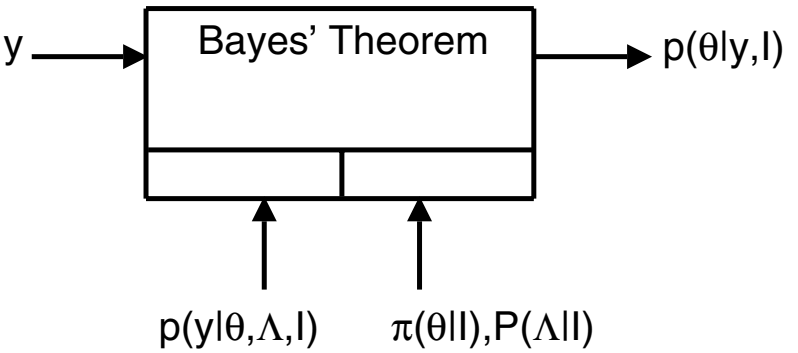

Fig. 2.6. Shows the faulty sensor model. Input to the model are three probabilities: the continuous *a priori* pdf, $\pi(\boldsymbol{\theta}|I)$, and likelihood function, $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$, and the discrete sensor reliability $P(\boldsymbol{\Lambda}|I)$. The model uses Bayes' theorem to calculate the joint probability distribution $p(\boldsymbol{\theta}, \boldsymbol{\Lambda}|\boldsymbol{y}, I) \sim p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\Lambda}, I)\ \pi(\boldsymbol{\theta}|I)P(\boldsymbol{\Lambda}|I)$ and then eliminates $\boldsymbol{\Lambda}$ by marginalization. Output of the model is the *a posterior* pdf $p(\boldsymbol{\theta}|\boldsymbol{y}, I) \sim \pi(\boldsymbol{\theta}|I) \int p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\Lambda}, I)P(\boldsymbol{\Lambda}|I)\mathrm{d}\boldsymbol{\Lambda}$.

may be multi-dimensional. Unless stated otherwise we shall also assume $\boldsymbol{\theta}$ and $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, are continuous. In this case, a probability which is a function of $\boldsymbol{\Theta}$ or $\boldsymbol{y}_i$ should be interpreted as a probability density function (pdf) or distribution. To emphasize the change in interpretation we shall denote probability density functions with a small p.

[6] For variables with continuous values we do not normally distinguish between the random variable and its value. Thus from now on we shall write the pdf $p(\boldsymbol{\Theta} = \boldsymbol{\theta}|I)$ as $p(\boldsymbol{\theta}|I)$. However, in the case of variables with discrete values, this may cause some confusion. For these variables we shall therefore continue to write $P(\boldsymbol{\Lambda} = \boldsymbol{\lambda}|I)$.

are two states: $\Lambda = \{\lambda_0, \lambda_1\}$ where λ_0 denotes fault-free operation and λ_1 denotes faulty operation, where ordinarily $P(\Lambda = \lambda_0) \approx 1$.
The status of the sensor may, of course, change from measurement to measurement. To emphasize this dependency we write the sensor status as $\boldsymbol{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_N)^T$, where Λ_i denotes the status of the sensor when it makes the ith measurement $\boldsymbol{y}_i$.

Likelihood $p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\Lambda}, I)$. This is a continuous function which describes how the raw sensor measurements $\boldsymbol{y}$ depend on the true value $\boldsymbol{\theta}$, the background information I and the sensor status $\boldsymbol{\Lambda}$.

The following example illustrates how we may use this model to perform integrity monitoring in a multi-sensor data fusion navigation system.

Example 2.8. Integrity Monitoring in a Satellite Navigation System [62, 233]. In a satellite navigation system, integrity monitoring refers to the detection and isolation of faulty measurement sensors. We consider a system which is described by a linear Gaussian model (see Sect. 9.6):

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{b}(\boldsymbol{\Lambda}) + \boldsymbol{w} \ ,$$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ denotes the vector of N input measurements and $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_M)^T$ is the unknown vector of navigation parameters. $\boldsymbol{H}$ is a known $N \times M$ measurement matrix, $\boldsymbol{b}(\boldsymbol{\Lambda}) = (b_1(\Lambda_1), b_2(\Lambda_2), \ldots, b_N(\Lambda_N))^T$ is a vector of unknown measurement biases which we describe using the following two-state model:

$$b_i = \begin{cases} 0 & \text{if } \Lambda_i = \lambda_0 \ , \\ B_i & \text{if } \Lambda_i = \lambda_1 \ , \end{cases}$$

and $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)^T$ is a vector of random measurement noise. We assume a zero-mean Gaussian pdf for $\boldsymbol{w}$:

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \ .$$

If we assume that multiple failures do not occur, then we may use E_i to denote the case of a failure in the ith measurement and E_0 to denote the case of no failures. In this case, we may write the *a posteriori* pdf $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$ as a sum over all E_i:

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \frac{\sum_{i=0}^{N} P(E = E_i|I) p(\boldsymbol{y}|\boldsymbol{\theta}, E = E_i, I)}{p(\boldsymbol{y}|I)} \ .$$

Ref. [233] shows the *a posteriori* pdf may be rewritten as

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = c_i \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{H}^{+}_{(i)}\boldsymbol{y}_{(i)}, \boldsymbol{S}_{(i)}) \ ,$$

where $\boldsymbol{H}_{(i)}$ and $\boldsymbol{y}_{(i)}$ are, respectively, the matrix $\boldsymbol{H}$ and the vector $\boldsymbol{y}$ with the ith row removed, $\boldsymbol{\Sigma}_{(i)}$ is the covariance matrix $\boldsymbol{\Sigma}$ with the ith row and column removed, and

$$\begin{aligned}\boldsymbol{H}^{+}_{(i)} &= (\boldsymbol{H}^{T}_{(i)}\boldsymbol{\Sigma}^{-1}_{(i)}\boldsymbol{H}_{(i)})^{-1}\boldsymbol{H}^{T}_{(i)}\boldsymbol{\Sigma}^{-1}_{(i)} \ , \\ \boldsymbol{S}_{(i)} &= (\boldsymbol{H}^{T}_{(i)}\boldsymbol{\Sigma}^{-1}_{(i)}\boldsymbol{H}_{(i)})^{-1}\end{aligned}$$

In this case, $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$ describes the *a posteriori* distribution of the unknown vector $\boldsymbol{\theta}$ and c_i describes the integrity or relative probability that the ith measurement is in error. □

In the above example we have followed standard practice and used a Gaussian function for likelihood $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$. However, there are applications which require a different shaped pdf. For example in modeling an ultrasonic sensor we often use a non-Gaussian asymmetric likelihood function.

Example 2.9. Konolige Model for a Time-of-Flight Ultrasonic Sensor [158]. We consider a time-of-flight ultrasonic sensor (see Ex. 2.3). Suppose we obtain a range reading equal to R. Then the fault-free (i. e. $\Lambda = \lambda_0$) likelihood function is $p(r \circ R|r_0)$, where $r \circ R$ denotes that the *first* detected echo corresponds to a distance R ("$r = R$") and that no return less than R was received ("$r \not< R$"). Let $p(r = R|r_0)$ and $p(r \not< R|r_0)$ denote the conditional pdfs corresponding to $r = R$ and $r \not< R$, then the likelihood function for a time-of-flight ultrasonic sensor is

$$p(r \circ R|r_0) = p(r = R|r_0) \times p(r \not< R|r_0) \ ,$$

where

$$p(r \not< R|r_0) = 1 - \int_0^R p(r = x|r_0)dx \ .$$

To a good approximation,

$$p(r = R|r_0) \propto \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2}\left(\frac{r - r_0}{\sigma}\right)^2 + F \ ,$$

where F is a small constant which takes into account multiple targets which may reflect the sonar beam, in addition to the target at r_0 (see Sect. 10.3.3).

In practice, we observe that the range error becomes proportionally larger and the probability of detection becomes proportinally smaller at increasing range. Incorporating this effects into the above likelihood function we obtain the Konolige likelihood function:

$$\begin{aligned}p(r \circ R|r_0) = \gamma\Bigg[\frac{\alpha(r)}{\sqrt{2\pi\sigma^2(r)}} \exp -\frac{1}{2}\left(\frac{r - r_0}{\sigma(r)}\right)^2 + F\Bigg] \\ \times\left(1 - \int_0^R p(r = x|r_0)dx\right) \ .\end{aligned}$$

where γ is a normalization constant, $\alpha(r)$ describes the attenuation of the detection rate with increasing distance and $\sigma(r)$ describes the increase in the range variance with increasing distance. □

2.9 Further Reading

General references on the use of smart sensors in multi-sensor data fusion are [72, 73, 87, 160]. Sensor models are discussed in [69, 72, 76, 77]. Specific references on the sensor model for the time-of-flight ultrasonic sensors are [158, 231].

3

Architecture

3.1 Introduction

In Chapt. 1 we introduced a formal framework in which we represented a multi-sensor data fusion system as a distributed system of autonomous modules. In this chapter we shall consider the architecture of a multi-sensor fusion system and, in particular, the architecture of the "data fusion block" (Sect. 1.4).

The modules in the data fusion block are commonly called *fusion nodes*. A communication system allows the transfer of information from one node to another node via an exchange of messages. An algorithmic description of the fusion block is provided by the software which is embedded in the nodes and which determines the behaviour of the block and coordinates its activities.

Although the data fusion block is represented as a distributed system we perceive it as a whole rather than as a collection of independent nodes. This phenomenom is known as *transparency* [238] and is concerned with the "concealment" of nodes in a distributed system. There are many different forms of transparency which reflect on the different motivations and goals of the distributed systems. Two important forms of transparency are:

Replication Transparency. Here we increase the reliability and performance of the system by using multiple copies of the same sensor. However the user, or the application program, has no knowledge of the duplicate sensors [238, 247].

Failure Transparency. Here we conceal failures in the hardware or software components and thereby allow the users and application programs to complete their tasks. However the user, or the application program, has no knowledge of the failures [238].

The following example illustrates a *fault-tolerant framework* using replication transparency.

Example 3.1. Fault-Tolerant Framework [14]. We consider the communication between a sensor node S and a fusion node F via an interface file system (IFS) (see Fig. 3.1). In general the environmental image produced by F is sensitive to errors in the value y received from S. We may reduce this sensitivity by using a fault-tolerant framework (Fig. 3.2) as follows. We replace the sensor S by a set of M redundant sensors $S_m, m \in \{1, 2, \ldots, M\}$. If y_m denotes the y value produced by S_m, then we insert an extra component into the fusion node F in order to process the $y_m, m \in \{1, 2, \ldots, M\}$, and produce a new y value which is less likely to be in error. Suppose y_{NEW} denotes the new y value, then the fusion algorithm remains unchanged: It operates on y_{NEW} value and has no knowledge of the y_m.

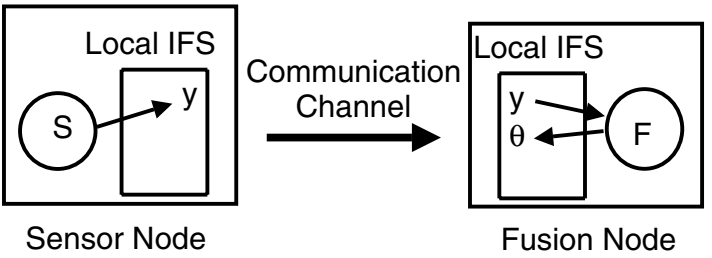

Fig. 3.1. Shows the communication and processing of data. If the sensor S provides a false value for y, then the fusion cell F will be affected by the false value and will in turn produce a wrong value for its output θ.

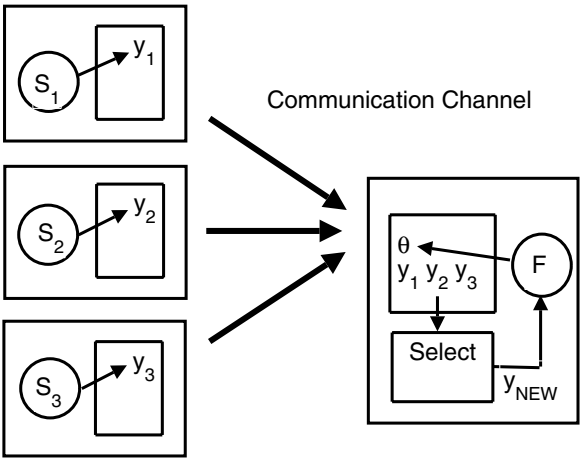

Fig. 3.2. Shows fault-tolerant communication and processing of data. As long as not more than one of the sensor nodes $S_m, m \in \{1, 2, 3\}$, provides a false value for y, then the fusion cell F will *not* be affected by the false value and will in turn produce a correct value for its output θ.

The fault-tolerance processing in Fig. 3.2 can be of various kinds. For example, in a *voting algorithm* we select one observation out of the multiple input observations (see Chapt. 13). □

3.2 Fusion Node

We model the data fusion block as a distributed network of autonomous fusion cells or nodes. Each node receives one, or more, input *sensor observations*

$$O_i = \langle E_i, \boldsymbol{x}_i, t_i, \boldsymbol{y}_i, \Delta\boldsymbol{y}_i \rangle, \qquad i \in \{1, 2, \ldots\} \; . \tag{3.1}$$

If necessary, the observations and any *a priori* knowledge, are converted to a common representational format, which are then fused together to produce new output observations. The output observation may, or may not, differ in position ($\boldsymbol{x}$), time (t), value ($\boldsymbol{y}$) and uncertainty ($\Delta\boldsymbol{y}$) from the input observation(s). However, the output observations are *always* assigned new entity names.

Example 3.2. Image Smoothing. Consider three input images $I_i, i \in \{1, 2, 3\}$, each characterized by an array of pixel gray-levels $\boldsymbol{g}_i$. The corresponding (censored) observations are $O_i = \langle I_i, \boldsymbol{x}_i, t_i, \boldsymbol{g}_i, * \rangle$. The fusion cell F performs pixel-by-pixel smoothing and generates two outputs O' and O'', where O' is a *new* image (entity name E') characterized by an array of pixels with relative positions $\boldsymbol{x}'$ and gray-levels $\boldsymbol{g}'$ and O'' is a score function (entity name E'') characterized by the average change in signal-to-noise ratio, ΔSNR:

$$\begin{aligned} O' &= \langle E', \boldsymbol{x}', t', \boldsymbol{g}', * \rangle \; , \\ O'' &= \langle E'', *, *, \Delta SNR, * \rangle \; . \end{aligned}$$ □

Fig. 3.3 shows a graphical representation of a single fusion cell or node [123, 317]. The node may receive three distinct types of input data or information:

Sensor Information. This includes data which comes *directly* from the sensors $S_m, m \in \{1, 2, \ldots, M\}$, as well as input which has been produced by other fusion cells. Ordinarily, this constitutes most of the input data into the fusion cell.

Auxiliary Information. This includes additional data which is derived by specific processing of the sensor information.

External Knowledge. This includes additional data and consists of all the elements of the *a priori*, or external, knowledge.

Example 3.3. Sensor Information vs. Auxiliary Information. The fusion cell F receives information from two sensors, S_1 and S_2, and from two fusion cells, F_1 and F_2, in the form of the results R_1 and R_2. In classifying R_1 and R_2 we have two options: (1) We may regard F_1 and F_2 as logical sensors in which case we regard R_1 and R_2 as sensor information, or (2) we may regard

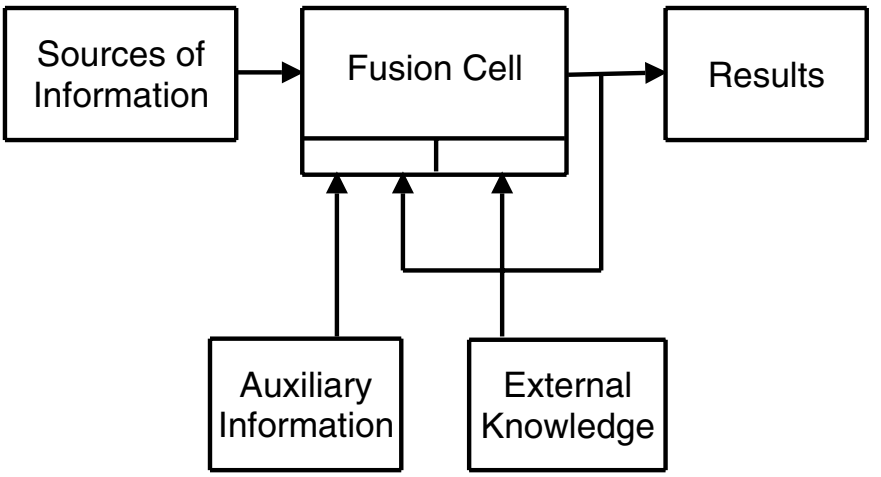

Fig. 3.3. Shows a single fusion node, its three types of input data or information: sources of information, auxiliary information and *a priori* external knowledge and its output results. In the figure we show feedback of the results as auxiliary information.

F_1 and F_2 as performing specific processing in which case we regard R_1 and R_2 as auxiliary information. Ordinarily, we regard $R_m, m \in \{1, 2\}$, as a new logical sensor, or source of information, if it is of the same "type" as S_1 or S_2, otherwise we regard it as auxiliary information. □

3.2.1 Properties

The chief properties [72, 73] of the fusion node are:

Common Structure. Each node has the same structure. This allows designers to easily implement and modify all nodes.

Complete. The node possesses all the information that it requires to perform its tasks. Some of this information, such as the sensor observations, is received through a communications channel.

Relevant. Each node only has information relevant to the functions it is required to perform.

Uncertainty. Observation error and noise mean that the information in a node is uncertain to a greater, or lesser, extent. To ensure a reasonable degree of operational reliability, we must take this uncertainty in account when processing the information in a node.

Communication. Individual nodes are only able to communicate with other nodes that are directly connected to it. A node is not able to send a message to, or receive a message from, another node if it is not directly connected to it.

Other properties which are sometimes important are

Self-Identification. This refers to the ability of the node to automatically identify itself. This facility is important in large networks containing very many nodes.

Self-Testing. Self-testing is the process of inspection and verification which ensures the node is functioning properly. In case of a malfunction the node is able to inform the network about its condition and thus avoid corrupting the system with wrong data.

3.3 Simple Fusion Networks

Fig. 3.3 is a graphical representation of the single fusion node or cell. It is the basic building block for all multi-sensor fusion networks. We now illustrate its versatility by using it to construct four different fusion networks.

3.3.1 Single Fusion Cell

The simplest multi-sensor fusion network consists of a single fusion cell. This cell may be used to fuse together raw sensor measurements. In the network we have the option of introducing external knowledge, in which case, the network represents a *supervised* process

3.3.2 Parallel Network

After the single fusion cell is the parallel network which represents the next higher level of complexity. In the parallel network (Fig 3.4) each fusion cell $F_m, m \in \{1, 2, \ldots, M\}$, separately processes the input data it receives from the sensor S_m and delivers its results R_m to a fusion process F which generates the final result R.

Typically, the intermediate result $R_m, m \in \{1, 2, \ldots, M\}$, are *redundant* and the fusion process F is used to increase reliability and accuracy of the final result R.

Example 3.4. Tire Pressure Monitoring Device [243]. Maintaining the correct tire pressure for road vehicles is very important: incorrect pressure may adversely affect vehicle handling, fuel consumption and tire lifetime. The simplest way to monitor the tire pressure is to directly monitor the pressure in each tire using a separate pressure sensor mounted at the rim of each tire. This is, however, very expensive. The alternative is to indirectly measure the tire pressure using existing sensors. Two such sensors are:

Vibration Sensor S_1. This sensor relies on the fact that the rubber in the tire reacts like a spring when excited by road roughness. The idea is to monitor the resonance frequency which in turn is correlated with the tire pressure.

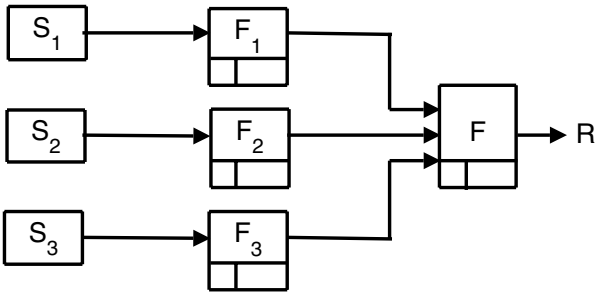

Fig. 3.4. Shows a parallel arrangement of three fusion cells $F_m, m \in \{1, 2, 3\}$. Each cell F_m acts as a virtual sensor which produces the input data R_m. The $R_m, m \in \{1, 2, 3\}$, are then fused together by F.

Wheel Radius Sensor S_2. This sensor relies on the fact that the tire pressure affects the rolling radius of the tire. The idea is to monitor the difference in wheel speeds, which should be close to zero when the tires are equally large.

The two fusion cells $F_m, m \in \{1, 2\}$ carry information about the current tire inflation pressure in the form of two variables: (1) a flag, or *indicator*, I_m, which indicates whether, or not, the tire is under-inflated and (2) a confidence σ_m, which is associated with I_m. The two pairs of variables can be used *independently*: we may issue a warning that the tire is under-inflated if I_1 indicates under-inflation and σ_1 is greater than a given threshold T_1, *or* I_2 indicates under-inflation and σ_2 is greater than a given threshold T_2. In this case the output is the indicator I:

$$I = \begin{cases} 1 \text{ if } \max\big(I_1(\sigma_1 - T_1), I_2(\sigma_2 - T_2)\big) > 0 \ , \\ 0 \text{ otherwise} \ , \end{cases}$$

where $I = 1$ indicates the presence of under-inflation and $I = 0$ indicates the absence of under-inflation.

We may, however, enhance both the robustness and the sensitivity of the system by *fusing* the two sets of measurements in a simple voting scheme (Fig. 3.5). In this case the output is the fused indicator I_{F}, where

$$I_{\mathrm{F}} = \begin{cases} 1 \text{ if } \max\big(I_1(\sigma_1 - T_1), I_2(\sigma_2 - T_2), \frac{I_1 + I_2}{2}\big(\frac{\sigma_1 + \sigma_2}{2} - T_F\big)\big) > 0 \ , \\ 0 \text{ otherwise} \ , \end{cases}$$

and most importantly, $T_{\mathrm{F}} < \min(T_1, T_2)$. □

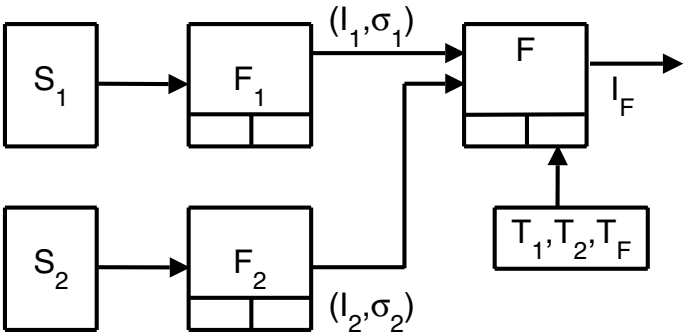

Fig. 3.5. Shows the operation of the new tire pressure monitoring system. The sensor S_1 sends measurements of the resonance frequency to F_1 which outputs the variables (I_1, σ_1). Similarly the sensor S_2 sends measurements of the wheel radii to F_2 which outputs the variables (I_2, σ_2). The cell F receives (I_1, σ_1) and (I_2, σ_2) as input data and generates the signal I_{F} as output. The thresholds T_1, T_2, T_{F} are calculated using a set of training samples D and are regarded as "external knowledge".

3.3.3 Serial Network

The serial, or cascaded, network (Fig. 3.6) contains M fusion cells, each cell $F_m, m \in \{1, 2, \ldots, M-1\}$, receives one, or more, observations from the sensor S_m and it sends its results R_m to the next cell F_{m+1}. In a typical application, the intermediate results $R_m, m \in \{1, 2, \ldots, M\}$, are *heterogeneous* and *complementary*.

Example 3.5. Multi-Modal Biometric Identification Scheme [226]. Fig. 3.7 shows a multi-modal biometric identification scheme built using serial, or cascaded, architecture. The use of this architecture can improve the user convenience as well as allow fast and efficient searches in appropriate databases.

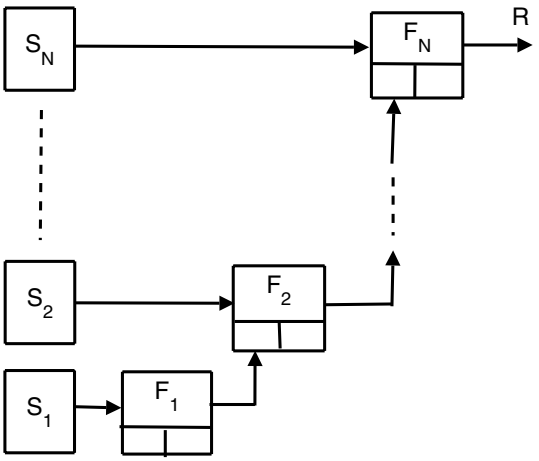

Fig. 3.6. Shows the serial arrangement of multiple fusion cells.

For example, suppose, after processing the first k biometric features, we have sufficient confidence concerning the identity of the user. At this point, we may stop the process and the user will not be required to provide the remaining biometric features. □

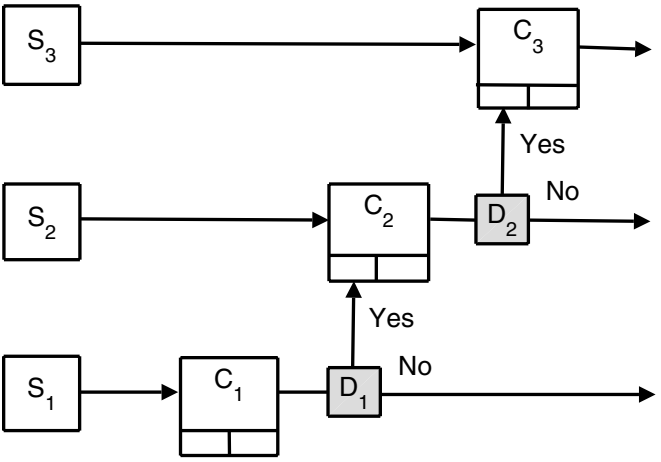

Fig. 3.7. Shows the serial arrangement of multiple classification cells $C_m, m \in \{1, 2, 3\}$, in a three-stage biometric identification scheme. Apart from the decision-boxes $D_m, m \in \{1, 2\}$, the architecture is identical to that shown in Fig. 3.6. The function of the box D_m is to determine whether or not a biometric measurement from sensor S_{m+1} is required.

3.3.4 Iterative Network

Fig. 3.8 represents an iterative network in which we re-introduce the result R as auxiliary knowledge into the fusion cell F. This network is generally used in applications which possess dynamical data sources. In that case, S consists of data changing through time. This scheme integrates many applications such as temporal tracking of objects in an image sequence, or the navigation of a mobile robot. F could be, for example, a Kalman filter.

Example 3.6. Heart-Rate Estimation using a Kalman Filter [1] [275]. Analysis of heart rate variability requires the calculation of the mean heart rate which we adaptively estimate using a one-dimensional Kalman filter [31, 325]. Let $\mu_{k|k}$ denote the estimated mean heart rate at time step k given the sequence of measured heart-rates $y_{1:k} = (y_1, y_2, \ldots, y_k)$. Then we assume $\mu_{k|k}$ follows a random walk process:

[1] This example contains advanced material. It assumes the reader has some familiarity with the concept of a Kalman filter and its notation (see Chapt. 11). The example is not, however, essential for what follows and it may be left for a second reading.

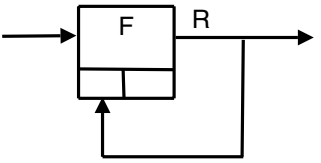

Fig. 3.8. Shows single fusion cell operating in an iterative mode. The result R is re-introduced as auxiliary knowledge into F.

$$\mu_{k|k} = \mu_{k-1|k-1} + v_{k-1} \ ,$$

where $v_{k-1} \sim \mathcal{N}(0, Q)$ represents the process noise at time step $k-1$. We suppose the measured heart-rate at time step k is y_k:

$$y_k = \mu_k + w_k \ ,$$

where $w_k \sim \mathcal{N}(0, R)$ represents the measurement noise. The Kalman filter estimates $\mu_{k|k}$ in a predictor/corrector algorithm: In the prediction phase we project the filtered heart rate $\mu_{k-1|k-1}$ and its covariance matrix $\Sigma_{k-1|k-1}$ to the next time step k. Let $\mu_{k|k-1}$ and $\Sigma_{k|k-1}$ be the corresponding *predicted* values, then

$$\begin{aligned} \mu_{k|k-1} &= \mu_{k-1|k-1} \ , \\ \Sigma_{k|k-1} &= Q + \Sigma_{k-1|k-1} \ . \end{aligned}$$

In the correction phase we fuse the predicted values with the measurement y_k made at the kth time step:

$$\begin{aligned} \mu_{k|k} &= \mu_{k|k-1} + K_k(y_k - \mu_{k|k-1}) \ , \\ \Sigma_{k|k} &= (I - K_k)\Sigma_{k|k-1} \ , \end{aligned}$$

where I is a unit matrix and

$$K_k = \Sigma_{k|k-1}(\Sigma_{k|k-1} + R)^{-1} \ . \qquad \square$$

3.4 Network Topology

Fundamentally the fusion nodes may be arranged in three different ways (topolgies), or architectures: *centralized*, *decentralized* and *hierarchical* (see Table 3.1) in which we list several multi-sensor data fusion We now consider each of these architectures in turn.

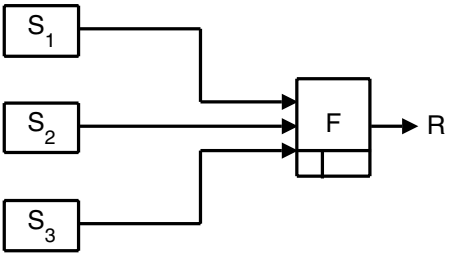

Fig. 3.9. Shows the operation of a centralized node F. The inputs to F are regarded as being produced by separate logical sensors, or sources-of-information, S_1, S_2, S_3.

3.4.1 Centralized

In a centralized system, the sensor fusion unit is treated as a central processor or a node that collects all information from the different sensors (see Fig. 3.9). All decisions are made at this node and instructions, or task assignments, are given out to the respective sensors.

Theoretically, a single fusion node has the best performance if all the sensors are accurately aligned [2]. However, small errors in the correct alignment of the sensors may cause substantial reduction in the performance of the fusion

Table 3.1. Multi-Sensor Data Fusion Applications: Their Networks and Architectures

Network	Application
Single (C)	Ex. 3.7 Satelite Imaging of the Earth.
Parallel/Serial (H)	Ex. 3.4 Tire Pressure Monitoring Device.
Serial (H)	Ex. 3.5 Multi-Modal Biometric Identification Scheme.
Iterative (C)	Ex. 3.6 Heart-Rate Estimation using a Kalman Filter.
Parallel (D)	Ex. 3.8 Covariance Intersection in a Decentralized Network.
Parallel/Serial (H)	Ex. 3.10 Data Incest.
Parallel/Serial (H)	Ex. 3.11 Audio-Visual Speech Recognition System.
Parallel/Serial (H)	Ex. 3.12 Track-Track Fusion.
Parallel/Serial (H)	Ex. 7.8 Fusing Similarity Coefficients: A Numerical Example.
Single (C)	Ex. 9.6 Nonintrusive Speech Quality Estimation Using GMM's.
Single (C)	Ex. 11.1 INS/Radar Altimeter: A Hybrid Navigation Principle.
Single (C)	Ex. 11.2 Tracking a Satellite's Orbit Aound the Earth.
Single (C)	Ex. 11.4 Tracking Metrological Features.

The designations C, D and H refer, respectively, to the centralized, decentralized and hierarchical network topologies.

[2] Technically. a centralized architecture requires a single common representational format in which all sensors are accurately located. See Chapt. 4.

algorithm. In addition the centralized architecture suffers from the following drawbacks:

Communication. All sensors communicate directly with the centralized fusion node. This may cause a communication bottleneck.

Inflexibility. If the application changes, the centralized system may not easily adapt. For example, the central processor might not be powerful enough to handle the new application.

Vulnerability. If there is a failure in the central processor, or the central communications facility, the entire system fails.

Non-modularity. If more sensors are added to a centralized system, some reprogramming is generally required: The communications bandwidth and computer power would need to increase (assuming a fixed total processing time constraint) in proportion to the amount of extra data.

Example 3.7. Satellite Imaging of the Earth [317]. Fig. 3.10 shows the mapping of the earth from different satellite images. For this application the fusion method is a classifier (see Chapt. 12). The output is a segmented image showing the distribution of the different classes. □

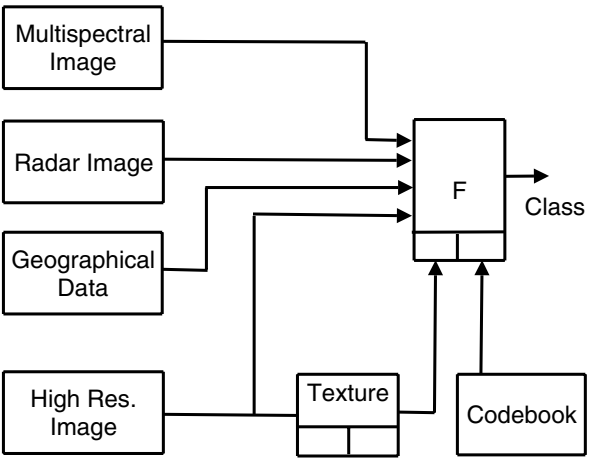

Fig. 3.10. Shows the mapping of a landscape using Earth observation satellite images. The fusion is performed using a centralized architecture. The texture information is produced by a specific processing of the high resolution image and is passed onto the fusion cell F as auxiliary information. The codebook used in the classification procession is regarded as external knowledge.

3.4.2 Decentralized

In a decentralized system, the sensor measurements, or information, is fused locally using a set of local fusion nodes rather than by using a single central

fusion node. The main advantage of a decentralized architecture is the lack of sensitivity regarding the correct alignment of the sensors. Other advantages of a decentralized architecture are:

Communication. The architecture is relatively immune to communication and signal processing bottlenecks. This is because the information flow between nodes is reduced, and the data processing load is distributed.

Scalable. The decentralized architecture is scalable in structure without being constrained by centralized computational bottleneck or communication bandwidth limitations.

Robustness. The decentralized architecture is robust against loss of sensing nodes and any changes in the network.

Modular Design. The decentralized architecture is modular in the design and implementation of the fusion nodes.

However, the decentralized network suffers from effects of *redundant* information [307]. For example, suppose a node A receives a piece of information from a node B, the topology of the network may be such that B is passing along the information it originally received from A. Thus if A were to fuse the "new" information from B with the "old" information it already has under the assumption of independence, then it would *wrongly* decrease the uncertainty in the fused information.

Theoretically we could solve the problem of redundant information by keeping track of all the information in the network. This is not, however, practical. Instead, fusion algorithms have been developed which are robust against redundant information. The method of *covariance intersection* [42, 144] is one such algorithm and is illustrated in the following example.

Example 3.8. Covariance Intersection in a Decentralized Network [42]. We consider a decentralized network in which a given node F receives the *same* piece of information from m different sources of information $S_m, m \in \{1, 2, \ldots, M\}$. We suppose the piece of information sent by S_m is a scalar measurement y_m, and an uncertainty σ_m^2. If we regard the measurements and uncertainties as independent, then the corresponding fused maximum likelihood measurement value and uncertainty (see Chapt. 9) are given by:

$$y_{\mathrm{ML}} = \sum_{m=1}^{M} \frac{y_m}{\sigma_m^2} \Big/ \sum_{m=1}^{M} \frac{1}{\sigma_m^2} \ ,$$

$$\sigma_{\mathrm{ML}}^2 = 1 \Big/ \sum_{m=1}^{M} \frac{1}{\sigma_m^2} \ .$$

Since the sources S_m all supply the same piece of information, then

$$y_1 = y = y_2 = \ldots = y_M \ ,$$
$$\sigma_1 = \sigma = \sigma_2 = \ldots = \sigma_M \ ,$$

and thus

$$y_{\mathrm{ML}} = y \ , \quad \sigma^2_{\mathrm{ML}} = \sigma^2/M \ .$$

The $1/M$ reduction in uncertainty is clearly incorrect: it occurs because the node F assumes that each $(y_m, \sigma^2_m), m \in \{1, 2, \ldots, M\}$, represents an *independent* piece of information.

In the method of covariance intersection we solve this problem by fusing the measurements $y_m, m \in \{1, 1, \ldots, M\}$, using a *weighted* maximum likelihood estimate. The weights Ω_m, are chosen such that $\Omega_m \geq 0$ and $\sum_{m=1}^{M} \Omega_m = 1$. In this case, the corresponding fused estimate and uncertainty are

$$y_{\mathrm{CI}} = \sum_{m=1}^{M} \Omega_m \frac{y_m}{\sigma^2_m} / \sum_{m=1}^{M} \frac{\Omega_m}{\sigma^2_m} \ ,$$
$$\sigma^2_{\mathrm{CI}} = 1 / \sum_{m=1}^{M} \frac{\Omega_m}{\sigma^2_m} \ .$$

If we assume the Ω_m are all equal to $1/M$, then the covariance intersection estimates are:

$$y_{\mathrm{CI}} = y \ , \quad \sigma^2_{\mathrm{CI}} = \sigma^2 \ .$$

In this case, we observe that by using the method of covariance intersection we no longer obtain a reduction in uncertainty. □

In the general case we have M multi-dimensional measurements $\boldsymbol{y}_m, m \in \{1, 2, \ldots, M\}$, with covariance matrices $\boldsymbol{\Sigma}_m$. The covariance intersection estimate is

$$\boldsymbol{y}_{\mathrm{CI}} = \sum_{m=1}^{M} \Omega_m \boldsymbol{\Sigma}_{\mathrm{CI}} \boldsymbol{\Sigma}_m^{-1} \boldsymbol{y}_m \ , \tag{3.2}$$

where

$$\boldsymbol{\Sigma}_{\mathrm{CI}}^{-1} = \sum_{m=1}^{M} \Omega_m \boldsymbol{\Sigma}_m^{-1} \ . \tag{3.3}$$

A fast algorithm for calculating the Ω_m is given in [89]. It is based on the following approximation

$$\Omega_m \approx \frac{|\boldsymbol{S}| - |\boldsymbol{S} - \boldsymbol{\Sigma}_m^{-1}| + |\boldsymbol{\Sigma}_m^{-1}|}{M|\boldsymbol{S}| + \sum_{n=1}^{M} \left(|\boldsymbol{\Sigma}_n^{-1}| - |\boldsymbol{S} - \boldsymbol{\Sigma}_n^{-1}| \right)} \ , \tag{3.4}$$

where $\boldsymbol{S} = \sum_{m=1}^{M} \boldsymbol{\Sigma}_m^{-1}$ and $|\boldsymbol{S}|$ denotes the determinant of $\boldsymbol{S}$.

Example 3.9. Covariance Intersection of Two 2D Observations [144]. Fig. 3.11 shows two two-dimensional observations $O_1 \equiv (\boldsymbol{y}_1, \boldsymbol{\Sigma}_1)$ and $O_2 \equiv (\boldsymbol{y}_2, \boldsymbol{\Sigma}_2)$ and their covariance intersection $O_{\mathrm{CI}} \equiv (\boldsymbol{y}_{\mathrm{CI}}, \boldsymbol{\Sigma}_{\mathrm{CI}})$. □

3.4.3 Hierarchical

This design can be thought of as a hybrid architecture in which we mix together the centralized and decentralized systems. The hierarchical architecture combines the advantages of the centralized and decentralized architectures without some of their disadvantages. For example, the performance of the hierarchical architecture is relatively insensitive to any errors in the correct alignment of the sensors. However, the hierarchical network may still suffer from the effects of redundant information, or *data incest*, as the following example shows.

Example 3.10. Data Incest [201]. Three mobile sensors A, B, C send information concerning a target T to a fusion cell F (Fig. 3.12). C is always out of range of F and it can only relay its information to F via A or B. However C is not always in range of A and B. To increase the chances of its information reaching F, C broadcasts its information to *both* A and B. Assuming F fuses the pieces of information it receives as if they were independent, then data incest occurs when the information from C reaches both A and B and is used twice over by F. Table 3.2 shows the *a posteriori* probability density function (pdf) that F may calculate assuming the transmission of information from C to A (via the channel $C \rightarrow A$) and from C to B (via the channel $C \rightarrow B$) are successful or not. Data incest only occurs when when both $C \rightarrow A$ and $C \rightarrow B$ are successful and the corresponding *a posteriori* pdf is $p(T|A, B, C, C)$. □

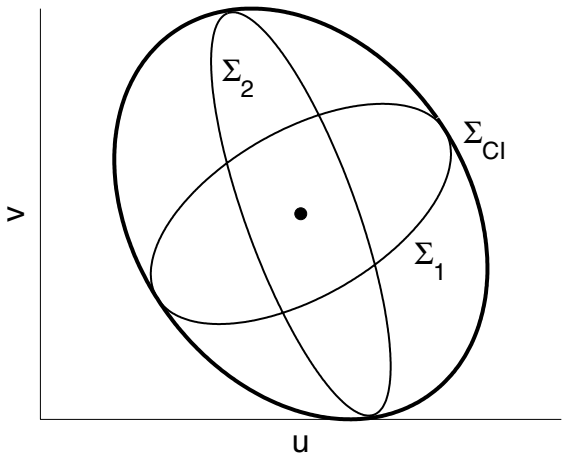

Fig. 3.11. Shows the covariance of two coincident sensor observations $O_1 = (\boldsymbol{y}_1, \boldsymbol{\Sigma}_1)$ and $O_2 = (\boldsymbol{y}_2, \boldsymbol{\Sigma}_2)$ in two dimensions (u, v). The position of the observations is shown by a small filled circle. The covariances, $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, of the two observations are two ellipses drawn with thin solid lines. The covariance of the covariance intersection, $\boldsymbol{\Sigma}_{\mathrm{CI}}$, is the ellipse drawn with a thick solid line.

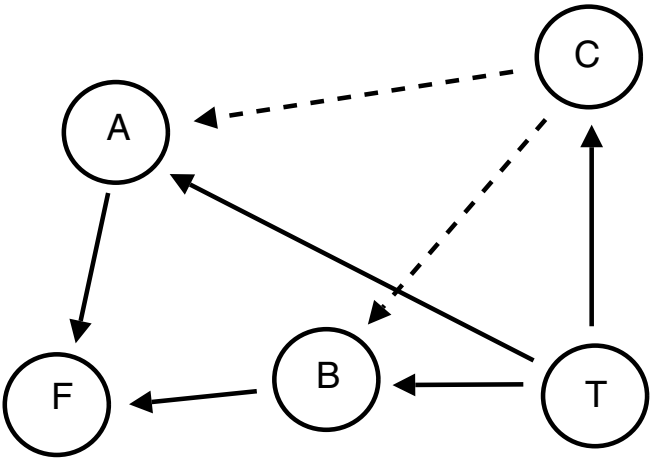

Fig. 3.12. Illustrates the possibility of data incest when the sensor C successfully sends its information regarding the target T to A (via communication channel $C \rightarrow A$) and B (via communication channel $C \rightarrow B$) which in turn forward the information onto the fusion cell F. *Note*: Unreliable communication channels are drawn with dashed arrows.

In the hierarchical architecture there are often several hierarchical levels where the top level contains a single centralized fusion node and last level is made up of several decentralized (local) fusion nodes. Each of the local fusion nodes may receive inputs from a small group of sensors or from an individual sensor. The next two examples illustrate the implementation of a fusion system using a hierarchical architecture.

Example 3.11. Audio-Visual Speech Recognition System [218]. Fig. 3.13 shows the implementation of the audio-visual speech recognition system using a hierarchical architecture (see Ex. 1.7). □

Example 3.12. Track-to-Track Fusion [12]. Fig. 3.14 shows the implementation of the track-to-track fusion system using a hierarchical architecture (see Sect. 11.6.2). □

Table 3.2. *A Posteriori* pdf at F.

$C \rightarrow A$	$C \rightarrow B$	*A Posteriori* pdf
No	No	$p(T\|A,B)$
Yes	No	$p(T\|A,B,C)$
No	Yes	$p(T\|A,B,C)$
Yes	Yes	$p(T\|A,B,C,C)$

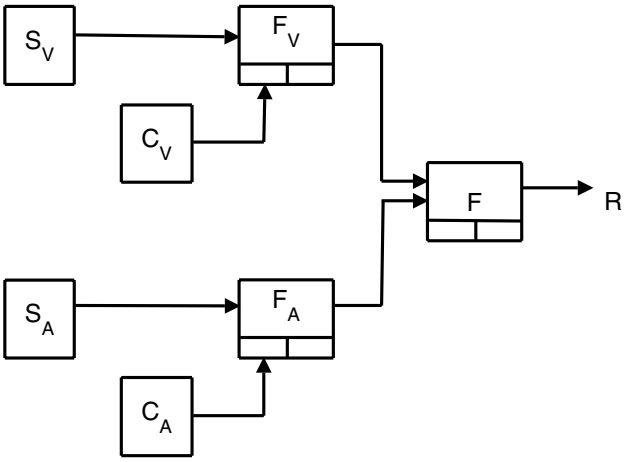

Fig. 3.13. Shows the hierarchical architecture for an audio-visual speech recognition system. The network consists of a single central fusion node F which receives input from an audio fusion node F_{A} and a visual fusion node F_{V}. These nodes receive, respectively, *raw sensor data* from the audio and visual sensors S_{A} and S_{V} and *auxiliary data* from the secondary classifiers C_{A} and C_{V}.

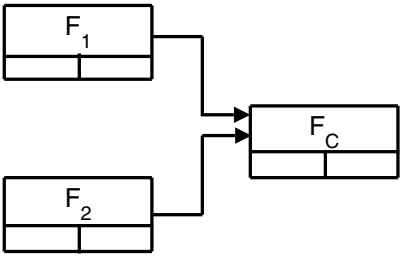

Fig. 3.14. Shows the hierarchical architecture for a track-to-track fusion system. The network consists of two local fusion nodes F_1 and F_2 which form local tracks T_1 and T_2 using the observations received from the sensors S_1 and S_2. the local tracks are sent to the central fusion node F_C where they are fused together into a single track T_C.

3.5 Software

Matlab code for the covariance intersection is given in [144]. However, for fast computation of the covariance intersection we recommend the algorithm in [89].

3.6 Further Reading

General references on the different architectures used in multi-sensor data fusion are: [60, 76, 123, 159, 303]. Covariance intersection is now widely used in decentralized and hierarchical systems. Recent references on this topic are [144, 307].

Part II

Representation

4

Common Representational Format

4.1 Introduction

The subject of this chapter is the *common representational format* . Conversion of all sensor observations to a common format is a basic requirement for all multi-sensor data fusion systems. The reason for this is that only after conversion to a common format are the sensor observations compatible and sensor fusion may be performed. The following example illustrates the concept of a common representational format as used in brain research.

Example 4.1. A Standardized Brain Atlas: A Common Representational Format for Brain Research [296]. In order to compare different brains and, to facilitate comparisons on a voxel-by-voxel basis, we use a standardized anatomically-based coordinate system or *brain atlas*. The idea is that, in the new coordinate system, all brains have the same orientation and size. The transformation to this coordinate system also gives us the means to enhance weak, or noisy, signals by averaging the transformed images. The standardized brain atlas allows us to catalogue the anatomical, metabolic, electrophysiological, and chemical architecture of different brains into the same coordinate systems.

The original atlas brain atlas was derived from a single human brain. Since then research on brain mapping and neuroimaging has grown enormously and the original brain atlas has now been extended in several different ways as listed in Table 4.1. □

In (2.1) we introduced the following notation for a sensor observation:

$$O = \langle E, \boldsymbol{x}, t, \boldsymbol{y}, \Delta\boldsymbol{y} \rangle \ , \tag{4.1}$$

where $\boldsymbol{x}$, t, $\boldsymbol{y}$ and $\Delta\boldsymbol{y}$ are, respectively, the spatial position, the time, the value and the uncertainty of the given property as measured by the sensor. Corresponding to the three types of information - spatial, temporal and value - the process of converting the sensor observations into a common representational format involves the following three functions:

Table 4.1. Different Brain Atlases

Name	Description
Adaptable Brain Atlas	This is a brain atlas which can be individualized to the anatomy of new subjects. This allows the automated labeling of structures in new patients' scans.
Probabilistic Brain Atlas	This is a population-based atlas in which the atlas contains detailed quantitative information on cross-subject variations in brain structure and function. The information stored in the atlas is used to identify anomalies and label structures in new patients [197].
Disease-Specific Brain Atlas	This is a brain atlas which relates to the anatomy and physiology of patients suffering from a specific disease.
Dynamic ($4D$) Brain Atlas	This is an atlas which incorporates probabilistic information on the structural changes which take place during brain development, tumor growth or degenerative disease processes.

Spatial Alignment. Transformation of the local spatial positions $\boldsymbol{x}$ to a common coordinate system. The process is known as *spatial alignment* and involves geo-referencing the location and field-of-view of each sensor. The process is considered in Chapt. 5.

Temporal Alignment. Transformation of the local times t to a common time axis. The process is known as *temporal alignment* and is often performed using a dynamic time warping algorithm. The process is considered in Chapt. 6.

Sensor Value Normalization. The sensor values $\boldsymbol{y}$ and their uncertainties $\boldsymbol{\Delta y}$ are normalized to a common scale. This process is often performed using *robust statistics* and is considered in Chapt. 7.

In many multi-sensor data fusion applications, the construction of the common coordinate system is the primary fusion algorithm. In Table 4.2 we list some of these applications together with the classification of the type of fusion algorithm involved. In the following example we describe an application in which we fuse K direction-only coordinate systems into a single Cartesian coordinate system.

Example 4.2. Bayesian Triangulation of Direction-Bearing Measurements [1] [33, 229]. We reconsider Ex. 1.5 from a Bayesian point-of-view. The position of the object O is estimated from M direction-bearing observations made from known locations. If (x, y) denotes the Cartesian coordinates of the object

[1] This example contains advanced material. It assumes the reader has some basic familarity with the concepts of Bayesian statistics (Chapt. 8). The example is not, however, essential for what follows and it may be left for a second reading.

Table 4.2. Applications in which Construction of the Common Coordinate System is the Primary Fusion Algorithm

Class		Application
DaI-DaO	Ex. 4.1	A Standardized Brain Atlas: A Common Representational Format for Brain Research.
	Ex. 4.3	A Distributed Surveillance System.
	Ex. 4.7	Debiased Cartesian Coordinates.
	Ex. 4.8	Myocardial Imaging.
	Ex. 5.5	Non-Rigid Spatial Alignment in Cardiac MR Images.
	Ex. 5.6	Image Fusion: Pan Sharpening Using Co-Kriging.
	Ex. 5.8	Mosaic Fingerprint Image.
	Ex. 5.9	Mosaic Image of the Retina.
	Ex. 6.1	Cardiac MR Image Sequences.
DaI-FeO	Ex. 1.5	Triangulation: Cooperative Fusion.
	Ex. 4.2	Bayesian Triangulation of Direction-Bearing Measurements.
	Ex. 4.9	Representation of the Spatial Environment for a Mobile Robots.
	Ex. 4.13	Chemical Sensor Analysis: Principal Discriminant Analysis (PDA) for Small-Sample Size Problems.
	Ex. 5.7	Multi-Modal Cardac Imaging.
	Ex. 6.2	Visualization of the Dysphagia (Disorders Associated with Swallowing).

The designations DaI-DaO, DaI-FeO refer, respectively, to the Dasarathy input/output classifications: "Data Input-Data Output" and "Data Input-Feature Output" (Sect. 1.3.3).

and ϕ_m denote its *true* bearing from a sensor S_m located at $(X_m, Y_m), m \in \{1, 2, \ldots, M\}$, then

$$\phi_m = \tan^{-1} \frac{x - X_m}{y - Y_m} \, .$$

The *measured* bearings are distorted by random noise w_m:

$$\theta_m = \phi_m + w_m \, ,$$

where we assume the w_m are independent and identically distributed (iid) according to a zero-mean Gaussian distribution, $w_m \sim \mathcal{N}(0, \sigma_m^2)$. Given the bearings $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_M)^T$, the *a posteriori* probability density $p(\boldsymbol{\theta}|\boldsymbol{\phi}, I)$ is

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{\phi}, I) &= \prod_{m=1}^{M} \frac{1}{\sigma_m \sqrt{2\pi}} \exp - \Big(\frac{\theta_m - \phi_m}{\sigma_m \sqrt{2}} \Big)^2 \, , \\ &= \exp - \sum_{m=1}^{M} \frac{1}{2\sigma_m^2} \left(\theta_m - \tan^{-1} \frac{x - X_m}{y - Y_m} \right)^2 \Big/ \prod_{m=1}^{M} \sigma_m \sqrt{2\pi} \, , \end{aligned}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_M)^T$. Inverting this equation using Bayes' theorem gives

$$p(x, y|\boldsymbol{\theta}) \sim \pi(x, y|I) \exp - \sum_{m=1}^{M} \frac{1}{2\sigma_m^2} \left(\theta_m - \tan^{-1} \frac{x - X_m}{y - Y_m} \right)^2 \, ,$$

where $\pi(x, y|I)$ is an *a priori* probability density which we postulate for each position (x, y) given the background information I. The estimated location of the object O is given by the mean values:

$$\hat{x} = \int xp(x, y|\boldsymbol{\theta})dxdy \ ,$$
$$\hat{y} = \int yp(x, y|\boldsymbol{\theta})dxdy \ . \qquad \square$$

4.2 Spatial-Temporal Transformation

Let T denote a transformation which maps the position $\boldsymbol{x}$ and time t of a sensor observation O to a position $\boldsymbol{x}'$, and a time, t', in a common representational format:

$$(\boldsymbol{x}', t') = T(\boldsymbol{x}, t) \ . \tag{4.2}$$

Then, in general, T has the following form:

$$T(\boldsymbol{x}, t) = (T_{\boldsymbol{x}}(\boldsymbol{x}, t), T_t(\boldsymbol{x}, t)) \ , \tag{4.3}$$

where both the spatial transformation $T_{\boldsymbol{x}}(\boldsymbol{x}, t)$ and the temporal transformation $T_t(\boldsymbol{x}, t)$ are functions of $\boldsymbol{x}$ and t. However, in many multi-sensor data fusion applications we approximate (4.3) with a "decoupled" spatial-temporal transformation :

$$T(\boldsymbol{x}, t) = (T_{\boldsymbol{x}}(\boldsymbol{x}), T_t(t)) \ . \tag{4.4}$$

Even when (4.4) is not very accurate we often find it convenient to use as a first-order approximation. In some applications (4.4) is true by definition e. g. when the system under observation is not changing in time. The following example illustrates the construction of a common representational format for an environment which is essentially static and in which the sensors are all of the same type. In this case the temporal alignment and value normalization functions are not required and the construction of the common representational format reduces to the construction of a common spatial coordinate system.

Example 4.3. A Distributed Surveillance System [308]. The demand for surveillance activities for safety and security purposes has received particular attention for remote sensing in transportation applications (such as airports, maritime environments, railways, motorways) and in public places (such as banks, supermarkets, department stores and parking lots). Such systems typically consist of a number of video-based television cameras located in multiple locations. Consider a sequence of M narrow field-of-view "spot" images $I_m, m \in \{1, 2, \ldots, M\}$, taken of a wide surveillance area. We establish a common coordinate system by building a panoramic or "mosaic" image I^* from the sequence of images I_m (Fig. 4.1). For each image I_m, we find a geometric transformation T_m which maps the local "camera-centered" coordinate system

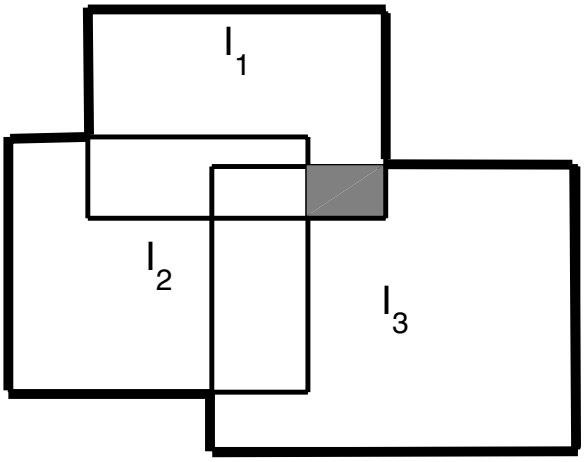

Fig. 4.1. Shows the surveillance of a wide-area site with a sequence of "spot" images $I_m, m \in \{1, 2, 3\}$. Each spot image I_m is transformed to a common coordinate system using a transformation T_m. The region defined by the heavy border is the mosaic image I^*. The shaded area denotes a region where the transformed images $T_1(I_1)$ and $T_3(I_3)$ overlap. In this region the mosaic image I^* is calculated by averaging $T_1(I_1)$ and $T_3(I_3)$.

of I_m to the common "object-centered" coordinate system of I^*. The mosaic image I^* is then defined as the union of the transformed images $T_m(I_m)$:

$$I^* \equiv T_1(I_1) \vee T_2(I_2) \ldots \vee T_M(I_M) \; .$$

In the regions where the spot images overlap we may define the mosaic image as the average of the corresponding intensity values. □

4.3 Geographical Information System

An increasingly important example of a common representational format is a *Geographical Information System*. In a Geographic Information System (GIS) we combine multiple images of the earth obtained from many different sensors and maps, including demographic and infrastructure maps, into a common coordinate system.

In building a GIS we often need to convert a set of sensor measurements $y(\boldsymbol{u}_i), i \in \{1, 2, \ldots, N\}$, made at specified spatial locations $\boldsymbol{u}_i$, into a continuous map. One way of doing this is known as *Kriging* [52, 106, 273], where the term Kriging refers to a family of best (in the least squares sense) linear unbiased estimators (BLUE) (see Table 4.3). We interpret the $y_i = y(\boldsymbol{u}_i)$ as the outcome of a random field $y(\boldsymbol{u})$. If $y_i, i \in \{1, 2 \ldots, N\}$, denotes the sensor

Table 4.3. Kriging Estimators

Estimator	Description
Simple	Assumes a known mean value $\mu(\boldsymbol{u}_0) = \mu_0$ in the local neighbourhood of $\boldsymbol{u}_0$.
Ordinary	Assumes a constant but unknown mean value $\mu(\boldsymbol{u}_0) = \mu_0$ in the local neighbourhood of $\boldsymbol{u}_0$.
Universal	Assumes the mean value $\mu(\boldsymbol{u})$ varies smoothly in neighbourhood of $\boldsymbol{u}_0$. $\mu(\boldsymbol{u}_0)$ is modelled as a linear combination of known basis functions $f_l(\boldsymbol{u})$: $\mu(\boldsymbol{u}_0) = \sum_{l=1}^{L} \beta_l f_l(\boldsymbol{u}_0)$, where the parameters $\beta_l, l \in \{1, 2, \ldots, L\}$ are found by solving the appropriate Kriging equations.
Indicator	Uses a threshold to create binary data. The ordinary Kriging procedure is then applied to the binary data. The covariance matrix used in the Kriging eqs. is calculated on the binary data.
Disjunctive	Uses multiple thresholds to create multiple binary input data. Cokriging procedure is then applied to the multiple binary data.
Co-Kriging	Combines the primary input measurements $y(\boldsymbol{u}_i)$ with K auxiliary variables $z_1(\boldsymbol{u}), z_2(\boldsymbol{u}), \ldots, z_K(\boldsymbol{u})$. Often used when the auxiliary variables are known at a relatively few places $\boldsymbol{u}_j$.
Kriging with External Drift	Similar to Co-Kriging except the auxiliary variables are known at all $\boldsymbol{u}$ including the required location $\boldsymbol{u}_0$.

measurements which are located in the neighborhood of a given point $\boldsymbol{u}_0$, then the estimated sensor value at $\boldsymbol{u}_0$, is

$$\hat{y}(\boldsymbol{u}_0) = \mu(\boldsymbol{u}_0) + \sum_{i=1}^{N} \lambda_i (y_i - \mu(\boldsymbol{u}_0)) \;, \tag{4.5}$$

where $\mu(\boldsymbol{u}_0)$ is the mean value at $\boldsymbol{u}_0$ and the weights $\lambda_i, i \in \{1, 2, \ldots, N\}$, are chosen to give the best unbiased solution. Mathematically, $\mu(\boldsymbol{u}_0)$ and λ_i, are solutions of a system of linear equations known as the *Kriging equations*.

Example 4.4. Equations for Ordinary Kriging [105]. In ordinary Kriging the mean $\mu(\boldsymbol{u})$ is equal to an unknown constant μ_0 in the local region of $\boldsymbol{u}_0$. In this case, the estimated sensor value at $\boldsymbol{u}_0$ is

$$\hat{y}(\boldsymbol{u}_0) = \mu_0 + \sum_{i=1}^{N} \lambda_i (y_i - \mu_0) \;.$$

The unknown mean μ_0 and the weights λ_i are given by the following matrix equation:

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \mu_0 \end{pmatrix} = \begin{pmatrix} \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_1) & \ldots & \Sigma(\boldsymbol{u}_N, \boldsymbol{u}_1) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_N) & \ldots & \Sigma(\boldsymbol{u}_N, \boldsymbol{u}_N) & 1 \\ 1 & \ldots & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \Sigma(\boldsymbol{u}, \boldsymbol{u}_1) \\ \vdots \\ \Sigma(\boldsymbol{u}, \boldsymbol{u}_N) \\ 1 \end{pmatrix} \;,$$

where $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$ is the spatial covariance,

$$\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j) = E\big((y(\boldsymbol{u}_i) - \mu(\boldsymbol{u}_i))(y(\boldsymbol{u}_j) - \mu(\boldsymbol{u}_j))^T\big) \ ,$$

and $\mu(\boldsymbol{u}_i)$ is the mean sensor value in the neighbourhood of $\boldsymbol{u}_i$.

In ordinary Kriging, the weights λ_i sum to one. In this case, we may write $\hat{y}(\boldsymbol{u}_0)$ more simply as:

$$\hat{y}(\boldsymbol{u}_0) = \sum_{i=1}^{N} \lambda_i y_i \ . \qquad \square$$

Example 4.5. A Comparison of Simple, Ordinary, Universal and Indicator Kriging [105]. Fig. 4.2 shows the simple, ordinary, universal and indicator Kriging solutions for a set of input measurements y_i plotted as a function of their distance u_i measured along a given straight-line. $\square$

Apart from assuming a given model for the spatial mean $\mu(\boldsymbol{u})$, the only information required by the Kriging equations is the spatial covariance function $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$.

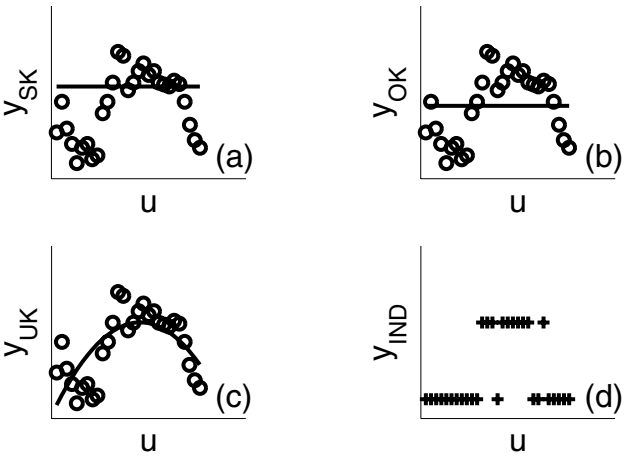

Fig. 4.2. Shows the simple, ordinary, universal and indicator Kriging solutions for a set of input measurements y_i plotted as a function of their distance u_i measured along a given straight-line. **(a)** Shows the simple Kriging solution. This is horizontal straight-line of given height μ_0. **(b)** Shows the ordinary Kriging solution. This is a horizontal straight-line whose height μ_0 is optimally calculated by the algorithm. **(c)** Shows the universal Kriging solution. This is smoothly varying curve $\mu(u)$. **(d)** Shows the indicator Kriging solution. This is a discontinuous curve. It has a value of 0 for all points which fall below a given threshold t. All other points receive a value of 1.

4.3.1 Spatial Covariance Function

Conventionally we learn the spatial covariance, $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$, from the input measurements y_i subject to any *a priori* knowledge and given constraints. Some of the constraints are *necessary*. For example, we require $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$ to be *positive definite*. Mathematically, this constraint requires $\sum_i \sum_j a_i a_j \Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j) \geq 0$ for all $\boldsymbol{u}_i$, $\boldsymbol{u}_j$ and real numbers a_i, a_j. Other constraints may *not* be necessary, but we shall, nevertheless, require them to be true. For example, we shall require $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$ to be *second-order stationary*. Mathematically, this constraint requires $\mu(\boldsymbol{u})$ to be the same for all $\boldsymbol{u}$ and $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$ to depend only on the difference $\boldsymbol{h} = (\boldsymbol{u}_i - \boldsymbol{u}_j)$. One way of satisfying these constraints is to fit [2] the experimentally measured covariance function, $\widetilde{\Sigma}(\boldsymbol{h})$, to a *parametric* curve $\Sigma(\boldsymbol{h}|\boldsymbol{\theta})$, which is known to be positive definite and second-order stationary. See Table 4.4 for a list of parametric univariate covariance functions which are positive-definite and second-order stationary.

Let $N(\boldsymbol{h})$ be the number of pairs of measurements $(y(\boldsymbol{u}_i), y(\boldsymbol{u}_j))$ whose separation $(\boldsymbol{u}_i - \boldsymbol{u}_j)$ is close to $\boldsymbol{h}$. Then an experimentally measured estimate of $\Sigma(\boldsymbol{h})$ is

$$\widetilde{\Sigma}(\boldsymbol{h}) \approx \frac{1}{N(\boldsymbol{h})} \sum_{(\boldsymbol{u}_i - \boldsymbol{u}_j) \approx \boldsymbol{h}} (y(\boldsymbol{u}_i) - \mu)(y(\boldsymbol{u}_j) - \mu)^T \,, \tag{4.6}$$

where $\mu = \frac{1}{N} \sum_{i=1}^{N} y(\boldsymbol{u}_i)$ is an estimate of the mean value of y [3].

Example 4.6. Calculation of the Covariance Matrix [37]. Given the 5×5 digital image

$$\begin{array}{c} 1\,1\,2\,2\,5 \\ 3\,2\,3\,1\,1 \\ 0\,1\,1\,0\,1 \\ 3\,2\,4\,0\,1 \\ 2\,1\,1\,2\,2 \end{array}$$

[2] See Ex. 9.8 for an example of the least square fitting of the experimental measured covariance $\widetilde{\Sigma}(\boldsymbol{h})$.

[3] Eq. (4.6) is sensitive to outliers (see Chapt. 10). If outliers are thought to be present, an alternative "robust"estimate should be used. Two such estimates are:

$$\widetilde{\Sigma}(\boldsymbol{h}) = (\sum a_{ij})^4 / (0.457 N(\boldsymbol{h}) + 0.494) \,,$$
$$\widetilde{\Sigma}(\boldsymbol{h}) = (\text{med}\{a_{ij}\})^4 / 0.457 \,,$$

where $a_{ij} = \sqrt{|(y(\boldsymbol{u}_i) - \mu)(y(\boldsymbol{u}_j) - \mu)^T|}$ and the summation and the median are taken over all pairs of points $(\boldsymbol{u}_i, \boldsymbol{u}_j)$ for which $(\boldsymbol{u}_i - \boldsymbol{u}_j) \approx \boldsymbol{h}$.

we compute the covariance matrix $\widetilde{\Sigma}(h)$ for $h = 2$ in the E-W direction. The solution is

$$\begin{aligned}\widetilde{\Sigma}(h) = \big((1-2)^2 + (1-2)^2 + (2-5)^2 + (3-3)^2 + (2-1)^2 \\ + (3-1)^2 + (0-1)^2 + (1-0)^2 + (1-1)^2 + (3-4)^2 \\ + (2-0)^2 + (4-1)^2 + (2-1)^2 + (1-2)^2 + (1-2)^2\big)/15 \,,\end{aligned}$$

or

$$\widetilde{\Sigma}(h) = 35/15 = 2\frac{1}{3} \,. \qquad \square$$

Table 4.4. Positive Definite Second-Order Stationary Univariate Covariance Functions

Name	$\Sigma(h\|\boldsymbol{\theta})$
Linear	$\alpha - \beta h.$
Spherical	$\alpha - \beta \min(\frac{3h}{2H} - \frac{1}{2}(\frac{h}{H})^3, 1).$
Exponential	$\alpha - \beta(1 - \exp -(\frac{h}{H})).$
Gaussian	$\alpha - \beta(1 - \exp -(\frac{h}{H})^2).$
Exponential Power	$\alpha - \beta(1 - \exp -(\lvert\frac{h}{H}\rvert^p)).$

The formulas in the table only apply for $h > 0$. By definition $\Sigma(h = 0|\boldsymbol{\theta}) = 1$, where $\boldsymbol{\theta}$ denotes the parameters $\{\alpha, \beta, H, p\}$ which are commonly referred to as nuggett (α), sill (β) and practical range (H).

4.4 Common Representational Format

It is clear from the above that the common representational format plays a crucial role in a multi-sensor data fusion system. In fact in many cases the choice of common representational format and, in particular, the method used to represent the sensor measurement uncertainty, will govern what type of fusion algorithm is used. The following two examples illustrate this.

Example 4.7. Debiased Cartesian Coordinates [208]. We consider tracking an object O in two-dimensions using a range/bearing sensor measurements (r_i, θ_i). Let (r_0, θ_0) denote the true range/bearing of the object O, then we assume

$$\begin{aligned} r_i &= r_0 + \delta r \,, \\ \theta_i &= \theta_0 + \delta\theta \,, \end{aligned}$$

where $\delta r \sim \mathcal{N}(0, \sigma_r^2)$ and $\delta\theta \sim \mathcal{N}(0, \sigma_\theta^2)$ are iid random measurement errors.

For applications which involve target tracking (see Chapt. 11) it is convenient to use a Cartesian coordinate system (x, y) since this defines an inertial reference frame. If (x_i, y_i) are the Cartesian coordinates corresponding to (r_i, θ_i), then formally we may write

$$x_i = r_i \cos\theta_i \ ,$$
$$y_i = r_i \sin\theta_i \ .$$

It turns out, however, that $E(x_i) \neq 0$ and $E(y_i) \neq 0$, i. e. x_i and y_i are *not* unbiased estimates of the true Cartesian coordinates $x_0 = r_0 \cos\theta_0$ and $y_0 = r_0 \sin\theta_0$. To correct for these bias effects we use modified, or "debiased", Cartesian coordinates $(x_{\mathrm{DB}}, y_{\mathrm{DB}})$ [86]. For range/bearing measurements, the bias is multiplicative and the corrected coordinates are found by dividing through by a correction factor λ:

$$(x_{\mathrm{DB}}, y_{\mathrm{DB}}) \approx \left(\frac{x_i}{\lambda}, \frac{y_i}{\lambda}\right) \ ,$$

where $\lambda = \exp(\sigma_\theta^2/2)$. □

Example 4.8. Myocardial Imaging [16]. Polar maps, or "bull's-eye" images, are a standard way of displaying myocardial functions and are well established in clinical settings. They are constructed by combining images from multiple planes so that information about the entire myocardium can be displayed in a single image. Polar maps can be compared to a three-dimensional cone-shaped heart activity image projected onto a single plane. Each image plane forms a ring in the polar map. Although the rings may be divided into an arbitrary number of sectors, in practice, a clinician uses four (anterior, lateral, inferior and septal) or six (anterior, anterior-lateral, inferior-lateral, inferior, inferior-septal and anterior-septal) sectors for his visual interpretation of the image. Fig. 4.3 shows the polar image representation of a left ventricle. □

Important factors we should take into account when choosing a common representational format are:

Completeness. The format should support a complete and non-redundant representation of the environment ("environmental image").

Uncertainty Representation. The format should support an *explicit* description of uncertainty in the environmental image.

Computation and Storage. The format should support efficient computations and have low storage requirements.

Adaptability. The format should be capable of adapting itself to a wide variety of circumstances and applications.

The following example illustrates how we take these factors into account when choosing an appropiate common representational format.

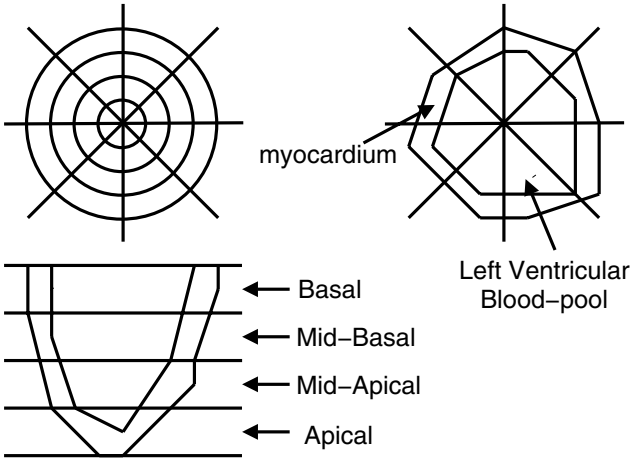

Fig. 4.3. Shows a polar image of a left ventricle. The polar image contains 32 sectors. The sectors are formed by dividing the left ventricle into four slices (Basal, Mid-Basal, Mid-Apical and Apical) and dividing each slice into 8 sectors.

Example 4.9. Representation of the Spatial Environment for Mobile Robots [297]. For mobile robots, a common representational format for a mobile robot must clearly include a full description of the spatial environment. Two common representational formats (Fig. 4.4) which have been used are:

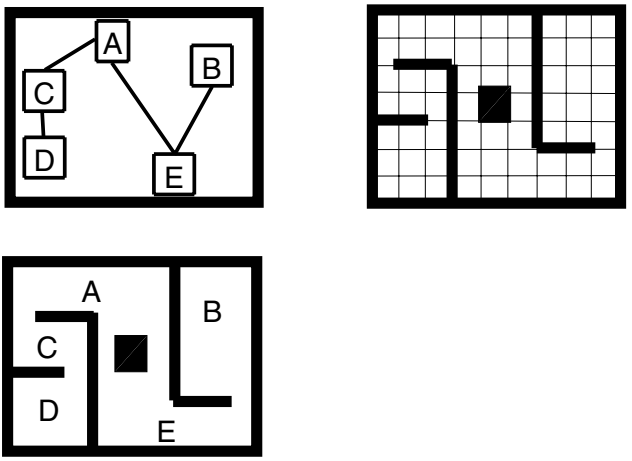

Fig. 4.4. Shows a simple two-dimensional spatial environment (bottom left-hand corner) and the corresponding topological (top left-hand corner) and occupancy grid (top right-hand corner) representations.

Topological Representation. This representation uses a graph-like structure to represent the spatial environment: Nodes in the graph correspond to specific land marks, or places, and arcs in the graph correspond to paths between the places. In the topological representation the position of the robot is determined relative to distinct spatial landmarks. In the topological paradigm, there is no need to accurately determine the absolute location of the robot in the common coordinate system.

Occupancy Grid Representation. This representation discretizes the spatial environment into a structure of square cells C_k each of size $\delta \times \delta$. Associated with each cell C_k is an *a posteriori* probability that the corresponding place in the environment is occupied. In order to create an accurate map of the environment, we require an accurate determination of the location of the robot in the global coordinate frame. In this representation we are sensitive to errors in the location of the robot.

Although the computational requirements of the occupancy grid are high, it provides us with a complete representation of the spatial environment and of its uncertainty. It is also highly adaptable. For these reasons the occupancy grid has, in recent years, become the dominant common representational paradigm for mobile robot applications. □

4.5 Subspace Methods

In many multi-sensor data fusion applications our paramount concern is to keep the computational load and/or the storage requirements low. This may be achieved by using a *low-dimensional* common representational format. One way of producing such a format is to apply a dimension-reducing, or subspace, technique to the raw input data.

Example 4.10. Common Representational Format for a Vehicle Sound Signature Application [335]. All moving vehicles make some kind of noise in which vehicles of the same kind and working in similar conditions generate similar noises. This fact is used to detect and classify vehicles moving in a given surveillance region: Each vehicle is classified by its power spectrum in the frequency range $\sim [5, 6500]$ Hz. If each spectrum is sampled every ~ 5 Hz, the corresponding common representational format has $N \sim 1200$ dimensions. In [335] the authors were able to successfully reduce the number of dimensions from $N \sim 1200$ to $L \sim 30$ by using the principal component analysis (PCA) subspace technique. □

Table 4.5 lists some of the principal subspace techniques which are commonly used for dimension-reducing purposes. Two popular subspace technique are *principal component analysis* (PCA) and *linear discriminant analysis* (LDA).

