Institut für Signalverarbeitung
Universität zu Lübeck
Direktor: Professor Dr.-Ing. Alfred Mertins

Statistical Pattern Recognition for Biometric Person Identification and Event Detection

Hysteresis and Sparse Classifiers, Dynamic Bayes Networks and the Gaussianity Assumption

Habilitationsschrift
zur Erlangung der *venia legendi* für das Fach Informatik
– Sektion Informatik/Technik –
Universität zu Lübeck

vorgelegt von
Dr.-Ing.
Alexandru Paul Condurache

Lübeck, 2012

Preface

This work has at least a threefold purpose. It is intended as fulfillment of a requirement on my path towards obtaining the academic degree of a lecturer. It is also intended as a detailed description of some new signal-analysis methods that resulted form my research from the last five years that have passed since achieving my PhD. This description is as detailed as I thought needed to allow the dissemination of these methods in the community, beyond the space-limited contributions to various journals and conferences. Finally, and in my opinion very important, this work is intended as a clear description of some signal-analysis related topics that I have encountered until now during my research and for which I have found no unified explanation, having to combine a large set of sources to "get the idea" before being able to implement in software and therefore use such methods. I hope this description will allow my students to better follow me during my lectures, meeting their (justified) complains about my calligraphy at the blackboard.

This work has evolved naturally from my old and constant interest in security applications. After concentrating on medical image analysis during my PhD, I felt the time was right to get back to one of the reasons why I got into image analysis and pattern recognition in the first place. For the record, I am not obsessed with big brother, I just find such applications extremely interesting and challenging and fun. Similar methods can be used as well to improve the welfare of elderly persons or the quality of medical services. Therefore I name the main practical purpose of this work for what it is: security applications, with a focus on person identification and event detection. Changing the focus of my research from medical to security applications was more difficult that I've imagined and easier than I've been told. The reason for this rather contradictory statement resides in the overlap in methods aimed at these two application fields. In my experience, working on interesting medical-related topics simply evolved into working on even more interesting security-related topics, therefore, here, in a way there is an increased focus, but definitely also a broader perspective.

This work is structured into four main parts. I have tried to organize each part as didactically sound as possible, going from general simple concepts to more difficult ones in small steps, keeping an eye on the way these methods evolved from a historical perspective. The first part contains an overview of the used methods. It includes explanations on various standard methods used in later chapters as well as connections between them. The next three parts are structured along the lines of a real-world pattern recognition system and represent the fruits of my research in: feature extraction, in the form of the Gaussianization transform, classifier design in the form of the hysteresis classifier – a topic that I have started during my PhD. and completed here – and, finally, the adaptation of various methods, like the sparse classifier or the conditional random fields, to specific security-related problems, like person identification and event detection.

At this point I would like to thank particularly Prof. Alfred Mertins for the constant support he has shown me, for the liberty he gave me into following my own research topics in his

institute and for the fruitful collaboration that we continue to enjoy.

I would also like to thank my colleagues and friends at the University of Lübeck: Prof. Erhardt Barth, MSc. Ole Jungmann, Prof. Ulrich Hofmann, Prof. Thomas Martinetz, M.Sc. Dierck Mattern, Dr. Radoslaw Mazur, Dipl.-Inf Florian Müller and MSc. Matthias Pohl for being there with me and for me in these last years. This work has been possible due to two great teachers that tragically, both passed away well ahead of their time: Prof. Vasile Buzuloiu and Prof. Til Aach. In memoriam.

I am grateful to my parents Anca and Şerban Condurache for emboldening me to follow my dreams and supporting me while I did this. I particularly cherish the memory of my father who regretfully also passed away far to soon. His sharp wits, love, care and his open, balanced and trustworthy demeanor molded me into what I am today, so he was, is and will remain part of me.

Last, but by no means least, my gratitude and all my love goes to my children, Iulia and Flavius and my wife Andreea. They are the reason for me being me and doing anything.

To my Family

Renningen, 01 November 2017
Alexandru Paul Condurache

Contents

Chapter 1

Introduction

Our ability as humans to recognize patterns has constituted an advantage, and as such we excel at recognizing patterns. Therefore, many of our activities imply per default the recognition of patterns. The advent of computers, with their unique abilities and possibilities to help and support us in our activities, has brought with it the interest into teaching them how to recognize patterns. On the way from the "possibility" to support us to the "ability" to support us, we need to teach a computer to recognize patterns as well. *Pattern recognition*, as a computer science discipline, is instrumental on this path.

Pattern recognition is a broadly defined term. A pattern can mean an object, like for example a cup (i.e., a receptacle with holder), but it also can be the way a person behaves, or some configuration of amplitude values in a signal recorded by some sensor. An instance is an individual reproduction of a pattern, like for example a blue cup, seven centimeters high and five centimeters in diameter. In general, one can say that the purpose of pattern recognition is to find an interesting pattern in a collection of patterns. In practice, each pattern exhibits certain properties that can be measured and translated into numerical features. Each pattern is then described by a set of features that have similar values for various instances of the same pattern. Then, given an unknown instance, the problem of pattern recognition is the problem of finding in a collection pf patterns the pattern whose expression the investigated instance is. After doing this, a pattern label may be assigned to the unknown instance, i.e., the instance is labeled as belonging to that pattern. Thus, the problem of pattern recognition becomes the problem of assigning labels to feature vectors. This problem can be described mathematically with relative ease as the problem of partitioning a vector space into regions. The mathematical method that yields such a partition is called a classifier, where a class corresponds to a region of the feature space. The classifier *shatters* the feature space and to do this it needs some information about the feature space. The standard way to obtain this information is with the help of a set of feature vectors that are representative for the targeted feature space. This sample from the feature space is called training set, as the classifier will learn how to separate the target feature space from this representation. In the case of a supervised classifier, the training set is already labeled, while for the unsupervised classifier, the training set is not labeled. There are also semi-supervised classifiers, where the training set contains a small set of labeled examples and a far larger one of unlabeled ones. The ability of a classifier to correctly label previously unseen feature vectors is called generalization and it represents a major performance measure.

Feature extraction represents the process that allows us to compute features for each instance of a pattern. The simplest features are direct measures of pattern characteristics that are called

raw features. These raw features are usually pledged by various problems that lead to poor performance for a classifier designed with their help. Here, *feature extraction* is the transformation that corrects (at least some of) the problems associated with raw features.

There is a myriad of classifiers available in the literature, each purposefully designed for a certain practical problem, as there is no single classifier to constantly perform optimally over all possible problems. For our purposes we will differentiate next between multi-class, binary and one-class classifiers. A multi-class classifier is able to directly assign three or more labels. A binary classifier can assign only two labels and a one-class classifier can assign only one label. Collections of binary classifiers may be used to shatter more than two classes. On the other hand a one-class classifier can be seen as a binary classifier separating one class from the rest. The difference between a one-class and a binary classifier consists in the way training is conducted. For the binary classifier the training set includes both classes, while for the one-class classifier it includes only the one class.

The dichotomy between one class and binary classifiers is emphasized here such as to underline the specifics and importance of the application areas that require the one-class classifier. The one-class classifier originates in *significance testing* and its main application area is novelty detection. In the case of novelty detection, we have a set of patterns that we know, and if an instance cannot be assigned to any of the known patterns, we assume we have detected something new. In such a setup, the training set includes only instances of the known patterns, and a classifier needs to be designed to decide whether some instance is so different from any of the known patterns that it must be the instantiation of an unknown, previously unseen, pattern. Some very important security applications, like for example event detection, fit into this framework. In the case of *event detection*, there is knowledge of the normal case and only of the normal case. An event represents something not normal and the event label is assigned when we are reasonably uncertain that it is normal.

There are several approaches to the design of classifiers, like, for example, syntactical pattern recognition where rules and sets of rules in the form of grammars are used to decide what label should we assign to a test feature vector. Many of these approaches however, are somewhat limited in the number of different types of pattern recognition problems they can accommodate, or in the way additional information about the problem at hand can be incorporated to improve the classification result. This is not the case for *statistical pattern recognition*, which is the main topic here. In this case, some probability relating label and feature vector is computed and then a label is assigned to the feature vector such as to maximize this probability. Usually, either the probability of a label given an observed feature vector or that of a feature vector given a label is used. To enable this procedure, a *probability space*[1] is defined over the feature space and an observed feature vector is considered as the realization of a random variable. A random variable is defined for each pattern or collection of patterns of interest and during training the underlying densities are estimated.

Intuitively, given an observed feature vector $\mathbf{x}$, we would like to compute the probabilities $p(\omega_i|\mathbf{x})$ of the labels $\omega_i,\ i = 1, \ldots, N_C$ and assign the observation to one of the N_C classes such as to maximize this probability. Computing this a-posteriori probability directly is usually

[1]The probability space includes a set of outcomes, a sigma-algebra defined with the help of the outcomes and a probability measure.

difficult and therefore we use the Bayes formula to express it as

$$p(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})},$$

using the probability $p(\mathbf{x}|\omega_i)$ of observation given label that can be easily estimated from the training set, the prior $P(\omega_i)$ and the evidence $p(\mathbf{x})$. This strategy is known under the name of *Bayesian classification*, but to enable it, the prior needs to be known. In theory, this prior has to be made available before starting the classification and it should incorporate knowledge about the problem at hand. In practice the prior is estimated from the training data. However this estimate can be widely inaccurate and/or not reflect the true essence of the tackled problem, in which case, even if, numerically, the classification is conducted with maximal a posteriori probability, in reality we do a poor classification job.

When working with feature vectors, the relationship among various features needs to be taken into consideration as well. The features may either be dependent or independent. Making the independence assumption is advantageous mathematically, but if untrue it may decrease the performance of the designed classifier. Bayesian classification with the independence assumption is called naïve Bayes.

To conduct classification within the statistical framework, we need to compute various types of conditional probabilities, or in other words conduct inference. The analytical approach to this problem involves complicated mathematical computations. To ease up this task, graphical models have been introduced, where conditional dependence and independence relationships are apparent after a simple optical inspection instead of cumbersome computations. A major type of graphical models are the *Bayes networks* that are going to use intensively within this work to conduct classification in particular for time series recorded within the context of event detection.

A classification system includes the following two major steps: feature extraction and classifier design. A classification system can not be conceived outside its application domain and both these steps need to be adapted to a certain problem or class of practical problems or applications. Therefore, the purpose followed here can be stated as:

to introduce novel, statistical methods for both feature extraction and classification, while concentrating on security applications, in principal person identification and event detection.

Accordingly, the three main topics and the novel contributions contained within the pages of this book are:

(i) **feature extraction** where a novel nonlinear feature-extraction transform called multiclass Gaussianization will be discussed.

(ii) **classification** where a new type of classifier, the relative hysteresis classifier, is described within the hysteresis classification paradigm.

(iii) **person identification and event detection** in which context the sparse classifier is introduced for such applications and also new stochastic-signal analysis algorithms are discussed and some established ones are adapted for event detection.

The rest of this chapter offers in Sections 1.1, 1.2 and 1.3 a short and focused description of these three main topics that are afterwards detailed in the main body. At this stage, they are

also placed in the proper context and the basis and the intuition on which they lay is underlined. Finally, Section 1.4 concludes this introductory chapter by showing how this topics fit together, but this time from a historical-conceptual point of view.

Organization of this book

After establishing in Chapter 2 a theoretical basis, Chapter 3 describes the multiclass Gaussianization feature extraction transform that is supposed to compensate for hidden assumptions of Gaussianity with respect to the densities of various random variables. Chapter 4 contains the hysteresis paradigm, which offers new ways to introduce prior knowledge into classifier design for improved results. The relative hysteresis classifier is a new development within this framework and improves upon available hysteresis classifiers. Finally, in Chapter 5, complete classification systems are described, including both purposefully designed feature extraction methods and classification algorithms. Concepts described in the previous chapters are put to use and also various other methods are adapted or introduce for the first time to security-related applications, mainly biometric person identification and event detection.

1.1 The Gaussian assumption

Generally, algorithm design is based on some intuitive insight of the designer in the problem at hand. This intuition usually comes from the way the designer perceives the reality surrounding him. In the case of signal-analysis algorithm design, this sort of intuition leads more often than wanted to poor solutions, because it does not correspond to the underlying reality of the analyzed problem. One major example in this direction is the intuition that change is not sudden and strong, but rather slow and small. We have this intuition, because it helps us infer from some examples what is going to happen next, we thus know what to expect and get prepared. We are accustomed to reason this way.

We can apply this intuition to a multitude of cases including observations from a random variable whose true distribution is not known, in which case we expect differences between consecutive observations to be small. In the case of pattern recognition, this small-change assumption is equivalent to assuming that instantiations of the same pattern differ by a small amount, and equivalently the classes cluster around a center in the feature space. If we decide to model this intuition statistically, then we usually do this by means of the *Gaussian assumption*, which amounts to assuming that the underlying statistic in the respective case is Gaussian. If we make the Gaussian assumption, then we expect a certain thing to happen or some small variations of that thing, but not large variations, or equivalently, we expect that any two consecutive observations from such a distribution are very similar to each other. Conversely, large variations are possible, but their probability is small and decreases the larger the variation.

Even though the intuition about small change is correct in many cases it is by far not always correct. It has nevertheless led, together with other factors, to the development of a myriad of methods spanning the entire signal-processing and analysis spectrum, which are optimal only under the Gaussian assumption, when this intuition is correct. These methods are in general characterized as elegant from a mathematical point of view and, as discussed before, intuitive, which has contributed strongly to making them ubiquitous.

Here the question is addressed of what can be done if the data we analyze does not support the Gaussian assumption? There are three possible answers to this question: (i) we can come

up with new analysis methods, not related to this assumption, or (ii) we can try to make the data Gaussian before applying the Gaussian-methods, or (iii) we could ignore the underlying assumption of the Gaussian-methods and simply apply them to the data. In the latter case, we are either hoping that the Gaussian methods will behave at least satisfactory, or we are considering that their limitations are unimportant for the problem at hand. The path followed here is the second one, new means being introduced to make the data more Gaussian [44], or equivalently to conduct *Gaussianization*. The concept of "Gaussianization" implies thus a transformation to change the distribution of the input variable to Gaussian.

Gaussianization is conducted in this case for pattern recognition purposes, thus the aim is to transform the analyzed data such that it follows for each class a Gaussian distribution, while at the same time keeping its informative power. At this stage it is intuitively clear that our multiclass Gaussianization transform should modify the data at a level that can be achieved only be nonlinear transformations. As discussed in more detail in Chapter 3, a nonlinear transform has virtually complete control over the input data, the challenge in this case being to compute the parameters of this transformation such that the original information available in the data is still present after applying the transform. A regularizing term is instrumental on this path.

Next we dwell into the Gaussian assumption and discuss some major application areas where this is encountered in Section 1.1.1. Afterwards, in Section 1.1.2 it is briefly described what can be done when the assumption is not valid, thus paving the way for introducing the Gaussianization transform in Chapter 3.

1.1.1 Gaussian signal analysis

As long as the input data is really Gaussian, the Gaussian assumption is valid and we can enjoy all its advantages. Besides the relationship between human intuition and Gaussian assumption, there are also other reasons that make this assumption appealing. One example is related to the statistical properties of this distribution. These give us the possibility to investigate complex statistical relationships by relatively simple mathematical means, like for example independence relationships considering only moments up to the second order.

Next we discuss in more detail some major areas of application where the Gaussian assumption has a large impact, like Mean Square Error (MSE) approximations and various classification and feature extraction methods. The purpose here is not to present an exhaustive list, but rather to show that a thorough analysis of the Gaussian assumption is justified by the number and importance of the signal-analysis domains it is encountered. During this entire contribution the Gaussian assumption will be pointed out and discussed, as for example in the cases of parameter estimation and dynamic Bayes networks in Chapter 2.

Mean square error approximations

Perhaps the most often encountered example of the implicit Gaussian assumption is that of the Mean Square Error (MSE) approximations. By MSE approximations we understand the class of problems where we would like to find the optimal representation of some data in terms of some procedure, where the degree of success is measured by the MSE between the data and the output of the procedure.

Here it is argued that the MSE approximations are related to the Gaussian assumption because they represent the optimal solution only when the approximation error is Gauss distributed

typically with zero mean. This means that for each observation we expect the error to be zero with equally probable positive or negative deviations and this expectation is well estimated by the mean of the error distribution. If the approximation error has for example a skewed distribution, and thus the mean does not represent the best estimate for what to expect[2], the MSE criterion will lead to methods that concentrate on reducing large low-probability errors and thereby loose sight of smaller high-probability errors. The total error would be minimized, but depending on the application this may not lead to the desired result. In such cases other objective functions are needed, like the one used in the case of the sparse PCA in Section 1.1.2. Indeed only under the Gaussian assumption do a large number of cost functions, including the MSE, lead to the same optimal result (see also the Paragraph **Parameter estimation and the Gaussian assumption.** in Section 2.1.1).

In this context, the Gaussian assumption and implicitly its consequences are often simply ignored, not because the Gaussian assumption holds, but because these consequences are either not important for the problem at hand or they are offset by the advantages of working in a Gaussian environment. Next we will discuss shortly the statistics of MSE approximations and their link to other objective functions in relation with the Gaussian assumption. We will then discuss some examples that illustrate the relationship between this type of approximations and the Gaussian assumption.

Approximations in a Gaussian environment. We assume we have several noisy observations $\mathbf{x} = \hat{\mathbf{x}} + \mathbf{n}$ of some true measurement vector $\hat{\mathbf{x}}$, with $\mathbf{n}$ some random noise. We would like to use the observations to compute a good estimate $\tilde{\mathbf{x}}$ of the true measurement vector. We measure the quality of the estimate by means of a cost function $C(\mathbf{x}, \tilde{\mathbf{x}})$ and we choose the approximation that minimizes the expectation of the cost function:

$$E\left\{C(\mathbf{x}, \tilde{\mathbf{x}})\right\} = \int C(\mathbf{x}, \tilde{\mathbf{x}}) p(\mathbf{x}) d\mathbf{x}.$$

If we take the cost function to be $C(\mathbf{x}, \tilde{\mathbf{x}}) = (\tilde{\mathbf{x}} - \mathbf{x})^2$, we obtain a MSE type of approximation. The solution of the minimization problem is obtained by setting the corresponding derivative over the minimization variable to zero, obtaining thus:

$$\tilde{\mathbf{x}} = \int \mathbf{x} p(\mathbf{x}) d\mathbf{x}.$$

The sought estimate is therefore given by the mean of the density of the observations [174].

Various cost functions lead to different estimates. For example taking the cost function to be $C(\mathbf{x}, \tilde{\mathbf{x}}) = |\tilde{\mathbf{x}} - \mathbf{x}|$ we obtain an estimate that is the median of the density of the observations [114]. If instead we take an uniform cost function defined as

$$C(\mathbf{x}, \tilde{\mathbf{x}}) = \begin{cases} 0 & \text{for } |\tilde{\mathbf{x}} - \mathbf{x}| \leq \delta \\ 1 & \text{otherwise} \end{cases},$$

which is minimized when the maximum error is minimal, we obtain an estimate that is given by the maximum of the density of the observations [174].

If the density of the observations is Gaussian (or equivalently the density of the noise is Gaussian with zero mean), all cost functions from above lead to the same estimate, because the

[2]The median would represent in such a case a better estimate for our expectation.

mean, the median and the maximum of a Gaussian density are all at the same position. As a matter of fact it can be shown that if this density is symmetric about its mean, as in the case of the Gaussian density, then any cost function that is a convex function of the distance $|\tilde{\mathbf{x}} - \mathbf{x}|$ leads to the same estimate [176]. Therefore, the MSE estimate represents the optimal estimate returned by a large number of cost functions only under the Gaussian assumption, otherwise, it is the best quadratic estimate and depending on the application it may not be the best estimate to use.

The Principal Components Analysis. The purpose of Principal Component Analysis (PCA), as a feature extraction tool, is to provide an optimal dimensionality-reduced representation of some data. For this purpose we seek a transformation between the original and the reduced feature space such that structure (i.e., information) available in the original feature space is kept in the transformed feature space as well. As a loss function, we use the MSE between the data in the original space and the reconstruction of the data from the transformed space. In other words, we search for a transformation such as to be able to successfully predict/estimate the original data from the reduced data, or equivalently, we look for the best data approximation method, where we use the MSE to measure the quality of the approximation. The PCA is a supervised method, i.e., it yields a data-dependent transformation.

In the case of the PCA we have a set of observations $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$ and we would like to approximate each $\mathbf{x}_l \in \mathbb{R}^n$ by $\tilde{\mathbf{x}}_l = \sum_{i=1}^{m} \alpha_i \mathbf{u}_i$, with $m < n$, such that the MSE computed as $\mathcal{E} = E\{\|\varepsilon\|^2\}$, with $\varepsilon = \mathbf{x} - \tilde{\mathbf{x}}$ is as small as possible. A shown in detail in Appendix B, it results that the orthonormal basis vectors $\mathbf{u}_i,\ i = 1, \ldots, m$ are the eigenvectors of the correlation matrix of $\mathbf{x}$. The components $\alpha_i = \langle \mathbf{x}, \mathbf{u}_i \rangle$ of the representation of $\mathbf{x}$ with respect to this basis are uncorrelated.

It appears that selecting orthonormal components for data approximation represents the best solution. However, this is not always true, as it can be seen in countless examples. Selecting orthonormal components for data approximation represents the best solution for the specific loss function we have used, i.e, the mean squared error. However, this loss function does not optimally accommodate the structure of the data, except for one particular case. To optimally accommodate the structure of the data means to reveal all dependencies between the various components of the data. This means that in the optimal feature space, where our data truly lives, the components are independent of one another. The PCA however, offers only decorrelation, which is sufficient for independence only under the Gaussian assumption. From here we infer that the mean squared error does a good job at measuring how far we really are from an optimal approximation only if the approximation error is Gaussian distributed.

Yet another way of looking at the connection between PCA, MSE approximations and the Gaussian assumption is by means of the intuition about small and slow change. Given an input representing a distorted representation of some true item, the PCA returns, by the nature of its MSE objective function, an approximation of the true item under the assumption that the differences (or equivalently the change) between the true item and the distorted representation taken as input are overall small. For example, in the case of pattern recognition, when the item is actually a vector containing characteristics of some object, starting from a distorted input vector, the PCA will return an approximation vector under the assumption that the absolute differences from each and every input component to its true value are small. In other words, with PCA, for any observation, one would expect that these differences are zero or very close

to zero. The appropriate statistical model for such a setup is the Gaussian distribution[3].

Other estimates and estimation methods. In a linear setup, where the observations and the vector to be estimated $\mathbf{a}$ are related in a linear manner, such that $\mathbf{r} = \mathbf{S}\mathbf{a} + \mathbf{n}$ (see Section 2.1.2), the least squares estimate $\hat{\mathbf{a}} = \mathbf{A}\mathbf{r}$ is the most efficient estimate when the noise $\mathbf{n}$ is white and Gaussian. To show this, consider that the covariance matrix of the least-squares estimate is

$$\begin{aligned}
\text{cov}[\hat{\mathbf{a}}] &= E\left\{(\hat{\mathbf{a}} - \mathbf{a})(\hat{\mathbf{a}} - \mathbf{a})^H\right\} \\
&= E\left\{(\mathbf{A}\mathbf{r} - \mathbf{a})(\mathbf{A}\mathbf{r} - \mathbf{a})^H\right\} \\
&= E\left\{\mathbf{A}\mathbf{n}\mathbf{n}^H\mathbf{A}^H\right\} \\
&= E\left\{(\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H\mathbf{n}\mathbf{n}^H\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1}\right\} \\
&= (\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H E\left\{\mathbf{n}\mathbf{n}^H\right\}\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1}, \qquad (1.1)
\end{aligned}$$

with $\mathbf{A} = (\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H$. As the error process is white, we have that

$$E\left\{\mathbf{n}\mathbf{n}^H\right\} = \sigma^2\mathbf{I},$$

thus, equation 1.1 may be rewritten as:

$$\begin{aligned}
\text{cov}[\hat{\mathbf{a}}] &= \sigma^2(\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1} \\
&= \sigma^2(\mathbf{S}^H\mathbf{S})^{-1} \\
&= \sigma^2\boldsymbol{\Phi}^{-1}. \qquad (1.2)
\end{aligned}$$

Because the error process is Gaussian, the elements of $\mathbf{n}$ being uncorrelated are also independent, and their joint probability density function is a product of scalar component densities. The Fisher information matrix (see Appendix A) can then be computed as [92]

$$\begin{aligned}
\mathbf{J} &= \frac{1}{\sigma^4}E\left\{\mathbf{S}^H\mathbf{n}\mathbf{n}^H\mathbf{S}\right\} \\
&= \frac{1}{\sigma^4}\mathbf{S}^H E\left\{\mathbf{n}\mathbf{n}^H\right\}\mathbf{S} \\
&= \frac{1}{\sigma^2}\mathbf{S}^H\mathbf{S} \\
&= \frac{1}{\sigma^2}\boldsymbol{\Phi},
\end{aligned}$$

and therefore

$$\mathbf{J}^{-1} = \sigma^2\boldsymbol{\Phi}^{-1}.$$

From equation (1.2) we can see that that $\sigma^2\boldsymbol{\Phi}^{-1}$ is the covariance matrix of the least-squares estimate $\hat{\mathbf{a}}$, and thus:

$$\text{cov}[\hat{\mathbf{a}}] = \mathbf{J}^{-1}. \qquad (1.3)$$

Conversely, the least-squares estimate is unbiased, and for any unbiased estimate $\tilde{\mathbf{a}}$ we have the Cramér-Rao bound defined as:

$$\text{cov}[\tilde{\mathbf{a}}] \geq \mathbf{J}^{-1}. \qquad (1.4)$$

From equations (1.3) and (1.4) we can see that, $\hat{\mathbf{a}}$ verifies the Cramér-Rao bound with equality and it is thus the most efficient estimate, with the smallest possible variance.

[3]The small-change model in this case implies isotropic Gaussian noise with zero mean.

Classifiers

A very popular statistical nonlinear classifier is the quadratic classifier. For a quadratic classifier, the separation function is a quadratic function of the inputs. The usual way to compute the parameters of a quadratic classifier implies assuming that the class-conditional densities are Gaussian. In the simplest case, for a binary classification problem, the quadratic classifier stems directly from the likelihood ratio test under the Gaussian assumption.

With the likelihood ratio

$$l(\mathbf{y}) = \frac{p(\mathbf{y}|\omega_1)}{p(\mathbf{y}|\omega_2)}$$

the decision rule is:

$$\text{assign } \mathbf{y} \text{ to } \begin{cases} \omega_1 \text{ for } l(\mathbf{y}) \geq \lambda \\ \omega_2 \text{ for } l(\mathbf{y}) < \lambda \end{cases} .$$

Under the Gaussian assumption, and defining

$$\begin{aligned} h(\mathbf{y}) &= -2\log(l(\mathbf{y})) \\ &= (\mathbf{y}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(\mathbf{y}-\boldsymbol{\mu}_1) - (\mathbf{y}-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1}(\mathbf{y}-\boldsymbol{\mu}_2) + \log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\right) \\ &= \mathbf{y}^T(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})\mathbf{y} + 2(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1)^T\mathbf{y} + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 + \log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\right), \end{aligned} \tag{1.5}$$

the decision becomes

$$\text{assign } \mathbf{y} \text{ to } \begin{cases} \omega_1 \text{ for } h(\mathbf{y}) \leq T \\ \omega_2 \text{ for } h(\mathbf{y}) > T \end{cases} ,$$

with $T = -2\log(\lambda)$. By inspecting equation (1.5) we observe that it is a quadratic function

$$h(\mathbf{y}) = \mathbf{y}^T\mathbf{A}\mathbf{y} + \mathbf{b}^T\mathbf{y} + c,$$

with

$$\begin{aligned} \mathbf{A} &= \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1} \\ \mathbf{b} &= 2(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1) \\ c &= \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 + \log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\right). \end{aligned}$$

The decision boundary $h(\mathbf{y}) = T$ is a quadratic surface. If we now continue on our path of making stronger assumptions, we obtain a linear classifier if we assume that the class-conditional densities have equal covariances and the nearest mean classifier, if we further assume that the covariance matrices equal the identity matrix.

Feature extraction methods

Many feature extraction methods make at some point the Gaussian assumption, with which assumption they either live or explicitly try to overcome. Again the best known example is the

PCA[4] that, as already pointed out, works under the Gaussian assumption. Yet another widely used example is the Linear Discriminant Analysis (LDA), whose purpose is to reach a separable feature space. Although best known as conducted with the help of the Fisher discriminant, the LDA has at its roots the Gaussian assumption. It can be shown that when modeling the data with Gaussian distributions, irrespective of the number of classes, the search for a linear transform under the constraints that the covariances are equal and the means have reduced rank leads to LDA [82].

In the previous section we have discussed Gaussian classifiers for binary classification problems. If we would like to shatter more than two classes, within the same framework, we define for each class discriminant functions $g_i(\cdot)$, $i = 1, \ldots, N_\omega$, with N_ω the number of classes, and decide as:

$$\text{assign } \mathbf{y} \text{ to } \omega_i \text{ when } g_i(\mathbf{y}) = \max_k g_k(\mathbf{y}).$$

Each discriminant function is computed as the class conditional a posteriori probability under the Gaussian assumption:

$$\begin{aligned} g_k(\mathbf{y}) &= p(\mathbf{y}|\omega_k)P(\omega_k) \\ &= \frac{P(\omega_k)}{(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{y}-\boldsymbol{\mu}_k)\right]. \end{aligned}$$

The entire classification procedure can be considered now a type of discriminant analysis, i.e., an analysis of the feature space with the help of discriminant functions. Assuming further that the class-conditional covariances are equal, in which case the discriminant functions are linear in their input, it follows that in this case we are conducting a *linear discriminant analysis*.

Further insight may be gained by returning to the linear binary classification case, where $N_\omega = 2$. With equal class-conditional covariance matrices $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$, we have:

$$\begin{aligned} \mathbf{A} &= \mathbf{0} \\ \mathbf{b} &= 2\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \\ c &= \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2. \end{aligned}$$

The decision is reached by comparing $h(\mathbf{y}) = \mathbf{b}^T\mathbf{y} + c$ with T, or equivalently

$$\mathbf{b}^T\mathbf{y} \underset{\omega_1}{\overset{\omega_2}{\gtrless}} T - c. \tag{1.6}$$

Looking carefully at (1.6) we notice that what we actually do is to transform $\mathbf{y}$ with the help of $\mathbf{b}$, and then conduct the classification in the transformed space $x = \mathbf{b}^T\mathbf{y}$. We notice that the dimension of the transformed space equals one, i.e., the number of classes minus one, and the transformation reaches a subspace of the original feature space where the classes are highly separable.

Now, going the other way around, we may ask ourselves what is the optimal linear transformation to reach a highly separable subspace of the original feature space. To answer this question, we define a criterion function to describe separability, that we optimize over the parameters of the transform. Intuitively, and this intuition is in itself related to the small-change

[4]Here, in comparison to the previous section, the PCA is used as a feature extraction transform that is supposed to reach a feature space where the vector components are decorrelated.

assumption and the "Gaussian" way of thought, two classes are separable if their means are well apart and their standard deviations are by comparison to the distance between the means small. Notice that our intuition considers only moments up to the second order. Separability-related relationships above the second-order moments are neglected. Conversely, relationships involving moments larger that two are interesting only in a non-Gaussian environment. This means that our intuition works as intended only in a Gaussian feature space. For two classes, with equal prior probabilities and $x = \mathbf{w}^T\mathbf{y}$, the criterion function coming out of this intuition is

$$\begin{aligned} J(\mathbf{w}) &= \frac{(\mu_1^x - \mu_2^x)^2}{\Sigma_1^x + \Sigma_2^x} \\ &= \frac{(\mathbf{w}^T\boldsymbol{\mu}_1^{\mathbf{y}} - \mathbf{w}^T\boldsymbol{\mu}_2^{\mathbf{y}})^2}{\mathbf{w}^T\boldsymbol{\Sigma}_1^{\mathbf{y}}\mathbf{w} + \mathbf{w}^T\boldsymbol{\Sigma}_2^{\mathbf{y}}\mathbf{w}} \\ &= \frac{\mathbf{w}^T(\boldsymbol{\mu}_1^{\mathbf{y}} - \boldsymbol{\mu}_2^{\mathbf{y}})(\boldsymbol{\mu}_1^{\mathbf{y}} - \boldsymbol{\mu}_2^{\mathbf{y}})^T\mathbf{w}}{\mathbf{w}^T(\boldsymbol{\Sigma}_1^{\mathbf{y}} + \boldsymbol{\Sigma}_2^{\mathbf{y}})\mathbf{w}} \\ &= \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}, \end{aligned} \tag{1.7}$$

with $\Sigma_i^x,\ i = 1, 2$ the variances in the transformed space, $\mathbf{S}_B = \sum_{i=1}^2(\boldsymbol{\mu}_i^{\mathbf{y}} - \bar{\boldsymbol{\mu}}^{\mathbf{y}})(\boldsymbol{\mu}_i^{\mathbf{y}} - \bar{\boldsymbol{\mu}}^{\mathbf{y}})^T$ the between-class scatter matrix and $\mathbf{S}_W = \sum_{i=1}^2 \boldsymbol{\Sigma}_i^{\mathbf{y}}$ the within-class scatter matrix, while $\bar{\boldsymbol{\mu}}^{\mathbf{y}} = \sum_{i=1}^2 \boldsymbol{\mu}_i^{\mathbf{y}}$. Equation (1.7) represents the well known Fisher discriminant for two classes[5]. To find the maximum of the Fisher discriminant criterion, we set its derivative to zero and obtain

$$\begin{aligned} 0 &= \frac{d}{d\mathbf{w}}J(\mathbf{w}) \\ &= (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\frac{d\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{d\mathbf{w}} - (\mathbf{w}^T\mathbf{S}_B\mathbf{w})\frac{d\mathbf{w}^T\mathbf{S}_W\mathbf{w}}{d\mathbf{w}} \\ &= (\mathbf{w}^T\mathbf{S}_W\mathbf{w})2\mathbf{S}_B\mathbf{w} - (\mathbf{w}^T\mathbf{S}_B\mathbf{w})2\mathbf{S}_W\mathbf{w}. \end{aligned}$$

With $\mathbf{w} \neq 0$ we have that

$$\begin{gathered} (\mathbf{w}^T\mathbf{S}_W\mathbf{w})2\mathbf{S}_B\mathbf{w} - (\mathbf{w}^T\mathbf{S}_B\mathbf{w})2\mathbf{S}_W\mathbf{w} = 0 \Leftrightarrow \\ \left(\frac{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}\right)\mathbf{S}_B\mathbf{w} - \left(\frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}\right)\mathbf{S}_W\mathbf{w} = 0, \end{gathered}$$

which taking into account equation (1.7) leads to:

$$\mathbf{S}_B\mathbf{w} - J\mathbf{S}_W\mathbf{w} = 0.$$

To solve this generalized eigenvalue problem we premultiply $\mathbf{S}_W^{-1}$ to obtain

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} - J\mathbf{w} = 0,$$

and thus the standard eigenvalue problem $\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = J\mathbf{w}$ that yields then

$$\begin{aligned} \mathbf{w}_{optim} &= \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \\ &= \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \end{aligned}$$

[5]The extension to more than two classes is straightforward, as shown in Appendix A.

which is equivalent (up to a constant) to $\mathbf{b}$ from equation (1.6), if we approximate the common class-conditional covariance matrix in the usual way, as:

$$\boldsymbol{\Sigma} = P(\omega_1)\boldsymbol{\Sigma}_1 + P(\omega_2)\boldsymbol{\Sigma}_2.$$

As pointed out before, the LDA does not only make the Gaussian assumption, but it implicitly considers also the class-covariances to be equal. This latter assumption is renounced in the case of the Heteroscedastic Discriminant Analysis (HDA) [158, 124]. The quest for separability criteria and the corresponding dimensionality reducing transforms that work without the Gaussian assumption has ignited a lot of research. Nevertheless, many of these methods have their origin in the theory reviewed here and therefore in an intuition linked to the Gaussian assumption.

1.1.2 Alternative formulations for non-Gaussian data

What can we do when our (Gaussianity-related) intuition does not suit the problem at hand? As previously discussed, we have three possible options. However, if we really need to do something about it, we have only two real options: we can either try to come up with new methods that work without the Gaussian assumption, thus effectively *forgetting about Gaussianity*, or we could make our data Gaussian and proceed with the familiar methods, thus *imposing Gaussianity*. Clearly the lion's share in this dichotomy is taken by the former approach. Nevertheless, there are practical applications where the latter approach is used as well, many of them related to speech processing [157, 56]. Next we are going to discuss in more detail some methods that work without the Gaussian assumption. Imposing Gaussianity is one of the main contributions of this work, being detailed in Chapter 3. In this section, its principle is only briefly described.

Forget about Gaussianity

In principle there are two ways to forget about Gaussianity: (i) take some Gaussian method and modify it such that it works without the Gaussian assumption and (ii) design a completely new method, that has nothing to do with the Gaussian assumption from the start. Both approaches are encountered in the literature, the former perhaps a bit more often, as it seems more easy to go in small steps, i.e, concentrate on overcoming limitations of existing methods rather than breaking with the past and come up with an all together new method. Nevertheless, completely new methods bring the largest improvements. An example in this direction is the path from the *Kalman filter* to the *Particle Filter* in the analysis of random signals. Next we are going to discuss both possibilities with the help of a few examples.

Robust Gaussian methods. For our purposes here, a good example of Gaussian methods adapted to non-Gaussian environments is the kernel PCA [159]. In essence the kernel PCA has two steps: a nonlinear transform to a high-dimensional space followed by applying the PCA in the transformed space. The main idea being that in the high-dimensional space, our data lives in an linear subspace that can be found by PCA. Once we have found it, we can start enjoying all advantages of a feature space whose main variance modes are linear (i.e., data that spreads along lines rather than other curves). One such advantage is the increased probability (particularly if we select the optimal variation modes in the new representation) that a linear classifier could successfully shatter this space.

Yet another related example is that of the sparse PCA [186, 24] (see also Section 2.3). The sparse (also robust) PCA considers the problem of estimating some true data from corrupt instances. With respect to equation (B.2) from Appendix B, the observed data is $\mathbf{x}$, the true data $\tilde{\mathbf{x}}$ and the corruption is ε, we have therefore $\mathbf{x} = \tilde{\mathbf{x}} + \varepsilon$. Assuming the noise ε is densely supported[6] and Gaussian, the standard PCA does a very good job. In the case of the *sparse PCA*, however, we discuss the case when the noise is sparsely supported and arbitrarily distributed.

As in the case of the standard PCA, we assume we have at our disposal a larger dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with n vectors of length m. Next, we arrange these n vectors as the columns of a matrix $\mathbf{D} \in \mathbb{R}^{m \times n}$. Similarly we get for the true data the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and for the noise the matrix $\mathbf{E} \in \mathbb{R}^{m \times n}$. Thus, we would like to recover a low-rank[7] matrix $\mathbf{A}$ such that $\mathbf{D} = \mathbf{A} + \mathbf{E}$, with $\mathbf{E}$ a sparse corruption matrix. This is an optimization problem with constraints, whose solution is the optimal pair $(\mathbf{A}_o, \mathbf{E}_o)$ found as [186]:

$$(\mathbf{A}_o, \mathbf{E}_o) = \arg\min_{\mathbf{A},\mathbf{E}} \left[\text{rank}(\mathbf{A}) + \gamma \left\|\text{vect}(\mathbf{E})\right\|_0\right] \text{ subject to } \mathbf{A} + \mathbf{E} = \mathbf{D}.$$

As discussed in more detail in Section 2.3.1, this is a NP-hard optimization problem. Nevertheless in certain conditions, its solution is the same as that of the equivalent problem [186, 24]

$$(\mathbf{A}_o, \mathbf{E}_o) = \arg\min_{\mathbf{A},\mathbf{E}} \left[\left\|\mathbf{A}\right\|_* + \lambda \left\|\text{vect}(\mathbf{E})\right\|_1\right] \text{ subject to } \mathbf{A} + \mathbf{E} = \mathbf{D},$$

with $\left\|\mathbf{M}\right\|_*$ the nuclear norm, computed as the sum of singular values of $\mathbf{M}$. In practice a suitable value of the Lagrangian multiplier $\lambda > 0$ is needed. A good choice for λ is $\lambda = m^{\frac{1}{2}}$. Furthermore, for algorithm complexity reasons, the equality constraint is replaced by a penalty term and thus we end up solving

$$(\mathbf{A}_o, \mathbf{E}_o) = \arg\min_{\mathbf{A},\mathbf{E}} \left[\mu \left\|\mathbf{A}\right\|_* + \lambda\mu \left\|\text{vect}(\mathbf{E})\right\|_1 + \frac{1}{2}\left\|\mathbf{D} - \mathbf{A} - \mathbf{E}\right\|_F^2\right],$$

with $\left\|\mathbf{M}\right\|_F$ the *Frobenius norm* defined as the square root of the sum of the squared singular values of $\mathbf{M}$ and μ a small constant.

Given a distorted observation, the standard PCA returns an approximation in a Gaussian setup, i.e., following the intuition that change is small. The question is what happens with PCA when this intuition is false? What if (to the limit) the change from the observed value to the true value is very large for just one component and zero for the rest? Intuitively, the PCA approximation will be such that every component of the approximation will include some of this large change, even if the respective component in the input had the true value. In other words, the PCA will distribute the large change in one component among all components in the approximation, thereby acting according to the intuition that the change from the true vector to the distorted vector can be only small overall. It will thus introduce relatively small errors in every vector component for the sake of one large error in a single component. The sparse PCA is then an adaptation of the standard PCA to such a (non-Gaussian) setup.

[6]Densely supported is another way of saying that the noise affects all components of $\tilde{\mathbf{x}}$ equally, or equivalently it is stationary with respect to the 1D signal given by the position-ordered succession of components of $\tilde{\mathbf{x}}$.

[7]Clearly the absence of noise decreases the information content and thus the noise-free data-matrix $\mathbf{A}$ has a reduced rank.

Non-Gaussian methods In this case, instead of overcoming Gaussianity, we now work with methods that are designed to be free of it. As already mentioned, a typical example in this direction is the particle filter (see Section 2.2.1). Yet another example is the Independent Components Analysis (ICA).

In principle the ICA asks for independence and, in comparison to PCA, it is thus able to successfully work in non-Gaussian environments. The practical implementation of this principle is at least problematic in some cases as various ICA methods lead to different results. Often, the ICA transform is implemented as a rotation of whitened data [180], using moments up to the fourth order in the process. However, is not guaranteed that a density is fully described by moments up to the fourth order. Nevertheless, there are implementations of ICA like Comon's algorithm [35] and Infomax [13] that theoretically could take into consideration dependencies above the fourth order moments.

We consider this to be a good example for the abuse of the Gaussian intuition that often leads researchers to think in Gaussian terms, even if they explicitly try to avoid it.

Impose Gaussianity

Instead of devising new methods that work without the Gaussian assumption or adapting existing ones, we could simply modify the input data, previous to processing it, such that it is Gauss distributed. Modifying the statistic properties of some data such that it is Gaussian distributed has been proposed in various contexts from density estimation [33] to signal analysis [140].

Here, this concept of Gaussianization is proposed for pattern recognition problems in the form of the multiclass Gaussianization feature-extraction transform [44], [48]. In this case, in contrast to the other Gaussianization methods, we are interested that each class-conditional density is Gaussian, not that the distribution of the entire data set after the transformation is Gaussian. We are interested in Gaussianity, while keeping separability. This additional constraint is new in the field of Gaussianization transforms and it provides the justification for the novel methods developed for this purpose, which are described in detail in Chapter 3.

The Gaussianization feature-extraction transform can be thought of performing in a similar manner to the first step of kernel PCA, i.e., reach in a nonlinear way a transformed space where the Gaussian assumption holds, such that we could, e.g., apply PCA to get all the advantages previously mentioned. Nevertheless, in comparison to the kernel methods, the transform described in this work is an explicit transform, from which not only PCA, but also all other Gaussian methods could benefit. Furthermore, due to computational constraints, the kernel PCA actually works only in a certain subspace of the space that could theoretically be reached by the transform corresponding to the used kernel, therefore its optimality in this respect is not guaranteed.

Gaussianization is conducted here in a supervised manner, with the help of a nonlinear transform whose parameters are computed such that they optimize a similarity measure between the estimated true distribution of the data and its ideal distribution. The ideal distribution is in our case a Gaussian mixture with one component for each class. This similarity measure is simply the sum of squared differences between the distribution nonparametrically estimated from the training data and the mixture parametrically (i.e., under the Gaussian assumption) estimated from the same training data. The approach used here is inspired from the performance analysis of nonparametric, kernel-based density estimation (see Section 2.1.1). Intuitively, as our target is Gaussianity, using the sum of squared differences, with its relationship to the

Gaussian assumption is justified.

As nonlinear transform the elastic transformation is used, similar to that proposed for tasks such as image registration. By its elastic constraint, this nonlinear transform is diffeomorphic, a property that is paramount to a successful Gaussianization for pattern recognition applications. This means that even though in the transformed space the data is not perfectly Gaussian, its "structure" and thus separability is preserved. Thus, a transformation is proposed that actually makes the data as Gaussian as possible while preserving the separability. Furthermore, there is no dimensionality reduction associated with this Gaussianization. The purpose followed here is not to find a nonlinear Gaussian subspace that is separable, but to modify the original feature space, such that it becomes as Gaussian as possible, while, again, keeping its separability.

1.2 Novell hysteresis classification methods

A classification problem implies distinguishing concepts. These concepts are gathered in a problem space. To solve a classification problem with the help of a computer we need a projection of the problem space that can be handled by mathematical means. This projection is constructed with the help of measurements or raw features. We consider that in the measurement space, the concepts are mapped to classes. In the problem space, concepts seldom overlap, ducks are definitely different from swans. In the measurement space however, classes usually overlap, taking size and weight as features, one will find that some ducks appear to be swans and some swans appear to be ducks. An adequate measurement process is the way to preserve the problem-space separability in the measurement space, however in many applications such a measurement process is simply not available. At the same time, the raw features can not solve separability issues in the problem space. Assuming separability is achieved in the raw-feature space, the next step would be to design a function over it that takes different values for different regions and define these regions such that they correspond to the classes. This function is called a classifier and the process by which the regions are assigned to classes is called training. Often it is easier to find the optimal classifier in a map of the measurement space rather than in the measurement space itself. This map of the measurement space is called the feature space and it is generated with the help of a feature-extraction transform. There is a plethora of feature-extraction transforms. The simplest ones effectively only select some of the raw features (i.e., the feature selection methods). Their transformation matrix contains only ones and zeros. More involved feature-extraction transforms may increase, decrease or conserve the dimensionality of the measurement space in the feature space and implement highly non-linear maps.

To avoid confusions, we divide the feature extraction process into two parts: the measurement part and the analysis part. In the first part we measure the problem space, often with dedicated hardware thereby constructing a measurement or raw-feature space. In the second part we analyze the measurement space and compute features. As mentioned above and discussed in more detail later the analysis step is needed to be able to effectively and efficiently find the rules by which classes can be separated, i.e., the classifier. Sometimes analyzing the measurements and finding the separation rules are combined into a single classification method[8].

Generating a good feature space is a difficult endeavor. Conversely, choosing the classifier seems to be the simplest thing to do in designing a classification system. There are modern

[8]Classification method is used next to describe the way by which classes can be shattered. Thus a classification method can include either a feature extractor and a classifier or only a classifier.

classifiers, proven to be universal approximators Ű meaning that they can perfectly fit any continuous separating surface in the feature space Ű which makes the choice of a classifier apparently self-evident. However, appearances are often deceiving as practically choosing the perfect architecture for the classifier and/or finding the separating surface from the available training sample turns out to be very difficult or even impossible to do with the same classifier for all possible classification problems.

In practice we start with a good sample of the problem space, meaning that we have data covering all and only situations from the problem space such that its statistical structure is well reflected. Then we extract features and *build a training sample*. The usual way to conduct feature extraction is to take a large number of measurements that have something to do with the problem space and hope that somewhere there in that raw feature space there is enough separability for a classification method to shatter the classes with a high-enough margin of success. While searching for the optimal classification method we may or may not apply a feature-extraction transform and may or may not modify the training sample. At each and every step along this path, the algorithm designer has to take various decisions while considering concepts such as the *curse of dimensionality*, the *ugly duckling*, the *bias-variance tradeoff* [91] and the *no free-lunch* theorem [66] that are briefly discussed in Section 1.2.1.

Feature extraction significantly simplifies the classification but may also lead to serious problems when conducting classification. Nevertheless, this step is unavoidable, as it is needed to grasp the problem space and construct a solution. An attempt is undertaken here to put things in a new perspective, for which purpose we need to ask, if it is possible to also explicitly bring the problem space to bear directly on the likelihood term of a generative approach to a classification problem or on the posterior in a discriminative approach instead of just implicitly with the help of feature extraction? Assuming that the problem is well posed and the *no-overlap condition* is respected in the problem space, the answer to this question is yes! In the hysteresis-classification framework that at this stage is briefly introduced in Section 1.2.2, to be then discussed in detail in Chapter 4, information from the problem space is used to improve the classification taking place in the feature space. We provide thus new ways to make use of prior knowledge.

1.2.1 Issues in statistical pattern recognition

Pattern recognition needs to deal with a set of issues, mostly related to the way computer-science experts go around designing and using such methods. Being aware of these issues represents the starting point for improving our methods or designing new better ones.

The curse of dimensionality. The curse of dimensionality is a well-known problem within the pattern recognition community. It describes the decrease in performance of a classification algorithm of constant architecture, with the decrease of the fraction of cardinality of the training sample and dimension of the feature space (e.g., increase the size of the feature vector while keeping the number of feature vectors in the training set constant). For a simple explanation, one should first be aware of the fact that in order to properly shatter a feature space, a classifier needs a certain minimal density of training-set data points per volume. Now, the volume increases exponentially with the dimension and therefore, to ensure that this data density is sufficient, so should the number of data points. In general, with k the number of samples in a training set, ρ the data density and d the dimension of the feature space, we may write that $k = \rho^d$, which

implies that $\rho = k^{1/d}$. In practice a data density of 10 is sometimes considered satisfactory[9]. The curse of dimensionality relates to the fact that a classifier trained with a training set where the data density is smaller than needed, will perform worse (in particular on unseen examples) than the same classifier trained on a training set with sufficient data density. For example, take a classifier A trained in an N-dimensional feature space on the basis of a set of k samples, this classifier will perform worse if it is trained in an M-dimensional feature space with $M \gg N$, using a set of k samples.

The real question is why is there a minimal data density needed? Intuitively we can see that locality is important in taking decisions, meaning that in order to find the class of a point in the feature space we should ideally evaluate information from points in its vicinity. If the data density is small, then there are large swath of training space where we don't have any evidence about how the classes are distributed. In order to be able to shatter these regions we need to interpolate a separation surface between available data points that lay far away. Thus classification decisions on points within these regions be essentially taken based on distant points from the training set, therefore ignoring locality. We can also see that decisions for points away from the training samples is taken ignoring locality and decisions for points close to the training samples is taken considering locality. This is the intuition for overfitting when under the curse of dimensionality.

Building a training sample. In general, the curse of dimensionality is related to a set of issues around the problem of how to sample properly the random variable describing the decision problem that needs to be solved. The correct number of samples depends on the information content of the problem, which is in practice related to the information capacity of the measurement vector (i.e., the output of the procedure by which we extract information from the problem), which in turn depends on the number of individual measurements in the vector. Assuming the measurement procedure is optimally adapted to the problem at hand, the information content of the problem and the capacity of the measurement vector are equal and thus the feature-vector random variable completely describes the problem at hand. Then, successful classification depends on the quality of the sample extracted from this random variable. A good-quality sample has to have a cardinality that allows it to be properly distributed over the feature space such as to ensure the theoretical reconstruction of the true underlaying statistical model of the problem. At the same time, it has to have a cardinality in relation to dimensionality to allow the training algorithm of a classifier to practically reconstruct and harness the underlaying statistical model of the problem in a successful way.

A first aspect is that for low-density training sets, the chances of having in there all datapoints configurations needed to unequivocally find the optimal separating surface are small, so the classifier can not be designed and is not prepared to properly classify new unseen data, in which case it is said to generalize poorly. Take for example a binary classification problem in a 2D feature space with a training set of just two examples, one per class. With this training set, following Occam's razor, the optimal classification solution is a linear classifier whose separation surface goes eventually through the middle of the line linking the two points in the training set, but is otherwise arbitrary oriented (as long as it separates the two examples). However, if in reality each class is Gauss distributed with different means and covariance matrices, a linear

[9]This is related to the a minimal tolerable variance of the estimate of the mean of a one dimensional Gaussian random variable.

classifier will do a rather poor job, as a quadratic classifier is needed. Should the data density have been higher, we would have had more chances of observing a configuration of training-set data points where the linear classifier does not work and thus we would have been able to see that we need to come up with a better solution.

At the same time it should be considered that there is a problem with the number of training examples versus the dimensionality of the feature space only under the assumption that the training sample is properly extracted from the feature space[10]. Should this not be the case, the performance remains poor even for very large (in relation to the dimensionality) training sets, as the data again does not capture the entire variability of the problem space, even if the number of examples in the training set would suggest otherwise. Note that in this case the data density over the entire features space is small even if it high over some parts of the feature space. In other words, the data density is not simply a question of the sheer number of training samples with respect to the size of the feature space, but also a question of the spread of the training samples. For some classifiers this is related as well to the fact that during training certain directions in the feature space are overemphasized. Meaning that the classifier relates mostly to these features to separate the classes. This search and usage of the most separable features is usually a desired property of the training process[11] but one which works against the generalization performance if the training sample is not adapted to the variability in the problem space such that it leads to emphasizing the false features due, e.g., to the fact that the available training sample covers only part of the entire feature space.

Another aspect only partially related to the first, is related to the complexity of the separating surface and thus of the classifier[12]. In general, the model capacity needs to be optimally adapted to the information content[13] in the problem at hand. However, often you have excess model capacity, because you have the necessary model capacity but to few data (i.e., poor sample) or enough data but too much model capacity or few data and too much model capacity. The curse of dimensionality issue afflicts powerful classifiers strongly than simpler ones for the same data density. The more complicated the separating surface, i.e., the higher the model capacity, the tighter the fit it is able to achieve on the training set, where a tight fit results in very good classification results on the available training set. Intuitively, this is because the more parameters the separating surface has the more difficult it is to set them up properly in particular in regions of the feature space with low density [14]. Even more, if we have high model capacity, we need a higher data density even in the regions that would otherwise have sufficient data density. This means that there is a minimal average data density at which a classifier of optimal complexity could successfully shatter the features space, however, classifiers of higher model complexity

[10]By properly extracted we mean that it covers the entire feature space and in doing so it theoretically provides the base on which to correctly estimate the a-posteriori probability for each class.

[11]In particular when considering that very often many features do not contribute to the separability in the feature space, as discussed more thoroughly in the section about "ugly duckling".

[12]In general, the more powerful the classifier, the more complicated the separating surface it can model and thus the higher the number of classification problems that it can solve perfectly.

[13]The maximal manageable information content in a problem can be estimated from the information capacity of the raw measurement vector. For example in the case of a 1024×768 pixel gray level image with eight bit per pixel, this is: $1024 \times 768 \times 2^8$ bits. It is important however to note that this is usually well in excess of the typical information content of the recognition problem at hand. The true information content in the problem at hand is estimated with the help of the validation data.

[14]Equivalently, we can say that the more possible separating surfaces you can interpolate, the higher the chances of getting the wrong one.

need a larger data density to function properly. If we have model capacity that is in excess of the available training sample and the training sample is poor, this tight fit will accommodate only the limited number of configurations present in this low-density training set and therefore it is a good chance that other configurations will not be properly classified. If we have model capacity that is in excess of the available training sample and the training sample is good, the tight fit will wrap around the sample components and not the problem, as the classifier will not learn the problem, but will spend the excess modeling capacity to learn by heart the examples in the training set. Therefore, again, the classifier will have a poor generalization performance, as it will work properly on the training set and the training set only, thus overfitting on this training set. Yet another aspect is that the number of possible decision surfaces increases with the dimension of the feature space when the data density is hold constant. Thus under such conditions the probability of finding the correct surface[15] decreases and thus training becomes more difficult. However, increasing the data density serves to increase this probability [16].

Ugly duckling. The ugly duckling theorem says that there is no predefined optimal set of features, and therefore, for each problem a particular set of features is needed. Conversely, this particular feature set is related to the technical framework of the respective problem, as well as the knowledge and therefore the bias of the person that designed the feature extraction process.

The ugly duckling theorem argues along the following lines: at the most general level ignoring any type of problem-related knowledge, features are logical predicates on patterns and similarity among patterns is defined as the number of shared logical predicates. Within this setup however, it is shown that any two patterns are similar.

More formally, considering for ease of understanding binary[17] features f_i. To characterize each pattern[18] we may use logical predicates defined over these features. Therefore a pattern may have several predicates. The logical predicates are in turn characterized by their rank. The rank is a natural number related to how general of a description the predicate may provide. Higher rank predicates are more general, such that they represent a suitable description for several patterns. Conversely lower rank predicates are more particular, rank one predicates being characteristic for a single pattern. Highest-rank predicates are characteristic for all patterns. Such predicates that describe either all patterns or no pattern at all are considered self evident. It can be demonstrated that the total possible number of unconstrained predicates excluding self-evident ones is $N = 2^n - 2$, where n is the number of regions in which the used features have divided the pattern space.

As shown in Figure 1.1, the Venn diagrams of features divide the pattern space into regions. In this example, the pattern for the white region has one rank-one predicate $f_1 \text{AND} \bar{f}_2$, three rank-two predicates $f_1, f_1 \text{XOR} f_2, \bar{f}_2$ and three rank four predicates $f_1 \text{OR} f_2, \bar{f}_1 \text{OR} \bar{f}_2, f_1 \text{OR} \bar{f}_2$. At the same time, the rank one predicate is specific for the white region alone, the rank two predicates are shared by the white region with one other region and rank three predicates are shared with unordered sets of two other different regions. Rank four predicates are self evident,

[15] This probability would be computed as the number of favorable cases divided by the number of possible cases.

[16] This happens because the number of possible cases decreases.

[17] In this case, a binary feature is related to the presence or the absence of an attribute for the pattern for which we conduct the feature extraction. The binary feature takes the value one when the attribute is present in the investigated pattern or zero when this is not. For example for the pattern chair, the attribute may be the backrest.

[18] A logical predicate is for example, $f_1 \text{AND} \bar{f}_2$, where $\bar{f}_2$ means NOT f_2. This logical predicate has the value one for a pattern where the binary feature f_1 is present (true) and f_2 is not present (false).

as they cover the entire set of four possible regions.

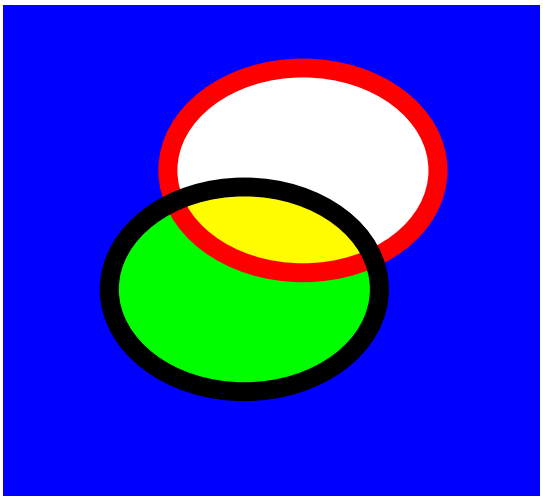


Figure 1.1: Shown here is the pattern space of a 2D binary-feature vector $\mathbf{v} = [f_1, f_2]^T$. The red Venn diagram corresponds to f_1 and the black one to f_2. Outside the red diagram you can find $\bar{f}_1$ and outside the black one $\bar{f}_2$. This 2D feature vector defines four regions in the pattern space: blue, white, yellow, and green. In the white region for example, the rank one predicate $f_1 \text{AND} \bar{f}_2$ is true.

It is important to notice that the same region-division of the pattern space can be achieved by different pairs of 2D features, so our discussion is independent of the particular choice of features.

Similarity between two patterns can be defined as the number S of shared predicates. This number however, depends on the number of patterns in the pattern space, but not on the patterns themselves such that for n patterns is $S = 2^{n-2} - 1$. For our example, the pair of patterns corresponding to the white and the yellow regions share three predicates; f_1, $f_1 \text{OR} f_2$, $f_1 \text{OR} \bar{f}_2$, the same number of predicates being shared by any other of the remaining four pairs of patterns.

Then, the *ugly duckling theorem* says that:

Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates, shared by the two patterns, then any two patterns are "equally similar"[184].

This theorem is valid nor only for discrete, but also for continuous feature spaces as well. The name "ugly duckling" comes from the fact that under this theorem, without any type of additional assumptions, an ugly duck A is similar to a beautiful swan B. Assumptions need thus to be made such as to define similarity in a way that leads to a successful separation of ducks from swans. For example, we could introduce also the rank of the predicates in the definition of similarity besides their number. However, as the possible rank of the predicates is related to the strength of the independence relationships among features, this implicitly assumes that the features are chosen such that the rank can be used to define similarity, i.e., they are independent.

In a way we went here backwards from features over properties/predicates to patterns, to show that from a feature perspective there is no predefined feature space where patterns are dissimilar so choosing good, separable features and even defining similarity depends on the problem space, i.e., depends on the particular application and the knowledge and thus the assumptions that we make while designing a classification system for that particular task.

On these grounds, the usual strategy while constructing a measurement/feature space is to take as many measurements as possible even if they are loosely related to the problem space such as to be certain to capture the entire problem variability and separability in the features (usually the information capacity of the feature space is larger than the information content of the problem). The problem in this case is that if you add features in excess of the information content of the problem, these features will serve to overfit the training sample. More precisely, these will capture variability of the training sample rather than of the problem space. With this as a starting point, we may argue that the strategy of achieving separability by blindly adding raw features (i.e., measurements) to an initial feature space is flawed. More precisely, if the newly added features are dependent and thus not certain to add to the separability they may actually help confuse the classifier training. Conversely, independent features may improve the overall separability even if they themselves are not separable at all (see Ref. [89]). To avoid overfitting the training sample, a feature extraction transform or a feature/measurement selection procedure is applied once we are finished defining measurements/features. When a feature transform is used this is usually designed such that the transformed space is also tuned on the problem at hand[19]. The balance between cardinality of the feature sample and dimensionality of the feature space is important and has been discussed in the 'Curse of dimensionality' section.

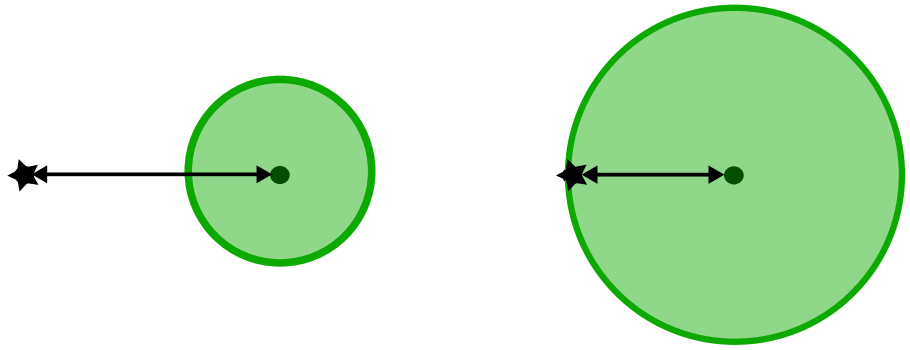
The bias-variance trade-off. Assuming we have at our disposal data that constitutes a good problem-space sample, we would like to build a classifier that generalizes well. The generalization performance is measured by the generalization error and it is influenced by the choice of feature space and classification method[20]. The generalization error can be expressed as a sum of three terms: the irreducible (Bayes) error that is related to the problem space and/or the inherent limitations of the measurement procedure, the bias and the variance of the classifier. The *bias* reflects the approximation properties or the modeling capacity of the classification method. The *variance* reflects the link between classification method and problem-space sample. The bias-variance trade-off describes the fact that there you cannot optimize both bias and variance at the same time. Figure 1.2 illustrates this concept.

A classification method has two components: a function space where all separating functions that can be implemented by the method live and a training procedure by which the optimal function can be found from some training data. Both bias and variance have accordingly two subcomponents. The first is related to the function space and the second to the training procedure. Thus, there is a model bias that has to do with the presence of the optimal classifier in the function space[21] and an estimation bias, which has to do with the ability of the training procedure to find the optimal classifier. There is also a model variance that has to do with the

[19]There are two important aspects here: representativity and separability. However, assuming the problem space is separable and the measurements capture this separability, a feature transform-based dimensionality reduction using an objective function that aims at representativity will help concentrate the separability available in the input space on less dimensions in the output space, serving thus as well a separability purpose.

[20]Here, a classifier is an instantiation of the classification method.

[21]A classification method with zero model bias over all problems is an universal approximator.



(a) (b)

Figure 1.2: Here it is shown the function space for two different classification methods. The optimal classifier is marked with a star, while the 'found' classifier is marked with a disc. The double-headed arrow marking the distance between the optimal and the found classifiers depicts the bias, while the green circle depicts the variance. (a) Large bias and small variance (b) Smaller bias and larger variance

fact that the classifier would yield the same error irrespective of where from in the problem space the test sample comes, and an estimation variance, which has to do with the ability of the training procedure to find the same classifier irrespective of where from in the problem space the training sample comes from. Virtually no machine-learning theoretical considerations on generalization deal directly with estimation bias and variance. These are usually somewhat hidden in terms measuring the performance on the training set. Often, practical machine learning follows this lead and assumes the training procedure to be optimal, concentrating on the model bias and variance.

The bias-variance trade-off is a concept instrumental in practically finding the best possible classification method for a problem. After dividing the data into a train a test and a validation set, one typically approximates the Bayes error with the human performance and measures the performance on these data subsets. Then, for example, a significant difference between the human performance and the train performance shows that we deal with large bias, while an insignificant difference between the performance on the train, validation and test set indicates a small variance. Such considerations are used to guide the search for the classification mode.

No free lunch. The no free-lunch theorem has to do with the fact that there is no unique classifier to equally well solve all classification problems and therefore each problem has its own optimal solution. The underlying assumption is that a classifier cannot know the entire feature space, because it is not trained on the entire feature space, but on a sample from the feature space, i.e., the training set, and the training set covers only a portion of the feature space. Therefore, of critical importance in this case is the generalization performance of the classifier.

The no free lunch theorem states that the generalization performance of a classifier cannot be good over all possible problems. Even more, its performance over all possible problems is constant and therefore improved performance on a certain set of problems leads to decreased performance on another set [66]. Therefore in practice we need to know for what problems a classifier is suited and apply it only there. In this context, the assumptions made by the algorithm designer have a major influence on the performance of the used classifier and as

classifier designer it is always better to be familiar with many various classification techniques than expert in just a few powerful ones.

Modern classifiers like for example kernel machines are universal approximators. Such an universal approximator will perfectly model any continuous separating surface while (at least in the case of kernel machines) taking care of the generalization performance so it apparently solves all classification problems perfectly and therefore equally well contradicting thus the no free-lunch theorem. However, even if it will model any separating surface, it can not find any separating surface efficiently, where efficiency includes [15]: computational complexity (related to the number of required computations), statistical efficiency (related to the needed training sample) and engineering efficiency (related to the needed human involvement). At the same time there are other methods that will do it, but they will in turn fail on other problems. Thus the no free-lunch theorem holds.

The "no-overlap“ condition. For well posed classification problems, the no-overlap condition expresses the intuition that in the problem space classes do not overlap. In pattern recognition one might imagine cases when the problem is ill posed, and then the no-overlap condition simply does not exist. The argumentation in this case would run along the same lines as in the case of the wave-particle dualism in quantum mechanics. However we work seldom if ever in the problem space. Normally we work in the measurement space, which represents the projection of the problem space over a measurement procedure [22]. When projecting the problem space into the measurement space the no-overlap condition often gets broken in a more or less subtle manner.

Take for example photographies of non-transparent objects in natural light recorded by means of a consumer camera with standard lenses. Each pixel belongs to either foreground or background. To segment the image – or in other words classify each pixel into one of the two possible classes – we want to use this exclusivity (i.e., the ”no-overlap“ condition) to improve our result. On the other hand, in the case of transparent objects recorded in the same way, one and the same pixel may belong to both foreground and background due to reflection effects, this being an example where the ”no-overlap“ condition does not exist in the measurement space. In both cases the measurement process implies measuring light with a camera observing the scene through standard lenses. This is adequate for the first case, but inadequate for the second, where a polarizing filter would have helped mitigate at least some of the reflections.

Statistical, density-based classification is good to decide in the region of the feature space where the classes overlap. The classification error may be reduced if the information that the classes do not overlap in the problem space, is put to use. However, we first need a way to express this type of knowledge (i.e., the knowledge that the classes do not overlap), and this depends on the particular problem.

1.2.2 Hysteresis classification

The hysteresis classification is a tool that is able to improve classification in the feature space by using the ”no-overlap“ condition of the problem space, thus offering a new venue for the usage of prior information. The discussion on hysteresis methods is focused in this work on the case of binary image segmentation, where knowledge from the problem space may be expressed

[22] Sometimes the result of measurements are called raw features.

with relative ease with the help of neighborhood relationships defined among labeled pixels. The main labeling takes place in a feature space, where each pixel has a feature vector related to its gray-level value. The final labels are afterwards computed by taking into consideration the fact that the object region is connected, as it does not overlap with the background. The foundations of the hysteresis classification framework have been laid in Reference [36]. New, powerful hysteresis classifiers for vessel segmentation, like the relative hysteresis classifiers, have been proposed in References [49, 50, 46] and represent together with Reference [42] the basis for the discussion on hysteresis classification from Chapter 4, and retina-based person identification from Section 5.1.2 of this work.

Vessel segmentation and biometric authentication

Even though we consider vessel segmentation here "only" as an application example for our hysteresis classification, it is a topic of sufficient practical importance to justify conducting research on algorithms purposefully designed for it. Vessel segmentation is useful in medical applications [112], but also in security applications, more precisely in biometric authentication. Biometric authentication, or in short *biometrics*[23], represents an integrating term covering methods aimed at establishing the identity of an individual based on some of his biological or behavioral characteristics. Often used biometric cues are for example face, palm and fingerprints, iris but also vascular patterns. Vascular pattern-based biometric authentication implies imaging the vascular patterns behind a palm- or a fingerprint, in the retina or on the sclera and using features related to this patterns for biometrics. In general it is considered safe to assume that irrespective which vascular patterns are used these are unique, specific to each individual and are not influenced by aging. Furthermore, they are difficult to forge, begin related to blood flowing through vessels within the body. The imaging of palm- and fingerprint as well as sclera vessels can be conducted even contact-less, which improves user acceptance. Conversely, the imaging of the retina vessels requires cooperation, but it is also the most difficult to forge and its uniqueness is considered fact, except for pathological cases.