Table 4.5. Subspace Techniques

Technique	Description
Principal Component Analysis (PCA)	Linear transformation chosen so the projected components have maximum variance [141].
Linear Discriminant Analysis (LDA)	Linear transformation for $K \geq 2$ classes. Transformation is chosen so the projected components for each class are maximally separated from the projected components of the other classes [351].
Independent Component Analysis (ICA)	Linear transformation chosen so the projected components have maximized independence [132].
Non-Negative Matrix Factorization (NMF)	Finds factors with non-negative elements.
Canonical Correlation Analysis (CCA)	For K classes defines K linear transformations [112]. For $K = 2$ the transformations are chosen so the projected components of each class are maximally correlated.

4.5.1 Principal Component Analysis

Principal component analysis (PCA) is an *unsupervised* technique which builds a low-dimensional commmon representational format using a set of input vectors $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. The following example explains the mathematical framework which underpins principal component analysis.

Example 4.11. Principal Component Analysis [141]. The aim of principal component analysis (PCA) is to find a L-dimensional *linear* projection that best represents the input data in a least squares sense. Let the input data be a set of M-dimensional input vectors $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, where $\boldsymbol{y}_i = (y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(M)})^T$. If we define a reduced L-dimensional space using a set of orthonormal axes $\boldsymbol{u}_l, l \in \{1, 2, \ldots, L\}$, then

$$\boldsymbol{\theta}_i = \boldsymbol{U}^T(\boldsymbol{y}_i - \boldsymbol{\mu}) \ ,$$

is a L-dimensional representation of $\boldsymbol{y}_i$ [4], where $\boldsymbol{\theta}_i = (\theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(L)})^T$ and $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2 \ldots, \boldsymbol{u}_L)$. Mathematically, the orthornormal axes $\boldsymbol{u}_l, l \in \{1, 2, \ldots, L\}$, are given by the L dominant eigenvectors of the sample covariance matrix $\boldsymbol{\Sigma}$, i. e.

$$\boldsymbol{\Sigma}\boldsymbol{u}_l = \lambda_l \boldsymbol{u}_l \ ,$$

[4] Given $\boldsymbol{\theta}_i$, the optimal reconstruction of $\boldsymbol{y}_i$ is $\widetilde{\boldsymbol{y}}_i = \boldsymbol{U}\boldsymbol{\theta}_i + \boldsymbol{\mu} = \boldsymbol{U}\boldsymbol{U}^T(\boldsymbol{y}_i - \boldsymbol{\mu}) + \boldsymbol{\mu}$.

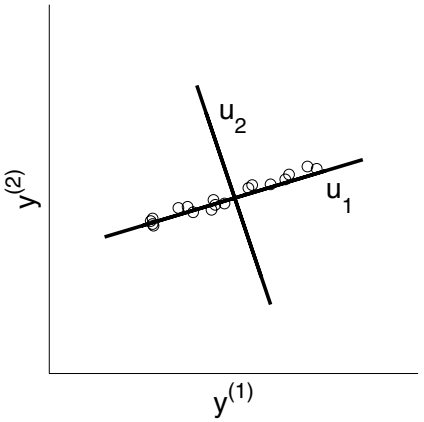

Fig. 4.5. Shows the two-dimensional input measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. The PCA representation consists of vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, where the dominant vector is $\boldsymbol{u}_1$.

where

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{y}_i - \boldsymbol{\mu})(\boldsymbol{y}_i - \boldsymbol{\mu})^T \ ,$$

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i \ .$$

Fig. 4.5 shows the PCA of a set of two-dimensional input measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. □

In many multi-sensor data fusion applications it is convenient to use a *probabilistic* common representational format in which we transform the measurements $\boldsymbol{y}_i$ into L-dimensional *a posteriori* probabilities $p(\boldsymbol{\theta}_i | \boldsymbol{y}_i, I)$. To carry out this transformation we use a probabilistic PCA (see Sect. 9.6.3).

In Table 4.6 we list some of the PCA variants which are in common use.

4.5.2 Linear Discriminant Analysis

The method of *linear discriminant analysis* (LDA) is a *supervised* technique in which we suppose each input measurement $\boldsymbol{y}_i$ is associated with a given class $C = c_k, k \in \{1, 2, \ldots, K\}$. The aim of LDA is to find a L-dimensional subspace $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_L)$ in which the different classes are maximally separated. If $\boldsymbol{\theta}_i = \left(\theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(L)}\right)^T$ denotes the L-dimensional LDA representation of $\boldsymbol{y}_i$, then

$$\boldsymbol{\theta}_i = \boldsymbol{U}^T \boldsymbol{y}_i \ , \tag{4.7}$$

Table 4.6. Principal Component Analysis and its Variants

Technique	Description
Principal Component Analysis (PCA)	The basic PCA algorithm. PCA maximizes the variance of all linear projections [141].
Probabilistic PCA (PPCA)	Places PCA in a probabilistic framework. See Sect. 9.6.3.
Weighted PCA	Each input measurement $\boldsymbol{y}_i$ is given a different weight before PCA is performed [283].
Incremental PCA	Given an existing PCA and a new measurement $\boldsymbol{y}$, incremental PCA calculates the new PCA without recalculating the covariance matrices [283].
Robust PCA	Performs PCA using robust techniques. See Chapt. 10.
Kernel PCA	Generalizes the PCA to non-linear transformations.
2D-PCA	Performs PCA on two-dimensional input matrices. Useful in image processing and computer visual applications [41, 342, 349].

where $\boldsymbol{u}_l$ is a solution of the eigenvector equation:

$$\boldsymbol{H}\boldsymbol{u}_l = \lambda_l \boldsymbol{u}_l \ . \tag{4.8}$$

In (4.8), $\boldsymbol{H} = \boldsymbol{\Sigma}_W^{-1}\boldsymbol{\Sigma}_B$, where $\boldsymbol{\Sigma}_B$ and $\boldsymbol{\Sigma}_W$ are two covariance matrices which measure, respectively, the scatter of the $\boldsymbol{y}_i$ "between-classes" and "within-classes". Mathematically, the covariance (scatter) matrices are defined as follows:

$$\boldsymbol{\Sigma}_B = \frac{1}{N}\sum_{k=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu}_G)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_G)^T \ , \tag{4.9}$$

$$\boldsymbol{\Sigma}_W = \frac{1}{N}\sum_{k=1}^{K} N_k \boldsymbol{\Sigma}_k \ , \tag{4.10}$$

where N_k is the number of input measurements which belong to class c_k, and $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are, respectively, the mean vector and covariance matrix for the kth class, and

$$\boldsymbol{\mu}_G = \frac{1}{N}\sum_{k=1}^{K} N_k \boldsymbol{\mu}_k \ , \tag{4.11}$$

is the global mean vector.

In (4.9) the rank of $\boldsymbol{\Sigma}_B$ is bounded from above to $K-1$. This places the following upper limit on L: $L \leq (K-1)$. In many multi-sensor data fusion applications this restriction on L is too severe [5]. If we require a richer common representational format, we may obtain it by replacing $\boldsymbol{\Sigma}_B$ with a

[5] For example, for two classes ($K = 2$) we have only one axis $\boldsymbol{u}_1$ (see Fig. 4.6).

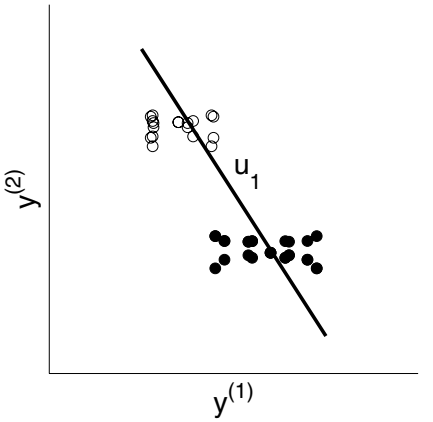

Fig. 4.6. Shows the two-dimensional input measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, with their class labels. Measurements which belong to class c_1 are denoted by an open circle and measurements which belong to class c_2 are denoted by a closed circle. For a two-class problem the LDA representation has a single vector labelled $\boldsymbol{u}_1$ which joins $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

modified matrix $\widetilde{\boldsymbol{\Sigma}}_B$ whose rank is greater than $K-1$. The following example illustrates one such modified scatter matrix.

Example 4.12. The Chen-Yang LDA Algorithm [46]. In the Chen-Yang LDA algorithm we use a modified between-class scatter matrix, $\widetilde{\boldsymbol{\Sigma}}_B$. For two classes $C = c_1$ and $C = c_2$, the (m, n)th component of $\widetilde{\boldsymbol{\Sigma}}_B$ is defined as

$$\widetilde{\boldsymbol{\Sigma}}_B^{(m,n)} = \frac{1}{N_1} \sum_{i=1}^{N} y_i^{(m)} y_i^{(m)} z_i - \frac{1}{N_2} \sum_{i=1}^{N} y_i^{(n)} y_i^{(n)} (1 - z_i) \ ,$$

where $N_1 = \sum z_i$, $N_2 = 1 - N_1$, and

$$z_i = \begin{cases} 1 \text{ if } \boldsymbol{y}_i \text{ belongs to class } C = c_1 \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

The rank of $\widetilde{\boldsymbol{\Sigma}}_B$ is greater than one and this increases the number of non-zero eigenvalues beyond the upper limit set by LDA. □

The decomposition of $\boldsymbol{H}$ in (4.8) is only possible if the matrix is non-singular. This requires the number of training samples, N, to be larger than L, i. e. to be larger than $K-1$. Even if $N > L$, the LDA has a tendency to "overfit" when $N \lesssim 10L$ (see Ex. 8.5) [124, 320, 356]. One way to prevent overfitting is to regularize the solution by performing the eigenvalue decomposition on a modified matrix, $\widetilde{\boldsymbol{H}}$, as illustrated by the following example.

Example 4.13. Chemical Sensor Analysis: Principal Discriminant Analysis (PDA) for Small-Sample Size Problems [320]. The PDA is based on the eigenvalue of a modified matrix, $\widetilde{\boldsymbol{H}}$, defined by:

$$\widetilde{\boldsymbol{H}} = (1-\epsilon)\boldsymbol{\Sigma}_W^{-1}\boldsymbol{\Sigma}_B + \epsilon\boldsymbol{\Sigma}_G \ ,$$

where $\boldsymbol{\Sigma}_G = E\big((\boldsymbol{x}-\boldsymbol{\mu}_G)(\boldsymbol{x}-\boldsymbol{\mu}_G)^T\big) = \boldsymbol{\Sigma}_W + \boldsymbol{\Sigma}_B$ is the global covariance matrix and $\epsilon \in [0,1]$ is a regularization parameter. For $\epsilon > 0$ the LDA solution is regularized by incorporating information on $\boldsymbol{\Sigma}_G$. As in Ex. 4.12, the PDA increases the number of non-zero eigenvalues beyond the upper limit set by LDA. □

Outliers and atypical observations might have an undue influence on the results obtained, since the LDA is based on non-robust estimates of the population parameters. If outliers are thought to be present, the population parameters should be calculated using robust techniques (see Sect. 10.5).

In Table 4.7 we list some of the LDA variants which are in common use.

Table 4.7. LDA and its Variants

Name	$\boldsymbol{H}$
Original LDA	$\boldsymbol{\Sigma}_W^{-1}\boldsymbol{\Sigma}_B$.
PCA+LDA	$(\boldsymbol{V}^T\boldsymbol{\Sigma}_W\boldsymbol{V})^{-1}(\boldsymbol{V}^T\boldsymbol{\Sigma}_B\boldsymbol{V})$, where $\boldsymbol{V}$ denotes the principal component matrix [17]. A variant of the PCA+LDA algorithm is to replace the PCA with a kernel PCA.
PDA	$(\boldsymbol{\Sigma}+\lambda\boldsymbol{\Omega})^{-1}\boldsymbol{\Sigma}_B$, where $\boldsymbol{\Omega}$ is an appropriate regularization matrix [113].
Heteroscedastic LDA	$\boldsymbol{\Sigma}_W^{-1}\widetilde{\boldsymbol{\Sigma}}_B$. For a two-class problem, $\widetilde{\boldsymbol{\Sigma}}_B = \boldsymbol{\Sigma}_W^{1/2}(\boldsymbol{\Sigma}_W^{-1/2}(\boldsymbol{\mu}^{(1)}-\boldsymbol{\mu}^{(2)})(\boldsymbol{\mu}^{(1)}-\boldsymbol{\mu}^{(2)})^T\boldsymbol{\Sigma}_W^{-1/2} - 2\ln(\boldsymbol{\Sigma}_W^{-1/2}\boldsymbol{\Sigma}^{(1)}\boldsymbol{\Sigma}_W^{-1/2}) - 2\ln(\boldsymbol{\Sigma}_W^{-1/2}\boldsymbol{\Sigma}^{(2)}\boldsymbol{\Sigma}_W^{-1/2}))\boldsymbol{\Sigma}_W^{1/2}$ [182].
Generalized LDA	$\boldsymbol{\Sigma}_G^{+}\boldsymbol{\Sigma}_B$, where $\boldsymbol{\Sigma}_G^{+}$ denotes the pseudo-inverse of $\boldsymbol{\Sigma}_G$ [174].
Maximum Margin Criterion	$\boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_W$ [175].
Uncorrelated LDA	LDA in which the extracted features are statistically uncorrelated [344].
Maximum Uncertainty LDA	$\widetilde{\boldsymbol{\Sigma}}_W^{-1}\boldsymbol{\Sigma}_B$, where $\widetilde{\boldsymbol{\Sigma}}_W = (N-K)\boldsymbol{\Phi}\widetilde{\boldsymbol{\Lambda}}\boldsymbol{\Phi}^T$; $\boldsymbol{\Phi}$ and $\boldsymbol{\Lambda}$ =diag$(\lambda_1,\lambda_2,\ldots,\lambda_L)$ are, respectively, the eigenvectors and eigenvalues of $\boldsymbol{\Sigma}_W/(N-K)$; $\widetilde{\boldsymbol{\Lambda}}$ = diag$(\max(\lambda_1,\bar{\lambda}), \max(\lambda_2,\bar{\lambda}),\ldots,\max(\lambda_L,\bar{\lambda}))$ and $\bar{\lambda} = \sum_{l=1}^{L}\lambda_l/L$ [294].
2D-LDA	LDA on two-dimensional input matrices. Useful in image processing and computer vision applications [173].

4.6 Multiple Training Sets

A recent development in multi-sensor data fusion is *ensemble learning* (see Chapt. 13) in which we employ an ensemble, or collection, of *multiple* functions, or classifiers, $S_m, m \in \{1, 2, \ldots, M\}$, where each function S_m is learnt on its own training set D_m. Given a common training set D we may generate an ensemble of training sets, $D_m, m \in \{1, 2, \ldots, M\}$, which share the same common representational format by simply sub-sampling (with, or without, replacement) D.

Example 4.14. Bagging: Multiple Training Sets Sharing a Common Representational Format. Let D denote a training set of N measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. In *bagging* we generate multiple training sets $D_m, m \in \{1, 2, \ldots, M\}$ from a common training set D. Each training set D_m is formed by *bootstrapping* D, i. e. randomly sampling D with replacement and a uniform probability of selection. Fig. 4.7 illustrates the creation of a training set D_m by bagging (bootstrapping) a common training set $D = \{V, W, X, Y, Z\}$. □

In Table 4.8 we list some methods for sampling the common training set D.

An important sampling method is boosting which is described in detail in Chapt. 13.

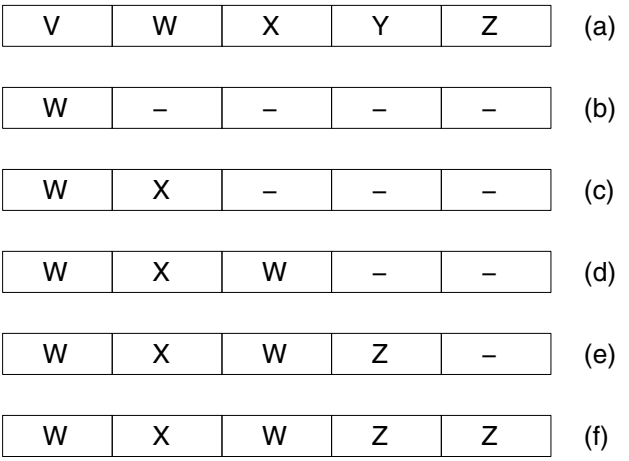

Fig. 4.7. Shows the creation of a training set D_m by bootstrapping a common training set $D = \{V, W, X, Y, Z\}$. **(a)** Shows the common training set D. **(b)** Shows the first element of D_m. This was chosen by randomly selecting one of the samples in D with a uniform probability of selection. **(c)**, **(d)** and **(e)** Show, respectively, the first two, three and four samples in D_m. **(f)** Shows the complete bootstrapped training set D_m. *Note.* The bootstrapped training set D_m contains, as expected, $\approx 63.2\%$ of the samples $\{V, W, X, Y, Z\}$ in D.

Table 4.8. Methods for Sampling a Common Training Set D

Method	Description
Cross-Validation	Partition the common training set D into M disjoint slices (similar to that used in cross-validation). Each classifier S_m is trained on a training set D_m, where D_m is the common training set D *less* the examples in the m slice.
Bagging (Bootstrap)	Perturb D by randomly sampling D with replacement. The sampling is made with a uniform probability random selection procedure. The entire procedure is repeated M times to create M different, although overlapping, training sets D_m. Each D_m contains N samples. On average each perturbed training set will have 63.2% of the samples in D, the rest being duplicates.
Wagging	Similar to bagging except that instead of sampling D using a uniform probability we use a non-uniform probability. The non-uniform probabilities are calculated by stochastically assigning a different weight to each sample in D and then normalizing the weights to one. Each training set D_k is generated using a different set of non-uniform probabilities.
Boosting	We use the classification results obtained with the mth classifier, S_m, to learn D_{m+1}. The classifier S_m is itself learnt on D_m. The training set D_{m+1} is created by re-sampling D such that samples which are misclassified by S_m have a higher chance of being chosen than samples which were correctly classified by S_m.
Multi-Boosting	Create a set of M data sets $D_m, m \in \{1, 2, \ldots, M\}$ by boosting a common training set D. Each boosted training set D_m is then bagged, or wagged, to generate M' new training sets $D_m^{(1)}, D_m^{(2)}, \ldots, D_m^{(M')}$.

Sometimes we require each training set D_m to have its own common representational format. This is a case of *multiple common representational formats*. Given a common training set D, we may generate an ensemble of training sets $D_m, m \in \{1, 2, \ldots, M\}$, where each D_m has a different common representational format, by applying a subspace technique to D and then subsampling (with, or without, replacement) the result. In Table 4.9 we list some variants of this technique.

Example 4.15. Input Decimated Samples [306]. Figure 4.8 shows the application of the method of input decimated samples on a common training set D which contains N two-dimensional samples $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. □

Table 4.9. Methods for Generating Multiple Common Representational Formats

Name	Description
Random Subspace Method (RSM)	Form training sets $D_m, m \in \{1, 2, \ldots, M\}$, by randomly selecting which dimensions to retain [118].
Random Projections	Form a random $L \times L'$ matrix $\boldsymbol{R}$, where L' is the number of dimensions which are to be retained. Each element in $\boldsymbol{R}$ is random number chosen from $\mathcal{N}(0, 1)$ distribution and normalizing the columns to one. We apply $\boldsymbol{R}$ to each sample $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, and thus obtain a new (L'-dimensional) sample $\boldsymbol{y}'_i = \boldsymbol{R}^T \boldsymbol{y}_i$ [81].
Input Decimated Samples	Divide samples in D according to their class variable. For each class, $C = c_k$, find the corresponding PCA transformation $\boldsymbol{U}$. Generate D_k by applying $\boldsymbol{U}$, or only the dominant columns in $\boldsymbol{U}$ (i. e. only eigenvectors with large eigenvalues), to all samples in D [306]. *Note*: The number of training sets is limited to the number of class K.
Rotation Forest	Similar to the method of Input Decimated Samples. However, in Rotation Forest the number of D_m is not limited to the number of classes present [267].
Skurichina-Duin	Find PCA transformation $\boldsymbol{U}$ on the common training set D. Then form the D_m as follows: D_1 is formed by applying the first K columns of $\boldsymbol{U}$ to D, D_2 is formed by applying the next K columns of $\boldsymbol{U}$ to D, etc., where K is a suitably chosen number [282].

In the table we assume that each training sample $\boldsymbol{y}_i$ in D is an L-dimensional vector: $\boldsymbol{y}_i = (y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(L)})^T$.

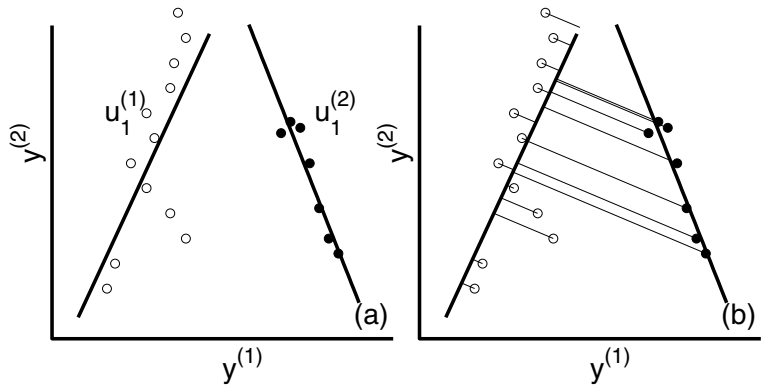

Fig. 4.8. (a) Shows a set of two-dimensional measurements $\boldsymbol{y}_i = (y_i^{(1)}, y_i^{(2)})^T, i \in \{1, 2, \ldots, N\}$, which belong to a common training set D. Measurements $i \in \{1, 2, \ldots, n\}$, belong to class $C = c_1$ (and are shown by open circles) and measurements $i \in \{n + 1, n + 2, \ldots, N\}$, belong to class $C = c_2$ (and are shown by closed circles). Overlaying the $\{\boldsymbol{y}_i\}$ are the dominant principal axis $\boldsymbol{u}_1^{(k)}$ for each class $C = c_k$. **(b)** Shows the creation of a training set D_1 by projecting the samples $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, onto the leading axis $\boldsymbol{u}_1^{(1)}$.

4.7 Software

BFD (Bayesian Fisher Discriminant). A matlab toolbox for performing linear and non-linear discriminant analysis.

DACE (Design and Analysis of Computer Experiments). A matlab toolbox for performing Kriging. The toolbox contains m-files for performing simple, ordinary and universal Kriging.

EFICA. A m-file for performing ICA.

FASTICA. A matlab toolbox for performing ICA.

LIBRA (Matlab Library for Robust analysis). A matlab toolbox for performing robust statistics (see Chapt. 10). The toolbox contains m-files on various robust subspace techniques.

MATLAB STATISTICAL TOOLBOX. The matlab statistical toolbox. The toolbox contains m-files on various subspace techniques.

NETLAB. A matlab toobox for performing neural network pattern recognition. The toolbox contains m-files on various subspace techniques including PPCA.

RADICAL (Robust Accurate Direct Independent Component Analysis). A matlab toolbox for performing ICA.

STPRTOOL (Statistical Pattern Recognition Tool). A matlab toolbox for performing statistical pattern recognition. The toolbox contains m-files on various subspace techniques, cross-validation and boosting.

4.8 Further Reading

The question of what constitutes an appropriate common representational format has been intensely investigated for mobile robots by Thrun [297, 298], Durrant-Whyte [68, 69] and others. For target tracking the choice of representational format has been discussed in [24, 215] and specific references on debiased Cartesian coordinates are given in [66, 86, 208]. Recent articles on the polar maps, or Bull's-eye images in coronary artery disease include [180, 217]. The classic reference on Kriging is [52]. Another more modern reference is [273]. Modern reviews of the brain atlas are given in [197, 296]. A large literature exists on the different subspace methods. Pointers to the literature are given in Table 4.5.

5

Spatial Alignment

5.1 Introduction

The subject of this chapter is *spatial alignment*. This is the conversion of local spatial positions to a common coordinate system and forms the first stage in the formation of a common representational format. To keep our discussion focused we shall limit ourselves to two-dimensional (x, y) image sensors. In this case the process of spatial alignment is more commonly referred to as *image registration*.

5.2 Image Registration

Let I_1 and I_2 denote two input images, where $I_m(x, y), m \in \{1, 2\}$, denotes the gray-level of I_m at the pixel (x, y). The images are referred to, respectively, as the *reference* image and the *test* image. The registration process is defined as finding the transformation T which "optimally" maps coordinates in I_1 to coordinates in I_2. Thus, given a pixel (x, y) in I_1, the corresponding location in I_2 is $(x', y') = T(x, y)$. Often a transformed point (x', y') does not fall on a pixel in I_2. In this case, strictly speaking, we have no gray-level for the point (x', y'). To handle such situations we resample/interpolate the test image so that a gray-level is defined for all points (x', y').

A popular technique for multi-sensor registration is to use an information theoretic topic known as "mutual information". The basic concept behind this approach is to find a transformation, which when applied to the test image, will maximize the mutual information between the test and reference images.

The success of mutual information image registration lies in its inherent simplicity. It makes few assumptions regarding the relationship that exists between different images. It only assumes a *statistical* dependence. The idea is that although different sensors may produce very different images, since they are imaging the same underlying scene, there will exist some inherent mutual information between the images.

Example 5.1. Medical Image Registration: A Thought Experiment [291]. Consider a pair of medical images from different modalities (e. g. Magnetic Resonance and Computerized Tomography) that are in perfect alignment. At many corresponding spatial locations in the two images, the associated gray-levels will exhibit a consistent mapping. In other words, there will exist some kind of global relation, or mapping, between the gray-levels of the reference image with the gray-levels of the test image. Whatever the mapping is (one-to-one, one-to-many, many-to-one or many-to-many) the statistical dependency between the images is strong and the mutual information measure has a high value.

Suppose now that we move out of registration by gradually deforming the test image. The mapping between the gray-levels of the two images will degrade, i. e. will become less consistent, and the value of the mutual information will decrease. In the extreme case, the two images are completely independent of one another, and the mutual information will take a value of zero which signifies that it is not possible to predict any part of one image from the information stored in the other. □

5.2.1 Mutual Information

[1] The mutual information between two random variables u and v is given by

$$MI(u,v) = H(u) + H(v) - H(u,v) \tag{5.1}$$

where $H(u)$ and $H(v)$ are the marginal entropies of the two variables u and v and $H(u,v)$ is their joint entropy.

In the continuous domain, the (differential) entropy of a random variable u and the joint (differential) entropy of two random variables u and v are defined as

$$H(u) = -\int_{-\infty}^{\infty} p(u)\log_2 p(u)du \;, \tag{5.2}$$

$$H(u,v) = -\int_{-\infty}^{\infty} p(u,v)\log_2 p(u,v)dudv \;, \tag{5.3}$$

where $p(u)$ and $p(u,v)$ are, respectively, the marginal and joint probability density functions of the random variables u and v. In this case the corresponding formula for the mutual information is

$$\begin{aligned} MI(u,v) &= -\int_{-\infty}^{\infty} p(u)\log_2\big(p(u)\big)du - \int_{-\infty}^{\infty} p(v)\log_2\big(p(v)\big)dv \\ &\qquad - \int_{-\infty}^{\infty} p(u,v)\log_2\big(p(u,v)\big)dudv \;, \\ &= \int\int p(u,v)\log_2\frac{p(u,v)}{p(u)p(v)}dudv \;. \end{aligned} \tag{5.4}$$

[1] The second half of this section contains advanced material. It assumes the reader has some familiarity with the concept of a probability density function and its estimation. This material is not, however, essential for what follows and it may be left for a second reading.

Mathematically, the mutual information can be related to the conditional entropies of u given v and v given u:

$$MI(u,v) = H(u) - H(u|v) = H(v) - H(v|u) \; . \tag{5.5}$$

As $H(u)$ is a measure of the amount of information about u, then $H(u|v)$ represents the amount of uncertainty left in u when v is known. Thus, mutual information can be interpreted as the reduction in the uncertainty of u when v is known, i. e. the amount of information that v contains about u. The same analogy can be drawn for the reverse case: a measure of the amount of information that u contains about v.

Given N simultaneous measurements $(u_i, v_i), i \in \{1, 2, \ldots, N\}$, the most straightforward, and widely used, approach to calculate the mutual information $MI(u,v)$ is to use a frequency histogram-based technique. We divide the (u,v) into K vertical columns $c_k, k \in \{1, 2, \ldots, K\}$, and L horizontal rows $r_l, l \in \{1, 2, \ldots, L\}$. If m_k and n_l are, respectively, the number of simultaneous measurements which lie in the kth column c_k and in the lth row r_l, then the mutual information is defined as

$$MI(u,v) \approx \log_2 N + \frac{1}{N} \sum_{kl} m_{kl} \log_2 \frac{m_{kl}}{m_k n_l} + \Delta MI \; , \tag{5.6}$$

where m_{kl} is the number of simultaneous measurements (u_i, v_i) which lie in the intersection of c_k and r_l, i. e. in the klth bin B_{kl} (see Fig. 5.1) and ΔMI is

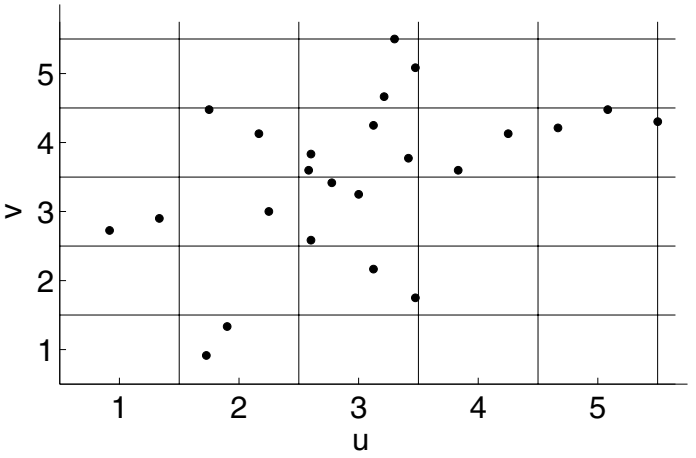

Fig. 5.1. Shows the division of the (u,v) space into KL rectangular bins, where the (k,l)th bin B_{kl} is formed by the intersection of the kth column c_k and the lth row r_l. In the figure the number of $z_i = (u_i, v_i)$ points which lie in the column c_3 and in the row r_4 are, respectively, $m_3 = 12$ and $n_4 = 11$. The number of points which lie in the intersection of c_3 and r_2 (i. e. in the bin $B_{3,4}$) is $m_{3,4} = 4$.

a *systematic* error correction and arises because we are using a finite number of measurements. If the bins are of equal size, then to first order

$$\Delta MI(u,v) = \frac{M'_B - M'_c - M'_r + 1}{2N} \,, \tag{5.7}$$

where M'_B, M'_c and M'_r are, respectively, the number of bins, columns and rows which are not empty.

Instead of using fixed intervals to divide the (u,v) space into discrete bins, we may improve the $MI(u,v)$ estimate by using an adaptive partitioning method. For example, we may construct a hierarchy of partitions which recursively divide the (u,v) plane into smaller and smaller bins [54, 55, 90]. The idea is that regions in the (u,v) plane, where the simultaneous measurements (u_i, v_i) are uniformly distributed, cease to contribute further to the estimated mutual information under further partitioning. For these regions, there is therefore no point in subdividing them further. The result is that dense regions containing many simultaneous measurements are covered with small bins, while regions containing few, or no, simultaneous measurements are covered with large bins. Although these estimators are much better than estimators using fixed bin size, they still suffer from systematic errors. These are the result, on the one hand, of approximating $MI(x,y)$ by a finite sum and, on the other hand, by approximating logarithms of probabilities by logarithms of frequency ratios. The later can be minimized by using corrections for finite m_k, m_{kl} and n_l. These corrections are in the form of asymptotic series which diverge for finite N but whose first two terms improve the estimates in typical cases. For a full discussion concerning these issues see [55, 161].

The following example illustrates the principles underlying the implementation of a variable partitioning MI algorithm.

Example 5.2. Darbellay-Tichavsky Variable Partitioning Algorithm [54]. In [54] the authors have implemented a variable partitoning MI algorithm as follows. Suppose at some stage in the above recursive procedure we have a rectangular bin B_{kl} which contains $m_{kl} > 4$ pairs of measurements (see Fig. 5.2). The corresponding vertical column and horizontal row contain, respectively, m_k and n_l measurement pairs. Tentatively, we divide B_{kl} into four rectangular sub-bins containing $m_{kl}^{(1)}, m_{kl}^{(2)}, m_{kl}^{(3)}$ and $m_{kl}^{(4)}$ measurement pairs. The corresponding vertical columns contain $m_k^{(1)}$ and $m_k^{(2)}$ measurement pairs and the corresponding hoizontal rows contain $n_l^{(1)}$ and $n_l^{(2)}$ measurement pairs. The division into four sub-bins is chosen such that $|m_k^{(1)} - m_k^{(2)}|$ and $|n_l^{(1)} - n_l^{(2)}|$ are as close to zero as possible. If the resulting sub-bins satisfy the equality:

$$m_{kl}^{(1)} \approx m_{kl}^{(2)} \approx m_{kl}^{(3)} \approx m_{kl}^{(4)} \approx \frac{m_{kl}}{4} \,,$$

then the division of B_{kl} is *not* carried out. Instead the bin remains unchanged. If we use a 5% significance test to measure the above approximate equalities,

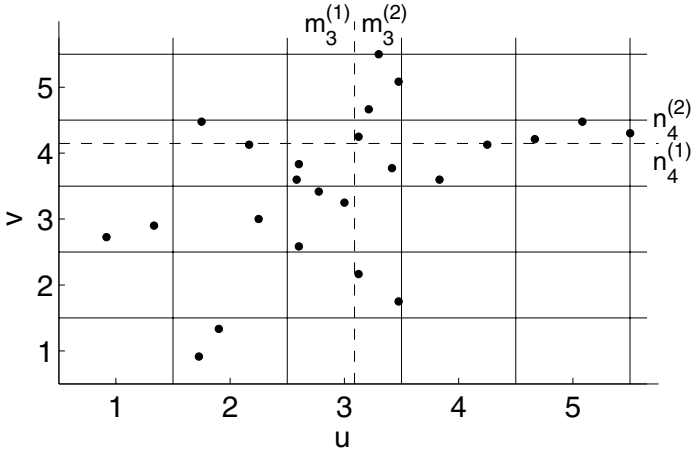

Fig. 5.2. Shows the division of the bin $B_{3,4}$ into four rectangles. The division is made by dividing the column c_3 into two vertical sub-columns (which contain, respectively, $m_3^{(1)} = 5$ and $m_3^{(2)} = 7$ sample points) and by dividing the row r_4 into two horizontal sub-rows (which contain, respectively, $n_4^{(1)} = 4$ and $n_4^{(2)} = 6$ sample points).

then the above equation becomes

$$\frac{4}{m_{kl}} \sum_{i=1}^{4} \Big(m_{kl}^{(i)} - \frac{m_{kl}}{4} \Big)^2 < 7.81 \; .$$

This algorithm has been implemented in an efficient matlab m-file. For details see Sect. 5.7. □

Instead of using discrete bins we may use continuous bins, or *kernels*, to calculate the probability densities $p(u)$, $p(v)$ and $p(u, v)$ which appear in (5.4). This is known as kernel, or Parzen-window, density estimation [276, 278] (see Sect. 12.3.3).

Both the histogram and the kernel methods are based on the idea of a "bin". As an alternative, we may calculate the mutual information $MI(u, v)$ using a "binless" approach [161]: If $z_i = (u_i, v_i)$, then we define the distance between two points, z_i and z_j, as

$$|z_i - z_j| = \max(|u_i - u_j|, |v_i - v_j|) \; , \tag{5.8}$$

where $|u_i - u_j|$ is the distance between u_i and u_j and $|v_i - v_j|$ is the distance between v_i and v_j. Let Δ_i be the Euclidean distance from z_i to its k-nearest neighbour and let δ_i and ϵ_i be the corresponding distances projected along the u and v axes (Fig. 5.3). Then, the estimate for the mutual information is

$$MI(u, v) = \psi(k) - \frac{1}{k} - \langle \psi(m_i) + \psi(n_i) \rangle + \psi(N) \; , \tag{5.9}$$

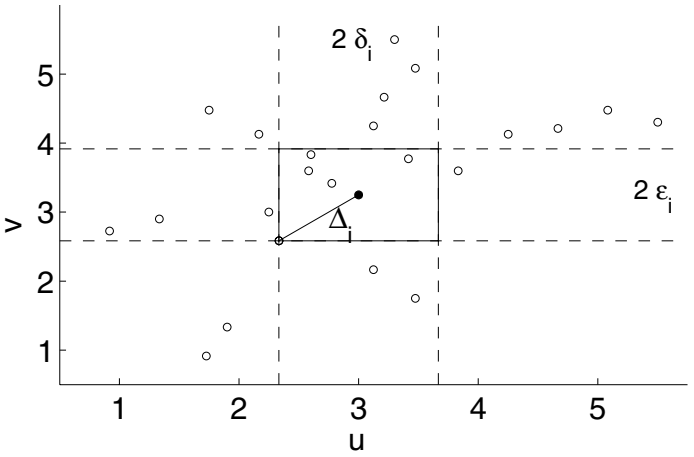

Fig. 5.3. Shows the calculation of δ_i and ϵ_i for the point $z_i = (u_i, v_i)$ (denoted by a filled circle). The rectangle surrounding z_i contains $k = 5$ points (apart from z_i). We join z_i with the fifth point which is furthest from z_i with a straight line whose (Euclidean) length is Δ_i. Then the distance δ_i and ϵ_i are, respectively, the projections of Δ_i along the u and v axes.

where m_i and n_i are, respectively, the number of points with $\|u_i - u_j\| \leq \delta_i$ and $\|v_i - v_j\| \leq \epsilon_i$, ψ is the digamma function[2] and $\langle\ \rangle$ denotes an average over all $i, i \in \{1, 2, \ldots, N\}$.

5.3 Resample/Interpolation

As we pointed out in Sect. 5.1, the resample/interpolation process is a crucial part of any registration scheme because it uniquely defines the behaviour of the transformation T when the samples do not fall on a pixel. Two detailed reviews on modern image interpolation methods are [170] and [290]. In image interpolation we reconstruct a two-dimensional continuous image $I(x, y)$ from its discrete pixel values $I(i, j)$. Thus the amplitude at the position (x, y) must be estimated from its discrete neighbours. This is often modeled as a convolution of the discrete image samples with a continuous two-dimensional impulse response $h(x, y)$:

$$I(x, y) = \sum_i \sum_j I(i, j) h(x - i, y - j)\ . \tag{5.10}$$

[2] Values of the digamma function are easily calculated using the recursion relation $\psi(x + 1) = \psi(x) + 1/x$, where $\psi(1) \approx -0.5772156 \cdots$.

Usually, symmetrical and separable interpolation kernels are used to reduce the computational complexity

$$h(x, y) = h(x)h(y) \ . \tag{5.11}$$

In Table 5.1 we list several popular kernel functions $h(x)$ used for image interpolation [170]. Apart from the fixed coefficient kernels listed in Table 5.1 we may also use the spatial Kriging estimators listed in Table 4.3.

In the case of image registration using mutual information, special attention must be given to the process of resampling/interpolation, because the process introduces new gray-levels which may interfere with the computation of the MI [245]. For example, in applications involving remote sensing, nearest neighbour interpolation is often found to perform better than either linear, or cubic convolution, interpolation [43].

Example 5.3. Entropy Changes and Interpolation [291]. Consider a 10×10 binary image centered on a step function. The linear interpolation of this image after a very small sub-pixel translation across the edge introduces 10 new members with a third gray-level. Before translation the entropy of the

Table 5.1. Kernel Functions $h(x)$ for Image Interpolation.

Name	$h(x)$
Nearest Neighbour	1 if $0 \leq \lvert x\rvert < \frac{1}{2}$; otherwise 0.
Linear	$1 - \lvert x\rvert$ if $0 \leq \lvert x\rvert < 1$; otherwise 0.
Quadratic	$\begin{cases} -2\lvert x\rvert^2 + \frac{1}{4} & \text{if } 0 \leq \lvert x\rvert < \frac{1}{2} \ , \\ \lvert x\rvert^2 - \frac{5}{2}\lvert x\rvert + \frac{3}{8} & \text{if } \frac{1}{2} \leq \lvert x\rvert < \frac{3}{2} \ , \\ 0 & \text{otherwise.} \end{cases}$
Cubic ($N = 4$)	$\begin{cases} (a+2)\lvert x\rvert^3 - (a+3)\lvert x\rvert^2 + 1 & \text{if } 0 \leq \lvert x\rvert < 1 \ , \\ a\lvert x\rvert^3 - 5a\lvert x\rvert^2 + 8a\lvert x\rvert - 4a & \text{if } 1 \leq \lvert x\rvert < 2 \ , \\ 0 & \text{otherwise} \ , \end{cases}$ where a can take the values $a = -\frac{1}{2}, -\frac{2}{3}, -\frac{3}{4}, -1, -\frac{4}{3}$.
Cubic ($N = 6$)	$\frac{1}{6}\begin{cases} 6\lvert x\rvert^3 - 11\lvert x\rvert^2 + 1 & \text{if } 0 \leq \lvert x\rvert < 1 \ , \\ -3\lvert x\rvert^3 + 16\lvert x\rvert^2 - 27\lvert x\rvert + \frac{14}{5} & \text{if } 1 \leq \lvert x\rvert < 2 \ , \\ \lvert x\rvert^3 - 8\lvert x\rvert^2 + 21\lvert x\rvert - 18 & \text{if } 2 \leq \lvert x\rvert < 3 \ , \\ 0 & \text{otherwise.} \end{cases}$
Cubic ($N = 8$)	$\frac{1}{56}\begin{cases} 67\lvert x\rvert^3 - 123\lvert x\rvert^2 + 1 & \text{if } 0 \leq \lvert x\rvert < 1 \ , \\ -33\lvert x\rvert^3 + 177\lvert x\rvert^2 - 300\lvert x\rvert + 156 & \text{if } 1 \leq \lvert x\rvert < 2 \ , \\ 9\lvert x\rvert^3 - 75\lvert x\rvert^2 + 204\lvert x\rvert - 180 & \text{if } 2 \leq \lvert x\rvert < 3 \ , \\ -3\lvert x\rvert^3 + 33\lvert x\rvert^2 - 120\lvert x\rvert + 164 & \text{if } 3 \leq \lvert x\rvert < 4 \ , \\ 0 & \text{otherwise.} \end{cases}$
Cubic (Ref. [214])	$\frac{1}{42}\begin{cases} 49\lvert x\rvert^3 - 2\lvert x\rvert^2 + 224 & \text{if } 0 \leq \lvert x\rvert < 1 \ , \\ 7\lvert x\rvert^3 + 2\lvert x\rvert^2 - 2\lvert x\rvert + 224 & \text{if } 1 \leq \lvert x\rvert < 2 \ , \\ 0 & \text{otherwise.} \end{cases}$

The above formulas in the table assume x and y are given in units of the sampling interval.

image is $H = 1$. After translation the entropy is $H = 1.03 + 0.33$, the last value being the contribution of the new gray-level which represents 10% of all pixels. □

5.4 Pairwise Transformation T

The (pairwise) transformation $T : (x', y') = T(x, y)$ is a mathematical relationship that maps a spatial location (x, y) in one image to a new location, (x', y'), in another image. It arises in many image analysis problems, whether we wish to remove optical distortions introduced by a camera or a particular viewing perspective, or to register an image with a given map or template, or simply to align two input images. The choice of transformation is always a compromise between a smooth distortion and a distortion which achieves a good match. One way to ensure smoothness is to assume a parametric form of low-order for the transformation [104, 359] such as that given in Table 5.2. In most applications the transformation T is chosen on the grounds of mathematical convenience. However, sometimes, we may have information regarding the physical processes which govern the formation of the pictures. In this case we may use physical arguments in order to *derive* the transformation T.

Example 5.4. Transformation for Retinal Images [35, 36]. Let T denote the pairwise transformation between a pair of retinal images. If we model the retina as a quadratic surface and assume a rigid weak-perspective transformation between the two images, then the corresponding transformation is

$$x' = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \ ,$$
$$y' = a_7x^2 + a_8xy + a_9y^2 + a_{10}x + a_{11}y + a_{12} \ .$$

The assumptions made in deriving this transformation may be justified on physical grounds [35, 36]. Alternatively we may "derive" the transformation by performing a Taylor expansion of the general polynomial transformation in x and y and only retaining terms upto, and including, second order. □

Table 5.2. Spatial Transformations $T : (x', y') = T(x, y)$

Name	Formula
Translation	$x' = x + a_1, y' = y + a_2.$
Similarity	$x' = a_1x + a_2y + a_3,\ y' = -a_2x + a_1y + a_4.$
Affine	$x' = a_1x + a_2y + a_3,\ y' = a_4x + a_5y + a_6.$
Perspective	$x' = (a_1x + a_2y + a_3)/(a_7x + a_8y + 1),$
	$y' = (a_4x + a_5y + a_6)/(a_7x + a_8y + 1).$
Polynomial	$x' = \sum a_{ij}x^iy^j,\ y' = \sum b_{ij}x^iy^j.$

In some multi-sensor data fusion applications, the images undergo local deformations. In this case, we cannot describe the alignment of two images using a single low-order transformation. In this case we often use instead a "non-rigid" transformation, which we write as a sum of a low-order global transformation $T_{\text{global}}(x, y)$ and a local transformation $T_{\text{local}}(x, y)$:

$$(x', y') = T(x, y) = T_{\text{global}}(x, y) + T_{\text{local}}(x, y) \ , \tag{5.12}$$

where the parameters of T_{local} change with (x, y). The following example illustrates such a transformation.

Example 5.5. Spatial Alignment in Cardiac MR Images [241, 242]. Magnetic imaging (MR) of the cardiovascular system is playing an increasingly important role is the diagnosis and treatment of cardiovascular diseases. During a cardiac cycle the heart changes size and shape. To perform spatial alignment of two MR images taken at different times during the cardiac cycle we therefore require a transformation $T(x, y)$ which includes both a global transformation and a local transformation. The purpose of the local transformation $T_{\text{local}}(x, y)$ is to correct for changes in the size and shape of the heart. We often use a product of one-dimensional cubic splines for this purpose [241]. □

5.5 Image Fusion

Image fusion is concerned with combining information from multiple *spatially registered* images. The following two examples illustrate two different approaches to performing image fusion.

Pan-Sharpening

Pan-Sharpening is an image fusion technique which is designed to overcome the limitations of present-day image sensors. In capturing a high-resolution multi-spectral image, we can either use a panochromatic sensor or a multi-spectral sensor. The images taken with the panochromatic sensor have high spatial resolution but low spectral resolution while the images taken with the multi-spectral sensor have have high spectral resolution but low spatial resolution. In [204] the authors use *co-Kriging* (see Sect. 4.3) to create a new image characterized by a high-spatial resolution and a high-spectral resolution.

Note: The method of co-Kriging requires the images to be spatially registered to a very high degree of accuracy. In practice this method of image fusion is limited, therefore, to satellite images such as LANDSAT.

Example 5.6. Image Fusion: Pan-Sharpening Using Co-Kriging [204, 237]. In pan-sharpening we have two types of variables: high spectral resolution data and high spatial resolution data. We represent these two types of variables as y and z respectively.

In co-Kriging the interpolated multi-spectral image at some point $\boldsymbol{u}_0$ is given by a linear combination of the high spectral resolution image y and the high spatial resolution image z:

$$\hat{y}_0 = \hat{y}(\boldsymbol{u}_0) = \sum_{i=1}^{K} \lambda_i y_i + \sum_{j=1}^{L} \nu_j z_j \ ,$$

where $y_i, i \in \{1, 2, \ldots, K\}$, are the high spectral resolution gray-levels at K nearby locations and $z_j, j \in \{1, 2, \ldots, L\}$, are the high spatial resolution gray-levels at L nearby locations.

In ordinary co-Kriging the weights λ_k and ν_l are chosen to give the best (in the least square sense) unbiased solution subject to the following constraints:

$$\sum_{i=1}^{K} \lambda_i = 1 \ ,$$
$$\sum_{j=1}^{L} \nu_j = 0 \ .$$

The solution is

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_K \\ \nu_1 \\ \vdots \\ \nu_L \\ \mu_0 \\ \mu_1 \end{pmatrix} = \begin{pmatrix} \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_1) & \ldots & \Sigma(\boldsymbol{u}_K, \boldsymbol{u}_1) & \Sigma(\boldsymbol{v}_1, \boldsymbol{u}_1) & \ldots & \Sigma(\boldsymbol{v}_L, \boldsymbol{u}_1) & 1 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_K) & \ldots & \Sigma(\boldsymbol{u}_K, \boldsymbol{u}_K) & \Sigma(\boldsymbol{v}_1, \boldsymbol{u}_K) & \ldots & \Sigma(\boldsymbol{v}_L, \boldsymbol{u}_K) & 1 & 0 \\ \Sigma(\boldsymbol{u}_1, \boldsymbol{v}_1) & \ldots & \Sigma(\boldsymbol{u}_K, \boldsymbol{v}_1) & \Sigma(\boldsymbol{v}_1, \boldsymbol{v}_1) & \ldots & \Sigma(\boldsymbol{v}_L, \boldsymbol{v}_1) & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \Sigma(\boldsymbol{u}_1, \boldsymbol{v}_L) & \ldots & \Sigma(\boldsymbol{u}_K, \boldsymbol{v}_L) & \Sigma(\boldsymbol{v}_1, \boldsymbol{v}_L) & \ldots & \Sigma(\boldsymbol{v}_L, \boldsymbol{v}_L) & 1 & 0 \\ 1 & \ldots & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & \ldots & 0 & 1 & \ldots & 1 & 0 & 0 \end{pmatrix}^{-1}$$
$$\times \begin{pmatrix} \Sigma(\boldsymbol{u}, \boldsymbol{u}_1) \\ \vdots \\ \Sigma(\boldsymbol{u}, \boldsymbol{u}_K) \\ \Sigma(\boldsymbol{u}, \boldsymbol{v}_1) \\ \vdots \\ \Sigma(\boldsymbol{u}, \boldsymbol{v}_L) \\ 1 \\ 0 \end{pmatrix} ,$$

where μ_0 and μ_1 are two Lagrange multiplers. *Note.* It may help to understand this equation by interpreting the co-Kriging procedure as the application of a low-pass filter to $y(\boldsymbol{u})$ and a high pass filter to $z(\boldsymbol{u})$ [237]. □

Fusion of PET and MRI Images

The fusion of PET and MRI images is concerned with image fusion between complementary sensors in which the output is a decision surface or image. In the decision image each pixel gray-level represents a given decision. In forming the decision surface we reduce the effects of spatial misregistration by using a low resolution polar map or common representational format (see Ex. 4.8).

Example 5.7. Multi-Modal Cardiac Imaging [16]. Diseases of the heart are characterized by a reduced flow of blood to the heart muscles. This reduced flow can be studied using different imaging sensors or modalities, each of which gives a specific view of the phenomenom. For example, the reduced flow of blood causes a deterioration of the myocardial perfusion which can be analyzed with positron emission tomography (PET) or with magnetic resonance imaging (MRI). Other changes which result from the reduced flow are metabolic changes in the heart tissues which can be highlighted using a PET sensor and the reduced capacity of the heart to eject blood into the body which can be monitored using an ultrasonic (US) sensor.

To assess the myocardial viability of the patient and to determine the proper therapeutic action it is important to fuse together images of all of the above phenomenom. In this example we consider the fusion of PET and MRI images [16]. The images are first converted to low-resolution polar maps which are then spatially registered. Parameters are extracted from each sector in the co-aligned PET and MRI polar maps which are then fused together using a single fusion cell F (Fig. 5.4). The fusion in F is performed using the fuzzy logic truth table shown in Fig. 5.5. The input to the table are two parameters: Inoptropic Reserve (IR) and 18-fluorodeoxyglucose (FDG) uptake and the

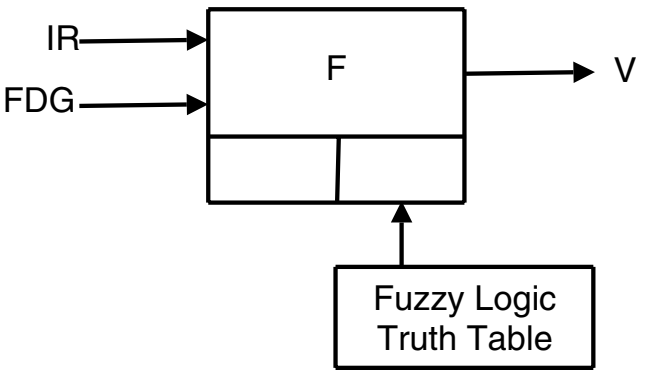

Fig. 5.4. Shows the fusion of the MRI and PET images using a single fusion cell F. Input to F is the Inoptroic Reserve, IR, which is derived from the MRI image and the 18-fluorodeoxyglucose, FDG, which is derived from the PET image. The output from F is the myocardial viability V. The fusion is performed using a fuzzy logic truth table which is supplied as external knowledge.

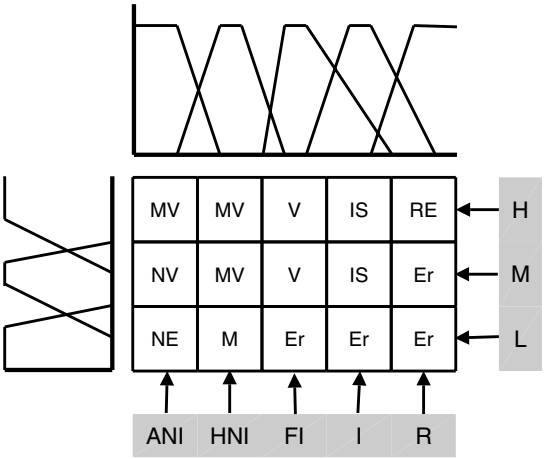

Fig. 5.5. Shows the fuzzy logic truth table used to fuse the inoptropic reserve (IR) and the 18-fluorodeoxyglucose (FDG) parameters. Inputs to the table are IR (horizontal axis) which is derived from the MRI image and FDG (vertical axis) which is derived from the PET image. The squares in the table contain the linguistic values of output variable V: Necrosis (NE), Maimed (M), Metabolic Viable (MV), Viable (V), Ischemic (IS), Remote (RE) and Error (Er), where the error Er is used to express an impossible combination of input values. The graph at the top of the figure shows the five membership functions associated with the linguistic values (ANI, HNI, FI, I, R) of the IR parameter. The graph at the side of the figure shows the three membership functions associated with the linguistic values (L, M, L) of the FDG parameter.

output of the table is the myocardial viability (V). All three parameters are normalized by converting them to *linguistic values* using fuzzy membership functions similar to that shown in Fig. 5.5.

The output is a decision image in which each pixel gray-level corresponds to a linguistic myocardial viability value. □

5.6 Mosaic Image

Thus far we have considered the problem of registering a pair of images. In some applications we are interested in building a single panoramic or "mosaic" image I^* from multiple images $I_m, m \in \{1, 2, \ldots, M\}$. To do this we need to find functions T_m which transform each input image I_m onto the image I^*.

Building a mosaic image from a sequence of partial views is a powerful means of obtaining a broader view of a scene than is available with a single view. Research on automated mosaic construction is ongoing with a wide range of different applications.

Example 5.8. Mosaic Fingerprint Image [139]. Fingerprint-based verification systems have gained immense popularity due to the high level of uniqueness attributed to fingerprints and the availability of compact solid-state fingerprint sensors. However, the solid-state sensors sense only a limited portion of the fingerprint pattern and this may limit the accuracy of the user verification. To deal with this problem we may construct a mosaic fingerprint image from multiple fingerprint impressions. □

Example 5.9. Mosaic Image of the Retina [35, 36]. One area in which mosaic imagess are particularly valuable is in the diagnosis and treatment of diseases of the retina. A seamless mosaic image which is formed from multiple fundus camera images aids in the diagnosis and provides a means for monitoring the progression of different diseases. It may also be used as a spatial map of the retina during surgical treatment. □

At first sight we may assume that the ability to register a pair of images is sufficient to solve the problem of forming a mosaic of the entire scene from multiple partial views. Theoretically, if one image can be established as an "anchor image" I_0 on which to base the mosaic image I^*, then the transformation of each remaining images onto this anchor may be estimated using pairwise registration, and the transformed images may be combined. Unfortunately, in practice, this approach does not work for two reasons:

1. Some images may not overlap with the anchor image at all. This makes a direct computation of the transformation impossible. In other cases, images may have insufficient overlap with the anchor image to compute a stable transformation. The straightforward solution is to compose transformations using an "intermediate" image. This is problematic, however, since repeated application of the transformation will often magnify the registration error.
2. The transformations T_m may be mutually inconsistent. This may happen even if all the image-to-anchor transformations have been accurately estimated. The reason for this is as follows: Although each image may individually register accurately with the anchor image and the non-anchor images may register accurately with each other, this does not ensure that the transformations onto the anchor image are *mutually consistent*.

One approach to solving this problem is to *jointly* estimate all of the image-to-mosaic transformations T_m [35, 36].

5.7 Software

MATLAB IMAGE PROCESSING TOOLBOX. The matlab image processing toolbox. The toolbox contains various m-files for performing image registration, resampling and interpolation.

MILCA (Mutual Information Based Least Dependent Component Analysis). A matlab toolbox for performing independent component analysis. It contains an m-file for calculating $MI(u, v)$ using a binless method.

MUTIN. A matlab m-file for calculating the mutual information $MI(u, v)$ using the method of variable partitioning.

5.8 Further Reading

The subject of image registration has been intensely investigated for many years. A modern review of image warping is [104]. Specific references on the use of mutual information for image registration are [43, 188, 246, 315, 326]. The calculation of mutual information has been considered by many authors including [54, 55, 161, 284]. References on non-separable interpolation methods include [262].

6

Temporal Alignment

6.1 Introduction

The subject of this chapter is *temporal alignment*, or registration, which we define as the transformation $T(t)$ which maps local sensor observation times t to a common time axis t'. Temporal alignment is one of the basic processes required for creating a common representational format. It often plays a critical role in applications involving in many multi-sensor data fusion applications. This is especially true for applications operating in real-time (see Sect. 2.4.2).

The following example illustrates the use of temporal registration in medical imaging.

Example 6.1. Cardiac MR Image Sequences [242]. Cardiovascular diseases are a very important cause of death in the developed world. Their early diagnosis and treatment is crucial in order to reduce mortality and to improve patient's quality of life. The recent advances in the development of cardiac imaging modalities have led to an increased need for cardiac registration, whose aim is to bring K sequences of cardiac images acquired from the same subject, or from the same subject at different times, into the same spatial-temporal coordinate system. To do this we must perform temporal registration which corrects for any *temporal misalignment* caused by different acquisition parameters, different lengths of cardiac cycles and different motion patterns of the heart. □

Let $O = \langle E, \boldsymbol{u}, t, \boldsymbol{y}, \Delta\boldsymbol{y} \rangle$ denote a given sensor observation. Then, strictly speaking the transformation $T(t)$ is a function of the position $\boldsymbol{u}$ and the time t. However, it is common practice to assume (at least as a first approximation) that $T(t)$ is independent of $\boldsymbol{u}$ [241].

The following example illustrates an application in which perform temporal alignment on sequences of input images. In this application we reduce each input image to a single point in space which reduces the errors involved in assuming $T(t)$ is independent of $\boldsymbol{u}$.

Example 6.2. Visualization of Dysphagia (Disorders Associated with Swallowing) [279]. We consider an application in which we form a high resolution video sequence of a patient swallowing. In this application, a patient repeatedly swallows. For a given swallow S, a magnetic resonance imaging (MRI) camera records the swallow in a sequence of MRI images $I_i, i \in \{1, 2, \ldots, N\}$, taken at times t_i, where $t_i < t_{i+1}$. For each sequence (swallow) we segment the bolus and track its path in the space-time volume of the swallow. We do this by extracting the centroid, $(\bar{x}, \bar{y})$, of the segmented region from each MRI image and creating a path of the centroid motion for each swallow (see Fig. 6.1).

After performing temporal registration, we form a high resolution video of the patient swallowing by combining the multiple swallows. The temporal registration is performed by matching the spatial-temporal paths of the bolus in multiple swallows. □

To keep our discussion focused we shall limit ourselves to the temporal alignment of a time series, $\boldsymbol{P}$, to a common time-axis t', where $\boldsymbol{P} = (P_1, P_2, \ldots, P_N)^T$ consists of M sensor observations, $P_i = \langle E_i, *, t_i, y_i, * \rangle, i \in \{1, 2, \ldots, M\}$, made with the same sensor and ordered in time, i. e. $t_i < t_{i+1}$. Furthermore, we shall assume (as is common in many multi-sensor data fusion systems) that we do not have an independent time-axis t'. Instead, we shall define the time-axis t' by a second time series $\boldsymbol{Q}$, where $\boldsymbol{Q}$ consists of N sensor observations, $Q_j = \langle E_j, *, t'_j, y'_j, * \rangle, j \in \{1, 2, \ldots, N\}$.

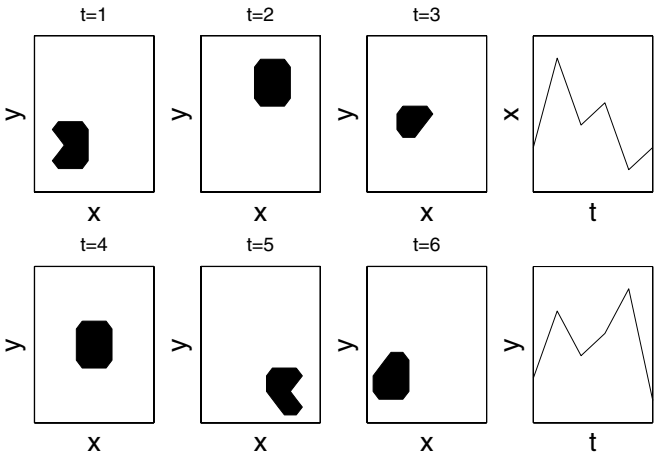

Fig. 6.1. Shows six two-dimensional images $I_i, i \in \{1, 2, \ldots, 6\}$, taken at times $t = \{1, 2, \ldots, 6\}$. In each image we segment the bolus (which is shown as a dark irregularly shaped area). To the right of the images I_i we show the x and y position of centroid of the bolus as a function of the time t.

6.2 Dynamic Time Warping

Dynamic Time Warping (DTW) [255, 259] is a general technique for performing temporal alignment between two time sequences, $\boldsymbol{P}$ and $\boldsymbol{Q}$. DTW was originally developed for use in speech recognition. However, recently, it has been used in many different areas, including data mining, pattern and shape recognition and biometric identification.

DTW finds the *optimal* alignment between two time series $\boldsymbol{P}$ and $\boldsymbol{Q}$ in the sense that it minimizes the sum of the local distances $d(i, j)$ between the aligned observation pairs (P_i, Q_j). In many applications we define $d(i, j)$ as the square of the Euclidean distance:

$$d(i, j) = (P_i - Q_j)^2 \,. \tag{6.1}$$

However, in fact any appropriate local distance measure may be used instead.

The procedure is called time warping because it warps the time axes, t and t', in such a way that corresponding sensor observations appear at the same location on a common time axis. In general, the time axes, t and t', are aligned in a non-linear way.

In the DTW algorithm we define a given alignment of the time series, $\boldsymbol{P}$ and $\boldsymbol{Q}$, by means of a "warping path" $\boldsymbol{W}$ (Fig. 6.2). This is a set of matrix

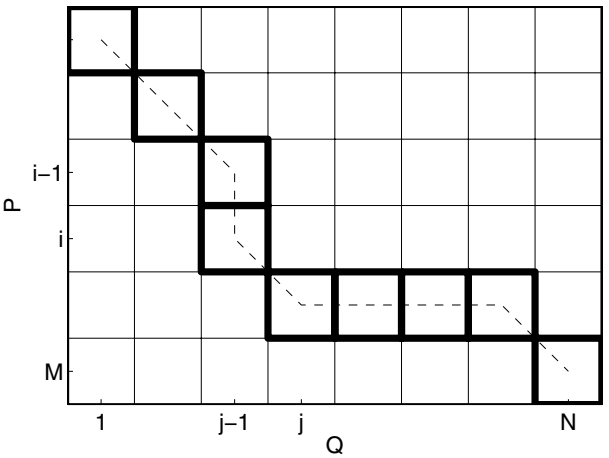

Fig. 6.2. Shows the Dynamic Time Warping (DTW) algorithm. We start at $w_1 = (1, 1)$ where $D(1, 1)$ is equal to $d(1, 1)$. We then proceed top-to-bottom and left-to-right updating the cumulative distance $D(i, j)$ using the cumulative distances of the neighbouring cells. The optimal matching cost, $\mathrm{DTW}(\boldsymbol{P}, \boldsymbol{Q})$, between the time series $\boldsymbol{P}$ and the second time series $\boldsymbol{Q}$ is $D(M, N)$. The optimal warping path W_{DTW} is represented by the dashed line and the thick-bordered cells which can be recovered by tracing back the sequence of optimal neighbours. $\boldsymbol{W}_{\mathrm{DTW}}$ provides information about the optimal correspondence (alignment) between the two time series $\boldsymbol{P}$ and $\boldsymbol{Q}$.

elements $w_k = (i_k, j_k), k \in \{1, 2, \ldots, K\}$, where w_k defines a mapping between the observations P_{i_k} and Q_{j_k}. Formally,

$$\boldsymbol{W} = (w_1, w_2, \ldots, w_K)^T \,, \tag{6.2}$$

where $\max(M, N) \leq K \leq M + N - 1$.

In general, we are only interested in warping paths (i.. e. alignments) which satisfy the following constraints:

Boundary Conditions. $w_1 = (1, 1)$ and $w_K = (M, N)$. This requires the warping path to start by matching the first observation of $\boldsymbol{P}$ with the first observation of $\boldsymbol{Q}$ and end by matching the last observation of $\boldsymbol{P}$ with the last observation of $\boldsymbol{Q}$.

Continuity. Given $w_k = (a, b)$, then $w_{k-1} = (a', b')$, where $a - a' \leq 1$ and $b - b' \leq 1$. This restricts the allowable steps in the warping path to adjacent matrix elements (including diagonally adjacent elements).

Monoticity. Given $w_k = (a, b)$ then $w_{k-1} = (a', b')$, where $a - a' \geq 0$ and $b - b' \geq 0$. This forces the warping path sequence to increase monotonically in time.

There are exponentially many warping paths that satisfy the above conditions, however, we are interested only in the path which minimizes the sum of the local distances $d(i_k, j_k) \equiv d(w_k)$. This is the optimal ("DTW") warping path and is defined as

$$\boldsymbol{W}_{\mathrm{DTW}} \equiv \boldsymbol{W}_{\mathrm{DTW}}(\boldsymbol{P}, \boldsymbol{Q}) = \min_{\boldsymbol{W}} \sum_{k=1}^{K} d(w_k) \,. \tag{6.3}$$

6.3 Dynamic Programming

A brute force method for finding the optimal warping path requires the evaluation of an exponential number of warping paths. Fortunately dynamic programming can be used to find the optimal warping path, $\boldsymbol{W}_{\mathrm{DTW}}$, and its cost, $\mathrm{DTW}(\boldsymbol{P}, \boldsymbol{Q})$, in $O(MN)$ time using the recursion:

$$D(i_k, j_k) = d(i_k, j_k) + \min(D(i_k, j_k - 1), D(i_k - 1, j_k), D(i_k - 1, j_k - 1)) \,, \tag{6.4}$$

where $D(i_k, j_k)$ is the cost of the optimal warping path from $(1, 1)$ to (i_k, j_k) and $D(1, 1) = d(1, 1)$.

The DTW algorithm starts by building a $M \times N$ table $D(m, n), m \in \{1, 2, \ldots, M\}, n \in \{1, 2, \ldots, N\}$, column by column. We start with $n = 1$ and compute $D(1, 1)$ using (6.4) and the boundary conditions

$$D(0, 0) = 0 \,, \tag{6.5}$$

$$D(0, j_k) = \infty = D(i_k, 0) \,. \tag{6.6}$$

We then proceed from $m = 2$ to $m = M$ and compute $D(2,1), D(3,1), \ldots, D(M,1)$. We then increase n by one and proceed from $m = 1$ to $m = M$ computing $D(1,2), D(2,2), \ldots, D(M,2)$. We continue the process until the last column $n = N$ and row $m = M$ are reached. By definition $D(M,N)$ is equal to $\mathrm{DTW}(\boldsymbol{P},\boldsymbol{Q})$, the cost of the optimal alignment of the sequences $\boldsymbol{P}$ and $\boldsymbol{Q}$. The optimal warping path, $\boldsymbol{W}_{\mathrm{DTW}}$, is obtained by tracing the recursion backwards from $D(M,N)$.

The following example illustrates the construction of the cumulative cost matrix $D(m,n)$ used in Fig. 6.2.

Example 6.3. Dynamic Programing. Given two time series $\boldsymbol{P} = (1.7, 4.1, 3.1, 2.2, 3.4, 1.6)^T$ and $\boldsymbol{Q} = (2.4, 3.5, 2.7, 3.2, 4.5, 4.7, 4.8, 5.9)^T$, we create a matrix of squared Euclidean distances $d(i,j) = (P_i - Q_j)^T$:

$$\begin{array}{cccccccc}
0.49 & 3.24 & 1.00 & 2.25 & 7.84 & 9.00 & 9.61 & 17.64 \\
2.89 & 0.36 & 1.96 & 0.81 & 0.16 & 0.36 & 0.49 & 3.24 \\
0.49 & 0.16 & 0.16 & 0.01 & 1.96 & 2.56 & 2.89 & 7.84 \\
0.04 & 1.69 & 0.25 & 1.00 & 5.29 & 6.25 & 6.76 & 13.69 \\
1.00 & 0.01 & 0.49 & 0.04 & 1.21 & 1.69 & 1.96 & 6.25 \\
0.64 & 3.61 & 1.21 & 2.56 & 8.41 & 9.61 & 10.24 & 18.49
\end{array}$$

The corresponding cumulative matrix D is

$$\begin{array}{cccccccc}
0.49 & 3.73 & 4.73 & 6.98 & 14.82 & 23.82 & 33.43 & 51.07 \\
3.39 & 0.85 & 2.81 & 3.62 & 3.73 & 4.14 & 4.63 & 7.87 \\
3.87 & 1.01 & 1.01 & 1.02 & 2.98 & 5.54 & 7.03 & 12.47 \\
3.91 & 2.70 & 1.26 & 2.01 & 6.31 & 9.23 & 12.30 & 20.72 \\
4.91 & 2.71 & 1.75 & 1.30 & 2.51 & 4.20 & 6.16 & 12.41 \\
5.50 & 6.32 & 2.96 & 3.86 & 9.71 & 12.12 & 14.44 & 24.65
\end{array}$$

The cost of the optimal alignment of $\boldsymbol{P}$ and $\boldsymbol{Q}$ is $\mathrm{DTW}(\boldsymbol{P},\boldsymbol{Q}) = 24.65$. By tracing back (see Fig. 6.2) we find the corresponding optimal warp path:

$$\boldsymbol{W}_{\mathrm{DTW}} = ((1,1),(2,2),(3,3),(4,3),(5,4),(5,5),(5,6),(5,7),(6,8))^T . \qquad \square$$

The original use of the DTW algorithm was in speech recognition. Although more advanced speech-recognition techniques are now available, the DTW is still widely used in small-scale embedded-speech recognition systems, such as those embedded in cellular telephones, which require minimal hardware.

Example 6.4. Speech-Recognition in an Embedded-Speech Recognition System [1]. A DTW speech-recognition system works as follows. Let $\omega^{(l)}$ and $\boldsymbol{Q}^{(l)}, l \in \{1, 2, \ldots, L\}$, denote, respectively, a set of L reference words and their corresponding time series. Given an unknown input time series $\boldsymbol{P}$ we use the DTW algorithm to optimally match $\boldsymbol{P}$ to each time series $\boldsymbol{Q}^{(l)}, l \in \{1, 2, \ldots, L\}$, in turn. If $\mathrm{DTW}(\boldsymbol{P},\boldsymbol{Q}^{(l)})$ denotes the cost of matching $\boldsymbol{P}$ with $\boldsymbol{Q}^{(l)}$, then we assign $\boldsymbol{P}$ to the word $\Omega = \omega_{\mathrm{OPT}}$, where

$$\omega_{\mathrm{OPT}} = \arg\min_{l} \mathrm{DTW}(\boldsymbol{P},\boldsymbol{Q}^{(l)}) . \qquad \square$$

The next example illustrates an unusual use of the DTW in matching images.

Example 6.5. Word-Spotting in Historical Manuscripts [260]. Easy access to handwritten historical manuscripts requires an index similar to that found in the back of this book. The current approach - manual transcription followed by index generation from the transcript - is extremely expensive and time-consuming.

For collections of handwritten manuscripts written by a single author the images of multiple instances of the same word are likely to look similar. For such collections, the *word spotting* algorithm provides an alternative approach to index generation. This works as follows. Each page in the collection of manuscripts is segmented into words. Then different instances of a given word are clustered together using an image matching algorithm. By annotating the clusters we create an index which links words to the locations where they occur.

A critical issue in word spotting is matching the word images. We may use the DTW algorithm to match the word images as follows. We divide each word image into columns of equal width Δ. In each column we extract four features, $(a_i, b_i, c_i, d_i)^T$, where i denotes the corresponding index of the column. In this manner we convert each word image into a multivariate series of feature values in which the horizontal image axis functions as a time axis. Once the conversion is complete we match any two word images P and Q by matching their multivariate feature values using the DTW algorithm [1]. □

6.3.1 Derivative Dynamic Time Warping

Although the DTW has been successfully used in many domains, it can produce pathological results. The reason is the algorithm may try to explain a difference between $\boldsymbol{P}$ and $\boldsymbol{Q}$ by unnaturally warping the two time axes t and t'. This can lead to unintuitive alignments. For example, a single point on one times series may map onto a large subsection of another time series (Fig. 6.3). One way of dealing with these singularities is to place an additional constraint on $\boldsymbol{W}$. Two constraints which are sometimes used for this purpose are:

Windowing. Locally we constrain $\boldsymbol{W}$ so it does not deviate too far from the diagonal path (as defined by the end points w_1 and w_K). The distance that the path is allowed to deviate is the window length R.

Slope Constraint. Locally we constrain $\boldsymbol{W}$ so it is not too steep and not too shallow. Mathematically we do this by replacing (6.4) by $D(i_k, j_k) = d(i_k, j_k) + \min(\alpha D(i_k, j_k - 1), \alpha D(i_k - 1, j_k), D(i_k - 1, j_k - 1))$, where α is positive real number.

[1] In [260] the authors define the local distance between the ith multivariate feature $(a_i^{(P)}, b_i^{(P)}, c_i^{(P)}, d_i^{(P)})^T$ in P and the jth multivariate feature $(a_j^{(Q)}, b_j^{(Q)}, c_j^{(Q)}, d_j^{(Q)})^T$ in Q as the squared Euclidean distance $d(i,j) = (a_i^{(P)} - a_j^{(Q)})^2 + (b_i^{(P)} - b_j^{(Q)})^2 + (c_i^{(P)} - c_j^{(Q)})^2 + (d_i^{(P)} - d_j^{(Q)})^2$.

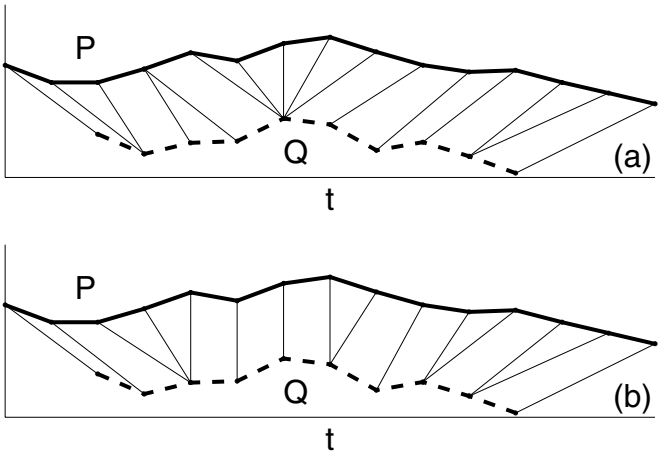

Fig. 6.3. (a) Shows the matching of similar time series $\boldsymbol{P}$ and $\boldsymbol{Q}$ using the traditional DTW algorithm with an Euclidean local cost function. **(b)** Shows the matching of the same curves using a DDTW algorithm.

An alternative approach is to use a different local cost function. In the derivative dynamic time warping (DDTW) algorithm [149] we use a local distance $d(i,j)$, which is defined as the square of the difference between the slopes of the curves $\boldsymbol{P}$ and $\boldsymbol{Q}$ at the times t_i and t'_j. Mathematically, the local distance is

$$d(i,j) = \left(\left.\frac{dP}{dt}\right|_i - \left.\frac{dQ}{dt'}\right|_j \right)^2 , \tag{6.7}$$

where $dP/dt|_{t_i}$ and $dQ/dt'|_{t'_j}$ are, respectively, the local slopes of $\boldsymbol{P}$ and $\boldsymbol{Q}$ at the times t_i and t'_j.

Example 6.6. Identifying a Face Profile Using the DDTW Algorithm [22]. Most current face profile recognition algorithms rely on the detection of fiducial points. Unfortunately features such as a concave nose, protrubing lips or a flat chin make the detection of such points difficult and unreliable. Also the number and position of the fiducial points vary when the expression changes even for the same person. As an alternative we may perform face profile recognition by matching an unknown face profile $\boldsymbol{P}$ to a data base of known face profiles $\boldsymbol{Q}^{(l)}, l \in \{1, 2, \ldots, L\}$, using the DDTW algorithm. □

6.3.2 Continuous Dynamic Time Warping

The continuous DTW [220, 221] is a continuous counter-part of the dynamic time warping algorithm in which an observation P_i is allowed to match apoint

which lies between two observations Q_j and Q_{j-1} and an observation Q_j is allowed to match a point which lies between two observations P_i and P_{i-1}. In order to generate the intermediate matching points we must assume a particular interpolation model. In [220, 221] the authors used a linear interpolation model. An alternative to the CDTW algorithm is to simply increase the number of points in $\boldsymbol{P}$ and $\boldsymbol{Q}$ by (linear) interpolation.

Example 6.7. Signature Verification Using Continuous Dynamic Time Warping [220, 221]. An area of increasing importance is the use of biometric techniques for personal identification. Signature verification is one such technique. In [220, 221] the authors demonstrated the use of the continuous DTW in online signature identification. Fig. 6.4 shows the matching of two curves using the traditional DTW algorithm and the continuous DTW algorithm. □

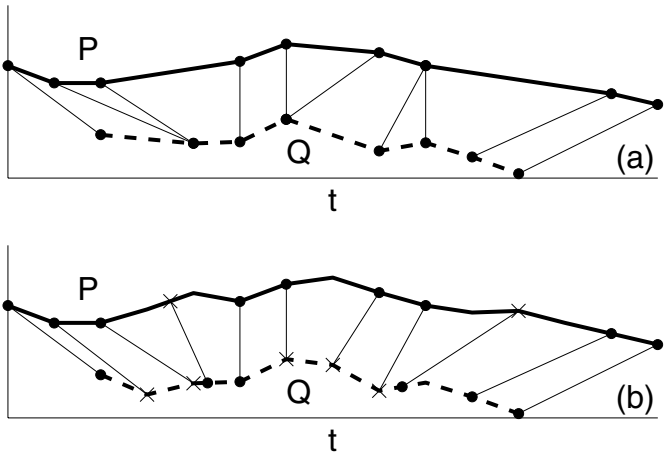

Fig. 6.4. (a) Shows the matching of two curves $\boldsymbol{P} = (P_1, P_2, \ldots, P_4)^T$ and $\boldsymbol{Q} = (Q_1, Q_2, \ldots, Q_5)^T$ using the traditional DTW algorithm. **(b)** Shows the matching of the same curves using a continuous DTW algorithm. Interpolated points are shown by crosses.

A variant of the DTW and CDTW algorithms occurs when we are only interested in warping one of the time series (say $\boldsymbol{Q}$) onto the second time series ($\boldsymbol{P}$). In this case, we use a warping path $\boldsymbol{V} = (v_1, v_2, \ldots, v_N)^T$, where $v_j = i$ specifies the mapping $Q_j \rightarrow P_i$. An important application of this model is video compression which is considered in the next section.

6.4 Video Compression

In video compression we have a original time series $\boldsymbol{P}$ which consists of a sequence of M images $P_i, i \in \{1, 2, \ldots, M\}$, taken at times t_i. This is compressed by forming a new time series $\boldsymbol{Q}$ which consists of N images $Q_j, j \in \{1, 2, \ldots, N\}$, at times t'_j. The images Q_j are created by selectively

1. "Dropping" the input images P_i.
2. "Repeating" the input images P_i.
3. Interpolating consecutive input images P_{i-1} and P_i.

A simple model which describes all of the above effects is (6.8):

$$Q_j = \alpha_j P_i + (1 - \alpha_j) P_{i-1} \ , \tag{6.8}$$

where $\alpha_j \in [0, 1]$ is a given interpolation weight. The index i changes with the compressed image Q_j and its aquisition time t'_j. To show this explicitly we rewrite (6.8) using a warping path $\boldsymbol{V} = (v_1, v_2, \ldots, v_N)^T$, where $v_j = i$ denotes a mapping between the compressed image Q_j and a pair of original images P_{v_j} and $P_{v_j - 1}$:

$$Q_j = \alpha_j P_{v_j} + (1 - \alpha_j) P_{v_j - 1} \ . \tag{6.9}$$

Example 6.8. Compressed Video Sequence [44]. Fig. 6.5 shows an original video sequence, $\boldsymbol{P} = (P_1, P_2, \ldots, P_8)^T$, which we compress by "frame drop", "frame repeat" and "frame interpolation". The corresponding compressed sequence is $\boldsymbol{Q} = (Q_1, Q_2, \ldots, Q_4)^T$, where Q_1 and Q_2 correspond, respectively, to the original images P_2 and P_4; Q_3 is a "repeat" of Q_2 (i. e. $Q_3 = Q_2$); Q_4 is

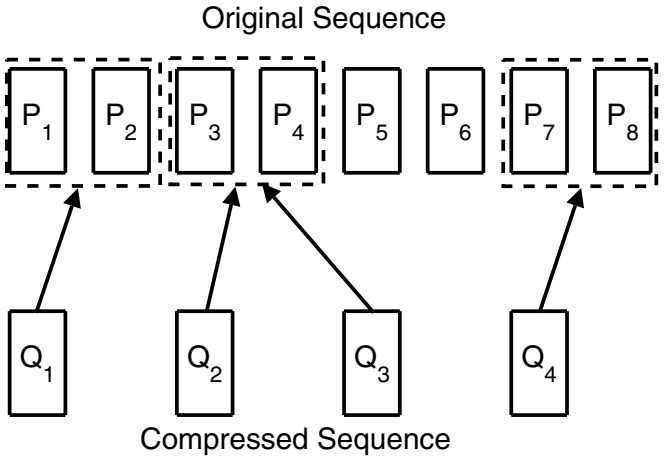

Fig. 6.5. Shows the use of the temporal model to represent a compressed video sequence.

Table 6.1. Mapping indices v_j and weights α_k for Ex. 6.8

Mapping Index v_j	Weight α_j
$v_1 = 2$	1.0
$v_2 = 4$	1.0
$v_3 = 4$	1.0
$v_4 = 8$	0.3

a weighted average of the images P_7 and P_8; and we ignore, or "drop", the images P_1, P_3, P_5 and P_6 [2]. The mappings v_j and weights α_j are listed in Table 6.1. □

In general we are only interested in warping paths, $\boldsymbol{V}$, which satisfy the following constraints:

Boundary Conditions. $v_1 = 2$ and $v_N = M$. This requires the warping path to start by matching the first compressed image, Q_1, with P_1 and P_2 and to end by matching the last compressed image, Q_N, with P_{M-1} and P_M.

Monoticity. Given $v_j = a$, then $v_{j-1} = a'$, where $a - a' \geq 0$. This forces the warping path to increase monotonically in time.

Note: Because of frame "dropping" we do *not* use a continuity constraint as is common in DTW.

There are exponentially many warping paths, $\boldsymbol{V}$, which satisfy the boundary and monoticity constraints, however, we are only interested in the path which minimizes the sum of the square errors between $\boldsymbol{Q}$ and $\boldsymbol{P}$. This jointly defines the optimal warping path, $\boldsymbol{V}_{\text{OPT}}$, and the optimal interpolation weights, $\boldsymbol{\alpha}_{\text{OPT}}$:

$$(\boldsymbol{V}_{\text{OPT}}, \boldsymbol{\alpha}_{\text{OPT}}) = \arg\min \sum_{j=1}^{N} |Q_j - (\alpha_j P_{v_j} + (1 - \alpha_j) P_{v_j - 1})|^2 \,. \tag{6.10}$$

Graphically, we may represent $\boldsymbol{V}_{\text{OPT}}$ as a monotonically non-decreasing path in a two-dimensional, (i, j), space. For example, the solution of Ex. 6.8 is the monotonically non-decreasing path shown in Fig. 6.6.

Apart from the boundary conditions and monoticity, we may include additional constraints which reflect *a priori* information concerning the application and what constitutes an optimal solution. For example, in video compression, frame repeat and frame drop are used infrequently and they are seldom repeated more than once. In [44] the authors included this information as an additional constraint C.

[2] Images P_1, P_3, P_5, P_6 are "drop" images since no compressed images Q_j are mapped to P_1, P_3, P_5, P_6.

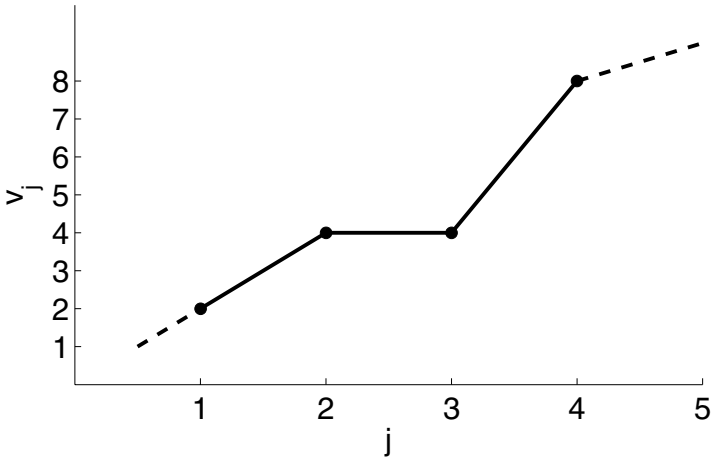

Fig. 6.6. Shows the monotonically non-decreasing path $\boldsymbol{V}_{\mathrm{OPT}}$ which represents the optimal matching indices v_j in Ex. 6.8.

Example 6.9. Frame Repeat Constraint C [44]. In video compression, frame repeat is used infrequently and is seldom repeated more than once. This means that the most likely reason for a large number of similar frames is that the scenes are relatively static and not that frame repeat has been used multiple times. An appropriate constraint, or cost function, for this case is shown in Fig. 6.7. This cost function enforces:

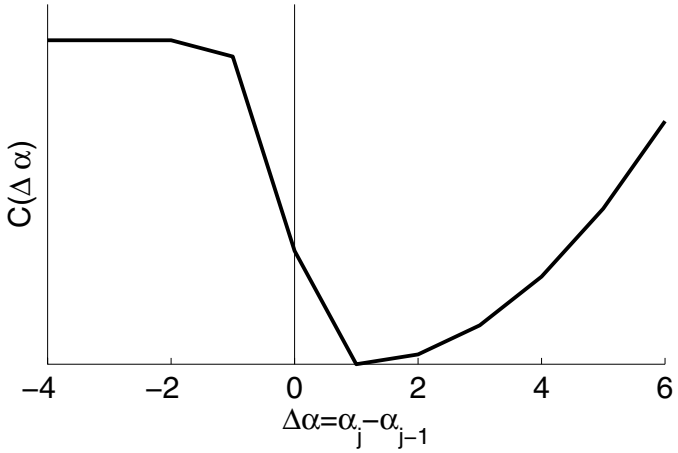

Fig. 6.7. Shows a contextual cost function $C(\Delta\alpha)$, where $\Delta\alpha = \alpha_j - \alpha_{j-1}$. This cost function enforces causal constraint and encourages smooth transition when frames have little motion or are static and penalizes the decision of frame repeat.

1. The causal constraint by setting $C(\Delta v)$ to the maximal distortion when $\Delta v < 0$, i. e. when $v_j < v_{j-1}$.
2. A smooth transition among static scenes by setting $C(\Delta v) = 0$ when $\Delta v = 1$, i. e. when $v_j = v_{j-1} + 1$.
3. Penalizes frame repeat by setting $C(\Delta\alpha)$ equal to a positive cost when $\Delta v = 0$, i. e. when $v_j = v_{j-1}$.

As in the case of $\boldsymbol{W}_{\mathrm{OPT}}$, we may find $\boldsymbol{V}_{\mathrm{OPT}}$ using *dynamic programming*. By definition, the solution is a monotonically non-decreasing path $\boldsymbol{V}_{\mathrm{OPT}}$ that has the minimal sum of square errors. Let $D(n)$ denote the accumulated square error over an initial segment of $\boldsymbol{V}_{\mathrm{OPT}}$ from $j = 1$ to $j = n$, then

$$\begin{aligned} D(n) =& \min_{\substack{\alpha_1,\alpha_2,\ldots,\alpha_n \\ v_1\le\ldots\le v_n}} \sum_{j=1}^{n} |Q_j - (\alpha_j P_{v_j} + (1-\alpha_j)P_{v_j-1})|^2 \,, \\ =& \min_{v_{n-1}\le v_n} \left\{ \min_{\substack{\alpha_1,\alpha_1,\ldots,\alpha_{n-1}, \\ v_1\le\ldots\le v_{n-1}}} \sum_{j=1}^{n-1} |Q_j - (\alpha_j P_{v_j} + (1-\alpha_j)P_{v_j-1})|^2 \right. \\ & \left. + \min_{\alpha_n} |Q_n - (\alpha_n P_{v_n} + (1-\alpha_n)P_{v_n-1})|^2 \right\} , \\ =& D(n-1) + \min_{v_{n-1}\le V_n} \left\{ \min_{\alpha_n} |Q_n - (\alpha_n P_{v_n} + (1-\alpha_n)P_{v_n-1})|^2 \right\} . \end{aligned} \tag{6.11}$$

Inspection of (6.11) shows that $\boldsymbol{V}_{\mathrm{OPT}}$ and $\boldsymbol{\alpha}_{\mathrm{OPT}}$ may be efficiently found using dynamic programming as follows:

1. Find the optimum weight, α_j, for each compressed image $Q_j, j \in \{1, 2, \ldots, N\}$:
$$\alpha_j = \arg\min_{\alpha_j} |Q_j - (\alpha_j P_{v_j} + (1-\alpha_j)P_{v_j-1})|^2 \,. \tag{6.12}$$
2. Recursively compute $D(j), j \in \{1, 2, \ldots, N\}$, using the optimum weight α_j, in (6.11).
3. After completing the calculation of $D(N)$, trace back to find the optimum warping path, $\boldsymbol{V}_{\mathrm{OPT}}$, and the optimum weight vector, $\boldsymbol{\alpha}_{\mathrm{OPT}}$.

6.5 Software

DTW. A matlab m-file for performing dynamic time warping.

6.6 Further Reading

Temporal alignment has been intensely investigated over the years. In recent years it has been increasingly used in various applications including: medical

imaging [155, 166], speech recognition [1, 255], image sequence compression and watermark tracking in video and digital cinema [44, 45, 185]. The DTW has been used in these applications and many others (e. g. word spotting, identifying facial profiles and signature verification) which do not explicitly involve temporal alignment.

7

Sensor Value Normalization

7.1 Introduction

The subject of this chapter is *sensor value normalization* which we define as the conversion of all sensor output values to a common scale. Sensor value normalization is the third function listed in Sect. 4.1 which is required for the formation of a common representational format and is the subject of the present chapter.

The range of different possible scales is enormous and we shall therefore find it convenient to divide them into several categories according to the following criteria:

Nature of Scale. This refers to the basic mathematical properties of the scales. For example, most scales are bounded and linearly ordered. This means that all the elements in the scale are comparable. The choice of scale has consequences on the combination operations which are allowed. For instance some numerical operators can only accommodate numerical values, others like "min" or "max" can also deal with ordinal values.

Homogeneity. This refers to whether, or not, the sensors are measuring same thing. For example a logical sensor may measure the degree to which a test object resembles a reference object. The sensor may output a distance, or a *dissimilarity* coefficient, or it may output a proximity measure, or a *similarity* coefficient.

Statistical Distribution. This refers to the empirical statistical distribution followed by the sensor values when applied to a given population.

Semantics. This refers to semantics, or meaning, of a scale e. g. probabilistic, possibilistic, utility or degree-of-similarity. It is often thought that problems of incommensurability will not occur if all the scales have the same semantics. This is not, however, always true. In particular, problems of incommensurability may occur when we use "utility" or "degree-of-similarity" scales. The reason is that utility and degree-of-similarity scales do not have a clear and well-defined meaning.

Example 7.1. Similarity Measures for Virtual Screening. [121, 331] "Virtual screening" involves the use of a computational scoring scheme to rank molecules in decreasing order of probability of activity. One way of carrying out virtual screening is to use similarity searching. This involves taking a molecule (normally called the target compound) with the required activity and then searching a database to find molecules that are structurally most similar to it.

A molecular similarity measure has two principal components: (1) the structural representation used to characterize the molecules and (2) the similarity coefficient used to compute the degree of resemblance between pairs of such representations. Most current systems for similarity based virtual screening use molecular "fingerprints" which are binary vectors encoding the presence, or absence, of substructural fragments within the molecule. Table 7.1 lists twenty-two similarity coefficients for the comparison of pairs of molecular fingerprints.

Table 7.1. Similarity Coefficients for a Virtual Screening Programme

Name	Type	Formula
Jaccard/Tanimoto	D	$a/(a+b+c)$.
Dice	D	$2a/(2a+b+c)$.
Russell/Rao	D	a/n.
Sokal/Sneath$_1$	D	$a/(a+2b+2c)$.
Sokal/Sneath$_2$	D	$2(a+d)/(a+d+n)$.
Sokal/Sneath$_3$	D	$(a+d)/(b+c)$.
Kulczynski$_1$	D	$a/(b+c)$.
Kulczynski$_2$	D	$a(2a+b+c)/(2(a+b)(a+c))$.
Simple Match	D	$(a+d)/n$.
Hamann	D	$(a+d-b-c)/n$.
Rogers/Tanimoto	D	$(a+d)/(b+c+n)$.
Baroni-Urbani/Buser	D	$(\sqrt{ad}+a)/(\sqrt{ad}+a+b+c)$.
Ochiai/Cosine	D	$a/\sqrt{(a+b)(a+c)}$.
Forbes	D	$na/((a+b)(a+c))$.
Fossum	D	$n(a-0.5)^2/((a+b)(a+c))$.
Simpson	D	$a/\min(a+b, a+c)$.
Mean Manhattan	P	$(b+c)/n$.
Pearson	C	$(ad-bc)/\sqrt{(a+b)(a+c)(b+d)(c+d)}$.
Yule	C	$(ad-bc)/(ad+bc)$.
McConnaughey	C	$(a^2-bc)/((a+b)(a+c))$.
Stiles	C	$\log_{10} n + 2\log_{10}(\lvert ad-bc\rvert - n/2) - \log_{10}(a+b)$ $(a+c)(b+d)(c+d)$.
Dennis	C	$(ad-bc)/\sqrt{n(a+b)(a+c)}$.

$n = a+b-c+d$ is the total number of attributes of an object, where a is the number of attributes which are present in molecule A, b is the number of attributes which are present in molecule B, c is the number of attributes which are present in both molecules A and B, d is the number of attributes which are not present in A and are not present in B.

Although all twenty-two similarity coefficients are calculated using the same binary vectors their meanings are not all the same. The coefficients labeled "*D*" are *distance coefficients*, in which the greater the degree of similarity between the two molecules the smaller the value of the coefficient, the coefficients labeled "*C*" are *correlation coefficients*, which measure the degree of statistical correlation between the molecules and *vice versa*, the coefficient labeled "*P*" is a *proximity coefficient* normalized to lie within the range of zero (no similarity at all) and unity (identical sets of descriptors). □

7.1.1 Sensor Value Normalization

The process of converting the values of different pieces of information to a common scale is known as *sensor value normalization*. Suppose we use a set of logical sensors, or sources-of-information, $S_i, i \in \{1, 2, \ldots, N\}$, to make N measurements x_i on an object O. Then, we transform the x_i into a common representational format by changing the scale and location parameters of the x_i. Mathematically we write the transformation of an input measurement x to a normalized value y as

$$y = f(x|\alpha, \beta, \gamma, \ldots, \delta) \ , \tag{7.1}$$

where f is a parametric *transformation function* with scale and location parameters α and β and possibly other parameters γ, δ etc.

In most cases the parameters $\alpha, \beta, \ldots, \delta$ are learnt from measurements made on a set of *training samples* $D = x_i, i \in \{1, 2, \ldots, N\}$. In this case the procedure is referred to as *fixed sensor value normalization*. In a few cases, the transformation is estimated from the current measurement values. This is known as *adaptive sensor value normalization* and its main advantage is its ability to adapt to variations in the input data.

For a good normalization scheme, the parameters $\alpha, \beta, \ldots, \delta$ must be robust and efficient, where *robustness* refers to insensitivity in the presence of outliers [1] and *efficiency* refers to the proximity of the obtained estimate to the optimal estimate when the distribution of the data is known.

7.2 Binarization

Binarization is probably the simplest normalization technique. It is a widely used robust technique and is often employed when many different sensor measurements $x_i, i \in \{1, 2, \ldots, N\}$, are made on the same object. Given an object O we threshold each x_i using a threshold t_i. The corresponding normalized values are y_i, where

$$y_i = \begin{cases} 1 \text{ if } x_i \geq t_i \ , \\ 0 \text{ otherwise} \ . \end{cases} \tag{7.2}$$

[1] See Chapt. 10 for a full discussion concerning robust statistics and outliers.

The thresholds $t_i, i \in \{1, 2, \ldots, N\}$, may be learnt (either on the current set of measurements or on a training set D) or may be established on physical grounds. The following example illustrates the use of a "soft" binarization procedure in multi-sensor data fusion.

Example 7.2. "Soft" Binarization. In many multi-sensor data fusion applications it is not possible to establish the correct threshold t_i exactly. To reduce the impact of choosing an incorrect threshold, we may replace (7.2) with a trimmed min-max transfer function (see Table 7.4):

$$y_i = \begin{cases} 1 & \text{if } x_i > t_i + \delta \ , \\ 0 & \text{if } x_i < t_i - \delta \ , \\ \frac{1}{2}(x_i - t_i + \delta)/\delta & \text{otherwise} \ . \end{cases}$$

or with a sigmoid-like transfer function (see Fig. 7.1). □

Example 7.3. Psychometric Transfer Functions [101]. Psychophysics explores the connection between physical stimuli and subjective human responses. The psychometric transfer function, $f(x|\alpha, \beta)$, relates the intensity of the stimulus, x, to an observers's response, y. It is a parametric function with two parameters: a scale parameter α and a location parameter β. The psychometric function is commonly chosen to have a sigmoid-like shape . Five sigmoid-like psychometric transfer functions are listed, together with their slopes $df(x|\alpha, \beta)/dx$ in Table 7.2 □

Binarization is often used as a preliminary step where it functions as *detection* step. For example, in calculating the similarity coefficients in Ex. 7.1 we assumed that all the features had undergone a preliminary binarization, or detection, process.

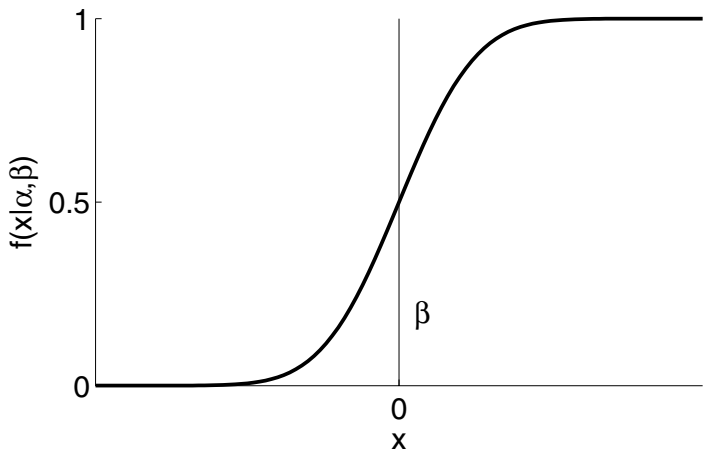

Fig. 7.1. Shows a Logistic sigmoid transfer function. See Table 7.2 for definition of this and other similar sigmoid-like transfer functions.

Table 7.2. Psychometric Transfer Functions $f(x|\alpha,\beta)$ and Slopes $df(x|\alpha,\beta)/dx$

Name	$f(x\|\alpha,\beta)$	$df(x\|\alpha,\beta)/dx.$
Logistic	$1/(1+\exp(\alpha(\beta-x)))$	$\alpha f(x\|\alpha,\beta)\times(1-f(x\|\alpha,\beta))$.
Probit	$\frac{\alpha}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp-\frac{1}{2}(\alpha(t-\beta))^2dt$	$\frac{\alpha}{\sqrt{2\pi}}\exp-(\alpha(x-\beta))^2/2)$.
Gumbel	$1-\exp(-\exp(\alpha\ln(x)-\alpha\ln(\beta)))$	$\frac{\alpha}{x}(1-f(x\|\alpha,\beta))\exp(\alpha\ln(x)-\alpha\ln(\beta))$.
Weibull	$1-\exp(-(\frac{x}{\beta})^{\alpha})$	$\alpha x^{\alpha-1}(1-f(x\|\alpha,\beta))/\beta^{\alpha}$.
Quick	$1-2^{-(x/\beta)^{\alpha}}$	$\ln(2)\alpha x^{\alpha-1}(1-f(x\|\alpha,\beta))/\beta^{\alpha}$.

Although very effective, problems may arise if, for a given object O, we do not have a complete set of N measurement values. The reason is that when we perform binarization we do not differentiate between (1) $y_i = 0$ because $x_i \leq t_i$ and (2) $y_i = 0$ because, for one reason or another, no measurement value x_i is available. For example, the sensor S, which is responsible for making the measurement x_i, may have malfunctioned when the object O was being processed. To handle such situations we use a *censored* binary scale: which is defined as follows:

$$y_i = \begin{cases} 1 \text{ if } x_i \geq t_i \ , \\ 0 \text{ if } x_i < t_i \ , \\ * \text{ if } x_i \text{ is missing} \ , \end{cases} \tag{7.3}$$

where $*$ is a special symbol which is handled differently from 0 and 1 by the data fusion algorithm. Censored binary scales are widely used when the information concerning the x_i are *subjective* in nature. The following example illustrates the use of a censored binary scale when the x_i are subjective replies to questions contained in a questionnaire.

Example 7.4. Automatic Disease Classification: Syndromic Surveillance [117, 134]. The concept of *syndromic surveillance* was developed to ensure early detection of an epidemic in a given population. Syndrome surveillance works by monitoring the temporal and spatial trends of a syndrome, or a group of related symptoms or diseases, in the local population. For example [134] used a censored questionnaire (similar to that shown in Table 7.3) to monitor cases of acute gastrointestinal syndrome which were admitted to the local hospital emergency departments. □

Binarization is widely used in image analysis. The following example illustrates the use of censored binarization to delinate the outline of one or more objects O_i in an input image .

Example 7.5. Image Thresholding Using Kriging [232]. We consider the problem of segmenting a 2D image consisting of two univariate populations: background (class $C = c_1$) and foreground (class $C = c_2$). We suppose we have sufficient *a priori* knowledge to perform an initial (censored) labelling of the input image:

$$y(\boldsymbol{u}) = \begin{cases} 1 \text{ if } g(\boldsymbol{u}) > T_2 \ , \\ 0 \text{ if } g(\boldsymbol{u}) < T_1 \ , \\ * \text{ otherwise} \ , \end{cases}$$

Table 7.3. Questionnaire Used for Detecting Acute Gastrointestinal Syndrome

Question	Medical Diagnosis	Possible Answer
Diarrhea present?	Yes, No	$[1, 0]$.
Diarrhea duration?	Acute, Chronic, Unknown	$[1, 0, *]$.
Diarrhea etiology?	Infectious, Non-infectious, Unknown	$[1, 0, *]$.
Stool culture taken?	Yes, No	$[1, 0]$.
Vomiting present?	Yes, No	$[1, 0]$.
Vomiting duration?	Acute, Chronic, Unknown	$[1, 0, *]$.
Vomiting etiology?	Infectious, Non-infectious, Unknown	$[1, 0, *]$.
Acute GI disorder?	Yes, No	$[1, 0]$.

where $g(\boldsymbol{u})$ denotes the gray-level of the pixel $\boldsymbol{u}$ and T_1 and T_2 are two thresholds (Fig. 7.2).

Our task is to label the unassigned pixels (i. e. the pixels with $T_1 \leq g(\boldsymbol{u}) \leq T_2$). Let $\boldsymbol{u}_0$ denote a given unassigned pixel and $g_i = g(\boldsymbol{u}_i), i \in \{1, 2, \ldots, N\}$, denote the gray-levels of the N nearest assigned pixels. Given the y_i we generate two binary lists:

$$z'_i = z'(\boldsymbol{u}_i) = \begin{cases} 1 \text{ if } g(\boldsymbol{u}_i) < T_1 \ , \\ 0 \text{ otherwise} \ , \end{cases}$$

and

$$z''_i = z''(\boldsymbol{u}_i) = \begin{cases} 1 \text{ if } g(\boldsymbol{u}_i) < T_2 \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

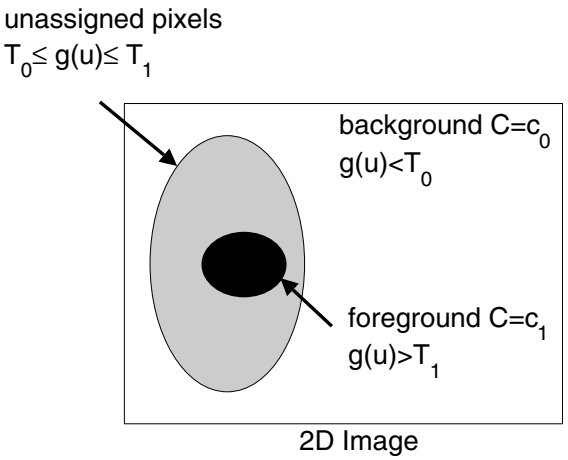

Fig. 7.2. Shows a 2D image consisting of background pixels (class $C = c_1$) whose gray-levels are less than a given threshold T_1 and foreground pixels (class $C = c_2$) whose gray-levels are greater than a given threshold T_2. Unassigned pixels are defined as pixels whose gray-levels fall between the two thresholds: $T_1 \leq g(\boldsymbol{u}) \leq T_2$.

We calculate interpolated values $z'(\boldsymbol{u}_0)$ and $z''(\boldsymbol{u}_0)$ by performing ordinary Kriging (see Sect. 4.3) on the z'_i and z''_i lists:

$$z'(\boldsymbol{u}_0) = \sum_{i=1}^{N} \lambda_i z'_i \ ,$$

$$z''(\boldsymbol{u}_0) = \sum_{i=1}^{N} \lambda_i z''_i \ .$$

We interpret the interpolated values $z'(\boldsymbol{u}_0)$ and $(1 - z''(\boldsymbol{u}_0))$ as the probabilities that the gray-level $g(\boldsymbol{u}_0)$ belongs, respectively, to the background, $y = 0$, and to the foreground, $y = 1$. For the final classification of $g(\boldsymbol{u}_0)$ we choose the class with the highest probability:

$$y(\boldsymbol{u}_0) = \begin{cases} 1 & \text{if } (1 - z''(\boldsymbol{u}_0)) \geq z'(\boldsymbol{u}_0) \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

□

7.3 Parametric Normalization Functions

The most common normalization technique is to use a *parametric* normalization function $y = f(x|\alpha, \beta, \ldots, \delta)$, whose parameters $\alpha, \beta, \ldots, \delta$, are learnt from a training set D of labelled bjects $O_i, i \in \{1, 2, \ldots, N\}$. One example of a parametric normalization function is the family of pyschometric transfer functions listed in Table 7.2. Additional parametric transfer functions are listed in Table 7.4. *Note*: Only the trimmed min-max, robust z-transform and robust tanh normalization functions are robust against outliers (see Chapt. 10). The remaining functions are not robust against outliers and should not be used if outliers are likely to be present.

Table 7.4. Parametric Normalization Functions

Function	Formula
Min-Max	$y = (x - a)/(b - a)$, where $a = \min_i(x_i)$, $b = \max_i(x_i)$. In trimmed min-max we replace, respectively, a and b by the lth smallest and largest x_i values. The resulting normalized value is $y = \min(\max(0, (a - x)/(b - a)), 1)$. *Note*: Only min-max values y_i retains the same distribution as the input x_i values.
Z-Transform	$y = (x - \mu)/\sigma$, where $\mu = \sum_k x_k/K$, $\sigma^2 = \sum_k (x_k - \mu)^2/(K - 1)$. In robust Z-transform we replace μ by median$\{x_i\}$ and Q by the interquartile distance $(x_{(3N/4)} - x_{(N/4)})/2$, where $x_{(l)}$ is the lth largest value in $\{x_i\}$.
Robust Tanh	$y = \frac{1}{2} \tanh(\alpha(x - \mu_H)/\sigma_H + 1)$, where α determines the spread of the normalized scores and μ_H and σ_H are robust Hampel mean and standard deviation estimates of the $x_i, i \in \{1, 2, \ldots, N\}$ [138, 226].

7.4 Fuzzy Normalization Functions

[2] Fuzzy logic provides us with a very powerful normalization technique.In this technique an input value x is converted into a *vector* of M components:

$$\boldsymbol{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(M)}) \,, \tag{7.4}$$

where $y^{(m)} = \mu_m(x)$ represents the degree to which x "belongs" to the mth membership function $\mu_m(x)$. The membership functions $\mu_m(x), m \in \{1, 2, \ldots, M\}$, are continuous functions of x for which $0 \leq \mu_m(x) \leq 1$. *Note*: In fuzzy logic there is no requirement that the components $y^{(m)}$ should sum to one. In fact, $\sum_m y^{(m)}$ can take any value between 0 and M. The following example illustrates the normalization of a given parameter using fuzzy logic.

Example 7.6. Normalization of the Inoptropic Reserve Using Fuzzy Logic [16]. In Ex. 5.7, we considered the fusion of the Inoptropic Reserve (IR) and the 18-fluorodeoxyglucose (FDG). Before fusion both variables were normalized by using fuzzy membership functions each of which was interpreted as a *linguistic variable*. Fig. 7.3 shows the five membership functions used to normalize IR together with their linguistic interpretation. □

7.5 Ranking

Ranking is a robust normalization technique which has enjoyed a growing popularity in recent years. Ranking is an example of *adaptive score normalization*,

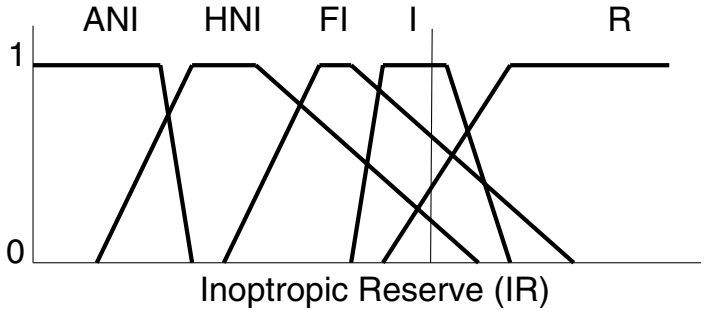

Fig. 7.3. Shows the five membership functions $\mu_i(x)$ used to normalize the Inoptropic Reserve (IR). Each membership function is given a name, ANI, HNI, FI, I, R, corresponding to its linguistic, or medical, interpretation, "Akinetic and No Improvement", "Hypokinetic with No Improvement", "Function Improvement", "Ischemic", "Remote" The thin vertical line shows a given value of IR $= x$ which is converted to the vector $\boldsymbol{y} = (\mu_{\rm ANI}(x), \mu_{\rm HNI}(x), \mu_{\rm FI}(x), \mu_{\rm I}(x), \mu_{\rm R}(x))$, where $\mu_{\rm ANI}(x) = 0.0$, $\mu_{\rm HNI}(x) = 0.15$, $\mu_{\rm FI}(x) = 0.55$, $\mu_{\rm I}(x) = 1.0$ and $\mu_{\rm R}(x) = 0.35$.

[2] This section contains advanced material. It assumes the reader has some familiarity with fuzzy logic. The section is not, however, essential for what follows and it may be left until the reader has aquired some basic knowledge of fuzzy logic.

i. e. the normalization is based on the current input measurements and it does not require the use of a training set D. Its main advantages are: it is simple and fast to implement, linear, robust against outliers and can be applied to all kinds of input information without the need to make additional assumptions regarding their distributions etc [3].

Mathematically, rank normalization is defined as follows. Suppose the input data consists of N objects $O_i, i \in \{1, 2, \ldots, N\}$. On each object O_i we make a measurement x_i. Then the corresponding rank is r_i, where

$$r_i = N - j \text{ if } x_i \text{ is the } j\text{th largest input measurement value} . \tag{7.5}$$

The following example illustrates the use of ranking in an application involving the fusion of different similarity measures.

Example 7.7. Fusing Similarity Coefficients in Virtual Screening Programme [103]. In similarity-based virtual screening (Ex. 7.1) we match a bio-active target structure A against a database of molecules $B_i, i \in \{1, 2, \ldots, N\}$, using any one of several different similarity operators $S_k, k \in \{1, 2, \ldots, K\}$. *A priori* we cannot know which operator will be best for a given target structure or for a given type of activity. Let $x_k(A, B_i)$ denote the corresponding similarity value as measured by the kth operator S_k. Then the authors in [103] showed that we may produce a more effective and robust activity measure by fusing together all K values $x_k(A, B_i), k \in \{1, 2, \ldots, K\}$. Since, in general, the $x_k(A, B_i)$ are incommensurate we must normalize them before fusion. This we do by ranking. Let $r_k(A, B_i)$ denote the rank of $x_k(A, B_i)$, then the fused similarity value, $\widetilde{r}(A, B_i)$, is defined as the sum of the $r_k(A, B_i)$:

$$\widetilde{r}(A, B_i) = \sum_{k=1}^{K} r_k(A, B_i) . \qquad \square$$

The following example illustrates the calculation of $\widetilde{r}(A, B_i)$.

Example 7.8. Fusing Similarity Coefficients: A Numerical Example. Let $\boldsymbol{x}_k(A, \boldsymbol{B}) = (x_k(A, B_1), x_k(A, B_2), \ldots, x_k(A, B_N))^T$ denote a vector of similarity values, measured by a similarity operator $S_k, k \in \{1, 2, \ldots, K\}$, between a given target structure A against a database of molecules $B_i, i \in \{1, 2, \ldots, N\}$. The experimentally measured similarity values are:

$$\boldsymbol{x}_1(A, \boldsymbol{B}) = (5.3, 6.1, 7.2, 1.2, 2.0, 3.3, 4.9, 8.8, 9.2, 10.3)^T ,$$
$$\boldsymbol{x}_2(A, \boldsymbol{B}) = (2.2, 5.3, 9.1, 5.3, 5.3, 2.2, 2.2, 2.2, 2.2, 2.2)^T .$$

The corresponding rank vectors $\boldsymbol{r}_k(A, \boldsymbol{B}) = (r_k(A, B_1), r_k(A, B_2), \ldots, r_k(A, B_N)^T, k \in \{1, 2\}$ [4] are:

[3] Robust techniques are discussed at length in Chapt. 10.

[4] The raw measurement values for the sensor S_1 are all distinct. As a consequence the corresponding ranks are the integers from 1 to 10. On the other hand, the raw measurement values for the sensor S_2 are not distinct. In this case, measurements with the same value receive the same *average* rank which may now be non-integer.

$$\boldsymbol{r}_1(A, \boldsymbol{B}) = (5, 6, 7, 1, 2, 3, 4, 8, 9, 10)^T ,$$
$$\boldsymbol{r}_2(A, \boldsymbol{B}) = (3.5, 8, 10, 8, 8, 3.5, 3.5, 3.5, 3.5, 3.5)^T .$$

The fused similarity rank vector is $\widetilde{\boldsymbol{r}}(A, \boldsymbol{B}) = (\widetilde{r}(A, B_1), \widetilde{r}(A, B_2), \ldots, \widetilde{r}(A, B_N))^T$, where $\widetilde{r}(A, B_i) = \frac{1}{2}(r_1(A, B_i) + r_2(A, B_i))$ is:

$$\widetilde{\boldsymbol{r}}(A, \boldsymbol{B}) = (8.5, 14, 17, 9, 10, 6.5, 7.5, 11.5, 12.5, 13.5)^T .$$

The entire process may be implemented using a parallel network of fusion, or similarity, cells as shown in Fig. 7.4. □

Example 7.9. Remote Sensing [227]. Remotely sensed data are increasingly used for mapping and monitoring the physical environment. One of the advantages of monitoring with remotely sensed data is that temporal sequences can accurately indicate environmental changes, assuming that the input data is radiometrically consistent for all scenes. Factors contributing to the potential inconsistency in measured radiance include changes in surface condition, illumination geometry, sensor calibration, observation geometry and atmospheric condition. By using a radiometric normalization technique, we may however, correct for data inconsistencies resulting from many different effects. Image normalization is carried out in one step by converting image values to ordinal ranks. Ordinal ranking allows us to assign each pixel a new value based on its reflectance value, relative to all other pixels. When image pairs are converted to ordinal ranks the global characteristics of the distributions of pixel values are matched.

Pixel ranking does not require atmospheric details, sensor information, or selection of subjective pseudo-invariant features, and therefore allows images

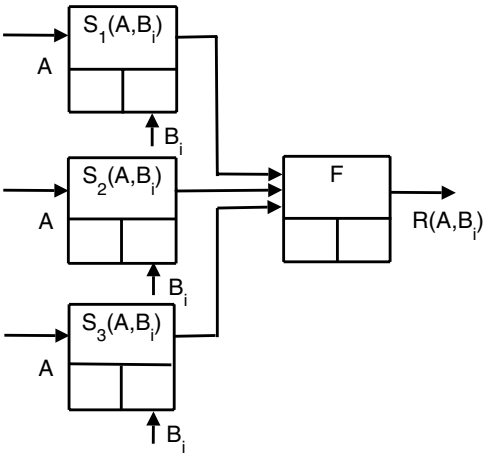

Fig. 7.4. Shows a parallel arrangement of three fusion cells $S_k(A, B_i), k \in \{1, 2, 3\}$, which perform ranking of the samples B_i according to different criteria. The output from F are the fused similarity ranks, $\widetilde{r}(A, B_i)$.

to be simply and efficiently normalized and processed for changes with minimal *a priori* knowledge. In general, for small pictures, pixel ranking is an effective image normalization technique. It is, however, less effective on very large digital images because in this case we obtain many tied ranks. □

7.6 Conversion to Probabilities

Until now we have only considered normalized techniques in which a sensor measurement x, and its corresponding normalized value y, have the same physical properties. We now consider a new normalization technique in which this is no longer true. In particular we give y an explicit physical meaning as an *a posteriori* probability [5].

We start with the simplest case in which all objects O belong to either a class $C = c_1$ or to a class $C = c_2$. Suppose x is the measured value of O. Then our aim is to find two functions $y = f_1(x)$ and $y = f_2(x)$ such that $f_k(x)$ closely approximates the *a posteriori* probability density function $p(C = c_k|x, I)$:

$$f_1(x) \approx p(C = c_1|x, I) \ , \tag{7.6}$$

$$f_2(x) \approx p(C = c_2|x, I) \ . \tag{7.7}$$

Since, by definition,

$$p(C = c_1|x, I) = 1 - p(C = c_2|x, I) \ , \tag{7.8}$$

we need only calculate one function $f_2(x)$. We now consider several different approaches to finding $f_2(x)$. For notational convenience we shall, from now on, write $f(x)$ in place of $f_2(x)$. *Note.* The formulas which follow all assume balanced training sets, i.e. the number of objects in each training set are equal.

7.6.1 Platt Calibration

In Platt calibration we use a parametric function, $f_2(x|\alpha, \beta, \ldots, \delta)$, with a given shape. The following example shows how we learn the parameters $\alpha, \beta, \ldots, \delta$, using a database D of labeled objects $O_i, i \in \{1, 2, \ldots, N\}$.

Example 7.10. Converting SVM Scores into Probability Estimates [244, 348]. Given a measured value x derived from a support vector machine (SVM) we convert it into an estimate of $p(C = c_2|x, I)$ by assuming a sigmoid function for $f(x|\alpha, \beta)$:

[5] Theoretically, this is the preferred normalization technique since, if successful, we have at our disposal all of the methods of Bayesian analysis (Chapt. 8). However, in practice, conversion to *a posteriori* probabilities is often found to perform badly and for this reason the methods considered previously are often used instead.

$$f(x|\alpha,\beta) = \frac{1}{1+\exp(\alpha(\beta-x))} ,$$

where α and β are two parameters which we adjust for maximum likelihood as follows. Suppose the object O_i has a value x_i and a class label z_i, where

$$z_i = \begin{cases} 0 \text{ if } O_i \text{ belongs to class } c_1 , \\ 1 \text{ if } O_i \text{ belongs to class } c_2 . \end{cases}$$

Then we find the parameters α and β by minimizing the negative log likelihood of the data set D:

$$(\alpha,\beta) = \arg\min \quad -\sum_{i=1}^{N} \big(z_i \ln f(x_i|\alpha,\beta) + (1-z_i)\ln(1-f(x_i|\alpha,\beta)) \big) .$$

In performing the above optimization procedure we must be careful not to *overfit*. This may happen if the number of objects O_i which belong to $C = c_1$ or to $C = c_2$ is too low. In this case, we mitigate the effects of overfitting by using modified labels, z_i', in the above minimization procedure, where

$$z_i' = \begin{cases} 1/(N+2-\sum z_i) & \text{if } O_i \text{ belongs to class } C = c_1 , \\ (\sum z_i + 1)/(\sum z_i + 2) & \text{if } O_i \text{ belongs to class } C = c_2 . \end{cases}$$ □

7.6.2 Binning

In *binning* we do not know the shape of the function $f(x)$ and we cannot, therefore, use Platt calibration. Instead we use a non-parametric method such as frequency histogram or *binning*: Suppose the objects $O_i, i \in \{1,2,\ldots,N\}$, in D are arranged in order of increasing value, i. e. $x_1 \le x_2 \le \ldots \le x_N$. Then the x_i are divided into M non-overlapping subsets of called *bins*, $B_m, i \in \{1,2,\ldots,M\}$ (see Sect. 12.3.3). For each bin we compute lower and upper boundary values, x_m' and x_m''. Then, we define $f(x) = p(C = c_2|x, I)$ as

$$f(x) = \frac{n_m^{(2)}}{n_m^{(1)} + n_m^{(2)}} \qquad x_m' \le x \le x_m'' , \tag{7.9}$$

where $n_m^{(k)}$ is the number of $C = c_k$ objects $O_i, i \in \{1,2,\ldots,N\}$, which fall in the mth bin B_m [6].

7.6.3 Kernels

Instead of using discrete bins we may use continuous bins, or *kernels*, to calculate the pdf $f(x)$. In this case, the equation corresponding to (7.9) is

[6] To avoid problems of overfitting (see Sect. 7.6.1) we often use the following equation: $f(x) = (n_m^{(2)} + 1)/(n_m^{(1)} + n_m^{(2)} + 2)$, where $x_m' \le x \le x_m''$, instead of (7.9).

$$f(x) = \frac{F^{(2)}}{F^{(1)} + F^{(2)}} \,, \tag{7.10}$$

where

$$F^{(1)} = \sum_{i=1}^{N} (1 - z_i) \times K\left(\frac{x - x_i}{H}\right) \,, \tag{7.11}$$

$$F^{(2)} = \sum_{i=1}^{N} z_i \times K\left(\frac{x - x_i}{H}\right) \,. \tag{7.12}$$

In (7.11-7.12) K is a bounded non-negative function satisfying $\int K(x)dx = 1$ and H is a positive number usually called the kernel *bandwidth* [7].

7.6.4 Isotonic Regression

In *isotonic regression* we do not know the shape of the function $y = f(x)$ except that it belongs to the class of *isotonic*, or non-decreasing, functions. Isotonic regression may therefore be regarded as an intermediary approach between sigmoid fitting and binning.

If we assume that the classifier ranks examples correctly, the function $f(x)$ from scores into probabilities is non-decreasing, and we can use isotonic regression to learn this mapping. A commonly used algorithm for computing the isotonic regression is the pair-adjacent violators (PAV) algorithm. This algorithm finds the stepwise-constant isotonic function that best fits the data according to a mean-squared error criterion.

Example 7.11. Isotonic Regression: Pair-Adjacent Violator Algorithm [347]. Let $O_i, i \in \{1, 2, \ldots, N\}$, be the training samples arranged such that the measurement values are in order of increasing value, i. e. $x_1 \leq x_2 \leq \ldots \leq x_N$. Let $f^*(x)$ be the sought after isotonic function. Let $f(x_i) = z_i$, then if f is already isotonic, we return $f^* = f$. Otherwise, there must be a subscript l such that $f(x_{l-1}) \leq f(x_l)$. The examples x_{l-1} and x_l are called *pair-adjacent violators*, because they violate the isotonic assumption. The values of $f(x_{l-1})$ and $f(x_l)$ are then replaced by their average, so that the $l-1$th and lth examples now comply with the isotonic assumption. If this new set of $N-1$ values is isotonic, then $f^*(x_{l-1}) = f^*(x_l) = (f(x_{l-1}) + f(x_l))/2$ otherwise $f^*(x_l) = f(x_l)$. This process is repeated using the new values until an isotonic set of values is obtained.

Fig. 7.5 illustrates the action of the PAV algorithm. □

[7] See Sect. 12.3.3 for information regarding the kernel method and how K and H are chosen.

7.6.5 Multi-Class Probability Estimates

Thus far we have considered two-class, or binary, probability estimates. We now apply the notion of calibration to *multi-class* probability estimates. Let $c_k, k \in \{1, 2, \ldots, K\}$, denote K classes. Then our aim is to find a multi-dimensional function $f(x)$ such that $f(x)$ closely approximates the *a posteriori* probability density function $p(\{c_k\}|x, I)$:

$$f(x) \approx p(\{c_k\}|x, I) \; . \tag{7.13}$$

The function $f(x)$ represents a multidimensional mapping from one K-dimensional space to another K-dimensional space. In this case, it is not clear what function shape we should assume for the transformation function. For this reason, the direct calibration of the multi-class probabilities is not recommended. Instead we reduce the multi-class problem into $K(K-1)/2$ binary, or two-class, problems and then combine the binary probability estimates using a *voting* or other similar algorithm.

The following example illustrates the use of a simple voting algorithm to combine the binary probability estimates.

Example 7.12. Multi-Class Probability Estimates Using a Voting Algorithm [334, 346]. We consider a multi-class probability estimate involving K classes $c_k, k \in \{1, 2, \ldots, K\}$. Let $f_{kl}(x|\alpha_{kl}, \beta_{kl})$ denote a probabilistic calibration curve for classes $c_k, c_l, l > k$. The curves f_{kl} may be obtained directly from

2.0	3.0	3.0	3.5	7.0	9.0	9.5	9.8	9.9	(a)
0	0	1	0	0	1	0	1	1	(b)
0	0	1	0	0	1	0	1	1	(c)
0	0	0.5	0.5	0	0.5	0.5	1	1	(d)
0	0	0.5	.25	.25	0.5	0.5	1	1	(e)
0.0	0.0	.33	.33	.33	0.5	0.5	1.0	1.0	(f)

Fig. 7.5. Shows the Pair-Adjacent Violator (PAV) algorithm in action. **(a)** Shows the training samples O_i arranged in order of increasing value x_i. **(b)** Shows class label z_i value for each sample O_i. **(c)** Boxes with heavy margins show the adjacent samples which violate isotonicity. **(d)** and **(e)** Shows two iterations of the PAV algorithm in which we average the labels of the adjacent samples which violate isotonicity. **(f)** Shows the final isotonic function $f^*(y)$.

the classification algorithm or may be estimated using any one of the methods listed in Sect. 7.6. Then the *a posteriori* probabilities $p(c_k|x, I)$ for an input x are:

$$p(C = c_k|x) = \frac{1}{Z} \sum_{\substack{l=1 \\ l \neq k}}^{K} f_{kl}(x|\alpha_{kl}, \beta_{kl}) \ ,$$

where $f_{kl}(x|\alpha_{kl}, \beta_{kl}) = 1 - f_{lk}(x|\alpha_{lk}, \beta_{lk})$ and $Z = \sum_{m=1}^{K} \sum_{n=1}^{K} f_{mn}(y|\alpha_{mn}, \beta_{mn})$ is a normalization factor which ensures the *a posteriori* probabilities sum to one. □

Additional methods for estimating the multi-class probabilities $\boldsymbol{p} = (p_1, p_2, \ldots, p_K)^T$ are listed in Table 7.5.

Table 7.5. Methods for Estimating Multi-Class Probabilities $\boldsymbol{p}$

Method	Description
PKPD	$p_k = -1/(K - 2 - \sum_{l,l \neq k} r_{kl}^{-1})$; where r_{kl} is the estimate that a given object O belongs to $C = c_k$ assuming it belongs to either $C = c_k$ or $C = c_l$ [250].
Hastie-Tibshani	$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_k [\sum_{l \neq k} (r_{kl}/K - p_k/2)]^2$ [114].
WLW_1	$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_k [\sum_{l \neq k} (r_{kl} p_l - r_{kl} p_k)]^2$ [334].
WLW_2	$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_k \sum_{l \neq k} (r_{kl} p_k - r_{lk} p_k)^2$)[334].

7.7 Software

LIBRA (Matlab Library for Robust analysis). A matlab toolbox for performing robust statistics. The toolbox contains m-files on various robust normalization techniques.

LIBSVM. A matlab toolbox for performing SVM. The toolbox contains an m-file for Platt calibration [179]. *Note:* This code is an improvement on the pseudo-code given in [244] which was found to give biased *a posteriori* probability estimates [207].

MATLAB STATISTICAL TOOLBOX. The matlab statistical toolbox. The toolbox contains m-files for performing various normalization procedures.

7.8 Further Reading

Similarity measures have been intensely investigated in many different fields. Willett and co-workers have published widely on the use of similarity measures

in the chemical industry (see e. g. [329, 330, 332]). A comprehensive review of these similarity measures is to be found in [19]. Binarization is widely used in applications involving image analysis. In these applications an important issue is specification of the threshold t [25]. Finally, the advantages of using *calibrated* probability estimates are discussed at length in [50].

Part III

Data Fusion

8

Bayesian Inference

8.1 Introduction

In this chapter we give an overview of Bayesian statistics, and in particular, the methods of Bayesian inference as used in multi-sensor data fusion. The basic premise of Bayesian statistics is that all unknowns are treated as random variables and that the knowledge of these quantities is summarized via a probability distribution. The main advantages of using Bayesian statistics are

1. Bayesian statistics is the only known coherent system for quantifying objective and subjective uncertainties.
2. Bayesian statistics provides principled methods for the model estimation and comparison and the classification of new observations.
3. Bayesian statistics provides a natural way to combine different sensor observations.
4. Bayesian statistics provides principle methods for dealing with missing information.

In Table 8.1 we list some of the basic formulas which are used in Bayesian statistics. In a multi-sensor data fusion system, we use Bayesian statistics to represent the multiple sources of information. The corresponding probability distributions act as a powerful *common representational format.*

8.2 Bayesian Analysis

In Bayesian statistics we treat all quantities under consideration as random variables. This includes the observed data, $\boldsymbol{y}$ and any missing data, $\boldsymbol{z}$, unknown parameters, $\boldsymbol{\theta}$, and models, M. The full process of Bayesian analysis can be described as consisting of three stages:

Probability Model. In this stage we create a joint probability distribution that captures the relationship among all the variables under consideration.

Table 8.1. Basic Formulas Used in Bayesian Inference

Name	Formula
Probability Density Function (pdf)	$p(y\|I)dy = P(y \leq Y \leq y + dy\|I)$.
Normalization	$\int p(y\|I)dy = 1$.
Expectation of $f(Y)$	$E(f(Y)) = \int f(y)p(y\|I)dy$.
Expected Value	$E(Y) = \int yp(y\|I)dy$.
Moment of Order r	$M_r(Y) = \int y^r p(y\|I)dy$.
Variance	$\sigma^2 = \int (y - E(Y))^2 p(y\|I)dy$.
Product Rule	$p(x, y\|I) = p(x\|y, I)p(y\|I)$.
Independence	$p(x, y\|I) = p(x\|I)p(y\|I)$.
Marginalization	$p(x\|I) = \int p(x, y\|I)dy$.
Decomposition	$p(x\|I) = \int p(x\|y, I)p(y\|I)dy$.
Bayes' Rule	$p(x\|y, I) = p(y\|x, I)p(x\|I)/p(y\|I)$; $p(x\|y, z, I) = p(x\|I)p(y\|x, I)p(z\|x, y, I)/(p(y\|I)p(z\|I))$.
Likelihood	$L = p(y\|x, I)$.

In the table X and Y denote two random variables whose instantiations are, respectively, x and y. For variables with continuous values we do not normally distinguish between the random variable and its instantiation. Thus we write $p(y|I)$ and not $p(Y = y|I)$. In the case of variables with discrete values this may, however, cause confusion. For these variables we shall therefore continue to write $p(Y = y|I)$. In the table, I describes any background information we may have concerning the problem in hand. For definitions of additional terms, see e. g. the glossary of selected statistical terms in [251].

A Posterior Distribution. In this stage we summarize all of our information regarding the different quantities of interest in a set of *a posteriori* distributions. Typically the *a posteriori* distribution is a conditional probability density function (pdf) $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$.

Model Selection. In this stage we evaluate the appropriateness of using a given model. We may then select the best model, or create a new model, or suggest improvements in one of the existing models.

The following example illustrates the role played by Bayesian analysis in the scientific process.

Example 8.1. Scientific Theories [5, 125, 190]. Fig. 8.1 is adapted from [190] and illustrates that part of the scientific process in which data is collected and modeled. The two heavily-framed boxes denote processes which involve Bayesian inference. In the first box we infer what the model parameters might

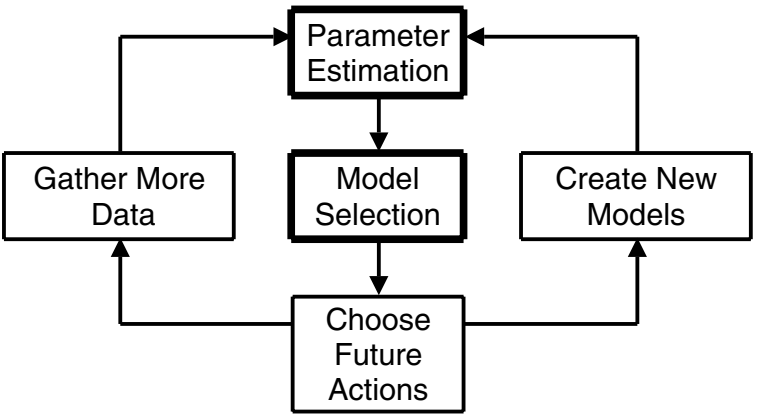

Fig. 8.1. Shows the role played by Bayesian inference in the scientific process.

be given the model and the data. In the second box we infer what model is probably closer to the correct model given the data. □

8.3 Probability Model

The standard procedure for setting up a full probability model, or joint probability distribution, is to write down the likelihood function, i. e. the probability of the observed data given the unknowns and multiply it by the *a priori* distribution of all the unobserved variables and parameters. We assume the observed data, $\boldsymbol{y}$, comes from a parametric pdf p characterized by the parameters $\boldsymbol{\theta}$. Then the joint pdf can be represented as

$$p(\boldsymbol{y}, \boldsymbol{\theta}|I) = p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I) \,, \tag{8.1}$$

where $\boldsymbol{\theta}$ is assumed to come from some *a priori* distribution, π, with parameters $\boldsymbol{\lambda}$ [1] and I denotes all of our relevant background knowledge.

A convenient representation of the model is the graphical model (Fig. 8.2). The input data, $\boldsymbol{y}$, the missing data, $\boldsymbol{z}$, and the model parameters, $\boldsymbol{\theta}$, are represented as nodes in a graph. Relationships and influences between the nodes are then represented as straight lines, or edges, which link the nodes.

[1] Hereinafter the parameters $\boldsymbol{\lambda}$ are referred to as *hyperparameters* to differentiate them from the parameters $\boldsymbol{\theta}$.

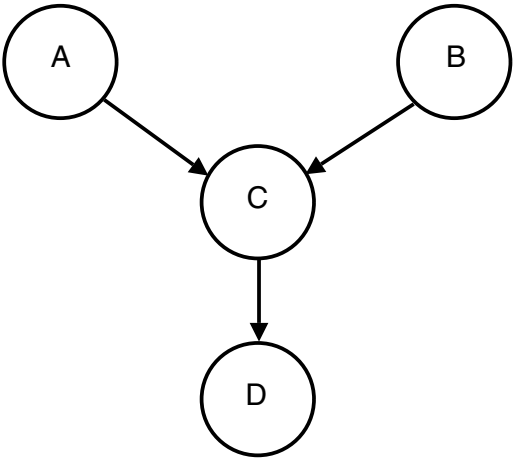

Fig. 8.2. Shows a simple graphical model with nodes $\{A, B, C, D\}$. The set $\{A, B\}$ is the parents of C and C is the child of $\{A, B\}$ and the parent of D. Mathematically we write these relationships as: $\text{pa}(C) = \{A, B\}$, $\text{ch}(A) = C$, $\text{ch}(B) = C$ and $\text{pa}(D) = C$. The parents and child of a node, plus the children's other parents, constitute a *Markov Blanket*. The Markov Blanket of A is $\{B, C\}$. Mathematically we write this as $\text{mk}(A) = \{\text{pa}(A), \text{ch}(A), \text{pa}(\text{ch}(A))\}$ or $\text{mk}(A) = \{B, C\}$.

8.4 A Posteriori Distribution

The Bayesian inference is drawn by examining the probability of all possible values of the parameters, $\boldsymbol{\theta}$, after considering the data, $\boldsymbol{y}$. If the hyperparameters, $\boldsymbol{\lambda}$, are known (or are estimated), then we obtain the *a posteriori* pdf through the application of Bayes' theorem

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{\theta}, I)}^{\text{likelihood}} \overbrace{\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)}^{a\ priori\ \text{pdf}}}{\underbrace{p(\boldsymbol{y}|I)}_{\text{evidence}}} ,$$

$$= \frac{p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)}{\int p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)d\boldsymbol{\theta}} . \tag{8.2}$$

The denominator $p(\boldsymbol{y}|I) = \int p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)d\boldsymbol{\theta}$ is known as the *evidence* and is a normalizing constant which is required so that $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$ will integrate to one. The evidence is obtained by integrating out all the variables, except for the observed data, $\boldsymbol{y}$, from the joint distribution.

If the hyperparameters, $\boldsymbol{\lambda}$, are not known, then we may also remove them by integration:

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \frac{\int p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)\pi(\boldsymbol{\lambda}|I)d\boldsymbol{\lambda}}{\int\int p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|\boldsymbol{\lambda}, I)\pi(\boldsymbol{\lambda}|I)d\boldsymbol{\lambda}d\boldsymbol{\theta}} , \tag{8.3}$$

where $\pi(\boldsymbol{\lambda}|I)$ is the *a priori* pdf for $\boldsymbol{\lambda}$ [2].

Equation (8.3) tells us how we may systematically update our knowledge of $\boldsymbol{\theta}$ given the data $\boldsymbol{y}$. For example, if the observations are obtained one at a time, we can update the *a posteriori* distribution as follows:

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{y}_1, I) &\sim p(\boldsymbol{y}_1|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|I) , \\ p(\boldsymbol{\theta}|\boldsymbol{y}_1, \boldsymbol{y}_2, I) &\sim p(\boldsymbol{y}_2|\boldsymbol{\theta}, I)p(\boldsymbol{\theta}|\boldsymbol{y}_1, I) , \\ &\dots \\ p(\boldsymbol{\theta}|\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N, I) &\sim p(\boldsymbol{y}_N|\boldsymbol{\theta}, I)p(\boldsymbol{\theta}|\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{N-1}, I) . \end{aligned} \tag{8.4}$$

When there is more than one unknown parameter, e. g. $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, and we are only interested in one component, say $\boldsymbol{\theta}_1$, those unknown quantities that are not of immediate interest, but are needed by the model, are known as *nuisance parameters*, and are removed by integration:

$$p(\boldsymbol{\theta}_1|\boldsymbol{y}) = \frac{\int p(\boldsymbol{y}|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, I)\pi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2|I)d\boldsymbol{\theta}_2}{\int p(\boldsymbol{y}|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, I)\pi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2|I)d\boldsymbol{\theta}_1 d\boldsymbol{\theta}_2} . \tag{8.5}$$

Despite the deceptively simple-looking form of (8.5), the application of Bayesian statistics can be difficult. The most challenging aspects being: (1) the development of a model, $p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}, I)$, which must effectively capture the key features of the underlying physical problem; and (2) the necessary computation required for deriving the *a posteriori* distribution.

8.4.1 Standard Probability Distribution Functions

In developing adequate models for the likelihood $p(\boldsymbol{\theta}|\boldsymbol{y})$ and the *a priori* and *a posteriori* distributions, $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$ and $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$, we often use standard mathematical distributions [360]. Table 8.2 is a list of the most important univariate distributions $f(y)$.

8.4.2 Conjugate Priors

An early effort at making the integration required by (8.2) accessible was the development of the *conjugate priors* [3, 99, 168]. Formally, a conjugate prior is a family of distributions for $\pi(\boldsymbol{\theta}|I)$ that has the same functional form as

[2] Hereinafter $h(\boldsymbol{\lambda})$ is referred to as a *hyperprior* distribution to differentiate it from the *a priori* distribution $\pi(\boldsymbol{\theta}|I)$.

Table 8.2. Important Univariate Distributions $f(y)$

Name	Formula
Beta $\mathcal{B}e(\alpha,\beta)$	$\Gamma(\alpha+\beta)y^{\alpha-1}(1-y)^{\beta-1}/(\Gamma(\alpha)\Gamma(\beta))$; $0\le y\le 1;\alpha,\beta>0$.
Binomial $\mathcal{B}in(n,q)$	$C(n,q)q^y(1-q)^{n-y}$; $n\in N$.
Cauchy $\mathcal{C}a(a,b)$	$1/b\pi\times(1+(y-a)^2/b^2)^{-1}$; $b>0$.
Chi-Square $\chi^2(n)$	$\exp-(y/2)y^{(n-2)/2}/(2^{n/2}\Gamma(n/2);y\ge 0,n\in N$.
Exponential $\mathcal{E}(\lambda)$	$\lambda\exp-(\lambda y);y\ge 0;\lambda>0$.
Gamma $\mathcal{G}a(\alpha,\beta)$	$y^{\alpha-1}\exp-(y/\beta)\beta^\alpha/\Gamma(\alpha);\alpha>0;\beta>0$.
Gaussian $\mathcal{N}(\mu,\sigma^2)$	$\frac{1}{\sqrt{2\pi\sigma^2}}\exp-\frac{1}{2}((y-\mu)/\sigma)^2;\sigma>0$.
Inverse Gamma $\mathcal{IG}(\alpha,\beta)$	$y^{-(\alpha+1)}\exp-(\beta/y)\beta^\alpha/\Gamma(\alpha);\ \alpha>0$.
Laplace $\mathcal{L}a(\mu,\sigma)$	$\exp-(\lvert y-\mu\rvert\sigma)/\sigma;\sigma>0$.
Logistic $\mathcal{L}o(\alpha,\beta)$	$\frac{1}{\beta}(\exp-(y-\alpha)/\beta)/(1+\exp-(y-\alpha/\beta))^2$.
Pareto $\mathcal{P}a(a,\theta)$	$\theta a^\theta/y^{\theta+1};y\ge a,\theta>2,a>0$.
Student-t $\mathcal{S}t(\mu,\sigma,\nu)$	$\Gamma((\nu+1)/2)/(\sigma\sqrt{\nu\pi}\Gamma(\nu/2))\times[1+(y-\mu)^2/\nu\sigma^2]^{-(\nu+1)/2};\nu\in N,\nu>2,\sigma>0$.
Uniform $\mathcal{U}(a,b)$	$1/(b-a),0\le y\le 1;a,b>0$.

the likelihood function. As a consequence, when a conjugate prior is used, the functional form of the *a posteriori* distribution is the same as that of the *a priori* distribution. Thus although we can choose any functional form for $\pi(\boldsymbol{\theta}|I)$, the conjugate prior enjoys the greatest mathematical and computational advantage. In Table 8.3 we list the important families of conjugate likelihood and *a priori* distributions [99, 168].

The following examples illustrate the use of Table 8.3 in calculating the *a posteriori* pdf for a Gaussian likelihood.

Table 8.3. Conjugate Distributions

Likelihood $p(y_i\|\theta,I)$	*A Priori* $\pi(\theta\|I)$	*A Posteriori* $p(\theta\|\{y_i\},I)$
$\mathcal{N}(\theta,\sigma^2)$[known σ^2]	$\mathcal{N}(\mu,\tau^2)$	$\mathcal{N}((N\tau^2\bar{y}+\sigma^2\mu)/\Delta,(\sigma^2\tau^2)/\Delta$; where $\Delta=\sigma^2+N\tau^2$.
$\mathcal{N}(\mu,1/\theta)$[known μ]	$\mathcal{G}a(\alpha,\beta)$	$\mathcal{G}a(\alpha+N/2,(2\beta+\sum_{i=1}^N(y_i-\mu)^2)/2)$
$\mathcal{N}(\theta_1,\theta_2)$		see Ex. 8.3
$\mathcal{P}ois(\theta)$	$\mathcal{G}a(\alpha,\beta)$	$\mathcal{G}a(\alpha+\sum_{i=1}^N y_i,\beta+N)$
$\mathcal{G}a(\nu,\theta)$[known ν]	$\mathcal{G}a(\alpha,\beta)$	$\mathcal{G}a(\alpha+N\nu,\beta+\sum_{i=1}^N y_i)$
$\mathcal{U}(0,\theta)$	$\mathcal{P}a(\theta_0,\alpha)$	$\mathcal{P}a(\max(\theta_0,y),\alpha+1)$
$\mathcal{G}a(\nu=\frac{1}{2},2\theta)$ [known ν]	$\mathcal{IG}(\alpha,\beta)$	$\mathcal{IG}(\frac{1}{2}+\alpha,1/(\frac{y}{2}+\beta^{-1}))$
$\mathcal{E}(\theta)$	$\mathcal{G}a(\alpha,\beta)$	$\mathcal{G}a(\alpha+1,\beta+\sum_{i=1}^N y_i)$
$\mathcal{B}in(n,\theta)$[known n]	$\mathcal{B}e(\alpha,\beta)$	$\mathcal{B}e(\alpha+y,\beta+n-y)$

Example 8.2. Conjugate Prior for a Gaussian Distribution with Known Variance [99]. We observe N independent samples $y_i, i \in \{1, 2, \ldots, N\}$, which are distributed according to the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with unknown mean μ. The variance σ^2 is assumed known and is set equal to the unbiased maximum likelihood value:

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu)^2 \ .$$

According to Table 8.3 the *a posteriori* pdf $p(\mu|\{y_i\}, I)$ is

$$p(\mu|\{y_i\}, \sigma^2 = \sigma^2_{\mathrm{ML}}, I) = \mathcal{N}\left(\frac{\tau^2 \sum_{i=1}^{N} y_i + \sigma^2_{\mathrm{ML}}\mu}{\sigma^2_{\mathrm{ML}} + N\tau^2}, \frac{\sigma^2_{\mathrm{ML}}\tau^2}{\sigma^2_{\mathrm{ML}} + N\tau^2}\right) \ . \qquad \square$$

Example 8.3. Conjugate Prior for a Gaussian Distribution with Unknown Mean and Unknown Variance [99]. We reconsider Ex. 8.2 but this time assume that both the mean μ and the variance σ^2 are unknown. The *a priori* distribution for μ and σ^2 is

$$\begin{aligned}\pi(\mu, \sigma^2|I) &= \pi(\mu|\sigma^2, I)\pi(\sigma^2|I) \ , \\ &= \mathcal{N}(\mu_0, \sigma^2/k_0)\mathcal{IG}(v_0, \sigma^2_0) \ .\end{aligned}$$

The joint *a posteriori* pdf is

$$p(\mu, \sigma^2|\{y_i\}, I) = \mathcal{N}\Big(\frac{k_0\mu_0 + N\bar{y}}{k_N}, \frac{\sigma^2}{k_N}\Big)\mathcal{IG}(v_N, \sigma^2_N) \ ,$$

where

$$\begin{aligned} k_N &= k_0 + N \ , \\ v_N &= v_0 + N \ , \\ \mu_N &= (k_0\mu_0 + N\bar{y})/k_N \ , \\ v_N\sigma^2_N &= v_0\sigma^2_0 + S + k_0 N(\bar{y} - \mu_0)^2/k_N \ , \\ S &= \sum_{i=1}^{N}(y_i - \bar{y})(y_i - \bar{y})^T \ . \end{aligned}$$

The marginal distributions $p(\mu|\{y_i\}, I)$ and $p(\sigma^2|\{y_i\}, I)$ are found, respectively, by integrating out σ^2 and μ. The results are

$$\begin{aligned} p(\mu|\{y_i\}, I) &= \int p(\mu, \sigma^2|\{y_i\}, I)d\sigma = \mathcal{S}t(\mu_N, \sigma^2_N/k_N, v_N) \ , \\ p(\sigma^2|\{y_i\}, I) &= \mathcal{IG}(v_N, \sigma^2_N) \ . \end{aligned} \qquad \square$$

8.4.3 Non-Informative Priors

Another class of *a priori* distributions is the *non-informative* or "flat" *a priori* distributions. These distributions are designed for those situations when we essentially we have no knowledge of what the true distributions should be. The simplest non-informative *a priori* distribution for a given variable θ is to assume $\pi(\theta|I)$ is proportional to one: $\pi(\theta|I) \propto 1$. However this choice of *a priori* pdf is not invariant to scale changes.

Considerations of this sort led to the concept of Jeffrey's *a priori* pdfs which are invariant to changes of variables. They are defined as follows:

$$\pi(\theta|I) \propto |I(\theta)|^{1/2} , \tag{8.6}$$

where $I(\theta) = -E(d^2 \log p(\boldsymbol{y}|\theta)/d\theta^2)$ is the *Fisher information*.

Other non-informative *a priori* distributions are the reference and the maximal data information prior (MDIP) distributions [147, 340]. In Table 8.4 we list the uniform (U), Jeffrey's (J), reference (R) and the MDIP (M) non-informative priors for a Gaussian distribution.

Note: The use of non-informative priors is controversal especially their use for multivariate distributions. As we explained not all of the priors given there are location and scale independent [3]. For example, for a Gaussian likelihood in which both μ and σ^2 are unknown, Ref [212] recommends $\pi(\mu, \sigma^2|I) \sim \frac{1}{\sigma^3}$ (this is the only *a priori* pdf which is invariant to changes in location and scale).