Extracting features related to the vascular patterns, such as to be able to conduct biometric authentication, implies searching for feature points like vessel bendings and bifurcations. The simplest most direct way to do this is on the segmented vessel tree, for which purpose we need a fast and accurate vessel segmentation method, like hysteresis classification-based segmentation.

1.3 Security applications

Who is this guy and what is he doing there? These are rather intrusive questions, but if one wants to be certain that the guy being given the banking data to is one's finance adviser and not someone else, or if one wants to be certain that the guy by one's car is the neighbor checking his own car parked nearby and not a thief, these are precisely the questions one will be asking. So one would like to know the identity of another person and to be able to find out if the way this person acts is normal.

[23]From a historical perspective, the term *biometrics* has been first used to describe the statistical analysis of biological data. More recently, it is widely use in relation to *biometric authentication*, which represents the way we use it here as well.

Such applications represent currently an active research area. Automatically verifying and establishing the identity of a person based on his or hers biometric characteristics is already ubiquitous as it will be pointed out in Section 1.3.1. If we take for example fingerprints, they replace passwords on computers and are stored in passports and identity cards for verification purposes besides their older uses as access control and for the purpose of forensic investigations. Action recognition and abnormality- or event-detection (see Section 1.3.2) is on the verge of becoming ubiquitous. Cameras recording data that needs to be analyzed in this respect are present almost everywhere, from buses to busy intersections in many cities.

Algorithms for security applications need to be designed to meet certain requirements that arise from the very nature of their application field. The filed of security applications has several subfields. As these requirements are application dependent, they are shared by algorithms only if their respective subfields intersect. In this work the accent lies on the subfields of event detection and person identification, thus the requirements introduced in Section 1.3.3 and therefore the algorithms resulting from them are tailored to these applications.

In this work, novel solutions for problems such as fingerprint and retina-based person identification, as well as event detection will be described. These solutions include the use of the sparse classifier but also the use of new stochastic signal analysis methods and are discussed in detail in Chapter 5. The sparse classifier in particular is a versatile algorithm that can be successfully used for person verification/identification [45] [42] and that will be introduced here for the first time to event detection [47]. The CRFs and the related Maximum Entropy Markov Models (MEMMs) are stochastic models, whose application to event detection is pioneered here [135]. In the same field of stochastic-signal analysis for event detection, a new Linear Predictors Mixture (LPM) [134] will also be introduced.

1.3.1 Biometric person verification and identification

Nowadays there is an acute need to automatically relate a person accessing a certain resource to a previously established identity. This need arises from the ubiquitousness of activities and applications where such a procedure is required, like for example access control, authorization of a money transfer or login into a network. The initial approach to such problems is token based. The individuals whose access should be controlled make use of a token in the form of a card or of a password, which can identify them with respect to the controlling authority. This traditional approach is however pledged by some principle problems related to its token-based nature. The token may be stolen or lost, accidentally shared or malevolently duplicated.

The alternative to a token-based system is a biometric system, where the individual is the token. Biometric features represent a set of person-related unique properties that may identify an individual. This individual characteristics are in a majority of cases biological features like the face, the eyes, the fingerprints, the teeth, the form of his hand, the pattern of vessels on his palm, and so on. Behavioral characteristics as for example the gait, the speech the handwriting are being used as well. These biometric features can be easily acquired with the help of dedicated hardware. In some cases, like for example that of a person's face captured with a camera, this works even without explicit interaction with the acquisition hardware.

Within the framework of biometric authentication [103], a distinction is usually made between verification, meaning one-to-one correspondence and identification, meaning one-to-many correspondence. In a *verification* application one will usually check if a person is the one he or she claims to be, by comparing this persons biometric features against those of the

claimed identity. In an *identification* setup, one will establish the identity of an unknown person, usually by comparing his or hers biometric features against those of a set of known persons. The core procedure in both cases is the comparison between two sets of biometric features.

Person verification is needed in direct communications, when one needs to be certain that the one at the other end of a communication channel is really the one he pretends to be. Person identification is needed in access control, when one needs to make sure that only certain individuals are allowed to enter a secured area.

In general, person authentication algorithms need to compare sets of biometric credentials. One set is acquired from the target person and the other(s) are available in a database. There is usually a tradeoff between the quality with which the biometric credentials have been acquired and the difficulty of authenticating the respective person. The challenge is currently to conduct person authentication despite poorly acquired biometric credentials, which for the case of identification, represents our focus here as well.

1.3.2 Event detection

An event represents an unexpected pattern in the analyzed data. The detection of such events is required in many applications, and therefore, from a historical perspective, it was given many names besides event detection, like anomaly detection, rare-class mining, exception mining, chance discovery, novelty detection or outlier detection. Some applications where event detection plays a major role are: intrusion detection in network security, security monitoring, video indexing, fraud detection, biosurveillance, traffic incident detection, and quality control in manufacturing.

For security applications in particular, event detection involves usually the analysis of a time series. In this context events are rare and in this contribution, novel methods will be discussed to detect rare events in time series. A security-related example where one could envisage the use of such methods is that of a human observer monitoring the live feed of a security camera. In such a case, this person would have difficulties to concentrate over a sufficiently long period of time to effectively detect events and thus, the need for automatic event detection methods arises to support the human observer. These algorithms would constantly analyze the data flow and select for the analysis of the human observer only what they would detect as event.

Designing an event detection algorithm is challenging. The boundary between what is normal and what constitutes an event is often not precise and on top of this, what is normal and what is event usually keeps evolving. Therefore, the first task, and one of the most difficult ones, is to define what represents an event such as to be able to conduct feature extraction accordingly and gather training data representing the normal case. Four types of events have been identified in the literature [32]: point events, context events, collective events and online events. In the case of *point events*, the event observations are very different from the rest of the data and appear very rarely. Under this definition looking for point events is in some cases similar to detecting outliers. For *context events*, one no longer analyzes single observations, but a group of successive observations and marks an event if a single observation is strikingly different from the others in its group, even if the observation itself is no outlier with respect to entire data. In the case of *collective events*, we analyze several groups of observations, and mark an event if their union exhibits certain characteristics, even if each group in itself is not related to an event. Finally, *online events* are related to the case when what is considered event changes with time. However, this separation is somewhat too specific, as proper pattern definition and

feature extraction would lead in most cases to the same setup of looking for one unexpected pattern. The true difficulty is represented then by the fact that the unexpected may become expected after our event detection system has been put into place, in which case the normal case would be effectively evolving to include the event case. Therefore, we need to differentiate also between static-event applications, including point, context and collective events and dynamic-event applications, i.e., the online events. In particular the later type of applications needs special algorithmic solutions that should be seemingly retrained or that can be retrained automatically.

Most event detection algorithms use statistical modeling, including statistical classifiers. Other approaches to event detection stem from various different classification paradigms, such as for example structural pattern recognition where both graphs and grammars have been used for such purposes. Algorithms for event detection usually work on the premises that what is "normal" is often encountered and therefore easily available for training, while what is "event" is very rarely encountered and is not available for training. Defining the event as something that is not normal, event detection algorithms are usually designed and trained to recognize "normal" and decide "event" when their confidence in the recognition result is too small. This strategy is followed here as well and is applied to the analysis of time series.

1.3.3 Algorithms for security applications

We introduce next a set of requirements and performance indicators to guide the development of algorithms for security applications. Adam and Rivlin [4] introduced a set of operational requirements to be satisfied by event-detection algorithms such as to ensure deployment in practical applications. They are:

- Simple and fast tuning for a given video stream.
- The algorithm should be adaptive, such as to accommodate an evolving normal case.
- Robustness to interferences, including hardware-induced noise but also higher-level, problem-related disruptions, like for example spurious scene motions in the case of an algorithm analyzing the behavioral pattern of an individual.
- Automatic learning and operation.
- Low computational requirements.

In the same context of event detection, but for the specific case of the analysis of human behavior, Turaga et al. [178] recognized that algorithms specifically designed for this task need to exhibit a set of invariances:

- Invariance to scaling and viewpoint.
- Execution rate invariance.
- Anthropometric invariance.

Cappelli et al. [30] established in their fingerprint verification contest a set of measures of performance for algorithms designed for such purposes. These measures consider accordingly only one-to-one correspondences. The most important such measures are:

- False match rate (FMR) and false non-match rate (FNMR).
- Failure to enroll rate[24].
- Average execution time.
- Average memory allocation.

Next the discussion will focus on algorithms and classes of algorithms that are well suited for security applications. The sparse classifier is particular is a versatile algorithm well suited for both person identification and event detection. Conversely, event detection can be successfully approached with algorithms designed for the analysis of random signals, like for example the Conditional Random Field (CRF).

The sparse classifier for person identification and event detection

According to the principle of parsimony, also known as Occam's razor, if we are to select among several explanations for some facts, then we should choose the most simple one. The sparse classifier can be considered an expression of the principle of parsimony in the case of pattern recognition. Sparsity as a data analysis and representation paradigm is currently widely investigated for various purposes like compressed sensing [144, 26, 58], but also feature extraction [54], and data compression. Sparse representations have already been used for recognition tasks [187, 115] and even for the recognition of human actions using data from a wearable motion-sensor network [189] or using video data [85]. In this context sparse-representations-based classification shares its heritage with established methods in data analysis, such as the minimum description length model selection strategy, or the Support Vector Machines (SVM). According to the minimum description length principle if we are to choose from among several algorithmic solution to a classification problem, we should choose the least complex (as in most simple) solution that allows the most compact representation of the training data. On the other hand, the support vector machines use a small (as in simple) subset of most relevant features vectors from the training set to conduct the classification such that the generalization performance is optimal.

In principle the sparse classifier attempts to describe a test sample in terms of a combination of training samples and implicitly assumes that the most compact representation is obtained using training samples from the same class as the test sample. This intuition is very similar to that used by the k-Nearest-Neighbor (k-NN) classifier [66] and as such it has been confirmed in practice countless times.

The sparse classifier is particularly robust to noise in the training space in the form of strong corruption affecting a large number of the components of the feature vector. Furthermore, in some cases, when using the sparse classifier the choice of the feature extraction process becomes second to the size of the feature space. It is thus more important to have a dimension of the feature space that is large-enough to be able to properly compute the sparse representation inherent to the sparse classifier, than to choose features that are particularly representative or lead to a particularly separable feature space. Nevertheless, good separability in the feature space is still a necessity.

[24]Enrolling means here extracting (from a raw fingerprint image) the information that will be processed by the fingerprint recognition system to return a decision.

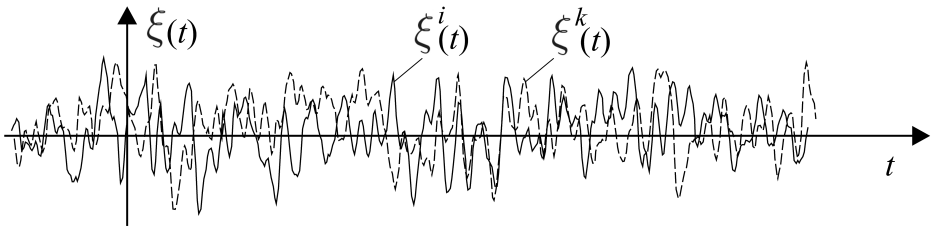


Figure 1.3: A random signal.

The sparse classifier is properly used for person identification when there are several measurements available of the same biometric traits for each person in the database used for identification. For event detection, it is optimal if the sparse classifier uses a training set in which each sub-case of the normal case is accordingly labeled, as discussed later in Chapter 2 and in Chapter 5.

Event detection by analysis of random signals

Conducting event detection often has an inherent time component that the sparse classifier can take into account only at the feature level. As in this case we analyze time series of feature vectors, it should be instrumental to take this into account during the classification phase as well. The most general assumption to be made with respect to a time series is that it represents a stochastic or random signal and thus we concentrate here on the way random-signal analysis methods can be used for event detection.

Random signals require special handling, as the tools used to describe deterministic signals are not always good for random signals. A random signal ξ depends upon two parameters the realization index k and the time index t such that we can write $\xi^{(k)}(t)$. For a particular time index t_1, $\xi^{(k)}(t_1)$ is a random variable. A certain realization k_1 corresponds to a time signal $\xi^{(k_1)}(t)$. A random signal is illustrated in Figure 1.3.

A random signal is called a random process if t is continuous and a random series if t is discrete. To describe the statistical properties of random signals we use measures developed for random variables. For $t = t_1$, with $x_1 = \xi(t_1)$, the distribution function is $F_1(\chi_1) = P(x_1 \leq \chi_1)$, where we've dropped the index k for simplicity reasons. The density function is then $p_1(\chi_1) = \dfrac{\partial F_1(\chi_1)}{\partial \chi_1}$. The statistical description of the random process is improved if we have access to the joint distribution and density functions defined as

$$F_n(\chi_1, \ldots, \chi_n) = P(x_1 \leq \chi_1, \ldots, x_n \leq \chi_n)$$

and

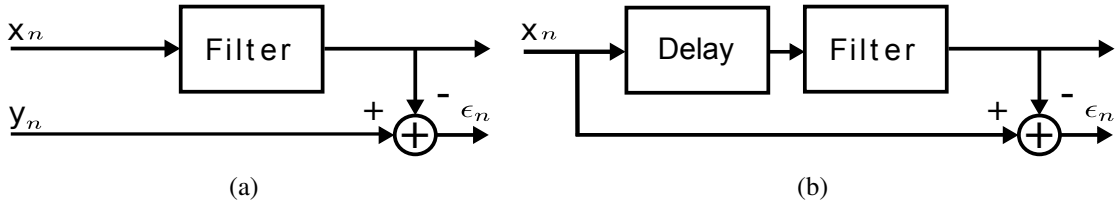
$$p_n(\chi_1, \ldots, \chi_n) = \frac{\partial^n F_n}{\partial \chi_1, \ldots, \partial \chi_n}$$

respectively.

A prerequisite for event detection is change detection, and depending on the application the separation between this two concepts is more or less significant. In general an event is always a change, while a change is not always an event. To detect changes and events, random signals are processed by methods like the Wiener filter, the Kalman filter, the particle filter or related stochastic models such as the Hidden Markov Model (HMM), the MEMM and the CRF.

As we discuss next, for event detection dedicated algorithmic chains are built around different characteristics of such methods.

Residual-based change and event detection. Stochastic models for random signals are generally used in two main setups, both equally well suited for event detection: as a *filter* where observations including the one at the current time step are processed and as a *predictor* where only observations up to the previous time step are used. In keeping with the literature [88] we will refer here generically to filters to describe stochastic methods for residuals-based event detection. Filters can be seen as residual generators, as shown in Figure 1.4, and as such they can be used for change and event detection. Assuming that $y_n = y(n)$, the ideal "normal-case" signal that contains no event is available, if there is no change in the observed signal x_n, then the residuals ϵ_n are a white noise process, when the filter is tailored to the normal case. A variation in the statistical properties of the residuals is an indicator for change. Should y_n not be available, one can connect the filter in predictor configuration with $y_n = x_n$ and proceed.



(a) (b)

Figure 1.4: Residual generators: (a) Filter, (b) Predictor configuration.

The main problem is to decide when is the change, as captured in the variation of the residuals, significant. For this purpose decision or stopping rules are used. A stopping rule includes: (i) a test statistic s_n that is a function of the output (usually the residuals) of the filter, (ii) some test function $G(s_n)$ that is supposed to extract from the test statistic the information needed to take a decision, and (iii) a way to take this decision, which happens by comparing $G(s_n)$ against a threshold T.

Change detection algorithms must decide among two hypothesis:

$$\begin{array}{ll} H_0: & \text{no change} \\ H_1: & \text{change.} \end{array}$$

With the help of the test function, the decision is taken as:

$$\begin{array}{ll} \text{Decide } H_0 \text{ if} & G_n(s_n) < T. \\ \text{Decide } H_1 \text{ if} & G_n(s_n) \geq T. \end{array}$$

Two popular test functions are (i) the CUmulative SUM test (CUSUM) [10] and the Geometric Moving Average test (GMA) . For the CUSUM, the test function is defined as

$$G_n = \max(G_{n-1} + s_n - \upsilon, 0),$$

with υ a parameter of the test. For the GMA we have:

$$G_n = \lambda G_{n-1} + (1-\lambda)s_n,\ 0 \leq \lambda \leq 1.$$

The test statistic s_n is computed from the filter residuals ϵ_n. Some of the statistics that can be used are:

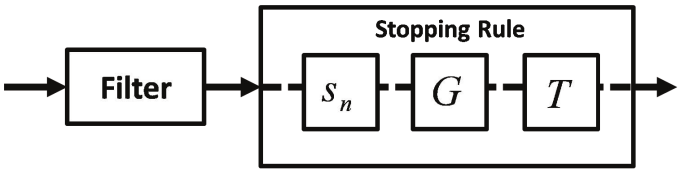


Figure 1.5: Change detection with one filter.

- $s_n = \epsilon_n$ that relates to changes in the mean of the residuals.
- $s_n = \epsilon_n^2 - \beta$ that relates to changes in the variance of the residuals.
- $s_n = \epsilon_n x_{n-k}$ that relates to changes in the correlation between residual and input.
- $s_n = \epsilon_n o_{n-k}$ that relates to changes in the correlation between residual and output.
- $s_n = \text{sign}(\epsilon_n \epsilon_{n-1})$ that uses the fact that a white zero-mean process changes its sign every second sample on average.

The stopping rule measures essentially the deviation from the no-change hypothesis. Change is detected when this deviation is too large. This entire procedure, for the one-filter case is illustrated in Figure 1.5.

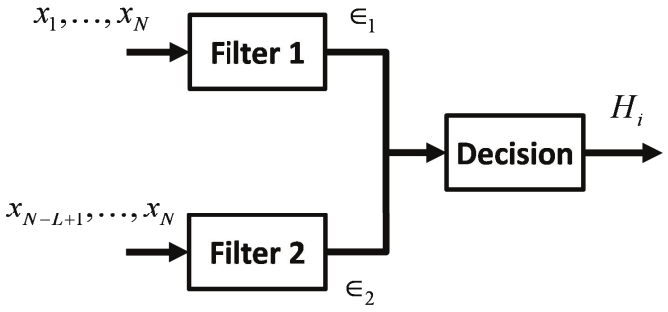


Figure 1.6: Change detection with two filters.

Two filter approaches are used in applications where one would like to detect slower-varying changes and ignore fast ones. In this case a filter based only on recent data is compared to a filter that includes a larger history. Recent data for the first filter is extracted from a short sliding window. The choice of L, the length of the short sliding window, is then critical. The other filter either considers all data available until the processing time or data extracted from a larger sliding window.

$$\text{Data: } \overbrace{x_1, x_2, \ldots, x_{N-L}, \underbrace{x_{N-L+1}, \ldots, x_N}_{\text{Filer } F_2}}^{\text{Filter } F_1}$$

Theoretically, when the filter based on the larger data window has significantly larger residuals then the one based on the shorter data window, a change is detected. The difficulty consists in choosing an appropriate test statistic, related to the residuals of both filters and deciding what are small residuals. This test statistic can then be used, for example, with the CUSUM test to obtain a stopping rule. Change detection with two filters is depicted in Figure 1.6.

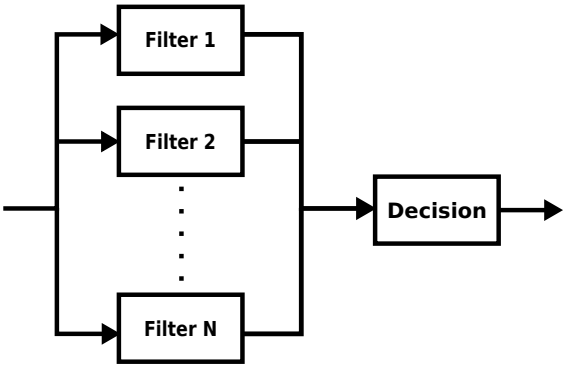


Figure 1.7: Change detection with several filters.

In the two-filter approach sudden changes were considered normal and slow ones represent the event. There are also applications where not one but several types of changes are considered normal. The solution in this case is a multi-filter approach as shown in in Figure 1.7.

In practice we often encounter such setups where the normal case has several subcases. Depending on how much training data and what kind of data we have at our disposal we can proceed in several ways. When labeled data is available - similar in this respect with the sparse classifier - a filter would be trained for each sub-case of the normal case and an event would be detected when all filters yield large residuals. When unlabeled data is available, we would have to devise a new unsupervised-training procedure that should be able to train at least one filter for each sub-case without the corresponding labels. The Linear Predictors Mixture introduced in Section 5.2.1 includes besides such an unsupervised training also a way to decide when an event has occurred. When no data is available we can use our prior knowledge on the problem space to design an expert system, where each filter component is set by hand, or alternatively, to properly set up methods that learn online in an unsupervised manner, like adaptive filters.

State-space models for event detection. Some very popular stochastic models, like some of the Dynamic Bayes Networks from Section 2.2.1 assign each observation to a discrete state that can be seen as a label. Such methods offer alternative ways to design event-detection algorithms. In this case we can define normal states/labels and event states/labels and detect an event the moment the most probable state given an observation is an event state. The difficulty resides in the fact that we typically do not have events in the set of observations used to train the model, as events are assumed very rare. Therefore the model characteristics and the way we use it to conduct inference and training must be adapted to such a setup. Such a state-space-based approach stemming from CRFs is discussed in more detail in Section 5.2.2.

1.4 From vessel segmentation to Gaussianization

This book represents a collection of research topics in the larger field of signal-analysis methods either supporting or directly used for biometric person identification and event detection. This work is not an exhaustive review of signal-analysis methods for such security applications, because this purpose, however tempting and interesting, is beyond the size limitations imposed by the context of a professorial dissertation. The various topics discussed here are related not

only as stages of a statistical decision-making process in the security field, but also as concepts evolving from each other, as pointed out next.

The process of classification is one of decision making, as we have to decide for a new sample to which class it belongs. From this perspective, segmentation is decision-making at a pixel level, as we have to classify each pixel to either background or object. The hysteresis classification paradigm, even though applicable to binary classification problems, was designed for difficult segmentation tasks and as such it is able to successfully segment vessels.

Vessel segmentation is needed in a myriad of medical applications but also in security applications. The prime example in this latter filed being that of person identification, where vessel segmentation is needed when using the vasculature as biometric cue, like for example in the case of retina vessels, but also the palm vessels or the eye-ball vessels. Retina vessels are used in high-security applications. Analyzing the palm vessel tree besides the palm print offers improved performance. The vessels are used not only as a liveliness detector, but also to improve the specificity of this method. The conjunctival vasculature can be used to improve iris-based person identification, in particular for uncooperative subjects. Vessel segmentation is generally used to extract a biometric feature vector, the identification is then conducted with the help of this vector by various classification methods.

Assuming that several biometric feature vectors of the same person are available, the sparse classifier represents a classification method very well suited to identification tasks. It offers high accuracy despite small sample size and high robustness to distortions of the biometric feature vector as well as speed, but it also offers inbuilt outlier-detection capability, which makes the sparse classifier well suited for event detection as well.

For event detection, the sparse classifier needs, besides labeled sub-cases of what has been defined as the normal case, also tailored feature extraction such that a set of conditions that allow the usage of the sparse classifier are met. Furthermore, as usually event detection for security applications implies the analysis of a time-dependent signal, this feature extraction step not only supports the sparse-classification paradigm, but also helps adapt the sparse classifier for the analysis of time signals.

Classifiers that inherently take into account the time-evolution of the analyzed data can be devised from stochastic signal analysis methods. In a large number of cases, these methods make the assumption that the data they analyze is Gaussian distributed.

To improve the appropriateness of such methods, we may want to transform the input data such that it is Gaussian distributed, thus effectively ”gaussianizing“ the data. The Gaussianization transform has then the purpose of modifying the distribution of the input data to Gaussian.

Chapter 2

Theoretical background

The purpose of this chapter is to offer a review of various methods and algorithms related to the research described in the next chapters. It is supposed to be used as a theoretical reference for the novel approaches introduced in this work. In many cases, besides introducing the needed concepts, a complete perspective over the respective topic is offered. This serves to underline certain ways of thought whose influence can be then observed in the following chapters. Each method is first put into context before being described in detail, trying as much as possible to go from simple general considerations to detail in steps as small as necessary.

Section 2.1 reviews various estimation methods for both densities and signals. Section 2.2 is dedicated to graph-based statistical inference and finally Section 2.3 to the sparse classification framework. Density estimation is used throughout the entire book, with an emphasis on the Gaussianization transform in Chapter 3, signal estimation, sparse classification and statistical inference are used in particular in the applications Chapter 5.

2.1 Estimation

Estimation theory deals with the problem of finding values from observations to which they are locked in a relationship. The estimate $\hat{\mathbf{a}}$ of the value $\mathbf{a}$ is computed as a function of the observations and we may write $\hat{\mathbf{a}}_N = f(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$ when we use N observations $\mathbf{r}_i, i = 1, 2, \ldots, N$ to compute the estimate. Usually the observations are afflicted by various disturbances and the main difficulty resides in finding a good estimate of the sought values. The quality of an estimate is established with the help of its properties.

Properties of Estimators. As the observations are random, the estimate itself is also a random variable. The properties of a random variable need to be described in a statistical sense. For this task we consider the process $\hat{\mathbf{a}}(\mathbf{r}) : \hat{avek}_1, \hat{avek}_2, \hat{avek}_3 \ldots$, with $\mathbf{r} = \mathbf{r}(\mathbf{a})$, where $\mathbf{a}$ is some fixed parameter vector, and discuss Bias, Consistency and Efficiency.

Bias. An estimate $\hat{\mathbf{a}}$ is called unbiased if

$$E\left\{\hat{\mathbf{a}}(\mathbf{r})\right\} = \mathbf{a}.$$

Consistency. An estimate is consistent if

$$\lim_{N\to\infty} P\{|\hat{\mathbf{a}}_N - \mathbf{a}| < \varepsilon\} = 1,$$

for an arbitrarily small $\varepsilon \geq 0$. The sequence of estimates $\hat{\mathbf{a}}_N, N = 1, 2, \ldots$ is said to converge in probability to $\mathbf{a}$.

Efficiency. An estimate is more efficient than another estimate if it has a smaller variance. If $\hat{\mathbf{a}}$ is any unbiased estimate, then the variance of any component $\hat{a}_i$ is bounded by

$$E\left\{(\hat{a}_i - a_i)^2\right\} \geq j_{ii}^{-1}, \tag{2.1}$$

with $i = 1, \ldots, L$. The lower bound j_{ii}^{-1} is an element of the diagonal of the inverse of the Fisher information matrix (see Appendix A).

If the sought parameter is a scalar, than (2.1) becomes

$$E\left\{(\hat{a}_N - a)^2\right\} \geq \frac{1}{E\left\{\left[\frac{d}{da}\ln p_{\mathbf{r}|a}(\mathbf{r}|a)\right]^2\right\}}, \tag{2.2}$$

with $\mathbf{r} = \{r_1, r_2, \ldots, r_N\}$ the set of available observations. The relation from (2.2) represents the Cramér-Rao bound for the scalar estimate $\hat{a}$. An estimate that satisfies the Cramér-Rao bound with equality is the most efficient estimate of all. If $\hat{\mathbf{a}}_N$ is unbiased and efficient with respect to $\hat{\mathbf{a}}_{N-1}$ for all N, then it is also consistent.

The properties of estimates guide the search for the best estimate for various estimation problems. We discuss density estimation in Section 2.1.1 and signal estimation in Section 2.1.2. Section 2.1.3 concludes our discussion on estimation topics with the Expectation Maximization algorithm.

2.1.1 Density estimation

To conduct statistical inference, we often need to know the *Probability Density Function* (pdf) of the involved random variables. The pdf is related to the likelihood of various events defined over the experiment described by the respective random variable. Density functions that are sufficiently common in practice are described by mathematical formulas that involve a set of parameters. The purpose of parametric density estimation is to compute these parameters from the available data. Conversely, non-parametric density estimation is used when the generic form of the density function is not known. Next we discuss both parametric and non-parametric density estimation, for further references see [183, 167, 139].

Parametric estimation

In this case of parametric estimation, we determine one or more unknown parameters of a known density function from noisy observations. The unknowns may be scalars, vectors, etc. There are two main assumptions that can be made with respect to these parameters and that we discuss separately next, the parameters can be assumed to be: (i) unknown constants[1], or

[1]This is sometimes called "The frequentist approach".

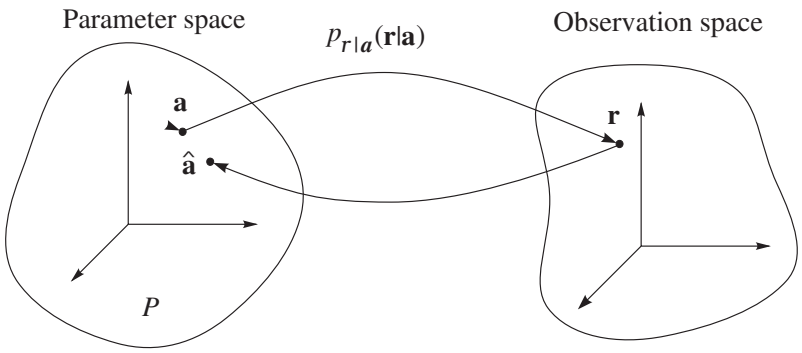


Figure 2.1: The ML setup.

(ii) random variables. Starting from a set of observations $\{\mathbf{r}_i\},\ i = 1, \ldots, N$, representing a sample from the random variable $\mathbf{r}$ that is characterized by the parametric density $p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a})$ with parameters $\mathbf{a}$, we search for methods to compute $\hat{\mathbf{a}}$ the estimate of $\mathbf{a}$ from the available sample.

When the parameter vector $\mathbf{a}$ is considered to be a random vector, we implicitly assume knowledge of the *a-priori* density $p_{\mathbf{a}}(\mathbf{a})$ and we can thus use the joint density $p_{\mathbf{r},\mathbf{a}}(\mathbf{r}, \mathbf{a}) = p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a})p_{\mathbf{a}}(\mathbf{a})$. Parametric estimation procedures that use this joint density are sometimes gathered under the name *Bayes estimation* [174]. As we discuss next, these procedures are related to each other and many of them yield the same estimate under the Gaussian assumption.

Maximum Likelihood Estimation. The Maximum-Likelihood (ML) estimation yields an estimate under the assumption that $\mathbf{a}$ is some unknown constant (i.e., a realization of a random variable). The ML estimation can be used as well when we have no knowledge of $p_{\mathbf{a}}(\mathbf{a})$ or we do not desire to let this knowledge influence our estimation. The setup valid in this case is shown in Figure 2.1.

We start by defining the *likelihood*. For a given $\mathbf{r}_i$, we can compute the following function of the parameter vector $\mathbf{a}$: $l_i(\mathbf{a}) = p_{\mathbf{r}_i|\mathbf{a}}(\mathbf{r}_i|\mathbf{a})$. The likelihood is then computed from the joint probability of the entire set of observations $\mathbf{r}_{1:N} = \{\mathbf{r}_i\},\ i = 1, \ldots, N$, under the parameter vector

$$p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a}) = p_{\mathbf{r}_{1:N}|\mathbf{a}}(\mathbf{r}_{1:N}|\mathbf{a}) = \prod_{i=1}^{N} l_i(\mathbf{a}) = \prod_{i=1}^{N} p_{\mathbf{r}_i|\mathbf{a}}(\mathbf{r}_i|\mathbf{a}),$$

assuming each realization in the available sample is conditionally independent under $\mathbf{a}$. The ML estimate of the parameter vector is then computed as:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a}). \tag{2.3}$$

As the gradient disappears at the maximum, (2.3) leads to the *Likelihood equation*

$$\frac{d}{d\mathbf{a}} p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a}) = 0. \tag{2.4}$$

Since the logarithm is a strictly monotonic function, we can also use the equivalent *Log-Likelihood equation*:

$$\frac{d}{d\mathbf{a}} \ln p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a}) = 0. \tag{2.5}$$

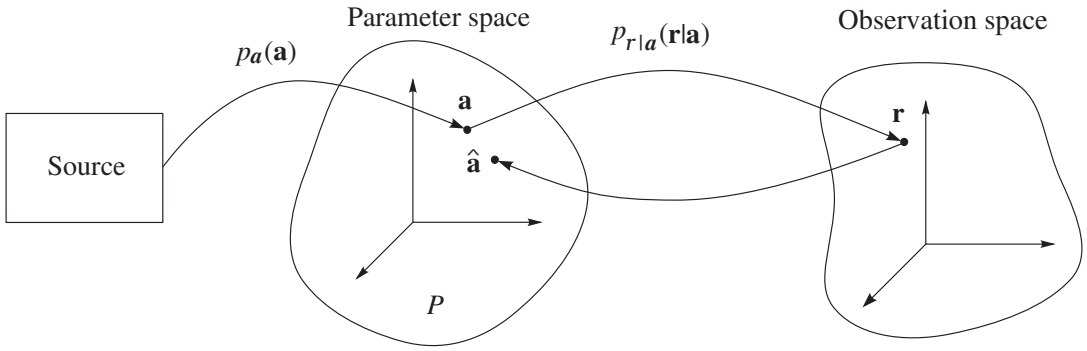


Figure 2.2: The MAP setup.

Maximum-a-Posteriori Estimation. The Maximum-a-Posteriori (MAP) estimation yields an estimate under the assumption that $\mathbf{a}$ is a random variable. In this case, the following setup is valid: a source generates a parameter vector $\mathbf{a}$ with the *a-priori* density function $p_{\mathbf{a}}(\mathbf{a})$ and $p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a})$ is the pdf of the observation $\mathbf{r}$ under the parameter $\mathbf{a}$. This is illustrated in Figure 2.2. In comparison to the ML estimation we consider that in this case we have additional information with respect to $\mathbf{a}$ in the form of $p_{\mathbf{a}}(\mathbf{a})$ and we want to use this information hoping for a better estimate. Conversely, starting from the MAP setup we could reach the ML setup by assuming that $p_{\mathbf{a}}(\mathbf{a})$ is the uniform distribution and thus any $\mathbf{a}$ is equally probable.

We look for $\mathbf{a}$ such that the *a-posteriori* density $p_{\mathbf{a}|\mathbf{r}}(\mathbf{a}|\mathbf{r})$ is maximal

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} p_{\mathbf{a}|\mathbf{r}}(\mathbf{a}|\mathbf{r}). \tag{2.6}$$

Using the Bayes rule we may express the a posteriori density as:

$$p_{\mathbf{a}|\mathbf{r}}(\mathbf{a}|\mathbf{r}) = \frac{p_{\mathbf{a}}(\mathbf{a})\; p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a})}{p_{\mathbf{r}}(\mathbf{r})}. \tag{2.7}$$

Because the evidence term $p_{\mathbf{r}}(\mathbf{r})$ in the denominator of (2.7) does not depend on the sought parameter vector, we obtain:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} p_{\mathbf{a}}(\mathbf{a})\; p_{\mathbf{r}|\mathbf{a}}(\mathbf{r}|\mathbf{a}). \tag{2.8}$$

By comparison to (2.6), (2.8) is easier to compute in practice.

The MAP estimate can be better than the ML estimate in particular when the estimation uses a limited number of measurements and thus any prior information on a is a welcomed addition. With a higher number of measurements, the ML estimate becomes asymptotically optimal what consistency and efficiency are concerned.