Example 8.4. Non-Informative Prior for a Gaussian Distribution with Unknown Mean and Unknown Variance [212]. We reconsider Ex. 8.3 but this time we use non-informative *a priori* distributions. We assume no *a priori*

Table 8.4. Non-Informative Priors Distributions

Likelihood $p(y_i\|\theta, I)$	*A Priori* $\pi(\theta\|I)$	*A Posteriori* $p(\theta\|\{y_i\}, I)$
$\mathcal{N}(\theta, \sigma^2)$ [σ^2 known]	1 (U,J,R,M)	$\mathcal{N}(\bar{y}, \sigma^2/N)$; $\bar{y} = \sum_{i=1}^N y_i/N$.
$\mathcal{N}(\mu, \theta)$ [μ known]	1 (U)	$\mathcal{IG}((N-2), 2/S^2)$; $S^2 = \sum_{i=1}^N (y_i - \bar{y})^2$.
	$1/\sigma^2$ (J, R, M)	$\mathcal{IG}(N/2, 2/S^2)$.
$\mathcal{N}(\theta_1, \theta_2)$	1 (U)	$\theta_1 \sim \mathcal{S}t(\bar{y}, S^2/(N-3), N-3)$.
		$\theta_2 \sim \mathcal{IG}((N-3)/2, 2/S^2)$.
	$1/\sigma^4$ (J)	$\theta_1 \sim \mathcal{S}t(\bar{y}, S^2/(N(N+1)), N+1)$.
		$\theta_2 \sim \mathcal{IG}((N+1)/2, 2/S^2)$.
	$1/\sigma^2$ (R,M)	$\theta_1 \sim \mathcal{S}t(\bar{y}, S^2/(N(N-1)), N-1)$.
		$\theta_2 \sim \mathcal{IG}((N-1)/2, 2/S^2)$.

[3] Nevertheless reasonable results are often obtained with *a priori* distributions which are not location or scale independent.

knowledge of μ and σ^2. In this case, a non-informative distribution, which is invariant to translation and scale changes, is

$$\begin{aligned}\pi(\mu, \sigma^2|I) &= \pi(\mu|\sigma^2, I)\pi(\sigma^2|I) \ , \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \times \frac{1}{\sigma^2} \ .\end{aligned}$$

By Bayes' rule, the corresponding *a posteriori* pdf is

$$p(\mu, \sigma^2|\{y_i\}, I) = \prod_{i=1}^{N} \frac{\mathcal{N}(y_i|\mu, \sigma^2)\pi(\mu, \sigma^2|I)}{p(y_i|I)} = \frac{1}{\sqrt{2\pi}\sigma^3} \prod_{i=1}^{N} \frac{\mathcal{N}(y_i|\mu, \sigma^2)}{p(y_i|I)} \ .$$

Ref [212] shows this may be rewritten as

$$p(\mu, \sigma^2|\{y_i\}, I) = \frac{1}{(\Gamma(N/2)\sigma^2)} \sqrt{\frac{N}{\pi S}} \Big(\frac{S}{2\sigma^2}\Big)^{(N+1)/2} \exp -\frac{1}{2\sigma^2}(N(\mu - \bar{y})^2 + S) \ ,$$

where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ and $S = \sum_{i=1}^{N}(y_i - \bar{y})^2$. □

8.4.4 Missing Data

In many problems, it is often fruitful to distinguish between two kinds of unknowns: parameters and hyperparameters on the one hand and *missing* data on the other hand. Although there is no absolute distinction between the two types of unknowns, the missing data is usually directly related to the individual data and its dimensionality tends to increase as more and more data are observed. On the other hand, the parameters and hyperparameters usually characterize the entire population of observations and are fixed in number.

When missing data, $\boldsymbol{z}$, occurs in a statistical problem, the inference can be achieved by using the "observed-data likelihood", defined as $L = p(\boldsymbol{y}|\boldsymbol{\theta}, I)$, which can be obtained by integrating out the missing data from the "complete-data likelihood", i. e.

$$L = \int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}, I) d\boldsymbol{z} \ . \tag{8.7}$$

Bayesian analysis for missing data problems can be achieved coherently through integration:

$$p(\boldsymbol{y}|I) \sim \int\int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}, I) p(\boldsymbol{\theta}|I) d\boldsymbol{z} d\boldsymbol{\theta} \ . \tag{8.8}$$

Examination of (8.8) shows that the $\boldsymbol{z}$ are treated as random variables and that the integration required for eliminating missing data is no different from that required for eliminating nuisance parameters and unknown hyperparameters.

Many applications which involve the use of missing data are solved using the *expectation-maximization* (EM) algorithm. The following example illustrates the use of the EM algorithm to model an arbitrary M-dimensional pdf as a sum of Gaussian densities.

Example 8.5. Gaussian Mixture Model [199]. In the Gaussian Mixture Model (GMM) we model a probability density function, $\boldsymbol{\theta} = f(\boldsymbol{y})$, as a sum of K Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k \in \{1, 2, \ldots, K\}$:

$$f(\boldsymbol{y}) \approx \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \ ,$$

where $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k \in \{1, 2, \ldots, K\}$, are unknown and are to be learnt from a set of training data $(\boldsymbol{\theta}_i, \boldsymbol{y}_i), i \in \{1, 2, \ldots, N\}$, subject to the constraints

$$\begin{aligned} \alpha_k &> 0 \ , \\ \sum_{k=1}^{K} \alpha_k &= 1. \end{aligned}$$

One way of learning the parameters $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, is to reformulate the mixture model in terms of missing data and then use the expectation-maximization (EM) algorithm.

We associate a "hidden" variable z_i with each $\boldsymbol{y}_i$, where z_i can only take on the values $\{1, 2, \ldots, K\}$. Then we suppose that each value $\boldsymbol{\theta}_i = f(\boldsymbol{y}_i)$ was generated by randomly choosing z_i from $\{1, 2, \ldots, K\}$ and drawing $\boldsymbol{y}_i$ from the Gaussian $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $k = z_i$. In this case, α_k is given by the proportion of z_i for which $z_i = k$:

$$\alpha_k = \frac{1}{N} \sum_{i=1}^{N} I_{ik} \ ,$$

where I_{ik} is an indicator function:

$$I_{ik} = \begin{cases} 1 \text{ if } z_i = k \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

The parameters $(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k \in \{1, 2, \ldots, K\}$, are calculated using the EM algorithm by repeating the following steps until convergence.

E-step. Calculate the *a posteriori* probability of the hidden variable, z_i, given the input data $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, and using the current parameter values $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$:

$$p(z_i, k|\boldsymbol{y}_i, \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim p(\boldsymbol{y}_i|z_i = k, \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) p(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k|z_i = k) \ .$$

M-step. Update the parameters $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$:

$$\alpha_k = \frac{1}{N} \sum_{i=1}^{N} w_{ik} \ ,$$

$$\boldsymbol{\mu}_k = \sum_{i=1}^{N} w_{ik} \boldsymbol{y}_i / \sum_{i=1}^{N} w_{ik} \ ,$$

$$\boldsymbol{\Sigma}_k = \sum_{i=1}^{N} w_{ik} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)(\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T / \sum_{i=1}^{N} w_{ik} \ ,$$

where

$$w_{ik} = p(z_i = k | \boldsymbol{y}_i, \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \ .$$

Although widely used, the EM/GMM algorithm has three major drawbacks.

1. The EM algorithm converges to a local minimum. The results obtained are therefore sensitive to the initial conditions used.
2. The EM algorithm is liable to fail as a result of singularities, or degeneracies, in the input data.
3. The GMM suffers from the "curse of dimensionality" (Sect. 12.2): As the dimensionality of the input data increases, the number of input vectors which are required to specify the means $\boldsymbol{\mu}_k, k \in \{1, 2, \ldots, K\}$, and the covariance matrices, $\boldsymbol{\Sigma}_k$, increases exponentially [4].

Recent research into the EM algorithm have shown that the impact of the first two drawbacks may be substantially reduced by carefully choosing the initial conditions and "regularizing" the covariance matrices [8]. With regard to the curse of dimensionality, the traditional solution has been to constrain the structure of the covariance matrices [5]. Recently, a less drastic solution has been proposed in which we use a mixture of probabilistic PCA's instead of a mixture of Gaussians (see Sect. 9.6.3). □

The following example illustrates an unusual application of the EM algorithm to perform principal component analysis (PCA).

Example 8.6. PCA Using the EM Algorithm [283]. The aim of principal component analysis (PCA) is to find a L-dimensional linear projection that best represents a set of M-dimensional input vectors $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$ (see Ex. 4.11). The conventional approach to performing PCA is to find the L

[4] Specification of a M-dimensional Gaussian pdf with a full covariance matrix requires $M(M+3)/2$ parameters.

[5] Traditionally the covariance matrix is constrained to be diagonal (see Sect. 12.2). In this case the number of parameters required to specify a M-dimensional Gaussian pdf with a diagonal covariance matrix is $2M$.

dominant eigenvectors $\boldsymbol{u}_l, l \in \{1, 2, \ldots, L\}$, of the sample covariance matrix $\boldsymbol{\Sigma}$, where

$$\boldsymbol{\Sigma}\boldsymbol{u}_l = \lambda_l \boldsymbol{u}_l \ ,$$

and $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_M$. This calculation requires $\sim O(NM^2)$ operations whereas the following EM algorithm [283] performs the PCA in $\sim O(LNM)$ operations per iteration.

E-step. Project the $\widetilde{\boldsymbol{y}}_i = \boldsymbol{y}_i - \boldsymbol{\mu}$ onto an assumed set of principal axes $\boldsymbol{U}$, where $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_L)$. The result is a $L \times N$ projection matrix $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N)$, where

$$\boldsymbol{\theta} = (\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{U}^T\widetilde{\boldsymbol{y}} \ ,$$

and $\widetilde{y} = (\widetilde{\boldsymbol{y}}_1, \widetilde{\boldsymbol{y}}_2, \ldots, \widetilde{\boldsymbol{y}}_N)$ is a $M \times N$ matrix of input measurements with their mean $\boldsymbol{\mu}$ removed.

M-step. Find new principal axes $\boldsymbol{U}$ which minimize the squared reconstruction error. The new principal axes are

$$\boldsymbol{U} = \widetilde{\boldsymbol{y}}\boldsymbol{\theta}^T(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1} \ . \qquad \square$$

8.5 Model Selection

At times we may have more than one model, and then our interest may focus on assessing the fitness of each model and conducting model selection. To illustrate the Bayesian model selection procedure, we focus on the comparison between a "null" model, $M = 0$, and an alternative model, $M = 1$. In this case, the corresponding joint pdf is:

$$p(\boldsymbol{y}, \boldsymbol{\theta}, M = m|I) = p(\boldsymbol{y}|\boldsymbol{\theta}, M = m, I)\pi(\boldsymbol{\theta}, M = m|I) \ . \tag{8.9}$$

If we assume that $\boldsymbol{y}$ depends on the models through their parameters, then (8.9) becomes:

$$p(\boldsymbol{y}, \boldsymbol{\theta}, M = m|I) = p(\boldsymbol{y}|\boldsymbol{\theta}_m)\pi(\boldsymbol{\theta}_m|M = m, I)\pi(M = m|I) \ , \tag{8.10}$$

where $\pi(\boldsymbol{\theta}_m|M = m, I)$ is the *a priori* distribution for the parameters, $\boldsymbol{\theta}_m$, given the model $M = m$ and $\pi(M = m|I)$ is the *a priori* probability of the model $M = m$. The *a posteriori* probability for the model $M = m$ is proportional to the evidence,

$$p(M = m|\boldsymbol{y}, I) \propto \int p(\boldsymbol{y}|\boldsymbol{\theta}_m, I)\pi(\boldsymbol{\theta}_m|M = m, I)\pi(M = m|I)d\boldsymbol{\theta}_m \ , \tag{8.11}$$

and the optimal model is

$$m_{\text{OPT}} = \arg\max_m \big(p(M = m|\boldsymbol{y}, I)\big) \ . \tag{8.12}$$

The choice of the *a priori* distribution, $\pi(M = m|I)$, is clearly dependent on the application. If both models appear to be equally likely and contain the same number of parameters, then $\pi(M = 0|I) = \frac{1}{2} = \pi(M = 1|I)$.

In comparing different models we often use a simple approximation to the *a posteriori* pdf, $p(\boldsymbol{y}|\boldsymbol{\theta}, M = m, I)$, which is known as the *Laplace approximation*.

8.5.1 Laplace Approximation

The Laplace approximation assumes that the *a posteriori* distribution, $p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I)$, has only one mode, which we approximate with a Gaussian function. We proceed as follows. Let

$$\begin{aligned} t(\boldsymbol{\theta}) &= \ln p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I) \ , \\ &= \ln p(\boldsymbol{y}|\boldsymbol{\theta}, M = m, I) + \ln \pi(\boldsymbol{\theta}|M = m, I) + \text{const} \ . \end{aligned} \tag{8.13}$$

Then we expand $t(\boldsymbol{\theta})$ about $\hat{\boldsymbol{\theta}}$, its maximum *a posteriori* (MAP) value, i. e. around the value of $\boldsymbol{\theta}$ that maximizes $p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I)$:

$$\begin{aligned} t(\boldsymbol{\theta}) &= t(\hat{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \left.\frac{\partial t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \left.\frac{\partial^2 t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) + \ldots \\ &\approx t(\hat{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) H(\boldsymbol{\theta}) (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \ , \end{aligned} \tag{8.14}$$

where $H(\hat{\boldsymbol{\theta}})$ is the Hessian of the natural logarithm of the *a posteriori* distribution and is defined as the matrix of the second derivative of $t(\boldsymbol{\theta})$ evaluated at $\hat{\boldsymbol{\theta}}$,

$$H(\hat{\boldsymbol{\theta}}) = \left.\frac{\partial^2 \ln p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \ . \tag{8.15}$$

Substituting (8.14) into the logarithm of (8.9) and integrating over $\boldsymbol{\theta}$, yields

$$\begin{aligned} \ln p(\boldsymbol{y}|M = m, I) &= \ln \int \exp\big(t(\boldsymbol{\theta})\big) d\boldsymbol{\theta} \ , \\ &\approx t(\hat{\boldsymbol{\theta}}) + \frac{1}{2} \ln |2\pi H^{-1}| \ , \\ &\approx \ln p(\boldsymbol{y}|\hat{\boldsymbol{\theta}}, M = m, I) + \ln \pi(\hat{\boldsymbol{\theta}}|M = m, I) + \\ &\quad \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |H| \ , \end{aligned} \tag{8.16}$$

where d is the dimensionality of the parameter space $\boldsymbol{\theta}$. The corresponding approximate expression for $p(\boldsymbol{y}|M = m, I)$ is

$$p(\boldsymbol{y}|M = m, I) \approx \sqrt{(2\pi)^d |H^{-1}|}\, p(\boldsymbol{y}|\hat{\boldsymbol{\theta}}, M = m, I) \pi(\hat{\boldsymbol{\theta}}|M = m, I) \ . \tag{8.17}$$

In Table 8.5 we list some approximations to the natural logarithm of the Bayesian evidence which are commonly used in model selection.

Table 8.5. Approximations to the Natural Logarithm of the Bayesian Evidence $\ln p(\boldsymbol{y}|M=m,I)$

Name	Approximation to $\ln p(\boldsymbol{y}|M=m,I)$
Laplace Approximation	$\ln p(\boldsymbol{y}|\hat{\boldsymbol{\theta}},M=m,I)+\ln\pi(\hat{\boldsymbol{\theta}}|M=m,I)+\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln\lvert H\rvert$, where $\hat{\boldsymbol{\theta}}=\arg\max(p(\boldsymbol{\theta}|\boldsymbol{y},M=m,I))$, where d is the dimensionality of the parameter space $\boldsymbol{\theta}$.
Bayesian Information Criterion (BIC)	$\ln p(\boldsymbol{y}|\hat{\boldsymbol{\theta}},\ M=m,I)-\ \frac{d}{2}\ln N.$
Akaike Information Criterion (AIC)	$\ln p(\boldsymbol{y}|\ \hat{\boldsymbol{\theta}},M=m,I)-\ d.$
Cheeseman and Stutz Approximation	$\ln p(\boldsymbol{y},\hat{\boldsymbol{z}}|M=m,I)+\ln p(\boldsymbol{y}|\hat{\boldsymbol{\theta}},M=m,I)-\ln p(\boldsymbol{y},\hat{\boldsymbol{z}}|\ \hat{\boldsymbol{\theta}},M=m,I)$, where $\hat{\boldsymbol{z}}=\arg\max p(\boldsymbol{z}|\boldsymbol{y},\hat{\boldsymbol{\theta}},M=m,I)$.

In the following example we illustrate the concept of Bayesian model selection by using it to select the optimum number of Gaussians in a Gaussian mixture model (GMM) [199].

Example 8.7. Gaussian Mixture Model [83]. In a Gaussian mixture model (GMM) (see Ex. 8.5) we approximate a function $f(y)$ using a sum of K Gaussian pdf's:

$$f(y)\approx\sum_{k=1}^{K}\alpha_k\mathcal{N}(y|\mu_k,\sigma_k^2)\ ,$$

subject to the constraints

$$\begin{aligned}\alpha_k&>0\ ,\\ \sum_k\alpha_k&=1\ .\end{aligned}$$

Traditionally the user supplies the value of K. However, if the user is unable to supply a value for K, it may be determined by selecting the best GMM from a set of candidate models, each model having a different number of Gaussians. After optimally fitting the parameters of the different models (using the EM algorithm) we find the optimum value for K (i. e. the optimum model) by choosing the model with the largest Bayesian evidence. □

The following example illustrates the use of Bayesian model selection to estimate the intrinsic dimensionality of a given data set $\{y_i\}$.

Example 8.8. Intrinsic Data Dimensionality [209]. A central issue in pattern recognition is to estimate the intrinsic dimensionality d of the input data $\{\boldsymbol{y}_i\}$. Ref. [34] provides a modern reference to data dimensionality estimation methods. One approach is to identify d as the optimum number principal components we need to retain in a principal component analysis (PCA) of the

$\{\boldsymbol{y}_i\}$ (see Sect. 4.5). By interpreting PCA within a Bayesian framework, we are able to derive the following BIC approximation for the evidence:

$$p(\{\boldsymbol{y}_i\}|L) = \Big(\prod_{j=1}^{L} \lambda_j\Big)^{-N/2} \sigma^{-N(D-L)} N^{-(m+L)/2} ,$$

where D is the dimension of the input data and $m = DL - L(L+1)/2$. The intrinsic dimenionality of the input data is then defined as

$$d = L_{\max} = \arg\max p(\{\boldsymbol{y}_i\}|L) .$$

See [209] for a derivation of this approximation. □

Apart from the model selection methods listed in Table 8.5 there are a large number of non-Bayesian model selection methods available. The more popular of these methods are listed in Table 12.9. Recently, Bouchard and Celeux [26] have questioned the use of the Bayesian model selection approximations given in Table 8.5 for applications involving the classification of a given object O. They suggest a new criterion, known as *Bayesian Entropy Criterion*, which is defined as follows. Suppose each measurement $\boldsymbol{y}_i$ has a label z_i associated with it. Then,

$$\ln p(\boldsymbol{z}|\boldsymbol{y}, M = m, I) \approx \ln p(\boldsymbol{y}, \boldsymbol{z}|\hat{\boldsymbol{\theta}}, M = m, I) - \ln p(\boldsymbol{y}|\widetilde{\boldsymbol{\theta}}, M = m, I) , \quad (8.18)$$

where $\widetilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln p(\boldsymbol{y}|\boldsymbol{\theta}, M = m, I)$.

8.5.2 Bayesian Model Averaging

An alternative to model selection is Bayesian Model Averaging (BMA) [119]. In BMA we do not select a single "optimal" model. Instead we take the uncertainty involved in selecting an optimal model into consideration by averaging over the *a posteriori* distribution of multiple models. In a BMA framework, the *a posteriori* pdf, $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$, is given by the following formula:

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{y}, I) &= \sum_m p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I) p(M = m|\boldsymbol{y}, I) , \\ &= \frac{\sum_m p(\boldsymbol{\theta}|\boldsymbol{y}, M = m, I)\pi(M = m|I)p(\boldsymbol{y}|M = m, I)}{\sum_m \pi(M = m|I)p(\boldsymbol{y}|M = m, I)} , \quad (8.19) \end{aligned}$$

where $\pi(M = m|I)$ is the *a priori* probability of the model $M = m$. In general, BMA has a better performance than any single model that could reasonably have been selected. In practice, we limit the summation in (8.19) to a small number of models [6]. In Sect. 13.2 we illustrate the use of the BMA as a general framework for combining classifiers in a multiple classifier system (MCS).

[6] Madigan and Raftery [191] recommend discarding all models whose *a priori* probabilities $\pi(M = m|I)$ are less than 5% of the *a priori* probability of the optimum model.

8.6 Computation

In many practical problems, the required computation is the main obstacle to applying the Bayesian method. In fact, until recently, this computation was so difficult that practical applications employing Bayesian statistics were very few in number. The introduction of iterative simulation methods, such as data augmentation and the more general Markov chain Monte Carlo (MCMC), which provide Monte Carlo approximations for the required optimizations and integrations, has brought the Bayesian method into mainstream applications.

Table 8.6 lists some of the optimization and integration algorithms which are used in Bayesian inference (see also algorithms given in Table 8.5). *Note*: Apart from the method of conjugate priors all the algorithms in Table 8.6 are approximate.

8.6.1 Markov Chain Monte Carlo

Although the integrals in Bayesian inference are often intractable, sampling methods can be used to numerically evaluate them. There are a wide range of practical sampling techniques which go under the generic name "Monte Carlo". Given an *a posteriori* pdf, $p(\boldsymbol{y}|I)$, we may approximate the integral $\int f(\boldsymbol{y})p(\boldsymbol{y}|I)d\boldsymbol{y}$ by using the average of N independent samples, $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$, which are drawn from $p(\boldsymbol{y}|I)$. Mathematically, we have

$$\int f(\boldsymbol{y})p(\boldsymbol{y}|I)d\boldsymbol{y} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{y}_i) \; . \tag{8.20}$$

Unfortunately, in many practical applications, $p(\boldsymbol{y}|I)$ is only non-zero in an extremely small region of the parameter, or $\boldsymbol{y}$, space. The problem, then, is how

Table 8.6. Algorithms Used in Bayesian Inference

Name	Description
Conjugate Priors	This is an *exact* method for Bayesian Inference.
Laplace Approximation	This is an approximate method which is often used in point parameter estimation and in model selection.
Variational Algorithms	This uses a set of variational parameters which are iteratively updated so as to minimize the cross-entropy between the approximate and the true probability density.
Belief Propagation (BP)	This involves propagating the probability distributions using a local message passing algorithm.
Expectation-Maximization (EM)	This is an optimization algorithm which is very convenient to use when the problem involves missing data.
Markov Chain Monte Carlo (MCMC)	This is a relatively new method which has quickly established itself as a favourite all-purpose algorithm for Bayesian inference.

to efficiently generate the samples $\boldsymbol{y}_i$. The Markov chain Monte Carlo method works by constructing a random walk (Markov chain) in the parameter space such that the probability for being in a region of the space is proportional to the posterior density for that region. We start with some value of $\boldsymbol{y}$, calculate the *a posteriori* density there, and take a small step and then recalculate the density. The step is accepted or rejected according to some rule and the process is repeated. The resulting output is a series of points in the sample space. The various methods differ according to the rules used to make the moves in the parameter space and the rules determining whether or not a step is accepted. It turns out to be relatively simple to invent such rules that are guaranteed to produce a set of steps that correctly sample the *a posteriori* distribution.

8.7 Software

BAYESLAB. A collection of m-files for performing Bayesian inference.

BNT (Bayes' Net Toolbox). A matlab toobox for performing Bayesian inference.

GLMLAB (Generalized Linear Model Laboratory). A matlab toolbox for fitting and learning generalised linear models.

GMMBAYES. A matlab toolbox devoted to the Gaussian mixture model.

MCMCSTUFF. A matlab toolbox for implementation of the MCMC technique.

NETLAB. A matlab toolbox for performing neural network pattern recognition. The toolbox contains m-files on the Gaussian mixture model and the EM algorithm.

STPRTOOL (Statistical Pattern Recognition Toolbox). A matlab toolbox for performing statistical pattern recognition. The toolbox contains m-files on the Gaussian mixture model and the EM algorithm.

8.8 Further Reading

A useful glossary of selected statistical terms is to be found in [251]. Several modern textbooks on Bayesian inference are: [3, 99, 168, 190, 206, 280]. In addition there is a wide choice of specialist review articles on Bayesian inference which includes [4, 64, 189]. Useful books and review articles on approximate methods for Bayesian inference include [15, 78, 210, 264]. Specific references on MCMC are: [96, 102] and on the EM algorithm are: [8, 23, 199, 200]. Full-length discussions on model selection are given in [32, 189, 190]. For a discussion of the philosophy which underlays the Bayesian paradigm we recommend [5, 125]. Finally mention should be made of a full-length biography of Thomas Bayes [57].

9

Parameter Estimation

9.1 Introduction

In this chapter we consider *parameter estimation*, which is our first application of Bayesian statistics to the problem of multi-sensor data fusion. Let $\boldsymbol{y} = (\boldsymbol{y}_i^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_N^T)^T$ denote a set of N multi-dimensional sensor observations $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, where $\boldsymbol{y}_i = (y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(M)})^T$. We suppose the $\boldsymbol{y}_i$ were generated by a parameteric distribution $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$, where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_K)^T$ are the parameters and I is any background information we may have. Then parameter estimation involves estimating $\boldsymbol{\theta}$ from $\boldsymbol{y}$.

In the Bayesian framework we assume an *a priori* distribution $\pi(\boldsymbol{\theta}|I)$ for $\boldsymbol{\theta}$, where I denotes any background information we may have regarding $\boldsymbol{\theta}$. The *a posteriori* distribution, $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$, represents the probability density function of $\boldsymbol{\theta}$ after observing $\boldsymbol{y}$ and the relation between the *a priori* and the *a posteriori* distributions is given by Bayes' Theorem:

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|I)}{p(\boldsymbol{y}|I)} \ . \tag{9.1}$$

In many applications, parameter estimation represents the major fusion algorithm in a multi-sensor data fusion system. In Table 9.1 we list some of these applications together with the classification of the type of fusion algorithm involved.

9.2 Parameter Estimation

Once the *a posterior* distribution is available we can use it to estimate the parameters $\boldsymbol{\theta}$. In what follows we shall suppose $\boldsymbol{\theta}$ is a scalar parameter, θ. We define a *loss function*, $L = L(t, \theta)$, which links θ to its estimated value t. By

Table 9.1. Applications in which Parameter Estimation is the Primary Fusion Algorithm

Class		Application
DaI-DaO	Ex. 2.3	Time-of-Flight Ultrasonic Sensor.
	Ex. 3.2	Image Smoothing.
	Ex. 3.6	Heart Rate Estimation Using a Kalman Filter.
	Ex. 4.7	Debiased Cartesian Coordinates.
	Ex. 7.3	Psychometric Functions.
	Ex. 9.4	Kriging with an External Trend: Fusing Digital Elevation Model and Rainfall Gauge Measurements.
DaI-FeO	Ex. 4.2	Bayesian Triangulation of Direction-Bearing Measurements.
	Ex. 4.11	Principal Component Analysis.
	Ex. 4.12	The Chen-Yang LDA.
	Ex. 4.13	Chemical Sensor Analysis: Principal Discriminant Analysis (PDA) for Small Sample Size Problems.
	Ex. 9.5	Fitting a Finite-Length Straight-Line Segment in Two Dimensions: A Bayesian Solution.
	Ex. 9.6	Non-Intrusive Speech Quality Estimation Using GMM's.
	Ex. 10.10	Target track initialization using the Hough transform
FeI-FeO	Ex. 7.7	Fusing Similarity Coefficients in a Virtual Screening Programme.
	Ex. 7.10	Converting SVM Scores into Probability Estimates.
	Ex. 12.5	Feature Selection: Drug Discovery.
	Ex. 12.6	mrMR: Filter Feature Selection.

The designations DaI-DaO, DaI-FeO and FaI-FeO refer, respectively, to the Dasarathy input/output classifications: "Data Input-Data Output", "Data Input-Feature Output" and "Feature Input-Feature Output" (see Sect. 1.3.3).

minimizing the expected value of L, we obtain the *optimum* estimated value $\hat{\theta}$, where

$$\hat{\theta} = \min_t \int_{-\infty}^{\infty} L(t,\theta) p(\theta|\boldsymbol{y}, I) d\theta \; . \tag{9.2}$$

Clearly the estimates, $\hat{\theta}$, will vary with the choice of loss function. Three commonly used loss functions are:

Quadratic Loss Function. This is defined as $L(t,\theta) = \Delta^2$, where $\Delta = t - \theta$. In this case, the corresponding MMSE (minimum mean square error) estimate, $\hat{\theta}_{\mathrm{MMSE}}$, is equal to the expected value of θ:

$$\hat{\theta}_{\mathrm{MMSE}} = E(\theta) \; . \tag{9.3}$$

Absolute Loss Function. This is defined as $L(t,\theta) = |\Delta|$. In this case, the corresponding MMAE (minimum mean absolute error) estimate, $\hat{\theta}_{\mathrm{MMAE}}$, is equal to the median of the distribution of θ:

$$\hat{\theta}_{\mathrm{MMAE}} = \arg\mathrm{med}\big(p(\theta|\boldsymbol{y}, I)\big) \; . \tag{9.4}$$

Zero-One Loss Function. This is defined as:

$$L(t,\theta) = \begin{cases} 0 & \text{if } |\Delta| \le b \\ 1 & \text{if } |\Delta| > b \end{cases} , \tag{9.5}$$

In this case, the corresponding estimate is the center of the interval of width $2b$ which has the highest probability. If b goes to zero, then the zero-one estimate converges to the maximum *a posteriori* (MAP) estimate:

$$\hat{\theta}_{\text{MAP}} = \arg\max\big(p(\theta|\boldsymbol{y}, I)\big) . \tag{9.6}$$

Assuming $p(\theta|\boldsymbol{y}, I)$ is differentiable, $\hat{\theta}_{\text{MAP}}$ is found from

$$\left.\frac{\partial \ln p(\theta|\boldsymbol{y}, I)}{\partial\theta}\right|_{\theta=\hat{\theta}_{\text{MAP}}} = 0 . \tag{9.7}$$

Although the above loss functions are widely used, in some circumstances it may be better to use a different loss function. The following example illustrates one such case and involves the use of a *robust* version of the quadratic loss function [1].

Example 9.1. Color Constancy and the Minimum Local Mass (MLM) Loss Function [29]. Color constancy is defined as the ability of a visual system to maintain object colour appearance across variations in factors extrinsic to the object. Human vision exhibits approximate color constancy. Let $p(\theta|\boldsymbol{y}, I)$ denote the *a posteriori* distribution for the illuminants and the surface colors in a given scene. By definition, the best estimate of the surface color estimate is

$$\hat{\theta} = \min_t \int_{-\infty}^{\infty} L(t,\theta) p(\theta|\boldsymbol{y}, I) d\theta ,$$

where L is an appropriate loss function. For this application, the authors in [29], found that none of the above loss functions were appropriate. Instead they used a "robust" version of the quadratic loss function:

$$L(t,\theta) = \begin{cases} \Delta^2 & \text{if } |\Delta| < \alpha , \\ \alpha^2 & \text{otherwise} , \end{cases}$$

where $\Delta = t - \theta$ and α is a fixed constant.

In Table 9.2 we list some of the common loss functions along with their Bayesian estimators. □

The following example illustrates the calculation of the MAP estimate for the exponential distribution.

[1] The concept of a *robust* loss function is explained in more detail in Chapt. 10, where we consider the subject of *robust* statistics.

Table 9.2. Loss Functions $L(t,\theta)$ and their Bayesian Estimators $\hat{\theta}$

Name	$L(t,\theta)$	$\hat{\theta} \equiv \min_t \int_{-\infty}^{\infty} L(t,\theta)p(\theta\|\boldsymbol{y},I)d\theta$
Quadratic	Δ^2	$\hat{\theta} = E(p(\theta\|\boldsymbol{y},I))$, i. e. $\hat{\theta}$ is the expectation of $p(\theta\|\boldsymbol{y})$.
Absolute	$\|\Delta\|$	$\hat{\theta}$ is the median of $p(\theta\|\boldsymbol{y},I)$, i. e. $\int_{-\infty}^{\hat{\theta}} p(\theta\|\boldsymbol{y},I)d\theta = \frac{1}{2} = \int_{\hat{\theta}}^{\infty} p(\theta\|\boldsymbol{y},I)d\theta$.
Zero-One	1 if $(\|\Delta\| - b) \geq 0$, otherwise 0	$\hat{\theta}$ is the center of the interval of width $2b$ which has the highest probability. If $b \to 0$, then $\hat{\theta} = \arg\max p(\theta\|\boldsymbol{y},I)$.
Asymmetric Quadratic	$a\Delta^2$ if $\theta \geq t$, otherwise $b\Delta^2$	$\hat{\theta} = \min_t \int L(t,\theta)p(\theta\|\boldsymbol{x},I)d\theta$.
Asymmetric Absolute	$a\|\Delta\|$ if $\theta \geq t$ otherwise $b\|\Delta\|$	$\hat{\theta}$ is the $b/(a+b)$th quantile of $p(\theta\|\boldsymbol{y},I)$, i. e. $\int_{-\infty}^{\hat{\theta}} p(\theta\|\boldsymbol{y},I)d\theta = b/(a+b)$, $\int_{\hat{\theta}}^{\infty} p(\theta\|\boldsymbol{y},I)d\theta = a/(a+b)$.
Linex	$\beta(\exp(\alpha\Delta) - \alpha\Delta - 1)$	$\hat{\theta} = -\ln E(\exp -\alpha\theta)/\alpha$. If $y \sim \mathcal{N}(\mu,\sigma^2)$, then $\hat{\theta} = \mu - \alpha\sigma^2/2$.
Local Mass	$\min(\Delta^2, \alpha^2)$	$\hat{\theta} = \min_t \int L(t,\theta)p(\theta\|\boldsymbol{y},I)d\theta$.
Hubert-t	$\min(\|\Delta\|, \alpha)$	$\hat{\theta} = \min_t \int L(t,\theta)p(\theta\|\boldsymbol{y},I)d\theta$.

Example 9.2. Exponential Distribution. We suppose the likelihood $p(y|\theta,I)$ is exponential:

$$p(y|\theta,I) = \begin{cases} \theta\exp(-\theta y) & \text{if } y > 0\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

The corresponding prior conjugate distribution is also exponential:

$$\pi(\theta|I) = \begin{cases} \alpha\exp(-\alpha\theta) & \text{if } \theta > 0\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

By definition, the MMSE estimate is given by

$$\hat{\theta}_{\mathrm{MMSE}}(y) = \int_{-\infty}^{\infty} \theta p(\theta|y,I)d\theta\ ,$$

where

$$p(\theta|y,I) = \frac{p(y|\theta,I)\pi(\theta|I)}{\int p(y|\theta,I)\pi(\theta|I)d\theta}\ .$$

Substituting the expressions for $p(y|\theta,I)$ and $\pi(\theta|I)$ into $\hat{\theta}_{\mathrm{MMSE}}(y)$ gives

$$\begin{aligned} \hat{\theta}_{\mathrm{MMSE}}(y) &= \int_0^{\infty} (\alpha+y)^2\theta^2\exp\bigl(-(\alpha+y)\theta\bigr)d\theta\ , \\ &= \frac{2}{\alpha+y}\ . \end{aligned}$$

By using the appropriate loss function we may also find the MMAE and MAP solutions. All three solutions are listed in Table 9.3. □

Table 9.3. Bayesian Estimators for the Exponential Distribution

Estimator	Formula
MMSE	$\hat{\theta} = 2/(\alpha + y)$.
MMAE	$\hat{\theta} = T/(\alpha + y)$, where $T \approx 1.68$ is the solution of $(1 + T)\exp(-T) = \frac{1}{2}$.
MAP	$\hat{\theta} = 1/(\alpha + y)$.

9.3 Bayesian Curve Fitting

Curve fitting is an important technique in multi-sensor data fusion where it is commonly used to fuse together multiple sensor observations. We may regard curve fitting as a problem in parameter estimation: Given a set of pairs of input measurements $(y_i, u_i), i \in \{1, 2, \ldots, N\}$, we estimate the parameters of an algebraic curve which best fits these data. A classic example of curve fitting is straight-line regression where we fit a straight-line to several sensor measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$.

The following example illustrates the use of multiple straight-line regression algorithms in a multi-target tracking applications.

Example 9.3. Track Initialization Using a Multiple Straight-Line Regression Algorithm [13, 24]. In surveillance applications we often use a Kalman filter (Sect. 11.3) to track all of the objects which move in a given surveillance area. The Kalman filter is a sequential algorithm in which the tracks are updated as new sensor observations are received. To initiate the Kalman filter we must, however, create an initial set of tracks, one track for each object. This is often performed using a Hough transform (see Ex. 10.10). □

A curve fitting technique which is widely used in multi-sensor data fusion is Kriging (see Sects. 4.3 and 5.5). The following example illustrates the fusion of a digital elevation model and rain gauge measurements using a Kriging estimator.

Example 9.4. Kriging with an External Trend: Fusing Digital Elevation Model and Rainfall Gauge Measurements [106]. High resolution estimates of the spatial variability of rainfall are important for the identification of locally intense storms which can lead to floods and especially to flash floods. Given a limited number of rainfall gauges we may generate the rainfall at any point $\boldsymbol{u}_0$ by ordinary Kriging, or curve-fitting the N nearby rain gauge measurements $y(\boldsymbol{u}_i), i \in \{1, 2, \ldots, N\}$ (see Sect. 4.3). We now consider the situation where the rain gauge measurements are supplemented by elevation data which we assume is correlated with the rainfall and which is available for all $\boldsymbol{u}$.

In Kriging wixth an external trend (KED) we estimate the rainfall at the point $\boldsymbol{u}_0$ using a Kriging estimator (4.5):

$$\hat{y}(\boldsymbol{u}_0) = \mu(\boldsymbol{u}_0) + \sum_{i=1}^{N} \lambda_i (y_i - \mu(\boldsymbol{u}_0)) \ ,$$

in which we model the mean value $\mu(\boldsymbol{u}_0)$ as a linear function of z. The unknown mean, $\mu_0 = \mu(\boldsymbol{u}_0)$, and the weights, λ_i, are given by the following matrix equation:

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \mu_0 \end{pmatrix} = \begin{pmatrix} \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_1) & \dots & \Sigma(\boldsymbol{u}_1, \boldsymbol{u}_N) & 1 & z(\boldsymbol{u}_1) \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \Sigma(\boldsymbol{u}_N, \boldsymbol{u}_1) & \dots & \Sigma(\boldsymbol{u}_N, \boldsymbol{u}_N) & 1 & z(\boldsymbol{u}_N) \\ 1 & \dots & 1 & 0 & 0 \\ z(\boldsymbol{u}_1) & \dots & z(\boldsymbol{u}_N) & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \Sigma(\boldsymbol{u}_0, \boldsymbol{u}_1) \\ \vdots \\ \Sigma(\boldsymbol{u}_0, \boldsymbol{u}_N) \\ 1 \\ z(\boldsymbol{u}_0) \end{pmatrix} ,$$

where $\Sigma(\boldsymbol{u}_i, \boldsymbol{u}_j)$ is the spatial covariance of the rainfall y.

Note: In KED the weights λ_i sum to one. In this case we can write $\hat{y}(\boldsymbol{u}_0)$ more simply as

$$\hat{y}(\boldsymbol{u}_0) = \sum_{i=1}^{N} \lambda_i y_i \; . \qquad \square$$

The following example graphically illustrates the power of *Bayesian* curve fitting. We consider the problem of optimally fitting a straight-line *segment* to a set of two-dimensional sensor measurements $\boldsymbol{y}_i, i \in \{1, 2, \dots, N\}$. The algorithm calculates both the slope and intercept of the corresponding straight-line and the end points of the straight-line segment.

Example 9.5. Fitting a Straight-Line Segment in Two Dimensions [327]. We consider the problem of fitting a straight-line, $L \equiv L(\boldsymbol{\theta})$, to a set of points, $\boldsymbol{z}_i = (x_i, y_i), i \in \{1, 2, \dots, N\}$, where $\boldsymbol{\theta} = \big((x_\mathrm{A}, y_A), (x_\mathrm{B}, y_B)\big)$ denotes the Cartesian coordinates of the end points, A and B (Fig. 9.1). Let $\{\boldsymbol{z}_i\} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_N\}$. Then our problem reduces to calculating the *a posteriori* probability density function $p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, I)$. Using Bayes' theorem and introducing the dummy variable, $\boldsymbol{P}$, gives

$$\begin{aligned} p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, \boldsymbol{P}, I) &= \frac{p(\{\boldsymbol{z}_i\}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|I)}{p(\{\boldsymbol{z}_i\}|I)} \; , \\ &= \frac{p(\{\boldsymbol{z}_i\}|\boldsymbol{P}, I)\pi(\boldsymbol{P}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|I)}{p(\{\boldsymbol{z}_i\}|I)} \; , \end{aligned}$$

where $\boldsymbol{P} = (x, y)$ is any point on the straight-line $L(\boldsymbol{\theta})$. We treat $\boldsymbol{P}$ as a nuisance parameter which we eliminate by integration. If the $\boldsymbol{z}_i$ are independent, and assuming a uniform distribution for $\pi(\boldsymbol{\theta}|I)$, we obtain

$$p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, I) \propto \prod_{i=1}^{N} \int_{L(\boldsymbol{\theta})} p(\boldsymbol{z}_i|\boldsymbol{P}, I)\pi(\boldsymbol{P}|\boldsymbol{\theta}, I) d\boldsymbol{P} \; .$$

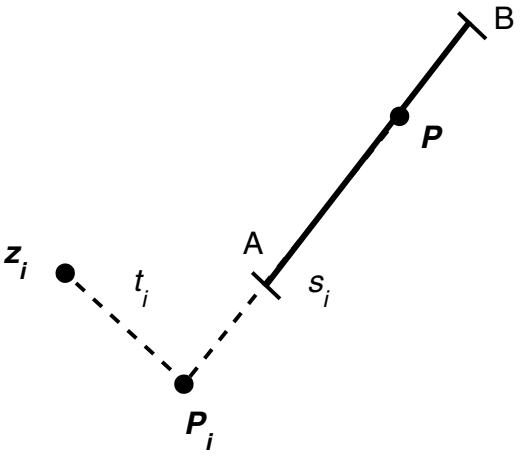

Fig. 9.1. Shows the straight-line segment AB, of finite-length, and a single data point $\boldsymbol{z}_i$ beside it. $\boldsymbol{P}_i$ denotes the closest point to $\boldsymbol{z}_i$, which lies on AB or on its extension. $\boldsymbol{P}$ denotes any point which lies on the segment AB

Assuming a zero-mean, variance σ^2, Gaussian noise model and a uniform distribution for $\pi(\boldsymbol{P}|\boldsymbol{\theta}, I)$, gives

$$p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, I) \propto \prod_{i=1}^{N} \int_{L(\boldsymbol{\theta})} \frac{1}{\sqrt{2\pi\sigma^2}} \left[\exp - \left(\frac{\boldsymbol{z}_i - \boldsymbol{P}}{\sqrt{2\sigma^2}} \right)^2 \right] d\boldsymbol{P} \, .$$

Let $\boldsymbol{P}_i$ denote the point on $L(\boldsymbol{\theta})$ which is nearest to $\boldsymbol{P}$, then $|\boldsymbol{z}_i - \boldsymbol{P}_i| = \sqrt{s_i^2 + t_i^2}$, where $s_i = |\boldsymbol{P} - \boldsymbol{P}_i|$ and $t_i = |\boldsymbol{z}_i - \boldsymbol{P}_i|$. We obtain

$$p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, I) \propto \prod_{i=1}^{N} \int_{s_A}^{s_B} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[- \left(\frac{s_i^2 + t_i^2}{2\sigma^2} \right) \right] ds_i \, ,$$

which may be solved numerically.

Special Case. An important special case occurs when we let the line extend to $\pm\infty$. In this case the *a posteriori* probability is

$$p(\boldsymbol{\theta}|\{\boldsymbol{z}_i\}, I) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{t_i^2}{2\sigma^2} \, .$$

The corresponding maximum *a posteriori* (MAP) value for θ is

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\max \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{t_i^2}{2\sigma^2} \, ,$$

or equivalently,

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\min \sum_{i=1}^{N} t_i^2 \ .$$

We see that, in this case, the maximum *a posteriori* straight-line, $L(\hat{\boldsymbol{\theta}}_{\mathrm{MAP}})$, is identical to the straight-line found using the standard least square approach (see Sect. 9.5). □

9.4 Maximum Likelihood

In some circumstances we do not assume any *a priori* information or, equivalently, we assume a uniform *a priori* distribution $\pi(\boldsymbol{\theta}|I)$. In this case, the *a posteriori* density $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$ is proportional to the likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$:

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = p(\boldsymbol{y}|\boldsymbol{\theta}, I)\pi(\boldsymbol{\theta}|I)/p(\boldsymbol{y}|I) \ , \tag{9.8}$$

$$\propto p(\boldsymbol{y}|\boldsymbol{\theta}, I) \ , \tag{9.9}$$

and the maximum *a posteriori* solution is identical to the maximum likelihood solution:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} \equiv \hat{\boldsymbol{\theta}}_{\mathrm{ML}} \ , \tag{9.10}$$

where

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{y}, I) \ , \tag{9.11}$$

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{\theta}, I) \ . \tag{9.12}$$

Assuming a differentiable function, $\hat{\boldsymbol{\theta}}_{\mathrm{ML}}$ is found from

$$\left.\frac{\partial \ln p(\boldsymbol{y}|\boldsymbol{\theta}, I)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{\mathrm{ML}}} = \mathbf{0} \ . \tag{9.13}$$

If multiple solutions exist, then the one that maximizes the likelihood function is $\hat{\boldsymbol{\theta}}_{\mathrm{ML}}$.

The maximum likelihood estimator is the most widely used estimation method and is often used in combination with other optimization/estimation procedures. For example, the expectation-maximization (EM) algorithm, relies on the maximum-likelihood estimator and for this reason the algorithm is often referred to as the ML-EM algorithm. The following example illustrates the use of the ML-EM algorithm in a speech quality estimation procedure.

Example 9.6. Nonintrusive Speech Quality Estimation Using GMM's [79]. The evaluation and assurance of speech quality is critically important for telephone service providers. A nonintrusive speech quality measurement algorithm is

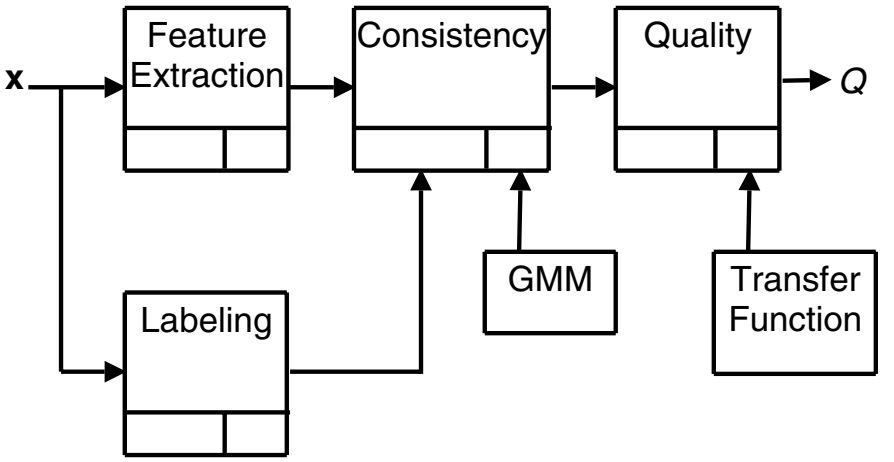

Fig. 9.2. Shows the non-intrusive speech quality estimation algorithm. The main steps in the algorithm are: (1) The segmentation, or labeling, module which labels each time frame of the speech signal as voiced (V), unvoiced (U) or inactive (I). (2) The feature extraction module which extracts perceptual features from each time frame. (3) The consistency module which calculates the consistency measure $c(\Lambda)$ and (4) The quality measure module which calculates the speech quality Q.

shown in Fig. 9.2. Perceptual features are extracted from test speech. The time segmentation (labeling) module labels the feature vectors of each time frame as belonging to one of three possible classes: voiced (V), unvoiced (U) or inactive (I). Offline, high-quality undistorted speech signals are used to produce a reference model of the behaviour of clean speech features. This is accomplished by modeling the probability distribution of the features for each class $\Lambda \in \{U, V, I\}$ with a mixture model GMM(Λ). Features extracted from the test signal which belong to a class Λ are assessed by calculating a "consistency" measure with respect to GMM(Λ). The consistency measure is defined as

$$c(\Lambda) = \frac{1}{N} \sum_{i=1}^{N} \log p(y_i|\Lambda) \ ,$$

where $p(y_i|\Lambda)$ is the likelihood function of the ith test signal, y_i, given the model GMM(Λ). The three consistency values $c(\Lambda), \Lambda \in \{U, V, I\}$ are fused together and then converted to a speech quality Q by passing through an appropriate transfer function (see Ex. 7.3). □

Although the ML procedure is widely used in many applications, a certain degree of caution should be exercised when using it. The reason is that the ML has no mechanism for regulating its solution. As a consequence, the solution may prove to be completely wrong, or as in the following example, wildly optimistic.

Example 9.7. Completion of a Doctoral Thesis: ML vs. MAP [47]. A doctoral student gives his supervisor a *precise* time, T, for the completion of his thesis. If the supervisor belongs to the "maximum likelihood" school then he does not use any *a priori* information and will uncritically accept the students estimate. If, on the other hand, the supervisor belongs to the "Bayesian" school, he will use all the *a priori* information which is available. For example, he may use his knowledge of human psychology and conjecture that a student working under pressure will almost certainly underestimate the time required to complete his PhD thesis. The result is that although the likelihood function is only high for $t \sim T$, the *a priori* probability that T is a reasonable date, given the student is working under pressure, is sufficiently low to make the estimate $t \approx T$ highly unlikely. □

9.5 Least Squares

For some applications we cannot define an *a posterior* probability density function $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$, where $\boldsymbol{y} = (\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_N^T)^T$. In this case we may optimally determine the parameters $\boldsymbol{\theta}$ by using a deterministic, i. e. *non-probabilistic*, method such as the method of least squares. For simplicity, we shall restrict ourselves to one-dimensional scalar measurements $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$, which are noisy samples of an "ideal" function $y = f(\boldsymbol{\theta})$. If the y_i are independent, then the least square estimate $\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = (\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_K)^T$, is defined as the set of parameter values which minimize the sum of the square differences $(y_i - f(\boldsymbol{\theta}))^2$. Mathematically,

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left(\frac{y_i - f(\boldsymbol{\theta})}{\sigma_i} \right)^2 , \tag{9.14}$$

where σ_i^2 is the noise variance in y_i.

If we assume $y_i - f(\boldsymbol{\theta}) \sim \mathcal{N}(0, \sigma_i^2)$, then the corresponding likelihood function is

$$p(\boldsymbol{y}|\boldsymbol{\theta}, I) = \prod_{i=1}^{N} \mathcal{N}(y_i | f(\boldsymbol{\theta}), \sigma_i^2) , \tag{9.15}$$

and the maximum likelihood solution is numerically identical to the least square solution:

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \hat{\boldsymbol{\theta}}_{\mathrm{LS}} , \tag{9.16}$$

where

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \mathcal{N}(y_i | f(\boldsymbol{\theta}), \sigma_i^2) , \tag{9.17}$$

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left(\frac{y_i - f(\boldsymbol{\theta})}{\sigma_i} \right)^2 . \tag{9.18}$$

If, additionally, we assume a uniform *a priori* distribution $\pi(\boldsymbol{\theta}|I)$ for $\boldsymbol{\theta}$, then the maximum *a posteriori* solution becomes equal to the maximum likelihood (and least square) solution.

In general, the MAP, ML and LS solutions are numerically equal, if the *a priori* distribution $\pi(\boldsymbol{\theta}|I)$ is uniform and the input measurement vector $\boldsymbol{y} = (\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_N^T)^T$ follows a multivariate Gaussian distribution $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{\Sigma})$. Mathematically,

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \hat{\boldsymbol{\theta}}_{\mathrm{LS}} \tag{9.19}$$

where

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|I)\mathcal{N}\big(\boldsymbol{y}|\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{\Sigma}\big) \,, \tag{9.20}$$

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathcal{N}\big(\boldsymbol{y}|\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{\Sigma}\big) \,, \tag{9.21}$$

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \big(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{\theta})\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{\theta})\big) \,. \tag{9.22}$$

The following example illustrates the least squares procedure by fitting a set of experimental covariance measurements with a parametric covariance model. This procedure is widely used in geostatistical applications including geographical information systems (see Sect. 4.3).

Example 9.8. Fitting Covariance Measurements to a Spatial Covariance Model [52]. In Sect. 4.3.1 we described the construction of a GIS using the Kriging estimators. The calculation of these estimators require a positive-definite spatial covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j)$. One way of ensuring this is to fit the experimentally measured $\boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j)$ values to a parametric model $\boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j|\boldsymbol{\theta})$ which has been designed to be positive definite. Traditionally, we determine the parameter vector $\boldsymbol{\theta}$ using the method of least squares in which we suppose all $\boldsymbol{y}_i$ are equally important. In this case the calculation of $\boldsymbol{\theta}$ reduces to

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \big(\boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j) - \boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j|\boldsymbol{\theta})\big)^T \big(\boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j) - \boldsymbol{\Sigma}(\boldsymbol{u}_i, \boldsymbol{u}_j|\boldsymbol{\theta})\big) \,.$$

This is known as the *ordinary* least square procedure. □

9.6 Linear Gaussian Model

The *Linear Gaussian Model* describes a linear model between two vectors $\boldsymbol{y}$ and $\boldsymbol{\theta}$ with Gaussian errors. Mathematically, the model is

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{w} \,, \tag{9.23}$$

where $\boldsymbol{y}$ is a $N \times 1$ vector of scalar measurements $(y_1, y_2, \ldots, y_N)^T$, $\boldsymbol{H}$ is a known $N \times K$ matrix, $\boldsymbol{\theta}$ is a $K \times 1$ random vector of linear coefficients

$(\theta_1, \theta_2, \ldots, \theta_K)^T$ with *a priori* pdf, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)^T$ is a $N \times 1$ noise vector with pdf, $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$, and independent of $\boldsymbol{\theta}$.

The linear Gaussian model forms the basis for much statistical modelling which is used in multi-sensor data fusion. In Table 9.4 we list some common statistical models which may be written in the form of a linear Gaussian model. The following example shows how we may re-write the polynomial model as a linear Gaussian model.

Example 9.9. Polynomial Model [253]. The polynomial model is defined as

$$y_i = \sum_{k=1}^{K} a_k u_i^{k-1} + w_i \ ,$$

or, in matrix form, as

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & u_0 & u_0^2 & \ldots & u_0^{K-1} \\ 1 & u_1 & u_1^2 & \ldots & u_1^{K-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_{N-1} & u_{N-1}^2 & \ldots & u_{N-1}^{K-1} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_K \end{pmatrix} + \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} \ .$$

By comparing the matrix equation with (9.23) we make the following identification:

$$\boldsymbol{H} = \begin{pmatrix} 1 & u_0 & u_0^2 & \ldots & u_0^{K-1} \\ 1 & u_1 & u_1^2 & \ldots & u_1^{K-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_{N-1} & u_{N-1}^2 & \ldots & u_{N-1}^{K-1} \end{pmatrix} \ ,$$

$$\boldsymbol{\theta} = (a_1, a_2, \ldots, a_N)^T \ . \qquad \square$$

By Bayes' theorem, the *a posteriori* pdf of the linear Gaussian model parameters is

$$p(\boldsymbol{H}, \sigma, \boldsymbol{\theta} | \boldsymbol{y}, I) = \frac{p(\boldsymbol{y} | \boldsymbol{H}, \boldsymbol{\theta}, \sigma, I) \pi(\boldsymbol{H}, \boldsymbol{\theta}, \sigma | I)}{p(\boldsymbol{y} | I)} \ , \tag{9.24}$$

Table 9.4. Representations which use the Linear Gaussian Model

Name	Equivalent Linear Gaussian Model
Sinusoidal	$y_i = \sum_{k=1}^{K} [a_k \sin(\omega_k t_i) + b_k \cos(\omega_k t_i)] + w_i.$
Autoregressive (AR)	$y_i = \sum_{k=1}^{K} a_k y_{i-k} + w_i.$
Autoregressive with External Input (ARX)	$y_i = \sum_{k=1}^{K} a_k y_{i-k} + \sum_{l=0}^{L} b_k u_{i-l} + w_i.$
Nonlinear Autoregressive (NAR)	$y_i = \sum_{k=1}^{K} a_k y_{i-k} + \sum_{l=1}^{L} \sum_{m=1}^{M} b_{lm} y_{i-l} y_{i-m} + \ldots + w_i.$
Volterra	$y_i = \sum_{k=1}^{K} a_k y_{i-k} + \sum_{l=1}^{L} \sum_{m=1}^{M} b_{lm} y_{i-l} y_{i-m} + \ldots + w_i.$
Polynomial	$y_i = \sum_{k=1}^{K} a_k u_i^{k-1}.$

where

$$p(\boldsymbol{y}|\boldsymbol{H},\sigma,\boldsymbol{\theta},I) = (2\pi\sigma^2)^{-N/2}\exp -\frac{(\boldsymbol{y}-\boldsymbol{H\theta})(\boldsymbol{y}-\boldsymbol{H\theta})^T}{\sigma^2}\ . \tag{9.25}$$

Assuming that $\boldsymbol{H}$, $\boldsymbol{\theta}$ and σ are statistically independent, we have

$$p(\boldsymbol{H},\sigma,\boldsymbol{\theta}|\boldsymbol{y},I) = \frac{p(\boldsymbol{y}|\boldsymbol{H},\boldsymbol{\theta},\sigma,I)\pi(\boldsymbol{H}|I)\pi(\boldsymbol{\theta}|I)\pi(\sigma|I)}{p(\boldsymbol{y}|I)}\ , \tag{9.26}$$

In the case where there is no *a priori* information concerning $\boldsymbol{H}$, $\boldsymbol{\theta}$ and σ we may use uniform priors for the elements of $\boldsymbol{H}$ and $\boldsymbol{\theta}$ and Jeffrey's prior for σ. In this case, the joint *a posteriori* pdf for $\boldsymbol{H}$, $\boldsymbol{\theta}$ and σ becomes

$$\begin{aligned} p(\boldsymbol{H},\boldsymbol{\theta},\sigma|\boldsymbol{y},I) &\propto \frac{p(\boldsymbol{y}|\boldsymbol{H},\boldsymbol{\theta},\sigma,I)}{p(\boldsymbol{y}|I)} \times \frac{1}{\sigma}\ , \\ &\propto \frac{1}{\sigma^{N+1}}\exp -\left(\frac{(\boldsymbol{y}-\boldsymbol{H\theta})^T(\boldsymbol{y}-\boldsymbol{H\theta})}{2\sigma^2}\right)\ . \end{aligned} \tag{9.27}$$

Until now we assumed the noise components w_i are iid, where $w_i \sim \mathcal{N}(0,\sigma^2)$. However, in some applications the noise components w_i are not independent. In this case $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$, where the off-diagonal components of $\boldsymbol{\Sigma}$ are not zero. However, we may always convert the measurements so that $w_i \sim \mathcal{N}(0,\sigma^2)$ by applying a *whitening transformation* $\boldsymbol{D}$.

Example 9.10. Whitening Transformation [148]. We assume $\boldsymbol{\Sigma}$ is positive definite. In this case it can be factored as $\boldsymbol{\Sigma}^{-1} = \boldsymbol{D}^T\boldsymbol{D}$, where $\boldsymbol{D}$ is an $N \times N$ invertible matrix. The matrix $\boldsymbol{D}$ acts as a *whitening* transformation when applied to $\boldsymbol{w}$, since

$$E\big(\boldsymbol{Dw}(\boldsymbol{Dw})^T\big) = \boldsymbol{D\Sigma D}^T = \boldsymbol{DD}^{-1}(\boldsymbol{D}^T)^{-1}\boldsymbol{D}^T = \boldsymbol{I}\ ,$$

where $\boldsymbol{I} = \operatorname{diag}(\overbrace{1,1,\ldots,1}^{N})$.

As a consequence, if we apply $\boldsymbol{D}$ to $\boldsymbol{y} = \boldsymbol{H\theta} + \boldsymbol{w}$ we obtain a linear Gaussian model:

$$\boldsymbol{y}' = \boldsymbol{Dy} = \boldsymbol{DH\theta} + \boldsymbol{Dw} = \boldsymbol{H}'\boldsymbol{\theta} + \boldsymbol{w}'\ ,$$

where $\boldsymbol{w}' = \boldsymbol{Dw} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. □

The linear Gaussian model and the *a posteriori* pdf $p(\boldsymbol{H},\boldsymbol{\theta},\sigma|\boldsymbol{y},I)$ are widely used in multi-sensor data fusion. By way of an example, we give three different types of application.

9.6.1 Line Fitting

In curve fitting the matrix $\boldsymbol{H}$ is completely specified. In this case, our objective is to estimate the vector $\boldsymbol{\theta}$ assuming $\boldsymbol{H}$ is given but having no *a priori*

knowledge regarding $\boldsymbol{\theta}$ or σ. In this case $\boldsymbol{H}$ does not appear in (9.27). We eliminate σ by marginalization. Then the resulting *a posteriori* pdf is

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \frac{1}{2}\Gamma\big(\frac{N-1}{2}\big)[(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{\theta})^T(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{\theta})/2]^{-\frac{N-1}{2}} \ . \tag{9.28}$$

Maximizing this expression gives us the MAP estimate for the parameter vector $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{\mathrm{MAP}} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y} \ . \tag{9.29}$$

The following example illustrates the fitting of a straight-line to a set of points $(u_i, y_i), i \in \{1, 2, \ldots, N\}$.

Example 9.11. Straight-Line Regression. We consider a set of points $(u_i, y_i), i \in \{1, 2, \ldots, N\}$ which we assume to obey the following straight-line equation:

$$y_i = mu_i + c + w_i \ ,$$

where $w_i \sim \mathcal{N}(0, \sigma^2)$ are independent random noise. The corresponding matrix equation is

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & u_1 \\ 1 & u_2 \\ \vdots & \vdots \\ 1 & u_N \end{pmatrix} \begin{pmatrix} c \\ m \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}$$

It follows that

$$\begin{aligned} \boldsymbol{H}^T\boldsymbol{H} &= \begin{pmatrix} N & \sum_{i=1}^N u_i \\ \sum_{i=1}^N u_i & \sum_{i=1}^N u_i^2 \end{pmatrix} \ , \\ (\boldsymbol{H}^T\boldsymbol{H})^{-1} &= \frac{1}{N\sum_{i=1}^N u_i^2 - (\sum_{i=1}^N u_i)^2} \begin{pmatrix} \sum_{i=1}^N u_i^2 & -\sum_{i=1}^N u_i \\ -\sum_{i=1}^N u_i & N \end{pmatrix} \ , \\ \boldsymbol{H}^T\boldsymbol{y} &= \begin{pmatrix} \sum_{i=1}^N y_i \\ \sum_{i=1}^N u_i y_i \end{pmatrix} \ . \end{aligned}$$

Substituting all these terms in (9.29) gives the following expression for the map estimate of $(c, m)^T$:

$$\begin{pmatrix} c \\ m \end{pmatrix} = \frac{1}{\sum_{i=1}^N u_i^2 - N\bar{u}^2} \begin{pmatrix} \bar{y}\sum_{i=1}^N u_i^2 - \bar{u}\sum_{i=1}^N u_i y_i \\ \sum_{i=1}^N u_i y_i - \bar{u}\bar{y} \end{pmatrix} \ ,$$

where $\bar{u} = \sum_{i=1}^N u_i/N$ and $\bar{y} = \sum_{i=1}^N y_i/N$. Not unexpectedly [2] this solution is identical to the solution obtained using the method of maximum likelihood or least squares. □

[2] This result is not unexpected since we assumed non-informative priors for $\boldsymbol{\theta}$ and σ.

9.6.2 Change Point Detection

The central concept which underlies a linear change point detector is that an observed time sequence is modelled by different models at different points in time. The models are known but the time instants at which the models change are not known. In this case, our objective is to estimate $\boldsymbol{H}$. We eliminate $\boldsymbol{\theta}$ and σ from (9.27) by marginalization [271]. The resulting *a posteriori* pdf is

$$p(\boldsymbol{H}|\boldsymbol{y}, I) \propto \frac{[\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}]^{\frac{N-K}{2}}}{\sqrt{|\boldsymbol{H}^T\boldsymbol{H}|}} \,. \tag{9.30}$$

The following example illustrates (9.30) given a set of scalar input measurements, $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$, assuming there is a single (unknown) change point.

Example 9.12. Single Step in Piecewise Constant Time Series [84, 271]. We consider a time series y_i:

$$y_i = \begin{cases} \mu_1 + w_i \text{ if } i \leq m \,, \\ \mu_2 + w_i \text{ if } i > m \,, \end{cases}$$

where m is an unknown integer ($1 \leq m \leq N$) and $w_i \sim \mathcal{N}(0, \sigma^2)$ are independent random noise. The corresponding matrix equation is

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \\ y_{m+1} \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \\ w_{m+1} \\ \vdots \\ w_N \end{pmatrix} \,.$$

It follows that

$$\begin{aligned} \boldsymbol{H}^T\boldsymbol{H} &= \begin{pmatrix} m & 0 \\ 0 & N-m \end{pmatrix} \,, \\ (\boldsymbol{H}^T\boldsymbol{H})^{-1} &= \frac{1}{m(N-m)} \begin{pmatrix} N-m & 0 \\ 0 & m \end{pmatrix} \,, \\ \boldsymbol{y}^T\boldsymbol{H} &= (\boldsymbol{H}^T\boldsymbol{y})^T = \Big(\sum_{i=1}^{m} y_i \quad \sum_{i=m+1}^{N} y_i\Big) \,, \\ \boldsymbol{y}^T\boldsymbol{y} &= \sum_{i=1}^{N} y_i^2 \,. \end{aligned}$$

Substituting all these terms in (9.30) gives the following expression for the *a posteriori* pdf $p(m|\boldsymbol{y}, I)$:

$$p(m|\boldsymbol{y}, I) \sim \sum_{i=1}^{N} y_i^2 - (N-m)\Big(\sum_{i=1}^{m} y_i\Big)^2 - m\Big(\sum_{i=m+1}^{N} y_i\Big)^2 .$$

The corresponding MAP estimate of m is

$$m_{\mathrm{MAP}} = \arg\max_m p(m|\boldsymbol{y}, I) ,$$

and the associated μ_1 and μ_2 values are:

$$\mu_1 = \frac{1}{m_{\mathrm{MAP}}} \sum_{i=1}^{m_{\mathrm{MAP}}} y_i ,$$

$$\mu_2 = \frac{1}{N - m_{\mathrm{MAP}} + 1} \sum_{i=m_{\mathrm{MAP}}+1}^{N} y_i . \qquad \square$$

Single change point detection with more complicated curves are also easily handled. In the following two examples we derive the matrix $\boldsymbol{H}$ for a single ramp and a mixture of different order AR processes.

Example 9.13. Single Ramp in a Time Series [84, 271]. We consider a time series y_i:

$$y_i = \begin{cases} \mu + w_i & \text{if } i \le m , \\ \mu + r(i-m) + w_i & \text{if } i > m , \end{cases}$$

where $w_i \sim \mathcal{N}(0, \sigma^2)$ are independent random noise. The corresponding matrix equation is

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \\ y_{m+1} \\ y_{m+2} \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & N-m \end{pmatrix} \begin{pmatrix} \mu \\ r \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \\ w_{m+1} \\ w_{m+2} \\ \vdots \\ w_N \end{pmatrix} \qquad \square$$

Example 9.14. Mixed Two and Three-Order AR Process [84, 271]. We consider a time series y_i:

$$y_i = \begin{cases} \sum_{j=1}^{2} \alpha_j y_{i-j} + w_i & \text{if } i \le m , \\ \sum_{j=1}^{3} \beta_j y_{i-j} + w_i & \text{if } i > m , \end{cases}$$

where $w_i \sim \mathcal{N}(0, \sigma^2)$ are independent random noise variables. The corresponding matrix equation is

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{m-1} \\ y_m \\ y_{m+1} \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} y_0 & y_{-1} & 0 & 0 & 0 \\ y_1 & y_0 & 0 & 0 & 0 \\ y_2 & y_1 & 0 & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ y_{m-2} & y_{m-3} & 0 & 0 & 0 \\ y_{m-1} & y_{m-2} & 0 & 0 & 0 \\ 0 & 0 & y_m & y_{m-1} & y_{m-2} \\ \vdots & & & & \\ 0 & 0 & y_{N-1} & y_{N-2} & y_{N-3} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{m-1} \\ w_m \\ w_{m+1} \\ \vdots \\ w_N \end{pmatrix} \qquad \square$$

In the most general case the number of change points is unknown. In each segment the y_i are modeled as independent polynomial functions of unknown order. Efficient algorithms for solving the more general case have been developed, including an exact curve fitting algorithm [80].

9.6.3 Probabilistic Subspace

Subspace techniques are a family of techniques which may be used to generate a low dimensional common representational format (see Sect. 4.5). Recently probabilistic versions of several subspace techniques have been described. In this section we concentrate on the *probabilistic principal component analysis* (PPCA) [300, 301] which is a probabilistic version of the principal component analysis (PCA) algorithm (see Ex. 4.11).

In the probabilistic PCA we model the sensor measurements $\boldsymbol{y}_i$ using a linear Gaussian model of the form

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\mu} + \boldsymbol{w} \,, \tag{9.31}$$

where we assume

$$p(\boldsymbol{\theta}|I) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \,, \tag{9.32}$$

$$p(\boldsymbol{w}|I) \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}) \,. \tag{9.33}$$

Under these assumptions, we find

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{H}, \boldsymbol{\mu}, \sigma^2, I) &= \int p(\boldsymbol{y}|\boldsymbol{H}, \boldsymbol{\theta}, \boldsymbol{\mu}, \sigma^2, I) p(\boldsymbol{\theta}|I) d\boldsymbol{\theta} \,, \\ &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{H}\boldsymbol{H}^T + \sigma^2\boldsymbol{I}) \end{aligned} \tag{9.34}$$

where

$$p(\boldsymbol{y}|\boldsymbol{H}, \boldsymbol{\theta}, \boldsymbol{\mu}, \sigma^2, I) \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I}) \,. \tag{9.35}$$

We use (9.32-9.35) in Bayes' rule to give us the *a posteriori* pdf $p(\boldsymbol{\theta}|\boldsymbol{y}, I)$:

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{H}, \boldsymbol{\mu}, \sigma^2, \boldsymbol{y}, I) &= \frac{p(\boldsymbol{y}|\boldsymbol{H}, \boldsymbol{\theta}, \boldsymbol{\mu}, \sigma^2, I) p(\boldsymbol{\theta}|I)}{p(\boldsymbol{y}|\boldsymbol{H}, \boldsymbol{\mu}, \sigma^2, I)} , \\ &= \frac{\mathcal{N}(\boldsymbol{y}|\boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\mu}, \sigma^2 I) \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{0}, \boldsymbol{I})}{\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{H}\boldsymbol{H}^T + \sigma^2 \boldsymbol{I})} , \\ &= \mathcal{N}(\boldsymbol{H}^T(\boldsymbol{y} - \boldsymbol{\mu})/\boldsymbol{\beta}, \sigma^2/\boldsymbol{\beta}) , \end{aligned} \tag{9.36}$$

where

$$\boldsymbol{\beta} = \boldsymbol{H}^T\boldsymbol{H} + \sigma^2 \boldsymbol{I} . \tag{9.37}$$

Maximum-likelihood estimates of the unknown parameters in (9.36) are given by

$$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i , \tag{9.38}$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{M - L} \sum_{i=L+1}^{M} \lambda_i , \tag{9.39}$$

$$\boldsymbol{H}_{\mathrm{ML}} = \boldsymbol{U}(\boldsymbol{\Lambda} - \sigma^2_{\mathrm{ML}} \boldsymbol{I})^{1/2} \boldsymbol{R} , \tag{9.40}$$

where $\lambda_{L+1}, \ldots, \lambda_M$ are the smallest eigenvalues of the sample covariance matrix $\boldsymbol{\Sigma}$, the L columns in the $M \times L$ orthogonal matrix $\boldsymbol{U}$ are the dominant L eigenvectors of $\boldsymbol{\Sigma}$, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_L)$ and $\boldsymbol{R}$ is an arbitrary $L \times L$ orthogonal matrix [3].

Apart from using the PPCA to create a low-dimensional probabilistic common representational format (Sect. 4.5), we can also model arbitrary pdf's using a mixture of PPCA's instead of a mixture of Gaussians. The main advantage is that by using PPCA's we can help reduce the impact of the "curse of dimensionality" [4] (see Ex. 8.5). The following example compares a mixture of PPCA's with the GMM.

Example 9.15. PPCA vs. GMM [301]. A partially open circle in (x, y)-space is sampled after adding iid (independent and identically distributed) Gaussian noise. In this case $M = 2$. We model the data using a mixture of full covariance, and diagonal covariance, Gaussian pdf's and a mixture of one component ($L = 1$) PPCA's. Fig. 9.3 shows the results obtained. □

[3] The presence of the matrix $\boldsymbol{R}$ in (9.40) means that $\boldsymbol{H}_{\mathrm{ML}}$ is rotationally ambiguous. We may eliminate the ambiguity and obtain the true principal axes by multiplying $\boldsymbol{H}_{\mathrm{ML}}$ by the rotation matrix $\boldsymbol{R}_{\mathrm{ML}}$, where the columns of $\boldsymbol{R}_{\mathrm{ML}}$ are the eigenvectors of $\boldsymbol{H}^T_{\mathrm{ML}}\boldsymbol{H}_{\mathrm{ML}}$.

[4] The number of free parameters in a mixture of full covariance M-dimensional Gaussian pdf's is $M(M+3)/2$ parameters per pdf. In a mixture of diagonal covariance M-dimensional Gaussian pdf's the number of free parameters is $2M$ per pdf and in a mixture of L component PPCA's the number of free parameters is $M(M+1)+1+L(L-1)/2$ per PPCA. *Note*: The term $L(L-1)/2$ arises from the need to specify the rotation matrix $\boldsymbol{R}$.

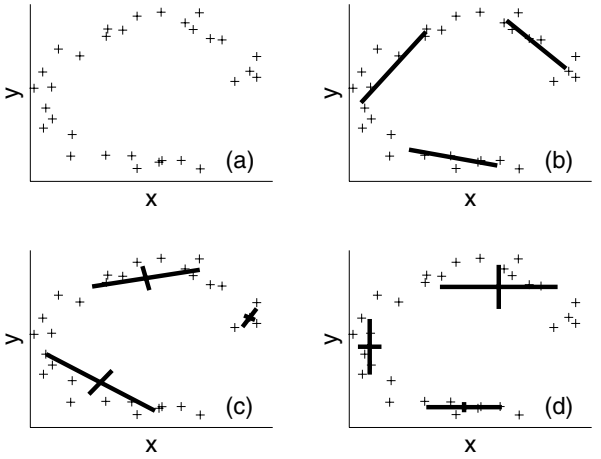

Fig. 9.3. Shows the modeling of a noisy partially open circle in two-dimensions. **(a)** Shows the data points in (x, y) space. **(b)** Shows the three one-component PPCA's we use to model the points. **(c)** Shows the three full-covariance Gaussian mixture we use to model the points. **(d)** Shows the three diagonal covariance Gaussian mixture we use to model the points.

The following example illustrates an unusual application which uses a mixture of PPCA's instead of the more usual mixture of Gaussians.

Example 9.16. Voice Conversion Using a Mixture of PPCA's [328]. Voice conversion is the art of transforming one speaker's voice to sound as if it were uttered by another speaker. One application of voice conversion deals with the dubbing of a "original" film sound track. When a popular movie is exported to a foreign country, the sound track is often re-recorded by replacement speakers who speak a translated version of the original performer's part. By using voice conversion we obtain a sound track spoken in a foreign language by a speaker whose voice matches that of the original performer. For this application the authors found it advantageous to use a mixture of PPCA's instead of the more conventional mixture of Gaussians. □

9.7 Generalized Millman Formula

The *generalized Millman's formula* [277] deals with the optimal linear combination of an arbitrary number of correlated estimates. Suppose we have K local estimates, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$, of an unknown L-dimensional random vector $\boldsymbol{\theta}$. Then the optimal *linear* estimate of $\boldsymbol{\theta}$ is

$$\widetilde{\boldsymbol{\mu}} = \sum_{k=1}^{K} \boldsymbol{W}_k \boldsymbol{\mu}_k \ , \tag{9.41}$$

where

$$\sum_{k=1}^{K} \boldsymbol{W}_k = \mathrm{diag}(\overbrace{1, 1, \ldots, 1}^{L}) \,, \tag{9.42}$$

and the $\boldsymbol{W}_k$ are $L \times L$ constant weighting matrices determined from the least square error criterion

$$(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_K) = \min_{\boldsymbol{W}_k} \Big(|\boldsymbol{\theta} - \sum_{k=1}^{K} \boldsymbol{W}_k \boldsymbol{\mu}_k|^2 \Big) \,. \tag{9.43}$$

The solution to this problem is given by the following set of $(K-1)$ linear equations

$$\begin{aligned}
&\sum_{k=1}^{K-1} \boldsymbol{W}_k (\boldsymbol{\Sigma}_{k,1} - \boldsymbol{\Sigma}_{k,K}) + \boldsymbol{W}_K (\boldsymbol{\Sigma}_{K,1} - \boldsymbol{\Sigma}_{K,K}) = 0 \,, \\
&\sum_{k=1}^{K-1} \boldsymbol{W}_k (\boldsymbol{\Sigma}_{k,2} - \boldsymbol{\Sigma}_{k,K}) + \boldsymbol{W}_K (\boldsymbol{\Sigma}_{K,2} - \boldsymbol{\Sigma}_{K,K}) = 0 \,, \\
&\qquad\qquad \vdots \\
&\sum_{k=1}^{K-1} \boldsymbol{W}_k (\boldsymbol{\Sigma}_{k,K-1} - \boldsymbol{\Sigma}_{k,K}) + \boldsymbol{W}_K (\boldsymbol{\Sigma}_{K,K-1} - \boldsymbol{\Sigma}_{K,K}) = 0 \,,
\end{aligned} \tag{9.44}$$

subject to (9.41), where $\boldsymbol{\Sigma}_{k,k} \equiv \boldsymbol{\Sigma}_k$ is the $L \times L$ covariance matrix $\boldsymbol{\Sigma}_{k,k} = E\big((\boldsymbol{\theta} - \boldsymbol{\mu}_k)(\boldsymbol{\theta} - \boldsymbol{\mu}_k)^T\big)$ and $\boldsymbol{\Sigma}_{k,l}, k \neq l$ is the $L \times L$ cross-covariance matrix $\boldsymbol{\Sigma}_{k,l} = E\big((\boldsymbol{\theta} - \boldsymbol{\mu}_k)(\boldsymbol{\theta} - \boldsymbol{\mu}_l)^T\big)$. In many applications, only the covariance matrices, $\boldsymbol{\Sigma}_{k,k}, k \in \{1, 2, \ldots, K\}$, are known, while the cross-covariances, $\boldsymbol{\Sigma}_{k,l}, k \neq l$, are unknown. In this case, we often assume $\boldsymbol{\Sigma}_{k,l} = 0, k \neq l$. This may, however, cause problems, especially in decentralized systems (see Sect. 3.4.2).

Example 9.17. Millman Formula [98]. Given two *uncorrelated* estimates $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, the generalized Millman formula reduces to

$$\widetilde{\boldsymbol{\mu}} = \boldsymbol{\Sigma}_{22}(\boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{22})^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{11}(\boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{22})^{-1} \boldsymbol{\mu}_2 \,,$$

where $\widetilde{\boldsymbol{\mu}}$ is the optimum linear estimate. □

Example 9.18. Bar-Shalom-Campo Formula [12]. Given two *correlated* estimates $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, the generalized Millman formula reduces to the Bar-Shalom-Campo formula:

$$\begin{aligned}
\widetilde{\boldsymbol{\mu}} = &(\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21})(\boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12} - \boldsymbol{\Sigma}_{21})^{-1} \boldsymbol{\mu}_1 \\
&+ (\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12})(\boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12} - \boldsymbol{\Sigma}_{21})^{-1} \boldsymbol{\mu}_2 \,,
\end{aligned}$$

where $\widetilde{\boldsymbol{\mu}}$ is the optimum linear estimate. □

9.8 Software

MATLAB STATISTICS TOOLBOX. The matlab statistics toolbox. The toolbox contains various m-files for performing parameter eestimation.

MILES (Maximum Likelihood via Iterative Least Squares Estimation). A matlab toolbox for fitting different maximum likelihood models using least squares algorithms.

NETLAB. A matlab toolbox for performing neural network pattern recognition.

STPRTOOL (Statistical Pattern Recognition Toolbox). A matlab toolbox for performing statistical pattern recognition.

9.9 Further Reading

Bayesian parameter estimation is discussed in many books and monographs. Good accounts are to be found in [3, 4, 64, 99, 148, 189, 190, 280]. The linear Gaussian models are discussed at length in [268, 270, 333]. A full description of Bayesian curve fitting is given in [61].

10

Robust Statistics

10.1 Introduction

In this chapter we shall consider the subject of *robust statistics* and, in particular, *robust parameter estimation*. Robust statistics is defined as the study of statistical methods which are relatively insensitive to the presence of outliers, i. e. input data which is "strange" or "incompatible" with the remaining input data. It might be thought that outliers are a rare event. This is not, however, true. In fact, a common experience in all scientific experiments is that repeated measurements of supposedly one and the same quantity result, occasionally, in values which are in striking disagreement with all the others.

Increasingly, the need to use robust statistics in multi-sensor data fusion is being recognized. The reason for this is the devastating effect outliers may have on standard statistical procedures as illustrated in the following example.

Example 10.1. Fitting a Straight-Line in the Presence of Outliers. We reconsider Ex. 9.5 except this time we suppose that in addition to the N inlier points $(x_i, y_i), i \in \{1, 2, \ldots, N\}$, we have a single outlier (x_{N+1}, y_{N+1}). If the outlier lies sufficiently far from the inliers, then by inspection the optimal straight-line (i. e. the line with the minimum sum of square deviations) will pass through the center of mass $(\bar{x}, \bar{y})$ and through the outlier (x_{N+1}, y_{N+1}) (line BB' in Fig. 10.1). This is in spite of the fact that all the inliers lie on the line AA'. □

In many applications robust parameter estimation represents the major fusion algorithm in a multi-sensor data fusion system. In Table 10.1 we list some of these applications together with the classification of the type of fusion algorithm involved.

Apart from the applications listed in Table 10.1, robust parameter estimation techniques are widely used in forming a common representational format (Chapts. 4-7).

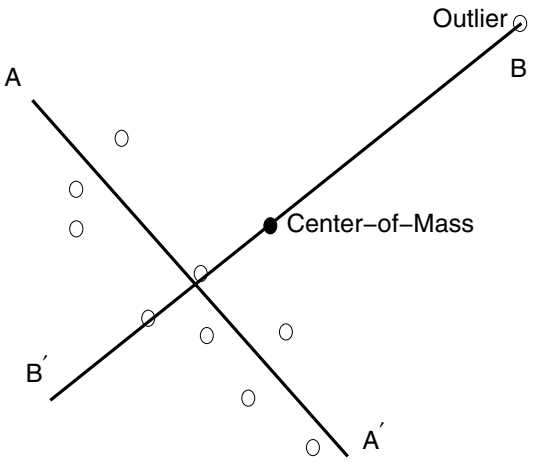

Fig. 10.1. Shows a set of points (including an outlier) in two-dimensions. Without the outlier the optimal straight-line is AA' while with the outlier the optimal straight-line is BB'.

Table 10.1. Applications in which Robust Parameter Estimation is the Primary Fusion Algorithm

Class		Application
DaI-DaO	Ex. 2.8	Adaptive Smoothing in a Smart Sensor
	Ex. 9.1	Color Constancy and the Minimum Local Mass (MLM) Loss Function
	Ex. 10.2	Gating Sensor Observations in Target Tracking
	Ex. 10.3	Calibration of the SI Unit of the Volt
	Ex. 10.6	Neutron Fission Cross Sections
	Ex. 10.7	Estimating Hubble's Constant Using a "Good-and- Bad" Likelihood Function
DaI-FeO	Ex. 10.4	Student Mixture Model
	Ex. 10.8	Background Estimation in a Powder Diffraction Pattern
	Ex. 4.13	Chemical Sensor analysis: Principal Discriminant Analysis (PDA) for Small-Sample Problems
	Ex. 10.9	Robust Discriminant Analysis
	Ex. 10.5	Robust Probabilistic Principal Component Analysis
	Ex. 10.10	Target Tracking Initialization Using a Hough Transform

The designations DaI-DaO and DaI-FeO refer, respectively, to the Dasarathy input/output classifications: "Data Input-Data Output" and "Data Input-Feature Output" (see Sect. 1.3.3).

10.2 Outliers

The simplest approach to dealing with outliers is to eliminate them before parameter estimation takes place. One area in which outliers are routinely eliminated before any parameter estimation takes place is in target tracking when it is known as *gating* and is illustrated in the following example.

Example 10.2. Gating Sensor Observations in Target Tracking [1] [13]. Recursive filters (see Chapt. 11) are routinely used to track a moving target. A necessary part of a tracking filter is to eliminate spurious sensor observations which are not associated with the target. In gating this is done by drawing a validation volume, or *gate*, around the predicted sensor observation (Fig. 10.2). The gate is usually formed in such a way that the probability that a true observation (i. e. an observation which is associated with the target) falls within the gate (assuming it is detected) is given by a gating probability P_G [24, 13].

Let $\boldsymbol{\theta}_k$ denote the state of the target at time step k and let $\boldsymbol{y}_k$ denote the corresponding sensor measurement, where

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{\theta}_k + \boldsymbol{w}_k \ ,$$

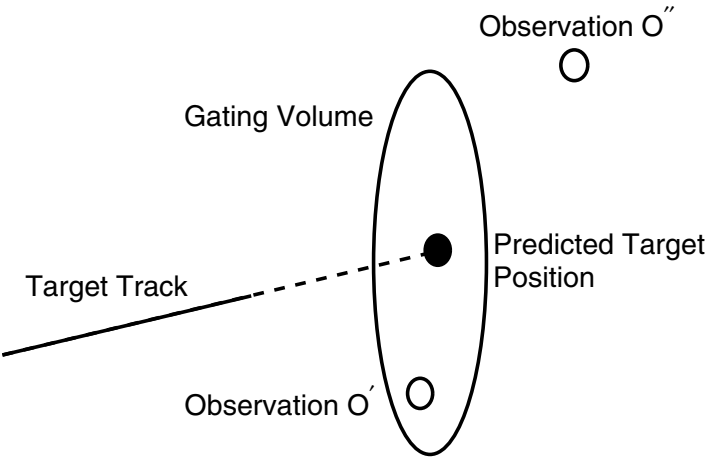

Fig. 10.2. Shows the gating of two sensor observations $O' = \langle E',\ *,\ k,\ \boldsymbol{y}',\ * \rangle$ and $O'' = \langle E'',\ *,\ k,\ \boldsymbol{y}'',\ * \rangle$ made at time step k. Observation O' falls outside the gate while O'' falls inside the gate. The gate is shown as an ellipse centered on the expected (predicted) sensor observation $\hat{\boldsymbol{y}}$ for the time step k. Note: For an explanation of the notation used in representing O' and O'', see Sect. 2.5.

[1] This example contains advanced material. It assumes the reader has some familiarity with sequential Bayesian estimation (see Chapt. 11). The example is not, however, essential for what follows and it may be left for a second reading.

and $\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$. If a Kalman filter is used to track the target, then the predicted state of the target at time step k is

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I) = \mathcal{N}(\boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1}) \ ,$$

where $\boldsymbol{y}_{1:k-1} = (\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_{k-1}^T)^T$ is the sequence of measurements made upto, and including, the time step $k-1$. In this case, the gating region G is defined as the ellipsoid

$$G = \{\boldsymbol{y}|(\boldsymbol{y} - \hat{\boldsymbol{y}}_{k|k-1})^T \hat{\boldsymbol{S}}_{k|k-1}^{-1}(\boldsymbol{y} - \hat{\boldsymbol{y}}_{k|k-1}) \leq \gamma\} \ ,$$

where $\hat{\boldsymbol{y}}_{k|k-1} = \boldsymbol{H}_k \boldsymbol{\mu}_{k|k-1}$ is the predicted sensor measurement and $\hat{\boldsymbol{S}}_{k|k-1} = \boldsymbol{H}_k \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k$ is the predicted sensor covariance matrix. The threshold γ maybe chosen to give a specified gating probability P_G. Alternatively, we may choose a threshold such that a true measurement falling within the gate is more likely than a false alarm. In two-dimensions the corresponding threshold is given by

$$\gamma = 2\ln(P_D) - 2\ln(2\pi\lambda(1 - P_D)|\hat{\boldsymbol{S}}_{k|k-1}|^{1/2}) \ ,$$

where P_D is the probability of detection of the target and λ is the spatial density of false alarms. □

Eliminating spurious observations may, however, cause unforseen problems as the following example shows.

Example 10.3. Calibration of the SI Unit of the Volt [65]. In 1973 and again in 1986 a committee of the International Council of Scientific Unions performed a least squares adjustment of several independent measurements concerning K_V (a constant involved in the calibration of the SI unit of the Volt [357]). In performing the least square adjustment the committee eliminated measurements which were deemed incompatible with the remaining measurements. A large change was found between $K_V(1973)$ and $K_V(1986)$ although as the committee explain, the large change in K_V could have been avoided if two measurements which seemed to be discrepant with the remaining measurements had *not* been deleted in the 1973 adjustment. □

In order to avoid problems of this sort we often prefer to use "robust" techniques which reduce the influence of the outliers without actually identifying the outliers and eliminating them.

10.3 Robust Parameter Estimation

The aim of robust parameter estimation techniques is to reduce the influence of the outliers without actually identifying and eliminating the outliers. The basic idea is to use a likelihood function, $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$, whose tail size varies in

Table 10.2. Robust Likelihood Functions

Name	Formula
Student-t Function	$\Gamma(\frac{\nu+1}{2})/(\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})\sigma) \times (1 + ((y-\mu)/\sigma\sqrt{\nu})^2)^{-(\nu+1)/2}$. The parameter ν controls the thickness of the tails, the thickness decreasing with an increase in ν.
"Good-and-Bad" Function	$\alpha\mathcal{N}(y\|\mu,\sigma^2) + (1-\alpha)\mathcal{N}(y\|\mu,\beta^2\sigma^2)$, where $\mathcal{N}(y\|\mu,\sigma^2)$ is the "Good", or "normal", likelihood function which is used to represent the inliers and $\mathcal{N}(y\|\mu,\beta^2\sigma^2)$ is the "Bad", or "aberrant", likelihood function which is used to represent the outliers, where $\beta >> 1$. The parameter α represents the expected proportion of inliers in the input data.
Gaussian + Step Function	$\mathcal{N}(y\|\mu,\sigma^2) + F$, where F is a constant which ensures the probability of an outlier never falls too low.
Uncertain Error Bar	$\text{erf}(\|\mu - y\|/\sqrt{2a^2})/\|\mu - y\|$.

proportion to the number of outliers which are present. Our aim is that as we move away from $\hat{\boldsymbol{\theta}}$, the mean value of $\boldsymbol{\theta}$, the tail will decrease but not too quickly nor too slowly [2].

In Table 10.2 we list some standard functions which may be used to model a robust one-dimensional likelihood function.

10.3.1 Student-t Function

The multivariate Student-t likelihood function is

$$p(\boldsymbol{\theta}|\boldsymbol{y}, I) = \mathcal{S}t(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) \;, \tag{10.1}$$

where the Student-t function, $\mathcal{S}t(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu)$, is a unimodal function with adjustable tails. In one-dimension it is defined as

$$\mathcal{S}t(y|\mu,\sigma^2,\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi} \times \Gamma(\frac{\nu}{2})\sigma}\Big(1 + \frac{(y-\mu)^2}{\nu\sigma^2}\Big)^{-(\nu+1)/2} \;, \tag{10.2}$$

where the parameter, ν, controls the thickness of the tails, the thickness decreasing with an increase in ν. When $\nu > 30$, there is virtually no difference between $\mathcal{S}t(y|\mu,\sigma,\nu)$ and $\mathcal{N}(y|\mu,\sigma^2)$.

The following example illustrates the use of K Student-t likelihood functions in a *robust* finite mixture model.

Example 10.4. Student Mixture Model (SMM) [8]. We reconsider Ex. 8.5. However this time we use a robust mixture of K Student-t functions, instead of

[2] *Note*: In this context, the Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, is *not* robust since its likelihood drops off very quickly as we move more than a few standard deviations away from $\boldsymbol{\mu}$.

a mixture of K one-dimensional Gaussian functions. Our aim is to model a one-dimensional function $\theta = f(y)$:

$$f(y) \simeq \sum_{k=1}^{K} \alpha_k \mathcal{S}t(y|\mu_k, \sigma_k^2, \nu_k) \ ,$$

where $\alpha_k, \mu_k, \sigma_k, \nu_k, k \in \{1, 2, \ldots, K\}$, are unknown and are to be learnt from a set of training data $(y_i, \theta_i), i \in \{1, 2, \ldots, N\}$, subject to the constraints

$$\alpha_k > 0 \ ,$$
$$\sum_{k=1}^{K} \alpha_k = 1 \ .$$

The maximum likelihood parameter values can be estimated using the expectation-maximization (EM) algorithm (Sect. 8.5). Theoretically the degrees of freedom, ν_k, can also be estimated by the EM algorithm. However, in practice [8] it is better to consider the ν_k as regular hyperparameters which are learnt by an exhaustive search.

We associate a "hidden" variable z_i with each $\boldsymbol{y}_i$, where z_i can only take on the values $\{1, 2, \ldots, K\}$. Then, in one-dimension, the EM algorithm for this problem consists of the following two steps, which are repeated until convergence:

E-Step:

$$p(z_i = k \mid y_i, \phi_k, \nu_k, I) = \frac{p(y_i|z_i = k, \phi_k, \nu_k, I) p(\phi_k|z_i = k, I)}{p(y_i|I)} \ ,$$
$$Q(z_i = k \mid y_i, \phi_k, \nu_k) = \frac{\nu_k + 1}{\nu_k + \frac{1}{\nu_k \sigma_k^2}(y_i - \mu_k)^2} \ ,$$

where

$$\phi_k = (\alpha_k, \mu_k, \sigma_k) \ .$$

M-Step:

$$\alpha_k = \frac{1}{N} \sum_{i=1}^{N} w_{ik} \ ,$$
$$\mu_k = \sum_{i=1}^{N} w_{ik} Q(z_i = k|y_i, \phi_k, \nu_k) y_i / \sum_{i=1}^{N} w_{ik} Q(z_i = k|y_i, \phi_k, \nu_k) \ ,$$
$$\sigma_k^2 = \sum_{i=1}^{N} p(z_k = k|y_i, \phi_k, \nu_k, I) Q(z_i = k|y_i, \phi_k, \nu_k)(y_i - \mu_j)^2 / \sum_{i=1}^{N} p(z_i = k|y_i, \phi_k, \nu_k, I) \ ,$$

where

$$w_{ik} = p(z_i = k|y_i, \phi_k, \nu_k, I) \; . \qquad \square$$

Subspace techniques (Sect. 4.5) such as principal component analysis (PCA) and linear discriminant analysis (LDA), are sensitive to outliers and atypical observations. For this reason robust version of the different subspace techniques have been developed. The following example describes a robust probabilistic PCA algorithm.

Example 10.5. Robust Probabilistic Principal Component Analysis [9, 283]. In the robust PPCA we model the sensor measurements $\boldsymbol{y}_i$ using a linear model of the form

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\mu} + \boldsymbol{w} \; ,$$

where we use a Student-t distribution for $\boldsymbol{w}$ and $\boldsymbol{\theta}$:

$$\begin{aligned} \boldsymbol{w} &\sim \mathcal{S}t(\mathbf{0}, \sigma^2 \boldsymbol{I}, \eta) \; , \\ \boldsymbol{\theta} &\sim \mathcal{S}t(\mathbf{0}, \boldsymbol{I}, \eta) \; . \end{aligned}$$

The result is a "robust" *a posteriori* pdf:

$$p(\boldsymbol{y}|\boldsymbol{\theta}, I) \sim \mathcal{S}t(\boldsymbol{y}|\boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\mu}, \tau \boldsymbol{I}, \eta) \; ,$$

which may be efficiently solved using an EM algorithm [9, 283]. $\square$

10.3.2 "Good-and-Bad" Likelihood Function

The "Good-and-Bad" likelihood function uses a weighted sum of two Gaussian functions:

$$p(\boldsymbol{y}|\boldsymbol{\theta}, I) = \alpha \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) + (1 - \alpha)\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \beta^2 \boldsymbol{\sigma}^2) \; , \tag{10.3}$$

where $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is a "Good", or "normal", likelihood function and is used to represent the inliers, $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \beta^2 \boldsymbol{\sigma}^2)$ is a "Bad", or "aberrant", likelihood function and is used to represent the outliers, and $\beta >> 1$. We weight the two Gaussian functions according to the expected proportion of inliers and outliers in the input data.

The "Good-and-Bad" likelihood function is often used to fuse together discordant measurements. The following example illustrates the use of the "Good-and-Bad" likelihood function to fuse five independent neutron fission cross-sections.

Example 10.6. Neutron Fission Cross-Sections [111]. Table 10.3 lists $N = 5$ experimental measurements $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ and error bars $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)^T$ obtained in a given neutron fission cross-section experiment. The corresponding (Gaussian) likelihoods, $p(y_i|\theta, I) \sim \mathcal{N}(y_i, \sigma_i^2)$, are

Table 10.3. Experimental Values y_i and Error Bars σ_i for Neutron Fission Cross-Section

Parameter	Experimental values
y_i	2.385 2.521 2.449 2.420 2.670
σ_i	0.027 0.027 0.027 0.027 0.027

shown in Fig. 10.3. Data point #5 is clearly an outlier being eight standard deviations away from the mean of the other four points. Assuming a flat prior on the cross-section θ, the *a posteriori* probability $p(\theta|\boldsymbol{y}, I)$ is proportional to the product of the individual likelihoods:

$$p(\theta|\boldsymbol{y}, I) \propto \prod_{i=1}^{N} p(y_i|\theta, I) \ .$$

Using "Good" and "Bad" Gaussian likelihood functions, $\mathcal{N}(y_i, \sigma_i^2)$ and $\mathcal{N}(y_i, \beta^2\sigma_i^2)$, the *a posteriori* probability becomes

$$p(\theta|\boldsymbol{y}, I) \propto \int \prod_{i=1}^{5} \big(\alpha\mathcal{N}(\theta|y_i, \sigma_i^2) + (1-\alpha)\mathcal{N}(\theta|y_i, \beta^2\sigma_i^2)\big)\pi(\alpha|I)d\alpha \ ,$$

where $\pi(\alpha|I)$ is the *a priori* pdf of α. Assuming a flat distribution for $\pi(\alpha|I)$, we obtain the *a posteriori* probability density function shown in Fig. 10.3(b). This result is close to the product of $\mathcal{N}(\theta|y_i, \sigma_i^2)$ for $i \in \{1, 3, 4\}$, which is what we might expect if we believe that the points #2 and #5 are outliers that should be discounted. □

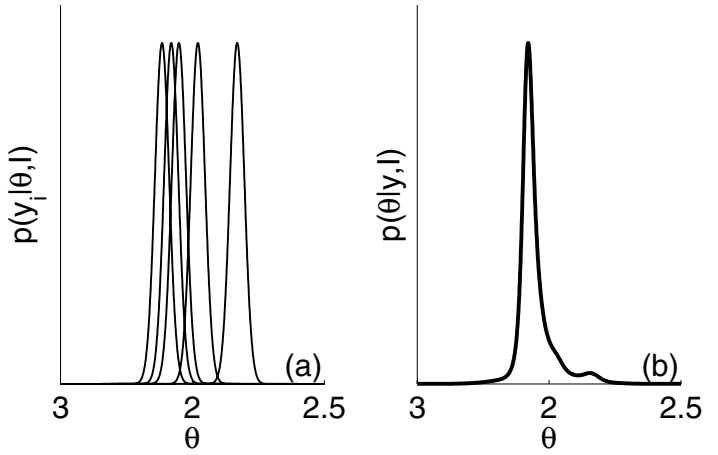

Fig. 10.3. (a) Shows the Gaussian likelihoods $p(y_i|\theta, I) = \mathcal{N}(y_i, \sigma_i^2)$ for cross-sections y_i and error bars σ_i. **(b)** Shows the *a posteriori* probability density function $p(\theta|\boldsymbol{y}, I)$ calculated using a "Good-and-Bad" likelihood function with $\beta = 10$.

We may modify the "Good-and-Bad" likelihood function by including the probability that each input value belongs to the inlier distribution. This is illustrated in the following example.

Example 10.7. Estimating Hubble's Constant Using an "Good-and-Bad" Likelihood Function [2, 249, 281]. Table 10.4 lists $N = 13$ experimental measurements $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ and error bars $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)^T$ of Hubble's constant θ[3]. The measurements include a wide variety of different techniques and are assumed to be independent. We divide the measurements $\{y_i\}$ into "Good" and "Bad" measurements. Altogether there are 2^N different ways, or combinations, of doing this. We use an indicator function $I_{ki}, i \in \{1, 2, \ldots, N\}$, to indicate which measurement is "Good" and which measurement is "Bad" in a given combination. Mathematically,

$$I_{ki} = \begin{cases} 1 \text{ if } i\text{th measurement in } k\text{th combination is "Good"} \\ 0 \text{ if } i\text{th measurement in } k\text{th combination is "Bad"} \end{cases} .$$

Since the measurements are independent, the joint likelihood function for the kth combination is

$$p_k(\boldsymbol{y}|\theta, I) = \prod_{i=1}^{N} \left(I_{ki} \mathcal{N}(y_i|\theta, \sigma_i^2) + (1 - I_{ki}) \mathcal{N}(y_i|\theta, \beta^2 \sigma_i^2) \right) ,$$

and the corresponding *a posteriori* probability density function $p(\theta|\boldsymbol{y}, I)$ is proportional to

$$\sum_{k=1}^{2^N} \pi(k|I)\ p_k(\boldsymbol{y}|\theta, I) ,$$

where $\pi(k|I)$ is the *a priori* probability of the kth combination. By carrying out the summation over all 2^N combinations [4], we obtain the *a posteriori* distribution shown in Fig. 10.4. The pdf, $p(\theta|\boldsymbol{y}, I)$, bears a superficial resemblance to a Gaussian function, which is an *outcome* of the calculation and *not* an assumed input. □

Table 10.4. Experimental Values y_i and Error bars σ_i for Hubble's Constant

Parameter	Experimental values
y_i	49 51 55 61 89 68 67 80 82 76 81 73 80
σ_i	10 13 25 19 10 22 10 25 18 19 17 10 12

[3] The values y_i and the error bars σ_i were read from Fig. 2 in [249]. The values are given in the appropriate units.

[4] The curve shown in Fig. 10.4 was calculated assuming that $\pi(k|I)$ is inversely proportional to the total number of combinations which have the same number of "Good" measurements.

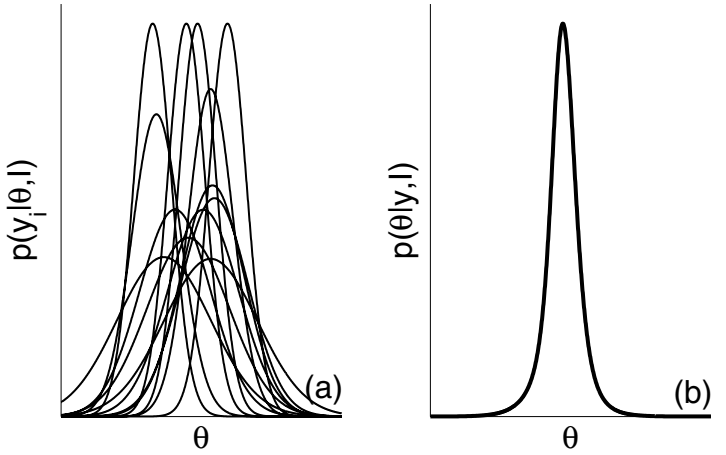

Fig. 10.4. (a) Shows the Gaussian likelihood functions $p(y_i|\theta, I) = \mathcal{N}(y_i, \sigma_i^2)$ for the experimental measurements y_i and error bars σ_i. **(b)** Shows the *a posteriori* probability density function $p(\theta|\boldsymbol{y}, I)$ calculated using a "Good-and-Bad" likelihood function.

10.3.3 Gaussian Plus Constant

The Gaussian plus constant likelihood function is defined as

$$p(\boldsymbol{y}|\boldsymbol{\theta}, I) \propto \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) + F \; . \tag{10.4}$$

The purpose of the constant F is to ensure that the probability $p(\boldsymbol{y}|\boldsymbol{\theta}, I)$ never falls below a given value F. An example of this function is the Konolige function (see Ex. 2.9).

10.3.4 Uncertain Error Bars

In one-dimension, the *uncertain error bar* likelihood function is defined as

$$p(y|\mu, I) = \frac{1}{2|\mu - y| \ln(b/a)} (\mathrm{erf}(|\mu - y|/\sqrt{2a^2}) - \mathrm{erf}(|\mu - y|/\sqrt{2b^2})) \; , \tag{10.5}$$

where the standard deviation σ is assumed to be in the interval $[a, b]$. The uncertain error bar likelihood function is shown in Fig. 10.5. When $a = b$, i. e. the upper and low bounds on σ coincide, (10.5) reduces to $\mathcal{N}(y|\mu, a^2)$ and when $b \to \infty$, (10.5) simplifies to

$$p(y|\mu, I) \propto \frac{1}{|\mu - y|} \mathrm{erf}(|\mu - y|/\sqrt{2a^2}) \; . \tag{10.6}$$

The uncertain error bar likelihood function (10.5) is derived as follows. We assume there are *no* "true" outliers, but rather a y value may give the *appearance* of being an outlier because of a gross error in its uncertainty σ. We

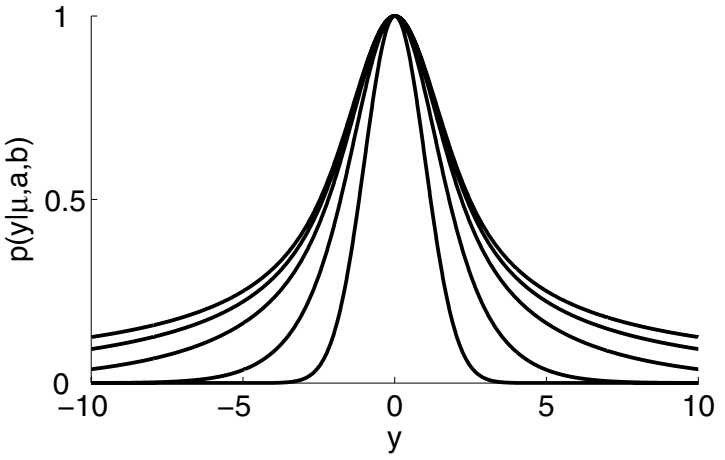

Fig. 10.5. Shows the likelihood $p(y|\mu, a, b, I)$ for $\mu = 0$ and $a = 1$ vertically scaled to unit amplitude. The curves shown are, in order of increasing width and tail size, for $b = 1, 3, 9, 27$ and ∞.

may correct for the error in σ by integrating σ out of the conditional pdf $p(y|\mu, \sigma, I)$:

$$p(y|\mu, I) = \int p(y|\mu, \sigma, I)\pi(\sigma|I)d\sigma \ , \tag{10.7}$$

where $\pi(\sigma|I)$ is the *a priori* distribution for σ. A convenient non-informative distribution (see Sect. 8.4.3) for $\pi(\sigma|I)$ is *Jeffrey's* pdf:

$$\pi(\sigma|I) = \begin{cases} \frac{1}{\sigma}\ln(b/a) & \text{if } a \leq \sigma \leq b \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{10.8}$$

The following example illustrates the use of the uncertain error bar likelihood function in estimating the background signal B in a powder diffraction pattern experiment.

Example 10.8. Background Estimation in a Powder Diffraction Pattern [58]. We consider a powder diffraction pattern. This consists of an unknown and noisy background signal $B(\lambda)$ which varies with the wavelength λ. Imposed on the background are *positive* high amplitude Bragg diffraction peaks A (Fig. 10.6). We consider a small area of the diffraction pattern near a given wavelength λ. In this case we model the background signal is a constant, but unknown, value B. Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ denote N diffraction pattern amplitudes taken in the vicinity of λ. Then given a noise variance σ^2 and any background information I (such as the Bragg peak positivity), the probability distribution for the background signal, B, is

$$p(B|\sigma, \boldsymbol{y}, I) = \int p(A, B|\sigma, \boldsymbol{y}, I)dA \ .$$

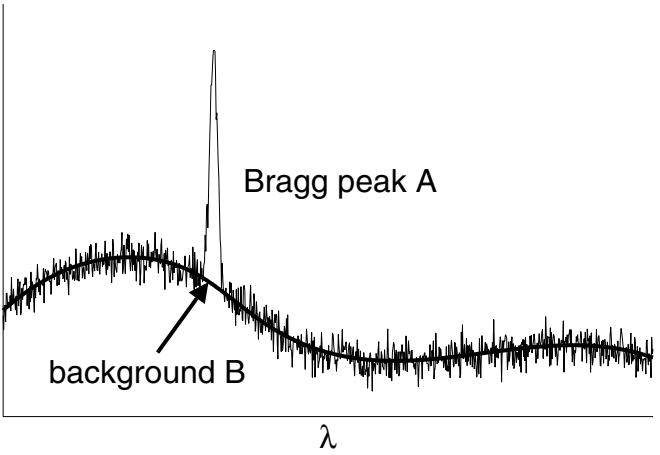

Fig. 10.6. Shows the amplitude of powder diffraction pattern as a function the wavelength λ. Overlaying the noisy diffraction pattern is a thick smooth curve. This is the background signal B. The positive high peak is the Bragg diffraction peak.

Invoking Bayes' theorem, assuming the y_i are independent and separating the peak and background distributions gives

$$\begin{aligned} p(B|\sigma, \boldsymbol{y}, I) &\propto \int p(\boldsymbol{y}|\sigma, A, B, I)\pi(A, B|I)dA \ , \\ &= \pi(B|I) \int p(\boldsymbol{y}|\sigma, A, B, I)\pi(A|I)dA \ , \\ &= \pi(B|I) \prod_{i=1}^{N} \int p(y_i|\sigma, A, B, I)\pi(A|I)dA \ . \end{aligned}$$

A priori it is difficult to scale the Bragg peak contributions relative to the background. As an alternative we model the probability distribution for A using Jeffrey's pdf (10.8):

$$\pi(A|I) = \frac{1}{A \ln(b/a)} \ .$$

where a and b are, respectively, the lower and upper limits for A. Substituting this *a priori* distribution in $p(B|\sigma, \boldsymbol{y}, I)$ gives:

$$p(B|\sigma, \boldsymbol{y}, I) \propto \pi(B|I) \prod_{i=1}^{N} p(y_i|\sigma, B, I) \ ,$$

where

$$p(y|\sigma, B, I) = \int_a^b \frac{1}{A} \exp -\frac{1}{2}\Big(-\frac{A + B - y}{\sigma}\Big)^2 dA \ .$$

denotes the likelihood of y. The natural logarithm of $p(y|\sigma, B, I)$ is shown in Fig. 10.7. □

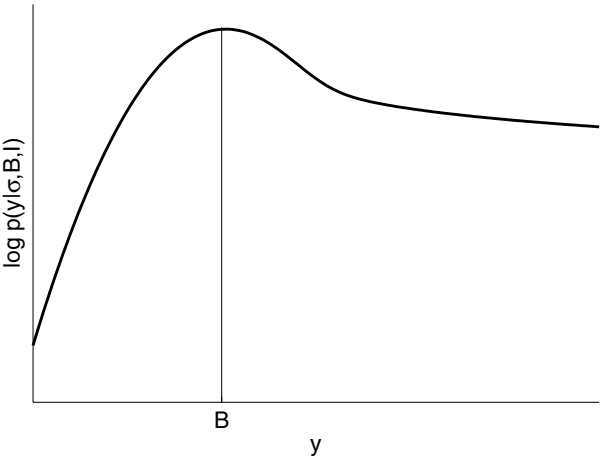

Fig. 10.7. Shows $\log p(y|\sigma, B, I)$. This has the form of a Gaussian distribution for $y < B$, i. e. for measurements y, which lie below the background, B. For measurements which lie above the background, the corresponding robust probability distribution falls off less fast than the Gaussian distribution.

10.4 Classical Robust Estimators

One popular classical robust technique is the M estimators which are a generalization of the maximum likelihood and least squares estimators. The M-estimate of a is defined as

$$\hat{a} = \arg\min_a \sum_i \rho\Big(\frac{y_i - a}{\sigma_i}\Big) \,, \tag{10.9}$$

where $\rho(u)$ is a robust loss function that grows subquadratically and is monotonically nondecreasing with increasing $|u|$ and σ_i^2 is the variance, or scale, associated with the residual errors $\epsilon_i = y_i - a$. The many M-estimators that have been proposed differ in shape of the loss functions $\rho(u)$. Some common loss functions are listed in Table 10.5 . Inherent in the different M-estimators is the need to estimate the standard deviation σ of the residual errors ϵ_i. A robust standard deviation which may be used for this purpose is

$$\sigma = 1.4826\Big(1 + \frac{5}{N-1}\Big)\mathrm{med}_i|y_i - a| \,. \tag{10.10}$$

10.4.1 Least Median of Squares

A widely used alternative to the M estimators is the *least median of squares* (LMedS) estmator in which we calculate the unknown parameter a by solving the following nonlinear minimization problem:

$$a = \arg\big(\min \mathrm{med}_i (y_i - a)^2\big) \,. \tag{10.11}$$

Table 10.5. Robust Loss Functions $L(t, \theta)$

Name	$L(t, \theta)$
L_1	$\|y\|$.
$L_1 - L_2$	$2(\sqrt{1 + y^2/2} - 1)$.
L_p	$\|y\|^p/p$.
"Fair"	$c^2(\frac{\|y\|}{c} - \log(1 + \frac{\|y\|}{c}))$.
Geman-McClure	$(y^2/2)/(1 + y^2)$.
Beaton and Tukey	$c^2(1 - (1 - (u/c)^2)^3)/6$ if $\|u\| \leq c$; otherwise $c^2/6$.
Cauchy	$c^2 \log(1 + (u/c)^2)/2$.
Huber	$u^2/2$ if $\|u\| \leq k$, otherwise $k(2\|u\| - k/2)$.

10.5 Robust Subspace Techniques

Subspace techniques (see Sect. 4.5), such as principal component analysis (PCA) and linear discriminant analysis (LDA), are sensitive to outliers and to atypical observations. In creating robust subspace techniques, the basic idea is to replace non-robust estimators with robust versions. The following example describes a robust LDA algorithm.

Example 10.9. Robust Discriminant Analysis [130]. One commonly used robust estimator is the minimum covariance determinant (MCD) estimator [130]. In the MCD estimator we search for a subset containing n input measurements for which the determinant of their covariance matrix is minimal. Then the MCD-estimator of location is given by the mean of these n measurements and the MCD-estimator of covariance is given by their covariance matrix. □

The probabilistic subspace techniques use an explicit noise model. For example, the probabilistic PCA and LDA algorithms (see Sect. 9.6.3), use a Gaussian noise model. These algorithms are easily robustified by replacing the Gaussian distribution with a corresponding Student-t distribution. See Ex. 10.5 for an example of a robust probabilistic principal component analysis.

10.6 Robust Statistics in Computer Vision

Special purpose robust statistical techniques are required for computer vision and image analysis applications. The input data in these applications are characterized by multiple structures and models, in which each model accounts for a relatively small percentage of the input data [319]. In these circumstances the classical M and LMedS estimators, which require at least 50% inliers, are no longer reliable. In Table 10.6 we list some of the special robust techniques which have been developed for computer vision and image analysis applications [319]. The following example illustrates the use of the Hough transform [319] to initiate target tracks made by a moving object O.

Table 10.6. Robust Techniques for Computer Vision and Image Analysis

Name	Description
Hough Transform	Hough Transform is a voting technique in which we count the number of data features that are mapped into each cell in quantized parameter space. Its main drawbacks are: excessive storage space and computational complexity and limited parameter precision.
RANSAC	Random Sample Consensus. Subsets of input data are randomly selected and are fitted to a given parametric model. The subsets are known as p-subsets and contain p points where p is the dimension of the parameter space. The quality of model parameters are evaluated on all the input data. Different cost functions may be used, the conventional cost is the number of inliers, i. e. the number of input data points which are consistent with the model.
MINIPRAN	Minimize the Probability of Randomness.
MUSE	Minimum Unbiased Scale Eestimator. MUSE randomly selects p-subsets and then estimates fits based on these p-subsets. It then calculates the unbiased estimate of the scale for each fit's k smallest smallest residuals where k is set to all possible values and satisfies $1 \leq k \leq N - p$. The smallest scale estimate over all possible k is chosen as th representative value of hypothesized fits. Finally the fit from the p-subset with the smallest scale estimate is chosen as the optimum fit.
MDPE	Maximum Density Power Estimator.
ALKS	Adaptive Least kth Order Square. ALKS is based on the least kth order squares (LKS) estimator. The diference between LKS and LMedS is that LKS uses k data points out of N data points $p < k < N$) while LMedS uses $N/2$ data points. ALKS uses a multi-step procedure in which in each step ALKS employs LKS with a different k value. In order to estimate the correct k, a random sampling technique similar to that used in RANSAC is employed. ALKS is robust against multiple structures but it cannot resist the influence of extreme outliers.
RESC	Residual Consensus. The idea of RESC is that if a model is correctly fitted, the residuals of the inliers should be small and at the same time the histogram of the residuals should concentrate within a small range in the lower part of the histogram. The RESC is very robust. However its main disadvantage is that it requires many parameters which must be tuned by the user for optimal performance.

Example 10.10. Target Tracking Initialization Using the Hough Transform [24]. We consider an object O which starts to move at time step 0 in a two-dimensional, (x, y), space. A sensor, S, detects (with a probability $p_D < 1$) the position of O at time steps $k, k \in \{0, 1, 2, \ldots\}$. In addition, the sensor may make several false detections at these times. If we assume that initially O

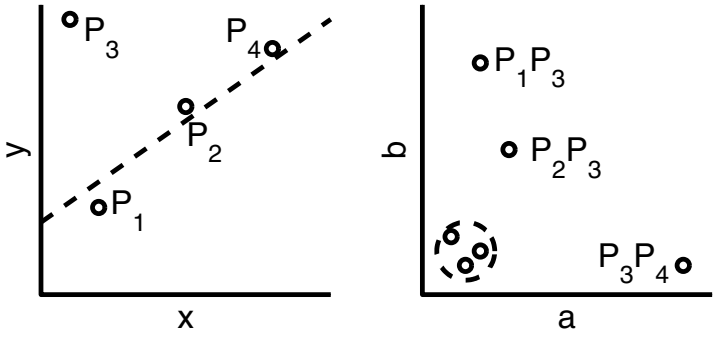

Fig. 10.8. (a) Shows four observations $P_i = (x_i, y_i), i \in \{1, 2, \ldots, 4\}$ made at different times t_i by a sensor S. Observations P_1, P_2 and P_4 are associated with the object and observation P_3 is a false alarm. **(b)** Shows the corresponding six points (a_{ij}, b_{ij}) which represent the straight-lines P_iP_j in Hough space. There is a single cluster of three (a_{ij}, b_{ij}) points which represents the straight-lines P_1P_2, P_1P_3 and P_2P_3 and three isolated (a_{ij}, b_{ij}) points which represent the straight-lines P_1P_3, P_2P_3 and P_4P_3.

moves along a straight-line, we may use the Hough transform to identify the initial track as follows.

Consider a pair of detections $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$ made at *different* time steps k_i and k_j. The parameters a_{ij} and b_{ij} of the straight-line P_iP_j is

$$\frac{x}{a_{ij}} + \frac{y}{b_{ij}} = 1 \,,$$

where

$$a_{ij} = \frac{x_i y_j - x_j y_i}{y_j - y_i} \,,$$
$$b_{ij} = \frac{y_i x_j - y_j x_i}{x_j - x_i} \,.$$

We record each of pair of a_{ij} and b_{ij} values in a Hough space (i. e. the two-dimensional (a, b) space). Then we identify the initial track of the object O by finding the point (a^*, b^*) around which there is a cluster of (a_{ij}, b_{ij}) values (see Fig. 10.8). We often only accept (a^*, b^*) if the number of points in the cluster exceeds a given threshold. The strength of the Hough transform is that we do not specify the number of straight-line tracks. Instead, the Hough transform automatically estimates the number of tracks present. □

10.7 Software

LIBRA (Matlab Library for Robust Analysis). A matlab toolbox for performing robust statistical analysis.

MATLAB STATISTICS TOOLBOX. The matlab statistics toolbox.

10.8 Further Reading

Robustifying the Bayesian framework, i. e. making the Bayesian framework robust against outliers, is a controversial subject which is still being intensively investigated. Some modern references on the subject are: [21, 230, 281]. The development of robust EM algorithms based on Student-t functions are discussed in detail in [8, 200]. A comprehensive review of robust subspace techniques is [283]. Classical references on robust estimation are [110, 129, 269]. See also [311]. Special robust techniques which have been developed for computer vision which are described and in [203, 285, 319, 352]. A full description of Bayesian curve fitting including robust Bayesian curve fitting is given in [61].

11

Sequential Bayesian Inference

11.1 Introduction

The subject of this chapter is *sequential Bayesian inference* in which we consider the Bayesian estimation of a dynamic system which is changing in time. Let $\boldsymbol{\theta}_k$ denote the state of the system, i. e. a vector which contains all relevant information required to describe the system, at some (discrete) time k. Then the goal of sequential Bayesian inference is to estimate the *a posteriori* probability density function (pdf) $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:l}, I)$, by fusing together a sequence of sensor measurements $\boldsymbol{y}_{1:l} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_l)$. In this chapter we shall only consider the calculation of pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:l}, I)$ for $k = l$ which is known as (sequential) Bayesian *filtering* or *filtering* for short[1].

In the first half of the chapter we shall assume the measurements are all made by the same sensor. In this case we have *multi-temporal data fusion*. In the second half of the chapter we remove this restriction and assume the measurements are made by more than one sensor. In this case we have *multi-sensor multi-temporal data fusion*.

For many applications an estimate of $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ is required every time step that a measurement is received. In this case it is common practice to rewrite $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ using a recursive formulation of Bayes' theorem:

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I) = \frac{\overbrace{p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, I)}^{\text{likelihood}}\overbrace{p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)}^{\textit{a priori}\text{ pdf}}}{\underbrace{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, I)}_{\text{evidence}}}\,, \tag{11.1}$$

[1] The calculation of $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:l}, I)$ for the cases $k < l$ and $k > l$ are known, respectively, as (sequential) Bayesian *smoothing* and *forecasting*.

where

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I) = \int p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, I)p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)d\boldsymbol{\theta}_{k-1} \ , \tag{11.2}$$

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, I) = \int p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, I)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)d\boldsymbol{\theta}_k \ . \tag{11.3}$$

Eqs. (11.1)-(11.3) represent a convenient solution to the problem of (sequential) Bayesian filtering: the sensor measurements are processed sequentially and not as a batch. At each time step only the current sensor measurement, $\boldsymbol{y}_k$, is used and it is not necessary to maintain, or re-process, previous measurements.

The main difficulty in calculating $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ involves solving the integrals defined in (11.2) and (11.3) and this will form the main topic of our discussion in this chapter.

In many applications the recursive filter is the primary fusion algorithm in a multi-sensor multi-temporal data fusion system. In Table 11.1 we list some examples of these applications together with their input/output classification.

Table 11.1. Applications in which the Sequential Filter is the Primary Fusion Algorithm

Class		Application
DaI-DaO	Ex. 3.6	Heart Rate Estimation Using a Kalman Filter.
	Ex. 9.6	Non-Intrusive Speech Quality Estimation Using GMM's.
	Ex. 11.2	Tracking a Satellite's Orbit Around the Earth.
	Ex. 11.3	Tracking a Moving Target.
	Ex. 11.4	Tracking Metrological Features.
	Ex. 11.5	Correcting near surface temperature forecasts using a Kalman filter
	Ex. 11.10	The Lainiotis-Kalman Filter.
	Ex. 11.11	Multi-Sensor Multi-Temporal Data Fusion: Measurement Fusion.
DaI-FeO	Ex. 11.8	Recursive Parameter Estimation for a First-Order Autoregressive Process.
FeI-FeO	Ex. 11.1	INS/Radar Altimeter: A Hybrid Navigation Algorithm.
	Ex. 11.11	Multi-Sensor Multi-Temporal Data Fusion: Measurement Fusion.
	Ex. 11.12	Multi-Sensor Multi-Temporal Data Fusion: Track-to-Track Fusion.
	Ex. 11.14	Modified Track-to-Track Fusion Algorithm.

The designations DaI-DaO, DaI-FeO and FeI-FeO refer, respectively, to the Dasarathy input/output classifications: "Data Input-Data Output", "Data Input-Feature Output" and "Feature Input-Feature Output" (see Sect. 1.3.3).

11.2 Recursive Filter

The recursive filter is defined in (11.1)-(11.3). These equations may be derived as follows.

1. Eq. (11.1) represents a recursive formulation of Bayes' theorem. It is obtained by applying Bayes' theorem to the last measurement in $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$:

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I) &= p(\boldsymbol{\theta}_k|\boldsymbol{y}_k, \boldsymbol{y}_{1:k-1}, I) \ , \\ &= \frac{p(\boldsymbol{\theta}_k, \boldsymbol{y}_k, \boldsymbol{y}_{1:k-1})}{p(\boldsymbol{\theta}_k, \boldsymbol{y}_{1:k-1}, I)} \ , \\ &= \frac{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta}_k, I)p(\boldsymbol{y}_{1:k-1}|\boldsymbol{\theta}_k, I)p(\boldsymbol{\theta}_k|I)}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, I)p(\boldsymbol{y}_{1:k-1}|I)} \ , \\ &= \frac{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta}_k, I)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)p(\boldsymbol{y}_{1:k-1}|I)}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, I)p(\boldsymbol{y}_{1:k-1}, I)} \ , \\ &= \frac{p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, I)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, I)} \ , \end{aligned} \tag{11.4}$$

   where the last equation follows from the assumption that the $\boldsymbol{y}_k$ obey a *Markov measurement model* in which $\boldsymbol{y}_k$ is independent of the previous states $\boldsymbol{\theta}_{1:k-1}$ and the previous measurements $\boldsymbol{y}_{1:k-1}$, i. e.

$$p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, \boldsymbol{y}_{1:k-1}, I) = p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, I) \ . \tag{11.5}$$

2. Eq. (11.2) represents an integral formulation of the *a priori* pdf. It is obtained by marginalizing $\boldsymbol{\theta}_{k-1}$ out of the joint pdf $p(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)$:

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I) &= \int p(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)d\boldsymbol{\theta}_{k-1} \ , \\ &= \int p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{y}_{1:k-1}p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)d\boldsymbol{\theta}_{k-1} \ , \\ &= \int p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, I)p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)d\boldsymbol{\theta}_{k-1} \ , \end{aligned} \tag{11.6}$$

   where the last equation follows from the assumption that the $\boldsymbol{\theta}_k$ obey a *Markov process model* in which the $\boldsymbol{\theta}_k$ depends only on $\boldsymbol{\theta}_{k-1}$:

$$p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{y}_{1:k-1}, I) = p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, I) \ . \tag{11.7}$$

3. Eq. (11.3) represents an integral formulation of the Bayesian *evidence*. It follows directly from the definition of evidence which acts as a normalization factor in Bayes' rule.

In order to carry out the integrals in (11.2) and (11.3) we rewrite the Markov process and measurement models as stochastic equations:

$$\boldsymbol{\theta}_k = \boldsymbol{f}_k(\boldsymbol{\theta}_{k-1}, \boldsymbol{v}_{k-1}) \ , \tag{11.8}$$

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{\theta}_k, \boldsymbol{w}_k) \ , \tag{11.9}$$

where $\boldsymbol{v}_{k-1}$ is a noise term which represents the stochastic disturbances in the system and $\boldsymbol{w}_k$ is a noise term which represents the errors in the measurement process. Substituting (11.8) in (11.2) gives

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, I) &= \int p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{v}_{k-1}, I) p(\boldsymbol{v}_{k-1}|\boldsymbol{\theta}_{k-1}, I) d\boldsymbol{v}_{k-1} \ , \\ &= \int \delta\big(\boldsymbol{\theta}_k - \boldsymbol{f}_k(\boldsymbol{\theta}_{k-1}, \boldsymbol{v}_{k-1})\big) p(\boldsymbol{v}_{k-1}|I) d\boldsymbol{v}_{k-1} \ , \quad (11.10) \end{aligned}$$

where the last equation follows from the assumption that the noise is independent of the state. Likewise, substituting (11.9) in (11.3) gives

$$\begin{aligned} p(\boldsymbol{y}_{1:k}|\boldsymbol{\theta}_k, I) &= \int p(\boldsymbol{y}_{1:k}|\boldsymbol{\theta}_k, \boldsymbol{w}_k, I) p(\boldsymbol{w}_k|\boldsymbol{\theta}_k, I) d\boldsymbol{w}_k \ , \\ &= \int \delta(\boldsymbol{y}_k - h_k(\boldsymbol{\theta}_k, \boldsymbol{w}_k)) p(\boldsymbol{w}_k|I) d\boldsymbol{w}_k \ . \quad (11.11) \end{aligned}$$

In Fig. 11.1 we show the main processing steps in the Bayesian recursive filter. The filter is initialized by specifying an *a priori* pdf, $p(\boldsymbol{\theta}_0|\boldsymbol{y}_0, I)$, where $\boldsymbol{y}_0$ denotes that no measurements have been made. The filter then operates recursively, performing a single cycle each time a new measurement becomes available. Suppose the *a posteriori* pdf, $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)$, has been estimated at time step $k-1$ and that a new measurement has become available at time

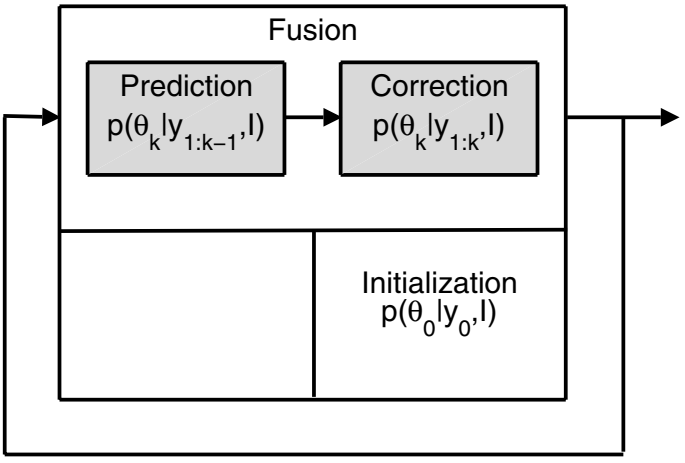

Fig. 11.1. Shows the main processing steps in the recursive filter: (1) Initialization. We give an initial pdf $p(\boldsymbol{\theta}_0|\boldsymbol{y}_0, I)$. This step is performed only once. (2) Prediction. We calculate a predicted pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)$ using the process model $\boldsymbol{\theta}_k = \boldsymbol{f}_{k-1}(\boldsymbol{\theta}_{k-1}, \boldsymbol{v}_{k-1})$ and the *a posteriori* calculated in the previous time step. (2) Correction. We calculate the *a posteriori* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ by correcting the predicted pdf using the current measurement $\boldsymbol{y}_k$. The architecure used to represent the filter is the iterative fusion cell (see Sect. 3.3.4).

step k. Then we calculate a *predicted* pdf, $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)$, by propagating $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)$ from the time step $k-1$ to k. The predicted pdf acts as the *a priori* pdf for the time step k and is calculated using a *process model* which describes the evolution of the system. In general the predicted pdf will contain errors due to inaccuracies in the process model and noise in the input measurements. To eliminate these errors we correct $p\,(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)$ using the latest sensor measurement $\boldsymbol{y}_k$. This is achieved by using Bayes' theorem which is a mechanism for updating our knowledge in the light of new information (see Chapt. 8).

Fig. 11.2 shows the shapes of the different probability distributions in a recursive filter.

The following example illustrates the use of (11.1)-(11.3) in terrain-aided navigation.

Example 11.1. INS/Radar Altimeter: A Hybrid Navigation Principle [85]. Terrain-aided navigation is a method for aircraft navigation in which radar altimeter measurements are combined with inertial measurements (INS) to achieve an optimal (minimum mean square error) estimate of the position of the aircraft. Terrain-aided navigation works as follows: At each time step k we use the INS measurements (accelerometer and gyro-meter) to compute $\boldsymbol{\theta}_k$, the INS estimate of the target. We then use sequential Bayesian inference to fuse $\boldsymbol{\theta}_k$ with the radar altimeter measurements y_k. The output is the INS drift

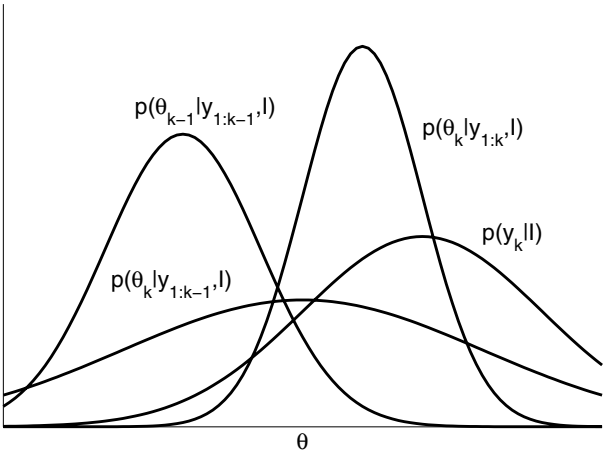

Fig. 11.2. Shows the shapes of the different probability distributions in a generic recursive filter. In the prediction step we use the process model (11.7) to propagate forward the pdf $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}, I)$ to the next time step k. The result is a predicted pdf which functions as the *a priori* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I)$ for the time step k. Because of the noise $\boldsymbol{v}_{k-1}$ the *a priori* pdf is a deformed, spread-out version of $p(\boldsymbol{\theta}_{1-k}|\boldsymbol{y}_{1:k-1}, I)$. In the correction step we use the sensor measurement $p(\boldsymbol{y}_k|I)$ to modify the *a priori* pdf. The result is a sharp *a posteriori* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ for the time step k.

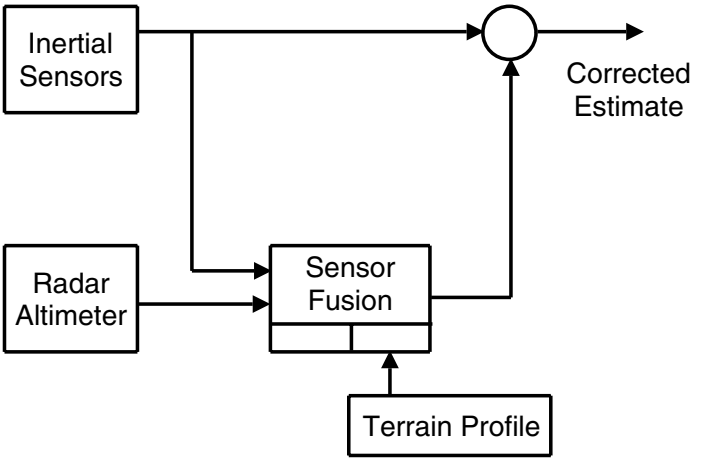

Fig. 11.3. Shows the sensor-fusion filter for terrain-aided navigation. The sensor-fusion filter outputs the INS drift estimate which is then removed from the initial INS estimate to give a corrected INS estimate.

$\delta\boldsymbol{\theta}_k$ (i. e. error in $\boldsymbol{\theta}_k$). The INS drift is then used to correct $\boldsymbol{\theta}_k$ as shown in Fig. 11.3. Mathematically, the hybrid navigation algorithm works as follows. We write the INS estimate at time step k as a linear Gaussian equation (see Sect. 9.6)

$$\boldsymbol{\theta}_k = \boldsymbol{F}_k\boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ ,$$

where $\boldsymbol{\theta}_k$ denotes the INS estimates (latitude, longitude, altitude, roll, pitch, yaw and speed) and $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{k-1})$ is the noise vector in $\boldsymbol{\theta}_{k-1}$.

The scalar radar altimeter measurement equation is given by

$$y_k = z_k - h(l_k, L_k) + w_k \ ,$$

where l_k, L_k and z_k are, respectively, the longitude, the latitude and the altitude of the aircraft. The function h stands for the terrain profile which is stored on-board the aircraft. The noise $w_k \sim \mathcal{N}(0, R_k)$ represents the measurement error.

Using the above equations for $\boldsymbol{\theta}_k$ and y_k we may compute the *a posteriori* pdf $p(\boldsymbol{\theta}_k|y_{1:k}, I)$ using (11.1) and (11.3). The INS drift is given by

$$\delta\boldsymbol{\theta}_k = \boldsymbol{\mu}_k - \boldsymbol{\theta}_k \ ,$$

where $\boldsymbol{\mu}_k = \int \boldsymbol{\theta}_k p(\boldsymbol{\theta}_k|y_{1:k}, I) d\boldsymbol{\theta}_k$ is the mean INS value at time step k. □

11.3 Kalman Filter

For most applications, (11.2) and (11.3) are analytically intractable and approximate solutions must be used. However, for the special case of linear Gaussian systems (Sect. 9.6), the equations are tractable and a closed-form

recursive solution for the sequential Bayesian filter is available. This is the *Kalman* filter (KF) [31] and because of its computational efficiency, it quickly established itself as the favourite algorithm for sequential Bayesian inference.

Mathematically, the KF assumes a linear Gaussian process model:

$$\boldsymbol{\theta}_k = \boldsymbol{F}_k \boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ , \tag{11.12}$$

and a linear Gaussian measurement model:

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{\theta}_k + \boldsymbol{w}_k \ , \tag{11.13}$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{k-1})$ and $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ are independent Gaussian noise sources (see Sect. 9.6). If we assume the initial pdf is Gaussian, $\pi(\boldsymbol{\theta}_0|\boldsymbol{y}_0, I) \sim \mathcal{N}(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$, then both the predicted pdf (11.2) and the *a posteriori* pdf (11.1) are Gaussian:

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}, I) \sim \mathcal{N}(\boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1}) \ , \tag{11.14}$$

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I) \sim \mathcal{N}(\boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k}) \ , \tag{11.15}$$

where

$$\boldsymbol{\mu}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{\mu}_{k-1|k-1} \ , \tag{11.16}$$

$$\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{Q}_{k-1} + \boldsymbol{F}_k \boldsymbol{\Sigma}_{k-1|k-1} \boldsymbol{F}_k^T \ , \tag{11.17}$$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k \boldsymbol{\mu}_{k|k-1}) \ , \tag{11.18}$$

$$\boldsymbol{\Sigma}_{k|k} = \boldsymbol{\Sigma}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{\Sigma}_{k|k-1} \ , \tag{11.19}$$

and $\boldsymbol{K}_k$ is the *Kalman gain matrix*,

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \ . \tag{11.20}$$

Note. For reasons of numerical stability we often rewrite (11.19) as

$$\boldsymbol{\Sigma}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{\Sigma}_{k|k-1} (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^T + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^T \ , \tag{11.21}$$

where $\boldsymbol{I}$ is the unit matrix.

Fig. 11.4 is a block diagram which shows the main processing steps in the KF.

The KF is used in a very wide variety of multi-sensor data fusion applications. Ex. 3.6 illustrates the use of a KF to estimate a patients heart rate and the next example illustrates the use of the KF to track a satellite's orbit around the earth.

Example 11.2. Tracking a Satellite's Orbit Around the Earth [202]. The unknown state $\boldsymbol{\theta}_k$ is the position and speed of the satellite at time step k with respect to a spherical coordinate system with origin at the center of the earth. These quantities cannot be measured directly. Instead, from tracking stations around the earth, we obtain measurements of the distance to the satellite and

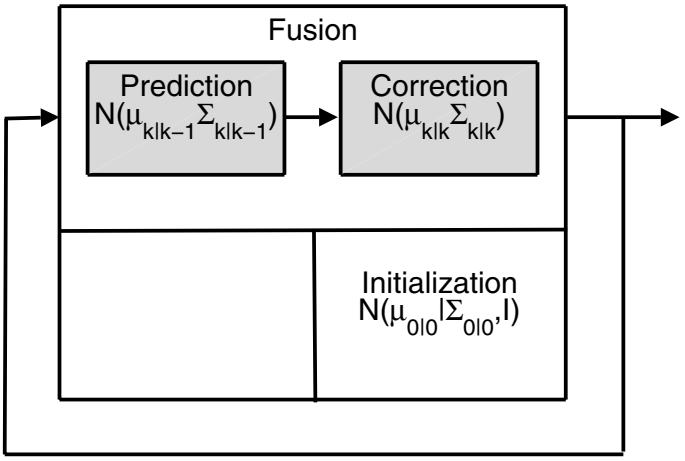

Fig. 11.4. Shows the main processing steps in the Kalman filter: (1) Initialization. We give initial parameter values $(\boldsymbol{\mu}_{0|0}|I)$ and $\boldsymbol{\Sigma}_{0|0}|I)$ and the noise covariances $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$. This step is performed only once. (2) Prediction. We calculate a predicted pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:k}, I) = \mathcal{N}(\boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1})$ using the process model $\boldsymbol{\theta}_k = \boldsymbol{F}_k(\boldsymbol{\theta}_{k-1}, \boldsymbol{v}_{k-1})$ and the *a posteriori* calculated in the previous time step. (2) Correction. We calculate the *a posteriori* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I) = \mathcal{N}(\boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k})$ by correcting the predicted pdf using the current measurement $\boldsymbol{y}_k$.

the accompanying angles of the measurement. These form the measurements $\boldsymbol{y}_k$. The geometrical principles of mapping $\boldsymbol{y}_k$ into $\boldsymbol{\theta}_k$, are incorporated into $\boldsymbol{H}_k$, while $\boldsymbol{w}_k$ reflects the measurement error; $\boldsymbol{F}_k$ prescribes how the position and speed change in time according to the physical laws governing orbiting bodies, while $\boldsymbol{v}_k$ allows for deviations from these laws owing to such factors as nonuniformity of the earth's gravitational field etc. □

The following example describes the tracking of a moving target which is subject to stocastic accelerations.

Example 11.3. Tracking a Moving Target [13, 24]. We consider a target moving in one-dimension at a nomimal constant velocity while being subjected to *stochastic* accelerations. If θ_k denotes the true position of the target at the kth time step, then the corresponding matrix equation of motion of the target is:

$$\begin{pmatrix} \theta_k \\ \dot{\theta}_k \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{k-1} \\ \dot{\theta}_{k-1} \end{pmatrix} + \begin{pmatrix} \frac{T^2}{2} \\ T \end{pmatrix} \ddot{\theta}_{k-1} \ ,$$

where $\ddot{\theta}_k \sim \mathcal{N}(0, \sigma^2)$ is the (stochastic) acceleration which acts upon the target at time step k and T is the interval between successive time steps. The corresponding process model is

$$\begin{pmatrix} \theta_k \\ \dot{\theta}_k \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_k \\ \dot{\theta}_k \end{pmatrix} + \boldsymbol{v}_{k-1} \ .$$

Equating the matrix equation of motion model with the process model leads to the identification of $\begin{pmatrix} \frac{T^2}{2} \\ T \end{pmatrix} \ddot{\theta}_k$ with the process noise $\boldsymbol{v}_{k-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$. In this case,

$$\boldsymbol{Q} = \sigma^2 \begin{pmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{pmatrix} . \qquad \square$$

Ex. 11.2 and 11.3 illustrate a common situation in which we use the KF to track a well-defined object with known process and measurement models. However, the KF has been successfully used to track objects which are not well-defined as the following example illustrates.

Example 11.4. Tracking Metrological Features [192]. In general metrological features, such as storm centers, are not well-defined: they can change size and intensity; they can split into separate storms; and they can merge with other storms. In addition, individual storms rarely satisfy the linear-Gaussian assumption of the KF. Nevertheless, in practice [192], the KF has been used to track storm centers with reasonable accuracy. $\square$

11.3.1 Parameter Estimation

The KF is defined in (11.14)-(11.20). It contains the parameters $\boldsymbol{\mu}_{0|0}$, $\boldsymbol{\Sigma}_{0|0}$, $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ which must be defined before running the filter. In general the values chosen for $\boldsymbol{\mu}_{0|0}$ and $\boldsymbol{\Sigma}_{0|0}$ do not seriously affect the medium and long-term results obtained with the filter, i. e. the values $\boldsymbol{\mu}_{k|k}$ and $\boldsymbol{\Sigma}_{k|k}$ for $k >> 0$. For this reason we often use simple *a priori* values for $\boldsymbol{\mu}_{0|0}$ and $\boldsymbol{\Sigma}_{0|0}$ or use a track initiation filter (see Ex. 10.10).

Regarding $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$, the way that $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ are specified crucially affects $\boldsymbol{\mu}_{k|k}$ and $\boldsymbol{\Sigma}_{k|k}$. In mathematical terms, the values of $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ affect the Kalman gain $\boldsymbol{K}_k$ which in turn affects the capability of the filter to adjust itself to changing conditions.

The following example illustrates a simple, but very effective, procedure for *adaptively* calculating $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$.

Example 11.5. Correcting near surface temperature forecasts using a Kalman filter [95]. Numerical weather prediction models often exhibit systematic errors in the forecasts of near surface weather parameters. One such parameter is the near surface temperature forecast T which is a commonly biased variable, where the magnitude of the bias depends among other factors, on the geographical location and the season. To correct for the systematic error in T let the measurement y_k be the difference between an observed temperature and the weather prediction model forecast and let the state vector θ_k be the

systematic part of this error. We assume the changes in θ_k are random. The corresponding process and measurement models are then

$$\theta_k = \theta_{k-1} + v_{k-1} \ ,$$
$$y_k = \theta_k + w_k \ ,$$

where $v_k \sim \mathcal{N}(0, Q_k)$ and $w_k \sim \mathcal{N}(0, R_k)$. In [95] the authors estimate Q_k and R_k using the sample variances of the N latest $v_{k-n} = \theta_{k-n} - \theta_{k-n-1}$ and $w_{k-n} = y_{k-n} - \theta_{k-n}$ values:

$$Q_k = \frac{1}{N-1} \sum_{n=0}^{N-1} \big((\theta_{k-n} - \theta_{k-n-1}) - \bar{Q}_k \big)^2 \ ,$$
$$R_k = \frac{1}{N-1} \sum_{n=0}^{N-1} \big((y_{k-n} - \theta_{k-n}) - \bar{R}_k \big)^2 \ ,$$

where

$$\bar{Q}_k = \frac{1}{N} \sum_{n=0}^{N-1} (\theta_{k-n} - \theta_{k-n-1}) \ ,$$
$$\bar{R}_k = \frac{1}{N} \sum_{n=0}^{N-1} (y_{k-n} - \theta_{k-n}) \ . \qquad \square$$

See [13, 24, 31] for more sophisticated procedures for calculating $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$.

11.3.2 Data Association

In real-world applications spurious measurements are often present (see Sect. 2.5.1). These are called false alarms, or clutter, and their presence creates an ambiguity regarding the origin of the measurements, i. e. it is often not clear which measurement corresponds to the system (a "true" measurement) and which measurement corresponds to clutter (a "false" measurement) [2]. This is known as the *data association* problem [51] and it is clear that it must be solved before we can use the KF.

In general, the solution of the data association problem can be *hard* or *soft*. In a hard solution we assign a given measurement to the system. In this case, the state of the system, $\boldsymbol{\theta}_k$, is updated assuming that the assigned measurement is the true measurement. On the other hand, in a soft solution we assign more than one measurement to the system, each measurement having its own assignment probability. In this case, we use each measurement which has an

[2] We also allow for the possibility that the sensor will not always return a true measurement.

assignment probability greater than than zero to update $\boldsymbol{\theta}_k$. The collection of updated states is then combined using the assignment probabilities.

In general, different applications will often require different data association algorithms. However, the algorithms given in Table 11.2 have been found useful in a wide-range of applications although they were originally developed for single-target tracking [13, 24]. In general the algorithms listed in Table 11.2 are combined with the KF to create a a new recursive filter. We shall consider three such filters.

Table 11.2. Data Association Algorithms

Name	Description
Nearest Neighbour (NN)	Assumes the measurement that is closest to the predicted state of the system is the true measurement. If the true measurement is not detected at time step k, then the nearest neighbour will always be a false measurement which may cause erratic behaviour in the KF. To avoid this we normally restrict ourselves to *validated* measurements. In general, the performance of the NN and the validated NN algorithms is low.
Strongest Neighbour (SN)	Assumes the measurement with the strongest amplitude is the true measurement. As in the case of nearest neighbour association it is common practice to restrict ourselves to validated measurements. In general, the performance of the SN algorithms are better than the NN algorithms but worse than PDA algorithm. However, the performance of the refined SN filter (Ex. 11.6) is found to greatly exceed both the NN and the PDA algorithms.
Probabilistic Data Association (PDA)	The state of the system is separately updated using each validated measurement in turn. The collection of updated states are then combined together to form the final state of the system. Each validated measurement $\boldsymbol{y}_k^{(m)}, m \in \{1, 2, \ldots, M_k\}$, is given a weight $\beta^{(m)}$ which is the probability that $\boldsymbol{y}_k^{(m)}$ is the true measurement. Also computed is a weight $\beta^{(0)}$ which is the probability that none of the validated measurements are the true measurement. These events encompass all possible interpretations of the data, so $\sum_{m=0}^{M_k} \beta^{(m)} = 1$.
Multiple Hypothesis Testing (MHT)	Enumerates all possible hard assignments taken over all time steps. The number of possible assignments increases exponentially with time. As a consequence at each time step the MHT algorithm discards assignments which have a very low probability of being true.

Validated measurements are measurements which fall inside a given region, or validation gate. In general, the validation gate is an area centered on the predicted state of the system. See Ex. 10.2.

Nearest Neighbour Filter

Let $\widetilde{\boldsymbol{y}}_k = \{\boldsymbol{y}_k^{(1)}, \boldsymbol{y}_k^{(2)}, \ldots, \boldsymbol{y}_k^{(M_k)}\}$, denote the set of M_k measurements received at time step k [3]. In the nearest neighbour (NN) filter we assume the true measurement is

$$\boldsymbol{y}_k^{(*)} = \arg\min_m (D^{(m)}) \tag{11.22}$$

where

$$D^{(m)} = (\boldsymbol{y}_k^{(m)} - \hat{\boldsymbol{y}}_{k|k-1})^T \hat{\boldsymbol{S}}_{k|k-1}^{-1} (\boldsymbol{y}_k^{(m)} - \hat{\boldsymbol{y}}_{k|k-1}) \,, \tag{11.23}$$

is the normalized distance between $\boldsymbol{y}_k^{(m)}$ and the predicted measurement $\hat{\boldsymbol{y}}_{k|k-1}$, and $\hat{\boldsymbol{S}}_{k|k-1}$ is the predicted covariance matrix $\boldsymbol{H}_k^T \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k$.

In the modified nearest neighbour rule we only use *validated* measurements (i. e. measurements $\boldsymbol{y}_k^{(*)}$ for which $D^{(*)} \leq \gamma$, where γ is a given threshold [4]). The corresponding *nearest neighbour filter* combines the KF with a validated NN solution of the data association problem:

$$\boldsymbol{\mu}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{\mu}_{k-1|k-1} \,, \tag{11.24}$$

$$\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{Q}_{k-1} + \boldsymbol{F}_k \boldsymbol{\Sigma}_{k-1|k-1} \boldsymbol{F}_k^T \,, \tag{11.25}$$

$$\boldsymbol{\mu}_{k|k} = \begin{cases} \boldsymbol{\mu}_{k|k-1} + \boldsymbol{K}_k (\boldsymbol{y}_k^{(*)} - \boldsymbol{H}_k \boldsymbol{\mu}_{k|k-1}) & \text{if } m_k \geq 1 \,, \\ \boldsymbol{\mu}_{k|k-1} & \text{otherwise} \,, \end{cases} \tag{11.26}$$

$$\boldsymbol{\Sigma}_{k|k} = \begin{cases} \boldsymbol{\Sigma}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{\Sigma}_{k|k-1} & \text{if } m_k \geq 1 \,, \\ \boldsymbol{\Sigma}_{k|k-1} & \text{otherwise} \,, \end{cases} \tag{11.27}$$

where m_k is the number of validated measurements at time step k, and

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{\Sigma}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \,. \tag{11.28}$$

Strongest Neighbour Filter

The strongest neighbour (SN) filter is identical to the NN filter except we identify the true measurement as $\boldsymbol{y}^{(*)}$, where $\boldsymbol{y}^{(*)}$ is the measurement with the largest *validated* amplitude. The following example describes a refined, or probabilistic, SN filter.