Bayes estimation and the Gaussian assumption. Next, instead of building the likelihood function, we assign a cost to each estimate $\hat{\mathbf{a}}$ in relation to the true parameter vector $\mathbf{a}$. This is done with the help of the non-negative cost function $C(\hat{\mathbf{a}}, \mathbf{a})$ that becomes zero when $\hat{\mathbf{a}} = \mathbf{a}$. The task we now face is to find the estimate that minimizes the risk $\mathcal{R}$, defined as the expectation of the cost function over the joint probability $p_{\mathbf{r},\mathbf{a}}(\mathbf{r}, \mathbf{a})$

$$\begin{aligned}
\mathcal{R} = &\; E\left\{C(\hat{\mathbf{a}}, \mathbf{a})\right\} \\
= &\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} C(\hat{\mathbf{a}}, \mathbf{a}) p_{\mathbf{r},\mathbf{a}}(\mathbf{r}, \mathbf{a}) d\mathbf{r} d\mathbf{a} \\
= &\; \int_{-\infty}^{\infty} I(\hat{\mathbf{a}}) p_{\mathbf{r}}(\mathbf{r}) d\mathbf{r},
\end{aligned}$$

with $I(\hat{\mathbf{a}}) = \int_{-\infty}^{\infty} C(\hat{\mathbf{a}}, \mathbf{a}) p_{\mathbf{a}|\mathbf{r}}(\mathbf{a}|\mathbf{r}) d\mathbf{a}$. Under the assumption that $I(\hat{\mathbf{a}})$ is strictly convex, to minimize the risk we need to minimize $I(\hat{\mathbf{a}})$ and thus the sought estimate is the one for which

$$\frac{dI(\hat{\mathbf{a}})}{d\hat{\mathbf{a}}} = \mathbf{0}. \tag{2.9}$$

Choosing as cost function the squared error between the estimate and the true value, $C(\hat{\mathbf{a}}, \mathbf{a}) = (\hat{\mathbf{a}} - \mathbf{a})^2$, we obtain, after solving (2.9)

$$\hat{\mathbf{a}} = \int_{-\infty}^{\infty} \mathbf{a} p_{\mathbf{a}|\mathbf{r}}(\mathbf{a}|\mathbf{r}) d\mathbf{a}, \tag{2.10}$$

and therefore, the Minimum Mean Square Error (MMSE) estimate that minimizes the expected squared error, is the mean of the a-posteriori density.

Another widely used cost function is the uniform cost function defined as [174]

$$C(\hat{\mathbf{a}}, \mathbf{a}) = \begin{cases} 0 & \text{for } |\hat{\mathbf{a}} - \mathbf{a}| \leq \varsigma \\ 1 & \text{otherwise} \end{cases},$$

which assigns a zero cost only to estimates within a small distance from $\mathbf{a}$ and maximal cost otherwise. In this case, we obtain after solving (2.9) that the corresponding estimate is given by the maximum of the a-posteriori density, and is thus the MAP estimate.

If the posterior is a Gaussian distribution, then the maximum of the density is at the position of the mean value, and therefore the MMSE estimate is a MAP estimate. Thus, both the squared error and the uniform cost functions lead to the same estimate only when the posterior is Gaussian. Furthermore, it can be shown[176] that with a Gaussian posterior[2] any cost function that is a convex function of the distance $\delta = |\hat{\mathbf{a}} - \mathbf{a}|$ leads to the same estimate (2.10). Therefore, not only the squared error and the uniform cost functions are equivalent under the Gaussian assumption but also the absolute error function $C(\hat{a}, a) = |\hat{\mathbf{a}} - \mathbf{a}|$ and any cost function that is a convex function of the distance δ.

Non-parametric estimation

In the case of non-parametric estimation, we need to find an unknown density function from a set of observations. Next we give a concise revision of kernel smoothing for density estimation, discussing both the univariate and the multivariate case.

Univariate kernel density estimation. We introduce univariate kernel density estimation starting from the histogram and continue our discussion with methods designed to analyze the performance of such estimation procedures. The methods discussed here are designed to approximate an unknown function. For this purpose they rely on a simple function called kernel. Various such functions exist and the choice of kernel is a major design step in non-parametric estimation. This simple functions have a parameter called bandwidth, which again has a major influence on the quality of the computed estimate. The discussion on univariate kernel density estimation methods is closed with practical modalities to select the bandwidth.

[2] Actually any posterior that is symmetric about its mean is enough.

From histogram to kernel. The simplest univariate non-parametric density estimator is the histogram. The histogram is a step function, the step hight being proportional to the number of observation falling in an interval of the real axis related to the step length. This intervals are called bins and they partition the real axis without overlap constructing a bin system. The histogram as density estimator is computed as

$$\tilde{p}(x;b) = \frac{\text{\# observations in the bin containing } x}{Nb}, \tag{2.11}$$

where N is the total number of available, independent observations and b is the width of a bin. An additional factor that has to be taken into consideration is where the bin starts and where it ends. For example, with respect to the origin (corresponding to zero), the bin system may be centered there, may have it as left or as right edge, etc.

Defining the bin width as $b = 2h$, the probability of an observation x falling in a bin centered at $\hat{x}$ is $P(|\hat{x} - x| \leq h) = \tilde{p}(\hat{x}) \cdot 2h$. This probability can be approximated also by the relative frequency, i.e., the number of observations $k_{\hat{x}}$ falling in a bin centered at $\hat{x}$, in which case the probability is $P(|\hat{x} - x| \leq h) = \dfrac{k_{\hat{x}}}{N}$. Then we have that

$$\tilde{p}(\hat{x}) = \frac{k_{\hat{x}}}{2hN},$$

and considering the function

$$K_h(z) = \begin{cases} \dfrac{1}{2h} & \text{for } |z| \leq h \\ 0 & \text{otherwise} \end{cases},$$

we get

$$\tilde{p}(\hat{x}) = \frac{1}{N} \sum_{i=1}^{N} K_h(\hat{x} - x^i), \tag{2.12}$$

where x_i is the i-th observation from the set of available observations $\{x_1, \ldots, x_N\}$.

In this setup $K_h(z)$ is a *kernel function* with *bandwidth* h. Equation (2.12) can be written equivalently as

$$\tilde{p}(\hat{x}) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{\hat{x} - x_i}{h}\right), \tag{2.13}$$

where we have used $K_h(u) = 1/h \cdot K(u/h)$. To ensure that the estimate is indeed a density (i.e, is positive and integrates to one) the kernel K has to satisfy the same conditions.

When using the histogram to approximate non-parametrically a density, the estimate is a step function. In the kernel setup, one can switch to other kernels (other than the histogram-equivalent uniform kernel) to generate continuous estimates. One widely used example is the Gaussian kernel, defined as:

$$K_h(z) = \frac{1}{\sqrt{2\pi h}} e^{-\dfrac{z^2}{2h}}.$$

Performance analysis. The performance of a density estimation procedure is defined by how good does the computed estimate approximate the true density. This depends on the choice of the kernel and of the bandwidth. As pointed before, various kernels lead to various estimates. But more important than the kernel choice is the choice of the bandwidth. A too small bandwidth leads to undersmoothing, while a too large one to oversmoothing.

To find the optimal kernel, various optimality criteria have been proposed. These criteria are based on the squared error between the estimate $\tilde{p}(\cdot)$ and the true density $p(\cdot)$. Considering these two functions, the Integrated Squared Error (ISE) is given by

$$ISE\ [\tilde{p}(\cdot;h)] = ISE\ (h) = \int [\tilde{p}(x;h) - p(x)]^2\ dx.$$

The ISE can be used for a single set of observations, taking into account other possible sets, we need to build the mean ISE as

$$MISE\ (h) = E\left\{ISE(h)\right\}.$$

Bandwidth selection. Based on $MISE$, efficient optimality criteria can be built that allow the computation of the bandwidth h_{opt} in close form, in relation to the kernel $K(\cdot)$ and the total number of available observations N as

$$h_{opt} = \left[\frac{R(K)}{\mu_2^2(K)R(f'')N}\right]^{\frac{1}{5}}, \tag{2.14}$$

with $\mu_2(K) = \int z^2K(z)dz$ and $R(g) = \int g(x)^2dx$, for any square-integrable function $g(\cdot)$. It can be shown [167] that when $p(x)$ is a member of the Gaussian family of distributions, equation (2.14) can be simplified to

$$h_{opt} = \left[\frac{8 \cdot \sqrt{\pi} \cdot R(K)}{3 \cdot \mu_2^2(K) \cdot N}\right]^{\frac{1}{5}} \tilde{\sigma}, \tag{2.15}$$

which leads to

$$h_{opt} \approx 1.06 N^{-\frac{1}{5}} \tilde{\sigma}, \tag{2.16}$$

with $\tilde{\sigma}$, e.g., the parametric ML estimate of the standard deviation under the Gaussian assumption. Equation (2.16) is known as *Silverman's rule-of-the-thumb* for estimating the bandwidth of a kernel.

Multivariate kernel density estimation. The extension of kernel density estimation to multidimensional random variables is not straightforward in practice, as many more bandwidth-related parameters are required. The method is strongly afflicted by the curse of dimensionality. Under these circumstances, similar to the univariate case, we discuss here besides the multivariate estimator also performance analysis methods and bandwidth selection modalities.

The multivariate estimator. The kernel-based estimate of the density of a d-dimensional multivariate random variable is given by

$$\tilde{p}(\hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}}(\hat{\mathbf{x}} - \mathbf{x}_i), \tag{2.17}$$

where $\mathbf{H}$ is the symmetric, positive definite, $d \times d$ bandwidth matrix. Similar to the univariate case, we have that $K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}} K\left(\mathbf{H}^{-\frac{1}{2}}\mathbf{x}\right)$ where the kernel K must have all properties of a density function. A popular choice for a kernel, is the standard multivariate Gaussian kernel defined as

$$K(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{1}{2}\mathbf{x}^T\mathbf{x}}$$

that leads directly to

$$K_{\mathbf{H}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{H}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{x}^T\mathbf{H}^{-1}\mathbf{x}}.$$

A further simplification is to assume that the bandwidth matrix is a diagonal matrix $\mathbf{H} = \text{diag}(h_1^2, \ldots, h_d^2)$, in which case the kernel estimator becomes

$$\tilde{p}(\hat{\mathbf{x}}) = \frac{1}{N\left(\prod_{l=1}^{d} h_l\right)} \sum_{i=1}^{N} K\left(\frac{\hat{x}_1 - x_{i,1}}{h^1}, \cdots, \frac{\hat{x}_d - x_{i,d}}{h_d}\right),$$

with $\mathbf{x} = [x_1, \ldots, x_d]^T$. In the most simple case, when $\mathbf{H} = h^2\mathbf{I}$ the number of parameters in the multivariate bandwidth reduces to a single one, and we get:

$$\tilde{p}(\hat{\mathbf{x}}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\hat{\mathbf{x}} - \mathbf{x}_i}{h}\right). \tag{2.18}$$

Performance analysis. As in the scalar case, the performance of the estimator depends on the choice of kernel and bandwidth, but in the multivariate case, the choice of bandwidth includes the type of bandwidth and the precise entries in the bandwidth matrix. Performance measures are needed to allow us to make these choices in an optimal way. As in the univariate setting, these measures have as starting point the $MISE$.

Bandwidth selection. It can be shown that if the density to be estimated $p(\cdot)$ is actually a multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the optimal bandwidth matrix is given by

$$\mathbf{H} = c\boldsymbol{\Sigma},$$

and thus the precision of the estimates depends only on d and N. If one assumes further that $\mathbf{H} = \text{diag}(h_1^2, \ldots, h_d^2)$, then the components of the bandwidth matrix of a Gaussian kernel may be estimated by

$$h_i = \left(\frac{4}{d+2}\right)^{\frac{2}{d+4}} N^{-\frac{2}{d+4}} \tilde{\sigma}_i, \tag{2.19}$$

with $\tilde{\sigma}_i$ the estimate of the standard deviation for vector component i. If all components of the diagonal of $\mathbf{H}$ are equal we have an *Isotropic Kernel*, otherwise we have an *Anisotropic Kernel*.

2.1.2 Linear signal estimation

We concentrate next on estimating random signals (see Figure 1.3). In general, for this purpose we assume that there exists a relationship linking the signal values to the available observations,

which are also afflicted by random disturbances. The problem of signal estimation can then be recast as the problem of finding the parameters of the inverse model of this relationship, such that the estimate obtained from the observations is as close as possible to the true signal.

Assuming that the relationship linking observations and values is linear we work with the following *productive model* that explains how the observations are "produced" from the true signal values

$$\mathbf{r} = \mathbf{S}\mathbf{a} + \mathbf{n}, \tag{2.20}$$

with $\mathbf{r}$ a vector, $\mathbf{S}$ a matrix, $\mathbf{a}$ another vector and $\mathbf{n}$ zero-mean noise accounting for the random disturbances. This model has several interpretations. In a possible interpretation, $\mathbf{r}$ is the observation, $\mathbf{S}$ is the relationship matrix and $\mathbf{a}$ is the true signal[3]. In an alternative interpretation, $\mathbf{r}$ is the true signal, $\mathbf{S}$ is a signal matrix with consecutive chunks[4] of the observed signal along its lines, and $\mathbf{a}$ is the inverse model[5].Thus, according to our model, the observations and true signal are linearly related in the presence of noise. Next, we will discuss methods to obtain an estimate of the vector $\mathbf{a}$.

Although at this stage there are no further assumptions on the additive noise term, the estimation procedures we describe next use only moments up to the second order. Therefore, what concerns various independence assumptions, they are suited only if the analyzed signals are white and Gaussian. Furthermore, these methods work only for stationary signals. The nonlinear and nonstationary case is discussed in Section 2.2.

We start our discussion with the least-squares estimation method that we apply at the design of the linear least-squares filter. The unbiased least-squares estimate is concerned with minimizing the error of fit between model output and observed data, but there is no assertion made about its variance. In this respect, the least-squares estimate works in the same setup as the ML estimate. When minimizing this variance we obtain another type of estimator, the minimum mean square error estimator, which works in a setup similar to the MAP estimate. We continue our discussion with the Wiener filter that represents the practical implementation of the MMSE in the case of signal estimation. Finally we describe the Wiener-filter related one-step linear predictor. For further references see [139, 92].

Least-Squares Estimator

We look for an unbiased estimate $\hat{\mathbf{a}}$, when $E\{\hat{\mathbf{a}}(\mathbf{r})|\mathbf{a}\} = \mathbf{a}$. It follows that in the case of linear estimation $\hat{\mathbf{a}} = \mathbf{A}\mathbf{r}$, the condition $\mathbf{A}\mathbf{S} = \mathbf{I}$ must be fulfilled, as $E\{\hat{\mathbf{a}}(\mathbf{r})|\mathbf{a}\} = E\{\mathbf{A}\mathbf{S}\mathbf{a} + \mathbf{A}\mathbf{n}\}$ and $E\{\mathbf{n}\} = \mathbf{0}$.

With $\mathbf{f} = \mathbf{r} - \mathbf{S}\boldsymbol{\alpha}$, the error vector, the least-squares estimate minimizes the approximation error $\boldsymbol{\epsilon}^2 = \mathbf{f}^T\mathbf{f} = \| \mathbf{r} - \mathbf{S}\boldsymbol{\alpha} \|_{\ell_2}^2$ and thus:

$$\hat{\mathbf{a}}(\mathbf{r}) = \arg\min_{\boldsymbol{\alpha}} \| \mathbf{r} - \mathbf{S}\boldsymbol{\alpha} \|_{\ell_2}^2 .$$

By setting $\frac{\partial \boldsymbol{\epsilon}^2}{\partial \boldsymbol{\alpha}} = 0$ we obtain

$$\hat{\mathbf{a}}(\mathbf{r}) = \left[\mathbf{S}^H\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{r},$$

[3] An example for this interpretation is to consider $\mathbf{r}$ a time-domain observed signal, $\mathbf{S}$ the inverse of the Fourier transform and $\mathbf{a}$ the Fourier-transformed true signal.

[4] Consecutive chunks of signal may be obtained with a sliding window.

[5] In a related interpretation, $\mathbf{r}$ is the observation, $\mathbf{S}$ is a signal matrix with consecutive chunks of the true signal along its lines, and $\mathbf{a}$ is a relationship vector.

and therefore, the sought estimator is given by $\mathbf{A} = \left[\mathbf{S}^H\mathbf{S}\right]^{-1}\mathbf{S}^H$. With this estimator, $\hat{\mathbf{a}}(\mathbf{r})$ represents the least-squares estimate for signal spaces where the scalar product is defined as $\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{y}^H\mathbf{x}$.

For spaces where the scalar product is defined as $\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{y}^H\mathbf{G}\mathbf{x}$, the estimate is

$$\hat{\mathbf{a}}(\mathbf{r}) = \left[\mathbf{S}^H\mathbf{G}\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{G}\mathbf{r}, \tag{2.21}$$

and is called the generalized least-squares estimate, the corresponding estimator being $\mathbf{A} = \left[\mathbf{S}^H\mathbf{G}\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{G}$.

Best Linear Unbiased Estimator. By the *Gauss-Markov theorem*, for the model in (2.20), under conditions that we introduce next, the least squares estimator represents the Best Linear Unbiased Estimator (BLUE).

If we choose $\mathbf{G} = \mathbf{R}_{nn}^{-1}$ with $\mathbf{R}_{nn} = E\{\mathbf{n}\mathbf{n}^H\}$, then (2.21) yields the unbiased estimate with the smallest variance (i.e., minimal diagonal elements for $\mathbf{R}_{ee}$):

$$\hat{\mathbf{a}}(\mathbf{r}) = \left[\mathbf{S}^H\mathbf{R}_{nn}^{-1}\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{R}_{nn}^{-1}\mathbf{r}.$$

For the BLUE, the estimator is computed as $\mathbf{A} = \left[\mathbf{S}^H\mathbf{R}_{nn}^{-1}\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{R}_{nn}^{-1}$. If the noise process is white with covariance matrix $\mathbf{R}_{nn} = \sigma^2\mathbf{I}$, then $\mathbf{A} = \left[\mathbf{S}^H\mathbf{S}\right]^{-1}\mathbf{S}^H$ and the standard least-squares estimator is the BLUE.

The Least Squares Estimator and the Gaussian Assumption. Within the context of least squares signal estimation, the Gaussian assumption is made with respect to the noise term $\mathbf{n}$. We will show next that when the noise term is white and Gaussian the least-squares estimator achieves the Cramer-Rao bound. We observe that when the components of the involved random variable are independent the least squares estimator yields the best possible result.

When the error process is white with zero mean and common variance over all elements such that $E\left\{\mathbf{n}\mathbf{n}^H\right\} = \sigma^2\mathbf{I}$, the covariance matrix $\text{cov}[\hat{\mathbf{a}}] = E\left\{(\hat{\mathbf{a}} - \mathbf{a})(\hat{\mathbf{a}} - \mathbf{a})^H\right\}$ of the least-squares estimate is:

$$\begin{aligned}
\text{cov}[\hat{\mathbf{a}}] &= E\left\{(\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H\mathbf{n}\mathbf{n}^H\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1}\right\} \\
&= (\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H E\left\{\mathbf{n}\mathbf{n}^H\right\}\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1} \\
&= \sigma^2(\mathbf{S}^H\mathbf{S})^{-1}\mathbf{S}^H\mathbf{S}(\mathbf{S}^H\mathbf{S})^{-1} \\
&= \sigma^2(\mathbf{S}^H\mathbf{S})^{-1} \\
&= \sigma^2\boldsymbol{\Phi}^{-1}.
\end{aligned}$$

If on top of this the error process $\mathbf{n} = \mathbf{r} - \mathbf{S}\mathbf{a}$ is also Gaussian, the Fisher information matrix $\mathbf{J}$

is computed as [92]:

$$\begin{aligned}\mathbf{J} &= \frac{1}{\sigma^4} E\left\{\mathbf{S}^H \mathbf{n}\mathbf{n}^H \mathbf{S}\right\} \\ &= \frac{1}{\sigma^4} \mathbf{S}^H E\left\{\mathbf{n}\mathbf{n}^H\right\} \mathbf{S} \\ &= \frac{1}{\sigma^2} \mathbf{S}^H \mathbf{S} \\ &= \frac{1}{\sigma^2} \boldsymbol{\Phi}\end{aligned}$$

It follows that the covariance matrix of the estimator is under these conditions equal to the inverse of the Fisher information matrix and thus the least-squares estimate $\hat{\mathbf{a}}$ satisfies the Cramer-Rao bound with equality, being thus the most efficient estimate.

The Linear Least-Squares Filter

The Least-Squares (LS) estimator can be used to compute the parameters of a filter. The purpose of this filter is to estimate the true random signal from a set of noisy observations. The estimation procedure is linear such that the estimate is computed as a weighted sum of noisy-signal values. The corresponding setup is shown in Figure 2.3.

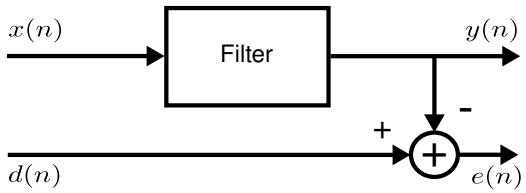


Figure 2.3: Problem setup of the linear least-squares filter.

By this method we have the possibility to compute the weights of the filter, without invoking assumptions on the statistics of the filter input. The method of least squares involves the use of time averages, hence the filter depends on the number of samples used in the computation.

For a transversal filter with weights $h(n)$, $n = 0, \ldots, p-1$, the estimation error [6] (i.e., residual) is

$$\begin{aligned}e(i) &= d(i) - y(i) \\ &= d(i) - \sum_{n=0}^{p-1} h^*(n) r(i-n)\end{aligned}$$

where $d(i)$ is the desired filter output and $y(i) = \sum_{n=0}^{p-1} h^*(n) r(i-n)$ is the realized output for the input $\mathbf{r}(i) = [r(i), r(i-1), \ldots, r(i-p+1)]^T$.

$\mathbf{h}$ is chosen such that the cost function

$$J(\mathbf{h}) = \sum_{i=i_1}^{i_2} |e(i)|^2$$

[6] The error process is usually assumed white with zero mean and variance σ^2.

is minimized. $i_{1,2}$ define the index limits at which the error minimization occurs, i.e., the training data. With $i_1 = p$ and $i_2 = N$ we have to minimize $J(\mathbf{h}) = \mathbf{e}^H\mathbf{e}$ with

$$\mathbf{e} = \mathbf{d} - \mathbf{S}\mathbf{h}$$

where the input data is arranged in matrix form as

$$\mathbf{S} = \begin{bmatrix} r(p) & r(p+1) & \cdots & r(N) \\ r(p-1) & r(p) & \cdots & r(N-1) \\ \vdots & \vdots & & \vdots \\ r(1) & r(2) & \cdots & r(N-p+1) \end{bmatrix}.$$

The least-squares estimator of the filter weights is then computed as:

$$\mathbf{h} = [\mathbf{S}^H\mathbf{S}]^{-1}\mathbf{S}^H\mathbf{d}.$$

The filter resulting from the minimization is termed a *linear least-squares* filter.

Minimum Mean Square Error Estimation

The correlation matrix of the estimation error of a linear estimate $\hat{\mathbf{a}}(\mathbf{r}) = \mathbf{A}\mathbf{r}$ is:

$$\mathbf{R}_{ee} = E\left\{\mathbf{e}(\mathbf{r})\mathbf{e}^H(\mathbf{r})\right\} = E\left\{\left[\hat{\mathbf{a}}(\mathbf{r}) - \mathbf{a}\right]\left[\hat{\mathbf{a}}(\mathbf{r}) - \mathbf{a}\right]^H\right\}.$$

For an unbiased estimate, the diagonal elements of $\mathbf{R}_{ee}$ represent the variances of each component of $\hat{\mathbf{a}}$. We search now for an estimate with minimal variance. For this purpose we minimize over $\mathbf{A}$ the elements on the diagonal of $\mathbf{R}_{ee}$ to obtain [139] (see also Appendix A)

$$\mathbf{A} = \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1} \tag{2.22}$$

with $\mathbf{R}_{ar} = E\{\mathbf{r}\mathbf{a}^H\} = \mathbf{R}_{ra}^H$ and $\mathbf{R}_{rr} = E\{\mathbf{r}\mathbf{r}^H\}$. Hence, the minimum mean square error (MMSE) estimate is

$$\hat{\mathbf{a}}(\mathbf{r}) = \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{r}.$$

For the model in (2.20), assuming $\mathbf{n}$ is not correlated to $\mathbf{a}$, we have that $\mathbf{R}_{ra} = \mathbf{R}_{ar}^H = \mathbf{R}_{aa}\mathbf{S}^H$ and $\mathbf{R}_{rr} = \mathbf{S}\mathbf{R}_{aa}\mathbf{S}^H + \mathbf{R}_{nn}$. Thus, we obtain

$$\mathbf{A} = \mathbf{R}_{aa}\mathbf{S}^H\left[\mathbf{S}\mathbf{R}_{aa}\mathbf{S}^H + \mathbf{R}_{nn}\right]^{-1},$$

which is equivalent to

$$\mathbf{A} = \left[\mathbf{R}_{aa}^{-1} + \mathbf{S}^H\mathbf{R}_{nn}^{-1}\mathbf{S}\right]^{-1}\mathbf{S}^H\mathbf{R}_{nn}^{-1}.$$

The correlation matrix of the estimation error is $\mathbf{R}_{ee} = \left[\mathbf{R}_{aa}^{-1} + \mathbf{S}^H\mathbf{R}_{nn}^{-1}\mathbf{S}\right]^{-1}$.

The connection between MMSE and BLUE and the link to the Gaussian Assumption. If the matrix $\mathbf{R}_{aa}$ is not known, the we set $\mathbf{R}_{aa}^{-1} = \mathbf{0}$ and it follows that

$$\mathbf{A} = \left[\mathbf{S}^H \mathbf{R}_{nn}^{-1} \mathbf{S}\right]^{-1} \mathbf{S}^H \mathbf{R}_{nn}^{-1}$$

which represents the BLUE estimator. The MMSE estimation method may yield an estimate that has a smaller variance than BLUE, but this estimate will be biased.

We have seen in the previous section, that under the Gaussian assumption (on the posterior of the estimate) the MMSE estimator is a MAP estimator. Ignoring $\mathbf{R}_{aa}$ is equivalent to switching from the MAP to the ML setup. Thus, we conclude that when ignoring $\mathbf{R}_{aa}$ under the Gaussian assumption, MMSE becomes BLUE and is a type of ML estimator. As discussed above, if the error process is white, the least-square estimator is the BLUE. Therefore when the error process is white and Gaussian, the least-square estimator is a ML type of estimator.

The Wiener Filter

As previously discussed, we consider that the true signal is related to the noisy observations in a linear manner and it can be thus estimated from those with the help of a linear model. The linear model tells us how to combine the noisy observations (up to and including the current one) for the purpose of computing an estimate of a true signal sample. Under these circumstances, with the Wiener filter we follow the purpose of estimating the form of a signal from noisy observations. Therefore, we make use of discrete linear filters and in particular we concentrate on FIR-filter structures. We proceed by means of MMSE in a setup depicted in Figure 2.3 and obtain the Wiener-Hopf equations, whose solution yields the Wiener filter.

The Wiener-Hopf Equations. We search for an FIR filter $h(n)$ such that its output $y(n) = h(n) * x(n) = \sum_{i=0}^{p-1} h(i)x(n-i)$, which represents the sought estimate, verifies

$$E\left\{|e(n)|^2\right\} = E\left\{|d(n) - y(n)|^2\right\} \to \min$$

when $x(n)$ is the noise-corrupted signal and $d(n)$ the desired output, i.e., the true signal. This cost function may be written as:

$$\begin{aligned} J &= E\left\{e(n)e^*(n)\right\} \\ &= E\left\{|d(n)|^2\right\} - \sum_{i=0}^{p-1} h^*(i) E\left\{x(n-i)d^*(n)\right\} - \\ &\quad - \sum_{i=0}^{p-1} h(i) E\left\{x^*(n-i)d(n)\right\} + \sum_{k=0}^{p-1}\sum_{i=0}^{p-1} h^*(k)h(i) E\left\{x(n-k)x^*(n-i)\right\} \\ &= \sigma_d^2 - \sum_{i=0}^{p-1} h^*(i) r_{xd}(-i) - \sum_{i=0}^{p-1} h(i) r_{xd}^*(-i) + \sum_{k=0}^{p-1}\sum_{i=0}^{p-1} h^*(k)h(i) r_{xx}(i-k), \end{aligned} \tag{2.23}$$

with

$$r_{xx}(m) = E\left\{x^*(n)x(n+m)\right\},$$

$$r_{xd}(m) = E\left\{x^*(n)d(n+m)\right\}.$$

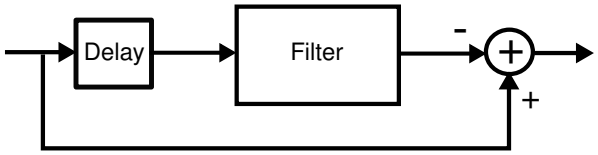


Figure 2.4: A one-step linear predictor. The Filter corresponds to the linear operator relating the predicted value to the available observations.

Assuming a stationary process, J is a second-order function of $h(n)$ and by setting to zero its derivative with respect to the filter weights, we obtain the *Wiener-Hopf Equations* (see also Appendix B):

$$\sum_{i=0}^{p-1} h(i) r_{xx}(j-i) = r_{xd}(j), \quad j = 0, 1, \ldots, p-1. \tag{2.24}$$

The discrete Wiener Filter. The discrete Wiener filter is obtained by solving the Wiener-Hopf Equations (2.24). In matrix form (2.23) becomes:

$$J(\mathbf{h}) = \sigma_d^2 - \mathbf{h}^H \mathbf{r}_{xd} - \mathbf{r}_{xd}^H \mathbf{h} + \mathbf{h}^H \mathbf{R} \mathbf{h}.$$

From $\nabla J(\mathbf{h}) = 0$ we obtain

$$\mathbf{R}_{xx} \mathbf{h} = \mathbf{r}_{xd}, \tag{2.25}$$

with

$$\mathbf{h} = [h(0), h(1), \ldots, h(p-1)]^T,$$
$$\mathbf{r}_{xd} = [r_{xd}(0), r_{xd}(1), \ldots, r_{xd}(p-1)]^T,$$

and

$$\mathbf{R}_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \cdots & r_{xx}(-p+1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(-p+2) \\ \vdots & \vdots & & \vdots \\ r_{xx}(p-1) & r_{xx}(p-2) & \cdots & r_{xx}(0) \end{bmatrix}.$$

The Wiener Filter is then computed as:

$$\mathbf{h} = \mathbf{R}_{xx}^{-1} \mathbf{r}_{xd}. \tag{2.26}$$

In the case of the Wiener filter one needs to know the desired output for the corresponding input only in the training phase, when $\mathbf{h}$ is computed.

One-step Linear Prediction

The linear predictor (see Figure 2.4) predicts one step into the future, i.e., it predicts the next value of the input signal, using a linear combination of the current and past values.

A predicted signal value $\hat{x}(n)$ is computed as a linear combination[7] of p signal values $x(n-p), \ldots, x(n-1)$ as:

$$\hat{x}(n) = -\sum_{i=1}^{p} a(i)\, x(n-i). \tag{2.27}$$

[7]The reason for the negative sign in the formula will become clear later.

- The numbers $a(1), a(2), \ldots, a(p)$ are the (fixed) coefficients of the above mentioned linear combination.
- The sequence
$$h(n-1) = -a(n)$$
can be seen as the impulse response of a length-p FIR filter.
- With $y(n) = \hat{x}(n)$ we have the following equivalent to (2.27):
$$y(n) = \sum_{i=0}^{p-1} h(i)x(n-i).$$

The one-step linear predictor enjoys a clear link to the Wiener filter and through the Yule-Walker equations to autoregressive processes as well. Both these relationships are discussed next, as well as the relationship between linear signal prediction and the Gaussian assumption.

Coefficient optimization. The "optimal" coefficients $a(n)$ depend on the statistics of the random process $x(n)$. In the following, we will describe how these coefficients can be found.

The prediction error can be written as

$$\begin{aligned} e(n) &= x(n) - \hat{x}(n) \\ &= x(n) + \sum_{i=1}^{p} a(i)\, x(n-i). \end{aligned} \tag{2.28}$$

The aim is now to minimize the mean squared error

$$F = E\left\{|e(n)|^2\right\}$$

under the assumption of a stationary input process $x(n)$. We have

$$\begin{aligned} F &= E\left\{x^2(n)\right\} + E\left\{2\sum_{i=1}^{p} a(i)\, x(n-i)x(n)\right\} \\ &\quad + E\left\{\sum_{i=1}^{p} a(i)\, x(n-i) \sum_{j=1}^{p} a(j)\, x(n-j)\right\}. \end{aligned}$$

Only $x(n)$ is random, so we can write

$$\begin{aligned} F &= E\left\{x^2(n)\right\} + 2\sum_{i=1}^{p} a(i) E\left\{x(n-i)x(n)\right\} \\ &\quad + \sum_{i=1}^{p}\sum_{j=1}^{p} a(i)\, a(j)\, E\left\{x(n-i)x(n-j)\right\}. \end{aligned}$$

The expected values turn out to be values of the autocorrelation sequence of the input process $x(n)$. For a real-valued stationary process, this is defined as

$$r_{xx}(m) = E\left\{x(n)x(n+m)\right\}.$$

Because $r_{xx}(m) = r_{xx}(-m)$ for a real-valued stationary process, we have

$$E\left\{x^2(n)\right\} = r_{xx}(0),$$
$$E\left\{x(n-i)x(n-j)\right\} = r_{xx}(i-j),$$
$$E\left\{x(n-i)x(n)\right\} = r_{xx}(i).$$

Thus, we have that

$$F = r_{xx}(0) + 2\sum_{i=1}^{p} a(i) r_{xx}(i) + \sum_{i=1}^{p}\sum_{j=1}^{p} a(i)\, a(j)\, r_{xx}(j-i).$$

Minimizing the error with respect to the unknown coefficients $a(1), \ldots, a(p)$ yields the equations

$$-\sum_{i=1}^{p} a(i)\, r_{xx}(j-i) = r_{xx}(j), \qquad j = 1, 2, \ldots, p,$$

which are known as the *normal equations of linear prediction*. In matrix notation they are

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \ldots & r_{xx}(-p+1) \\ r_{xx}(1) & r_{xx}(0) & \ldots & r_{xx}(-p+2) \\ \vdots & \vdots & & \vdots \\ r_{xx}(p-1) & r_{xx}(p-2) & \ldots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ \vdots \\ a(p) \end{bmatrix} = - \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(p) \end{bmatrix} \tag{2.29}$$

In short we get

$$\mathbf{R}_{xx}\mathbf{a} = -\mathbf{r}_{xx}(1) \quad \text{with} \quad \mathbf{a}^T = [a(1), \ldots, a(p)]\,. \tag{2.30}$$

By comparing equation (2.25) to equation (2.30), we can observe that the one step linear predictor is nothing more than the Wiener filter in predictor setup, i.e., when the desired response is just the signal itself, more precisely the next observation from the analyzed random signal.

Yule-Walker equations. An autoregressive (AR) model of a random signal is obtained by applying white noise[8] $w(n)$ to the input of an all-pole LTI system with coefficients $\mathbf{a} = [a(1), \ldots, a(p)]^T$, such that the corresponding difference equation is

$$x(n) = w(n) - \sum_{i=1}^{p} a(i) x(n-i).$$

The Yule-Walker equations return the parameters of an AR model, given the autocorrelation of the modeled process and the power of the corresponding white noise:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \ldots & r_{xx}(-p) \\ r_{xx}(1) & r_{xx}(0) & \ldots & r_{xx}(-p+1) \\ \vdots & \vdots & & \vdots \\ r_{xx}(p) & r_{xx}(p-1) & \ldots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} 1 \\ a(1) \\ \vdots \\ a(p) \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.31}$$

As it may be seen, the Yule-Walker equations include the normal equations of linear prediction (2.29) and an additional relation describing the AR model. It may thus be concluded that a linear predictor is best suited to analyze a signal that represents an AR process.