Example 11.6. A Probabilistic Strongest Neighbour Filter for Tracking in Clutter [177]. In the probabilistic SN filter we weight the strongest neighbour measurement (in the validation gate) with the probability that it is the true measurement. This is in opposition to the SN filter which makes the unrealistic assumption that the SN measurement is always the true measurement. The result is a filter which has substantially better performance than the NN, SN and PDA filters. □

[3] The reader, may find it convenient to regard each measurement $\boldsymbol{y}_k^{(m)}$ as originating from a separate sensor $S^{(m)}$. In this case, at most, one of the sensors returns a true measurement. Each of the other sensors returning a false measurement.

[4] See Sect. 10.2.

Covariance Union Filter

The *covariance union filter* is similar to the nearest neighbor and strongest neighbour filters with one crucial difference. The measurement which is selected by the covariance filter is a "pseudo" meaurement characterized by a "pseudo" value $\boldsymbol{y}_k^{(*)}$ and a "pseudo" covariance matrix $\boldsymbol{\Sigma}_k^{(*)}$ which are chosen to be *consistent* with all validated measurements.

Given two observations $O^{(1)} = (\boldsymbol{y}_k^{(1)}, \boldsymbol{R}_k^{(1)})$ and $O^{(2)} = (\boldsymbol{y}_k^{(2)}, \boldsymbol{R}_k^{(2)})$ (see Fig. 11.5) we may calculate $\widetilde{O} = (\boldsymbol{y}_k^{(*)}, \boldsymbol{R}_k^{(*)})$ as follows [146]:

$$\boldsymbol{y}_k^{(*)} = \frac{1}{2}(\boldsymbol{y}_k^{(1)} + \boldsymbol{y}_k^{(2)}) \,, \tag{11.29}$$

$$\boldsymbol{R}_k^{(*)} = \boldsymbol{S}_k^T \boldsymbol{V}_k \max(\boldsymbol{D}_k, \boldsymbol{I}) \boldsymbol{V}_k^T \boldsymbol{S}_k \,, \tag{11.30}$$

where $\boldsymbol{S}_k = |\boldsymbol{R}_k^{(2)}|^{1/2}$, and $\max(\boldsymbol{A}, \boldsymbol{B})$ denotes the element-by-element maximum of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\boldsymbol{V}_k$ and $\boldsymbol{D}_k$ are, respectively, the matrices of the eigenvectors and eigenvalues of $(\boldsymbol{S}_k^{-1})^T \boldsymbol{R}_k^{(1)} \boldsymbol{S}_k^{-1}$.

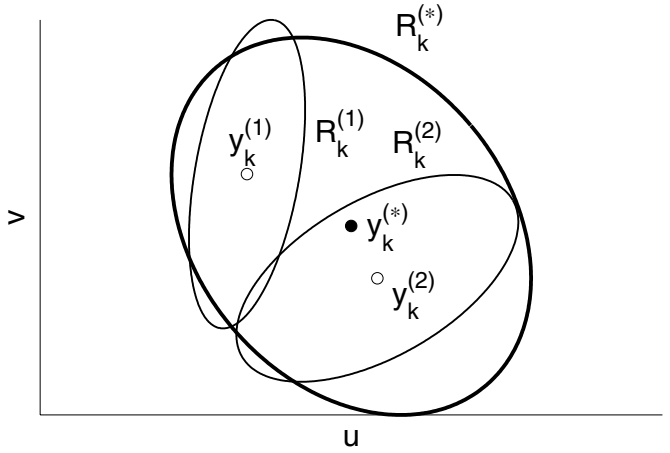

Fig. 11.5. Illustrates the covariance union of two observations $O_1 = (\boldsymbol{y}_k^{(1)}, \boldsymbol{R}_k^{(1)})$ and $O_2 = (\boldsymbol{y}_k^{(2)}, \boldsymbol{R}_k^{(2)})$ in two dimensions (u, v). The observation means, $\boldsymbol{y}_k^{(1)}$ and $\boldsymbol{y}_k^{(2)}$, are shown by small open circles and the observation covariances, $\boldsymbol{R}_k^{(1)}$ and $\boldsymbol{R}_k^{(2)}$, are shown by two ellipses drawn with thin solid lines. The mean, $\boldsymbol{y}_k^{(*)}$, and covariance, $\boldsymbol{R}_k^{(*)}$, of the covariance union are shown, respectively, by a small filled circle and an ellipse drawn with a thick solid line. The CU is calculated using (11.29)-(11.30).

Probabilistic Data Association Filter

The NN, SN and CU filters work by choosing a *single* validated measurement $\boldsymbol{y}_k^{(*)}$ [5] which is then used to update the state vector $\boldsymbol{\theta}_k$. However, the probabilistic data association (PDA) filter works differently by using *all* of the validated measurements, $\boldsymbol{y}_k^{(m)}, m \in \{1, 2, \ldots, m_k\}$, and their probabilities $\beta^{(m)}$ [6] to update $\boldsymbol{\theta}_k$. See [13, 24] for a detailed discussion concerning the PDA filter.

11.3.3 Model Inaccuracies

In the KF we model the system using a linear Gaussian process model (11.12) and a linear Gaussian measurement model (11.13). In practice most systems can never be perfectly modeled and this will set a lower limit on the final performance that can be obtained. However, even when the conditions are far from Gaussian, the KF can often give sensible results by giving more weight to the input measurements, i. e. by increasing the processes noise covariance $\boldsymbol{Q}$.

The following example illustrates an important case in which there is a clear mismatch between the assumed process model and the true system dynamics.

Example 11.7. Maneuvering Target [13, 24]. We track a target moving in one-dimension with a nominally constant velocity which is subject to stochastic accelerations. The corresponding process model (see Ex. 11.3) is

$$\begin{pmatrix} \theta_k \\ \dot{\theta}_k \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{k-1} \\ \dot{\theta}_{k-1} \end{pmatrix} + \boldsymbol{v}_{k-1} \ ,$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ and $\boldsymbol{Q} = \sigma^2 \begin{pmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{pmatrix}$. If the process model is correct, the KF will provide optimum estimates of the target's position and velocity. However, if the target initiates and sustains a driver-induced maneuver, then the above model is incorrect. This is because a driver-induced maneuver is not well-modeled as a stochastic random variable.

One solution to this problem is to use a maneuver detector [13]. As long as the maneuver detector has not detected a target maneuver we continue to use the above process model. However, once a maneuver has been detected, we should initialize a *new* recursive filter whose process model describes target maneuver. □

[5] In the NN and SN filters we use a "real" measurement as $\boldsymbol{y}_k^{(*)}$ while in the CU filter we use a "pseudo" measurement for $\boldsymbol{y}_k^{(*)}$.

[6] $\beta^{(m)}$ is defined as the probability that $\boldsymbol{y}_k^{(*)}$ is the true measurement.

Table 11.3. One-Dimensional Maneuver Models

Name	α	β	Maneuver F
Constant Position	∞	∞	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
Time Correlated Velocity	> 0	∞	$\begin{pmatrix} 1 & a & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Constant Velocity	0	∞	$\begin{pmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
Time Correlated Acceleration (Singer Model)	0	> 0	$\begin{pmatrix} 1 & T & c \\ 0 & 1 & d \\ 0 & 0 & e \end{pmatrix}$
Constant Acceleration	0	0	$\begin{pmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix}$
Time Correlated Acceleration and Velocity	> 0	> 0	$\begin{pmatrix} 1 & a & c \\ 0 & b & d \\ 0 & 0 & e \end{pmatrix}$

Table 11.3 lists some one-dimensional maneuver models which are used for tracking maneuvering targets [172, 216]. The models given in the table are all derived from a general one-dimensional maneuver model:

$$F = \begin{pmatrix} 1 & a & c \\ 0 & b & d \\ 0 & 0 & e \end{pmatrix} \ , \tag{11.31}$$

in which both the target velocity and acceleration are described by zero-mean first-order Markov processes, where

$$a = \frac{(1 - e^{-\alpha T})}{\alpha} \ , \tag{11.32}$$

$$b = e^{-\alpha T} \ , \tag{11.33}$$

$$c = \frac{1}{2!}T^2 - \frac{\alpha + \beta}{3!}T^3 + \frac{\alpha^2 + \alpha\beta + \beta^2}{4!}T^4 + \dots \ , \tag{11.34}$$

$$d = T - \frac{\alpha + \beta}{2!}T^2 + \frac{\alpha^2 + \alpha\beta + \beta^2}{3!}T^3 + \dots \ , \tag{11.35}$$

$$e = e^{-\beta T} \ . \tag{11.36}$$

and $\alpha = 1/\tau_v$ and $\beta = 1/\tau_a$ are, respectively, the reciprocal of the velocity and acceleration correlation times.

11.3.4 Multi-Target Tracking

An important application of the recursive filter is the tracking of multiple targets. If the targets are widely separated one from the other then we may track the targets using a separate KF for each target using the NN, SN, CU or PDA filters. In most real-world systems, the targets are not sufficiently separated and modified tracking/data association algorithms must be used. A full discussion of these and other issues is given in [13, 24].

11.4 Extensions of the Kalman Filter

The KF assumes a single linear Gaussian process model (11.12) and measurement model (11.13). In practice these models may not adequately represent the truth in which case more sophisticated filtering techniques are necessary. Some standard non-linear filter techniques which involve modifications to the KF are listed in Table 11.4.

Table 11.4. Kalman Filter and its Modifications

Filter	Description
Kalman Filter (KF)	Assumes linear Gaussian process and measurement models.
Robust Kalman Filter	Assumes linear Gaussian process model and a linear Gaussian measurement model which is contaminated with a small fraction of outliers. Robust estimation techniques (Chapt. 10) are used to reduce influence of outliers [49, 257].
Kriged Kalman Filter	KF is combined with spatial Kriging model [193]. See Sect. 11.4.5.
Augmented Kalman Filter	KF in which we use parameterized linear Gaussian process and measurement models. We augment the state vector with the process and measurement model parameters α,β,...,γ.
Extended Kalman Filter (EKF)	First-order local linearization of non-linear Gaussian process and measurement models. See Sect. 11.4.2.
Unscented Kalman Filter (UKF)	UKF uses a deterministic set of weighted points $(\omega_k^{(m)}, \Omega_k^{(m)}, \boldsymbol{\theta}_k^{(m)}), m \in \{1, 2, \ldots, M\}$, to represent the *a priori* pdf $p(\boldsymbol{\theta}_{k-1}\|\boldsymbol{y}_{1:k-1}, I)$. See Sect. 11.4.3.
Switching Kalman Filter	Contains multiple KF's but at each time step k we use only one KF. The selected filter is specified by a switching variable Λ_k. See Sect. 11.4.4.
Mixture Kalman Filter	Similar to the switched KF except that at each time step k we use a collection of multiple KF's.
Ensemble Kalman Filter (EnKF)	Used when the dimensions of $\boldsymbol{\theta}_k$ is too large to permit the conventional calculation of the covariance matrix. In the EnKF we estimate the covariance matrices using an ensemble of trajectories.

11.4.1 Robust Kalman Filter

The KF assumes Gaussian process and measurement noise. The filter is therefore sensitive to outliers (see Chapt. 10) which may be present in the measurement $\boldsymbol{y}_k$. If the $\boldsymbol{y}_k$ are thought to be contaminated by outliers we may limit their effect by gating (see Sect. 11.3.2) or by using a robust KF which is designed to be insensitive to outliers. Cipra and Romera [49] described a simple robust KF which is identical to the conventional KF except we replace the Kalman gain matrix $\boldsymbol{K}_k$ with the following expression:

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k^{1/2}\boldsymbol{W}_k\boldsymbol{R}_k^{1/2})^{-1} \ . \tag{11.37}$$

In (11.37), $\boldsymbol{W}_k = \text{diag}(W_k^{(1)}, W_k^{(2)}, \ldots, W_k^{(M)})$ denotes a set of weights given by

$$W_k^{(m)} = \frac{\psi_H\left([\boldsymbol{R}_k^{-1/2}(\boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{\mu}_{k|k-1})]^{(m)}\right)}{[\boldsymbol{R}_k^{-1/2}(\boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{\mu}_{k|k-1})]^{(m)}} \ , \tag{11.38}$$

where $\boldsymbol{X}^{(m)}$ is the mth component of $\boldsymbol{X}$ and $\psi_H(z)$ is the Huber influence function (see Ex. 11.8) [129].

Example 11.8. Recursive Parameter Estimation for a First-Order Autoregressive Process [49]. Consider a first-order autoregressive process:

$$y_k = \theta y_{k-1} + w_{k-1} \ .$$

We wish to recursively estimate θ given the sequence of measurements $y_{1:k}$ which may be contaminated by outliers. In this case, the corresponding process and measurement equations are:

$$\begin{aligned} \theta_k &= \theta_{k-1} \ , \\ y_k &= y_{k-1}\theta_k + w_k \ , \end{aligned}$$

where $F = 1$, $H = y_{k-1}$, $v_k = 0$ and we use the "Good-and-Bad" model (Sect. 10.3.2) to represent the measurement noise w_k. If α is the expected relative number of outliers, then

$$w_k \sim (1-\alpha)\mathcal{N}(0, \sigma^2) + \alpha\mathcal{N}(0, \beta\sigma^2) \ ,$$

where $\beta >> 1$. The *a posteriori* pdf is $p(\theta_k|y_{1:k}, I) \sim \mathcal{N}(\mu_{k|k}, \Sigma_{k|k})$, where

$$\begin{aligned} \mu_{k|k} &= \mu_{k-1|k-1} + \frac{1}{\sigma}\Sigma_{k|k-1}y_{k-1}\psi_H\left(\frac{\sigma(y_k - y_{k-1}\mu_{k-1|k-1})}{\Sigma_{k|k-1}y_{k-1}^2 + \sigma^2}\right) \ , \\ \Sigma_{k|k} &= \frac{\Sigma_{k|k-1}\sigma^2}{\Sigma_{k|k-1}y_{k-1}^2 + \sigma^2} \ , \\ \psi_H(z) &= \begin{cases} z & \text{if } |z| \le c \ , \\ cz/|z| & \text{otherwise} \ . \end{cases} \end{aligned}$$

The recommended choice of c is the α-quantile of $\mathcal{N}(0,1)$ e. g. $c = 1.645$ for an assumed 5% contamination. □

11.4.2 Extended Kalman Filter

In the extended Kalman filter (EKF) we apply the KF to non-linear process and measurement models by linearizing the models. We assume a non-linear Gaussian process model and a non-linear measurement model:

$$\boldsymbol{\theta}_k = \boldsymbol{f}_k(\boldsymbol{\theta}_{k-1}) + \boldsymbol{v}_{k-1} \ , \tag{11.39}$$

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{\theta}_k) + \boldsymbol{w}_k \ , \tag{11.40}$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{k-1})$ and $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ are independent Gaussian noise sources. We linearize (11.39) and (11.40) by performing the following Taylor series expansions:

$$\boldsymbol{\theta}_k \approx \boldsymbol{f}_k(\boldsymbol{\mu}_{k-1|k-1}) + \widetilde{\boldsymbol{F}}_k(\boldsymbol{\theta}_{k-1} - \boldsymbol{\mu}_{k-1|k-1}) + \boldsymbol{v}_{k-1} \ , \tag{11.41}$$

$$\boldsymbol{y}_k \approx \boldsymbol{h}_k(\boldsymbol{\mu}_{k|k-1}) + \widetilde{\boldsymbol{H}}_k(\boldsymbol{\theta}_k - \boldsymbol{\mu}_{k|k-1}) + \boldsymbol{w}_k \ , \tag{11.42}$$

where

$$\widetilde{\boldsymbol{F}}_k = \left. \frac{\partial \boldsymbol{f}_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\mu}_{k-1|k-1}} \ , \tag{11.43}$$

$$\widetilde{\boldsymbol{H}}_k = \left. \frac{\partial \boldsymbol{h}_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\mu}_{k|k-1}} \ . \tag{11.44}$$

By substituting the terms $\boldsymbol{\alpha}_k = \boldsymbol{f}_k(\boldsymbol{\mu}_{k-1|k-1}) - \widetilde{\boldsymbol{F}}_k \boldsymbol{\mu}_{k-1|k-1}$ and $\boldsymbol{\beta}_k = \boldsymbol{h}_k(\boldsymbol{\mu}_{k|k-1}) - \widetilde{\boldsymbol{H}}_k \boldsymbol{\mu}_{k|k-1}$ into (11.41) and (11.42) we obtain the following linear Gaussian models:

$$\boldsymbol{\theta}_k = \boldsymbol{\alpha}_k + \widetilde{\boldsymbol{F}}_k \boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ , \tag{11.45}$$

$$\boldsymbol{y}_k = \boldsymbol{\beta}_k + \widetilde{\boldsymbol{H}}_k \boldsymbol{\theta}_k + \boldsymbol{w}_k \ , \tag{11.46}$$

which may be filtered using the standard KF.

Eqs. (11.47)-(11.53) are the corresponding EKF equations.

$$p(\boldsymbol{\theta}_k | \boldsymbol{y}_{1:k-1}, I) \approx \mathcal{N}(\boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1}) \ , \tag{11.47}$$

$$p(\boldsymbol{\theta}_k | \boldsymbol{y}_{1:k}, I) \approx \mathcal{N}(\boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k}) \ , \tag{11.48}$$

where

$$\boldsymbol{\mu}_{k|k-1} = \boldsymbol{f}_k(\boldsymbol{\mu}_{k-1|k-1}) \ , \tag{11.49}$$

$$\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{Q}_{k-1} + \widetilde{\boldsymbol{F}}_k \boldsymbol{\Sigma}_{k-1|k-1} \widetilde{\boldsymbol{F}}_k^T \ , \tag{11.50}$$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{h}_k(\boldsymbol{\mu}_{k|k-1})) \ , \tag{11.51}$$

$$\boldsymbol{\Sigma}_{k|k} = \boldsymbol{\Sigma}_{k|k-1} - \boldsymbol{K}_k \widetilde{\boldsymbol{H}}_k \boldsymbol{\Sigma}_{k|k-1} \ , \tag{11.52}$$

and

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1}\widetilde{\boldsymbol{H}}_k^T(\widetilde{\boldsymbol{H}}_k\boldsymbol{\Sigma}_{k|k-1}\widetilde{\boldsymbol{H}}_k^T + \boldsymbol{R}_k)^{-1} . \tag{11.53}$$

The EKF is widely used since it enables us to use the KF framework to perform sequential Bayesian inference on non-linear Gaussian systems. Its main drawbacks are that it requires the *analytical* computation of $\widetilde{\boldsymbol{F}}_k$ and $\widetilde{\boldsymbol{H}}_k$ and it is only accurate to first-order.

11.4.3 Unscented Kalman Filter

In the unscented Kalman filter (UKF) we retain the Gaussian approximations in (11.47) and (11.48) but do not perform a Taylor series expansion on the process and measurement models $f_{k-1}(\boldsymbol{\theta}_{k-1})$ and $h_k(\boldsymbol{\theta}_k)$. Instead $\boldsymbol{\theta}_{k-1}$ and $\boldsymbol{\theta}_k$ are specified using a minimal set of carefully chosen sample points. These points (known as sigma-points) completely capture the true mean and covariance of $\boldsymbol{\theta}_{k-1}$ and $\boldsymbol{\theta}_k$ and when propagated, respectively, through the non-linear process and measurement models, they capture the *a posteriori* means and covariances to the third order [7]. The unscented Kalman filter is a straightforward extension of the UT to the recursive estimation in (11.39) and (11.40), where the state random variable is redefined as the concatenation of the original state $\boldsymbol{\theta}$ and the noise variables $\boldsymbol{v}$ and $\boldsymbol{w}$: $\boldsymbol{\phi}_k = (\boldsymbol{\theta}_k^T, \boldsymbol{v}_k^T, \boldsymbol{w}_k^T)^T$. The UT sigma point selection is applied to the new augmented state $\boldsymbol{\phi}_{k-1}$ to calculate the corresponding sigma points. The final UKF equations are:

$$\boldsymbol{\mu}_{k|k-1} = \sum_{m=1}^{2M} \omega_{k-1}^{(m)} \hat{\boldsymbol{\theta}}_k^{(m)} , \tag{11.54}$$

$$\boldsymbol{\Sigma}_{k|k-1} = \sum_{m=0}^{2M} \Omega_{k-1}^{(m)} (\hat{\boldsymbol{\theta}}_k^{(m)} - \boldsymbol{\mu}_{k|k-1})(\hat{\boldsymbol{\theta}}_k^{(m)} - \boldsymbol{\mu}_{k|k-1})^T , \tag{11.55}$$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \boldsymbol{K}_k(\hat{\boldsymbol{y}}_k - \bar{\boldsymbol{y}}_k^{(m)}) , \tag{11.56}$$

$$\boldsymbol{\Sigma}_{k|k} = \boldsymbol{\Sigma}_{k|k-1} + \boldsymbol{K}_k \boldsymbol{S}_k \boldsymbol{K}_k^T , \tag{11.57}$$

[7] This is true for any non-linear transformation and is the basis of the *unscented transformation* (UT) [144]. The UT is a method for calculating the statistics of a random variable which undergoes a non-linear transformation. Consider propagating a M-dimensional random variable $\boldsymbol{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(M)})^T$ (mean vector $\bar{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{R}$) though a non-linear function $\boldsymbol{y} = g(\boldsymbol{x})$. Then the mean and covariance of $\boldsymbol{y}$ are:

$$\bar{\boldsymbol{y}} \approx \sum_{m=0}^{2M} \omega^{(m)} y^{(m)} ,$$

$$\boldsymbol{S} \approx \sum_{m=0}^{2M} \Omega^{(m)} (\boldsymbol{y}^{(m)} - \bar{\boldsymbol{y}})(\boldsymbol{y}^{(m)} - \bar{\boldsymbol{y}})^T ,$$

where $\boldsymbol{x}^{(m)}, m \in \{0, 1, \ldots, 2M\}$ are a set of deterministically selected sigma-points with weights $\omega^{(m)}$ and $\Omega^{(m)}$ and $\boldsymbol{y}^{(m)} = g(\boldsymbol{x}^{(m)})$.

where

$$\hat{\boldsymbol{\theta}}_{k-1}^{(m)} = f_{k-1}(\boldsymbol{\theta}_{k-1}^{(m)}) + \boldsymbol{v}_{k-1} \ , \tag{11.58}$$

$$\hat{\boldsymbol{y}}_{k}^{(m)} = h_{k}(\boldsymbol{\theta}_{k-1}^{(m)}) + \boldsymbol{w}_{k} \ , \tag{11.59}$$

$$\bar{\boldsymbol{y}}_k = \sum_{m=0}^{2M} \omega_{k-1}^{(m)} \hat{\boldsymbol{y}}_k^{(m)} \ , \tag{11.60}$$

$$\boldsymbol{S}_k = \sum_{m=0}^{2M} \Omega_{k-1}^{(m)} (\hat{\boldsymbol{y}}_k^{(m)} - \bar{\boldsymbol{y}}_k)(\hat{\boldsymbol{y}}_k^{(m)} - \bar{\boldsymbol{y}}_k)^T \ , \tag{11.61}$$

and

$$\boldsymbol{K}_k = \sum_{m=0}^{2M} \Omega_{k-1}^{(m)} (\hat{\boldsymbol{\theta}}_k^{(m)} - \boldsymbol{\mu}_{k|k-1})(\hat{\boldsymbol{y}}_k^{(m)} - \bar{\boldsymbol{y}}_k)^T \boldsymbol{S}_k^{-1} \ . \tag{11.62}$$

The *sigma*-points $\boldsymbol{\theta}^{(m)}, m \in \{0, 1, \ldots, 2M\}$, and their weights, $\omega^{(m)}$ and $\Omega^{(m)}$, are selected in deterministic fashion (see Table 11.5).

11.4.4 Switching Kalman Filter

In the switching Kalman filter [192, 222] we generalize (11.12) and (11.13) by assuming that $\boldsymbol{F}_k$, $\boldsymbol{H}_k$, $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ are functions of an indicator, or switching, variable Λ_k: $\boldsymbol{F}_k = \boldsymbol{F}(\Lambda_k)$, $\boldsymbol{H}_k = \boldsymbol{H}(\Lambda_k)$, $\boldsymbol{Q}_k = \boldsymbol{Q}(\Lambda_k)$ and $\boldsymbol{R}_k = \boldsymbol{R}(\Lambda_k)$. In this case, (11.12) and (11.13) become

$$\boldsymbol{\theta}_k = \boldsymbol{F}(\Lambda_k)\boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ , \tag{11.63}$$

$$\boldsymbol{y}_k = \boldsymbol{H}(\Lambda_k)\boldsymbol{\theta}_k + \boldsymbol{w}_k \ , \tag{11.64}$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}(\Lambda_{k-1}))$ and $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}(\Lambda_k))$ are independent Gaussian noise sources.

We often suppose that Λ_k obeys a first-order Markov process, where

$$P(\Lambda_k = \lambda_n | I) = \sum_{m=1}^{M} P(\Lambda_k = \lambda_n | \Lambda_{k-1} = \lambda_m, I) P(\Lambda_{k-1} = \lambda_m | I) \ , \tag{11.65}$$

Table 11.5. Sigma-Points and their Weights

Variable	Formula
Sigma-Points $\boldsymbol{\theta}^{(m)}$	$\boldsymbol{\theta}^{(0)} = \boldsymbol{\mu}$; $\boldsymbol{\theta}^{(m)} = \boldsymbol{\mu} + [\sqrt{(n+\lambda)\boldsymbol{\Sigma}}]_m$; $\boldsymbol{\theta}^{(m+M)} = \boldsymbol{\mu} - [\sqrt{(n+\lambda)\boldsymbol{\Sigma}}]_m, m \in \{1, 2, \ldots, M\}$; where $\lambda = \alpha^2(M+K)$ and $[\sqrt{(n+\lambda)\boldsymbol{\Sigma}}]_m$ is the mth column of the matrix square root of $(n+\lambda)\Sigma$.
Weights $\omega^{(m)}$	$\omega^{(0)} = \lambda/(M+\lambda), \omega^{(m)} = \lambda/(2(M+\lambda)), m \in \{1, 2 \ldots, 2M\}$.
Weights $\Omega^{(m)}$	$\Omega^{(0)} = \lambda/(M+\lambda) + (1-\alpha^2+\beta), \Omega^{(m)} = \lambda/(2(M+\lambda)), m \in \{1, 2, \ldots, 2M\}$.

and Λ_k is limited to M distinct values $\{\lambda_1, \lambda_2, \ldots, \lambda_M\}$. In this case, at a given time step k, the *a posteriori* probability density $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$ is an exponentially growing mixture of M^k Gaussian pdf's, each pdf corresponding to a different possible history of the switching variable [8].

In order to limit the exponential growth in the number of Gaussian pdf's we may proceed as follows. At each time step k we "collapse" the mixture of M^k Gaussians onto a mixture M^r Gaussians where $r < k$. This is called the *Generalized Pseudo Bayesian* (GPB) approximation of order r. The following example illustrates the action of the GPB approximation.

Example 11.9. Generalized Pseudo Bayesian Approximation. We consider the GPB approximation for $r = 1$. In this case, we approximate a mixture of Gaussians with a single Gaussian by matching the mean and covariance of the mixture to that of a single Gaussian, i. e.

$$\begin{aligned} p(\boldsymbol{\theta}) &= \sum_{m=1}^{M} w^{(m)} \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)}) \ , \\ &\approx \mathcal{N}(\boldsymbol{\theta}|\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}) \ , \end{aligned}$$

where

$$\begin{aligned} \bar{\boldsymbol{\mu}} &= \sum_{m=1}^{M} w^{(m)} \boldsymbol{\mu}^{(m)} \ , \\ \bar{\boldsymbol{\Sigma}} &= \sum_{m=1}^{M} w^{(m)} \boldsymbol{\Sigma}^{(m)} \ , \end{aligned}$$

where $\boldsymbol{\Sigma}^{(m)} = (\boldsymbol{\mu}^{(m)} - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}^{(m)} - \bar{\boldsymbol{\mu}})^T$. □

The following example illustrates a switched KF using a GPB approximation of order one.

Example 11.10. The Lainiotis-Kalman Equations. [277]. We assume the process and measurement equations:

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{F}_{k-1}\boldsymbol{\theta}_k + \boldsymbol{v}_{k-1} \ , \\ \boldsymbol{y}_k &= \boldsymbol{H}_k(\Lambda_k)\boldsymbol{\theta}_k + \boldsymbol{w}_k \ , \end{aligned}$$

where the indicator Λ is limited to a finite set of values $\Lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_M\}$, and $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{R}_k(\Lambda))$. Then, we approximate the *a posteriori* pdf, $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$,

[8] At time step $k = 1$ we do not know which KF generated the observation $\boldsymbol{y}_1$, so a Gaussian pdf for each KF must be maintained, giving a mixture of M Gaussian pdf's to describe $\boldsymbol{\theta}_{1|1}$. At time step $k = 2$, which KF generated the observation $\boldsymbol{y}_2$ is also unknown. Because the KF for time step $k = 1$ is also unknown, we require M^2 Gaussian pdf's to describe $\boldsymbol{\theta}_{2|2}$. Consequently at time step k, we require M^k Gaussian pdf's to describe $\boldsymbol{\theta}_{k|k}$.

using a single Gaussian pdf,

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I) \approx \mathcal{N}(\widetilde{\boldsymbol{\mu}}_{k|k}, \widetilde{\boldsymbol{\Sigma}}_{k|k}) \ , \tag{11.66}$$

whose parameters, $\widetilde{\boldsymbol{\mu}}_{k|k}$ and $\widetilde{\boldsymbol{\Sigma}}_{k|k}$, are given by the *Lainiotis-Kalman* equations:

$$\begin{aligned}
\widetilde{\boldsymbol{\mu}}_{k|k} &= \sum_{m=1}^{M} p(\Lambda_k = \lambda_m|\boldsymbol{y}_{1:k}, I)\boldsymbol{\mu}_{k|k}(\Lambda_k = \lambda_m) \ , \\
\widetilde{\boldsymbol{\Sigma}}_{k|k} &= \sum_{m=1}^{M} p(\Lambda_k = \lambda_m|\boldsymbol{y}_{1:k}, I)[\boldsymbol{\Sigma}_{k|k}(\Lambda_k = \lambda_m)(\boldsymbol{\mu}_{k|k}(\Lambda_k = \lambda_m) \\
&\qquad - \widetilde{\boldsymbol{\mu}}_{k|k}(\Lambda_k = \lambda_m)) \times (\boldsymbol{\mu}_{k|k}(\Lambda_k = \lambda_m) - \widetilde{\boldsymbol{\mu}}_{k|k}(\Lambda_k = \lambda_m))^T] \ ,
\end{aligned}$$

where $\boldsymbol{\mu}_{k|k}^{(m)} \equiv \boldsymbol{\mu}_{k|k}(\Lambda_k = \lambda_m)$ and $\boldsymbol{\Sigma}_{k|k}^{(m)} \equiv \boldsymbol{\Sigma}_{k|k}(\Lambda_k = \lambda_m)$ are the $\boldsymbol{\mu}_{k|k}$ and $\boldsymbol{\Sigma}_{k|k}$ values as computed by the KF assuming $\Lambda_k = \lambda_m$ and

$$P(\Lambda = \lambda_m|\boldsymbol{y}_{1:k}, I) = \frac{p(\boldsymbol{y}_{1:k}|\Lambda = \lambda_m, I)P(\Lambda = \lambda_m|I)}{p(\boldsymbol{y}_{1:k}|I)} \ ,$$

is the *a posteriori* probability of $\Lambda_k = \lambda_m$ given $\boldsymbol{y}_{1:k}$. □

11.4.5 Kriged Kalman Filter

The Kriged Kalman filter (KKF) is a particular type of Kalman filter designed for modeling the evolution of a spatial-temporal field in time (see Chapt. 4).

Let $y(\boldsymbol{u}, t)$ denote the value of a given spatial-temporal field at the spatial position $\boldsymbol{u}$ and at time step k. At each time step k we model the spatial field as a continuous function $\boldsymbol{h}_k(\boldsymbol{u})$. We assume that at each time step k we may write $\boldsymbol{h}_k(\boldsymbol{u})$ as a linear combination of fixed spatial basis fuctions $h^{(m)}(\boldsymbol{u}), m \in \{1, 2, \ldots, M\}$:

$$\boldsymbol{h}_k(\boldsymbol{u}) = \sum_{m=1}^{M} \alpha_k^{(m)} h^{(m)}(\boldsymbol{u}) \ , \tag{11.67}$$

where we have attached a subscript k to the weights $\alpha^{(m)}$ in (11.67) to emphasize that the weights evolve in time. We may use (11.67) to define the state of the system as a vector $\boldsymbol{\alpha}_k$:

$$\boldsymbol{\alpha}_k = (\alpha_k^{(1)}, \alpha_k^{(2)}, \ldots, \alpha_k^{(M)})^T \ . \tag{11.68}$$

The state vector $\boldsymbol{\alpha}_k$ is then recursively filtered using the Kalman filter. Together the standard KF equations (11.14–11.20) and (11.67–11.68) define the Kriged Kalman filter.

11.5 Particle Filter

The techniques discussed in Sect. 11.4 all relax some of the Gaussian linear assumptions which are present in the KF. However, in many practical applications these techniques are still not sufficiently accurately to model all of the aspects of the *a posteriori* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}, I)$. If this happens then we may use a different type of filter based on a Monte Carlo approximation. This filter is known as a *particle filter* filter [10].

The key idea underlying the particle filter is to represent the density $p(\boldsymbol{\theta}_k|\boldsymbol{y}_k, I)$ by a set of N weighted random samples, or particles, which constitutes a discrete approximation of the pdf. In this regard the particle filter is similar to the sigma filter. The key difference being that in the former we may use several thousand particles while in the sigma filter we use a very small number of dterministically selected particles.

11.6 Multi-Sensor Multi-Temporal Data Fusion

Until now we have considered multi-temporal data fusion in which use a recursive filter to fuse together a sequence of measurements $\boldsymbol{y}_{1:k}$ made using a single sensor S. We now extend this analysis and consider multi-sensor multi-temporal data fusion in which we fuse together several sequences of measurements $\boldsymbol{y}_{1:k}^{(m)}, m \in \{1, 2, \ldots, M\}$, each sequence $\boldsymbol{y}_{1:k}^{(m)}$ being made with a different sensor $S^{(m)}$. We start with the simplest method of multi-sensor multi-temporal data fusion which is known as *measurement fusion*.

11.6.1 Measurement Fusion

In measurement fusion we place all the measurements $\boldsymbol{y}_k^{(m)}, m \in \{1, 2, \ldots, M\}$, obtained at time step k into a single augmented measurement vector $\widetilde{\boldsymbol{y}}_k = ((\boldsymbol{y}_k^{(1)})^T, (\boldsymbol{y}_k^{(2)})^T, \ldots, (\boldsymbol{y}_k^{(M)})^T)^T$. We then fuse the $\widetilde{\boldsymbol{y}}_{1:k} = (\widetilde{\boldsymbol{y}}_1, \widetilde{\boldsymbol{y}}_2, \ldots, \widetilde{\boldsymbol{y}}_k)^T$ using a single recursive filter. In the following example we illustrate the technique using a Kalman filter.

Example 11.11. Multi-Sensor Multi-Temporal Data Fusion: Measurement Fusion [97]. We consider the tracking of a single target. Let $\boldsymbol{\theta}_k$ denote the filtered state of the target at time step k and $\boldsymbol{y}_k^{(m)}$ denotes the corresponding measured position of the target as recorded by two sensors $S^{(1)}$ and $S^{(2)}$. The corresponding process and measurement equations are:

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{F}_k\boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ , \\ \boldsymbol{y}_k^{(1)} &= \boldsymbol{H}_k^{(1)}\boldsymbol{\theta}_k + \boldsymbol{w}_k^{(1)} \ , \\ \boldsymbol{y}_k^{(2)} &= \boldsymbol{H}_k^{(2)}\boldsymbol{\theta}_k + \boldsymbol{w}_k^{(2)} \ , \end{aligned}$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_k)$, $\boldsymbol{w}_k^{(1)} \sim \mathcal{N}(0, \boldsymbol{R}_k^{(1)})$ and $\boldsymbol{w}_k^{(2)} \sim \mathcal{N}(0, \boldsymbol{R}_k^{(2)})$.

We estimate $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}^{(1:2)}, I) \approx \mathcal{N}(\boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k})$ using the KF equations:

$$\begin{aligned}
\boldsymbol{\mu}_{k|k-1} &= \boldsymbol{F}_k \boldsymbol{\mu}_{k-1|k-1} \ , \\
\boldsymbol{\Sigma}_{k|k-1} &= \boldsymbol{Q}_{k-1} + \boldsymbol{F}_k \boldsymbol{\Sigma}_{k-1|k-1} \boldsymbol{F}_k^T \ , \\
\boldsymbol{\mu}_{k|k} &= \boldsymbol{\mu}_{k|k-1} + \widetilde{\boldsymbol{K}}_k (\boldsymbol{y}_k - \widetilde{\boldsymbol{H}}_k \boldsymbol{\mu}_{k|k-1}) \ , \\
\boldsymbol{\Sigma}_{k|k} &= \boldsymbol{\Sigma}_{k|k-1} - \widetilde{\boldsymbol{K}}_k \widetilde{\boldsymbol{H}}_k \boldsymbol{\Sigma}_{k|k-1} \ ,
\end{aligned}$$

where

$$\begin{aligned}
\widetilde{\boldsymbol{K}}_k &= \boldsymbol{\Sigma}_{k|k-1} \widetilde{\boldsymbol{H}}_k^T (\widetilde{\boldsymbol{H}}_k \boldsymbol{\Sigma}_{k|k-1} \widetilde{\boldsymbol{H}}_k^T + \widetilde{\boldsymbol{R}}_k)^{-1} \ , \\
\widetilde{\boldsymbol{H}}_k &= \begin{pmatrix} \boldsymbol{H}_k^{(1)} \\ \boldsymbol{H}_k^{(2)} \end{pmatrix} \ , \\
\widetilde{\boldsymbol{w}}_k &= \begin{pmatrix} \boldsymbol{w}_k^{(1)} \\ \boldsymbol{w}_k^{(2)} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{0}, \widetilde{R}_k) \ ,
\end{aligned}$$

and

$$\widetilde{\boldsymbol{R}}_k = \begin{pmatrix} \boldsymbol{R}_k^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_k^{(2)} \end{pmatrix} \ .$$

Fig. 11.6 is a graphical illustration of the process of measurement fusion for the two sensors $S^{(1)}$ and $S^{(2)}$. □

Although theoretically optimum, measurement fusion requires a centralized architecture in which all the measurements $\boldsymbol{y}_k^{(m)}, m \in \{1, 2, \ldots, M\}$, are

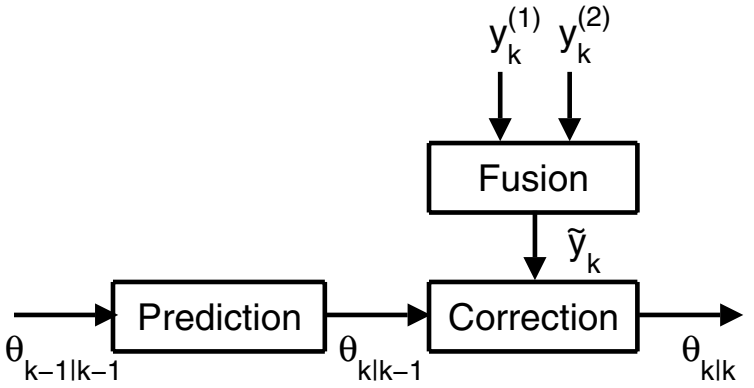

Fig. 11.6. Illustrates the measurement fusion process for two measurement sequencies $\boldsymbol{y}_{1:k}^{(1)}$ and $\boldsymbol{y}_{1:k}^{(2)}$. The individual measurements are placed in an augmented measurement sequence $\widetilde{\boldsymbol{y}}_{1:k} = (\widetilde{\boldsymbol{y}}_1, \widetilde{\boldsymbol{y}}_2, \ldots, \widetilde{\boldsymbol{y}}_k)$, where $\widetilde{\boldsymbol{y}}_l = ((\boldsymbol{y}_l^{(1)})^T, (\boldsymbol{y}_l^{(2)})^T)^T$. The augmented measurement vector is then fused using a single KF.

transmitted to a single fusion node. The difficulties and drawbacks associated with this architecture were described in Sect. 3.4.1. The most serious drawback associated with the centralized architecture is its sensitivity to errors involved in forming a common representational format, i. e. in spatial and temporal alignment of the sensors $S^{(m)}$. An alternative method of multi-sensor multi-temporal data fusion is to separately filter the measurements received from each sensor and then fuse the results together. This is known as track-to-track fusion and is considered in the next section. Its main advantage is that it is relatively insensitive to the errors in the spatial and temporal alignment of the sensors.

11.6.2 Track-to-Track Fusion

In track-to-track fusion we first perform multi-temporal fusion using M recursive filters, in which the mth filter fuses the measurements $\boldsymbol{y}_{1:k}^{(m)} = \{y_1^{(m)}, y_2^{(m)}, \ldots, y_k^{(m)}\}$ made by the mth sensor $S^{(m)}$. Let $p(\boldsymbol{\theta}_k^{(m)}|\boldsymbol{y}_{1:k}^{(m)}, I)$ be the corresponding *a posteriori* pdf. Then, at each time step k, we fuse together the M *a posteriori* pdf's $p(\boldsymbol{\theta}_k^{(m)}|\boldsymbol{y}_{1:k}^{(m)}, I), m \in \{1, 2, \ldots, M\}$, using a linear least squares (minimum mean square error) estimate (see Sect. 9.5). Track-to-track fusion uses a hierarchical architecture (Sect. 3.4.3) containing M local fusion nodes and one central fusion node (Fig. 11.7). In general track-to-track fusion has a lower performance [38, 97] than the corresponding measurement fusion algorthm (Sect. 11.6.1). The reason for the reduction in performance is that the track-to-track fusion is performed using a *linear* estimate.

The following example illustrates track-to-track fusion for two sensors $S^{(1)}$ and $S^{(2)}$ using the KF [12].

Example 11.12. Multi-Sensor Multi-Temporal Data Fusion: Track-to-Track Fusion [12, 277]. We re-analyze Ex. 11.11 using the Bar-Shalom-Campo track-to-track fusion algorithm of [12]. If $\boldsymbol{\theta}_k$ denotes the filtered state of the target at time step k and $\boldsymbol{y}_k^{(m)}$ denotes the measured position of the target at time step k as recorded by the sensor $S^{(m)}$, then the corresponding process and measurement model equations are:

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{F}_k \boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \ , \\ \boldsymbol{y}_k^{(m)} &= \boldsymbol{H}_k^{(m)} \boldsymbol{\theta}_k + \boldsymbol{w}_k^{(m)} \ , \end{aligned}$$

where $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_k)$ and $\boldsymbol{w}_k^{(m)} \sim \mathcal{N}(0, \boldsymbol{R}_k^{(m)})$.

Let $\boldsymbol{\theta}_k^{(m)} \sim \mathcal{N}(\boldsymbol{\mu}_{k|k}^{(m)}, \boldsymbol{\Sigma}_{k|k}^{(m)})$ denote the KF estimate of $\boldsymbol{\theta}_k$ given the sensor measurements $\boldsymbol{y}_{1:k}^{(m)}$, where $\boldsymbol{\mu}_{k|k}^{(m)}$ and $\boldsymbol{\Sigma}_{k|k}^{(m)}$ are calculated using the KF equations (11.14-11.20). For $M = 2$ we fuse together the two (correlated) estimates

$\boldsymbol{\mu}_{k|k}^{(m)}, m \in \{1,2\}$, using the Bar-Shalom-Campo formula:

$$\widetilde{\boldsymbol{\theta}}_{k|k} = \boldsymbol{\mu}_{k|k}^{(1)}(\boldsymbol{\Sigma}_{k|k}^{(22)} - \boldsymbol{\Sigma}_{k|k}^{(21)})(\boldsymbol{\Sigma}_{k|k}^{(11)} + \boldsymbol{\Sigma}_{k|k}^{(22)} - \boldsymbol{\Sigma}_{k|k}^{(12)} - \boldsymbol{\Sigma}_{k|k}^{(21)})^{-1} + \boldsymbol{\mu}_{k|k}^{(2)}(\boldsymbol{\Sigma}_{k|k}^{(11)} - \boldsymbol{\Sigma}_{k|k}^{(12)})(\boldsymbol{\Sigma}_{k|k}^{(11)} + \boldsymbol{\Sigma}_{k|k}^{(22)} - \boldsymbol{\Sigma}_{k|k}^{(12)} - \boldsymbol{\Sigma}_{k|k}^{(21)})^{-1}\boldsymbol{\mu}_{k|k}^{(2)} ,$$

where $\widetilde{\boldsymbol{\mu}}_{k|k}$ is the optimum linear estimate, $\boldsymbol{\Sigma}_{k|k}^{(mm)} \equiv \boldsymbol{\Sigma}_{k|k}^{(m)}$ is the covariance matrix of $\boldsymbol{y}_k^{(m)}$ and $\boldsymbol{\Sigma}_{k|k}^{(mn)}$ is the cross-covariance matrix between $\boldsymbol{y}_k^{(m)}$ and $\boldsymbol{y}_k^{(n)}$.

Fig. 11.7 shows the track-to-track fusion of observations received from two sensors $S^{(1)}$ and $S^{(2)}$ (see also Ex. 3.12). □

Recently the Bar-Shalom-Campo equations have been extended to the case when $M > 2$ [9]. In the following example we give the equations for track-to-track fusion using $M > 2$ sensors.

Example 11.13. Track-to-Track Fusion using the Generalized Millman Eequations [277]. Recently the Bar-Shalom-Campo equations have been extended to more than two sensors. We reconsider Ex. 11.12 except this time we allow $M > 2$. In this case, we contruct the global (subpotimal) estimate $\widetilde{\boldsymbol{\mu}}_{k|k}$ from the local KF estimates $\boldsymbol{\mu}_{k|k}^{(m)}, m \in \{1, 2, \ldots, M\}$, using the generalized Millman formula (GMF):

$$\widetilde{\boldsymbol{\mu}}_{k|k} = \sum_{m=1}^{M} \boldsymbol{C}_k^{(m)} \boldsymbol{\mu}_{k|k}^{(m)} ,$$

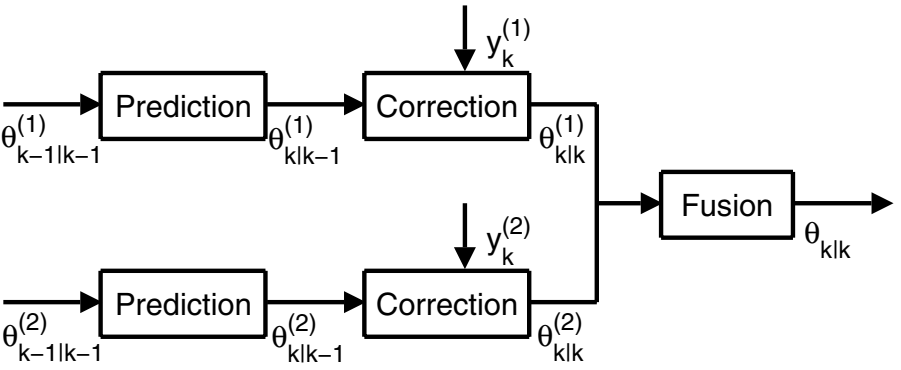

Fig. 11.7. Illustrates the track-to-track fusion process for the measurements $\boldsymbol{y}_{1:k}^{(1)}$ and $\boldsymbol{y}_{1:k}^{(2)}$ obtained from two sensors $S^{(1)}$ and $S^{(2)}$. Each sequence of measurements are separately fused together using a KF. At each time step k the outputs $N(\boldsymbol{\mu}_{k|k}^{(1)}, \boldsymbol{\Sigma}_{k|k}^{(1)})$ and $N(\boldsymbol{\mu}_{k|k}^{(2)}, \boldsymbol{\Sigma}_{k|k}^{(2)})$ are then fused together using the Bar-Shalom-Campo formula.

[9] See also Sect. 9.7.

where the time-varying matrices $\boldsymbol{C}_k^{(m)}, m \in \{1, 2, \ldots, M\}$, are given by

$$\begin{aligned}
\sum_{m=1}^{M-1} \boldsymbol{C}_k^{(m)}(\boldsymbol{S}_{k|k}^{(m,1)} - \boldsymbol{S}_{k|k}^{(m,M)}) + \boldsymbol{C}_k^{(M)}(\boldsymbol{S}_{k|k}^{(M,1)} - \boldsymbol{\Sigma}_{k|k}^{(M,M)}) = 0 \,, \\
\sum_{m=1}^{M-1} \boldsymbol{C}_k^{(m)}(\boldsymbol{S}_{k|k}^{(m,2)} - \boldsymbol{S}_{k|k}^{(m,M)}) + \boldsymbol{C}_k^{(M)}(\boldsymbol{S}_{k|k}^{(M,2)} - \boldsymbol{\Sigma}_{k|k}^{(M,M)}) = 0 \,, \\
\vdots \\
\sum_{m=1}^{M-1} \boldsymbol{C}_k^{(m)}(\boldsymbol{S}_{k|k}^{(m,M-1)} - \boldsymbol{S}_{k|k}^{(m,M)}) + \boldsymbol{C}_k^{(M)}(\boldsymbol{S}_{k|k}^{(M,M-1)} - \boldsymbol{S}_{k|k}^{(M,M)}) = 0 \,,
\end{aligned}$$

where

$$\sum_{m=1}^{M} \boldsymbol{C}_k^{(m)} = \boldsymbol{I} \,,$$

and

$$\boldsymbol{S}_{k|k}^{(m,n)} = \begin{cases} (\boldsymbol{I} - \boldsymbol{K}_k^{(m)} \boldsymbol{H}_k^{(m)}) \boldsymbol{\Sigma}_{k|k-1}^{(m)} & \text{if } n = m \,, \\ (\boldsymbol{I} - \boldsymbol{K}_k^{(m)} \boldsymbol{H}_k^{(m)})(\boldsymbol{F}_k \boldsymbol{S}_{k-1|k-1}^{(m,n)} \boldsymbol{F}_k^T + \boldsymbol{Q}_k) & \\ \quad \times (\boldsymbol{I} - \boldsymbol{K}_k^{(n)} \boldsymbol{H}_k^{(n)})^T & \text{otherwise} \,. \end{cases}$$

□

Track-to-track fusion is widely used and many different variants of the basic scheme have been described. The following example illustrates one such variant.

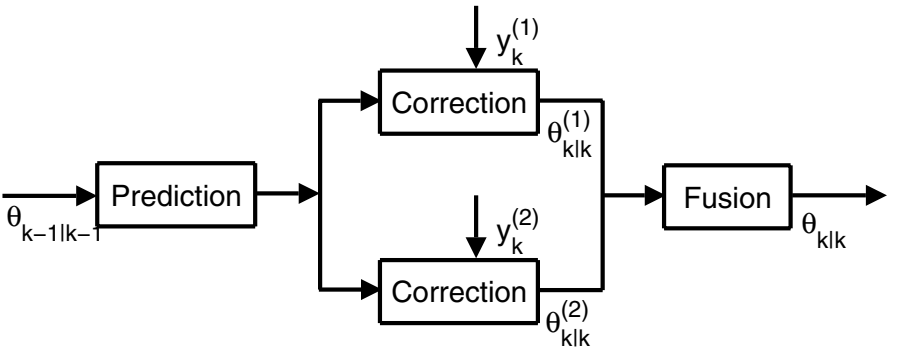

Fig. 11.8. Illustrates the modified track-to-track fusion process for the measurements $\boldsymbol{y}_{1:k}^{(1)}$ and $\boldsymbol{y}_{1:k}^{(2)}$ obtained from two sensors $S^{(1)}$ and $S^{(2)}$. As in track-to-track fusion each sequence of measurements are separately fused together using a KF. However, the KF's use a *common* predictor which is based on the fused output $N(\boldsymbol{\mu}_{k-1|k-1}, \boldsymbol{\Sigma}_{k-1|k-1})$ obtained at the previous time step.

Example 11.14. Modified track-to-track fusion algorithm [97]. In the modified track-to-track fusion algorithm of Gao and Harris [97] we calculate a single predicted pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}^{(1:M)}, I)$ using the *fused* pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}^{(1:M)}, I)$. Fig. 11.8 shows a schematic implementation of the Gao-Harris track-to-track fusion algorithm. □

11.7 Software

KALMTOOL (Kalman Toolbox). This is a matlab toolbox for simulating the Kalman filter.

KFT (Kalman Filter Toolbox). This is a matlab toolbox for simulating the Kalman filter.

UPF (Unscented Particle Filter). This is a matlab toolbox for the unscented particle filter.

11.8 Further Reading

Chen [40] has provided a very readable, comprehensive survey of Bayesian filtering. Additional modern references on Bayesian filters include [10, 196]. References on specific topics in sequential Bayesian inference include: [31] (Kalman filters), [205] (unscented Kalman and particle filters), [10] (particle filters), [13, 24, 215] (multi-target tracking). A useful taxonomy on multi-target tracking is [252].

12

Bayesian Decision Theory

12.1 Introduction

The subject of this chapter, and the one that follows it, is *Bayesian decision theory* and its use in multi-sensor data fusion. To make our discussion concrete we shall concentrate on the pattern recognition problem [137] in which an unknown pattern, or object, O is to be assigned to one of K possible classes $\{c_1, c_2, \ldots, c_K\}$. In this chapter we shall limit ourselves to using a single logical sensor, or classifier, S. We shall then remove this restriction in Chapt. 13 where we consider multiple classifier systems.

In many applications Bayesian decision theory represents the primary fusion algorithm in a multi-sensor data fusion system. In Table 12.1 we list some of these applications together with their Desararthy classification.

12.2 Pattern Recognition

Formally, the pattern recognition problem deals with the optimal assignment of an object O to one of K possible classes, $\{c_1, c_1, \ldots, c_K\}$. We suppose O is characterized by L measurements, or features, $y^{(l)}, l \in \{1, 2, \ldots, L\}$. Then, mathematically, the classifier defines a mapping between a class variable C and the feature vector $\boldsymbol{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(L)})^T$:

$$\boldsymbol{y} \rightarrow C \in \{c_1, c_2, \ldots, c_K\} \; . \tag{12.1}$$

If we wish to minimize the number of classification errors, we use a zero-one loss function (see Sect. 9.2) and assign O to the class with the largest *a posteriori* probability:

$$C = c_{\text{MAP}} \qquad \text{if } p(C = c_{\text{MAP}}|\boldsymbol{y}, I) = \max_k (p(C = c_k|\boldsymbol{y}, I)) \; , \tag{12.2}$$

where I denotes all relevant background information we have regarding O and its measurement vector $\boldsymbol{y}$.

Table 12.1. Applications in which Bayesian Decision Theory is the Primary Fusion Algorithm

Class		Description
FeI-DeO	Ex. 1.7	Automatic Speech Recognition: Preventing Catastrophic Fusion.
	Ex. 3.4	Tire Pressure Monitoring Device.
	Ex. 3.5	Multi-Modal Biometric Identification Scheme.
	Ex. 6.4	Speech-Recognition in an Embedded-Speech Recognition System.
	Ex. 6.5	Word-Spotting in Historical Manuscripts.
	Ex. 12.3	Plug-In MAP Bayes' Classifier vs. Naive Bayes' Classifier.
	Ex. 12.4	Naive Bayes' Classifier: Correlated Features.
	Ex. 12.5	Gene Selection: An application of feature selection in DNA microarray experiments.
	Ex. 12.6	mRMR: A Filter Feature Selection Algorithm.
	Ex. 12.7	Wrapper Feature Selection.
	Ex. 12.8	Tree Augmented Naive (TAN) Bayes' Classifier.
	Ex. 12.9	Adjusted Probability Model.
	Ex. 12.10	Homologous Naive Bayes' Classifier.
	Ex. 12.11	Selective Neighbourhood Naive Bayes' Classifier.
DeI-DeO	Ex. 12.12	Naive Bayes' Classifier Committee (NBCC).
	Ex. 12.13	Multi-Class Naive Bayes' Classifier.
	EX 13.1	Bayesian Model Averaging of the Naive Bayes' Classifier
	Ex. 13.16	Boosting a Classifier.

The designations FeI-DeO and DeI-DeO refer, respectively, to the Dasarathy input/output classifications: "Feature Input-Decision Output" and "Decision Input-Decision Output" (Sect. 1.3.3).

Example 12.1. Confusion Matrix. A useful tool for recording the performance of a given classifier is the *confusion matrix* $\boldsymbol{\Omega}$. This is a $K \times K$ matrix in which the (l,k)th element $\Omega^{(lk)}$ records the *a posteriori* probability $P(C_{\text{true}} = c_l | C = c_k)$ where C_{true} and C are, respectively, the true and predicted class variables.

In practice the elements $\Omega^{(l,k)}$ are approximated as follows. Let D denote a training set of N objects $O_i, i \in \{1, 2, \ldots, N\}$. Then

$$\Omega^{(l,k)} = \frac{n(l,k)}{\sum_{l=1}^{K} n(l,k)} \ ,$$

where $n(l,k)$ is the number of objects O_i which belong to the class $C_{\text{true}} = c_l$ but which are assigned by S to the class $C = c_k$. □

Equation (12.2) is known as the maximum *a posteriori* (MAP) rule. The computation of the *a posteriori* pdf, $p(C = c_{\text{MAP}} | \boldsymbol{y}, I)$, is performed by applying Bayes' Theorem:

$$\begin{aligned} c_{\text{MAP}} &= \arg\max_k p(C = c_k | \boldsymbol{y}, I) \ , \\ &= \arg\max_k \left(\frac{\pi(C = c_k | I) p(\boldsymbol{y} | C = c_k, I)}{p(\boldsymbol{y} | I)} \right) \ , \end{aligned} \tag{12.3}$$

where $p(\boldsymbol{y}|I)$ serves as a normalization constant. This constant can be ignored for classification purposes, giving,

$$c_{\text{MAP}} = \arg\max_k \pi(C = c_k|I)p(\boldsymbol{y}|C = c_k, I) \ . \tag{12.4}$$

For *known* probability densities the MAP rule is the best any classifier can do with a zero-one loss function. However, in practice the pdf's in (12.4) are unknown. In this case we use the following empirical, or *plug-in*, MAP rule:

$$c_{\text{PLUG-IN}} = \arg\max_k \hat{\pi}(C = c_k|I)\hat{p}(\boldsymbol{y}|C = c_k, I) \ , \tag{12.5}$$

where $\hat{\pi}(C = c_k|I)$ and $\hat{p}(\boldsymbol{y}|C = c_k, I)$ denote *estimated* pdf's [293, 295].

Direct calculation of the *a priori* probability $\hat{\pi}(C = c_k|I)$ is straightforward: Let D denote a training set of N objects $O_i, i \in \{1, 2, \ldots, N\}$. Each object has an instantiated measurement vector $\boldsymbol{y}_i$ and a label z_i, where $z_i = k$ if O_i belongs to the class $C = c_k$. If D is a representative sample of the underlying true population, then an unbiased maximum likelihood estimate of the true *a priori* probability is

$$\hat{\pi}(C = c_k|I) = \sum_{i=1}^{N} I_i/N \ , \tag{12.6}$$

where

$$I_i = \begin{cases} 1 \text{ if } z_i = k \ , \\ 0 \text{ otherwise} \ . \end{cases} \tag{12.7}$$

However, in many applications, D is not a random sample of the true population. In this case, and in the absence of any specific information, we often simply let $\hat{\pi}(C = c_k|I) = \frac{1}{K}$.

Direct calculation of the likelihood function, $\hat{p}(\boldsymbol{y}|C = c_k, I)$, is more difficult, especially for high-dimensional data [278]. The problem is that the required size of D increases *exponentially* with L [261]. This is known as the *curse of dimensionality* [92] and it affects all aspects of classification theory. For example, the curse of dimensionality implies that for a given number of objects O_i in D, there is a maximum number of features above which the performance of Bayes' classifier will degrade rather than improve. This is illustrated in the following example.

Example 12.2. The Small Sample Problem [304]. We consider a two-class classification problem with equal *a priori* probabilities. Each class is characterized by an L-dimensional Gaussian distribution: $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{N}(\boldsymbol{y}| - \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\mu} = \left(1, 1/\sqrt{2}, 1/\sqrt{3}, \ldots, 1/\sqrt{L}\right)^T \ .$$

and

$$\boldsymbol{\Sigma} = \text{diag}(\overbrace{1, 1, \ldots, 1}^{L}) \ .$$

The features are statistically independent and the discriminating power of the successive features decrease monotonically with the first feature providing the maximum discrimination between the two classes. If $\boldsymbol{\mu}$ is *known* then the probability of a misclassification error $P_\epsilon(L)$ decreases with L as shown in Fig. 12.1. However, if $\boldsymbol{\mu}$ is *unknown*, then we estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from a training set D containing N labelled objects $O_i, i \in \{1, 2, \ldots, N\}$, with instantiated measurements $\boldsymbol{y}_i$. In this case, the probability of misclassification, $P_\epsilon(N, L)$, is a function of both L and N: For a given value of N, $P_\epsilon(N, L)$ initially decreases with L but eventually it increases and takes on the value of $\frac{1}{2}$.

An explanation of this phenomenom is as follows. For a fixed value of N, the reliability of the likelihood functions decrease with an increase in L, and as a consequence, the performance of the classifiers also fall with an increase in L. □

From this perspective, Bayesian classification theory may be interpreted as an attempt to "negate" the curse of dimensionality, i. e. to make accurate estimates of the likelihoods, $\hat{p}(\boldsymbol{y}|C = c_k, I)$, while using a small number of objects O_i. One way of reducing the effect of the curse of dimensionality is to impose a given structure on $\hat{p}(\boldsymbol{y}|C = c_k, I)$. This idea forms the basis of the naive Bayes' classifier which is probably the most widely used classifier used today.

Note. To keep notation simple we shall not, in general, differentiate between estimated pdf $\hat{p}(\boldsymbol{y}|C = c_k, I)$ and the true pdf $p(\boldsymbol{y}|C = c_k, I)$. The

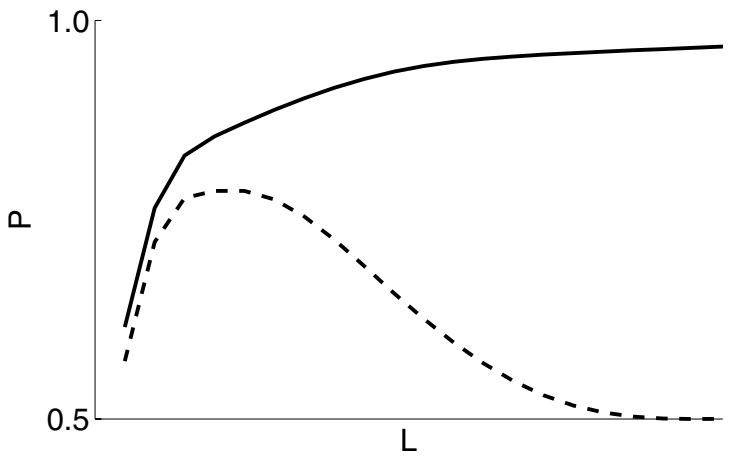

Fig. 12.1. Shows the probability of correct classification P (where P is defined as $1 - P_\epsilon$) as a function of the number of features L. The solid curve shows P calculated assuming $\boldsymbol{\mu}$ is known. The curve increases montonically to one as the number of features increases. The dashed curve shows P calculated assuming $\boldsymbol{\mu}$ is estimated from a training set of N labelled objects. This curve initially increases with L but eventually it decreases and takes on the value of $\frac{1}{2}$.

reader should observe, however, that in most practical cases, the pdf's are estimated using a maximum likelihood or maximum *a posteriori* estimator.

12.3 Naive Bayes' Classifier

In the naive Bayes' classifier (NBC) we "negate" the curse of dimensionality, by supposing that $p(\boldsymbol{y}|C = c_k, I)$ is a product of L one-dimensional likelihood functions $p(y^{(l)}|C = c_k, I)$:

$$p(\boldsymbol{y}|C = c_k, I) = \prod_{l=1}^{L} p(y^{(l)}|C = c_k, I) \; . \tag{12.8}$$

In this case, the required number of objects O_i in D increases only *linearly* with L.

Example 12.3. Plug-in MAP Classifier vs. Naive Bayes' Classifier. Consider a classifier with L features $y^{(1)}, y^{(2)}, \ldots, y^{(L)}$, whose instantiated values are limited to a discrete set of M_l values $\alpha_m^{(l)}, m \in \{1, 2, \ldots, M_l\}$. If we assume a minimum of one object O_i for each combination of feature values, then the plug-in MAP classifier requires $\prod_{l=1}^{L} M_l$ objects, while the NBC only requires $\sum_{l=1}^{L} M_l$ objects. □

Equation (12.8) is clearly not the only approximation which can be used and in some applications the following approximations [153] are used instead:

$$p(\boldsymbol{y}|C = c_k, I) = \begin{cases} \min_l p(y^{(l)}|C = c_k, I) \; , \\ \frac{1}{L} \sum p(y^{(l)}|C = c_k, I) \; , \\ \text{median}_l p(y^{(l)}|C = c_k, I) \; , \\ \max_l p(y^{(l)}|C = c_k, I) \; . \end{cases} \tag{12.9}$$

However, in general, (12.8) is the preferred approximation [1].

12.3.1 Representation

The NBC is a highly restricted Bayesian network in which all of the features $y^{(l)}, l \in \{1, 2, \ldots, L\}$, have one and the same independent class variable C. When depicted graphically, a NBC has the form shown in Fig. (12.2), in which all arcs are directed from the class variable C to the features $y^{(l)}, l \in \{1, 2, \ldots, L\}$. If $pa(y^{(l)})$ represents the *parents* of a given feature $y^{(l)}$, then by definition the NBC has $pa(C) = 0$ and $pa(y^{(l)}) = C$.

[1] Equation 12.8 may be "justified" as follows: Let $\{\pi(C = c_k|I), p(y^{(1)}|C = c_k, I), \ldots, p(y^{(L)}|C = c_k, I)\}$ denote the set of "low-order" estimates used by the NBC. Then the product approximation contains the smallest amount of information (i. e. the maximum entropy) of all possible approximations to $p(\boldsymbol{y}|C = c_k, I)$ that use the same set of low-order estimates [314].

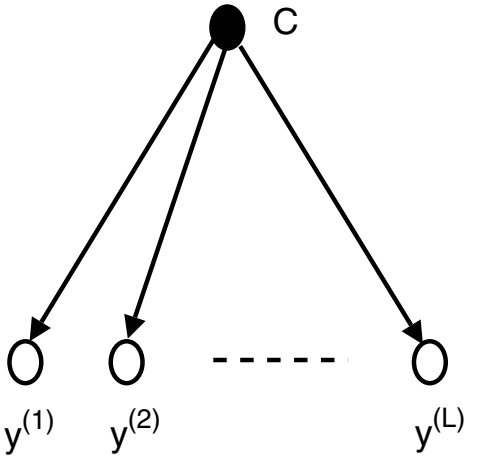

Fig. 12.2. Shows a NBC. It contains one class variable C and L features $y^{(l)}, l \in \{1, 2, \ldots, L\}$. The NBC has $pa(C) = 0$ and $pa(y^{(l)}) = C$.

12.3.2 Performance

The NBC rule corresponding to (12.8) is

$$c_{\text{NBC}} = \arg\max_k \pi(C = c_k|I) \prod_{l=1}^{L} p_l(y^{(l)}|C = c_k, I) \; . \tag{12.10}$$

This equation assumes the features $y^{(l)}$ are independent. At first sight, we may believe that the NBC is optimal, i. e. has the smallest number of classification errors, when the $y^{(l)}$ are independent, and perhaps close to optimal when the $y^{(l)}$ are weakly dependent. However, in practice the classifier has a good performance in a wide variety of domains, including many where there are clear dependencies between the $y^{(l)}$ [266]. The following example provides a plausible explanation of this phenomenom.

Example 12.4. Naive Bayes' Classifier: Correlated Features [63]. We consider the classification of an object O into two classes c_1 and c_2. The object O is described by the feature vector $\boldsymbol{y} = (y^{(1)}, y^{(2)}, y^{(3)}))^T$, where we assume $y^{(1)}$ and $y^{(3)}$ are independent, and $y^{(1)}$ and $y^{(2)}$ are completely dependent (i. e. $y^{(1)} = y^{(2)}$). For simplicity we shall further assume that $\pi(C = c_1|I) = \frac{1}{2} = \pi(C = c_2|I)$. In this case, the plug-in MAP and the NBC, classification rules are, respectively,

$$\begin{aligned} c_{\text{PLUG-IN}} &= \arg\max_k p(y^{(1)}|C = c_k, I) p(y^{(3)}|C = c_k, I) \; , \\ &= \begin{cases} c_1 \text{ if } B > 1 - A \; , \\ c_2 \text{ otherwise} \; , \end{cases} \end{aligned}$$

and

$$c_{\text{NBC}} = \arg\max_k p(y^{(1)}|C = c_k, I)^2 p(y^{(3)}|C = c_k, I) \ ,$$
$$= \begin{cases} c_1 \text{ if } B > (1-A)^2/(A^2 + (1-A)^2) \ , \\ c_2 \text{ otherwise} \ , \end{cases}$$

where $A = p(y^{(l)}|C = c_1, I)$ and $B = p(y^{(l)}|C = c_2, I)$. The two curves, $B = 1 - A$ and $B = (1-A)^2/(A^2 + (1-A)^2)$, are shown in Fig. (12.3). We observe that, although the independence assumption is decisively violated ($y^{(2)} = y^{(1)}$), the NBC disagrees with the plug-in MAP classifier only in the two narrow regions that are above one of the curves and below the other; everywhere else the two classifiers give identical classification results. □

12.3.3 Likelihood

Several different methods are available for estimating the (one-dimensional) likelihoods $p(y^{(l)}|C = c_k, I)$ from a training set D of labelled objects $O_i, i \in \{1, 2, \ldots, N\}$. The most commonly used methods are listed in Table 12.2.

Standard Parametric Distributions

The most common method for estimating the (one-dimensional) likelihoods $p(y^{(l)}|C = c_k, I)$ is to model them using a standard parametric distribution. Traditionally, we use a one-dimensional Gaussian pdf, $\mathcal{N}(\mu_k^{(l)}, \Sigma_k^{(l)})$, for this

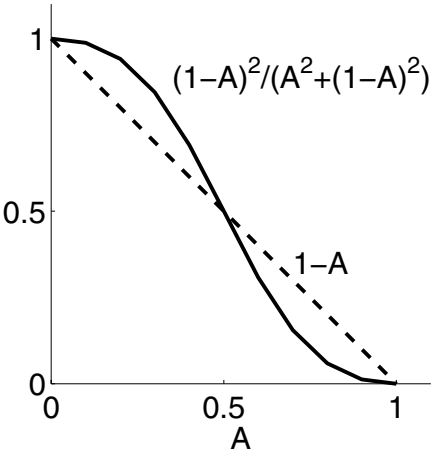

Fig. 12.3. Shows the curves $B = 1 - A$ and $B = (1-A)^2/(A^2 + (1-A)^2)$ as a function of A for the plug-in MAP classifier and the NBC. The only region where the classifiers give different results are regions where one curve is above the other. Everywhere else the two classifiers give identical results.

Table 12.2. Principal Methods Used for Estimating the Likelihood $p(\boldsymbol{y}|C = c_k, I)$.

Method	Description
Standard Parametric Function	Fit $p(y^{(l)}|C = c_k, I)$ to a one-dimensional parametric form using a maximum likelihood, or maximum *a posteriori*, algorithm.
Mixtures	Fit $p(y^{(l)}|C = c_k, I)$ to a mixture of one-dimensional parametric functions using the EM algorithm.
Binning	Convert the feature values $y^{(l)}$ into a finite set of discrete values $\alpha_m, m \in \{1, 2, \ldots, M_l\}$.
Kernel	Calculate $p(y^{(l)}|C = c_k, I)$ using a one-dimensional kernel density estimation procedure.

purpose, where the parameters, $\mu_k^{(l)}$ and $\Sigma_k^{(l)}$, are estimated using the training set D (which contains N labelled objects $O_i, i \in \{1, 2, \ldots, N\}$, with instantiated values $y_i^{(l)}$). For example, the unbiased maximum likelihood estimates of $\mu_k^{(l)}$ and $\Sigma_k^{(l)}$ are given by

$$\mu_k^{(l)} = \sum_{i=1}^{N} I_i y_i^{(l)} / \sum_{i=1}^{N} I_i \ , \tag{12.11}$$

$$\Sigma_k^{(l)} = \sum_{i=1}^{N} \left(y_i^{(l)} - \mu_k^{(l)} \right)^2 / \left(\sum_{i=1}^{N} I_i - 1 \right) \ , \tag{12.12}$$

where

$$I_i = \begin{cases} 1 \text{ if } z_i = k \ , \\ 0 \text{ otherwise} \ . \end{cases} \tag{12.13}$$

Note: Although the Gaussian pdf is the most popular distribution, other alternative pdf's are sometimes found beneficial [20].

Mixtures

The method of mixtures is a more flexible parametric method for estimating the one-dimensional likelihoods $\hat{p}(y^{(l)}|C = c_k, I)$. In this method we approximate $\hat{p}(y^{(l)}|C = c_k, I)$ using a finite mixture model. For example, if we use a Gaussian mixture model (see Sect. 8.5), then

$$\hat{p}(y^{(l)}|C = c_k, I) \approx \sum_{m=1}^{M} \alpha_{km}^{(l)} \mathcal{N}(\mu_{km}^{(l)}, \Sigma_{km}^{(l)}) \ , \tag{12.14}$$

where $\alpha_{km}^{(l)} \geq 0$ and $\sum_{m=1}^{M} \alpha_{km}^{(l)} = 1$. The parameters $\alpha_{km}^{(l)}, \mu_{km}^{(l)}$ and $\Sigma_{km}^{(l)}$ are found using the EM algorithm (see Sect. 8.4.4). Although traditionally we use a Gaussian mixture model, robust mixture models (see Sect. 10.4) are becoming more common.

Binning

Binning is a simple non-parametric method used for estimating $p(y^{(l)}|C = c_k, I)$ (see Sect. 7.6.2). In this method, we divide the feature space, $y^{(l)}$, into a discrete set of M non-overlapping bins. Let α_m denote the value of the lth feature at the center of the mth bin. Then the estimate of the likelihood at $y^{(l)} = \alpha_m$ is found using the Laplace formula:

$$p(y^{(l)} = \alpha_m | C = c_k, I) = \frac{n_m^{(k)} + 1}{\sum_{m=1}^{M} n_m^{(k)} + K} , \tag{12.15}$$

where $n_m^{(k)}$ is the number of $C = c_k$ objects $O_i, i \in \{1, 2, \ldots, N\}$, whose instantiated values $y_i^{(l)}$ fall within the mth bin. An effective method for choosing the size of the bins is to vary the bins so that each bin contains $\sim \sqrt{N}$ objects [339, 341].

Kernals

The method of Parzen windows, or kernels, is a more flexible non-parametric method for estimating the likelihoods $p(y^{(l)}|C = c_k, I)$ (see Sect. 7.6.3). In this method, the estimated likelihood functions is given by

$$p(y^{(l)}|C = c_k, I) = \frac{1}{H} \sum_{i=1}^{N} K\left(\frac{y^{(l)} - y_i^{(l)}}{H} \right) \times I_i / \sum_{i=1}^{N} I_i , \tag{12.16}$$

where $I_i = 1$ if O_i belongs to the class $C = c_k$, otherwise $I_i = 0$, $K(y)$ is a bounded non-negative function satisfying $\int K(x)dx = 1$ and H is a positive number usually called the bandwidth.

In general the accuracy of $p(y^{(l)}|C = c_k, I)$ is primarily determined by the choice of bandwidth and only in a minor way by the choice of kernel function [276, 278]. A popular choice of kernel is the Gaussian function [2].

In Table 12.3 we list some of the methods used for calculating the optimal bandwidth for one-dimensional densities. For multi-variate kernels see [67].

[2] The reader should note that choosing the Gaussian as the kernel function is *not* the same as fitting the input data to a Gaussian mixture model (GMM) (Sect. 8.4.4). In kernel density estimation, the Gaussian function is only used to weight the input data. Unlike GMM, kernel density is a more general approach which does not assume any specific shape for the density function.

Table 12.3. Methods for Calculating Optimum One-Dimensional Bandwidth H

Name	Description
Rule-of-Thumb	Suppose the input data (consisting of N measurements $y_i, i \in \{1, 2, \ldots, N\}$, is generated by a given parametric density function, e. g. a Gaussian function. In this case $H = 1.06\hat{\sigma}N^{-1/5}$, where $\hat{\sigma}$ is the sample standard deviation. Robust versions of this bandwidth are available: $H = 1.06\min(\hat{\sigma}, \hat{Q}/1.34)N^{-1/5}$ and $H = 1.06\hat{s}N^{-1/5}$, where $\hat{Q}$ is the sample interquartile distance and $\hat{s} = \text{med}_j\|y_j - \text{med}_i y_i\|$.
Cross-Validation (CV)	Use a CV procedure to directly minimize the MISE or the AMISE (see sect. 12.5). CV variants include least square, biased and smoothed CV [143].
Plug-in	Minimize the AMISE using a second bandwidth known as the *pilot* bandwidth L. In the *solve-the-equation plug-in* method we write L as a function of the kernel bandwidth H [143].

MISE is the mean integrated square error and is defined as $\text{MISE}(p, \hat{p}_H) = \int \big(p(y) - \hat{p}_H(y)\big)^2 dy$, where $\hat{p}_H(y)$ is the kernel approximation to $p(y)$. AMISE denotes the asymptotic MISE and represents a large number approximation of the MISE.

12.4 Modifications

The NBC has a good performance over a wide variety of domains, including many domains where there are clear dependencies between the attributes. The NBC is also robust to noise and irrelevant attributes. As a result, the NBC is widely used in many practical applications. It has also attracted much attention from many researchers who have introduced various modifications in an effort to improve its performance. Broadly speaking we group the modifications under four different headings:

Features. Select, or create, a small number of independent features.
Models. Relax the naive Bayes' model assumptions. For example, we may allow the features to be weakly dependent or of different importance.
Learning Methods. Replace the off-line learning procedure with an adaptive on-line lazy learning technique.
Multiple Classifiers. Apply the concept of ensemble learning to the naive Bayes' classifier.

12.4.1 Feature Space

In modifying the feature space our main concern is in reducing the number of features used. Table 12.4 lists the principal modifications of the feature space. The observant reader may wonder why it is necessary to reduce the

Table 12.4. Feature Space Modifications

Method	Description
Feature Selection	Select a small number of the input features. Two methods for doing this are: (i) *Filters* [48, 107, 120, 164, 240] and (ii) *Wrappers* [140, 156, 183, 184].
Feature Extraction	Create a small number of new independent features. Methods for doing this include the subspace techniques discussed in Sect. 4.5 [120, 178, 182, 254, 286].
Feature Joining (Constructive NBC)	Construct new features using the Cartesian product of the original features [239] This method is only applicable for features which are discrete or have been discretized.

number of features if the NBC has already reduced the effect of the curse of dimensionality. However, as the following example shows (see also Ex. 12.2) [3] we often find it very beneficial to limit the total number of features used.

Example 12.5. Gene Selection: An Application of Feature Selection in DNA Microarray Experiments [133]. DNA microarray experiments, generating thousands of gene expression measurements, are used to collect information from tissue and cell samples regarding gene expression differences that can be useful for diagnosis disease, distinction of the specific tumor type etc. One important application of gene expression microarray data is the classification of samples into known categories.

From a classification point of view, the biological samples can be seen as objects and the genes as features used to describe each object. In a typical microarray dataset the number of samples is small (usually less than 100) but the number of genes is several thousand, with many of the genes being correlated or irrelevant. For diagnostic purposes it is important to find small subsets of genes that are sufficiently informative to distinguish between cells of different types. Studies show that most genes measured in a DNA microarray experiment are not relevant for an accurate distinction among the different classes of the problem and simple classifiers with few genes (less than 15-20) achieve better accuracies. To avoid the "curse of dimensionality" feature selection is required to find the gene subset with the best classification accuracy for a given classifier. □

Feature Selection

In feature selection our goal is to find an optimal subset of the input features that optimizes some criteria $\mathcal{J}$. The simplest feature selection approach is to evaluate each feature individually and select the L' features with the highest

[3] Jain and Chandrasekaran [136] suggest using at least ten times as many training samples per class as the number of features, i. e. $M_k > 10L$.

scores. Unfortunately this approach ignores feature correlations and it will rarely find an optimal subset. A brute force approach would be to evaluate *all* possible subsets of L' features and select the global optimum, but the number of combinations $C(L, L') = \binom{L}{L'}$ becomes impractical even for moderate values of L' and L. A compromise between these two approaches is to selectively examine a limited number of feature subsets. Some ways of doing this are listed in Table 12.5.

Given a subset of features two different strategies are available to evaluate the degree to which the subset is optimal. In the first strategy we evaluate the degree to which a subset of features is optimal indirectly: it is based on a filter in which we evaluate the optimality of a feature subset on the basis of its "information content" Φ. The second strategy follows a more direct approach and evaluates the optimality of a feature subset on the basis of their estimated classification accuracy, or error probability, P_ϵ [156].

The following examples illustrate the two different feature selection strategies.

Example 12.6. mRMR: A Filter Feature Selection Algorithm [240]. The mRMR is a powerful filter feature selection algorithm in which our aim is to select a subset of features with *maximum relevancy* (i. e. high discriminatory power)

Table 12.5. Feature Selection Methods

Method	Description
Exhaustive Search	Evaluate all $C(L, L')$ possible subsets. Guaranteed to find the optimal subset. Not feasible for even moderately large values of L' and L.
Best Individual Features	Evaluate all the m features individually. Select the best L' individual features. Computationally simple but not likely to lead to an optimal subset.
Sequential Forward Selection	Select the best single feature and then add one feature at a time which in combination with the selected features maximizes the criterion function. Although computationally attractive its main drawback is that once a feature is retained it cannot be discarded.
Sequential Backward Selection	Start with all the L features and successively delete one feature at a time. Computationally attractive although it requires more computation than the sequential forward selection. Its main drawback is that once a feature is deleted it cannot be brought back into the optimal subset.
Sequential Forward Floating Search	First enlarge the feature subset by m features using forward selection and then delete n features using backward selection. This technique attempts to eliminate the drawbacks of the sequential forward and backward selection methods. The technique provides close to optimal solution at an affordable computational cost.

and *minimum redundancy* (i. e. low inter-feature correlations). The mRMR filter relies on the concept of mutual information (Sect. 5.2.1) to define Φ:

$$\Phi = \frac{1}{L}\sum_{l=1}^{L} MI(C, y^{(l)}) - \frac{2}{L(L-1)}\sum_{l=1}^{L-1}\sum_{h=l+1}^{L} MI(y^{(l)}, y^{(h)}) \; .$$

This equation consists of two terms after the equals sign. The first term, $\frac{1}{L}\sum MI(C, y^{(l)}), y^{(l)} \neq y^{(h)}$, represents the average amount by which the uncertainty in the class variable C is reduced after having observed a feature $y^{(l)}$. The second term, $\frac{2}{L(L-1)}\sum_l \sum_h MI(y^{(l)}, y^{(h)})$, represents the average amount by which the uncertainty in a given feature $y^{(l)}$ is reduced after having observed another feature $y^{(h)}, h \neq l$. □

Example 12.7. Wrapper Feature Selection [156]. In wrapper based feature selection we select feature subsets on the basis of their estimated classification accuracy, or probability of error, P_ϵ. Sophisticated methods for accurately estimating P_ϵ are listed in Table 12.9.

Although theoretically optimal, the procedure is liable to *overfit*. This happens when a feature subset has a low P_ϵ value (as measured on the training data) but which generalizes poorly on the test data [53, 156, 183]. Some effective ways of preventing overfitting are given in [184]. □

Feature Extraction

In feature selection we select a small number of the input features for use in the NBC. In *feature extraction* we follow a different approach and create a small number of new features from the input features. Mathematically, we define feature extraction as creating new features, $y^{(l)}, l \in \{1, 2, \ldots, L'\}$, from the original set of features $x^{(l)}, l \in \{1, 2, \ldots, L\}$. Our aim is to extract a small number of new features which contain the maximal information concerning the class variable C. Two popular linear methods for feature extraction are Principal Component Analysis (PCA) and Linear Discriminat Analysis (LDA) (see Sect. 4.5).

As in the case of feature selection, we may also perform feature extraction by directly optimizing the classifier performance, i. e. by minimizing the estimated probability of classification error P_ϵ.

Feature Joining

Given two correlated one-dimensional features, x_1 and x_2, we may construct a new (two-dimensional) feature, $\boldsymbol{x}$, using the Cartesian product of x_1 and x_2. The likelihood of $\boldsymbol{x}$ is modeled using a multi-dimensional version of one of the methods listed in Table 12.2 [239].

Table 12.6. Model Modifications in the Naive Bayes Classifier

Method	Description
Tree Augmented Naive Bayes (TAN)	Relax the NBC assumption of feature independence [91, 150].
Adjusted Probability Model (APM)	Give different weights to each feature [82, 88, 122].
Homologous Naive Bayes (HNB)	Classify multiple objects which belong to the same (unknown) class [127].

12.4.2 Model Assumptions

The second set of modifications all involve changes to the independence assumption as used in the NBC. Table 12.6 lists some of the modifications which have been introduced into the NBC. The following examples illustrate three different ways the model assumption of the NBC may be changed.

Example 12.8. Tree Augmented Naive (TAN) Bayes' Classifier [91, 150]. In the Tree Augmented Naive (TAN) Bayes' classifier we allow each feature in the NBC to depend on C, and at most, one additional feature. Mathematically, the TAN classifier therefore has: $pa(C) = 0$ and $|pa(y^{(l)})| \leq 2$. Graphically, the TAN is a restricted Bayesian network which uses a tree-structure imposed on the NBC structure (Fig. 12.4). The corresponding TAN classifier rule is

$$c_{\text{TAN}} = \arg\max_k \pi(C = c_k|I) \prod_{l=1}^{L} p_l(y^{(l)}|pa(y^{(l)}), I) \ .$$

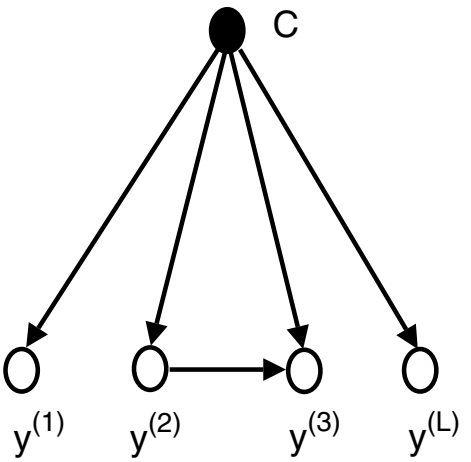

Fig. 12.4. Shows a Tree Augmented Naive (TAN) Bayes' classifier containing one class variable C and four features $y^{(l)}, l \in \{1, 2, 3, 4\}$. The parents of $y^{(1)}, y^{(2)}$ and $y^{(4)}$ are C. The parents of $y^{(3)}$ are C and $y^{(2)}$.

Many different methods are available for building the TAN classifier. One method is the super-parent algorithm [150]: The super-parent algorithm consists of three steps:

1. Search for a "super"-parent that has the best estimated classification performance [4]. A super-parent is a node with arcs pointing to all other nodes without a parent except for the class label.
2. Determine one favourite child for the super-parent chosen in the first step, based on its estimated classification accuracy.
3. Add the appropriate arc to the tree structure.

The entire process is repeated until no improvement is achieved, or $L-1$ arcs are added into the tree. □

Example 12.9. Adjusted Probability Model (APM) [82, 88, 122]. In the Adjusted Probability Model (APM) we allow the features to have different weights or importances $w_l, l \in \{1, 2, \ldots, L\}$. The corresponding APM classification rule is

$$c_{\text{APM}} = \arg\max_k \pi(C = c_k|I) \prod_{l=1}^{L} \left(\frac{p(C = c_k|y^{(l)}, I)}{\pi(C = c_k|I)} \right)^{w_l} .$$

Different authors have restricted the weights in various ways. For example, in an unrestricted APM [122] the authors place no resrictions on the w_l, and allow both postive and negative values. □

Example 12.10. Homologous Naive Bayes' Classifier [127]. In the Homologous Naive Bayes' (HNB) classifier we consider the problem of classifying multiple objects which belong to the same (unknown) class. The homologous classification rule is

$$c_{\text{HNB}} = \arg\max_k \pi(C = c_k|I) \prod_{i=1}^{N} \prod_{l=1}^{L} p_l(y_i^{(l)}|C = c_k, I) ,$$

where $y_i^{(l)}$ is the instantiated value of the lth feature in the ith object. □

12.4.3 Learning Methods

The third set of modifications involve changes to the method of learning the NBC. In particular the modifications involve the use of a lazy learning approach [354, 355]: Given an object O we construct a NBC using a set of *weighted* training samples, where the weights fall as we move away from O in feature space. The idea behind the lazy NBC is that by locally learning the likelihoods $\hat{p}(y^{(l)}|C = c_k, I)$ we may reduce the effect of any feature-feature correlations.

The following example illustrates a simple, but very effective, lazy NBC.

[4] Classification performance is conventionally measured using a cross-validation procedure (Sect. 12.5).

Example 12.11. A Selective Neighbourhood Naive Bayes' Classifier [336]. The selective neighbourhood NBC is a lazy learning algorithm which works as follows. Let D denote training set consisting of N objects $O_i, i \in \{1, 2, \ldots, N\}$, which belong to the class $C = c_k$. Each object O_i has a label z_i, where $z_i = k$ if O_i belongs to the class $C = c_k$. For a given test object O, we define a subset $S_l^{(k)}$ of the l objects O_i in D which are most similar to O. Then for each value of l we learn a naive Bayes' classifier, H_l, using the subsets $S_l^{(k)}$. We select the NBC with the highest accuracy (as estimated on the training data) to classify the test object O. □

12.4.4 Multiple Classifiers

The fourth set of modifications involve the use of multiple NBC's instead of a single classifier. Three such algorithms are listed in Table 12.7. The following example illustrates the Naive Bayes' Classifier Committee algorithm.

Example 12.12. Naive Bayes' Classifier Committee (NBCC) [353]. In the Naive Bayes' Classifier Committee (NBCC) we use an ensemble of L NBC's, where L is the number of independent features. Let H_0 denote the first NBC using all L features, and let E_0 be its estimated error rate [5]. Then we repeatedly create candidate NBC's by randomly selecting $L/2$ features. Assuming the candidate NBC is not already a member of the committee, then it becomes a member of the committee if its estimated error is less than E_0.

To classify a test object O, each NBC in the committee is invoked to produce the probability that O belongs to a class $C = c_k$. Then for each class c_k, the probabilities provided by all committee members are summed up. The optimum classification of O is equal to the class with the largest summed probability. □

Table 12.7. Multiple Naive Bayes' Classifiers

Method	Description
Naive Bayes' Classifier Committee	Use an ensemble of K NBC's, in which each classifier is trained on a different subset of features. Final classification is obtained by allowing the NBC's to vote, each classifier having an equal weight [353].
Boosted Naive Bayes	Use ensemble of K NBC's, in which each classifier is trained on a boosted training set (see Sect. 13.8). Final classification is obtained by allowing the NBC's to vote, each classifier having a weight according to the quality of its predictions [71, 263].
Bayesian Model Averaging NBC	Similar to NBCC except classifier has a weight according to the Bayesian evidence that it is the true, or correct, classifier.

[5] Zheng [353] recommends measuring the error rate using the method of cross-validation.

12.5 Error Estimation

The classification error P_ϵ is the ultimate measure of the performance of a classifier. Apart from using P_ϵ to measure the performance of a given classifier we may use it as a general optimization criterion. In Sect. 12.4.1 we used it in selecting an optimum subset of features ("wrapper") and more generally we may use it to select an optimum classifier.

Mathematically, the classification error of a Bayesian classifier is defined as

$$P_\epsilon = 1 - \sum_{k=1}^{K} \int_{A_k} \pi(C = c_k|I) p(\boldsymbol{y}|C = c_k, I) d\boldsymbol{y} \ , \tag{12.17}$$

where A_k is the region where the class $C = c_k$ has the highest *a posteriori* probability. In general, it is often very difficult to perform the integration in (12.17) and obtain a closed-form expression for P_ϵ. However, for the case of two classes c_1 and c_2 with equal *a priori* probabilities and Gaussian likelihoods, $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, closed-form expression for the bounds on P_ϵ are available (Table 12.8).

In practice, we cannot be sure that the above conditions hold and then the only option is to estimate P_ϵ from the measurements $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, as follows. We split the $\boldsymbol{y}_i$ into a set of training data D and a set of test data T. We design the classifier on D and measure its generalization ability, or $1 - P_\epsilon$, on T. If the number of measurements in D is small the classifier will not be robust and its generalization ability will be low. On the other hand, if the number of measurements in T is small, then the confidence in P_ϵ will be low. In Table 12.9 we list several methods for dividing the $\boldsymbol{y}_i$ into D and T and for estimating P_ϵ.

Table 12.8. Bounds on the Two-Class Probability of Error P_ϵ

Bound	Description
Mahalanobis	$P_\epsilon \leq 2/(4 + \Delta)$; where Δ is the Mahalanobis distance $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ [94].
Bhattacharyya	$\frac{1}{2}(1 - \sqrt{1 - e^{-2B}}) \leq P_\epsilon \leq \frac{1}{2} e^{-B}$; where B is the Bhattacharyya distance $\frac{1}{8}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2))^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + (\ln \lvert(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)/2\rvert)/(2\sqrt{\lvert\boldsymbol{\Sigma}_1\rvert\lvert\boldsymbol{\Sigma}_2\rvert})$ [94].
Chernoff	$P_\epsilon \leq \int \mathcal{N}(\mathbf{x}\lvert\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)^s \mathcal{N}(\mathbf{x}\lvert\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)^{1-s} d\mathbf{x}$, where $s \in [0, 1]$ is a parameter [94].
Lee and Choi	$P_\epsilon \approx 0.040219 - 0.70019B + 0.63578B^2 - 0.32766B^3 + 0.087172B^4 - 0.0091875B^5$ [169].
Hsieh *et. al.*	$P_\epsilon \approx \frac{1}{2} e^{-D}$; where D is the modified Bhattacharyya distance $D = \left[\frac{2}{3}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2))^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right]^{2/3} + \left[(\ln \lvert(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)/2\rvert)/\sqrt{\lvert\boldsymbol{\Sigma}_1\rvert\lvert\boldsymbol{\Sigma}_2\rvert}\right]^{2/3}$ [126].

Table 12.9. Error Estimation Methods

Method	Description
Resubstitution	Training and test databases are the same. Thus we use all the available data both for training and for testing. The P_ϵ is an optimistic biased estimate especially when the ratio of sample size to sample dimensionality is small.
Holdout	Half the data is used for training and the other half is used for testing. Pessimistic biased estimate. Different partitions will give different estimates.
Leave-One-Out	A classifier is designed using $(M-1)$ samples and evaluated on the one remaining sample. This is repeated M times with different training sets of size $(M-1)$. Estimate is unbiased but it has a large variance. When used for model selection it is asymptotically equivalent to the Akaike Information Criterion (AIC) when the number of observations is large.
K-Fold Cross Validation	A compromise between the holdout and leave-one-out methods. Divide the available samples into K disjoint subsets, $1 \leq K \leq M$. Use $(K-1)$ subsets for training and the remaining subset for testing. Estimate has lower bias than the holdout method. Recommended value for K is in the range $[M/5, M/2]$. When used for model selection it is asymptotically equivalent to Bayesian Information Criterion (BIC) for an appropriate K when the number of observations is large.
Bootstrap	Generate many bootstrap sample sets of size M by sampling all the available data with *replacement*. Bootstrap estimates can have lower variance than the leave-one-out method. Useful in small sample situations.

12.6 Pairwise Classifiers

Until now we have placed no restrictions on the number of possible classes K. However, in practice, finding features which are effective for all of the classes $c_k, k \in \{1, 2, \ldots, K\}$, becomes increasingly difficult as K increases. For this reason, we often decompose a K-class pattern recognition problem into $K(K-1)/2$ two-class pattern recognition problems. In each two-class, or pairwise problem, we may use different features and a different number of features. The final classification is then obtained by combining the $K(K-1)/2$ pairwise results. This procedure has already been considered in detail in Sect. 7.6.5. As a reminder to the reader we illustrate the combination of $K(K-1)/2$ pairwise NBC's using a simple voting scheme.

Example 12.13. Multi-Class Classification Using Multiple Pairwise Naive Bayes' Classifiers. We consider the classification of an object O into one of K classes $c_k, k \in \{1, 2, \ldots, K\}$. For each pair of classes, $c_k, c_l, l \neq k$, we obtain an optimal pairwise classification, or vote V_{kl}, using a pairwise NBC, where

$$V_{kl} = \begin{cases} 1 \text{ if } p(C = c_k|\boldsymbol{y}, I) > p(C = c_l|\boldsymbol{y}, I) \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

Then the optimal multi-class classification is $C = c_{\mathrm{OPT}}$, where

$$c_{\mathrm{OPT}} = \arg\max_k \sum_{\substack{l=1 \\ l \neq k}}^{K} V_{kl} \ ,$$

and $V_{lk} = 1 - V_{kl}$. □

12.7 Software

BNT (Bayes' Net Toolbox). A matlab toolbox for performing Bayesian inference.

GPML (Gaussian Processes for Machine Learning). A collection of m-files.

KDE (Kernel Density Estimation). A matlab toolbox for performing kernel density estimation.

NETLAB. A matlab toobox for performing neural network pattern recognition.

STPRTOOL (Statistical Pattern Recognition Toolbox). A matlab toolbox for performing statistical pattern recognition.

12.8 Further Reading

Bayesian decision theory and classification are discussed in detail in [94, 115, 163, 322]. Specific references on Bayesian classification with Gaussian processes are given in [258, 333]. The naive Bayes' classifier is discussed in many publications, including: [171, 181, 265, 266]. Extensions to the NBC include the references listed in Sect. 12.4 and [324, 350]. The NBC has been used in many applications, including text classification, fraud diagnosis [313] and other applications [292]. Finally, the issue of performing statistical analysis using a small number of samples discussed in detail in [194, 195].

13

Ensemble Learning

13.1 Introduction

The subject of this chapter is ensemble learning in which our system is characterized by an ensemble of M models. The models may share the same common representational format or each model may have its own distinct common representational format. To make our discussion more concrete we shall concentrate on the classification of an object O using a *multiple classifier system* (MCS). Given an unknown object O, our goal is to optimally assign it to one of K classes, $c_k, k \in \{1, 2, \ldots, K\}$, using an ensemble of M classifiers, $S_m, m \in \{1, 2, \ldots, M\}$. The theory of multiple classifier systems suggests that if the *pattern of errors* made by one classifier, S_m, is different from the pattern of errors made by another classifier, S_n, then we may exploit this difference to give a more accurate and more reliable classification of O. If the error rates of the classifiers are less than $\frac{1}{2}$, then the MCS error rate, E_{MCS}, should decrease with the number of classifiers, M, and with the mean diversity, $\bar{\sigma}$, between the classifiers [1]. Mathematically, if $\bar{E}$ is the mean error rate of the S_m, then

$$E_{\text{MCS}} \sim \alpha \bar{E} \,, \tag{13.1}$$

where $\alpha \sim 1/(\bar{\sigma} M)$ and $0 \leq \alpha \leq 1$.

The theory of ensemble learning deals with exploiting these differences and this will be the main emphasis in this chapter. In Table 13.1 we list some examples of these applications together with their input/output classification.

13.2 Bayesian Framework

In this section we present a general Bayesian framework [153, 288] for combining an ensemble of classifiers $S_m, m \in \{1, 2, \ldots, M\}$. Given an unknown

[1] The concept of diversity is discussed in Sect. 13.5. For the present we may regard $\bar{\sigma}$ as being equal to the inverse of the mean correlation between the S_m.

Table 13.1. Applications in which Ensemble Learning is the Primary Fusion Algorithm

Class		Application
DhI-DeO	Ex. 10.10	Target tracking initialization using a Hough transform.
	Ex. 13.8	Weighted majority Vote.
	Ex. 13.9	BKS Look-Up Table.
	Ex. 14.6	An adaptive Multi-Modal Biometric Management Algorithm.
DsI-DeO	Ex. 3.4	Tire pressure monitoring device.
	Ex. 3.5	Multi-modal biometric identification scheme.
	Ex. 3.11	Audio-visual speech recognition system.
	Ex. 5.7	Multi-modal cardiac imaging.
	Ex. 7.5	Image thresholding using Kriging.
	Ex. 9.6	Non-intrusive speech quality estimation using GMM's.
	Ex. 12.12	Naive Bayes' Classifier Committee.
	Ex. 13.1	Bayesian model averaging of the Naive Bayes Classifier.
	Ex. 13.10	Image orientation Using Borda Count.
	Ex. 13.16	Boosting a classifier.

The designations DsI-DeO and DhI-DeO refer, respectively, to the (modified) Dasarathy input/output classifications: "Soft Decision Input-Decision Output", and "Hard Decision Input-Decision Output" (see Sect. 1.3.3).

object O, we suppose each classifier S_m makes a measurement $\boldsymbol{y}^{(m)}$ on O and returns an estimate of the *a posteriori* class probability $p(C = c_k|\boldsymbol{y}^{(m)}, I)$:

$$\hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I) = p(C = c_k|\boldsymbol{y}^{(m)}, I) + \epsilon^{(m)} \ , \tag{13.2}$$

where $\epsilon^{(m)}$ is the error made by S_m in estimating $p(C = c_k|\boldsymbol{y}^{(m)}, I)$. We may improve the classification accuracy of O we estimating the joint *a posteriori* probability, $p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I)$, which we do by combining the approximate $\hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I)$ values.

We distinguish two extreme cases [288].

Shared Representation. In this case, the S_m share the same common representational format. It follows that the classifiers use the same input data $\boldsymbol{y}^{(1)} = \boldsymbol{y}^{(2)} = \ldots = \boldsymbol{y}^{(M)}$ and as a result,

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) = p(C = c_k|\boldsymbol{y}^{(m)}, I) \quad m \in \{1, 2, \ldots, M\} \ , \tag{13.3}$$

where $p(C = c_k|\boldsymbol{y}^{(m)}, I)$ is estimated by $\hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I)$. If we assume zero-mean errors for $\epsilon^{(m)}$, we may average the $\hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I)$ to obtain a less error-sensitive estimate. This leads to the mean, or average, rule

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) \approx \frac{1}{M} \sum_{m=1}^{M} \hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I) \ . \tag{13.4}$$

We may give each classifier S_m a different weight w_m (see Sect. 12.9), where $0 \leq \omega_m \leq 1$ and $\sum \omega_m = 1$. In this case, the weighted mean, or average, rule is

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) \approx \sum_{m=1}^{M} w_m \hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I) \; . \quad (13.5)$$

Different Representations. In this case the S_m use different common representational formats which we assume are independent (see Sect. 12.3). In this case

$$p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}|C = c_k, I) = \prod_{m=1}^{M} \hat{p}(\boldsymbol{y}^{(m)}|C = c_k, I) \; . \quad (13.6)$$

Assuming equal *a priori* probabilities, $\pi(C = c_k|I)$, and small errors, $\epsilon^{(m)}$, this reduces [153, 163] to the product rule:

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) \approx \frac{\prod_{m=1}^{M} \hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I)}{\sum_{l=1}^{K} \prod_{m=1}^{M} \hat{p}(C = c_l|\boldsymbol{y}^{(m)}, I)} \; . \quad (13.7)$$

In many applications we do not use (13.7) but use one of the following approximations:

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) \sim \begin{cases} \min_l \hat{p}(C = c_k|y^{(l)}, I) \; , \\ \frac{1}{L} \sum \hat{p}(C = c_k|y^{(l)}, I) \; , \\ \text{median}_l \hat{p}(C = c_k y^{(l)}, I) \; , \\ \max_l \hat{p}(C = c_k|y^{(l)}, I) \; . \end{cases} \quad (13.8)$$

Although these approximations are not very accurate they are often preferred because they are less sensitive to noise and outliers [153, 211, 289]. We may also give each classifier, S_m, a different weight w_m, where $0 \leq w_m \leq 1$. In this case the weighted product rule is

$$p(C = c_k|\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)}, I) \approx \frac{\prod_{m=1}^{M} \hat{p}(C = c_k|\boldsymbol{y}^{(m)}, I)^{w_m}}{\sum_{l=1}^{K} \prod_{m=1}^{M} \hat{p}(C = c_k|\boldsymbol{y}, I)^{w_m}} \; . \quad (13.9)$$

In the following example we illustrate the weighted mean rule (13.5), where the weights w_m are calculated using Bayesian model averaging (see Sect. 8.5.2).

Example 13.1. Bayesian Model Averaging of the Naive Bayes' Classifier [28, 100]. We consider Bayesian model averaging of the Naive Bayes' classifier (NBC) (see Sect. 12.3). We assume an unknown object O with an L-dimensional measurement, or feature, vector $\boldsymbol{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(L)})^T$. We may generate several different naive Bayes' classifiers $S_m, m \in \{1, 2, \ldots, M\}$, by using different subsets of features $y^{(l)}$. Let

$$a_{ml} = \begin{cases} 1 \text{ if } S_m \text{ includes the } l\text{th feature } y^{(l)} \ , \\ 0 \text{ otherwise} \ , \end{cases}$$

then, by definition, the *a posteriori* classification probability, as estimated by S_m, is

$$\hat{p}_m(C = c_k|\boldsymbol{y}, I) = \frac{\pi(C = c_k|I) \prod_{l=1}^{L} (\hat{p}(y^{(l)}|C = c_k, I))^{a_{ml}}}{\hat{p}(\boldsymbol{y}|I)} \ .$$

For each S_m we assume an *a priori* probability, $\pi(S_m|I)$, which describes our prior belief in S_m. Given a training data set D, we use Bayes' rule to calculate the *a posteriori* probability $\hat{p}(S_m|D, I) = \pi(S_m|I)\hat{p}(D|S_m, I)/\hat{p}(D|I)$. In Bayesian model averaging (BMA) we use $\hat{p}(S_m|D, I)$ to weight the predictions $\hat{p}_m(C = c_k|\boldsymbol{y}, I)$. The predicted BMA *a posteriori* probability of the object O is

$$\hat{p}_{\text{BMA}}(C = c_k|\boldsymbol{y}, I) \propto \sum_{m=1}^{M} p(S_m|D, I)\hat{p}_m(C = c_k|\boldsymbol{y}, D, I) \ .$$

The object O is assigned to the class $C = c_{\text{OPT}}$, where

$$c_{\text{OPT}} = \arg\max_k \hat{p}_{\text{BMA}}(C = c_k|\boldsymbol{y}, I) \ . \qquad \square$$

Although useful, the Bayesian framework is restricted to classifiers whose output is an *a posteriori* probability $\hat{p}_m(C = c_k|\boldsymbol{y}, I)$ [2]. Unfortunately, most classifiers do not directly measure the *a posteriori* probability $p_m(C = c_k|\boldsymbol{y}, I)$. Instead they produce a belief, $\mu_m^{(k)}$, that $C = c_k$ is the true assignment. Although theoretically it may be possible to convert $\mu_m^{(k)}$ to an *a posteriori* probability, it is rarely performed in practice (see Sect. 7.6). As a consequence, recent research on the MCS has relied less on a formal Bayesian framework, and more on an empirically-based framework.

13.3 Empirical Framework

The empirical framework is based on a wealth of experimental studies which all suggest that it is possible to reduce the classification error in a MCS by a factor which increases with the mean diversity $\bar{\sigma}$. In order to realize this reduction in classification error the empirical framework introduced four changes in the Bayesian framework:

[2] The above Bayesian framework is subject to additional restrictions [100]. The most important restriction is that the S_m are required to be mutually exclusive and exhaustive, i.e. the S_m must cover all possibilities concerning how the input data was generated.

Diversity Techniques. Empirical-based techniques were introduced for creating a diverse ensemble of classifiers.
Diversity Measures. Probabilistic and non-probabilistic methods were developed for measuring ensemble diversity.
Classifier Types. Classifiers were allowed which did not generate a probabilistic output.
Combination Strategies. Probabilistic and non-probabilistic techniques were developed for combining the outputs of the individual classifiers.

We shall now consider each of these changes in turn.

13.4 Diversity Techniques

At the heart of a MCS is an ensemble of diverse classifiers, each with a reasonable performance. Such an ensemble may be created in many different ways. One commonly used technique is to train a parametric classifier, S, on different training sets $D_m, m \in \{1, 2, \ldots, M\}$. The result is an ensemble of *fixed* classifiers S_m which do not have any adjustable parameters. Methods for constructing the training sets $D_m, m \in \{1, 2, \ldots, M\}$, are

Disjoint Partitions. Sample a common training set D without replacement. The resulting D_m share a common representational format and each D_m contains, on average, N/M labeled objects. In general, the S_m which are produced will be diverse but will have high error rates (due to the small size of the D_m).
Different D_m. Construct the $D_m, m \in \{1, 2, \ldots, M\}$, from different input sources. In this case, the D_m may, or may not, share a common representational format. In general, the S_m will be diverse.
Sampling D. Sample D with replacement [3]. The D_m share a common representational format and, in general, the S_m will be moderately diverse.
Subspace Techniques. Sample D after feature extraction or after applying a subspace transformation. In many cases, the D_m will have different common representational formats [4]. In general, the S_m will be diverse.

The following example illustrates *boosting*, a powerful, recently developed, technique for sampling D [274]. In boosting we create the D_m, one at a time, by sampling D as follows. Let S_m be a classifier which was learnt on D_m. Then we use the classification results obtained with S_m to create D_{m+1} by sampling D such that samples which are misclassified by S_m have a higher chance of being chosen than samples which were correctly classified by S_m.

[3] Different sampling techniques are listed in Tables 4.8.
[4] This is not however guarenteed and depends on the sampling technique which is used. For further details see Sect. 4.6.

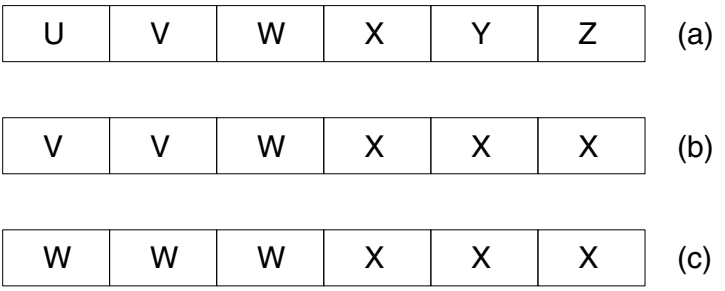

Fig. 13.1. Shows the process of boosting. The common training set D consists of $N = 6$ measurements $\{U, V, W, X, Y, Z\}$. **(a)** Shows the training set D_1 which is identical to D. We train a parameteric classifier S on D_1. The result is a "fixed" classifier S_1 (i. e. it has no free parameters) which misclassifiers X,V, W and Z. **(b)** Shows the training set D_2. This was formed by sampling D_1 such that U,V, W and Z have a high chance of being chosen. We train W on D_2. The result is a fixed classifier S_2 which misclassifies X and W. **(c)** Shows the training set D_3. This was formed by sampling D_2 such that X and W have a high chance of being chosen. We train S on D_3. The result is a fixed classifier S_3 which misclassifies W.

Example 13.2. Boosting a Common Training Set D. Let D denote a common training set consisting of $N = 6$ measurements $\{U, V, W, X, Y, Z\}$. Fig. 13.1 shows the creation of the first three training sets D_1, D_2 and D_3 by boosting D. □

An interesting variant of boosting is multi-boosting [323] which is illustrated in the next example.

Example 13.3. Multi-Boosting a Common Training Set D [323]. We consider multi-boosting the common training set, D, used in Ex. 13.2. We first boost D as explained in Ex. 13.2. Then we perform bagging or wagging (see Sect. 4.6) on each of the boosted training sets D_1, D_2 and D_3. Fig. 13.2 shows the multi-boosted training sets $D_2^{(1)}$, $D_2^{(2)}$ and $D_2^{(3)}$ obtained by bagging D_2. □

The following example illustrates the creation of an ensemble of training sets $D_m, m \in \{1, 2, \ldots, M\}$, using the Skuichina-Duina subspace technique.

Example 13.4. Skurichina-Duin Subspace Technique [282]. We perform principal component analysis on the common training set D. Let $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_L)^T$ denote the corresponding principal component axes. Then we find the training sets $D_m, m \in \{1, 2, \ldots, M\}$, by applying $(\boldsymbol{u}_{(m-1)K+1}, \boldsymbol{u}_{(m-1)K+2}, \ldots, \boldsymbol{u}_{mK-1})$ to D, where $K = L/M$. Thus D_1 is formed by applying $(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_K)$ to D, D_2 is formed by applying $(\boldsymbol{u}_{K+1}, \boldsymbol{u}_{K+2}, \ldots, \boldsymbol{u}_{2K})$ to D, and so on. See Fig. 13.3. □

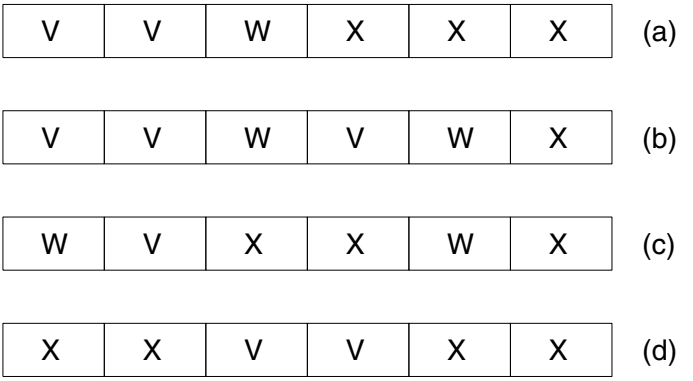

Fig. 13.2. Shows the the process of multi-boosting on the boosted training set D_2. **(a)** Shows the training set D_2. **(b)** Shows results of bagging D_2. The result is $D_2^{(1)}$. **(c)** and **(d)** Shows the results of bagging D_2 two more times. The results are, respectively, $D_2^{(2)}$ and $D_2^{(3)}$.

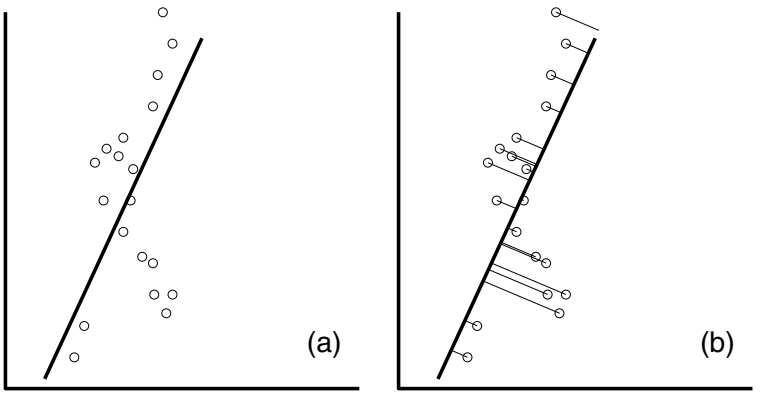

Fig. 13.3. Shows the Skurichina-Duin subspace technique. **(a)** Shows the common training set D which consists of N two-dimensional points $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$. Overlaying the $\boldsymbol{y}_i$ is the main principal axis $\boldsymbol{u}_1$. **(b)** Shows the training set D_1 which is formed by projecting the $\boldsymbol{y}_i$ onto $\boldsymbol{u}_1$.

13.5 Diversity Measures

In principle, the techniques discussed in Sect 13.4 may generate an unlimited number of classifier ensembles. In order to select the most appropriate ensemble we need to compare the diversity of one ensemble with the diversity of another ensemble. In Table 13.2 we list several quantitative diversity measures [287] which are commonly used for this purpose [5]. The simplest

[5] In Sect. 13.1 we defined the diversity of an ensemble as the degree to which the pattern of errors made by one classifier, S_m, differs from the pattern of errors made

Table 13.2. Diversity Measures

Name	Type	Formula
Yule statistic (Q)	P	$Q = (ad-bc)/(ad+bc)$, where a, b, c, d are, respectively, the number of objects $O_i, i \in \{1, 2, \ldots, N\}$, that are correctly classified by S_1 and S_2; are correctly classified by S_1 and incorrectly classified by S_2; are incorrectly classified by S_1 and correctly classified by S_2; are incorrectly classified by S_1 and S_2.
Correlation Coefficient (ρ)	P	$\rho = (ad - bc)/\sqrt{(a+b)(c+d)(a+c)(b+d)}$.
Disagreement Measure (D)	P	$D = (b + c)/N$.
Double Fault Measure (DF)	P	$\mathrm{DF} = d/N$.
Entropy E	N	$E = \sum_{i=1}^{N} \min(\xi_i, (M-\xi_i))/(N(M-\lceil M/2 \rceil))$, where $\lceil x \rceil$ is the ceiling operator and ξ_i is the number of classifiers that misclassify $O_i, i \in \{1, 2, \ldots, N\}$.
Kohavi-Wolpart Variance (KW)	N	$KW = \sum_{i=1}^{N} \xi_i(M - \xi_i)/(NM^2)$.
Measure of Difficulty (θ)	N	$\theta = \sum_{i=1}^{N} (\xi_i - \bar{\xi})^2/(NM^2)$, where $\bar{\xi} = \sum_{i=1}^{N} \xi_i/N$.

diversity measures are pairwise measures (P), which are defined between a pair of classifiers. If the ensemble contains $M > 2$ classifiers, we may estimate its diversity by averaging all of the $M(M-1)/2$ pairwise diversity measures. Apart from the pairwise measures, there are also non-pairwise measures (N) which work on an entire ensemble.

Example 13.5. Pairwise Diversity Measures [163]. We consider two classifiers S_1 and S_2. The measured a, b, c and d values are: $a = 5$, $b = 1$, $c = 1$ and $d = 3$ [6]. The total number of measurements is $N = a + b + c + d = 10$. The corresponding measures of diversity are calculated as follows:

Yule statistic Q.

$$Q = \frac{ad - bc}{ad + bc} = \frac{5 \times 3 - 1 \times 1}{5 \times 3 + 1 \times 1} = \frac{7}{8} .$$

by another classifier, $S_n, n \neq m$. This definition is not, however, very useful in comparing the diversity of one ensemble with another ensemble. For this purpose, we use a quantitative diversity measure.

[6] a, b, c and d are, respectively, the number of objects $O_i, i \in \{1, 2, \ldots, N\}$, that are correctly classified by S_1 and S_2; are correctly classified by S_1 and incorrectly classified by S_2; are incorrectly classified by S_1 and correctly classified by S_2; are incorrectly classified by S_1 and S_2.

Correlation coefficient ρ.

$$\begin{aligned}\rho &= \frac{ad-bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}\ ,\\ &= \frac{5\times 3-1\times 1}{\sqrt{(5+1)(1+3)(5+1)(1+3)}} = \frac{7}{12} \approx 0.58\ .\end{aligned}$$

Diagreement measure D.

$$D = \frac{b+c}{N} = \frac{1+1}{10} = 0.2\ .$$

Double fault measure DF.

$$\mathrm{DF} = \frac{d}{N} = \frac{3}{10} = 0.3\ . \qquad \square$$

Example 13.6. Non-Pairwise Diversity Measurs [163]. We consider three classifiers S_1, S_2 and S_3. The values of ξ_i [7] for the objects O_i are: $\xi_i = (3, 3, 3, 1, 1, 1, 2, 2, 1, 1)$. The corresponding measures of diversity are calculated as follows:

Kohavi-Wolpart KW.

$$\begin{aligned}\mathrm{KW} &= \frac{1}{NK^2}\sum_{i=1}^{N}\xi_i(K-\xi_i)\ ,\\ &= \frac{1}{10\times 3\times 3}(3\times 0+3\times 0+3\times 0+1\times 2+1\times 2+2\times 1\\ &\quad +2\times 1+1\times 2+1\times 2) = \frac{14}{90}\approx 0.16\ .\end{aligned}$$

Entropy E.

$$\begin{aligned}E &= \frac{2}{N(K-1)}\sum_{i=1}^{N}\min(\xi_i, K-\xi_i)\ ,\\ &= \frac{2}{10\times(3-1)}(0+0+0+1+1+1+1+1+1+1) = \frac{7}{10} = 0.7\ .\end{aligned}$$

Measure of difficulty θ.

$$\begin{aligned}\theta &= \frac{1}{NK^2}\sum_{i=1}^{N}(\xi_i-\bar{\xi})^2\ ,\\ &= \frac{1}{10\times 3\times 3}(1.2^2+1.2^2+1.2^2+0.8^2+0.8^2+0.8^2+0.2^2\\ &\quad +0.2^2+0.8^2+0.8^2) = \frac{7.6}{90}\approx 0.08\ .\end{aligned} \qquad \square$$

[7] ξ_i is the number of classifiers that misclassify $O_i, i \in \{1, 2, \ldots, N\}$.

13.5.1 Ensemble Selection

Given an accurate diversity measure we may select the optimal ensemble by exhaustive enumeration, that is, by comparing the diversity measure of all ensembles and choosing the ensemble with the largest diversity measure. However, this technique is only feasible if the number of ensembles is relatively small. When the number of ensembles is large, we may search for the optimal ensemble using any of the search algorithms used in feature selection and listed in Table 12.5.

Alternatively, and in practice this is the preferred method, we may select the optimal ensemble by directly assessing the classification accuracy of each ensemble (as measured on a given test set using any of the methods listed in Table 12.9).

13.6 Classifier Types

Let O denote an unknown object. In the empirical framework we assume that each classifier $S_m, m \in \{1, 2, \ldots, M\}$, generates a *belief* vector, $\boldsymbol{\mu}_m = (\mu_m^{(1)}, \mu_m^{(2)}, \ldots, \mu_m^{(K)})^T$, where $\mu_m^{(k)}$ is the belief that $C = c_k$ is the true assignment. The concept of a belief vector is more general than a probability vector but does include a probability vector as a special case. In fact, we may identify several different types of belief vectors [338]:

Probability Vector. The belief vector is equal to the *a posteriori* pdf vector $\boldsymbol{p}_m = (\hat{p}(C = c_1|\boldsymbol{y}, I), \hat{p}(C = c_2|\boldsymbol{y}, I), \ldots, \hat{p}(C = c_K|\boldsymbol{y}, I)^T$.

Measurement Vector. The belief vector is a measurement vector. The reader should note that this is *not* a probability vector but it may, in principle, be converted into one (see Sect. 7.6).

Fuzzy Membership Vector. The belief vector is a vector of fuzzy membership function values.

Rank Vector. The belief vector is a vector of preferences, or ranks, $\boldsymbol{R}_m = (R_m^{(1)}, R_m^{(2)}, \ldots, R_m^{(K)})^T$, where

$$R_m^{(k)} = K - l \qquad \text{if } \mu_m^{(k)} \text{ is } l\text{th largest value in } \mu_m^{(l)}, l \in \{1, 2, \ldots, K\} \ . \tag{13.10}$$

Identity Vector. The belief vector $\boldsymbol{\mu}_m$ is an identity vector $\boldsymbol{I}_m = (I_m^{(1)}, I_m^{(2)}, \ldots, I_m^{(K)})^T$, where

$$I_m^{(l)} = \begin{cases} 1 \text{ if } \mu_m^{(l)} = \max(\mu_m^{(1)}, \mu_m^{(2)}, \ldots, \mu_m^{(K)}) \ , \\ 0 \ \text{ otherwise} \ . \end{cases} \tag{13.11}$$

The following example illustrates the conversion of a belief vector $\boldsymbol{\mu}$ into a rank vector $\boldsymbol{R}$ and into an identity vector $\boldsymbol{I}$.

Example 13.7. Rank and Identity Vectors. Let $\boldsymbol{\mu}$ denote the following belief vector

$$\boldsymbol{\mu} = \begin{pmatrix} 0.3 & 0.4 & 0.2 & 0.1 & 0.9 \end{pmatrix}^T \ .$$

The corresponding rank and identity vectors are:

$$\boldsymbol{I} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T \ ,$$

and

$$\boldsymbol{R} = \begin{pmatrix} 2, 3, 1, 0, 4 \end{pmatrix}^T \ . \qquad \square$$

13.7 Combination Strategies

Let O denote an unknown object. Suppose each classifier $S_m, m \in \{1, 2, \ldots, M\}$, generates a *belief* vector regarding the classification of O. Then the MCS generates a new belief vector, $\widetilde{\boldsymbol{\mu}}$, by combining the $\boldsymbol{\mu}_m$:

$$\widetilde{\boldsymbol{\mu}} = f(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M)^T \ , \tag{13.12}$$

where f denotes a MCS combination function.

Broadly speaking we may divide the combination functions, f, into two groups:

Simple Combiners (S). The simple combiners are simple aggregation funcions such as product, mean, maximum and minimum. The functions, f, may also include weights and parameters which are learnt on the training data D. In general, the simple combiners are best suited for problems where the individual classifiers perform the same task and have comparable performances.

Meta-Learners (M). The meta-learners include the methods of stacked generalization and mixture of experts, where the combination function is a meta-classifier, S_0, which acts on the outputs of the individual $S_m, m \in \{1, 2, \ldots, M\}$ or on the input measurement vector $\boldsymbol{y}$. Meta-learners are more general than the simple combiners but thet are more vulnerable to problems of overfitting.

In Table 13.3 we list the most common simple combiners and meta-learners.

13.7.1 Simple Combiners

The choice of combination function f depends very much upon the nature of the belief vectors $\boldsymbol{\mu}_m, m \in \{1, 2, \ldots, M\}$, upon which it acts. For example, the maximum combination function cannot be used if the $\boldsymbol{\mu}_m$ are identity vectors. We shall therefore divide the simple combiners into three sub-classes: identity, rank and belief combiners.

Table 13.3. Combination Strategies

Name	Type	Description
Mean Vote	S	Used when each classifier S_m produces a belief vector $\boldsymbol{\mu}_m$. In this case, the mean vote for the kth class is $\frac{1}{M}\sum_{m=1}^{M}\mu_m^{(k)}$. The class with the highest average belief value in the ensemble wins.
Maximum, Minimum, Median Vote	S	Used when each classifier S_m produces a belief vector $\boldsymbol{\mu}_m$. In this case, the maximum, minimum or median vote for the kth class is $\max_m(\mu_m^{(k)})$, $\min_m(\mu_m^{(k)})$ or $\text{median}_m(\mu_m^{(k)})$. The class with the highest maximum, median or minimum belief value in the ensemble wins.
Borda count	S	Used when each classifier S_m produces a rank vector $\boldsymbol{R}_m$. In this case, the Borda count for the kth class is $B(k) = \sum_{m=1}^{M} B_m(k)$, where $B_m(k)$ is the number of classes ranked below class $C = c_k$ by the mth classifier. The class with the highest Borda count wins.
Majority Vote	S	Used when each classifier S_m produces an identity vector $\boldsymbol{I}_m$. In this case, the majority vote for the kth class is $\sum_{m=1}^{M} I_m^{(k)}$. The class with the majority vote in the ensemble wins.
Weighted Simple Combiners	S	In weighted versions of the above simple combiners, the output of each classifier S_m is given a weight w_m, where $0 \leq w_m \leq 1$ and $\sum_m w_m = 1$. The weight w_m is often a measure of the performance of S_m performance as measured on a separate validation test set.
Stacked Generalization	M	The output of the ensemble of $S_m, m \in \{1, 2, \ldots, M\}$, serves as a feature vector to a meta-classifier.
Mixture of Experts (ME)	M	The feature space is partitioned into different regions, with one expert (S_m) in the ensemble being responsible for generating the correct output within that region. The experts in the ensemble and the partition algorithm are trained simultaneously which can be performed using the EM algorithm.
Hierarchical ME	M	Mixture of experts is extended to a multi-hierarchical structures where each component is itself a mixture of experts.

Identity Combiners

The identity combiner only uses the class information contained in the identity vectors $\boldsymbol{I}_m, m \in \{1, 2, \ldots, M\}$. The *majority voter* is probably the most popular abstract-level combiner. It finds the optimal class c_{OPT} by selecting the class which is chosen most often by different classifiers. Mathematically, the optimal class is

$$c_{\text{OPT}} = \arg\max_k \sum_m I_m^{(k)} \ , \tag{13.13}$$

where $I_m^{(k)} = 1$ if the mth classifier S_m assigns O to the class $C = c_k$; otherwise $I_m^{(k)} = 0$.

A variant of the majority voter is the *weighted majority voter*. This is a majority voter in which each vector $\boldsymbol{I}_m$ is given a weight, w_m, which is learnt on a training data set D.

Example 13.8. Weighted Majority Vote [248]. Each classifier $S_m, m \in \{1, 2, \ldots, M\}$ is given a weight w_m in proportion to its estimated performance. Then the optimal class is

$$c_{\text{OPT}} = \arg\max_k \sum_{m=1}^{M} w_m I_m^{(k)} \ .$$

If the S_m are class independent, then the optimal weights are

$$w_m = \log \frac{p_m}{1 - p_m} \ ,$$

where p_m is the probability of correct classification if we were to use S_m by itself. □

Another trained abstract-level combiner is the behaviour-knowledge space (BKS) combiner [128]. In the BKS combiner we record how often the classifiers $S_m, m \in \{1, 2, \ldots, M\}$, produce a given combination of class labels. If there are K classes, then altogether there are K^M different combinations of class labels. The true class for which a particular combination of class labels is observed most often, is chosen every time that combination of class labels occurs during testing. Although, theoretically, BKS has a high performance, it is limited by the very large amounts of training data which are required.

Example 13.9. BKS Look-up Table. We consider the construction of a BKS look up table using a training set D containing N labeled objects $O_i, i \in \{1, 2, \ldots, N\}$. We assume $M = 3$ classifiers $S_m, m \in \{1, 2, 3\}$, and $K = 2$ classes $C \in \{c_1, c_2\}$. Altogether there are $L = K^M = 2^3 = 8$ different label combinations, $\omega_l, l \in \{1, 2, \ldots, 8\}$, where

$$\begin{aligned} \boldsymbol{\omega} &= (\omega_1, \omega_2, \ldots, \omega_L)^T \ , \\ &= ((1,1,1),(1,1,2),(1,2,1),(1,2,2),(2,1,1),(2,1,2),(2,2,1),(2,2,2))^T \ , \end{aligned}$$

where $\omega_l = (\alpha, \beta, \gamma)$ denotes that in the lth combination of class labels, ω_l, the classifiers S_1, S_2 and S_3 returned, respectively, the class labels $C = c_\alpha, C = c_\beta$ and $C = c_\gamma$. A possible distribution of the objects $O_i, i \in \{1, 2, \ldots, N\}$, among the $\omega_l, l \in \{1, 2, \ldots, L\}$, is given in table 13.4. Suppose a test object, O, has a class label $\Omega = \omega_l = (1, 2, 1)$, then we assign O to the class $C = c_{\text{OPT}}$:

$$\begin{aligned} c_{\text{OPT}} &= \max_k (p(C = c_k | \Omega = (1,2,1)) \ , \\ &= c_1 \ , \end{aligned}$$

Table 13.4. Distribution of training objects O_i among label combinations $\{\omega_l\}$

$C = c_k$	$n(k,l)$
c_1	100 50 76 89 54 78 87 5
c_2	8 88 17 95 20 90 95 100

$n(k,l)$ is the number of objects $O_i, i \in \{1,2,\ldots,N\}$, in D which belong to class $C = c_k$ and which have the lth combination of class labels ω_l.

where

$$p(C = c_1|\Omega = (1,2,1)) = \frac{76}{76+17} = 0.82 ,$$
$$p(C = c_2|\Omega = (1,2,1)) = \frac{17}{76+17} = 0.18 . \qquad \square$$

Rank Combiners

Rank combiners use the information contained in the rank vectors $\boldsymbol{R}_m, m \in \{1,2,\ldots,M\}$. These combiners are used when we do not believe the *a posteriori* probability distributions generated by a classifier but we do believe the relative ordering of the distributions. The Borda count is probably the most popular rank-level combiner. It works as follows: For any class c_k, let $B_m(k)$ be the number of classes ranked below the class c_k by the mth classifier S_m. Then the Borda count[8] for the class $C = c_k$ is

$$B(k) = \sum_{m=1}^{M} B_m(k) . \tag{13.14}$$

The optimum class, c_{OPT} is defined as the class with the largest Borda count:

$$c_{\mathrm{OPT}} = \arg\max_k \sum_{m=1}^{M} B_m(k) . \tag{13.15}$$

Example 13.10. Image Orientation Using Borda Count [186]. We describe an image orientation algorithm based on the Borda count. The orientation of an input image is measured in $M = 20$ different ways. Each classifier $S_m, m \in \{1,2,\ldots,M\}$, measures the orientation, θ, of an input image and returns a rank vector $\boldsymbol{R}_m = (R_m^{(1)}, R_m^{(2)}, \ldots, R_m^{(K)})^T$, where $R_m^{(k)}$ is the degree-of-support

[8] If there are no ties, then by definition, $B_m(k) = R_m^{(k)}$. In the case of ties we may let $B_m(k)$ be equal to the average number of classes below c_k if we randomly break the ties.

(expressed as a rank) that θ falls in the interval $[2\pi(k-1)/K, 2\pi k/K]$. In [186] the authors used $K = 4$. □

A variant of the Borda count is the *weighted Borda count*. This a Borda count combiner in which each vector $\boldsymbol{R}_m$ is given a weight w_m which is usually learnt on a training data set D.

Belief Combiners

The belief combiners use all of the information contained in the belief vectors $\boldsymbol{\mu}_m, m \in \{1, 2, \ldots, M\}$. The *mean*, or average, vote is a popular belief combiner. It finds the optimal class c_{OPT} by selecting the class with the highest average value. Mathematically, the optimal class is

$$c_{\text{OPT}} = \arg\max_k \frac{1}{M} \sum_m \mu_m^{(k)} \,, \tag{13.16}$$

where $\mu_m^{(k)}$ represents the belief as generated by the mth classifier for the kth class.

A variant of the mean vote is the *weighted mean* (*average*) *vote* in which each belief vector is given a weight w_m which is usually learnt.

Example 13.11. Weighted Mean Vote. The weighted mean approach is similar to the mean vote, however, the outputs of the various classifiers are multiplied with a weighting factor or

$$c_{\text{OPT}} = \arg\max_k \frac{1}{M} \sum_m^M w_m \mu_m^{(k)} \tag{13.17}$$

The weights $w_m, m \in \{1, 2, \ldots, M\}$, can be derived by minimizing the error of the different classifiers on the training set. □

In principle the belief vector $\boldsymbol{\mu}_m$ can be transformed into an *a posteriori* pdf. In this case we can interpret the weights w_m as the *a posteriori* probabilities $p(S_m|\boldsymbol{y}, I)$ and use the Bayesian model averaging technique (see Ex. 13.1). In the following example we compare the action of several different simple combiner functions.

Example 13.12. Combination Function: A Numerical Experiment [248]. Given an object O we assume the following decision profile:

$$\text{DP} = \begin{pmatrix} 0.85 & 0.01 & 0.14 \\ 0.30 & 0.50 & 0.20 \\ 0.20 & 0.60 & 0.20 \\ 0.10 & 0.70 & 0.20 \\ 0.10 & 0.10 & 0.80 \end{pmatrix} \,,$$

where the (m, k)th element of DP is defined to be the belief $\mu_m^{(k)}$ as calculated for the class, $C = c_k$, by the mth classifier, S_m. The corresponding results obtained when we use the following combination functions: product, maximum, mean, median, minimum, Borda count and majority vote, are listed in the following table together with their optimal class assignment $C = c_{\text{OPT}}$. □

Combiner f		$\widetilde{\boldsymbol{\mu}}^T$		$C = c_{\text{OPT}}$
Majority Vote	1	3	1	c_2
Borda Count	4	6.5	4.5	c_2
Product	0.0179	0.0002	0.009	c_1
Maximum	0.85	0.70	0.80	c_1
Mean	0.310	0.382	0.308	c_2
Median	0.20	0.50	0.20	c_2
Minimum	0.10	0.01	0.14	c_3

Example 13.13. Weighted Combination Functions: A Numerical Experiment [248] We repeat Ex. 13.12 assuming the classifiers S_m are assigned the following weights: $w_1 = 0.30, w_2 = 0.25, w_3 = 0.20, w_4 = 0.10, w_5 = 0.15$. In the following table we list the results obtained using the weighted majority vote and the weighted mean combination functions. □

Combiner f		$\widetilde{\boldsymbol{\mu}}^T$		$C = c_{\text{OPT}}$
Weighted Majority Vote	0.30	0.55	0.15	c_2
Weighted Mean	0.395	0.0333	0.272	c_1

13.7.2 Meta-Learners

In the meta-learners the combination function is a meta-classifier which we denote by S_0. Two common meta-learners are the stacked generalization model and the mixture of experts models which are described, respectively, in the following examples.

Example 13.14. Stacked Generalization Model [163, 299]. Fig. 13.4 illustrates the stacked generalization model, where the classifiers $S_m, m \in \{1, 2, \ldots, M\}$, are trained on the training sets D_m. The outputs of these classifiers and the corresponding true classes are then used as input/output training pairs for the meta-classifier S_0. □

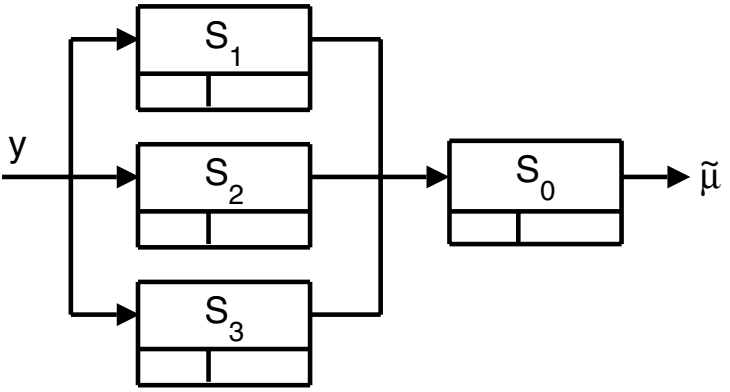

Fig. 13.4. Shows the stacked generalization. This consists of an ensemble of $M = 3$ classifiers $S_m, m \in \{1, 2, 3\}$, and a meta-classifier S_0. The input to S_0 is the belief vector $\boldsymbol{\mu}_m, m \in \{1, 2, 3\}$, generated by the S_m. The final output, as generated by S_0, is a new belief vector $\widetilde{\boldsymbol{\mu}}$, or in some cases, a new identity vector $\widetilde{\boldsymbol{I}}$.

Example 13.15. Mixture of Experts Model [135, 163]. The mixture of expert model (see Fig. 13.5) is conceptually similar, but different, to the stacked generalization model. In the mixture of experts model, the input to S_0 is the measurement vector $\boldsymbol{y}$ and the output is a weight vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_M)^T$. The final output of the mixture of experts model is a belief vector, $\widetilde{\boldsymbol{\mu}}$, which is formed by combining the belief vectors $\boldsymbol{\mu}_m, m \in \{1, 2, \ldots, M\}$, with $\boldsymbol{w}$ using a simple combiner f. The mixture of experts model is often interpreted as a

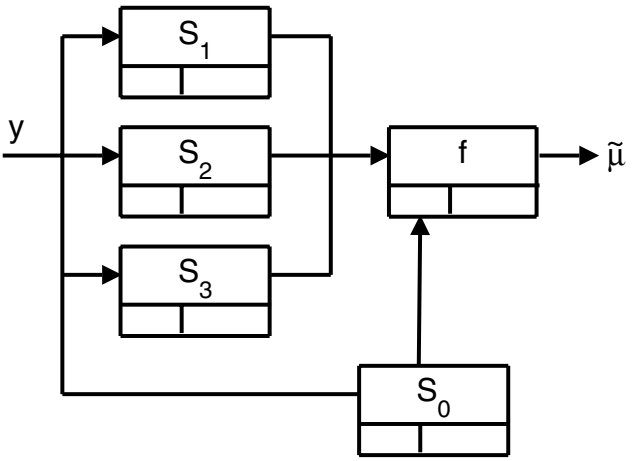

Fig. 13.5. Shows the mixture of experts model. This consists of an ensemble of $M = 3$ classifiers $S_m, m \in \{1, 2, 3\}$, and a meta-classifier S_0. The meta-classifier outputs a weight vector $\boldsymbol{w} = (w_1, w_2, w_3)^T$. The final output is formed by combining the belief vectors $\boldsymbol{\mu}_m, m \in \{1, 2, 3\}$, with the weight vector $\boldsymbol{w}$, using a simple combiner f.

classifier selection algorithm in which individual classifiers are regarded as experts in some portion of the feature space. In this case, for each measurement vector $\boldsymbol{y}$, S_0 selects the most appropriate classifier, S_n, by selecting a weight vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_M)^T$, where $w_n = 1$ and $w_m = 0, m \neq n$. □

13.8 Boosting

A recent development in ensemble learning is the idea of "boosting" [274]. Boosting has proved itself to be an effective ensemble learning algorithm. At present, several different versions of the boosting algorithm exist, some of which are listed in Table 13.5.

We first introduced the concept of boosting in Sect. 4.6, where we described it as a technique for generating an ensemble of training sets $D_m, m \in \{1, 2, \ldots, M\}$. Boosting is, however, much more than this. Boosting is a *complete* technique for ensemble learning which includes both an iterative method for generating multiple training sets D_m and an algorithm for combining the corresponding classifiers S_m [9].

The most common boosting algorithm is "Adaboost". It works on two class problems as follows. Let D denote a common training set containing N samples $\boldsymbol{y}_i, i \in \{1, 2, \ldots, N\}$, and let S denote a parametric classifier. In the mth iteration, we create a training set D_m, by randomly sampling the common training set D with replacement, using a non-uniform probability

Table 13.5. Boosting Algorithms

Name	Description
AdaBoost	The most popular boosting algorithm [274].
Real AdaBoost	A generalization of AdaBoost. Real Adaboost is now regarded as the basic boosting algorithm [274].
Modest AdaBoost	A regularized version of AdaBoost [312].
Gentle AdaBoost	A more robust and stable version of Real AdaBoost [93].
FloatBoost	An improved algorithm for learning a boosted classifier [176].
SmoteBoost	A boosting algorithm which improves the performance of classes which are under-represented in the training set D [39].
MadaBoost, EtaBoost, LogitBoost	Various robust versions of AdaBoost.
SpatialBoost	An algorithm which adds spatial reasoning to AdaBoost [11].

[9] From this point of view, bagging and wagging are also complete ensemble learning techniques, since they both contain a method for creating multiple training sets $D_m, m \in \{1, 2, \ldots, M\}$, on which the classifiers S_m are learnt, as well as an algorithm (majority vote) for combining the S_m.

distribution p_i, where $p_i = w_i / \sum w_i$. We then use D_m to learn the parameters of S. The result is a fixed classifier S_m, i. e. a classifier which does not have any free parameters. In boosting we adjust the weights $w_i, i \in \{1, 2, \ldots, N\}$, in accordance with the performance of the previous classifier, S_{m-1}, which was trained on the previous training set, D_{m-1}. By increasing the weights of misclassified samples we cause the learning algorithm to focus on different samples and thus lead to different classifiers S_m.

The following example describes the pseudo-code for creating a set of fixed classifiers $S_m, \in \{1, 2, \ldots, M\}$, by boosting a parametric classifier S.

Example 13.16. Boosting a Parametric Classifier S [71, 263]. In boosting a parametric classifier, S, we learn a series of fixed classifiers S_m on the training sets D_m. The steps in the boosting process are as follows:

1. For $m = 1$ to M perform steps 2–6.
2. Create a training set D_m. Let $w_i^{(m)}$ denote the weight associated with the object O_i. Then D_m is formed by randomly sampling the common training set D with replacement and according to the non-uniform probability distribution $p_i^{(m)} = w_i^{(m)} / \sum_{i=1}^{N} w_i^{(m)}$. For $m = 1$, the weights are: $w_i^{(1)} = 1, i \in \{1, 2, \ldots, N\}$.
3. Use D_m to learn the parameters of S. The result is the fixed classifier S_m.
4. Calculate the normalized error ϵ_m for the mth classifier S_m. If δ_i is a suitable error function, then

$$\epsilon_m = \sum_{i=1}^{N} p_i^{(m)} \delta_i \ .$$

5. Calculate weights $w_i^{(m+1)}, i \in \{1, 2, \ldots, N\}$, for the next iteration:

$$w_i^{(m+1)} = w_i^{(m)} \beta_m^{1-\delta_i} \ ,$$

   where $\beta_m = \epsilon_m / (1 - \epsilon_m)$.
6. Calculate the probability distribution for the $(m+1)$th iteration by normalizing the weights $w_i^{(m+1)}$:

$$p_i^{(m+1)} = \frac{w_i^{(m+1)}}{\sum_{i=1}^{N} w_i^{(m+1)}} \ .$$

Given the classifiers S_m and the weighted errors ϵ_m, we may calculate the *a posteriori* probability that an unknown object O belongs to the class $C = c$ as follows. Let $p_m(C = c|\boldsymbol{y}, I)$ denote the *a posteriori* probability that O belongs to $C = c$ as estimated by S_m. Then the final (boosted) *a posteriori* probability that O belongs to the class $C = c$ is

$$p(\boldsymbol{y}|C = c, I) = \frac{1}{1 + \prod_{m=1}^{M} \beta_m^{2r-1}} \ ,$$

where

$$r = \sum_{m=1}^{M} p_m(C = c|\boldsymbol{y}, I) \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right) / \sum_{m=1}^{M} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right) . \qquad \square$$

13.9 Recommendations

The various ensemble generation functions have been compared for a wide variety of different classification problems [248]. The consensus is that boosting usually achieves better generalization than bagging and wagging but it is somewhat more sensitive to noise and outliers. Regarding the combination functions: In general, the preferred functions are majority vote, Borda count and averaging.

13.10 Software

ENTOOL. A matlab toolbox for perfoming ensemble modeling.

GML. A matlab toolbox for performing variants of the AdaBoost algorithms: Real, Modest and Gentle-AdaBoost algorithms.

HME. A matlab toobox for implementing hierarchical mixture of experts algorithm.

MIXLAB. A matlab toolbox for implementing mixture of experts algorithm.

STPRTOOL (Statistical Pattern Recognition Toolbox). A matlab toolbox for performing statistical pattern recognition. The toolbox contains m-files for boosting and cross-validation.

13.11 Further Reading

Two modern up-to-date reviews of ensemble learning are [248] and [256]. In addition, Kuncheva [163] has written a full length book on the subject. For a good overview of boosting see the review article by Schapire [274]. For additional modern references see the references listed in Table 13.5. Recently boosting has been successfully combined with the naive Bayes' classifier [71, 263, 313].

Part IV

Sensor Management

14

Sensor Management

14.1 Introduction

In this chapter we complete our study of multi-sensor data fusion by analyzing the control application/resource management block shown in Fig. 1.2. To make our discussion more concrete, we shall consider the case when decisions made by the control application are fed back to the sensors. In this case, the control block is more commonly known as a *sensor management* (SM) block. Formally we define sensor management as "a process that seeks to manage, or coordinate, the use of a set of sensors in a dynamic, uncertain environment, to improve the performance of the system" [337].

In Fig. 14.1 we show our multi-sensor data fusion system with the sensor, data fusion and sensor management blocks. The sensor observations $O_i, i \in$

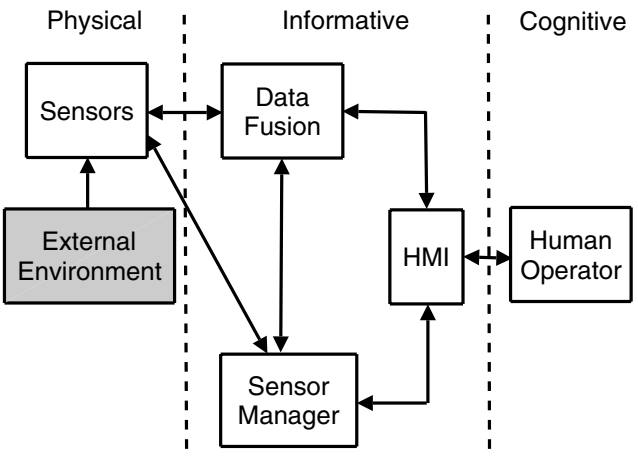

Fig. 14.1. Shows a complete multi-sensor data fusion framework with a sensor manager.

$\{1, 2, \ldots, N\}$, are sent from the sensors $S_m, m \in \{1, 2, \ldots, M\}$, to the data fusion block for processing. Information from the data fusion block is then passed, via the human-machine interface (HMI), to the human operator who monitors the entire scene. The inputs to the Sensor Manager may come from the data fusion block and also from operator via the HMI.

In general, the sensor observations, $\boldsymbol{y}$, are transmitted with minimal delay from the sensor block to the data fusion block via a real-time service (RS) interface. On the other hand, the information which is passed between the sensor manager and the sensor and data fusion blocks may be subjected to a significant delay. In this case, the information is transmitted via the slower diagnostic and management (DM) and configuration and planning (CP) interfaces. See Sect. 2.4.1.

14.2 Hierarchical Classification

Sensor management is clearly a very broad concept which involves many different issues which may help enhance the performance of the multi-sensor data fusion system. Broadly speaking, we may divide the field of sensor management into a hierarchy consisting of three levels, or categories: sensor control, sensor scheduling and resource planning [228]. We shall now consider each hierarchical level in turn.

14.2.1 Sensor Control

Sensor control is the lowest hierarchical level in sensor management. In it which we focus on the individual control of each sensor, including the specification of its parameters. At the sensor control level the SM acts as a classic feedback, or control, mechanism whose goal is to optimize the performance of the sensor and data fusion blocks given the present sensor measurements $\boldsymbol{y}$.

The following example illustrates this aspect of the SM.

Example 14.1. Building Security [235, 345]. The security of buildings is an increasing concern nowadays. One aspect of this concerns the level of access required by employees, clients and customers. For example, access to security, financial and sensitive information is usually restricted to specific employees who work in one, or more, exclusive regions of the building. On the other hand, other regions of the building must be open to employees, clients and cuustomers without any restrictions.

One approach to this problem is to employ a network of biometric sensors interfaced to door-locking mechanisms. The security level corresponding to the job description and biometric features constitute the information gathered when an employee is enrolled into the system. The biometric features are collected and matched against the enrollment features when the individual requests access. Control of the door-locking mechanisms is optimized for a

given criteria, e. g. to minimize the number of mistakes, i. e. to minimize the sum of the number of genuine employees who are refused access and the number of imposters who are allowed access. □

14.2.2 Sensor Scheduling

Sensor scheduling is the middle hierarchical level in sensor management. In it we focus on the actual tasks performed by the sensors and on the different sensor modes. At the sensor scheduling level the SM prioritizes the different tasks which need to be performed and determines when, and how, a sensor should be activated. An important task at this level is the sensor schedule.

The following example illustrates this aspect of the SM.

Example 14.2. Meteorological Command and Control [358]. Distributed Collobarative Adaptive Sensing (DCAS) is a new paradigm for detecting and predicting hazardous weather. DCAS uses a dense network of low-powered radars which periodically sense a search volume V which occupies the lowest few kilometers of the earth's atmosphere. At the heart of the DCAS system is a meteorological Command and Control (MCC) unit which performs the systems' main control loop. An important function performed by the MCC is the allocation/optimization processes that determines the strategy for taking radar measurements during the next radar scan. Within each voxel in V and each meteorological object which has been tracked until now in V has an utility that represents the value of scanning that voxel/object during the next scan. The utility value weights considerations such as the time since the voxel/object was last scanned, the object type (e. g. scanning an areas with a tornado vortex will have higher utility than sensing clear air) and user-based considerations, such as the distance from a population center (e. g. among two objects with identical features, the one closer to a population center will have higher utility). □

14.2.3 Resource Planning

Resource planning is the highest hierarchical level in sensor management. In it we focus on tasks such as the placement of the sensors, or the optimal mixture of the sensors required for a given task.