[8] White noise is a wide-sense stationary and uncorrelated stochastic signal. For a purely random signal, the set distributions are independent. Thus, white Gaussian noise is a strict-sense stationary purely random signal.

Sample prediction and the Gaussian assumption. We assume that all samples available for the prediction $x(n-i),\ i=1,\ldots,p$ are Gaussian with zero mean and covariance matrix $\mathbf{R}_p$ and we would like to find $\hat{x}(n)$ the best estimate for the next sample $x(n)$. As discussed before, this estimate can be found, with a quadratic cost function as the mean of the posterior density $p(x(n)|\mathbf{x}_p)$, where $\mathbf{x}_p = [x(n-1),\ldots,x(n-p)]^T$. It can be shown [174] that this posterior is given by

$$p(x(n)|\mathbf{x}_p) = \frac{1}{\sqrt{2\pi}} \exp\left[\frac{-\left(x(n)-\mathbf{g}^T\mathbf{x}_p\right)^2}{2\sigma^2}\right]$$

with $\sigma = \sqrt{\frac{|\mathbf{R}_{p+1}|}{|\mathbf{R}_p|}}$, $\mathbf{g} = \mathbf{R}_p^{-1}\boldsymbol{\varphi}$ and $\boldsymbol{\varphi} = E\{x(n)\mathbf{x}_p\}$. With this Gaussian posterior, the sought estimate[9] is given by $\hat{x}(n) = \mathbf{g}^T\mathbf{x}_p$, i.e., a linear combination of the previous samples, just like the one-step linear predictor. Thus, when the samples are jointly Gaussian, the one-step linear predictor represents the best predictor. However, it is not guaranteed that a linear combination of previous samples represents the best predictor under different circumstances.

2.1.3 Expectation maximization

Expectation Maximization (EM) [86] is an iterative method for computing a ML estimate $\hat{\mathbf{a}}$ of a set of parameters $\mathbf{a}$. The parameters are related to some complete data. The complete data is usually a set of realizations of the complete random variable $\mathbf{c}$. The estimate is computed with only some observations of the incomplete data that is generated by the incomplete random variable $\mathbf{r}$. The incomplete/observed data has to have a known relationship to the complete data. For example, assuming $\mathbf{c}$ has dimension N, $\mathbf{r}$ can be the vector containing only the first $k, k \in \{1,\ldots,N-1\}$ components of $\mathbf{c}$, or $\mathbf{r}$ may include a number k of functions of the other $N-k$ entries of $\mathbf{c}$. In general, we can assume that $\mathbf{c} = [\mathbf{r}, \mathbf{z}]$, where $\mathbf{z}$ is some hidden data. To compute our estimate we have at our disposal O observations not necessarily i.i.d. extracted from the incomplete data $R = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n\}$.

Next, the general EM algorithm is discussed and then an example is given of EM-based estimation for missing-data problems under the i.i.d. assumption, in which context it is shown how to compute the parameters of a Gaussian mixture model (GMM). GMMs are widely used models in all types of applications ranging from audio [195] to background modeling [120]. The are often employed, e.g., as parametric approximations to more complex distributions. We will conclude our discussion on the EM algorithm by analyzing some shortcomings, as well as emphasizing the link between this algorithm and some stochastic signal analysis methods that are used later on.

The general form of the EM algorithm

The EM algorithm computes a ML estimate. As discussed before, for this purpose it suffices to solve the corresponding log-likelihood equation (see equation (2.5)). As we have only the incomplete data at our disposal, we set about to formulate the log-likelihood of the incomplete data under the sought parameters $\mathbf{a}$ with the purpose of obtaining an expression where the

[9] As pointed out before, under such circumstances, the estimate represents the optimum by a number of cost functions and not only by the quadratic cost function.

optimum over $\mathbf{a}$ can be readily computed. We start from the logarithm of the density of the observed data, which we compute with the help of some function $q(\mathbf{z})$ that is member of a family of pdf approximations Q:

$$\begin{aligned}\log p(\mathbf{r}) = &\log \int p(\mathbf{r},\mathbf{z})d\mathbf{z} \\ = &\log \int q(\mathbf{z})\frac{p(\mathbf{r},\mathbf{z})}{q(\mathbf{z})}d\mathbf{z}.\end{aligned}$$

Using Jensen's inequality, while taking into consideration, that $\log$ is a concave function and making the dependency on the parameter vector $\mathbf{a}$ explicit, we obtain a lower bound for $\log p(\mathbf{r}|\mathbf{a})$ as:

$$\log p(\mathbf{r}|\mathbf{a}) \geq \int q(\mathbf{z}) \log \frac{p(\mathbf{r},\mathbf{z}|\mathbf{a})}{q(\mathbf{z})}d\mathbf{z}. \tag{2.32}$$

A careful look at the right-hand side of this inequation reveals it to be a functional on q and $\mathbf{a}$ that can be expanded as:

$$\begin{aligned}\mathcal{L}(q,\mathbf{a}) = &\int q(\mathbf{z}) \log \frac{p(\mathbf{r},\mathbf{z}|\mathbf{a})}{q(\mathbf{z})}d\mathbf{z} \\ = &\int q(\mathbf{z}) \log p(\mathbf{r},\mathbf{z}|\mathbf{a}) - q(\mathbf{z}) \log q(\mathbf{z})d\mathbf{z}.\end{aligned}$$

So our problem is now recast as finding q and $\mathbf{a}$ for which $\mathcal{L}(q,\mathbf{a})$ is maximal that will lead us to the $\mathbf{a}$ for which the log-likelihood $\log p(\mathbf{r}|\mathbf{a})$ is maximal. Then, as already implied in its name, the EM algorithm is a coordinate ascent in $\mathcal{L}$ including two main steps that repeat iteratively until the parameter vector does not change significantly anymore:

- **E-step:** Given a parameter $\hat{\mathbf{a}}_m$ from the previous iteration estimate:

$$\hat{q}_{m+1} = \arg\max_{q\in Q} \mathcal{L}(q,\hat{\mathbf{a}}_m).$$

- **M-step:** Given the conditional q_{m+1} from the previous step compute:

$$\hat{\mathbf{a}}_{m+1} = \arg\max_{\mathbf{a}\in A} \mathcal{L}(q_{m+1},\mathbf{a}).$$

To solve the E-step, we rewrite $\mathcal{L}$ as

$$\begin{aligned}\int q(\mathbf{z}) \log \frac{p(\mathbf{r},\mathbf{z})}{q(\mathbf{z})}d\mathbf{z} &= \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{r})p(\mathbf{r})}{q(\mathbf{z})}d\mathbf{z} \\ &= \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{r})}{q(\mathbf{z})}d\mathbf{z} + \int q(\mathbf{z}) \log p(\mathbf{r})d\mathbf{z} \\ &= \log p(\mathbf{r}) \int q(\mathbf{z})d\mathbf{z} + \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{r})}{q(\mathbf{z})}d\mathbf{z} \\ &= \log p(\mathbf{r}) - \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{r})}d\mathbf{z},\end{aligned} \tag{2.33}$$

where we have dropped the dependency on $\mathbf{a}$ for the sake of clarity. Looking at the right side of the equation (2.33), we recognize the second therm of the difference as the Kullback-Leibler

divergence between $q(\mathbf{z})$ and $p(\mathbf{z}|\mathbf{r})$. This is zero only when the two are equal, being otherwise positive. It appears thus clear that that solution of the E-step is $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{r})$, in which case the functional $\mathcal{L}(p(\mathbf{z}|\mathbf{r}), \mathbf{a})$ evaluates to $\log p(\mathbf{r}|\mathbf{a})$, the log-likelihood of the incomplete data, and the inequation (2.32) holds with equality.

In the M-step we need then to maximize over $\mathbf{a}$

$$\mathcal{L}(p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m), \mathbf{a}) = \int p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m) \log p(\mathbf{r}, \mathbf{z}|\mathbf{a}) - p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m) \log p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m) d\mathbf{z},$$

where we made the dependency on $\hat{\mathbf{a}}_m$ clear. This is equivalent to maximizing

$$L_c(\mathbf{a}) = \int p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m) \log p(\mathbf{r}, \mathbf{z}|\mathbf{a}) d\mathbf{z}$$

over $\mathbf{a}$ as the second term of the difference does not depend on $\mathbf{a}$. It is interesting to note that $L_c(\mathbf{a})$, called the *complete log likelihood*, is an estimate of the log-likelihood of the complete data computed as the expectation of the log-likelihood over the conditional of the hidden data on the observed data, as $p(\mathbf{c}|\mathbf{r}) = p(\mathbf{z}, \mathbf{r}|\mathbf{r}) = p(\mathbf{z}|\mathbf{r})$.

The two steps of the EM algorithm become then:

- **E-step:** Given an estimate $\hat{\mathbf{a}}_m$ from the previous iteration, compute $L_c(\mathbf{a})$.
- **M-step:** Compute $\hat{\mathbf{a}}_{m+1}$ as:

$$\hat{\mathbf{a}}_{m+1} = \arg\max_{\mathbf{a} \in A} L_c(\mathbf{a}).$$

It is advantageous to divide and complete the above set of steps, obtaining thus a five-steps algorithm:

1. Start with an initial (i.e., $m = 0$) estimate $\hat{\mathbf{a}}_m$ of the sought parameter vector.
2. Given the observed incomplete data and the estimate from the step above, build the conditional $p(\mathbf{z}|\mathbf{r}, \hat{\mathbf{a}}_m)$.
3. Use the conditional from the previous step to build the complete log likelihood $L_c(\mathbf{a})$.
4. Compute $\hat{\mathbf{a}}_{m+1}$ as the argument that maximizes the $L_c(\mathbf{a})$.
5. Increase m and go to step two if the estimate changes significantly, otherwise stop.

The EM algorithm is not guaranteed to find the global optimum, being thus susceptible to get stuck into local optima. The standard solution to this problem is to start from several initializations and pick the final result that has the largest likelihood.

EM-based estimation for missing data problems under the i.i.d. assumption

In many applications, like for example when learning the parameters of a GMM, from unlabeled data we have a missing data problem with i.i.d. observations. We speak of missing data problems, when there is no information available to link the complete and the incomplete data. Then, the complete data is a set of independent identically distributed samples $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n\}$,

each one consisting of an observed $\mathbf{r}_i$, and an unobserved $\mathbf{z}_i, i = 1, \ldots, n$, with no functional relationship between the two.

The first task we face is to find a mathematical expression for the conditional of the hidden data on the observed data under a give parameter vector. Here we can make use of our prior knowledge on the problem setup. We then use this conditional to express the complete log-likelihood as a function of the parameters given the available sample from the observed data. Finally we find the sought parameters at the location of the maximum of the complete log-likelihood. This procedure is illustrated next for finding the parameters of a GMM from unlabeled data.

Estimate the parameters of a GMM. Our purpose is to estimate the parameter vector $\mathbf{a} = \left[(w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right],\ j = 1, \ldots, k$, from n vectors $\mathbf{r}_1, \ldots, \mathbf{r}_n$, i.i.d. generated from a mixture of k Gaussians with density

$$p(\mathbf{r}_i|\mathbf{a}) = \sum_{j=1}^{k} w_j \gamma_j(\mathbf{r}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

with $\gamma_j(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, and where $w_j \geq 0$ and $\sum_{j=1}^{k} w_j = 1$. Our complete data is given by $\mathbf{c}_i = [\mathbf{r}_i, z_i],\ i = 1, \ldots, n$, where the hidden data z_i is the mixture-component label for each observation. Therefore in this case the hidden data is a scalar random variable whose sample space (that includes all outcomes/elementary events) is $\Omega_{z_i} = \{1, 2, \ldots, k\}$.

We now define the conditional of the hidden data on the observed data under a give parameter vector. This is the probability that at the m-th iteration, the i-th sample was generated by the j-th mixture component, computed as:

$$P(z_i = j|\mathbf{r}_i, \hat{\mathbf{a}}_m) = \frac{w_j^{(m)} \gamma(\mathbf{r}_i|\boldsymbol{\mu}_j^{(m)}, \boldsymbol{\Sigma}_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \gamma(\mathbf{r}_i|\boldsymbol{\mu}_l^{(m)}, \boldsymbol{\Sigma}_l^{(m)})}. \tag{2.34}$$

Given one instance $\mathbf{r}_i$ from the sample of observed data and the parameter vector from iteration m, the complete log-likelihood is computed as:

$$\begin{aligned} L_c^i(\mathbf{a}) = & \sum_{j=1}^{k} P(z_i = j|\mathbf{r}_i, \hat{\mathbf{a}}_m) \log\left[p(\mathbf{r}_i, z_i|\mathbf{a})\right] \\ = & \sum_{j=1}^{k} \alpha_{ij}^{(m)} \log\left[w_j \gamma(\mathbf{r}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right] \\ = & \sum_{j=1}^{k} \alpha_{ij}^{(m)} \left[\log(w_j) - \frac{1}{2}\log|\boldsymbol{\Sigma}_j| - \frac{1}{2}(\mathbf{r}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_j)\right] + K \end{aligned}$$

where for a compact notation we define $\alpha_{ij}^{(m)} = P(z_i = j|\mathbf{r}_i, \hat{\mathbf{a}}_m)$, with $\sum_{j=1}^{k} \alpha_{ij}^{(m)} = 1$. K is a constant, independent of $\mathbf{a}$, and can therefore be ignored in our optimization setup. With $\widehat{L_c^i}$ obtained from L_c^i by ignoring K we can complete the E-step as:

$$\begin{aligned} L_c(\mathbf{a}) = & \sum_{i=1}^{n} \widehat{L_c^i}(\mathbf{a}) \\ = & \sum_{i=1}^{n} \sum_{j=1}^{k} \alpha_{ij}^{(m)} \left[\log(w_j) - \frac{1}{2}\log|\boldsymbol{\Sigma}_j| - \frac{1}{2}(\mathbf{r}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_j)\right]. \end{aligned} \tag{2.35}$$

With the complete log-likelihood given in equation (2.35), the M-step is then:

$$\begin{aligned} \text{find } \mathbf{a} \text{ to maximize } & \quad L_c(\mathbf{a}), \\ \text{subject to } & \quad \sum_{j=1}^{k} w_j = 1,\ w_j > 0,\ \boldsymbol{\Sigma}_j \text{ is positive definite},\ j = 1 \ldots, k. \end{aligned} \tag{2.36}$$

Solving the M-step (2.36), leads to $\hat{\mathbf{a}}_{m+1} = \left[(w_j^{(m+1)}, \boldsymbol{\mu}_j^{(m+1)}, \boldsymbol{\Sigma}_j^{(m+1)})\right],\ j = 1, \ldots, k$ where

$$w_j^{(m+1)} = \frac{n_j^{(m)}}{\sum_{l=1}^{k} n_l^{(m)}}, \text{with } n_j^{(m)} = \sum_{i=1}^{n} \alpha_{ij}^{(m)}, \tag{2.37}$$

$$\boldsymbol{\mu}_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \alpha_{ij}^{(m)} \mathbf{r}_i$$

and

$$\boldsymbol{\Sigma}_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \alpha_{ij}^{(m)} (\mathbf{r}_i - \boldsymbol{\mu}_j^{(m+1)})(\mathbf{r}_i - \boldsymbol{\mu}_j^{(m+1)})^T.$$

As it can be seen, in equation (2.37), the EM algorithm establishes a relationship between the missing data given in this case by the unknown label of each instance of the observed-data sample and one of the sought parameters. Furthermore, with the help of equation (2.34) we can label the available data and therefore conduct unsupervised classification in a MAP approach with the prior given by w_j and the likelihood by $\gamma_j(\mathbf{x})$.

Although the EM algorithm can be randomly initialized, practically it is better to use the k-means clustering for initialization.

Concluding remarks on the EM algorithm

The EM algorithm is a powerful tool widely used in the statistical analysis of data. It has generated a multitude of other algorithms that follow similar principles to iteratively search for parameters and/or hidden data.

Depending on the application, the algorithm has also some drawbacks, like for example, the fact that the number k of clusters has to be set a-priori, when using the EM for unsupervised classification. In this case, to find the optimal solution, we need to use additional methods like Minimum Message Length or Bayes Information Criterion [66]. There are extension of the EM algorithm that incorporate such considerations directly and find the optimal k alone [70].

Extensions of the EM algorithm for various setups are usually referred to as *generalized EM*. The Baum-Welch algorithm, that is used to train a Hidden Markov Model (HMM) is such an example. A generalized EM is used also to train a Maximum Entropy Markov Model (MEMM) and EM-inspired algorithms are used to conduct training in CRFs as well.

The EM algorithm represents the application of a broader class of algorithms known as *variational methods* [107] to the problem of parameter estimation. Variational methods are particularly helpful for the practical deployment of graphical models as discussed in the next chapter.

2.2 Graphs in the probability theory

Probabilistic inference and learning can be described by both algebraic manipulations and by graphical models [20]. Probabilistic graphical models provide a simple way to visualize the structure of a probabilistic model. They substitute complex computations required to do inference and learning for graphical manipulations and may thus serve as a backbone for efficiently computing marginal and conditional probabilities. They also give insight into the properties of the model, in particular conditional independence properties.

General inference. *Inference* in a statistical setup is related to the computation of joint and conditional probabilities over various subsets of a set of random variables that completely describe the modeled reality (i.e. a probabilistic model). These probabilities are typically used to investigate relationships among the random variables. Two major tools in statistical inference are the *sum rule* and the *product rule*.

For a set of N discrete random variables $\{x_1, \ldots, x_N\}$, the sum rule is used to compute the probability $p(x_1, \ldots, x_{N-S})$ of a subset of random variables by marginalizing $p(x_1, \ldots, x_N)$ over the complementary subset of random variables $\{x_{N-S+1}, \ldots, x_N\}$:

$$p(x_1, \ldots, x_{N-S}) = \sum_{x_{N-S+1}} \cdots \sum_{x_N} p(x_1, \ldots, x_N). \tag{2.38}$$

The product rule relates a joint probability to a conditional and another joint probability. It is used to compute the conditional of a subset of random variables on another subset of random variables as a fraction of two joint probabilities:

$$p(x_{n_1}, \ldots, x_{n_K}, |x_{m_1} \ldots, x_{m_L}) = \frac{p(x_{n_1}, \ldots, x_{n_K}, x_{m_1} \ldots, x_{m_L})}{p(x_{m_1} \ldots, x_{m_L})}.$$

Efficient inference. Marginals can be computed efficiently by means of the *sum-product algorithm*, which makes use of the distributivity property of multiplication with respect to addition. An efficient algorithm implies less computations than the number needed to evaluate the sum in Equation 2.38. The sum-product algorithm uses a message-passing formalism.

Similar considerations lead to the efficient computation of other measures of interest, like the most probable realization $\mathbf{x}^{max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$, which can be achieved by the *max-sum algorithm*.

These methods that are typically introduced in the context of special types of graphical models (i.e., trees), can be extended to offer efficient solutions for exact inference in general graphical models in the form of the junction-tree algorithm that is discussed briefly in Section 2.2.2. However, in practice exact inference in graphical models of arbitrary topology even if conducted efficiently is often not feasible due to the share size of the model. Yet another issue with exact inference is the mathematical expression of the involved distributions, which may become intractable[10]. In such cases approximation methods are needed [107]. These approximation methods may target the inference procedure and/or the involved distributions. There are three main types of such methods: (i) relaxation methods that are adaptations of fast exact inference methods where some conditions needed for successful exact inference are ignored, like the *loopy belief propagation* that represents the application of the sum-product algorithm

[10]This may happen for example when the underlaying integrals can not be computed.

for trees (i.e., belief propagation) to graphs with loops, (ii) variational methods that are built around an optimization procedure like the one discussed in equation (2.33) and (iii) sampling or *Monte Carlo methods* that are based on sampling from distributions.

Conditional independence. A random variable (or set of random variables) a is conditionally independent of b given c if

$$p(a, b|c) = p(a|c)p(b|c) \tag{2.39}$$

or equivalently $a \perp\!\!\!\perp b|c$ if:

$$p(a|b, c) = p(a|c). \tag{2.40}$$

In this context, a and b are not independent, as they are related over c. They can be however looked upon as unrelated proofs for c. For example, measuring rainfall in a weather station is conditionally independent of how wet you get when you go outside given the current rainy weather.

A graph captures the way in which the joint distribution of several random variables can be decomposed into a product of factors – each depending only on a subset of the variables – based on their independence properties. Each node represents a random variable (or a group of random variables). Each arc expresses the probabilistic relationship between these variables. Depending on whether the arcs are directed or undirected, there are two types of graphical models: (i) *Directed graphical models*, or Bayes networks that are discussed in Section 2.2.1 with an accent on Dynamic Bayes Networks, and (ii) *Undirected graphical models*, or Markov random fields that are discussed in Section 2.2.2.

2.2.1 Bayes networks and dynamic Bayes networks

For *Bayes networks* , a directed arc signifies conditional dependence. A node A is parent to a node B if there is a directed arc from A to B. The descendants of a node are its children, children's children and so on. A directed path from A to B is a sequence of parent-child nodes starting at A and ending at B. An undirected path from A to B is a sequence of nodes such that each node in the sequence is a parent or child to the next one. Inference in Bayes networks is related to the search for conditional independences among variables of the graph.

Conditional independence. In a Bayes network each node is conditionally independent from its non-descendants given its parents and the absence of arcs implies conditional independence. More subtle conditional independence relationships may be investigated with the help of D-separation. *D-separation* is a graphical test for conditional independence. Two nodes A and B are conditionally independent given C if C d-separates them. A set of nodes S_D *d-separates* two disjoint sets of nodes S_1 and S_2, if every undirected path from S_1 to S_2 is blocked. A path is blocked if there is a node X such that either one of the following properties holds:

- X has converging arrows and neither it nor its descendants are in S_D.
- X does not have converging arrows and is in S_D.

In general we can say that a path is blocked when there is node on the path that belongs to the conditioning set, except for the case when this node has converging arrows. This is related to the phenomena of *"explaining away"*, where observing a child node (i.e., a node with converging

arrows) does not imply conditional independence of the parents. Indeed, assume that admission to an university depends on passing any one of two exams. Now given that admission has been successful and the second exam was passed we compute a probability for having passed the first exam as well that is smaller by comparison to the probability of having passed the first exam given that admission was successful and without considering the result of the second one. Thus equation (2.40) does not hold and the result on the first exam is not independent from the result of the second exam given that you know if the admission was successful or not. Intuitively, we can say, that knowing that admission was successful and the second exam was passed makes the result on the first exam unimportant so it may as well have been flunked, it does not matter, as the result on the second exam has explained the admission away.

Dynamic Bayes Networks. A Bayes network applied to a stochastic processes is called a *Dynamic Bayes network* (DBN). In this case, directed arcs point forward in time. We are thus considering only directed acyclic graphs, where there are no directed loops. DBNs usually work in a recursive manner, the model being extended at each time index, when a measurement becomes available. This has thus to do with the fact that the size (and thus final shape) of the model is not known beforehand. Conversely their architecture is well suited for such cases. As an example, consider a sequence of time-consecutive observations $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ constituting a first order Markov model. Their joint probability can be then factorized like:

$$p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T) = p(\mathbf{y}_1)p(\mathbf{y}_2|\mathbf{y}_1) \cdots p(\mathbf{y}_T|\mathbf{y}_{T-1}). \tag{2.41}$$

The DBN corresponding to this case is shown in Figure 2.5.

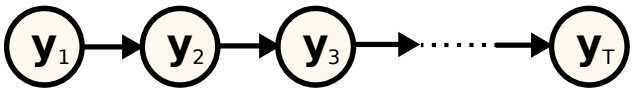


Figure 2.5: A Dynamic Bayes network

The architecture of a DBN - similar to the case of a Bayes network - must be adapted to the practical problem it models. There are however some simple yet powerful DBNs often encountered in practice on which we will focus next. Let us assume now, that observations are dependent on some hidden variables called *states*, and the states constitute a first-order Markov chain. We obtain thus a new type of stochastic model with increased modeling capacity. This may be used, among others, for purposes like estimating a "state signal" from an "observation signal". The hidden variables live in the state space that can be either discrete or continuous. A DBNs for a discrete state space case is shown in Figure 2.6 and a DBN for a continuous state space Figure 2.8. Each of these two cases will be analyzed in more detail, but first we will discuss about how to conduct inference in these simple state-observation DBNs. This represents the foundation upon which inference in more complicated DBNs lays. Furthermore, even this relatively simple DBNs are of huge practical importance, which serves to underline the versatility of the graphical models framework.

Inference in state-observation DBNs

As pointed out previously, inference implies the computation of various joint and conditional probabilities. For the case of state-observation DBNs, there are several quantities of interest.

A task often encountered in practice is to find out the sequence of states corresponding to an available sequence of observations. In a maximum likelihood approach, we seek to maximize the joint probability of observations and states over the states:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{y}_1, \ldots, \mathbf{y}_T) = p(\mathbf{x}_1)p(\mathbf{y}_1|\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{y}_t|\mathbf{x}_t).$$

For a given sequence of observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ we are therefore interested in the conditional $p(\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}|\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}) = p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$ whose maximum argument (usually computed by means of the expectation) returns the sought state sequence $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. Our model is described by $p(\mathbf{x}_1), p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$, $\forall t > 1$. Due to the sequential nature of the available data, rather than computing the maximum likelihood estimate of the state sequence for each new available observation from scratch, it is more advantageous to update an existing estimate in light of the newly acquired data [64]. The sought conditional may be computed as:

$$\begin{aligned} p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) =& \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}{p(\mathbf{y}_{1:t})} \\ =& \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}{\int p(\mathbf{y}_{1:t}, \mathbf{x}_{1:t}) d\mathbf{x}_{1:t}} \\ =& \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t}) d\mathbf{x}_{1:t}}. \end{aligned} \tag{2.42}$$

A recursive formula is then obtained as

$$\begin{aligned} p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) =& \frac{p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t})} \\ =& \frac{p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_{1:t})} \\ =& \frac{p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{1:t-2}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})} \\ =& \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})} \\ =& p(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1}) \frac{p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{y}_t|\mathbf{x}_t)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}, \end{aligned} \tag{2.43}$$

considering that each node is conditionally independent from its non-descendants given its parents and thus: (i) the current observation is independent from both previous states and previous observations, given the current state; and (ii) given the previous state, the current state is independent of the previous observations and any other previous states.

Yet another quantity of practical interest in this context is the marginal $p(\mathbf{x}_t|\mathbf{y}_{1:t}), \forall t$. With the help of the Bayes formula, starting from $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$, we can devise a recursion for this marginal directly, without having to go over $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$. The update step is then

$$p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})}, \tag{2.44}$$

considering that given the current state, the current and previous observations are independent, and we may thus write $p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = p(\mathbf{y}_t|\mathbf{x}_t)$.

The probability distribution of the next state given observations (predict step) is

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = & \int p(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) d\mathbf{x}_{t-1} \\ = & \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) d\mathbf{x}_{t-1}, \end{aligned} \tag{2.45}$$

considering that given the previous state, the current state and the previous observations are independent, and we may write $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_1 \ldots, \mathbf{y}_{t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$.

The probability of the current observation given the previous ones (evidence) is

$$\begin{aligned} p(\mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = & \int p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) d\mathbf{x}_t \\ = & \int p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) d\mathbf{x}_t \end{aligned} \tag{2.46}$$

again with $p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = p(\mathbf{y}_t|\mathbf{x}_t)$.

Finally, we are often interested in computing $p(\mathbf{x}_\tau|\mathbf{y}_1, \ldots, \mathbf{y}_\tau, \mathbf{y}_{\tau+1} \ldots \mathbf{y}_t)$ the probability of a past state given observations starting from before that past state and up to the present (smoothing). Using the Bayes rule and the assumed conditional independence relationships, this can be computed as:

$$\begin{aligned} p(\mathbf{x}_\tau|\mathbf{y}_1, \ldots, \mathbf{y}_\tau, \mathbf{y}_{\tau+1} \ldots \mathbf{y}_t) = & \frac{p(\mathbf{x}_\tau, \mathbf{y}_{1:\tau}, \mathbf{y}_{\tau+1:t})}{p(\mathbf{y}_{1:\tau}, \mathbf{y}_{\tau+1:t})} \\ = & \frac{p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau, \mathbf{y}_{1:\tau}) p(\mathbf{x}_\tau, \mathbf{y}_{1:\tau})}{p(\mathbf{y}_{1:\tau}, \mathbf{y}_{\tau+1:t})} \\ = & \frac{p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau, \mathbf{y}_{1:\tau}) p(\mathbf{x}_\tau|\mathbf{y}_{1:\tau}) p(\mathbf{y}_{1:\tau})}{p(\mathbf{y}_{\tau+1:t}|\mathbf{y}_{1:\tau}) p(\mathbf{y}_{1:\tau})} \\ = & \frac{p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau) p(\mathbf{x}_\tau|\mathbf{y}_{1:\tau})}{p(\mathbf{y}_{\tau+1:t}|\mathbf{y}_{1:\tau})}. \end{aligned} \tag{2.47}$$

In turn, $p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau)$ can be computed recursively as:

$$\begin{aligned} p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau) &= \int p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_\tau, \mathbf{x}_{\tau+1}) p(\mathbf{x}_{\tau+1}|\mathbf{x}_\tau) d\mathbf{x}_{\tau+1} \\ &= \int p(\mathbf{y}_{\tau+1:t}|\mathbf{x}_{\tau+1}) p(\mathbf{x}_{\tau+1}|\mathbf{x}_\tau) d\mathbf{x}_{\tau+1} \\ &= \int p(\mathbf{y}_{\tau+1}, \mathbf{y}_{\tau+2:t}|\mathbf{x}_{\tau+1}) p(\mathbf{x}_{\tau+1}|\mathbf{x}_\tau) d\mathbf{x}_{\tau+1} \\ &= \int p(\mathbf{y}_{\tau+1}|\mathbf{x}_{\tau+1}) p(\mathbf{y}_{\tau+2:t}|\mathbf{x}_{\tau+1}) p(\mathbf{x}_{\tau+1}|\mathbf{x}_\tau) d\mathbf{x}_{\tau+1}. \end{aligned} \tag{2.48}$$

All these equations can be derived directly by inspecting the corresponding Bayes network. Furthermore, the recursion relationships can be regarded as propagating a message forward or backward through the model. This observation represents the basis for the message-passing nomenclature used next.

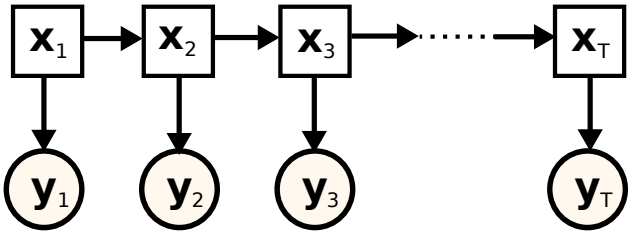


Figure 2.6: A HMM represented as a DBN.

Discrete hidden variables

Grid-based methods deal with the case of discrete hidden variables [7], when the states are associated to labels. Often, the number of states is also finite and these methods are then called finite-state models. A sequence of hidden variables represents thus a 1D discrete signal. The best-known algorithm of this type is the HMM, however it has a set of limitations that the MEMM overcomes. Both these algorithms are discussed next with an emphasis on the way they are related to each other.

Hidden Markov models and maximum entropy Markov models. The HMMs are *generative models* able to describe the joint probability $p(x_1, \ldots, x_T, \mathbf{y}_1, \ldots, \mathbf{y}_T)$ of a sequence of observations $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ and a corresponding sequence of states $\mathbf{x} = \{x_1, \ldots, x_T\}$. If we are able to compute the joint probability of all variables of interests, then joint probabilities of subsets of variables are obtained by marginalization and conditional probabilities are obtained as a ratio of two marginals. For HMMs, the input observations are assumed independent given the state and any form of relationship among observations is explained only at the level of the states under the Markovian assumption. The DBN equivalent to a HMM for a sequence$\{\mathbf{y}_1, \ldots \mathbf{y}_T\}$ is shown in Figure 2.6. Such a batch of measurements represents at the same time a realization of a stochastic process of length T, whose corresponding graphical model is shown in the Figure.

There are however applications where, for increased descriptive power, we may like to include into our model relationships stretching over several observations directly and not only over the states. MEMM represent the adaptation of the HMMs to such a setup. They are *discriminative models* working in an a posteriori manner, which can describe the probability of states given observations. This conditional probability may depend on arbitrary functions of the observations, thus being able to model also relationships extending over many observations, like for example attributes shared by several observations. The statistical description of these relationships is transparent to the model [116]. In a way, the MEMM setup could be described as including a transformation of the observations followed by establishing a DBN-like stochastic model in the transformed space.

In the following we introduce each of these two algorithms, while concentrating on the MEMM, as the HMM represents an established algorithm well covered in the literature.

Hidden Markov models. Consider a system that at any time is in one of N different states $X = \{x_1, x_2, \ldots, x_N\}$. At regularly spaced discrete times, the system undergoes a state change.

A state-transition matrix $\mathbf{A} = \{a_{ij}\}$ governs the changing of the states:

$$a_{ij} = P(x_{t+1} = j | x_t = i), \quad 1 \leq i, j \leq N \text{ and } \sum_{j=1}^{N} a_{ij} = 1, \ \ 1 \leq i \leq N.$$

The states are elementary events of a discrete state random variable and the state transition matrix is actually a compact representation of the state-conditional frequency functions for each state in the model. The state process is a first order Markov process. Each state emits an observation (that in the most general case is a n-tuple), and only the observations are monitored. Each state has assigned an observation pdf. This pdf can be either discrete or continuous, but it must have a known parametric expression (see [20] pp. 612). We therefore have $B = \{b_j(\mathbf{y})\}, \ j = 1 \ldots, N$ the set of all state-conditional densities with

$$b_j(\mathbf{y}) = p(\mathbf{y} | x = j).$$

The initial state distribution is $\pi = \{\pi_i\}, \ i = 1 \ldots, N$, where

$$\pi_i = P(s_1 = i).$$

The parameter vector corresponding to such a a model is

$$\boldsymbol{\lambda} = (\mathbf{A}, B, \pi).$$

Such a model can be used to solve one of three basic problems [153]:

1. *The evaluation.* Using the observation sequence $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, compute $P(\mathbf{Y}|\boldsymbol{\lambda})$ the probability of the observations given the model $\boldsymbol{\lambda}$. The solution to this problem is returned by the forward and by the backward algorithm.

2. *The quest.* Given the observation sequence $\mathbf{Y}$ and the model $\boldsymbol{\lambda}$ find a state or a sequence of states $\mathbf{x} = \{x_1, \ldots, x_T\}$ that are optimal. In this context, optimal means most probable and we can distinguish four cases:

   i. Find the most probable next state given all available states and observations. The solution to this problem is given again by the forward algorithm.

   ii. Find the most probable current state, when the observation sequence includes the current observation. The solution to this problem is given again by the forward algorithm.

   iii. Find the most probable past state, when the observation sequence includes observations both from before and after the observation corresponding to the sought state. The solution to this problem is given by the forward-backward algorithm.

   iv. Find the most probable sequence of states for the given sequence of observations. It is said that this sequence best explains the observations. The solution to this problem is returned by the Viterbi algorithm, that is a type of max-sum algorithm.

3. *The training.* Adjust the model parameters $\boldsymbol{\lambda}$ to maximize $P(\mathbf{Y}|\boldsymbol{\lambda})$. Starting from an observation sequence $\mathbf{Y}$ and a model $\boldsymbol{\lambda}$, find a better model $\boldsymbol{\lambda}_{new}$. The solution to this problem is computed by the Baum-Welch algorithm, which, as previously discussed, is a type of generalized EM.

For an excellent description of the algorithms representing the solution to each of these problems, see Rabiner's tutorial on HMMs [153].

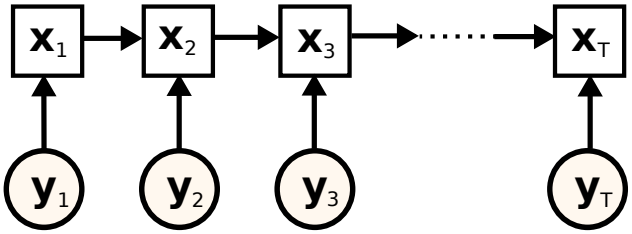


Figure 2.7: A MEMM represented as a DBN.

Maximum entropy Markov models. In the case of MEMMs, the state transitions and state-conditional observations densities are replaced by a set of transition functions, each single function describing the probability of the current state x, given the previous state x', upon making an observation $\mathbf{y}$. Thus, for a model with a number of N states, we have in this case N transition functions $p_j(x|x'_j, \mathbf{y}),\ j = 1, \ldots, N$, representing distributions over possible next states for each fixed x'_j. The DBN corresponding to a MEMM is shown in Figure 2.7. As it can be seen, the observations are nolonger independent given the state, as the current state nolonger d-separates the current and previous observations. With an MEMM we can therefore introduce in our model various non-independent relationships extending over several observations.

In the case of the MEMMs we avoid the direct modeling of the distribution of observations given the state. We make use of the maximum entropy framework to come up with a parametric expression for our transition probability. The parameters of each transition function are computed from a training set of observations using a type of generalized EM algorithm [136].

1. *The principle of maximum entropy.* We would like to determine the density of a random variable. Everything we know about this variable is that it generated a finite set of observations (i.e., a *sample*) S that we have at our disposal. Under a set of constraints (derived from the available sample), the density that best represents our knowledge on the random variable is the density making the least assumptions on the data [101]. This density has thus the largest entropy. Therefore, we look for the highest-entropy distribution incorporating the constraints.

   These constraints are pieces of information whose validity can be tested on the available sample. For example, the information that the expectations of various deterministic functions of the random variable have certain values $E\{f_i(x)\} = v_i,\ i = 1, \ldots, N_f$ can be tested on the sample. In this particular case, it can be shown [16] that if there exists a maximum entropy distribution, then this is

   $$p(x) = c \exp \sum_{i=1}^{N_f} \lambda_i f_i(x),\ \forall x \in S, \tag{2.49}$$

   with c a constant, and $\{\lambda_1, \ldots, \lambda_{N_f}\}$ a set of weights[11]. The density in (2.49,)which is a member of the exponential family and can be seen as a type of softmax function (see

[11] In an alternative motivation for this expression, the inner product between the weights vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{N_f}]^T$ and the corresponding vector of deterministic functions $\mathbf{f}(x) = [f_1(x), \ldots, f_{N_f}(x)]^T$ is interpreted as a measure for the plausibility of x. To construct a probability distribution out of this measure that can take both positive and negative values, the inner product is exponentiated (such as to always obtain a positive value) and then normalized. The normalization factor is given by $c = \int \exp \sum_i \lambda_i f_i(x) dx$. This argumentation

[20] pp. 198), is also known as a Log-Linear Model (LLM) because the logarithm of the density is a first-order polynomial with respect to the weights.

2. *The transition function.* In our MEMM case, we would like to take into account certain relationships among the observations. We do so with the help of the deterministic functions from before. We use these transition functions to link observations, observation properties and state transitions. They are usually boolean-valued and depend on the current observation and the possible next state, expressing also the presence or the absence of a certain attribute in the observation for a certain state (see also the discussion on "Ugly Duckling" in Section 1.2). For each pair $r = \langle a, x \rangle$, with $a : \mathcal{Y} \to \{0, 1\}$ a boolean-valued function of the observation and x the destination state, we define one function $f_r(\mathbf{y}_t, x_t)$ as

$$f_r(\mathbf{y}_t, x_t) = \begin{cases} 1 & \text{if } a(\mathbf{y}_t) = 1 \text{ and } x = x_t \\ 0 & \text{otherwise} \end{cases} \tag{2.50}$$

with $x \in X$ being one of the N possible states. The boolean function $a(\mathbf{y}_t)$ describes the presence or absence of an attribute in the observation.

3. *Training.* Next we assume that we have a training set of labeled observations. We introduce the constraints that the expectation of each such function equals its average over the training sample. Therefore we have that

$$\frac{1}{n_{x'_j}} \sum_{k=1}^{n_{x'_j}} f_r(\mathbf{y}_{t_k+1}, x_{t_k+1}) = \frac{1}{n_{x'_j}} \sum_{k=1}^{n_{x'_j}} \sum_{x \in X} p_j(x|x'_j, \mathbf{y}_{t_k+1}) f_r(\mathbf{y}_{t_k+1}, x),$$

with $t_1, \ldots, t_{n_{x'_j}}$ the time steps where $x_{t_k} = x'_j$, thus involving the transition function $p(x|x'_j, \mathbf{y}_{t_k+1})$. As discussed above, under these constraints, the sought distribution can be computed using equation (2.49) as

$$p(x|x'_j, \mathbf{y}) = \frac{1}{Z(\mathbf{y}, x'_j)} \exp \sum_r \lambda_r f_r(\mathbf{y}, x)$$

with λ_r parameters to be learned and $Z(\mathbf{y}, x'_j)$ a normalizing factor, such that $p(\cdot)$ is a pdf over the next states x. It is interesting to note that this is the probability of the current state x given an ***individual*** previous state $x'_j,\ j = 1, \ldots, N$ and the current observation $\mathbf{y}$, not the general probability of the current state given ***some*** previous state x' and the current observation.

As a matter of fact, the MEMM can be simplified by reducing the number of parameters when using such general densities. Under these circumstances, the observations and the previous state are treated as independent evidence for the current state [136]. As it can be seen from Figure 2.7, we then have

$$\begin{aligned} p(x|x', \mathbf{y}) &= p(x|x')p(x|\mathbf{y}) \\ &= p(x|x')c \exp \sum_r \lambda_r f_r(\mathbf{y}, x), \end{aligned}$$

applies even better when using the logistic sigmoid instead of the exponential function to construct the probability distribution. The logistic function takes values in the interval $(0, 1)$ rather than $(0, \infty)$ as in the case of the exponential.

the prior $p(x|x')$ describing the transition probability between states and the conditional $p(x|\mathbf{y})$, which is computed with the model stemming from the maximum-entropy principle, describing the influence of the observations. The precise choice of prior depends on the particular problem for which the MEMM is used.

Continuous hidden variables

For discrete DBNs, the state sequence is specified with the help of a set of state conditional probability mass functions, one for each state. In the case of HMMs, these are gathered in the state-transition matrix while in the case of MEMMs, these are the transition functions. For continuous DBNs however, the probability of a state taking a certain individual value is zero, thus we have to go from discrete to continuous distributions, i.e., from probability mass functions to probability density functions. These pdfs are derived from a state-transition function which returns the current state taking as argument the previous one, therefore, they link consecutive states directly instead of using statistical descriptions representing conditional probabilities of the next state given the current one (like the state-transition matrix of the HMMs or the transition distributions of the MEMMs). A DBN with continuous states is shown in Figure 2.8.

As the states are continuous, conducting inference is more cumbersome than for the grid-based methods. The majority of problems stem from the fact that we need to use integrals instead of sums for computing various probabilistic measures of interest (like, e.g., densities and their moments). Thus, the type of distributions involved with these methods represent a major point, as the equations (2.42) to (2.48) are tractable only under certain restrictions. The most important restriction is the assumption that the underlying probabilities are all Gaussian, as only when both distributions are Gaussian, is their product again Gaussian – which represents a huge advantage, as you don't have to integrate over difficult or even non-integrable functions. Yet another assumption, again related to mathematical tractability is that the dynamical system generating the states is linear. If all these assumptions are met, then the solution takes the form of a Kalman filter. If the assumption about the linearity is not met, then the solution is given by the extended Kalman filter and related approaches. If on top of this we renounce the Gaussianity assumption as well, the solution is given by the particle filter, where from a certain point of view, the involved integrals are solved with Monte Carlo integration, alternatively a variational approach can be used here. Ultimately both the Kalman and the particle filter represent modalities to conduct inference in a dynamical system with more respectively less constraints. Both these algorithms are discussed next, again placing an emphasis on the way they are related to each other.

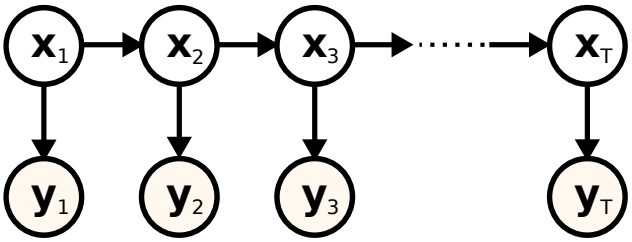


Figure 2.8: Bayes network for a dynamical system with continuous hidden variables.

Linear dynamical systems: the Kalman filter. A classic linear Gaussian state-space model is the Kalman filter [109], which in its essence is a least squares estimator of the states sequence [92]. For a given set of time-ordered observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ the Kalman filter returns the corresponding sequence of states $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. The Kalman filter maximizes the joint probability of observations and states, however, it does it sequentially finding at each time step the most probable state, given the available observations. In the case of the Kalman filter this procedure leads to a rather manageable mathematical apparatus, because of the assumption that all underlying distributions are Gaussian, and the Gaussian distribution is closed under multiplication, meaning that the joint distribution of two Gaussian variables is still Gaussian [20].

It uses the following assumptions:

- At each time step t, a d-dimensional real-valued observation $\mathbf{y}_t$ is generated from a k-dimensional, real-valued state $\mathbf{x}_t$.
- The initial state is Gaussian
$$\mathbf{x}_1 = \boldsymbol{\mu}_0 + \mathbf{u}$$
with $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_0)$
- The state sequence is a first order Markov signal and the observations are independent given the states.
- The state sequence is the realization of an AR process of the first order (linear dynamic system)
$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{v}^1, \tag{2.51}$$
with a Gaussian noise term $\mathbf{v}^1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma})$ and where $\mathbf{F}$ is the state-transition matrix.
- The output function generating the observation from a state is linear
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}^2, \tag{2.52}$$
again with Gaussian observation noise $\mathbf{v}^2 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and where $\mathbf{C}$ is the observation matrix.
- We look for $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$, that would allow us to compute:
$$E\{\mathbf{x}_t\} = \int \mathbf{x}_t p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t) d\mathbf{x}_t.$$

The parameters $\boldsymbol{\Theta} = \{\mathbf{F}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\mu}_\mathbf{0}, \mathbf{P}_\mathbf{0}\}$ are assumed to be known. They are either inferred from the problem framework or, they can be learned from some training data by a type of EM algorithm (see [20] pp. 642).

In general, $\mathbf{F}$ and $\mathbf{C}$ can be also time dependent, in which case we have $\mathbf{F}_t$ and $\mathbf{C}_t$ respectively. This is also valid for $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}$. If they are time-dependant, the values at each time t should be inferred from the problem-setup, usually over some features. If they remain constant, they may express some prior knowledge related to how much can observations be trusted over the states.

If the eignevalues of $\mathbf{F}$ are smaller than one, then the respective AR process will colapse to zero. Conversely, if they are larger then one, it will diverge to infinity [166].

The equations of the Kalman filter. Next we will make use of $\hat{\alpha}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$ to denote the pdf of a state given all available observations [20], underlining the fact, that this is a function of the current state $\mathbf{x}_t$. Due to our Gaussian assumption, we have that $\hat{\alpha}(\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{V}_t)$.

From equations (2.44) and (2.45) we have the following recursive relationship for $\hat{\alpha}(\mathbf{x}_t)$

$$\begin{aligned} c_t\hat{\alpha}(\mathbf{x}_t) &= p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) \\ &= p(\mathbf{y}_t|\mathbf{x}_t)\int p(\mathbf{x}_t|\mathbf{x}_{t-1})\hat{\alpha}(\mathbf{x}_{t-1})d\mathbf{x}_{t-1}, \end{aligned}$$

with $c_t = p(\mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$, which is again Gaussian. Therefore the weighted pdf of the current state given all available observations is computed as the product between the probability of the current observation given the current sate (evaluated as a function of the current state) and the pdf of the current state given all previous observations. This is illustrated in Figure 2.9.

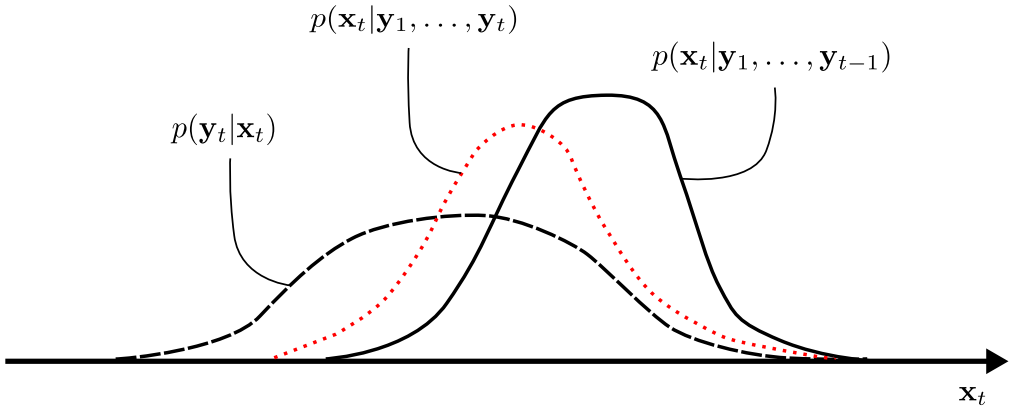


Figure 2.9: In this figure the updated density of the state upon making the current observation $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t) \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{V}_t)$ is shown together the probability density of the current observation as a function of the state $p(\mathbf{y}_t|\mathbf{x}_t) \sim \mathcal{N}(\mathbf{C}\boldsymbol{\mu}_t, \boldsymbol{\Sigma})$ and the predicted density of the state given all previous observations $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) \sim \mathcal{N}(\mathbf{F}\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1})$.

As all variables are Gaussian and assuming $\boldsymbol{\mu}_{t-1}$ and $\mathbf{V}_{t-1}$ known, we have

$$\begin{aligned} c_t\mathcal{N}(\boldsymbol{\mu}_t, \mathbf{V}_t) &= \mathcal{N}(\mathbf{C}\boldsymbol{\mu}_t, \boldsymbol{\Sigma})\int \mathcal{N}(\mathbf{F}\boldsymbol{\mu}_{t-1}, \boldsymbol{\Gamma})\mathcal{N}(\boldsymbol{\mu}_{t-1}, \mathbf{V}_{t-1})d\mathbf{x}_{t-1} \\ &= \mathcal{N}(\mathbf{C}\boldsymbol{\mu}_t, \boldsymbol{\Sigma})\mathcal{N}(\mathbf{F}\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}). \end{aligned} \tag{2.53}$$

Making use of equation (A.5) from Appendix A to compute

$$\mathcal{N}(\mathbf{F}\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}) = \int \mathcal{N}(\mathbf{F}\boldsymbol{\mu}_{t-1}, \boldsymbol{\Gamma})\mathcal{N}(\boldsymbol{\mu}_{t-1}, \mathbf{V}_{t-1})d\mathbf{x}_{t-1},$$

with

$$\mathbf{P}_{t-1} = \mathbf{F}\mathbf{V}_{t-1}\mathbf{F}^T + \boldsymbol{\Gamma},$$

we may compute the parameters of the densities on the left-hand side in equation (2.53) as a function of the parameters of the densities on the right-hand side, as described in equations (A.5) and (A.6). Thus, taking into consideration equations (2.51) and (2.52) the Gaussian density for

$\mathbf{x}_t$ has the parameters[12]

$$\begin{aligned}\boldsymbol{\mu}_t &= \mathbf{F}\boldsymbol{\mu}_{t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{C}\mathbf{F}\boldsymbol{\mu}_{t-1}) \\ \mathbf{V}_t &= (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{P}_{t-1},\end{aligned}$$

with

$$\mathbf{K}_t = \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}$$

the Kalman gain matrix. At the same time, the pdf of the current observation given all previous observations is

$$c_t = \mathcal{N}(\mathbf{C}\mathbf{F}\boldsymbol{\mu}_{t-1}, \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma}),$$

that can be used to compute $p(\mathbf{y}_1, \ldots, \mathbf{y}_t) = \prod_{i=1}^{t} c_i$, which is needed in various applications[13].

This gives us a complete set of recursive relations to compute the probability of the state given all available observations, upon receiving a new observation $\mathbf{y}_t$. The initial conditions are given by

$$p(\mathbf{y}_1)p(\mathbf{x}_1|\mathbf{y}_1) = p(\mathbf{x}_1)p(\mathbf{y}_1|\mathbf{x}_1) \Leftrightarrow c_1\hat{\alpha}(\mathbf{x}_1) = p(\mathbf{x}_1)p(\mathbf{y}_1|\mathbf{x}_1)$$

which, considering our initial assumptions, leads to

$$\begin{aligned}\boldsymbol{\mu}_1 &= \boldsymbol{\mu}_0 + \mathbf{K}_1(\mathbf{y}_1 - \mathbf{C}\boldsymbol{\mu}_0) \\ \mathbf{V}_1 &= (\mathbf{I} - \mathbf{K}_1\mathbf{C})\mathbf{P}_0 \\ c_1 &= \mathcal{N}(\mathbf{C}\boldsymbol{\mu}_0, \mathbf{C}\mathbf{P}_0\mathbf{C}^T + \boldsymbol{\Sigma})\end{aligned}$$

with

$$\mathbf{K}_1 = \mathbf{P}_0\mathbf{C}^T(\mathbf{C}\mathbf{P}_0\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}.$$

Summary of the Kalman filter Before applying the Kalman filter we need to find the initial parameters. This can be computed automatically from some training data or set manually based on a thorough understanding of the problem at hand. The set of initial parameters consists of the parameters of the density of the initial state $\mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0)$, the parameters of the linear dynamical system describing the states (i.e., the state-transition function) $\mathbf{F}$ and the state covariance matrix $\boldsymbol{\Gamma}$ together with $\mathbf{C}$ the parameters of the linear function relating states to observations and $\boldsymbol{\Sigma}$ the observation covariance matrix. With these initial parameters, upon making at time t a new observation $\mathbf{y}_t$, the Kalman filter works the following way:

- First we estimate the covariance matrix[14] of the density of the state at time t conditioned on the observations available up until $t-1$:

$$\mathbf{P}_{t-1} = \mathbf{F}\mathbf{V}_{t-1}\mathbf{F}^T + \boldsymbol{\Gamma}.$$

  Observation: The mean of this density is $\mathbf{m}_{t-1} = \mathbf{F}\boldsymbol{\mu}_{t-1}$. In some applications of the Kalman filter this is used to introduce into the model an additional set of parameters: the control variables $\mathbf{u}$, that are related to external influences, not inherent to the model. The control variables influence the mean $\mathbf{m}_{t-1}$ over the control matrix $\mathbf{B}$, such that this is computed now as: $\mathbf{m}_{t-1} = \mathbf{F}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_{t-1}$.

[12] If $\boldsymbol{\Gamma}, \boldsymbol{\Sigma}, \mathbf{F}$ and $\mathbf{C}$ are constant, then $\mathbf{P}_t$ and $\mathbf{K}_t$ will stabilze quickly and then remain constant as well.

[13] In tracking applications it can be used for gating, i.e., deciding if an observation belongs to the modelled process or not. For this purpose, the covariance matrix $\mathbf{S} = \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma}$ is used to compute a type of Mahalanobis distance d from the observation to the prediction $\mathbf{d} = (\mathbf{y} - \mathbf{C}\mathbf{x})^T\mathbf{S}^{-1}(\mathbf{y} - \mathbf{C}\mathbf{x}) + \ln|\mathbf{S}|$.

[14] This is actually the covariance matrix of the predicted state density.

- We continue by computing the Kalman gain matrix:

$$\mathbf{K}_t = \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \mathbf{\Sigma})^{-1}.$$

- The Kalman gain matrix $\mathbf{K}_t$ can be used directly to compute:
  - The mean $\boldsymbol{\mu}_t$ of the state density conditioned on all available observations, including the one made at time t as:

$$\boldsymbol{\mu}_t = \mathbf{F}\boldsymbol{\mu}_{t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{C}\mathbf{F}\boldsymbol{\mu}_{t-1}).$$

  - The corresponding covariance matrix[15] $\mathbf{V}_t$ as:

$$\mathbf{V}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{P}_{t-1}.$$

Furthermore, we can also compute the pdf of the new observation given all previous observations as:

$$c_t = \mathcal{N}(\mathbf{C}\mathbf{F}\boldsymbol{\mu}_{t-1}, \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \mathbf{\Sigma}).$$

As it can be observed, these formulae compute directly the mean of the probability density of the current state given all available observations. As the density is Gaussian and it thus takes its maximal value at the position of the mean, we can safely consider it as the best available estimate of the sought state.

Related methods. The Kalman filter has generated several related methods aimed at overcoming its limitations with respect to the type of dynamical system it models. The most important are the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). The former works with a piece-wise linear model, the latter can work with highly nonlinear dynamical systems using a special type of sampling.

The EKF uses a first-order linear approximation of the involved nonlinear functions. A Taylor series-based first order approximation of a nonlinear function $h(x)$ around a point p can be computed as:

$$\begin{aligned} h(x) &= h(p) + \frac{(x-p)}{1!}h'(p) + H.O.T. \\ &= xh'(p) + h(p) - ph'(p) \\ &= ax + b \end{aligned}$$

with $a = k'(p)$ and $b = h(p) - ph'(p)$. $H.O.T.$ is an acronym for Higher Order Terms and contains all the rest of the Taylor expansions terms of an order higher than one that are ignored in this case.

The EKF uses a Taylor series-based approximation of the known nonlinear dynamic system involved to compute an approximation of the sought state distribution. The same trick is used for the nonlinear state-conditional observation-generation function. The UKF uses a sample that allows the *exact* computation of the needed means and covariances then applies the nonlinearities to the sample and estimates again the (now nonlinearly transformed) parameters. Thus

[15] This is now the covariance matrix of the updated state density.

we obtain again an (mean and covariance-based) approximation of the sought state distribution, but this time using the exact, known dynamic system instead of its linear approximation as in the EKF case. The same type of processing applies to the conditional distribution of the observations as well. This procedure is called the *unscented transform* and yields in general better distribution approximations than what is used for the EKF. Because the used sample is generated directly from the known state distribution at the previous step and such that it perfectly reflects its mean and covariance, this is also called *deterministic sampling*.

The UKF should be used over the EKF if the nonlinearities at the level of the stat-transition equation and the observation generation function are too strong for a first-order approximation or if the Jacobi matrix involved in the computation of the EKF for vector-valued random variables can not be computed efficiently[16].

Neither the UKF nor the EKF explicitly renounce the Gaussianity assumption, they both process only moments up to the second order. As previously discussed Kalman filters are basically least-squares estimators, and such an estimator achieves the Cramer-Rao bound only under the Gaussian assumption. Therefore other methods are needed to achieve optimality when working in a non-Gaussian environment. Furthermore, if the involved distribution are "difficult" such that it is hardly possible to generate a sample to precisely reflect their mean and covariance, even the UKF would function poorly, as the deterministic sampling would fail. The solution to all these problems is the Particle Filter that is a Monte Carlo-based method using importance sampling.

Non-linear dynamical systems: the Particle Filter. If the state transition function is not linear, or/and the underlying distributions are not Gaussian, the Kalman filter returns just the best second order approximation of the true state process. To further minimize the approximation error in such cases, we need a new type of model. The main difficulty is given by the fact that once we leave the Gaussian assumption, the mathematical apparatus needed to update the state sequence when a new observation arrives (see equation (2.44)) becomes intractable. Thus, we need alternative methods to compute $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$. Furthermore, as we leave also the linearity assumption with respect to the way the states are generated, we need to evaluate some expectation over a nonlinear function.

The currently available solutions to these problems can be gathered into two groups: variational approximations [107] and Monte Carlo approximations [20]. Within the framework of Monte Carlo approximations the solution takes the form of the particle filter [7].

With particle filters, to approximate the sought expectation we use a set of samples randomly generated by the corresponding pdf. The main difficulty consists now in the fact that the density we would like to sample is unknown. At the same time we look for a solution able to work in a sequential manner.

Similar to the Kalman filter, the particle filter returns in a sequential manner for a set of time-ordered observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ the corresponding sequence of states $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, finding at each time step the most probable state, given the available observations. It uses the following assumptions:

- At each time step t, a d-dimensional real-valued observation $\mathbf{y}_t$ is generated from a k-dimensional, real-valued state $\mathbf{x}_t$.

[16]This may happen if the derivatives of the function are difficult to derive analytically or they come with a high computational cost when approximated numerically.

- The state sequence is a first order Markov signal and the observations are independent given the states.
- The initial-state pdf (i.e, the prior) is known.
- The state sequence is the realization a non linear dynamical system $f(\mathbf{x})$, with a known noise term
$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{v}_t^1 \tag{2.54}$$
- The output function $g(\mathbf{x})$ generating the observation from a state is also non linear and afflicted by noise whose density is known
$$\mathbf{y}_t = g(\mathbf{x}_t) + \mathbf{v}_t^2. \tag{2.55}$$
- The sought expectation is now again
$$E\{\mathbf{x}_t\} = \int \mathbf{x}_t p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t) d\mathbf{x}_t \tag{2.56}$$
but the posterior $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$ is difficult to evaluate analytically, due to the nonlinearity $f(\cdot)$. This is why we use here a sample-based estimate[17]. We could then approximate the expectation in a maximum likelihood approach with the help of a set of N_P i.i.d. samples from $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$
$$E\{\mathbf{x}_t\} \cong \frac{1}{N_P} \sum_{i=1}^{N_P} \mathbf{x}_t^{(i)}.$$

We assume here also that the way the noise influences the states and the observations is linear. The particle filter works even if this is not the case, however, for didactic purposes, we would like to emphasize here the relationship to the Kalman filter.

The equations of the Particle Filter. To begin with, let us remember that what we are actually looking for is $p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$, the probability of a states sequence given an observations sequence. In the particle filter setup we have no closed-form solution for this probability and hence no way to search for its maximum analytically.

Thus we set forth to estimate this probability. We are able to do this if we have a set of samples from the corresponding random variable together with a set of associated weights. Each sample represents an observed value (i.e., realization) of the random variable. We call this sample-weight pairs *particles*. The weights are chosen according to the principles of *importance sampling*. Afterwards, we need to make this estimate sequential, which leads to the problem of degeneracy where after a few iterations, all particles but one will have very small weights. This problem can be solved by resampling, meaning generating a new set of samples each time the *degeneracy* appears – for which purpose we need a way to measure the degeneracy.

1. *Sample-based estimate of a density.* Our purpose is to estimate a density function $p(x)$ from a set of samples that we know are i.i.d. drawn from the target distribution. The available set of samples is $\{x^1, \ldots, x^{N_P}\} \equiv \{x^{(i)}\}, i = 1, \ldots, N_P$. To introduce the

[17] In this case we use stochastic sampling, like e.g., importance sampling which is different from the deterministic sampling used for the UKF.

density estimate, we start with two examples illustrating various views on this issue, thus helping us to better grasp the expression used in the end.

Using $\{x^{(i)}\}$, the ML estimate of the mean is the sample average:

$$m_x \cong \frac{1}{N_P}\sum_{i=1}^{N_P} x^{(i)}.$$

On the other hand, for a discrete random variable, assuming $\{x^{(i)}\}$ covers its entire support, its probability frequency function can be written as

$$\begin{aligned} p(x) &= \sum_{i=1}^{N_P} P(x^{(i)})\delta(x - x^{(i)}) \\ &= \sum_{i=1}^{N_P} w^{(i)}\delta(x - x^{(i)}), \end{aligned}$$

with $w^{(i)} = P(x^{(i)})$ and $P(x^{(i)}) = \frac{N_{x^{(i)}}}{N}$ the probability of $x^{(i)}$, where $N_{x^{(i)}}$ is the number of times the value $x^{(i)}$ appears in $\{x^{(i)}\}$. This allows us at the same time to compute a continuous estimate of a continuous random variable x starting from a sample $\{x^{(i)}\}$ of x.

Assuming we have a set of i.i.d. samples $\{x^{(i)}\}, i = 1, \ldots, N_P$ from the random variable x, and a corresponding set of weights $\{w^{(i)}\}, i = 1, \ldots, N_P$, making together a set of particles $\{x^{(i)}, w^{(i)}\}, i = 1, \ldots, N_P$, an empirical estimate of the density function of x is:

$$p(x) \cong \sum_{i=1}^{N_P} w^{(i)}\delta(x - x^{(i)}). \tag{2.57}$$

For the particle filter, it follows directly that

$$p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k}) \cong \sum_{i=1}^{N_P} w^{(i)}\delta(\mathbf{x}_{1:k} - \mathbf{x}_{1:k}^{(i)}), \tag{2.58}$$

with $\{\mathbf{x}_{1:k}^{(i)}\}, i = 1, \ldots, N_P$ a set of support points, $\{w^{(i)}\}, i = 1, \ldots, N_P$ a set of weights and $\mathbf{x}_{1:k} = \{\mathbf{x}_j\}, j = 1, \ldots, k$ all states up to k, i.e., the entire trajectory.

2. *Importance sampling.* We would like now to provide a sample-based estimate for a density $p(x)$ according to equation (2.57). However, we assume that it is difficult to extract samples from $p(x)$ but it can be easily evaluated at every point[18]. Then, we make use of another density $q(x)$ termed *importance density*, from which samples can be generated with ease and it can also be evaluated at every point. The general idea is that a realization sampled from $q(x)$ is used to compute $p(x)$ while compensating for its frequency of appearance, which is different between the two densities.

[18]This happens very often in practice as we are technically able to properly generate samples directly only from a relatively limited number of distributions.

Imagine you have extracted N samples $\{x^{(j)}\}, j = 1, \ldots, N$ from $q(x)$ and now you would like to compute the probability $P([T_1 \leq x \leq T_2]) = \int_{T_1}^{T_2} p(x)dx$. Should you have extracted from $p(x)$ a good estimate of the sought probability would have been

$$\begin{aligned} P([T_1 \leq x \leq T_2]) &= \frac{\text{\# samples} \in [T_1 \leq x \leq T_2]}{N} \\ &= \frac{1}{N} \sum_{l=1}^{N_j} [\![x^{(l)} \in [T_1 \leq x \leq T_2]]\!] \\ &= \frac{N_j}{N} \end{aligned}$$

with $[\![\cdot]\!]$ the *Iverson bracket* – a binary function, that takes the value one only if its argument is true and zero otherwise – and N_j the number of samples in the interval $[T_1 \leq x \leq T_2]$. We have used this rather complicated formula only to rise attention to the fact that the probability is computed by adding a certain value (in this case one) for each sample in the target interval and dividing by the number of samples. Back to our original setup, we have N samples from $q(x)$. As before, we will compute the sought probability by adding a certain value for each component of the sample in the target interval, then dividing by the size of the sample. To compensate for the fact that we sample from another distribution as the one we need, we compute the probability as

$$P([T_1 \leq x \leq T_2]) = \frac{1}{N} \sum_{k=1}^{N_j} \frac{p(x^{(k)})}{q(x^{(k)})},$$

thus adding for each sample k the value $\frac{p(x^{(k)})}{q(x^{(k)})}$, which compensates for the different frequency of appearance of $x^{(k)}$ between $p(x)$ and $q(x)$.

This is shown in Figure 2.10. In this example, the sample $\{x^{(j)}\}, i = 1, \ldots, N_P$ was generated from the importance density $q(x)$. Using the sample together with the values of $p(x)$ and $q(x)$ corresponding to the realizations in the sample, we may compute $P(x \in \{1, 2\}) = \frac{1}{9} \cdot (\frac{4}{9} \cdot \frac{9}{1} + \frac{2}{9} \cdot \frac{9}{2} + \frac{2}{9} \cdot \frac{9}{2}) = \frac{1}{9} \cdot \frac{6}{1} = \frac{2}{3}$.

Based on such considerations, in our case we extract $\{x^{(i)}\}, i = 1, \ldots, N_P$ samples from $q(x)$ and estimate $p(x)$ with equation (2.57), where the sample weights are now

$$w^{(i)} \propto \frac{p(x^{(i)})}{q(x^{(i)})}.$$

For the particle filter, it follows that the samples $\mathbf{x}_{1:k}^{(i)}$ are drawn from $q(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$ and the corresponding weights are:

$$w^{(i)} \propto \frac{p(\mathbf{x}_{1:k}^{(i)}|\mathbf{y}_{1:k})}{q(\mathbf{x}_{1:k}^{(i)}|\mathbf{y}_{1:k})}. \tag{2.59}$$

3. *Sequential importance sampling.* Next we assume that $p(\mathbf{x}_{1:k-1}|\mathbf{y}_{1:k-1})$ is available and we would now like to find $p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$ upon observing $\mathbf{y}_k$. We choose $q(\cdot)$ such that it

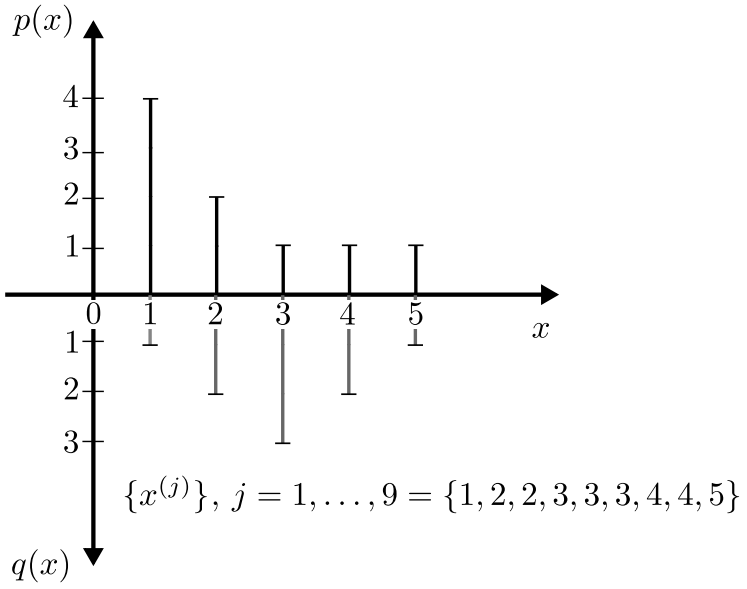


Figure 2.10: In this figure the target density $p(x)$ is depicted in black and the importance density $q(x)$ in gray. Also shown is the sample $\{x^{(j)}\}, j = 1, \ldots, 9$ extracted from $q(x)$.

can be described by the graphical model in Figure 2.8 as well. Then, $q(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$ can be computed as:

$$\begin{aligned} q(\mathbf{x}_{1:k}|\mathbf{y}_{1:k}) &= \frac{q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})q(\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})}{q(\mathbf{y}_{1:k})} \\ &= \frac{q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})q(\mathbf{y}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k-1})q(\mathbf{x}_{1:k-1},\mathbf{y}_{1:k-1})}{q(\mathbf{y}_{1:k})} \\ &= \frac{q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})q(\mathbf{y}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k-1})q(\mathbf{x}_{1:k-1}|\mathbf{y}_{1:k-1})q(\mathbf{y}_{1:k-1})}{q(\mathbf{y}_k|\mathbf{y}_{1:k-1})q(\mathbf{y}_{1:k-1})} \\ &= q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})q(\mathbf{x}_{1:k-1}|\mathbf{y}_{1:k-1})\frac{q(\mathbf{y}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k-1})}{q(\mathbf{y}_k|\mathbf{y}_{1:k-1})}. \end{aligned}$$

Upon observing $\mathbf{y}_k$, we have that:

$$q(\mathbf{x}_{1:k}|\mathbf{y}_{1:k}) \propto q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})q(\mathbf{x}_{1:k-1}|\mathbf{y}_{1:k-1}).$$

Then, the samples $\mathbf{x}_{1:k}^{(i)}$ drawn from $q(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$ can be obtained by augmenting each $\mathbf{x}_{1:k-1}^{(i)}$ drawn from $q(\mathbf{x}_{1:k-1}|\mathbf{y}_{1:k-1})$ with $\mathbf{x}_k^{(i)}$ drawn from $q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{y}_{1:k})$. The weight-update equation becomes [7]:

$$\begin{aligned} w_k^{(i)} &\propto \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})p(\mathbf{x}_{1:k-1}^{(i)}|\mathbf{y}_{1:k-1})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{1:k-1}^{(i)},\mathbf{y}_{1:k})q(\mathbf{x}_{1:k-1}^{(i)}|\mathbf{y}_{1:k-1})} \\ &\propto w_{k-1}^{(i)}\frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{1:k-1}^{(i)},\mathbf{y}_{1:k})}, \end{aligned}$$

where we have used equation (2.43) in the numerator, with the evidence $p(\mathbf{y}_k|\mathbf{y}_{1:k-1})$ as a proportionality constant. By inspecting the graphical model corresponding to $q(\cdot)$ we see

that $q(\mathbf{x}_k|\mathbf{x}_{1:k-1}, \mathbf{y}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k)$, then we have

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)} \tag{2.60}$$

and we can therefore discard the trajectory $\mathbf{x}_{1:k-2}^{(i)}$ and the history of observations $\mathbf{y}_{1:k-1}$. We observe that for each particle, the update weight depends on the prediction of the sample $\mathbf{x}_k$ based on its previous value $\mathbf{x}_{k-1}$ and on the probability of the current observation given the state sample corresponding to this particular particle. Furthermore we can now compute directly the posterior:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \cong \sum_{i=1}^{N_P} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}). \tag{2.61}$$

Note that the sample practically does not change, the particles do however, because the corresponding weights get updated (see Figure 2.11).