The following example illustrates this aspect of the SM.

Example 14.3. Mobile Sensors Observing an Area of Interest [337]. In this application we consider the coordination of several autonomous sensors $S_m, m \in \{1, 2, \ldots, M\}$. The sensors S_m, are required to co-operatively monitor an area of interest (Fig. 14.2). Each sensor S_m has its own dynamics (specified by a velocity vector $\mathbf{V}_m$) and can only perceive a limited local area A_m. The local areas can be shared by the sensors. Thus, a local picture from one sensor can be used to direct the attention of other sensors. The sensor manager is

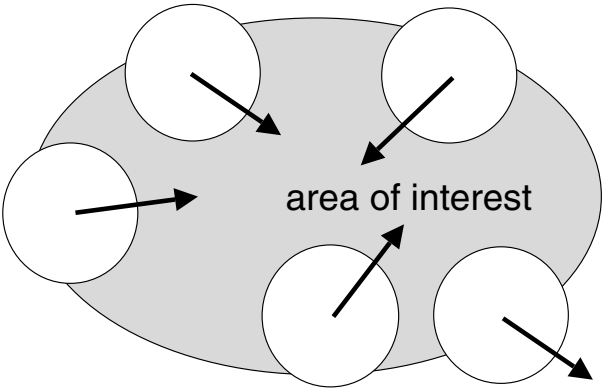

Fig. 14.2. Shows five mobile sensors $S_m, m \in \{1, 2, \ldots, 5\}$ cooperatively monitoring an area of interest (shown shaded). Each sensor S_m has a velocity $\mathbf{V}_m$ (denoted by an arrow) and can only monitor a local area A_m (denoted by a white circle).

responsible for coordinating the movements and sensing actions of the sensors so that we develop an optimal picture of the entire surveillance area with minimal consumption of time and resources. □

14.3 Sensor Management Techniques

Over the years a variety of SM techniques have been used for solving the problem of SM. We conclude the book by briefly reviewing two of these techniques.

14.3.1 Information-Theoretic Criteria

From the information-theoretic point of view, multi-sensor data fusion is concerned with increasing the information, i. e. reducing the uncertainty, about the state of the external world, or environment. In this case, the task of sensor management is to optimize the multi-sensor data fusion process such that the greatest possible amount of information is obtained whenever a measurement is made.

In the following example we consider the optimal selection of a sensor, using an information-theoretic criteria.

Example 14.4. Sensor Selection Using Information-Theoretic Criteria [162, 321]. We assume a target is known to be present in a given surveillance area. Let $\pi(\boldsymbol{x}|I)$ denote the *a priori* location of the target. Suppose $S_m, m \in \{1, 2, \ldots, M\}$, denotes a set of M sensors, whose observation likelihoods are $p(\boldsymbol{y}_m|\boldsymbol{x}, I)$. Our aim is to select the sensor S_m whose observation will maximize the mutual information $MI(\boldsymbol{x}, \boldsymbol{y}_m)$, where $\boldsymbol{x}$ denotes the unknown location

of the target and $\boldsymbol{y}_m$ denotes the observation $\boldsymbol{y}_m$ predicted for the sensor S_m. According to (5.4), the mutual information, $MI(\boldsymbol{x}, \boldsymbol{y}_m)$, is given by

$$MI(\boldsymbol{x}, \boldsymbol{y}_m) = \int p(\boldsymbol{x}, \boldsymbol{y}_m|I) \log \frac{p(\boldsymbol{x}, \boldsymbol{y}_m|I)}{p(\boldsymbol{x}|I)p(\boldsymbol{y}_m|I)} d\boldsymbol{x} d\boldsymbol{y}_m \ ,$$

where $p(\boldsymbol{x}, \boldsymbol{y}_m|I) = p(\boldsymbol{y}_m|\boldsymbol{x})p(\boldsymbol{x}|I)$ and $p(\boldsymbol{y}_m|I) = \int p(\boldsymbol{x}, \boldsymbol{y}_m|I)d\boldsymbol{x}$. In this case, we choose the observation, i. e. the sensor, which maximizes the mutual information $MI(\boldsymbol{x}_i, \boldsymbol{y}_m)$:

$$m_{\text{OPT}} = \arg\max MI(\boldsymbol{x}, \boldsymbol{y}_m) \ . \qquad \square$$

The above expression for the mutual information receives a simple form if we assume Gaussian distributions for the state of the target.

Example 14.5. Sensor Selection Using Mutual Information. Assuming a Gaussian distribution for the state of a given target O, then

$$MI(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \log(|P_{\boldsymbol{x}}|/|P_{\boldsymbol{y}}|) \ , \tag{14.1}$$

where $P_{\boldsymbol{x}}$ and $P_{\boldsymbol{y}}$ are, respectively the covariance matrices before and after a measurement has been made. $\square$

14.3.2 Bayesian Decision-Making

From the viewpoint of decision theory (see Chapt. 12), we may regard sensor management as a decision-making task in which our aim is to minimize a given loss function. As an example we shall consider the sensor control of the biometric sensors in Ex. 14.1.

Example 14.6. An Adaptive Multimodal Biometric Management Algorithm [235, 310]. We consider M independent biometric sensors $S_m, m \in \{1, 2, \ldots, M\}$. The task of identifying an unknown person O as a hypothesis testing problem, with the following two hypotheses:

$H = h_1$. The unknown person O is an imposter.
$H = h_2$. The unknown person O is genuine.

We suppose each sensor S_m receives a measurement vector $\boldsymbol{y}_m$ from O and outputs the decision variable $U_m \in \{u_1, u_2\}$, where

$$U_m = \begin{cases} u_1 \text{ if } p(U_m = u_1|H = h_1) \geq \lambda_m p(U_m = u_2|H = h_2) \ , \\ u_2 \text{ otherwise} \ . \end{cases}$$

where λ_m is an appropriate threshold.

Assuming each of the biometric sensors are idependent, then the optimal fusion rule [309] can be implemented by forming a weighted sum of the incoming local decisions $U_m, m \in \{1, 2, \ldots, M\}$, and then comparing it with a

threshold t. The weights and the threshold are determined by the reliability of the decisions, i. e. by the probabilities of false alarm and miss of the sensors S_m. In mathematical terms, the output decision variable is $\widetilde{U} = u_{\mathrm{OPT}}$, where

$$u_{\mathrm{OPT}} = \begin{cases} u_1 \text{ if } \left[\sum_{m=1}^{M} \left(z_m \log \frac{1-p_m^M}{p_m^F} + (1-z_m) \log \frac{p_m^M}{1-p_m^F} \right) \right] \geq t \ , \\ u_2 \text{ otherwise} \end{cases}$$

where

$$z_m = \begin{cases} 1 \text{ if } U_m = u_1 \ , \\ 0 \text{ otherwise} \ , \end{cases}$$

and p_m^F, and p_m^M are, respectively, the probabilities of false alarm and miss for the sensor S_m,

$$p_m^F = p(U_m = u_2 | H = h_1) \ ,$$
$$p_m^M = p(U_m = u_1 | H = h_2) \ .$$

We optimally choose the threshold t in order to minimize the cost of a output decision $\widetilde{U}$. This cost depends on the *a priori* probabilities $p(H = h_1|I)$ and $p(H = h_2|I)$ and on the loss function which is used [310]. □

14.4 Further Reading

Recent comprehensive reviews of sensor management are [198, 228, 337]. Ref. [219] describes a novel approach to sensor management based on e-commerce. A detailed description of sensor managment and in particular sensor scheduling, in a multi-function radar system is given in [213].

14.5 Postscript

In writing this book our aim was to provide the reader with an introduction to the basic theories and techniques in multi-sensor data fusion. The theories and techniques may be formulated in many different frameworks, but for reasons explained in the preface, we have, by and large, restricted ourselves to the Bayesian framework. The reader should, however, keep in mind that multi-sensor data fusion is a pragmatic activity which is driven by practicalities. When circumstances dictate the engineer may abandon the Bayesian framework and adopt an alternative framework instead.

During the time this book was being written we have witnessed a growing interest in multi-sensor data fusion: Many commercial applications now employ multi-sensor data fusion in one form or another. We have tried to convey this growing interest by giving examples drawn from a wide range of real-world applications and the reader is now encouraged to review these examples.

Part V

Appendices

A

Software Sources

In the following table we give a list of all the matlab toolboxes and m-files which appear in the book. Included in the list are full details regarding the software and authors. With this information the user should find it possible to locate any given software using an appropriate internet search engine.

Table A.1. Multi-Sensor Data Fusion: Matlab Software

Name	T'box/ m-file	Authors	Description
BAYESLAB	T'box	Bin Shi, H.M. Chen, Ying Hung	Bayesian inference.
BFD	T'box	T. Pena-Centemo, N. D. Lawrence	Non-linear discriminant analysis.
BNT	T'box	Kenvin Patrick Murphy	Bayesian networks [223, 224].
DACE	T'box	Soren N. Lophaven, Hans Brun Nielsen, Jacob Sondergaard	Kriging Analysis.
DTW	m-file	Timothy Felty	Dynamic time warping algorithm.
EFICA	m-file	Zbynek Koldovsky, Petr Tichavsky	Independent component analysis [157].
ENTOOL	T'box	C. Merkwirth, J. Wichard	Ensemble modeling.
FASTICA	T'box	Hugo Gavert, Jarmo Hurri, Jaakko Sarela, Aapo Hyvarinen	Independent component analysis (ICA).
GML	T'box	A. Vezhnevets, V. Vezhnevets	Boosting toolbox [312].
GMMBAYES	T'box	Joni Kamarainen, Pekka Paalanen	Gaussian mixture model.

Table A.1. (Continued)

Name	T'box/ m-file	Authors	Description
GPCLASS	T'box	David Barber, C. K. I. Williams	Bayesian classification.
GPML	T'box	Carl Edward Rasmussen, Chris Williams	Gaussian regression and classification [258].
GLMLAB	T'box	Peter K. Dunn	Generalized linear models.
HME	T'box	David R. Martin	Hierarchical mixture of Experts.
LIBRA	T'box	Sabine Verboven, Mia Hubert	Robust statistical techniques [311].
LIBSVM	T'box	Chih-Chung Chang, Chih-Jen lin	Support Vector Machine.
KALMTOOL	T'box	Magnus Norgaard	Kalman filter.
KDE	T'box	Alex Ihler, Mike Mandel	Kernel density estimation.
KFT	T'box	Kevin Patrick Murphy	Kalman filter.
LIGHTSPEED	T'box	Tom Minka	Fast m-files to replace slow low-level matlab functions.
MCMCSTUFF	T'box	Aki Vehtari	MCMC for Bayesian inference.
MILCA	T'box	Sergey Astakhov, Peter Grassberger, Alexander Kraskov, Harald Stogbauer	Independent component analysis. Includes an accurate mutual information estimator.
MIXLAB	T'box	Perry Moerland	Mixture of expert models.
MUTIN	m-file	Petr Tichavsky	Mutual information estimator [55].
NETLAB	T'box	Ian Nabney	Neural network pattern recognition [225].
RADICAL	T'box	Erik G. Learned-Miller	Independent component analysis[165].
STPRTOOL	T'box	Vojtech Franc, Vaclav Hlavac	Statistical pattern recognition.
UPF	T'box	Nando de Freitas	Unscented particle filter.
VBGP	T'box	Mark Girolami, Simon Rogers	Variational Bayesian inference.

B

Background Material

B.1 Probability Theory

Table B.1 contains a summary of basic formulas used in probability theory.

Table B.1. Basic Formulas in Probability Theory

Name	Formula
Probability Density Function (pdf)	$p(y\|I)dy = P(y \leq Y \leq y + dy\|I)$.
Normalization	$\int p(y\|I)dy = 1$.
Expectation of $f(Y)$	$E(f(Y)) = \int f(y)p(y\|I)dy$.
Expected Value	$E(Y) = \int yp(y\|I)dy$.
Moment of Order r	$M_r(Y) = \int y^r p(y\|I)dy$.
Variance	$\sigma^2 = \int (y - E(Y))^2 p(y\|I)dy$.
Product Rule	$p(x, y\|I) = p(x\|y, I)p(y\|I)$.
Independence	$p(x, y\|I) = p(x\|I)p(y\|I)$.
Marginalization	$p(x\|I) = \int p(x, y\|I)dy$.
Decomposition	$p(x\|I) = \int p(x\|y, I)p(y\|I)dy$.
Bayes' Rule	$p(x\|y, I) = p(y\|x, I)p(x\|I)/p(y\|I)$;
	$p(x\|y, z, I) = p(x\|I)p(y\|x, I)p(z\|x, y, I)/(p(y\|I)p(z\|I))$.
Likelihood	$L = p(y\|x, I)$.

In the table X and Y denote two random variables whose instantiations are, respectively, x and y and I describes any background information we may have concerning the problem in hand.

B.2 Linear Algebra

Table B.2 contains a summary of elementary results in linear algebra.

Table B.2. Elementary Results in Linear Algebra

Name	Formula
Definitions	x is a 1×1 scalar; $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^T$ is a $m\times 1$ column vector; and $\boldsymbol{A} = \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1n} \\ A_{21} & A_{22} & \ldots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \ldots & A_{mn} \end{pmatrix}$ is a $m \times n$ matrix; $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n)$, where $\boldsymbol{a}_i = (A_{1i}, A_{2i}, \ldots, A_{mi})^T$.
Transpose	$(\boldsymbol{A}^T)_{ij} = A_{ji}$
Multiplication	$(\boldsymbol{A}\boldsymbol{A}^T)_{ij} = \sum_k A_{ik}(\boldsymbol{A}^T)_{kj} = \sum_k A_{ik}A_{jk}$
Dimensions	$\boldsymbol{x}^T\boldsymbol{x}$ is a 1×1 scalar; $\boldsymbol{x}\boldsymbol{x}^T$ is a $m \times m$ scalar; $\boldsymbol{A}\boldsymbol{x}$ is a $m \times 1$ vector; $\boldsymbol{A}\boldsymbol{A}^T$ is a $m \times m$ matrix; $\boldsymbol{A}\boldsymbol{A}^T$ is a $n \times n$ matrix; $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ is a 1×1 scalar.
Combinations	$(\boldsymbol{A}\boldsymbol{B})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{C})$; $\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{A}\boldsymbol{C}$; $(\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$; $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$.

B.3 Square Matrices

Table B.3 contains a summary of elementary results on square matrices.

Table B.3. Basic Formulas Used in Square Matrix Theory

Name	Formula
Definitions	$\boldsymbol{B} = \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \dots & B_{nn} \end{pmatrix}$ is a $n \times n$ square matrix; $\boldsymbol{D} = \begin{pmatrix} D_{11} & 0 & \dots & 0 \\ 0 & D_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & D_{nn} \end{pmatrix} = \text{diag}(D_{11}, D_{22}, \dots, D_{nn})$ is a $n \times n$ diagonal matrix and $\boldsymbol{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} = \text{diag}(1, 1, \dots, 1)$ is a $n \times n$ identity matrix.
Identity Results	$\boldsymbol{Ix} = \boldsymbol{x}$; $\boldsymbol{IB} = \boldsymbol{B} = \boldsymbol{BI}$; $\boldsymbol{X}^T\boldsymbol{I} = \boldsymbol{x}^T$.
Inverse	$\boldsymbol{B}^{-1}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{B}^{-1} = \boldsymbol{I}$; $(\boldsymbol{B}^{-1})^{-1} = \boldsymbol{B}$; $(\boldsymbol{BC})^{-1} = \boldsymbol{C}^{-1}\boldsymbol{B}^{-1}$; $(\boldsymbol{B}^{-1})^T = (\boldsymbol{B}^T)^{-1}$.
Determinant	$\lvert\boldsymbol{B}\rvert = \prod_{i=1}^{n} \lambda^{(i)}$; $\lvert\boldsymbol{BC}\rvert = \lvert\boldsymbol{B}\rvert\lvert\boldsymbol{C}\rvert$; $\lvert x\rvert = x$; $\lvert x\boldsymbol{B}\rvert = x^n\lvert\boldsymbol{B}\rvert$; $\lvert\boldsymbol{B}^{-1}\rvert = 1/\lvert\boldsymbol{B}\rvert$.
Trace	$\text{Tr}(\boldsymbol{B}) = \sum_{i=1}^{n} B_{ii} = \sum_{i=1}^{n} \lambda^{(i)}$; $\text{Tr}(\boldsymbol{BCD}) = \text{Tr}(\boldsymbol{DBC}) = \text{Tr}(\boldsymbol{CDB})$; $\boldsymbol{x}^T\boldsymbol{Bx} = \text{Tr}(\boldsymbol{x}^T\boldsymbol{Bx}) = \text{Tr}(\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{B})$.
Eigenvector Equation	$\boldsymbol{B}\boldsymbol{u}^{(i)} = \lambda^{(i)}\boldsymbol{u}^{(i)}$, where $\boldsymbol{u}^{(i)}$ is the ith $n \times 1$ (eigen)vector.

References

1. Abdulla W. H., Chow D., Sin G. (2003) Cross-words reference template for DTW-based speech recognition systems. Proceedings of the IEEE Technology Conference TENCON 2003, Bangalore, India
2. D'Agostini G. (1999) Sceptical combinations of experimental results: General considerations and applications to ϵ'/ϵ. European Organization for Nuclear Research CERN-EP/99-139
3. D'Agostini G. (2003) Bayesian Reasoning in Data analysis. World Scientific, Singapore
4. D'Agostini G. (2003) Bayesian inference in processing experimental data: principles and basic applications. Reports on Progress in Physics **66**: pp. 1383–1419
5. D'Agostini G. (2004) From Observations to Hypotheses. Probabilistic Reasoning versus Falsificationism and its Statistical Variations. Invited talk at 2004 Vulcano Workshop on Frontier Objects in Astrophysics and Particle Physics, Vulcano, Italy, May 24–29, 2004
6. Anthony R. (1995) Principles of Data Fusion Automation. Published by Artech House, Inc.
7. Appriou A., Ayoun A., Benferhat S., Besnard P., Cholvy L., Cooke R., Cuppens F., Dubois D., Fargier H., Grabisch M., Kruse R., Lang J., Moral S., Prade H., Saffiotti, Smets P., Sossai C. (2001) Fusion: general concepts and characteristics. International Journal of Intelligent Systems **16**: pp. 1107–1134
8. Archambeau C. (2005) Probabilistic models in noisy environments. PhD thesis, Departement d'Electricite, Universite Catholique de Louvain, Belgium
9. Archambeau C., Delannay N., Verleysen M. (2006) Robust probabilistic projections. Proceedings of the 23rd International Conference on Machine Learning, Pittsburg, PA
10. Arulampalam M. S., Maskell S., Gordon N., Clapp T. (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing **50**: pp. 174–188
11. Avidan S. (2006) SpatialBoost: Adding spatial reasoning to AdaBoost. Proceedings 9th European Conference on Computer Vision, Graz, Austria, May 7-13, pp. 780–785
12. Bar-Shalom Y., Campo L. (1986) The effect of the common process noise on the two sensor fused track covariance. IEEE Transactions on Aerospace and Electronic Systems **22**: pp. 803–805

13. Bar-Shalom Y., Li X. (1995) Multitarget-Multisensor Tracking: Principles and Techniques. YBS Publishing, Storrs, CT
14. Bauer G. (2001) Transparent fault tolerance in a time-triggered architecture. PhD thesis. Institut fur Technische Informatik, Technischen Universitat Wien
15. Beal M. J. (2003) Variational algorithms for approximate Bayesian inference. PhD thesis, University of London, UK
16. Behloul F., Lelieveldt B. P. E., Boudraa A., Janier M., Revel D. and Reiber J. H. C. (2001) Neuro-fuzzy systems for computer-aided myocardial viability assessment. IEEE Transactions on Medical Imaging **20**: 1302–1313
17. Belhumeur P. N., Hespanha J., Kriegman D. J. (1997) Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**: pp. 711–720
18. Bellot D., Boyer A., Charpillet F. (2002) A new definition of qualified gain in a data fusion process: application to telemedicine. Proceedings Fifth International Conference on Information Fusion, Annapolis, Maryland, July 2002
19. Bender A., Glen R. C. (2004) Molecular similarity: a key technique in molecular informatics. Org. Biomol. Chem. **2**: pp. 3204–3218
20. Bennett P. N. (2003) Using asymmetric distributions to improve text classifier probability estimates. SIGIR '03 July 28–Aug 1, 2003, Toronto, Canada
21. Berger J. (1994) An overview of robust Bayesian analysis. Test **3**: pp. 5–124
22. Bhanu B., Zhou X. (2004) Face recognition from face profile using dynamic time warping. In: Proceedings of the Seventeeth International Conference on Pattern Recognition (ICPR '04), **4**: pp. 499–502
23. Bishop C. M., Svensen M. (2004) Robust Bayesian mixture modelling. Proceedings of European Symposium on Artificial Neural Networks, Bruges, Belgium, 28-30 April, pp. 69–74
24. Blackman S. S., Popoli R. F. (1999) Design and analysis of modern tracking Systems. Published by Artech House, Norwood, MA
25. Boast C. W., Baveye P. (2006) Alleviation of an indeterminacy problem affecting two classical iterative image thresholding algorithms. International Journal Pattern Recognition and Artificial Intelligence **26**: pp. 1–14
26. Bouchard G., Celeux G. (2006) Selection of generative models in classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**: pp. 544-554
27. Boudjemaa R., Forbes A.B. (2004). Parameter estimation methods for data fusion. National Physical Laboratory Report No. CMSC 38–04.
28. Boulle M. (2006) Regularization and averaging of the selective naive Bayes classifier. In: Proceedings of 2006 International Joint Conference on Neural Networks, Vancouver, Canada July 16–21, pp. 2989-2997
29. Brainard D. H., Freeman W. T. (1997) Bayesian color constancy. Journal Optical Society of America A **14**: 1393–1411
30. Brooks R. R., Iyengar S. S. (1998) Multi-Sensor Fusion: Fundamentals and Applications with Software. Published by Prentice-Hall, New Jersey, USA
31. Brown R. G., Hwang P. Y. C. (1997) Introduction to random signal analysis and Kalman filtering. Published by John Wiley and Sons Ltd
32. Burnham K. P., Anderson D. R. (1998) Model Selection and Inference: A practical information-theoretic approach. Springer-Verlag, New York, USA
33. Butterly P. J. (1971) Position finding with empirical prior knowledge. IEEE Transactions on Aerospace and Electronic Systems **8**: pp. 142–146

34. Camastra F. (2004) Kernel methods for unsupervised learning. PhD thesis, University of Genova, Italy
35. Can A., Stewart C. V., Roysam B., Tanenbaum H. L. (2002). A feature-based technique for joint, linear estimation of high-order image-to-mosaic transformations: mosaicing the curved human retina. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**: pp. 412–419
36. Can A., Stewart C. V., Roysam B., Tanenbaum H. L. (2002). A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**: pp. 347–364
37. Carr J. R., de Miranda F. P. (1998) The semivariogram in comparison to the co-occurence matrix for classification of image texture. IEEE Transaction on Geoscience and Remote Sensing **36**: pp.1945–1952
38. Chang K., Saha R., Bar-Shalom Y. (1997) On optimal track-to-track fusion. IEEE Transactions on Aerospace and Electronic Systems **33**: pp. 1271–1276
39. Chawla N. V., Lazarevic A., Hall L. O., Bowyer K. W. (2003) SmoteBoost: improving prediction of the minority class in boosting. Seventh European Conference on Principles and Practice of Knowledge Discovery in DataBases (PKDD) pp. 107–119
40. Chen Z. (2003) Bayesian filtering: from Kalman filters to particle filters, and beyond. Adaptive Systems Laboratory, Communications Research Laboratory, McMaster University,Canada
41. Chen S. C., Zhu Y. L., Zhang D. Q., Yang J. Y. (2005) Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA. Pattern Recognition Letters **26** pp. 1157–1167
42. Chen L., Arambel P. O., Mehra R. K. (2002) Estimation under unknown correlation: covariance intersection revisited. IEEE Transactions on Automatic Control **47**: pp. 1879–1882
43. Chen H-M, Arora M. K., Varshney P. K. (2003) Mutual information based image registration for remote sensing data. International Journal Remote Sensing **24**: pp. 3701–3706
44. Cheng H. (2003) Temporal registration of video sequences (2003) Proceedings ICASSP '03, Hong Kong, China, April 2003
45. Cheng H. (2004) A review of video registration for watermark detection in digital cimema applications. Proceedings ISCAS, Vancouver, British Columbia, Canada pp. 704–707
46. Chen S., Yang X. (2004) Alternative linear discriminant classifier. Pattern Recognition **37**: pp. 1545–1547
47. Choudrey R. A. (2002) Variational methods for Bayesian independent component analysis. PhD thesis, University of Oxford.
48. Chow T. W. S., Huang D. (2005) Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. IEEE Transactions on Neural Networks **16**: pp. 213–224
49. Cipra T., Romera R. (1997) Kalman filter with outliers and missing observations. Test **6**: pp. 379–395
50. Cohen I, Goldszmidt M. (2004) Properties and benefits of calibrated classifiers. Proceedings of 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, pp. 125–136
51. Cox I. J. (1993) A review of statistical data association techniques for motion correspondence. International Journal Computer Vision **10**: pp. 53–66

52. Cressie N. A. C. (1993) Statistics for spatial data. John Wiley and Sons, USA
53. Cunningham P. (2005) Overfitting and diversity in classification ensembles based on feature selection. Technical Report TCD-CS-2005-17, Department of Computer Science, Trinity College, Dublin, Ireland
54. Darbellay G. A. (1999) An estimator for the mutual information based on a criterion for independence. Journal of Computational Statistics and Data Analysis **32**: pp. 1–17
55. Darbellay G. A., Vajda I. (1999) Estimation of the information by an adaptive partitioning of the observation space. IEEE Transactions on Information Theory **45**: pp. 1315–1321
56. Dasarathy B. V. (1994) Decision Fusion. Published by IEEE Computer Society Press, Los Alamitos, CA 90720–1264
57. Date A. I. (2003) Most honourable remembrance: The life and work of Thomas Bayes. Springer-Verlag, New York.
58. David W. I. F., Sivia D. S. (2001) Background estimation using a robust Bayesian analysis. Journal of Applied Crystallography **34**: pp. 318–324
59. Deller Jr., J. R., Proakis J. G., Hansen J. H. L. (1993) Discrete-time processing of speech signals. Published by Macmillan, New York, USA
60. Delvai, Eisenmann U., Elmenreich W. (2003) A generic architecture for integrated smart transducers. Proceedings 13th International Conference on Field Programmable Logic and Applications, Lisbon, Portugal, pp. 753–744
61. Denison D. G. T., Holmes C., Mallick B. K., Smith A. F. M. (2002) Bayesian Methods for Nonlinear Classification and Regression. Published by John Wiley and Sons, Chichester, England
62. Dodier R. H. (1999) Unified prediction and diagnosis in engineering systems by means of distributed belief networks. PhD thesis, University of Colarado
63. Domingos P., Pazzani M. (1997) On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning **29**: pp. 103–130
64. Dose V. (2003) Bayesian inference in physics: case studies. Reports on Progress in Physics **66**: pp. 1421–1461
65. Dose V., Linden, Von der, W. (1999) Outlier tolerant parameter estimation. In: Maximum Entropy and Bayesian Methods (Eds. Dose V. *et. al.*). Published by Kluwer Academic Publishers, Dordrecht
66. Duan Z., Han C., Li X. R. (2004) Comments on "Unbiased converted measurements for tracking". IEEE Transactions on Aerospace and Electronic Systems **40**: pp. 1374–1376
67. Duong T. (2004) Bandwidth selectors for multivariate kernel density estimation. Phd thesis, University of Western Australia
68. Durrant-Whyte H. F. (1987) Consistent integration and propagation of disparate sensor observations. International Journal of Robotics Research **6**: pp. 3–24
69. Durrant-Whyte H. F. (1988) Sensor models and multisensor integration. International Journal of Robotics Research **7**: pp. 97–113
70. Eidson J. C., Lee K. (2003) Sharing a common sense of time. IEEE Instrumentation and Measurement Magazine Sept. 2003 pp. 26–32
71. Elkan C. (1997) Boosting and naive Bayes learning. Technical Report CS97-557, September 1997, University of California San Diego
72. Elmenreich W. (2002) An introduction to sensor fusion. Institut fur Technische Informatik, Technischen Universitat Wien, Research Report 47/2001

73. Elmenreich W. (2002) Sensor fusion in time-triggered systems PhD thesis, Institut fur Technische Informatik, Technischen Universitat Wien
74. Elmenreich W., Haidinger W., Kopetz H. (2001) Interface design for smart transducers. IEEE Instrumentation and Measurement Technology Conference (IMTC), Budapest, Hungary, **3**: pp. 1643–1647
75. Elmenreich W., Haidinger W., Kopetz H., Losert T., Obermaisser R., Paulitsch M., Trodhandl C. (2002) DSoS IST-1999-11585 Dependable systems of Systems. Initial Demonstration of Smart Sensor Case Study. A smart sensor LIF case study: autonomous mobile robot. 31 March, 2002
76. Elmenreich W., Pitzek S. (2001) The time-triggered sensor fusion model. Proceedings of the 5th IEEE International Conference on Intelligent Engineering Systems, pp. 297–300, Helsinki, Finland
77. Elmenreich W., Pitzek S. (2003) Smart transducers-principles, communications, and configuration. Proceedings 7th IEEE International Conference on Intelligent Engineering Systems (INES) pp. 510–515, Assuit, Luxor, Egypt.
78. Evans M., Swartz T. (1995) Methods for approximating integrals in statistics with special emphasis on Bayesian integration problems. Statistical Science **10**: pp. 254–272
79. Falk T. H., Chan W-Y (2006) Nonintrusive speech quality estimation using Gaussian mixture models. IEEE Signal Processing Letters **13**: pp. 108–111
80. Fearnhead P. (2005) Exact Bayesian curve fitting and signal segmentation. IEEE Transactions on Signal Processing. **53**: pp. 2160–2166
81. Fern X. Z., Brodley C. E. (2003) Random projection for high dimensional data clustering: a cluster ensemble approach. Proceedings 20th International Conference on Machine Learning (ICML), Washington, DC, USA, pp. 186–193
82. Ferreira J. T. A. S., Denison D. G. T., Hand D. J. (2001) Data mining with products of trees. In: Advances in Intelligent data analysis (Eds. Hoffman F., Hand D. J., Adams N., Fisher D. and Guimaraes G.) pp. 167-176. Published by Springer, Berlin, Germany
83. Figueiredo M., Jain A. K. (2002) Unsupervised learning of finite mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**: pp. 381–396
84. Fitzgerald W. J., Godsill S. J., Kokaram A. C., Stark J. A. (1999) Bayesian methods in signal and image processing. In: Bayesian Statistics **6** (Eds. J. M. Bernardo *et. al.*). Published by Oxford University Press, Oxford, England
85. Flament M., Fleury G., Davoust M.-E. (2004) Particle filter and Gaussian-mixture filter efficiency evaluation for terrain-aided navigation. XII European Signal Processing Conference EUSIPCO 2004, Sept. 6–10, 2004, Vienna, Austria
86. Fletcher F. K., Kershaw D. J. (2002) Performance analysis of unbiased and classical measurement conversion techniques. IEEE Transactions on Aerospace and Electronic Systems **38**: pp. 1441–1444
87. Fowler K. R., Schmalzel J. L. (2004) Sensors: The first stage in the measurement chain. IEEE Instrumentation and Measurement Magazine, Sept. 2004, pp. 60–65
88. Frank E., Hall M., Pfahringer B. (2003) Locally weighted naive Bayes. In: Proceedings of Uncertanty in Artificial Intelligence UAI '03
89. Franken D., Hupper A. (2005) Improved fast covariance intersection for distributed data fusion. Proceedings Information Fusion 2005

90. Fraser A. M., Swinney H. L. (1986) Independent coordinates for strange attractors from mutual information. Phys Rev **A33**: pp. 1134-1140
91. Friedman N., Geiger D., Goldszmidt M. (1997) Bayesian network classifiers. Machine Learning **29**: pp. 131–163
92. Friedman J. H. (1997) On bias, variance 0/1-loss and the curse of dimensionality. Data Mining and Knowledge Discovery **1**: pp. 55–77
93. Friedman J., Hastie T. and Tibshirani R. (2000) Additive logistic regression: a statistical view of boosting. Annals of Statistics **38**: pp. 337–374.
94. Fukunaga K. (1990) Introduction to Statistical Pattern Recognition. 2nd Edition. Published by: Academic Press, USA
95. Galanis G., Anadranistakis M. (2002) A one-dimensional Kalman filter for the correction of near surface temperature forecasts. Meteorological Applications **9**: pp. 437–441
96. Gamerman D. (1997) Markov Chain Monte Carlo: Stochastic simulation for Bayesian inference. Chapman and Hall, London UK
97. Gao J. B., Harris C. J. (2002) Some remarks on Kalman filters for the multisensor fusion. Information Fusion **3**: pp. 191–201
98. Gelb A. (1974) Applied Optimal estimation. MIT Press, Cambridge, MA, USA
99. Gelman A., Larlin J. S., Stern H. S., Rubin D. R. (2003) Bayesian Data Analysis. Chapman and Hall, Boca Raton, Florida, USA
100. Ghahramani Z, Kim H.-C. (2003) Bayesian classifier combination. Gatsby Technical Report, University College, University of London, UK
101. Gilchrist J. M., Jerwood D., Ismaiel H. S. (2005) Comparying and unifying slope estimates across pyschometric function models. Perception and Psychophysics **67**: pp. 1289-1303
102. Gilks W. R., Richardson S., Spiegelhalter D. J. (1996) Markov Chain Monte Carlo in Practice. Chapman and Hall, London, UK.
103. Ginn C. M. R., Turner D. B., Willett P., Ferguson A. M., Heritage T. W. (1997) Similarity searching in files of three-dimensional chemical structures: evaluation of the EVA descriptor and combination of rankings using data fusion. Journal of Chemical Information and Computer Sciences **37**: 23-37
104. Glasbey C. A., Mardia K. V. (1998) A review of image warping methods. Journal of Applied Statistics **25**: pp. 155–171
105. Goovaerts P. (1999) Geostatistics in soil science: state-of-the-art and perspectives. Geoderma **89**: pp. 1–45
106. Goovaerts P. (2000) Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. Journal of Hydrology **228**: pp. 113–129
107. Guyon I., Elisseeff A. (2003) An introduction to to variable and feature selection. Journal of Machine Learning Research **3**: pp. 1157–1182
108. Hall D. L., Llinas J. (Editors) (2001) Handbook of Multisensor Data Fusion, CRC Press, Boca Raton, Florida 33431, USA
109. Hall D. L., McMullen S. A. H. (2004) Mathematical Techniques in Multisensor Data Fusion, 2nd Edition, Artech House, Inc., Norwood, MA USA
110. Hampel F. Robust Statistics: The Approach Based on Influence Functions. Published by John Wiley and Sons, New York
111. Hanson K. M. (2005) Bayesian analysis of inconsistent measurements of neutron cross sections. In: Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 25th International Workshop on bayesian Inference and Maximum Entropy Methods in Science and Eengineering. AIP Conference Proceedings **803**: pp. 431–439

112. Hardoon D. R., Szedmak S., Shawe-Taylor J. (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Computation **16**: pp. 2639–2664
113. Hastie T., Tibshirani R. (1995) Penalized discriminant analysis. Annals of Statistics **23**: pp. 73–102
114. Hastie T., Tibshirani R. (1998) Classification by pairwise coupling. Annals of Statistics **26**: pp. 451–471
115. Hastie T., Tibshirani R., Friedman J. (2001) The elements of statistical learning. Published by Springer-Vaerlag, Berlin, Germany
116. Henderson T., Shilcrat E. (1984) Logical Sensor Systems. Journal of Robotic Systems **1**: 169–193
117. Henning K. J. (2003) What is syndromic surveillance? In: Syndromic Surveillance: Reports from a National Conference, 2003.
118. Ho T. K. (1998) The random subspace method for constructing decision forests. IEEE Transactions Pattern Analysis and Machine Intelligence **20**: pp. 832–844
119. Hoeting J. A., Madigan D., Raftery A. E., Volinsky C. T. (1999) Bayesian model averaging: a tutorial. Statistical Science **14**: pp.382–417
120. Holder L. B., Rusell I., Markov Z., Pipe A. G., Carse B. (2005) Current and future trends in feature selection and extraction for classification problems. International Journal Pattern Recognition and Artificial Intelligence **19**: pp. 133–142
121. Holliday J. D., Hu C.-Y., Willett P. (2001) Grouping of coefficients for the calculation of inter-molecular similarity and dissimilarity using 2D Fragment bit-strings. Combinatorial Chemistry and Computer Sciences **42**: pp. 375–385
122. Hong S. J., Hosking J., Natarajan R. (2002) Multiplicative adjustment of class probability: educating naive Bayes. Research Report RC 22393 (W0204-041), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598
123. Houzelle S., Giraudon G. (1994) Contribution to multisensor fusion formalization. Robotics and Autonomous Systems **13**: pp. 69–75
124. Howland P., Wang J., Park H. (2006) Solving the small sample size problem in face recognition using generalized discriminant analysis. Pattern Recognition **39**: pp. 277–287
125. Howson C., Urbach P. (1996) Scientific Reasoning: the Bayesian approach (2nd. Edition). Open Court Publishing Co., Peru, Illinois, USA
126. Hseih P-F., Wang D-S., Hsu C-W. (2006) A linear feature extraction for multiclass classification problems based on class mean and covariance discriminant information. IEEE Transactions Pattern Analysis and Machine Intelligence **28**: pp. 223–235
127. Huang H.-J., Hsu C.-N. (2003) Bayesian classification for data from the same unknown class. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics. **32**: pp. 137–145
128. Huang Y. S., Suen C. Y. (1995) A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence **17** pp. 90–94
129. Huber P. J. (1981) Robust Statistics. Published by John Wiley and Sons, New York
130. Hubert M., Van Driessen K. (2004) Fast and robust discriminant analysis. Computational Statistics and Data Analysis **45**: pp. 301–320

131. Hyder A. K., Shahbazian E., Waltz E. (Eds) (2002) Multisensor Fusion. Kluwer Academic Publishers, Dordrecht, Netherlands
132. Hyvarinen A., Karhunen J., Oja E. (2001) Independent Component Analysis. John Wiley and Sons, New York, USA
133. Inza I., Larranaga P., Blanco R., Cerrolaza A. J. (2004) Filter versus wrapper gene selection approaches in DNA microarray domains. Artificial Intelligence in Medicine **31** pp. 91–103
134. Ivanov O., Wagner M. WM., Chapman W. W., Olszewski R. T. (2002) Accuracy of three classifiers of acute gastrointestinal syndrome for syndromic surveillance. Proceedings AMIA Symposium, Philadelpha, Pennsylvania, pp. 345–349
135. Jacobs R. A. (1995) Methods for combining experts' probability assessments. Neural Computation **7**: pp. 867–888
136. Jain A. K., Chandrasekaran B. (1982) Dimensionality and sample size consideration in pattern recognition practice. In: Handbook of Statistics **2** (Eds. Krishnaiah P. R., Kanal L. N.) pp. 835–855
137. Jain A. K., Duin R. P. W., Mao J. (2000) Statistical Pattern Recognition: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**: pp. 4–37
138. Jain A., Nandakumar K., Ross A. (2005) Score normalization in multimodal biometric systems. Pattern Recognition **38**: pp. 2270–2285
139. Jain A., Ross A. (2002) Fingerprint mosaicking. Proccedings IEEE International Conference on Acquisition, Speech and Signal Processing, ICASSP, Orlando, Florida, May 13–17
140. Jain A., Zongker D. (1997) Feature Selection: Evaluation, application and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**: pp. 153–158
141. Jolliffe I. T. (1986) Principal Component Analysis. Springer-Verlag, New York, USA
142. Jones G. D., Allsop R. E., Gilby J. H. (2003) Bayesian analysis for fusion of data from disparate imaging systems for surveillance. Image and Vision Computing **21**: pp. 843–849
143. Jones M. C., Marron J. S., Sheather S. J. (1996) A brief survey of bandwidth selection for density estimation. Journal of the Americal Statistical Association **91**: pp. 401–407
144. Julier S., Uhlmann J. (2001) General decentralized data fusion with covariance intersection (CI). In: Handbook of Multidimensional Data Fusion (Eds. Hall D., Llians J.), Chapt. 12, CRC Press
145. Julier S. J., Uhlmann J. K. (2004) Unscented filtering and nonlinear estimation. Proceedings IEEE **92**: pp. 401–423
146. Julier S. J., Uhlmann J. K., Nicholson D. N. (2004) A method for dealing with assignment ambiguity. Proceedings of the 2004 American Control Conference,
147. Kass R. E., Wasserman L. (1994) Formal rules for selecting prior distributions: A review and annotated bibliography. Technical Report 583, Department of Statistics, Carnegie Mellon University
148. Kay S. M. (1993) Fundamentals of Statistical Signal Processing: Estimation Theory. Published by Prentice-Hall, New Jersey, USA
149. Keogh E. J., Pazzani M. J. (2001) Derivative dynamic time warping. First SIAM International Conference on Data Mining, Chicago, Illinois, USA

150. Keogh E. J., Pazzani M. J. (2002) Learning the structure of augmented Bayesian classifiers. International Journal of Artificial Intelligence Tools **11**: pp. 587–601
151. Keys R. G. (1981) Cubic convolution interpolation for digital image processing. IEEE Transactions Accoustics, Speech and Signal Processing **29**: pp. 1153–1160
152. Kirianaki N. V., Yurish S. Y., Shpak N. O., Deynega V. P. (2002) Data Aquistion and Signal Processing for Smart Sensors. Published by John Wiley and Sons, Ltd, Chichester, UK
153. Kittler J., Hatef M., Duin R. P. W., Matas J. (1998) On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**: pp. 226–239
154. Klein L. A. (1993) Sensor and Data Fusion: Concepts and Applications. Published by SPIE Optical Engineering Press
155. Klein G. J. (2002) Four-dimensional processing of deformable Cardiac PET data. Medical Image Analysis **6**: pp. 29–46
156. Kohavi R., John G. (1997) Wrappers for feature subset selection. Artificial Intelligence **97**: pp. 273–324
157. Koldovsky Z., Tichavsky P., Oja E. (2006) Efficient variation of algorithm FastICA for independent component analysis attaining the Cramer-Rao lower bound. IEEE Transactions on Neural Networks **17**: pp. 1265–1277
158. Konolige K. (1997) Improved occupancy grids for map building. Autonomous Robots **4**: pp. 351–367
159. Kopetz H., Bauer G. (2003) The time-triggered architecture. Proceedings IEEE **91**: pp. 112–126
160. Kopetz H., Holzmann M., Elmenreich W. (2000) A universal smart transducer interface: TTP/A. Third IEEE International Symposium on object-oriented realtime distributed computing ISORC 2000
161. Kraskov A., Stogbauer H., Grassberger P. (2004) Estimating mutual information. Physical Review **69E** Article No. 066138 Part 2, June 2004
162. Kreucher C., Kastella K., Hero III A. O. (2005) Sensor management using an active sensing approach. Signal Processing **85**: pp.607–624
163. Kuncheva L. I. (2004) Combining Pattern Classifiers. Published by John Wiley and Sons Ltd, Hoboken, New Jersey
164. Kwak N., Choi C.-H. (2002) Input feature selection using mutual information based on Parzen window. IEEE Pattern Analysis and Machine Intelligence **24**: pp. 1667–1671
165. Learned-Miller E. G., Fisher T. W. (2003) ICA using spacings estimates of entropy. Journal of Machine Learning Research **4**: pp. 1271–1295
166. Ledesma-Carbayo M. J., Kybic J., Desco M., Santos A., Suhling M., Hunziker P., Unser M. (2005) Spatio-temporal nonrigid registration for ultrasound cardiac motion estimation. IEEE Transactions on Medical Imaging **24**: pp. 1113–1126
167. Lee T. W. (1998) Independent Component Analysis: Theory and Applications. Kluwer Academic Publishers, Dordrecht
168. Lee P. M. (1997) Bayesian Statistics: An introduction. Oxford University Press
169. Lee C., Choi E. (2000) Bayes error evaluation of the Gaussian ML classifier. IEEE Transactions Geoscience and Remote Sensing **38**: pp. 1471–1475

170. Lehmann T. M., Gonner C., Spitzer K. (1999) Survey: Interpolation methods in medical image processing. IEEE Transactions in Medical Imaging **18**: pp. 1049–1075
171. Lewis D. D. (1998) Naive (Bayes) at forty: the independence assumption in information retrieval. Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, April 21–23, pp. 4–15
172. Li X. R., Jilkov V. P. (2003) A Survey of maneuvering target tracking. Part I: Dynamic models. IEEE Transactions Aerospace and Electronic Systems **39**: pp.1333–1363
173. Li. M., Yuan B. (2005) 2D-LDA: A novel statistical linear discriminat analysis for image matrix. Pattern Recognition Letters **26**: pp. 527–532
174. Li H., Zhang K., Jiang T. (2005a) Robust and accurate cancer classification with gene expression profiling. Proceeding of the 4th IEEE Computational Systems Bioinformatics Conference, Stanford, CA, USA, pp. 310–321
175. Li H., Jiang T., Zhang K. (2006) Efficient and Robust Feature Extraction by Maximum Margin Criterion. IEEE Transactions Neural Networks **17**: pp. 157–165
176. Li S. Z., Zhang Z.-Q. (2004) Floatboost learning and statistical face detection. IEEE Transactions on Pattern Analysis and Machine Learning **26**: pp. 1112–1123
177. Li X. R., Zhi X. (1996) PSNR: a refined strongest neighbor filter for tracking in clutter. Proceedings of 35th IEEE Conference on Decision and Control, Kobe, Japan
178. Liang Y., Gong W., Pan Y., Li W. (2005) Generalized relevance weighted LDA. Pattern Recognition **38**: pp. 2217–2219
179. Lin H.-T., Lin C.-J., Weng R. C. (2003) A note on Platt's probabilistic outputs for support vector machine. Tech. Report, Department of Computer Science and Information Engineering, National Taiwan University
180. Lindhal D., Palmer J., Pettersson J., White T., Lundin A., Edenbrandt L. (1998) Scintigraphic diagnosis of coronary artery disease: myocardial bull's-eye images contain the important information. Clinical Physiology **18**: pp. 554–561
181. Ling C. X., Zhang H. (2002) Toward Bayesian classifiers with accurate probabilities. Proceedings 6th Pacific Asia Conference on Advances in Knowledge Discovery and Data Mining, Taipei, Taiwan, pp. 123-134
182. Loog M., Duin R. P. W.(2004) Linear dimensionality reduction via a heteroscedastic extension of LDA: The Chernoff criterion. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**: pp. 732–739
183. Loughrey J., Cunningham P. (2004) Overfitting in wrapper-based feature subset selection: the harder you try the worse it gets. 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence pp. 33–43. See also Technical Report TCD-CS-2005-17, Department of Computer Science, Trinity College, Dublin, Ireland
184. Loughrey J., Cunningham P. (2005) Using early-stopping to avoid overfitting in wrapper-based feature selection employing stochastic search. Technical Report TCD-CS-2005-37, Department of Computer Science, Trinity College, Dublin, Ireland
185. Lubin J., Bloom J. A., Cheng H. (2003) Robust, content-dependent, high-fidelity watermark for tracking in digital cinema (2003). Proceedings SPIE **5020** pp. 536–545

186. Lumini A, Nanni L. (2006) Detector of image orientation based on Borda count. Pattern Recognition Letters **27**: pp. 180–186
187. Luo R. C., Yih C-C, Su K. L. (2002) Multisensor Fusion and Integration: Approaches, Applications and Future Research Directions. IEEE Sensors Journal **2**: pp. 107–119
188. Maes F., Vandermeulen D., Suetens P. (2003) Medical image registration using mutual information. Proceedings IEEE **91**: pp. 1699–1722
189. MacKay D. J. C. (1991) Bayesian methods for adaptive models. PhD thesis, California Institute of Technology, Pasadena, California, USA
190. MacKay D. J. C. (2003) Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge, UK
191. Madigan D. M., Raftery A. E. (1994) Model selection and accounting for model uncertainty in graphical models using Occam's window. Journal of American Statistical Association **89**: pp. 1335–1346
192. Manfredi V., Mahadevan S., Kurose J. (2005) Switching Kalman filters for prediction and tracking in an adaptive meteorological sensing network. Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, SECON 2005, Santa Clara, California, USA
193. Mardia K. V., Goodall C., Redfern E. J., Alonso F. J. (1998) The Kriged Kalman filter. Test **7**: pp. 217-284
194. Martin J. K., Hirschberg D. S. (1996) Small sample statistics for classification error rates I: error rate measurements. Technical Report No. 96-21, July 2, 1996, Department of Information and Computer Science, University of California, Irvine, CA 9297-3425, USA
195. Martin J. K., Hirschberg D. S. (1996) Small sample statistics for classification error rates II: confidence intervals and significance tests. Technical Report No. 96-22, July 7, 1996, Department of Information and Computer Science, University of California, Irvine, CA 9297-3425, USA
196. Maskell S. (2004) Sequentially structured Bayesian solutions. PhD thesis, Cambridge University
197. Mazziotta J., Toga A., Fox P., Lancaster J., Zilles K., Woods R., paus T., Simpson G., Pike B., Holmes C., Collins L., Thompson P., MacDonald D., Iacoboni M., Schormann T., Amunts K., Palomero-Gallagher N., Geyer S., Parsons L., Narr K., Kabani N., Le Goualher G., Feidler J., Smith K., Boomsma D., Pol H. H., Cannon T., Kawashima R., Mazoyer B. (2001) A four-dimensional probabilistic atlas of the human brain. Journal of the American Medical Informatics Association **8**: pp. 401–430.
198. McIntyre G. A., Hintz K. J. (1999) A comprehensive approach to sensor management, Part I: A survey of modern sensor management systems. Available on the internet.
199. McLachlan G. J., Krishnan T. (1997) The EM algorithm and Extensions. John Wiley and Sons, Ltd, USA
200. McLachlan G. J., Peel D. (1998) Robust cluster analysis via mixtures of multivariate t-distributions. Lecture Notes in Computer Science **1451**: pp. 658–666
201. McLaughlin S., Krishnamurthy V., Challa S. (2003) Managing data incest in a distributed sensor network. Proceedings IEEE International Conference on Accoustics, speech and Signal Processing **5**: pp. V-269–V-272, 6-10 April, 2003.
202. Meinhold R. J., Singpurwalla N. D. (1983) Understanding the Kalman filter. The American Statistician **37**: pp. 123–127

203. Meer P. (2004) Robust techniques for computer vision. In: Emerging Topics in Computer Vision, (eds) Medioni G., Bing Kang S. Published by Prentice-Hall
204. Memarsadeghi N., Le Moigne J., Mount D. M., Morisette J. (2005) A new approach to image fusion based on cokriging. The 8th International Conference on Information Fusion, PA, USA, July 25-29
205. Merwe R. van der (2004) Sigma-point Kalman filters for probabilistic inference in dynamic state-space models. PhD thesis, Oregon Health and Science University
206. Migon H. S., Gamerman D. (1999) Statistical Inference- An Integrated Approach. Arnold Publishers Ltd, London, UK
207. Milgram J., Cheriet M., Sabourin R. (2005) Estimating accurate multi-class probabilities with support vector machines. International Joint Conference on Neural Networks IJCNN 2005, Montreal, Canada, pp. 1906–1911
208. Miller M. D., Drummond O. E. (1999) Coordinate transformation bias in target tracking. Proceedings SPIE Conference on Signal and Data Processing of Small Targets **3809**: pp. 409–424.
209. Minka T. P. (2000) Automatic choice of dimensionality for PCA. NIPS 2000
210. Minka T. P. (2001) A family of algorithms for approximate Bayesian inference. Phd thesis, Masschusetts Institute of Technology, Massachusetts, USA
211. Minka T. P. (2003) The "summation hack" as an outlier model. Unpublished article. Available from the author's homepage.
212. Minka T. P. (2005) Inferring a Gaussian distribution. Unpublished article. Available from the author's homepage.
213. Miranda S., Baker C., Woodbridge K., Griffiths H. (2006) Knowledge-based resource management for multi-function radar. IEEE Signal Processing Magazine **67** January 2006, pp. 66-77
214. Mitchell D. P., Netravali A. N. (1988) Reconstruction filters in computer graphics. Comp. Graph. **22**: pp. 221–228
215. Moore J. R., Blair W. D. (2000) Practical aspects of multisensor tracking. In: Bar-Shalom Y., Blair W. D. (Ed) Multitarget-Multisensor tracking: Applications and Advances, Vol. III, pp. 1-76. Published by Artech House, Norwood, MA.
216. Moore M., Wang J. (2003) An extended dynamic model for kinematic positioning. Journal of Navigation **56**: pp. 79–88
217. Moro C. M. C., Moura L., Robilotta C. C. (1994) Improving reliability of Bull's Eye method. Computers in Cardiology pp. 485–487
218. Movellan J. R., Mineiro P. (1998) Robust sensor fusion: analysis and applications to audio-visual speech recognition. Machine Learning **32**: pp. 85–100
219. Mullen T., Avasarala V., Hall D. L. (2006) Customer-driven sensor management. IEEE Intelligent Systems March/April 2006, pp. 41–49
220. Munich M E. (2000) Visual input for pen-based computers. PhD thesis, California Institute of Technology, Pasadena, California, USA
221. Munich M. E., Perona P. (1999) Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. Proceedings of the 7th International Conference on Computer Vision (ICCV '99), Korfu, Greece
222. Murphy K. P. (1998) Switching Kalman Filters. Unpublished report. Available from authors homepage.
223. Murphy K. P. (2001) The Bayes net toolbox for matlab. (2001) Computing Science and Statistics **33**: pp. 331–350

224. Murphy K. P. (2002) Dynamic Bayesian networks: representation, inference and learning (2002) PhD thesis, University of California, Berkeley, USA
225. Nabney I. T. (2002) Netlab: Algorithms for pattern recognition. Springer-Verlag, London
226. Nandakumar K. (2005) Integration of multiple cues in biometric systems. MSc thesis, Department of Computer Science and Engineering, Michigan State University
227. Nelson T., Wilson H. G., Boots B., Wulder M. A. (2005) Use of ordinal conversion for radiometric normalization and change detection. International Journal of Remote sensing **26**: pp. 535–541
228. Ng G. W., Ng K. H. (2000) Sensor management - what, why and how. Information Fusion **1**: pp. 67–75
229. Nunn W. R. (1979) Position finding with prior knowledge of covariance parameters. IEEE Transactions on Aerospace and Electronic Systems **15**: pp. 204–208
230. O'Hagan A., Forster J. J. (2004) Bayesian Inference-Kendall's Advanced Theory of Statistics Vol. **2B**. Published by Arnold, London
231. O'Sullivan S.,Collins J. J., Mansfield M., Eaton M., Haskett D. (2004) A quantitative evaluation of sonar models and mathematical update methods for map building with mobile robots. Ninth International Symposium on Artificial Life and Robotics (AROB), Oita, Japan
232. Oh W., Lindquist B. (1999) Image thresholding by indicator Kriging. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**: pp. 590–602
233. Ober P. B. (2000) Integrity according to Bayes. IEEE Position Location and Navigation Symposium, pp. 325-332, San Diego, California, March 13–16
234. Opitz F., Henrich W., Kausch T. (2004) Data fusion development concepts within complex surveillance systems. Proceedings 7th International Conference Information Fusion June 28-July 1, 2004, Stockholm, pp. 308–315
235. Osadciw L., Veeramachaneni K. (2005) A controllable sensor management algorithm capable of learning. Proceedings of SPIE Defense and Security Symposium.
236. Oxley M. E., Thorsen S. N. (2004) Fusion or integration: what's the difference? Proceedings 7th International Conference Information Fusion June 28–July 1, 2004, Stockholm, pp. 429–434
237. Pardo-Iguzquiza E., Chica-Olmo M., Atkinson P. M. (2006) Downscaling cokriging for image sharpening. Remote Sensing of Environment **102**: pp. 86–98
238. Paulitsch M. (2002) Fault-tolerant clock synchronization for embedded distributed multi-cluster systems. PhD thesis, Institut fur Technische Informatik, Technischen Universitat Wien
239. Pazzani M. J. (1996) Searching for dependencies in Bayesian classifiers. In: Learning from Data: AI and Statistics V. (Eds. Fisher D., Lenz H-J.), Published by Springer-Verlag
240. Peng H., Long F., Ding C. (2005) Feature selection based on mutual information: criteria of max-dependency, max-relevance and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**: pp. 1226–1238
241. Perperidis D. (2005) Spatio-temporal registration and modelling of the heart using cardiovascular MR imaging. PhD thesis, University of London
242. Perperidis D., Mohiaddin R., Rueckert D. (2005) Spatio-temporal free-form registration of cardiac MR image sequences. Medical Image Analysis **9**: pp. 441–456.

243. Persson N., Gustafsson F., Drevo M. (2002) Indirect tire pressure monitoring using sensor fusion. Proceedings SAE 2002, Detroit, Report 2002-01-1250, Department of Electrical Engineering, Linkoping University, Sweden
244. Platt J. (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in large Margin Classifiers (Eds. Smola A. J., Bartlett P., Scholkopf B., Schurmans D). MIT Press pp. 61–74
245. Pluim J. P. W., Maintz J. B. A., Viergever M. A. (2000) Interpolation artifacts in mutual information-based image registration. Computer Vision and Image Understanding **77**: pp. 211-232
246. Pluim J. P. W., Maintz J. B. A., Viergever M. A. (2003) Mutual information based registration of medical images: a survey. IEEE Transactions on Medical Imaging **22**: pp. 986–1004
247. Poledna S. (1997) Replica determinism and flexible scheduling in hard real-time dependable systems. Research Report Nr. 21/97 Nov 1997, Institut fur Technische Informatik, Technishe Universitat Wien, Austria
248. Polikar R. (2006) Ensemble based systems in decision making. IEEE Circuits and Systems Magazine **6**, Number 3, pp. 21–45
249. Press W. H. (1996) Understanding data better with Bayesian and global statistical methods. Unsolved problems in astrophysics. Proceedings of conference in honor of John Bahcall (Ed. Striker J. P.). Princeton University Press
250. Price D., Knerr S., Personnaz L., Dreyfus G. (1995) Pairwise neural network classifiers with probabilistic outputs. In: Neural Information Processing Systems (Eds. Tesauro G., Touretzky D., Leen T.) **7**: pp. 1109-1116, The MIT Press, USA
251. Prosper H. B., Linnemann J. T., Rolke W. A. (2002) A glossary of selected statistical terms. Proceedings of Advanced Statistical Techniques in Particle Physics, Grey College, Durham, UK, 18–22 March, 2002
252. Pulford G. W. (2005) Taxonomy of multiple target tracking methods. IEE Proceedings-Radar, Sonar Navig **152**: pp. 291–304
253. Punska O. (1999) Bayesian approaches to multi-sensor data fusion. MPhil thesis, Signal Processing and Communications Laboratory, Department of Engineering, University of Cambridge
254. Qin A. K., Suganthan P. N., Loog M. (2005) Uncorrelated heteoscedastic LDA based on the weighted pairwise Chernoff criterion. Pattern Recognition **38**: pp. 613–616
255. Rabiner L., Juang B. (1993) Fundamentals of Speech Processing. Prentice-Hall, Inc., USA
256. Ranawana R. (2006) Multi-classifier systems - review and a roadmap for developers. International Journal of Hybrid Intelligent Systems **3**: pp.35–61
257. Rao R. P. N. (1996) Robust Kalman filters for prediction, recognition and learning. Rechnical Report 645, Computer Science Department, University of Rochester, December 1996
258. Rasmussen C. E., Williams C. K. I. (2006) Gaussian processes for machine learning. MIT Press, cambridge, USA
259. Ratanamahatano C. A., Keogh E. (2005) Three myths about dynamic time warping. SIAM 2005 Data Mining Conference, CA, USA.
260. Rath T. M., Manmatha R. (2003) Features for word spotting in historical manuscripts. Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), **1**: pp. 218–222, Edinburgh, Scotland

261. Raudys S. J., Jain A. K. (1991) Small sample size effects in statistical pattern recognition: recommendations for practitioners. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**: pp. 252–264
262. Reichenbach S. E., Geng F. (2003) Two-dimensional cubic convolution. IEEE Transactions on Image Processing **12**: pp. 857–865
263. Ridgeway G., Madigan D., Richardson T., O'Kane J. W. (1998) Interpretable boosted naive Bayes classification. Proceedings 4th International Conference on Knowledge Discovery and Data Mining, New York, NY, pp. 101–104
264. Rish I. (1999) Efficient reasoning in graphical models. Phd thesis, University of California, Irvine, USA
265. Rish I. (2001) An empirical study of the naive Bayes classifier. Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington
266. Rish I., Hellerstein J., Jayram T. S. (2001) An analysis of data characteristics that affect naive Bayes performance. IBM Technical Report RC2 1993, IBM Research division, Thomas J. Watson Research Center, NY 10598
267. Rodriguez J. J., Kuncheva L. I., Alonso C. J. (2006) Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence. **28**: pp. 1619–1630
268. Rosti A-V. I., Gales M. J. F. (2001) Generalized linear Gaussian models. Technical Report CUED/F-INFENG/TR-420, November 23, 2001, Cambridge Universityu Engineering Dpartment, Cambridge, England
269. Rousseeuw P. J., Leroy A. M. (1987) Robust Regression and Outlier Detection. Published by John Wiley and Sons and Sons, New York
270. Roweis S., Ghahramani Z. (1999) A unifying review of linear Gaussian models. Neural Computation **11**: pp. 305–345
271. Ruanaidh J. J. K. O, Fitzgerald W. J. (1996) Numerical Bayesian methods applied to signal processing. Springer-Verlag, New York, USA
272. Sanders-Reed J. N. (2001) Error propagation in two sensor three-dimensional position estimation. Optical Engineering **40**: pp. 627–636
273. Schabenberger O., Gotway C. A. (2005) Statistical Methods for Spatial Data Analysis. Chapman and Hall/CRC, Boca Raton, Floria, USA
274. Schapire R. E. (2002) The boosting approach to machine learning: An overview. Proceedings from MSRI Workshop on Nonlinear Estimation and Classification.
275. Schlogl A., Fortin J., Habenbacher W., Akay M. (2001) Adaptive mean and trend removal of heart rate variability using Kalman filtering. Proceedings 23rd. Annual International Conference of the IEEE Engineering in Medicine and Biology Society 25–28 October 2001
276. Scott D. W. (1992) Multivariate Density Estimation. Wiley
277. Shin V., Lee Y., Choi T-S (2006) Generalized Millman's formula and its application for estimation problems. Signal Processing **86**: pp. 257–266
278. Silverman B. (1986) Density estimation for statistical data analysis. Chapman-Hall
279. Singh M., Thompson R., Basu A., Rieger J., Mandal M. (2006) Image based temporal registration of MRI data for medical visualization. IEEE International Conference on Image Processing, Atlanta, Georgia, Oct. 8–11
280. Sivia D. S. (1996) Data Analysis: A Bayesian Tutoral. Oxford University Press, Oxford, UK

281. Sivia D. S. (1996) Dealing with duff data. Proccedings of the Maximum Entropy Conference. (Editors) Sears M., Nedeljkovic N. E., Sibisi S. NMB Printers, Port Elizabeth, South Africa, pp. 131–137
282. Skurichina M., Duin R. P. W. (2005) Combining feature subsets in feature selection. Proceedings Sixth International Workshop Multiple Classifier Systems (MCS '05), June 13-15, Seattle, California, pp. 165-175
283. Skocaj D. (2003) Robust subspace approaches to visual learning and recognition. PhD thesis, University of Ljubljana
284. Steuer R., Kurths J., Daub C. O., Weise J., Selbig J. (2002) The mutual information: detecting and evaluating dependencies between variables. Bioinformatics **18** Supplement 2, pp. S231–S240
285. Stewart C. V. (1999) Robust parameter estimation in computer vision. SIAM Review **41**: pp. 513–537
286. Tang E. K., Suganthan P. N., Yao X., Qin A. K. (2005) Linear dimensionality reduction using relevance weighted LDA. Pattern Recognition **38**: pp. 485–493
287. Tang E. K., Suganthan P. N., Yao X.(2006) An analysis of diversity measures. Machine learning **65**: pp. 247–271
288. Tax D. M. J. (2001) One-class classification. PhD thesis, Delft University, The Netherlands
289. Tax D. M. J., van Breukelan M., Duin R. P. W., Kittler J. (2000) Combining multiple classifiers by averaging or by multiplying? Pattern Recognition **33**: pp. 1475–1485
290. Thevenaz P., Blu T., Unser M. (2000) Interpolation Revisited. IEEE Transactions on Medical Imaging **19**: pp. 739–758
291. Thevenaz P., Unser M. (1996) A pyramid approach to sub-pixel image fusion based on mutual information. Proceedings IEEE International Conference on Image Processing, Lausanne, Switzerland, Sept. 16-19, Vol. 1 pp. 265–268
292. Thomas C. S. (1999) Classifying acute abdominal pain by assuming independence: a study using two models constructed directly from data. Technical Report CSM-153, Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland
293. Thomaz C. E., Gillies D. F. (2001) "Small sample size": A methodological problem in Bayes plug-in classifier for image rcognition. Technical Report 6/2001, Department of Computing, Imperial College of Science, Technology and Medicine, London
294. Thomaz C. E., Gillies D. F. (2005) A maximum uncertainty LDA-based approach for limited sample size problems - with application to face recognition. XVIII Brazalian Symposium on Computer Graphics and Image Processing SIGGRAPI '05 pp. 89–96
295. Thomaz C. E., Gillies D. F., Feitosa R. Q. (2004) A new covariance estimate for Bayesian classifiers in biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology **14**: pp. 214–223
296. Thompson P. M., Mega M. S., Narr K. L., Sowell E. R., Blanton R. E. and Toga A. W. (2000) Brain image analysis and atlas construction. In: Handbook of Medical Imaging Vol. 2 Medical Image Processing and Analysis. SPIE Press, Bellingham, Washington 98227-0010, USA
297. Thrun S. (1998) Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence **99**: pp. 21–71
298. Thrun S. (2003) Learning occupancy grids with forward sensor models. Autonomous Robots **15**: pp. 111–127

299. Ting K. M., Witten I. H. (1997) Stacked generalization: when does it work? Proceedings of the 15th International Joint Conference on Artificial Intelligence
300. Tipping M. E., Bishop C. M. (1999) Probabilistic principal component analysis. Journal Royal Statistical Society, **61B**: pp. 611-622
301. Tipping M. E., Bishop C. M. (1999) Mixtures of probabilistic principal component analysis. Neural Computation **11** pp. 443-482
302. Todini E. (2001) A Bayesian technique for conditioning radar precipitation estimates to rain-gauge measurements. Hydrology and Earth System Sciences **5**: pp. 187–199
303. Trodhandl C. (2002) Architectural requirements for TP/A nodes. MSc. thesis, Institut Technische Informatik, Technischen Universitat Wien
304. Trunk G. V. (1979) A problem of dimensionality: a simple example. IEEE Transactions on Pattern Analysis and Machine Intelligence **1**: pp. 306–307
305. Tumer K., Ghosh J. (2002) Robust combining of disparate classifiers through order statistics. Pattern Analysis and Applications **5** pp. 189–200
306. Tumer K., Oza N. C. (2003) Input decimated ensembles. Pattern Analysis and Applications **6**: pp. 65–77
307. Uhlmann J. K. (2003) Covariance consistency methods for fault-tolerant distributed data fusion. Information Fusion **4**: pp. 201–215
308. Valera M., Velastin S. A. (2005) Intelligent distributed surveillance systems: a review. IEE Proceedings.-Vision Image Signal Process **152**, pp. 192–204
309. Varshney P. K. (1996) Distributed detection and data fusion. Springer-Verlag, New York, USA
310. Veeramachaneni K., Osadciw L. A., Varshney P. K. (2005) An adaptive multimodal biometric managament algorithm. IEEE Transactions on Systems, Man and Cybernetics **35**, Part C, pp. 344–356
311. Verboven S., Hubert M. (2005) LIBRA: a MATLAB library for robust analysis. Chemometrics and Intelligent Laboratory Systems **75**: pp. 127–136
312. Vezhnevets A., Vezhnevets V. (2005) Modest AdaBoost-teaching AdaBoost to generalize better. Fifteenth International Conference on Computer Graphics and Applications, June 20–24, Novosibirsk, Russia
313. Viaene S., Derrig R., Dedene G. (2004) A case study of applying boosting naive Bayes to claim fraud diagnosis. IEEE Transactions Knowledge and Data Engineering **16**: pp. 612–619
314. Vilalta R., Rish I. (2003) A decomposition of classes via clustering to explain and improve naive Bayes. Proceedings 14th European Conference on Machine Learning ECML, Dubrovnik, Croatia
315. Viola P., Wells III, W. M., (1997). Alignment by maximization of mutual information. International Journal of Computer Vision **24**: pp. 137–154
316. Wald L. (1999) Some terms of reference in data fusion. IEEE Transactions on Geoscience and Remote Sensing **37**: pp. 1190–1193
317. Wald L. (2002) Data Fusion: Definitions and Architectures. Published by Les Presses de l'Ecole des Mines, 60 Boulevard Saint Michel, 75006 Paris, France
318. Waltz E., Llinas J. (1990) Multisensor Data Fusion, Artech House, Inc.
319. Wang H. (2004) Robust statistics for computer vision: model fitting, image segmentation and visual motion analysis. PhD thesis, Monash University, Clayton, Victoria 3800, Australia

320. Wang M., Perera A., Gutierrez-Osuna R. (2004) Principal discriminants analysis for small-sample-size problems: application to chemical sensing. Proceedings of the 3rd IEEE Conference on Sensors, Vienna, Austria, Oct. 24–27
321. Wang H., Yao K., Pottie G., Estrin D. (2004) Entropy-based sensor selection heuristic for target localization. Proceedings IPSN '04
322. Webb A. (1999) Statistical Pattern Recognition. Published by Arnold, London, England
323. Webb G. I. (2000) Multiboosting: A technique for combining boosting and wagging. Machine Learning **40**: pp. 159–197
324. Webb G. I., Boughton J., Wang Z. (2005) Not so naive Bayes: aggregating one-dependence estimators. Machine Learning **58**: pp. 5–24
325. Welch G., Bishop G. (2001) An introduction to the Kalman filter. Notes to accompany Siggraph 2001, Course 8. Available from `http://www.cs.unc.edu/~welch`
326. Wells W, Viola P, Atsumi H., Nakajima S., Kikinis R. (1996) Multi-modal volume registration by maximization of mutual information. Medical Image Analysis **1**: pp. 35–51 (1996)
327. Werman M., Keren D. (2001) A Bayesian method for fitting parametric and nonparametric models to noisy data. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** pp. 528–534
328. Wilde M. M. (2004) Controlling performance in voice conversion with probabilistic principal component analysis. MSc thesis, Tulane University, USA
329. Willett P. (1998) Chemical similarity searching. Journal Chemical Information Computer Science **38**: pp. 983–996
330. Willett P. (2000) Textual and chemical information processing: different domains but similar algorithms. Information Research **5** No. 2.
331. Willett P. (2003) Structural biology in drug metabolism and drug discovery. Biochemical Society Transactions **31** Part 3 pp. 603–606
332. Wilton D., Willet P., Lawson K., Mullier G. (2003) Comparison of ranking methods for virtual screening in lead-discovery programmes. Journal Chemical Information and Computer Sciences **43**: pp. 469–474
333. Williams C. K. I., Barber D. (1998) Bayesian classification with Gaussian processes. IEEE Transactions Pattern Analysis and Machine Intelligence **20**: pp. 1342–1351
334. Wu T.-F., Lin C.-J., Weng R. C. (2004) Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research **5**: pp. 975–1005
335. Wu H., Siegel M., Khosla P. (1999) Vehicle sound signature recognition by frequency vector principal component analysis. IEEE transactions on Instrumentation and Measurement **48**: pp. 1005–1009
336. Xie Z., Hsu W., Liu Z., Lee M. L. (2002) SNNB: A selective neighbourhood-based naive Bayes for lazy classification. Lecture Notes in Computer Science **2336**: pp. 104–114, Springer-Verlag
337. Xiong N., Svensson P. (2002) Multi-sensor management for information fusion: issues and approaches. Information Fusion **3**: pp. 163–186
338. Xu L, Krzyzak A., Suen C. Y. (1992) Several methods for combining multiple classifiers and their applications in handwritten character recognition. IEEE Transactions on Systems, Man and Cybernetics **22**: pp. 418–435
339. Yang Y. (2003) Discretization for Naive-Bayes learning. PhD thesis, School of Computer Science and Software Engineering, Monash University, July 2003

340. Yang R., Berger J. O. (1997) A catalog of noninformative priors. Planning Inference **79**: pp. 223-235. Discussion Paper 97-42, ISDS, Duke University, Durham, NC, USA
341. Yang Y., Webb G. I. (2003) Weighted proportional k-interval discretization for naive Bayes classifiers. PAKDD 2003 pp. 501–512
342. Yang J., Zhang D., Frangi A. F., Yang J. Y. (2004) Two-dimensional PCA: A new approach to appearance-based face representation and recognition. IEEE Transactions Pattern Analysis and Machine Intelligence **26**: pp. 131–137
343. Yang J., Jin Z., Yang J.-Y., Zhang D., Frangi A. F. (2004) Essence of kernel Fisher discriminant: KPCA plus LDA. Pattern Recognition **37**: pp.2097–2100
344. Ye J., Janardan R., Li Q., Park H. (2006) Feature reduction via generalized uncorrelated linear discriminant analysis. IEEE Transactions on Knowledge and Data Engineering. **18**: pp. 1312–1322
345. Yilmazer N., Osadciw L. A. (2004) Sensor management and Bayesian networks. Proceedings SPIE **5434**
346. Zadrozny B. (2001) Reducing multiclass to binary by coupling probability estimates. Neural Information Processing Systems Conference, Vancouver, British Columbia
347. Zadrozny B. (2003) Policy mining: learning decision policies from fixed sets of data. PhD thesis, University of California, San Diego
348. Zadrozny B., Elkan C. (2002) Transforming classifier scores into accurate multiclass probability estimates. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data, Edmonton, Alberta, Canada
349. Zhang D., Zhou Z-H., Chen S. (2006) Diagonal principal component analysis for face recognition. Pattern Recognition **39** pp. 140-142
350. Zhang H. (2005) Exploring conditions for the optimality of naive Bayes. International Journal of Pattern Recognition and Artificial intelligence **19**: pp. 183–198
351. Zhang P., Peng J., Riedel N. (2005) Discriminant analysis: a unified approach. Proccedings Fifth IEEE International Conference on Data Mining (ICDM '05), 27-30 November, Houston, Texas, USA
352. Zhang Z. (1997) Parameter estimation techniques: a tutorial with application to conic fitting. Image and Vision Computing **15**: 59–76
353. Zheng Z. (1998) Naive Bayesian classifier committee. 10th European Conference on Machine Learning, Chemnitz, Germany, April 21–23, pp. 196–207
354. Zheng Z., Webb G. I. (2000) Lazy learning of Bayesian rules. Machine Learning **41**: pp. 53–87.
355. Zheng Z., Webb G. I., Ting K. M. (1999) Lazy Bayesian rules: a lazy semi-naive Bayesian learning technique competitive to boosting decision trees. Proceedings of Sixteenth International Conference on Machine Learning, Bled, Slovenia,
356. Zheng W., Zhao L., Zou C. (2004) An efficient algorithm to solve the small sample size problem for LDA. Pattern Recognition **37**: pp. 1077–1079
357. Zimmerman N. M. (1998) A primer on electrical units in the Systeme International. American Journal of Physics **66**: pp. 324–331
358. Zink M, Westbrook D., Abdallah S., Horling B., Lakamraju V., Lyons E., Manfrdi V., Kurose J., Hondl K. (2005) Meteorological command and control. Proceedings EESR '05: Workshop on end-to-end, sense-and-respond systems, applications and services

359. Zitova B., Flusser J. (2003) Image registration methods: a survey. Image and Vision Computing **21**: pp. 977–1000
360. Zwillinger D., Kokoska S. (2000) CRC Standard probability and statistics tables and formulae. Chapman and Hall/CRC, Boca Raton, Florida, USA

Index