Choosing the importance density $q(\cdot)$ is a design requirement of the particle filter. Although better choices can be made, in practice $p(\mathbf{x}_k|\mathbf{x}_k^{(i)})$ is often used as importance density,as the weight update equation (2.60) turns to:

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(\mathbf{y}_k|\mathbf{x}_k^{(i)}).$$

4. *The degeneracy problem.* As the variance of the importance weights always increases with time, after some iterations, only very few particles will still have sufficiently large weights to matter when computing the estimate (see Figure 2.11). The degeneracy can be measured with the help of the effective number of samples defined as:

$$\widetilde{N}_{eff} = \frac{1}{\sum_{i=1}^{N_P} \left(w_k^{(i)}\right)^2}. \tag{2.62}$$

As it can be easily seen, $\widetilde{N}_{eff} \leq N_P$, the equality existing only when each weight has the value $\frac{1}{N_P}$. Thus the smaller $\widetilde{N}_{eff}$, the larger the degeneracy.

5. *Resampling.* The degeneracy problem can be tackled in two ways. The first would be to choose an optimal importance density that is generally viable only under additional constraints like a discrete and finite number of states or Gaussian noise sequences $\mathbf{v}_t^1$ and $\mathbf{v}_t^2$ and linear noise influences (see equations (2.54) and (2.55)). Alternatively we need to generate a new set of particles including a new sample $\{\mathbf{x}_k^{(i)new}\}, i = 1, \ldots, N_P$ each time the degeneracy becomes too large.

The resampling eliminates particles with small weights by sampling again N_P times from $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ as defined in equation (2.61) and such that $P(\mathbf{x}_k^{(i)new} = \mathbf{x}_k^{(i)}) = w_k^{(i)}$. In the resulting sample all weights are reset to $w_k^{(i)new} = \frac{1}{N_P}$. The resampling process is illustrated in Figure 2.11. The number of particles is kept constant. In some cases, not $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ is resampled, but rather $\{x^{(i)}\}, i = 1, \ldots, N_P$ is resampled according to $p(\mathbf{x}|\mathbf{y}_{1:k})$. In such cases, the samples $\mathbf{x}$ that get large weights will b repeated several times to reflect their importance. Several computation-efficient resampling schemes exist [7].

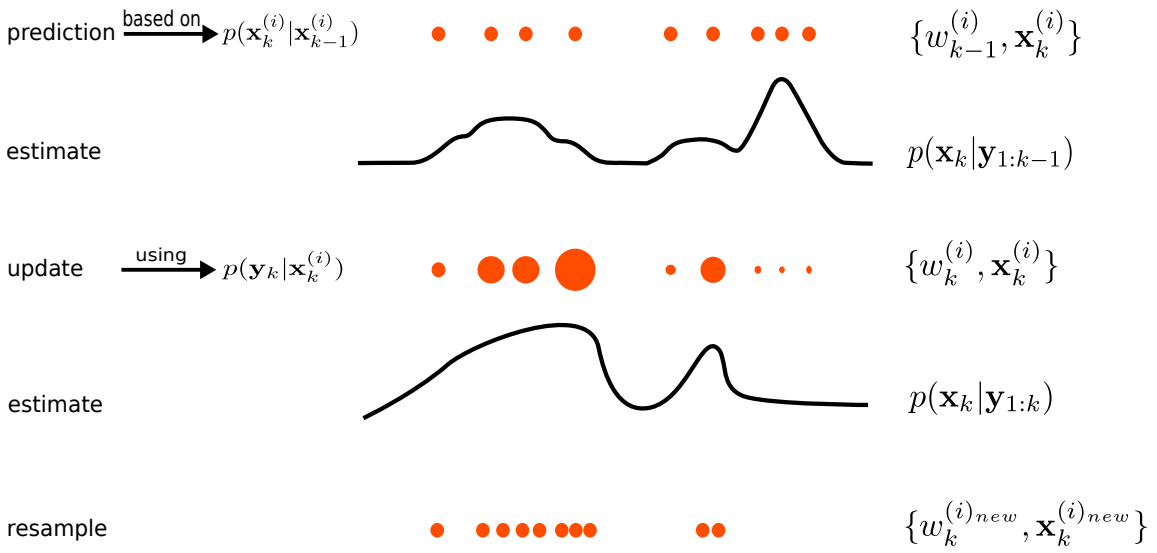


Figure 2.11: In this figure are depicted from top to bottom: the original particles as predicted at iteration $k-1$, the updated particles at iteration k (where some particles degenerate), the corresponding posterior at iteration k and the new particles, obtained after resampling.

Summary of the Particle Filter. In summary, the particle filter works like this:

(i). First, some particles are generated:

- Their sample values $\mathbf{x}_{k-1}$ are extracted according to some prior.
- Their weights are all equal.

(ii). The particles are predicted:

- Their sample values are computed from $\mathbf{x}_{k-1}$, using $f(\cdot)$.
- Their weights are all equal.

(iii). The particles are updated:

- Their sample values remain the same as at the previous step.
- The weights get updated upon arrival of $\mathbf{y}_k$. They reflect how good the corresponding sample $\mathbf{x}_k^{(i)}$ explains $\mathbf{y}_k$ according to $g(\cdot)$, such that the higher $p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$, the larger the weight. In other words, particles whose samples $\mathbf{x}_k$ better explain $\mathbf{y}_k$ according to $g(\cdot)$ (i.e., the probability $p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$ is larger) receive larger weights.

(iv). With the help of this particles, $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ is estimated.

(v). In the resample step, a new set of particles is generated[19]. Two main cases exist here (in both cases the number of particles remains constant.):

A. $*$ New sample values are generated according to $q(\cdot)$. Note that $q(\cdot)$ could also be approximated by $p(\mathbf{x}_k|\mathbf{y}_{1:k})$.

[19]Theoretically this happens only if N_{eff} is small enough, practically this happens at each iteration step.

- ∗ Their weights are set equal.

B.
- ∗ The sample values from the previous step are used. The samples corresponding to previous particles with large weights get duplicated at a rate that is proportional to the weight.
- ∗ The weights of these new particles are again set equal.

We have introduced here the particle filter at a conceptual level. This general particle filter is known as the Sequential Importance Sampling (SIS) particle filter. The practical implementation poses additional challenges this constituting an active research field loosely termed Sequential Monte Carlo [64].

The particle filter as a more general – as in making fewer assumptions – type of method is appropriate for a larger set of practical problems than the Kalman filter and its derivates. Nevertheless even for problems where the Kalman filter is not optimally suited, as its assumptions do not hold, it should still be taken into consideration. In such cases, it internal mechanics represent only an approximation to the underlying distributions, however, it often returns faster, better results by comparison to a standard (i.e., with a manageable number of particles) particle filter. Here appears thus a window of opportunity for complementary Gaussianization methods aimed at gaussianizing the data, such that the assumptions of the Kalman filter hold. Gaussianization methods are discussed in the context of classification problems in Chapter 3.

2.2.2 Markov random fields and conditional random fields

As we have discussed before, graphical models are related to conditional independence properties in a set of random variables. These independence properties lead to a certain factorization of their joint distribution into a set of conditional distributions.

Conditional independence. One of the main reasons for using this framework at all was the ability to gain insight into these independence relationships by inspecting the graph and applying graphical conditional independence tests. For Bayes networks, this test is the d-separation. For Markov random fields conditional independence is related directly and simply to actual graph separation. Considering the sets of nodes A, B and C, we can say that A is conditionally independent from C given B, if all paths from A to C are blocked by B, meaning that all connections from A to C must go through B. Such a Markov random field [20] is shown in Figure 2.12. In this case, a node is conditionally independent from all other nodes in the network given only its neighbors, i.e., the nodes to which it is connected by a direct path. The set of neighboring nodes including the other parents of child nodes is called the *Markov blanket* of a node.

Cliques, potentials and the Hammersley-Clifford theorem. To compute the joint probability of all variables in the field, we will use cliques. A *clique* is a subset of linked nodes such that each pair is linked. A *maximal clique* is a clique where the addition of any other node from the graph results in it not being a clique anymore. For each node, there is a number $\underline{C}$ of cliques, each one including $\mathbf{x}_C$ of its neighbor nodes. We can associate to each clique a strictly positive function defined over the set of configurations of the member nodes, where by a configuration we understand a set of realizations, one for each random variable at each node. This function

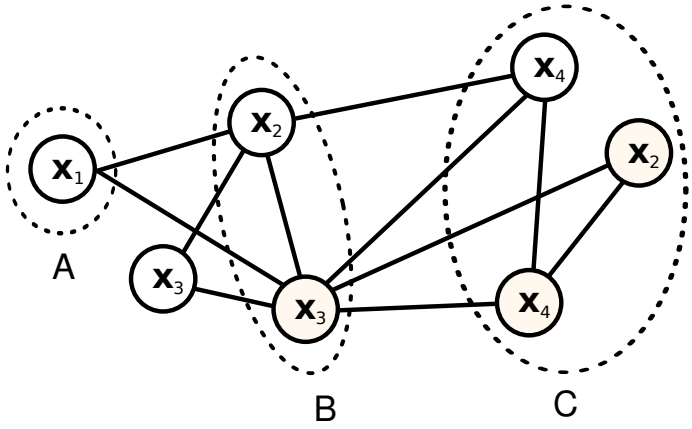


Figure 2.12: A MRF with sets of nodes related by the conditional independence relationship $A \perp\!\!\!\perp C|B$ and different states.

is called *potential*. The practical intuition behind these potential functions is that they select (by assigning larger values) configurations of realizations of random variables that should be preferred (having higher probability) over others.

The joint probability of all nodes is computed as a product of potentials of maximal cliques. The potential of a maximal clique can be defined for example as the product of the potentials corresponding to each clique that can be constructed from the nodes in the maximal clique[20]. Then, by the Hammersley-Clifford theorem [17, 80], the joint probability of all variables in the field is defined with the help of the set C of maximal cliques as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C} \psi_C(\mathbf{x}_C), \tag{2.63}$$

with $\psi(\mathbf{x}_C)$ potential functions over maximal cliques and Z a normalization constant such that $p(\mathbf{x})$ is a pdf. The *Gibbs measure* $p(\mathbf{x})$ is consistent with the conditional independence relationships that can be inferred by visual inspection of the corresponding graph as described above. In some applications, $p(\mathbf{x})$ is rewritten as

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left[-\sum_{C} E(\mathbf{x}_C)\right] \tag{2.64}$$

with $\psi_C(\mathbf{x}_C) = \exp\left[-E(\mathbf{x}_C)\right]$ and $E(\cdot)$ the energy function.

Comparing equation (2.41) with equation (2.63) we observe that both depend on functions of neighbors. The main difference is that in the former case, these functions have a precise statistical meaning, being themselves densities and thus normalized (i.e., they integrate to one), while in the latter case, these are some unnormalized functions, which explains also the presence of the normalizing factor Z. Making use of unnormalized functions provides for a larger model capacity.

From directed to undirected graphs. It is possible to convert a directed graph to an undirected one that uses as potentials the conditional densities from the directed graph. These special

[20]There are also other ways to define this potential. Factor graphs [20] are instrumental in keeping track of the factors in potentials. Each factor involves then only a subset of nodes from the maximal clique.

undirected graphs are called "moral" graphs. However, the factorization of $p(\mathbf{x})$ corresponding to a moral graph can not be directly related to conditional independence properties as in the case of densities and directed graphs.

In a *moral graph*, any two parents of any node are linked being thus "married" (i.e., forming a pair) and ensuring that all nodes involved in a conditional appear together within a clique. The process of moralization that transforms a directed graph into an undirected graph using the conditionals as potentials is helpful for conducting exact inference. More precisely, with the help of moralization we can apply the *junction-tree algorithm* for conducting exact inference in undirected models, to directed models as well. The junction-tree algorithm has also a "triangulation" step where it is ensured that no loop is chord-less [20] and which is the basis for computing probabilities in a recursive manner [107].

Conditional random fields. Conditional random fields (CRF) that are the main topic of this section, represent a combination of directed and undirected models. They can, among others, be used as an adaptation of Markov random fields to a setup in which we have sequential data.

With $\mathbf{x}$ a sequence of labels, $\mathbf{Y}$ a sequence of observations, and $G = (V, E)$ a graph with V vertices and E edges, such that $\mathbf{x}$ is indexed by the vertices of G, the definition of CRFs according to Lafferty [116] is:

$(\mathbf{Y}, \mathbf{x})$ is a CRF when conditioned on $\mathbf{Y}$, the random variables $\mathbf{x}$ obey the Markov property $p(x_v|\mathbf{Y}, x_w, w \neq v) = p(x_v|\mathbf{Y}, x_w, w \bowtie v)$ with respect to the graph, where "$w \bowtie v$" means that w and v are neighbors in G.

It appears now clear that the CRFs are actually a mix between a directed and an undirected graph. There is an undirected graph relating the states/labels and a directed graph relating the observations to the states. Only for the states there are explicit Markovian relationships, while relationships among observations do not need to be explicitly represented [173]. Depending on the type of CRF, it may be that either all, some or each of the observations are linked to either all, some or each of the states. Such mixed models are sometimes called chain graphs [20, 77]. The corresponding graphical model is shown in Figure 2.13.

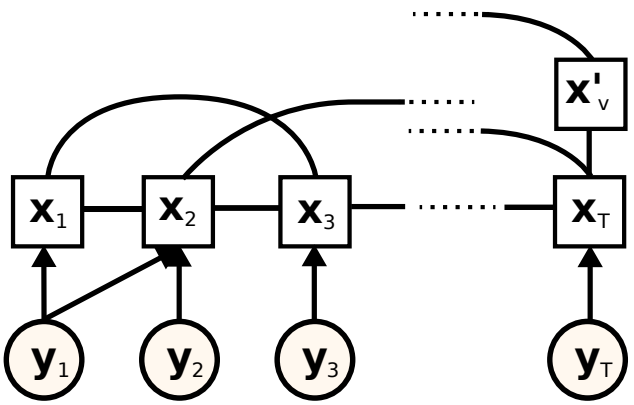


Figure 2.13: The graph representation of a CRF.

As we are in a sequential setup, where we need to find a set of labels for a set of observations, we assume next that we always have a state-observation pair. The CRF setup is more general

than this, but this assumption is enough to cover all applications of interest in our case while at the same time giving us the possibility to keep the discussion focused. Depending on how the states are connected to one-another, we differentiate between linear chain CRFs (LCCRFs) and arbitrary CRFs (ACRFs).

Linear-chain CRFs

LCCRFs are closely related to HMMs and MEMMs. In this case the states form a linear chain. The LCCRFs label a data sequence and are ill suited to label a single observation alone. Next we introduce LCCRFs starting from HMMs [173], and then discuss their relationship to MEMMs [116].

From HMMs to LCCRFs. HMMs describe $p(\mathbf{Y}, \mathbf{x}) = p(x_1)p(\mathbf{y}_1|x_1) \prod_{t=1}^{T} p(x_t|x_{t-1})p(\mathbf{y}_t|x_t)$, a joint probability of observations and states, that can be expressed also as

$$p(\mathbf{Y}, \mathbf{x}) = \exp\left(\sum_t \sum_{i,j \in X} \alpha_{ij} [\![x_t = i]\!][\![x_{t-1} = j]\!] + \sum_t \sum_{i \in X} \sum_{o \in Y} \beta_{io} [\![x_t = i]\!][\![\mathbf{y}_t = o]\!]\right),$$

with $[\![\cdot]\!]$ the Iverson bracket, $\alpha_{i,j} = \log\left(p(x = i|x' = j)\right)$, and $\beta_{i,o} = \log\left(p(\mathbf{y} = o|x = i)\right)$. Introducing the feature functions $f_{ij}(x_t, x_{t-1}, \mathbf{y}_t) = [\![x_t = i]\!][\![x_{t-1} = j]\!]\rceil(\mathbf{y}_t)$ and $f_{io}(x_t, x_{t-1}, \mathbf{y}_t) = [\![x_t = i]\!]\rceil(x_{t-1})[\![\mathbf{y}_t = o]\!]$, with $\rceil(x) = 1,\ \forall x \in \mathbb{C}$ we obtain

$$p(\mathbf{Y}, \mathbf{x}) = \exp\left(\sum_t \sum_{i,j \in X} \alpha_{ij} f_{ij}(x_t, x_{t-1}, \mathbf{y}_t) + \sum_t \sum_{i \in X} \sum_{o \in Y} \beta_{io} f_{io}(x_t, x_{t-1}, \mathbf{y}_t)\right),$$

which can be written in compact form (notice the similarity to equation 2.64) as

$$p(\mathbf{Y}, \mathbf{x}) = \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_t, x_{t-1}, \mathbf{y}_t)\right) \tag{2.65}$$

representing thus a LLM, as defined in equation (2.49). Equation (2.65) is a general expression, that has the advantage of allowing us to compute a probability measure also in the case when the parameters λ_k are no longer logarithms of densities. In this case however, if we desire the probability measure to be a density, we need to also multiply the right side of the relation in (2.65) with a normalization constant $\frac{1}{Z}$ such that it integrates to one.

For LCCRFs we are interested in $p(\mathbf{x}|\mathbf{Y})$ that, using (2.65), is computed as

$$\begin{aligned} p(\mathbf{x}|\mathbf{Y}) = & \frac{p(\mathbf{Y}, \mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{Y}, \mathbf{x}')} \\ = & \frac{1}{Z(\mathbf{Y})} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_t, x_{t-1}, \mathbf{y}_t)\right), \end{aligned} \tag{2.66}$$

with $Z(\mathbf{Y}) = \sum_{\mathbf{x}} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_t, x_{t-1}, \mathbf{y}_t)\right)$.

An HMM-like LCCRF has feature functions defined only for consecutive state pairs and for state-observation pairs, as shown in Figure 2.14. More general LCCRFs exist. Some of

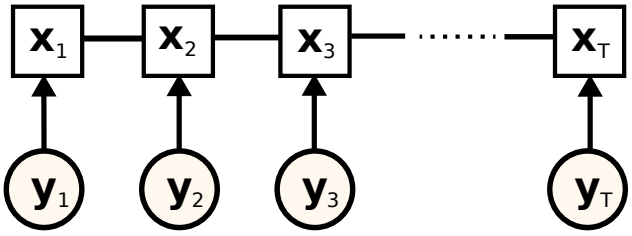


Figure 2.14: The graph representation of a HMM-like LCCRF.

them may allow a state transition to depend on one or even several observations – remembering that the LCCRFs label batches of observations. For example, if we want the state transition (j, i) to depend explicitly on the current observation o as well, we just add the feature function $f_k(x_t, x_{t-1}, \mathbf{y}_t) = [\![x_t = i]\!][\![x_{t-1} = j]\!][\![\mathbf{y}_t = o]\!]$. If we would like it to depend also on some previous observation p, we use the feature function $f_k(x_t, x_{t-1}, \mathbf{Y}_t) = [\![x_t = i]\!][\![x_{t-1} = j]\!][\![\mathbf{Y}_t = [p, o]^T]\!]$. A future observation f, defined with respect to some time stamp t within the currently labeled batch of observations, can be added the same way, with the help of the function $[\![\mathbf{Y}_t = [p, o, f]^T]\!]$. To the limit we can make any state-transition and also any state dependent on all observations. The graph structure corresponding to such LCCRFs is shown in Figure 2.15.

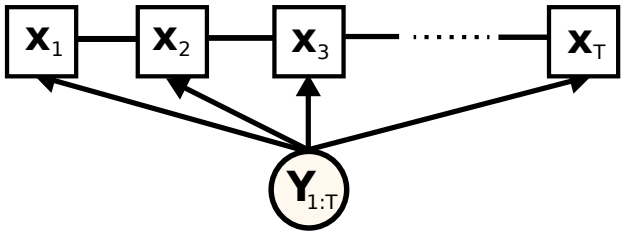


Figure 2.15: The graph representation of a general LCCRF.

Training LCCRFs implies the estimation of the parameters λ_k, given i.i.d. training data $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{Y}^{(i)}\}$, where $\mathbf{x}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots x_T^{(i)}\}$ and $\mathbf{Y}^{(i)} = \{\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, \dots \mathbf{y}_T^{(i)}\}$ are batches of labels and observations respectively. Note that the i.i.d. assumption is over the batches, not within every batch. There are various ways to conduct training and inference in LCCRFs [173, 116], many of them being derived from HMMs, like the Viterbi algorithm, or the usage of forward and backward paths.

LCCRFs and MEMMs. LCCRFs are closely related to MEMMs, making also use of feature functions that allow us to express various relationships among observations. While the MEMMs use an exponential model for each state, the LCCRFs use an exponential model for the joint probability of a batch of labels given a batch of observations. Thus, for LCCRFs the state-transition probabilities can be made to depend on the observations. This way, more problem-related information can be introduced into the model. Furthermore, the links among states are undirected for LCCRFs. Thus, dependencies with respect to both future and past states in the batch may be modeled. The LCCRFs do not suffer from the label-bias problem that appears during Viterbi decoding and afflicts the MEMMs, generating a bias towards states with fewer outgoing transitions [116].

Arbitrary CRFs.

Arbitrary CRFs (ACRFs) are not bound to a chain-like dependence among states. In this case we could even leave the assumption that every state has an observation attached and introduce virtual observation-less states that are there only to build certain cliques that are of interest. ACRFs are powerful models, when measuring the power of a graphical model by the number of distributions it can accommodate [20], and thus relating model-power to the generality of the model. However, in this case in general learning and inference are no longer exact and approximative methods are required. As previously discussed, exact solutions exist if we limit the CRF structure to a tree (or forest) [152], in the form of the sum-product algorithm or if we work with triangulated graphs, where the largest loop includes only three nodes, in the form of the junction-tree algorithm. Approximative methods include the loopy belief propagation algorithm [128], which is not guaranteed to converge [20], variational methods and sampling methods.

Grid CRFs. An ACRF encountered in practice is the Grid CRF [93], whose graph structure is shown in Figure 2.16. In this case, at each time instance, the state random variables enjoy also a spatial relationship, reflecting a similar relationship at the level of the observations. Such CRFs are well suited to work for example with video sequences where we can define a state (i.e., label) for each pixel in an image at each time instance, and relate labels over both space and time. Another example is the skip-chain CRF [173] that introduces dependencies of the

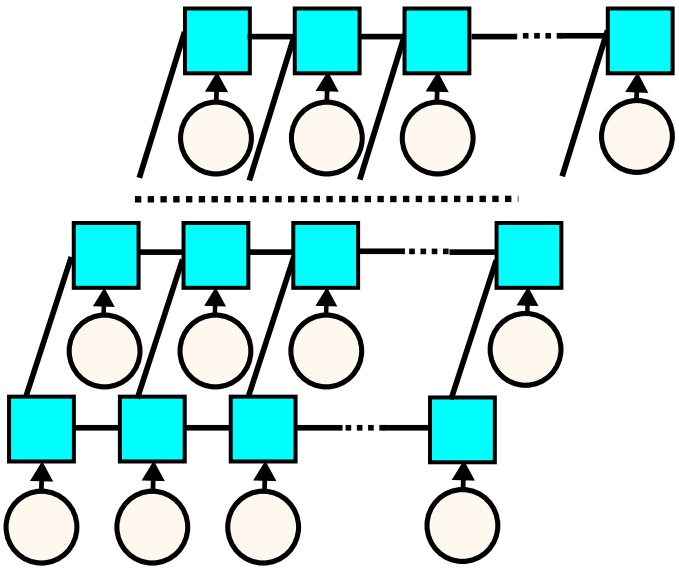


Figure 2.16: The graph representation of a grid CRF.

current state on other states than the previous one as well. The corresponding graph structure is shown in Figure 2.17.

Boltzmann machines. In the context of models for multidimensional random variables there is also a strong relationship between ACRFs and Boltzmann machines. *Boltzmann machines* are actually MRFs with a special type of cliques and accordingly energy functions. This type of cliques is known generally under the name Ising model. Conversely, the ACRFs are not

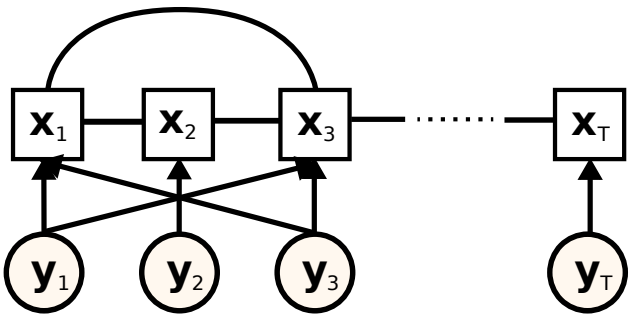


Figure 2.17: The graph representation of a skip-chain CRF.

limited to the Ising model. Such constraints make Boltzmann machines more easily tractable, however, they still involve too many computations for many practical applications [71] Further simplifications are achieved in the form of *restricted Boltzmann machines*(RBM), where the model topology is limited to a bipartite graph. RBMs are in turn inherently related to feed-forward neural networks [14, 71]. For example a multi-layer classification feed-forward neural network can be considered to consist of a RBM feature extractor with, e.g., a perceptron-based decision layer stacked on top of it.

2.3 Sparse representations

In their most general form, the *sparse representations* are methods that solve underdetermined systems of equations, when knowing that the sought solution is sparse. This setup is often encountered in many modern applications, perhaps the best known example being that of compressed sampling [25, 58]. After introducing the sparse framework in Section 2.3.1, we concentrate on using sparse representations for solving labeling problems and discuss the sparse classifier [187, 44] in Section 2.3.2.

2.3.1 Problem statement and applications

A system of equations $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ can be solved exactly if the matrix $\mathbf{A}$ is square and well conditioned, thus admitting an inverse $\mathbf{A}^{-1}$. This means that $\mathbf{A}$ corresponds to a set of M equations with N unknowns, with $M = N$ and none of the equations represents a linear combination of some (or all) of the others. The standard solution in this case – for systems with a manageable number of equations – is given by Cramer's rule. If the systems are very large, than their solution is found by methods like Gauss-Jordan Elimination or Low-Upper decomposition. Faster solutions are available if the matrix $\mathbf{A}$ fulfills certain properties, like for example the Cholesky decomposition for the cases when $\mathbf{A}$ is symmetric and positive definite, or the Levinson recursion for Toeplitz matrices.

When $M \neq N$, then we differentiate between two cases:

(i) When $M > N$ we have an overdetermined system of equations and in most cases no exact solution. We usually proceed by looking for the solution that best fulfills all conditions

expressed in the available equations. This solution is the one that minimizes the squared error term $\mathbf{e}^T\mathbf{e} = \| \mathbf{Ax} - \mathbf{b} \|_{\ell_2}^2$ and can be computed for example with the help of the Moore–Penrose pseudoinverse or with Singular Value Decomposition (SVD).

(ii) When $M < N$ we have an underdetermined system of equations and more than one solution. In this case, we have to choose from among the available solutions one of them. Usually we are after the one with minimum norm and to find it we solve the optimization problem:

$$\text{Find } \hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_{\ell_p} \quad \text{subject to } \mathbf{Ax} = \mathbf{b}. \tag{2.67}$$

Depending on how much information we have on our problem, we can choose the norm to use. The default is the ℓ_2 norm $\| \mathbf{x} \|_{\ell_2} = \| \mathbf{x} \|_2$, in which case a solution can be found for example again with the Moore–Penrose pseudoinverse or with the help of the SVD. However, if we know that $\mathbf{x}$ is sparse, we may want to use the ℓ_0 pseudonorm[21] instead.

We discuss next how to compute sparse solutions to underdetermined systems of equations and how are such considerations applied in the case of compressed sampling.

Sparse solutions to systems of equations

We would like to compute the solution to an underdetermined system of equations. We suppose that the columns of the system matrix have unit ℓ_2 norm. Assuming that the sought solution is sparse, we need to solve the following problem:

$$\text{Find } \hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_0 \quad \text{subject to } \mathbf{Ax} = \mathbf{b}. \tag{2.68}$$

However, this optimization problem is NP-complete, meaning that there is no procedure to find the sparsest solution that is more efficient than an exhaustive search, where we try all possible solutions $\mathbf{x}$. There are therefore several issues here: (i) is the sparse solution exact, (ii) is the sparse solution unique and (iii) can we find the solution faster? As we will discuss next, these issues tend to share answers because the exactness and the uniqueness of the sparse solution are analyzed in the context of finding the solution efficiently. Conversely, finding the sparse solution efficiently is paramount for the practical deployment of such methods.

The solution to the optimization problem (2.68) can be computed efficiently as the solution to the equivalent optimization problem (see also Figure 2.18):

$$\text{Find } \hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_1 \quad \text{subject to } \mathbf{Ax} = \mathbf{b}. \tag{2.69}$$

The main question becomes now under which conditions does this equivalence hold? Along this path, considering that sparse solutions always exist as soon as $k = M$, the conditions will yield also uniqueness and exactness of the sparse solution [60, 61, 57, 26]. These conditions are set on the system matrix $\mathbf{A}$ and usually give the maximal sparsity of the sparse solution that may be obtained, where the sparsity of a vector $\mathbf{x}$ is measured by its ℓ_0 norm, such that when $\mathbf{x}$ is *k-sparse*, we have that $\|\mathbf{x}\|_0 = k$.

There are several sets of such conditions that can be further placed into two groups: (i) conditions over the entries in $\mathbf{A}$ and (ii) conditions mainly related to the structure of $\mathbf{A}$. To the

[21] ℓ_0 is not a norm as it does not fulfill all properties of a norm ($\beta \| \mathbf{x} \|_0 \neq \| \beta\mathbf{x} \|_0$). For ease of notation and understanding we will use the term ℓ_0 *norm* for what actually is the ℓ_0 pseudonorm.

first group belong the *mutual coherence* [60] and the *restricted isometry* [26] properties, while the second group includes the analysis of large underdetermined systems of equations [57] and the theory of convex polytopes [61].

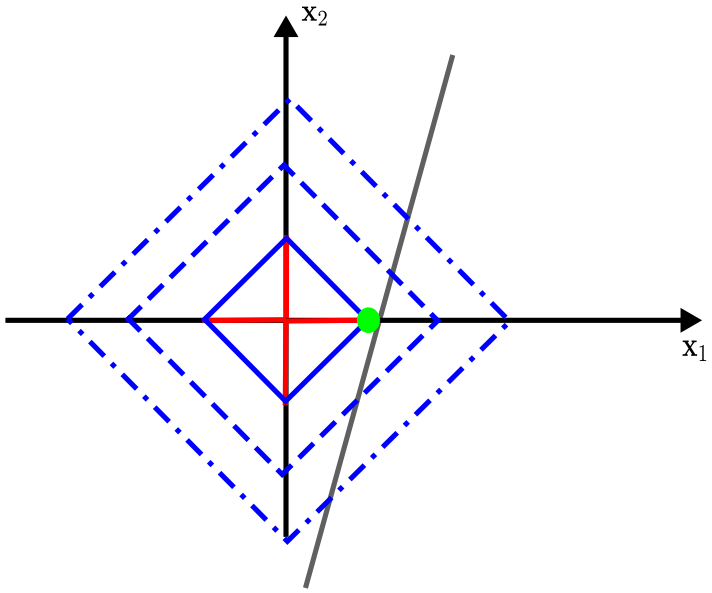


Figure 2.18: Geometrical interpretation of why the minimization of the ℓ_0 norm is usually the same as the minimization of the ℓ_1 norm in a 2D example. The red lines represent points of equal ℓ_0 norm. The blue squares represent points of equal ℓ_1 norm. The green dot is the sought sparse solution $\mathbf{x}_0$ of the optimization problem. The gray line represents the solution line of the underdetermined system of equations $\mathbf{Ax} = \mathbf{b}$, i.e., the constraint line of the optimization problem. As it can be seen, except for the case when the constraint line makes a 45^0 angle with one of the axis, the ℓ_0-based optimization problem and the ℓ_1-based optimization problem have a single same solution. In other words, when the ℓ_1 norm of the solution is minimal, a minimal ℓ_0 norm solution is also obtained. Note that for this example with $k = M = 1$ and $N = 2$, the coherence $\mathcal{C}(\mathbf{A})$ is high, thus the sparse solution is not unique.

In practical applications we have also to compute the solution to the respective optimization problem. In the next paragraphs we dwell into each of these problems in more detail.

Mutual coherence. The *coherence* of $\mathbf{A}$ is defined as $\mathcal{C}(\mathbf{A}) = \max_{i \neq j} |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$, where $\mathbf{a}_i$ $i = 1, \ldots, N$ are the columns of the matrix. When $\mathbf{A}$ has low coherence (typically $<< 0.5$) and $N = 2M$ there exists a unique, exact sparse solution when $k < \mathcal{C}(\mathbf{A})^{-1}$. Furthermore, the ℓ_0 solution can be recovered by ℓ_1 optimization provided the sparsity of the solution is $k \leq \frac{1+\mathcal{C}(\mathbf{A})^{-1}}{2}$ [57].

The mutual coherence provides an upper bound on the sparsity. However, this upper bound is impracticable in some situations. For example, if any two vectors in the matrix $\mathbf{A}$ are perpendicular to each other, this upper bound equals infinity. Conversely, a sparse solution always exist as soon as $k = M$, but this solution is not unique. The *restricted isometry* property yields a smaller bound that can be used under such conditions.

Restricted isometry. A matrix $\mathbf{A}$ possesses the restricted isometry property if for any matrix $\boldsymbol{\Phi}_u$ made out of $u \leq S$ columns of $\mathbf{A}$ we have that:

$$(1-\delta_S)\left\|\mathbf{y}\right\|_2^2 \leq \left\|\boldsymbol{\Phi}_u\mathbf{y}\right\|_2^2 \leq (1+\delta_S)\left\|\mathbf{y}\right\|_2^2,\ \forall\left\|\mathbf{y}\right\|_2<\infty,\ \delta_S>0. \tag{2.70}$$

This means that any u columns of $\mathbf{A}$ are close (as measured by δ_S) to an orthonormal basis, as for such a basis $\boldsymbol{\Phi}_u^T\boldsymbol{\Phi}_u = \mathbf{I}$ and the corresponding transform preserves the norm, such that we may write $\left\|\boldsymbol{\Phi}_u\mathbf{y}\right\|_2^2 = \left\|\mathbf{y}\right\|_2^2$. The restricted isometry constant δ_S (that depends on S, the maximal number of columns that approximately form an orthonormal basis) is the smallest number for which (2.70) holds. For a given u, the value of δ can be approximated as $\delta = 1-\det(\mathbf{Z})$, where $\mathbf{Z} = \boldsymbol{\Phi}_u\boldsymbol{\Phi}_u^T$.

It can be shown [25, 27, 55] that if the matrix $\mathbf{A}$ has a restricted isometry property such that $\delta_{2k} < \sqrt{2}-1$ then the k-sparse solution found by ℓ_1 minimization is with "overwhelming probability" exactly the same as with the ℓ_0 solution. Furthermore the sparse solution is unique for $\delta_{2k} < 1$.

That is equivalent to say that any sparse solution with $k \leq \frac{S}{2}$ (i.e., where the sparsity is less than half of the maximal number of columns of $\mathbf{A}$ that approximately form an orthonormal basis irrespective of which columns are chosen) is unique if $\delta_S < 1$ and can be recovered efficiently if $\delta_S < \sqrt{2}-1$. At the same time, $\delta_S = 1-\det(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_S^T)$ and $\det(\boldsymbol{\Phi}_S\boldsymbol{\Phi}_S^T) \neq 0$ usually only for $S \geq M$. Therefore, when the bounds on δ_S are fulfilled, the bound on the sparsity that arises from the restricted isometry property is $k \leq \frac{M}{2}$.

Thus, assuming any two columns of $\mathbf{A}$ are perpendicular, to find out if k-sparse solutions with $k \leq \frac{M}{2}$ exist and can be computed efficiently, one should verify that $\mathbf{A}$ has the restricted isometry property with $\delta_M < \sqrt{2}-1$.

Sparse enough? In this case we investigate the relationships between the sparsity of the sought solution and the "underdeterminedness" of the corresponding system of equations. Thus, assuming that the system admits a sparse solution, we ask ourselves if the solution is sparse enough such that the ℓ_1 optimization is able to find it. Donoho showed in Reference [57] that

For every system $\mathbf{Ax} = \mathbf{b}$ *allowing a solution with fewer than* ρM *nonzeros,* ℓ_1 *minimization uniquely finds that solution,*

where ρ is a constant such that $\rho(Z) > 0$ with $M < N \leq ZM$.

The first issue in this case is when does a system of equations allow a unique sparse solution? It has been shown [59, 57] that an underdetermined system of equations admits a k-sparse sparse solution if $k < \frac{M}{2}$.

The second issue is when is the unique sparse solution the same as the solution to the optimization problem (2.69) that can be solved efficiently? This happens with "overwhelming probability" when the the solution has at most Σ nonzeros, where $\Sigma \geq \rho M$ [57]. It can be shown empirically that for $Z = 2$, we have that $\rho \approx \frac{3}{10}$. Therefore we recover the unique k-sparse solution of any underdetermined system of equations that has $N \leq 2M$, when $k \leq \frac{3M}{10}$.

Geometric, convex-polytopes based considerations lead to similar bounds on the sparsity, such that we may conclude that as long as the sparsity of the sought solution is just a small fraction of the number of equations in the system, the ℓ_1 solution is equivalent to the ℓ_0 solution and is exact and unique [187].

In this case, the maximal sparsity of the solution that we can compute depends on the number M of lines in the system matrix $\mathbf{A}$ and on how underdetermined the system of equations is, i.e., on the relationship between M and the number of columns N. By comparison, when using the mutual coherence or the restricted isometry to investigate the sparseness in our system, the maximal sparsity of the solution depends on these properties and is usually larger.

Practical considerations. The constrained minimization of the ℓ_1 norm is a convex optimization problem that can be solved efficiently, in comparison to the NP-complete problem of optimizing the ℓ_0 norm. Figure 2.18 illustrates also why this happens. Imagine you are looking for the point of minimal ℓ_0 norm along the gray constraint line, starting somewhere in the upper-right quadrant. As all points in a vicinity of the starting point have the same ℓ_0 norm, you would not know in what direction to look for the minimum, so the best thing you can do is to try all possible solutions. This ambiguity does not exist when using the ℓ_1 norm instead. With respect to Figure 2.18, this becomes evident when looking at where various squares of equal ℓ_1 norm intersect the constraint line. Except for the optimal solution (i.e., the green point), all other solutions (i.e., intersection points) have larger ℓ_1 norms and the ℓ_1 norms of the various solutions become smaller, the closer you are to the optimum.

In practical applications, the optimization problem (2.69) is replaced by [187]

$$\text{Find } \hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \epsilon, \tag{2.71}$$

with ϵ a small constant. This is done because with real, noisy data it may be that the condition $\mathbf{A}\mathbf{x} = \mathbf{b}$ does not hold exactly. Therefore, this stable optimization problem is used instead. It can be shown [25] that (2.71) successfully recovers the same sparse solution as (2.69).

Usually, the optimization problem (2.71) is solved with the help of the Lasso [175]. The Lasso is a regularized least-squares fitting method that includes finding the optimum of $g(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1$, where for our purposes λ is the inverse of the Lagrange multiplier corresponding to the constraint. The Lasso solution is found in a quadratic programming approach, typically with the help of the conjugate gradient method, as $g(\mathbf{x})$ is usually a convex function[22]. When appropriate, other (faster) methods like Least Angle Regression (LARS) [67] may be used as well.

Compressed sampling

One of the most important applications of sparse representations is compressed sampling. In this case we sample a signal below the corresponding Nyquist rate and we would like nevertheless to be able to reconstruct it perfectly. This is possible with "overwhelming probability" if the signal admits a representation where it is sparse.

Usually we have a signal $\mathbf{x} \in \mathbb{R}^N$ that we measure with the help of N linear measurements $\mathbf{y} = \mathbf{B}\mathbf{x}$, where $\mathbf{B}$ is a $N \times N$ matrix. For example for a digital camera with $A \times B = N$ pixels, we have that $\mathbf{B}$ is the identity matrix $\mathbf{I}$. Conversely, in the case of Magnetic Resonance Imaging (MRI), $\mathbf{B}$ is the Fourier transform matrix. Now we would like to reconstruct $\mathbf{x}$ from $M << N$ measurements $\bar{\mathbf{y}} = \mathbf{B}_M\mathbf{x}$, in which case the measurement matrix $\mathbf{B}_M$ will have M lines and N

[22] $\mathbf{A}^T\mathbf{A}$ is always positive semidefinite, and thus the quadratic form $\zeta(\mathbf{x}) = \mathbf{x}\mathbf{A}^T\mathbf{A}\mathbf{x} + \mathbf{c}^T\mathbf{x}$ arising from $g(\mathbf{x})$ is a *convex function* and it has a global minimum. When $\mathbf{A}^T\mathbf{A}$ is positive *definite*, the quadratic form $\zeta(\mathbf{x})$ is strictly convex and it has thus an *unique* global minimum.

columns, everything representing thus an underdetermined system of equations. We could use the sparse representations setup to find $\mathbf{x}$ if we knew that $\mathbf{x}$ is sparse.

The solution in this case is to find a transformed domain where $\mathbf{x}$ is represented by $\mathbf{s}$ that is sufficiently sparse, as discussed above. Thus, we make a few measurements $\bar{\mathbf{y}} = \mathbf{B}_M\mathbf{x}$ in a domain where the signal is not sparse, but because the signal can be represented in a domain where it is sparse as $\mathbf{x} = \mathbf{C}\mathbf{s}$, we have that $\bar{\mathbf{y}} = \mathbf{B}_M\mathbf{C}\mathbf{s}$ or equivalently $\bar{\mathbf{y}} = \mathbf{A}\mathbf{s}$ with $\mathbf{A} = \mathbf{B}_M\mathbf{C}$. We reach therefore the sparse representations setup. The problem is now to find the sparsest solution to the underdetermined system of equations $\bar{\mathbf{y}} = \mathbf{A}\mathbf{s}$, where we know that if the measurement matrix $\mathbf{A}$ respects the restricted isometry property, we recover the exact solution.

Now everything depends on the matrix $\mathbf{A}$ having the restricted isometry property. There are matrices that have this property like, for example, a matrix whose entries are the realizations of a white noise process, which is used for one-pixel cameras. Yet another (better known) example is that of the product between the undersampled Fourier matrix $\mathbf{B}_M$ and the wavelet-transform matrix $\mathbf{C}$ that leads directly to the usage of the sparse representations for compressed sensing in MRI. In this case the sparse representations framework contributes to significantly increasing the speed of the MRI image-acquisition process, as we now need to measure at a smaller number of positions.

2.3.2 The sparse classifier

Sparse-representation based classification bares resemblances with nearest-subspace methods, which in turn stem from k-NN classification. Under these circumstances, the sparse classifier looks for the single sparsest representation of a test vector in terms of a matrix of training vectors. This representation is sparse because it ideally contains only vectors from the class to which the test vector belongs [187]. In practice, the sparse representation will be such that coefficients from the true class (typically the largest coefficients) will help reconstruct the test vector from the training matrix in an optimal way, as measured by the ℓ_2 norm between test vector and its reconstruction. Therefore, the label is assigned to the class whose training-set vectors best reconstruct the test vector, but only if the decision is based on a representation that is sufficiently sparse, as measured by a sparsity concentration index. To be able to properly apply the sparse-representations framework to classification, the dimension of the feature space and the number of examples per class need to be related to each other.

K-NN, nearest subspace and sparse classification

The k-NN algorithm has a set of drawbacks, among which is the poor generalization ability when the training-sample size is small (a problem often encountered in practice). Nearest-subspace methods attempt to solve this problem by inferring from the reduced sample available and under some assumptions, the support of each class subspace in the feature space. The core idea is that all vectors from one class lie in the same subspace of the feature space. Standard nearest-subspace methods make some (very) restrictive assumptions on the geometry of this subspace like: the components of the subspace are distributed along a line [121, 194], or along a curve [142], or each data point lives in the linear span of a few of its nearest neighbors [191]. The distance functions used are defined based on these assumptions. There have been attempts at defining more general distance functions, like for example in Ref. [9], where the distance

between a subspace $\mathcal{S}$ and a point $\mathbf{q}$ is defined as $dist^2(\mathbf{q}\mathcal{S}) = \mathbf{q}^T\mathbf{Z}\mathbf{Z}^T\mathbf{q}$, where $\mathbf{Z}$ contains the basis vectors of the null space of $\mathbf{S}$ that in turn contains the basis vectors of $\mathcal{S}$. However, this is a type of Mahalanobis distance and it thus accurately measures relationships up to second order moments of the class-conditional distributions that in turn implies a Gaussian assumption.

Sparse classification is a type of nearest subspace method. However, the distance function used is the quality of the reconstruction of a test vector by a linear combination of vectors from a class subspace. The coefficients of this reconstruction are computed based on the principle of parsimony. It offers thus a principle-based approach in comparison to the rather heuristic methods employed in other nearest-subspace methods. In the case of sparse classification, we do not bother with the geometry of a class subspace. The geometry of this subspace may indeed be complicated, but the simplifying assumption is that we can infer it from linear combinations of the training-space vectors of that class. In other words any vector in a class can be expressed as a linear combination of other vectors in the same class and hence a new vector will lie in the linear span of the training vectors. The training vectors represent the spanning set for the class subspace. Such considerations have previously been used for classification (an early example being the work of Ullman in Ref. [179]) and have ignited research in the field of appearance-based classification. Sparse classification is usually fast. It has a complexity order $O(t^3 + ndt)$ [63], with t the number of nonzero elements in the sparse vector, n the size of the training set and d the dimension of the training set. In comparison, fast nearest-subspace methods have a complexity order of $O(nd^2)$ [9].

Building on such premises, sparse-representation based classification looks for the sparsest representation of a test vector in terms of a matrix of training vectors. This representation is sparse because it should contain only vectors from the class to which the test vector belongs.

Sparse representations for classification

Our basic assumption to conduct classification using sparse representations is that each class in the shattered feature space has its own linear span. In the case of the sparse classifier, given a test vector we search for its spanning set. Assuming further that the spanning set of each class is present in the training set, we assign a new vector to the class whose spanning set best explains it.

Let the training matrix be denoted by $\mathbf{T} = [\mathbf{T}_1, \ldots, \mathbf{T}_k]$, containing the class-submatrices $\mathbf{T}_i = [\mathbf{v}_{i,1}, \ldots, \mathbf{v}_{i,N_i}]$ with $i = 1, \ldots, k$, where N_i is the number of vectors in class i and k the number of classes. The total number of vectors in $\mathbf{T}$ is $N = \sum_{i=1}^{k} N_i$ and each vector $\mathbf{v}_j, j = 1, \ldots, N$ has M entries. Then, for each new vector $\mathbf{y}$ we ideally have

$$\begin{aligned} \mathbf{y} &= \mathbf{T}\mathbf{x} \\ &= x_{i,1}\mathbf{v}_{i,1} + \cdots + x_{i,N_i}\mathbf{v}_{i,N_i}, \end{aligned} \tag{2.72}$$

where the coefficient vector $\mathbf{x} = [x_1, \ldots, x_N]^T$ has entries $x_j := x_{i,l}, l = 1, \ldots, N_i$, different from zero only for the training-space vectors from the class i to which $\mathbf{y}$ belongs. This system of equations is usually underdetermined with the number N of vectors in the training space being well larger than the dimension M of the vectors. Thus, there are infinitely many solutions to (2.72), and the ones that interest us are the ones with a sparse representation $\mathbf{x}$. An illustration of this relationships is shown in Figure 2.19. The sparse vector $\mathbf{x}$ depicted there corresponds to a classification problem with three classes. The first class ω_1 has four vectors in the training set, ω_2 has three and ω_3 has five.

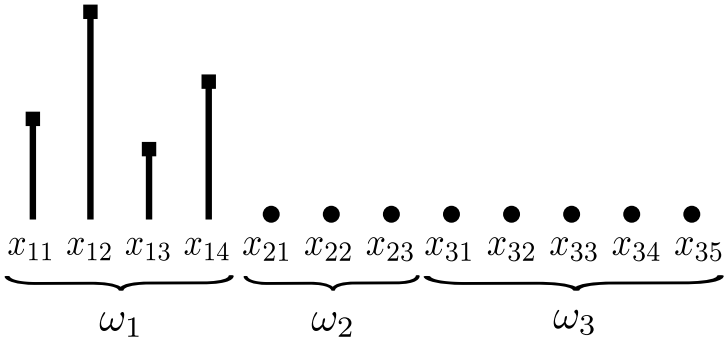


Figure 2.19: We assume a three-class classification problem with 12 vectors in the training set and shown in this figure: (i) the sparse vector $\mathbf{x} = [x_1, \ldots, x_{12}]^T$ with entries $x_j \equiv x_{i,l}, l = 1, \ldots, N_i$, $i = 1, 2, 3$, $j = 1, \ldots, 12$, where $N_1 = 4$, $N_2 = 3$, $N_3 = 5$; and (ii) the class regions corresponding to the the three classes ω_1, ω_2 and ω_3. The hight of the bars is related to the value of the corresponding entry in the sparse vector. A dot means that the corresponding entry has the value zero.

To better understand why we are interested in a sparse $\mathbf{x}$, assume for example equal number of training vectors per class. Then, as $\mathbf{x}$ should select only vectors from one class, the more classes we have in our classification problem, the sparser $\mathbf{x}$ becomes. Therefore, we do not search for some $\hat{\mathbf{x}} \in \mathbb{R}^N$, but for the *sparsest* vector that solves equation (2.72). As previously discussed, we can find this $\hat{\mathbf{x}}$ by optimizing over the ℓ_0 pseudo norm, solving:

$$\hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_0 \quad \text{subject to } \mathbf{T}\mathbf{x} = \mathbf{y}. \tag{2.73}$$

Ideally, assuming that a vector $\mathbf{y}$ is represented solely with the training vectors from the correct class (that is, the coefficients in $\mathbf{x}$ for training vectors from other classes are all zero), the vector $\mathbf{y}$ can be classified by looking up to which class the nonzero entries in $\mathbf{x}$ belong.

In practice, of course, again, questions about the required amount of sparsity and the exactness/uniqueness of the sparsest solution arise. As previously discussed, if some $\mathbf{x}$ with less than $\frac{M}{2}$ nonzero entries verifies $\mathbf{y} = \mathbf{T}\mathbf{x}$, then this is the unique sparsest solution. This means that we have a good chance of finding the correct and unique sparsest solution to (2.72) even for two-class problems or for configurations in which the number of training vectors per class is not the same over all classes, provided we have enough vectors in the training set in relation to the dimension of the corresponding feature space. The dimension of the feature space must be $M \geq 2 \cdot c$, with c the largest number of training vectors per class in the training set. These relationships are discussed in more detail later in this section.

The sparse classifier is not directly appropriate for one-class classification problems, as in this case, $\mathbf{x}$ will no longer be sparse.

Assigning a class label

In practice, because the solution to equation (2.73) is computationally difficult to find, and we work with real data, we solve instead:

$$\hat{\mathbf{x}} = \arg\min \| \mathbf{x} \|_1 \quad \text{subject to } \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \epsilon. \tag{2.74}$$

As discussed before, minimizing over the ℓ^1 norm instead of the ℓ^0 norm yields the same unique solution if $\mathbf{x}$ is sparse enough and $\mathbf{T}$ has certain properties.

The vector $\mathbf{x}$ found after we solve (2.74) will usually have the largest entries for one class only, and small non-zero entries for other classes as well. Next, we use the following notation: $1_i = [b_1, \ldots, b_n]^T$, $b_l \in \{0, 1\}$, $l = 1, \ldots, n$ is the selection vector for class i, whose entries are everywhere zero, except for the positions of the columns of $\mathbf{T}$ that contain the training vectors of class i, where they are one and $\mathbf{v}_1 \odot \mathbf{v}_2$ is the component-wise product of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Thus, $1_i \odot \mathbf{x}$ selects the entries of $\mathbf{x}$ where the coefficients of class i reside. $C(\mathbf{y})$, with $C : \mathbb{R}^m \to \{1, \ldots, k\}$ is the function that assigns a class label to the vector $\mathbf{y}$.

A classification rule that harnesses the class-subspace structure has been proposed by [187]. It assigns the test vector $\mathbf{y}$ to the class whose training-set vectors best reproduce $\mathbf{y}$. The decision rule reads:

$$C(\mathbf{y}) = \arg\min_i \parallel \mathbf{y} - \mathbf{T}(1_i \odot \hat{\mathbf{x}}) \parallel_2 . \tag{2.75}$$

The sparse classifier looks for the spanning set that best represents the test vector and assigns the test vector the label of the class corresponding to this spanning set. It is not guaranteed that the spanning set of each class is present among the training-set vectors. Nevertheless, when all conditions are verified the sparse representations framework will return a unique spanning set for the test vector and the components of this spanning set will be from among the vectors in the training set. At this point we can only assume that there is an intersection between the found linear span and the sought one. Under such circumstances we assign the vector to the class where this intersection is the largest, as measured in equation (2.75) by the ℓ_2 norm of the difference between the test vector and its respective reconstruction.

The sparsity concentration index and the "unsure" decision

If for a certain test vector, the nonzeros of the corresponding coefficient vector are not concentrated over a single class, then the confidence in the sparse-classifier result is small. This may happen for example if we attempt to assign a label to a vector whose true class has no representatives in the training set or if some components of the training set have been wrongly labeled. If a test vector cannot be assigned with sufficient confidence to any of the classes represented in $\mathbf{T}$, then it receives the label "unsure". In order to express such a confidence, for the decision rule in (2.75), a *sparsity concentration index* is defined as

$$SCI\ (\mathbf{x}) = \frac{l \max\limits_i \left(\frac{\parallel 1_i \odot \mathbf{x} \parallel_1}{\parallel \mathbf{x} \parallel_1} \right) - 1}{l - 1}. \tag{2.76}$$

The vector is labeled "unsure" when $SCI\ (\mathbf{x}) \leq \tau$. The parameters l and τ need to be set empirically.

The dimension of the feature space and the number of examples per class

A question of major importance for the practical deployment of the sparse classifier is: what is the optimal dimension M of the feature space? The answer to this question involves the maximal number of examples per class that gives the maximal theoretical ℓ_0 norm of the sparse vector $\mathbf{x}$ under perfect conditions (i.e., a query vector can be represented without error by a combination of training vectors from that one class).

As previously discussed, with c the largest number of examples per class[23] we have that $M \geq c \cdot 2$, such that $\mathbf{x}$ is the unique sparsest solution and $M \geq c \cdot \varrho$, usually with $\varrho > 2$ such that we can compute this solution by solving an ℓ_1 optimization problem.

To obtain a classification-related intuition behind this statement, imagine you would like to label a test vector in a feature space with a training set such that $c > M$. According to equation (2.72) a test vector will be exactly reconstructed under ideal conditions by a linear combination of training-set vectors from the same class. To conduct classification we need to find this one sparse representation. However, in our case there will not be just one but several such representations, each equally suitable, as any M non-degenerate training-set vectors of the same class are enough to exactly reconstruct our test vector. Conducting classification in the sparse representations setup in this case is difficult or even impossible, as we no longer have a convex optimization problem. There are not one but several sparse coefficients vectors that solve our problem. Conversely, when $c < M$ (more precisely when $c \leq \frac{M}{2}$) we have just one combination of training-set vectors from the same class that exactly reconstructs the test vector, and accordingly we have just one $\mathbf{x}$ that solves our problem.

At the same time, the training matrix $\mathbf{T}$ usually has to be underdetermined[24] such that $N > M$. These relationships provide the support for the various bounds on the dimension of the feature space in relation to the number of examples per class needed for the sparse classifier to work properly. One such bound is [187]

$$c < \left\lfloor \frac{M+1}{3} \right\rfloor, \tag{2.77}$$

with $\lfloor x \rfloor$ the *floor* function that returns the smallest integer less than or equal to x.

Yet another bound is [62]

$$D \geq 2 \cdot c \log\left(\frac{N}{D}\right),$$

assuming $c \ll N$ and with D the size of the feature space after applying a feature extraction transform. It can be shown that the sparse classification framework works for *any* D linear measurements. This last bound is particularly interesting, as it implies that theoretically, it is not the feature extraction process, but the dimension of the reached feature space that is more important for classification with a sparse classifier. Practice shows that the sparse classifier is indeed less sensible to the choice of the feature extraction process than other classifiers, at comparable accuracies [187].

[23] Clearly, c is *the* number of examples per class, if each class has the same number of examples.

[24] Empirical evidence (see Section 5.1.3) shows that the classifier works even for overdetermined systems, as long as $\mathbf{T}$ is rank deficient and $\mathbf{x}$ is sparse such that the rank $R \geq c \cdot 2$.

Chapter 3

Gaussian nonlinear feature extraction

The Gaussian assumption refers to the use of a Gaussian model to describe a random variable. This assumption is often made in practice, being not only intuitive (as discussed in Section 1.1.1) but also justified by the Central Limit Theorem (CLT) [97]. In its classical form, the CLT states that the mean of N independent identically distributed random variables, with finite mean and variance, has a limiting distribution for $N \to \infty$ that is Gaussian. Random variables of practical interest are measurements of real-world processes being thus available only as the result of a combination of many unobserved random influences. In the framework generated by the CLT, this combination is considered a summation, and therefore the target random variable is assumed Gaussian.

The Gaussian distribution has a set of appealing characteristics related to its mathematical tractability, like, e.g., the equivalence of decorrelation and independence or the fact that all cumulants of an order larger than two are zero. These properties make inference and reasoning more easy in Gaussian environments. Therefore, the Gaussian assumption is made actually more often than needed, and a lot of effort has been put into developing methods optimally suited to such environments.

The main problem is, however, that the Gaussian assumption does not always hold. Rather than ignoring this or rethinking everything, we can try to adapt to the Gaussian setup by Gaussianizing the data. Gaussianization has already been discussed for purposes such as density estimation [33], independent component analysis [127], blind source separation [56], speaker adaptation [157], adaptive filtering [111], and system identification [140]. However, all these methods are "holistic" in the sense that they Gaussianize the entire input data irrespective of the class label, thus destroying separability and being unsuited for pattern recognition. Here the accent lies on Gaussianization for pattern recognition applications.

A nonlinear transform is proposed, such that the class-conditional densities are Gaussian in the transformed space, and thus the pdf of the data in the transformed space is a Gaussian mixture model (GMM). For the purposes followed here, a difference is made in Section 3.1 between factorial and non-factorial distributions, only to concentrate then in the rest of the chapter on the non-factorial case.

Any invertible and differentiable transformation $\mathbf{y} = G(\mathbf{x})$ modifies the statistical properties of the input data according to the well-known formula [19] [101]

$$p(\mathbf{y}) = p(\mathbf{x}) \frac{1}{\left| |G'(\mathbf{x})| \right|} \quad \forall \mathbf{x} = G^{-1}(\mathbf{y}), \tag{3.1}$$

where $|G'(\mathbf{x})|$ is the determinant of the Jacobian matrix of the transform. Linear transforms have the advantage of mathematical tractability, but by their inherent constraint they cannot achieve the desired Gaussianization. As equation (3.1) shows, they can at most scale the original density. It is thus clear that only a nonlinear transform can achieve the desired Gaussianization. Nevertheless, there are linear transforms inherently related to Gaussianity that are often used in practice. As already discussed in Section 1.1.1, there is a strong relationship between the Gaussian assumption and both PCA and LDA.

A nonlinear transform has virtually complete control over the input data. The challenge in our case is to compute the parameters of this transform such that the original information available in the data and which allows us to shatter the feature space is still present after applying the transform. In the non-factorial case, as shown in Section 3.2, this is achieved by introducing an elastic constraint (i.e., regularizer) in the nonlinear transform.

The Gaussianization works in a supervised manner in the sense that it needs a labeled training set to compute its parameters. The corresponding elastic transform represents actually a displacement field that describes the way the data (as present in the feature-space sample from the training set) should be redistributed such as to become Gaussian. In the case of the standard Gaussianization, this displacement field is defined over a grid with a constant distance between grid points. The difficulty in this case is the computational complexity of the used elastic transform. The complexity increases with the dimension of the input space in such a way that it becomes prohibitive even for relatively moderate dimensions of e.g. 15. With this *fixed grid*, computations are spent for positions in the feature space where no training data is present. Assuming that the training space properly samples the feature space, this is counterproductive. To decrease the computational burden, and thus push the dimensionality limit up, the accelerated Gaussianization that uses an *adaptive grid* is introduced in Section 3.3. The distance between grid points in an adaptive grid is variable. The adaptive grid is defined in such a way that in the regions where the data is sparse, the density of grid points is lower than that of the fixed grid, while in other regions the density of grid points remains constant in comparison to the fixed grid. As a consequence, the total number of grid points decreases, thus decreasing the computational complexity, while still ensuring a proper link to the feature space.

In Section 3.4 the validity of this approach is demonstrated experimentally on both synthetic and real data. Also, various implementation aspects of the nonlinear Gaussianization are discussed. Finally, Section 3.5 contains a critical review of the method and points at the issues still open.

3.1 Factorial and non-factorial distributions

The pdf of a multidimensional random variable can be either factorial or non-factorial. A factorial multivariate pdf has the property that it can be factorized into independent components. We may therefore write:

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_N) = p(x_1)p(x_2)\cdots p(x_N).$$

Under such circumstances, multivariate Gaussianization is actually a set of univariate Gaussanizations, but only if we can obtain the independent components. Conversely, if the distribution is not factorial, or if the independent components cannot be found, we need to resort to other methods.

In the holistic setup, a more widely used non-factorial method is the iterative Gaussianization [118] that is strongly linked to projection pursuit density estimation [75]. The iterative Gaussianization was discussed and shown to converge only for a Gaussian distribution with zero mean and unit variance.

The main topic here is Gaussianization for pattern recognition, i.e., in a multiclass, non holistic setup, but still, the dichotomy between factorial and non factorial distributions is valid. Next we are going to discuss multiclass Gaussianization for factorial distributions in Section 3.1.1 and introduce the multiclass Gaussianization for non-factorial distributions, which represents the main topic of this chapter, in Section 3.1.2.

3.1.1 Gaussianization for factorial distributions

When the observed random variable can be factorized, the sought non-holistic multivariate Gaussianization becomes a set of 1D Gaussianizations, each of them finding the nonlinear function that can transform the corresponding 1D density to a GMM with as many modes as the number of classes in the feature space.

For univariate multiclass Gaussianization, the task is to find a function(transformation) $g(x)$ such that the pdf of the transformed viable y becomes $p(y) = \gamma(y)$, where $\gamma(y) = \sum_{i=1}^{L} P_i \phi_i(y)$ is a weighted sum of Gaussians, with L the number of classes. The weights are given by the a-priori probabilities P_i of each class. According to equation (3.1) we then need to solve:

$$p(x) = \gamma\left(g(x)\right) \cdot \left|g'(x)\right| .$$

The solution of this equation is:

$$g(x) = \Gamma^{-1}\left(F(x)\right) ,$$

where $\Gamma(\cdot)$ is the cumulative distribution function (CDF) of y and $F(\cdot)$ the CDF of x[1]. Usually $p(x)$ is in turn approximated with a GMM. A bimodal example is shown in Figure 3.1.

There are several difficulties related to such an approach. To make these more clear, a productive model is established. According to this model, the data was originally a collection of independent variables (naïve Bayes) such that for each variable, each class was Gauss distributed. Each variable was then transformed non-linearly and then the results were mixed to produce the observed distribution, which is sampled in the training set. The number of random variables (i.e., the dimension of the corresponding vectorial random variable) was preserved by the transformation. Such a model, with a linear mixing step, is illustrated in Figure 3.2. In this example the classes are separable along each axis.

Now the difficulties related to such an approach become apparent: first, due to the nonlinearity, we will be able – under optimal circumstances – to recover the GMM up to a projection, as the true means and variances cannot be recovered anymore (see Figure 3.3); second, we need to correctly unmix the components and third, the class information is only implicitly taken into account. Being unable to recover the true means and variances is less important for patterrecognition purposes, however, class information and properly unmixing the components is paramount to success. Extensive research has been dedicated to the latter problem within the framework of ICA and solutions exist for both linear and nonlinear mixing of the components.

[1]This gives us a way to generate a sample form any disitribution for which we can compute $\Gamma^{-1}(\cdot)$. For this purpose we need only to generate a sample according to the uniform distribution and then to transform it accordingly.

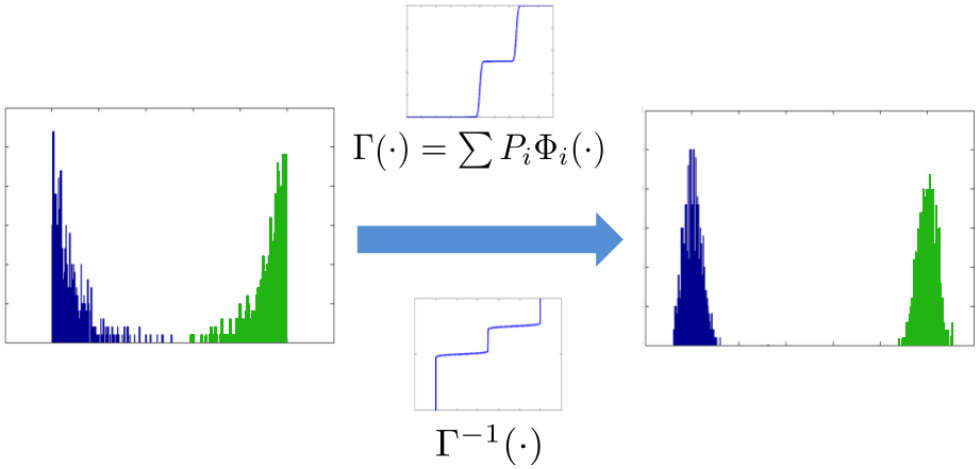


Figure 3.1: 1D Gaussianization for a feature space with two classes. The inverse Γ^{-1} is computed by reflecting Γ over the first diagonal.

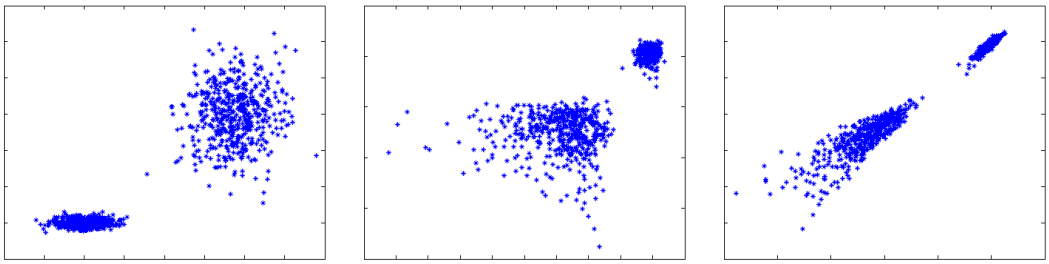


Figure 3.2: Separable Gaussian 2D data (a), nonlinearly transformed (b) and linearly mixed (c).

However, nonlinear ICA is somewhat limited with respect to the type of nonlinearity that can be unmixed and is therefore less general. As previously discussed in Section 1.1.2, even linear ICA has to be used with care, as various algorithms are available, some of them concentrating too strongly on lower order moments of the analyzed distributions. The class information is considered here only over a sum (see Figure 3.1) and as such it is not explicitly taken into account, with negative consequences. We look for a way to fit L modes with certain priors on our data, but we do not enforce that each mode should be responsible for data points with the same label. Should the classes overlap we will get a GMM but there is no guarantee that it will have one mode per class. It can happen that each Gaussian mode will include data-points with

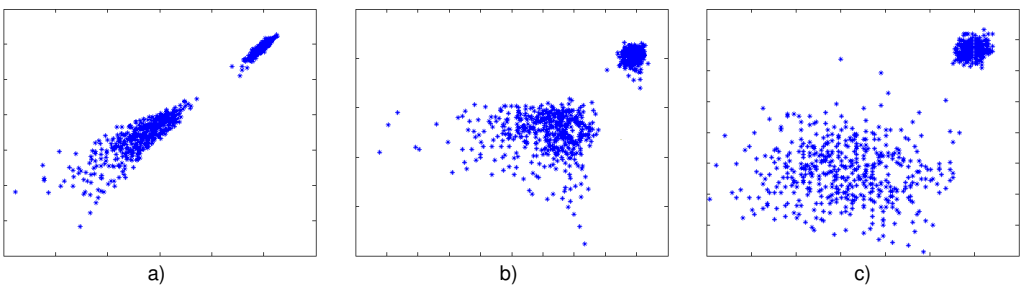


Figure 3.3: Observed data (a), linearly unmixed with ICA (b) and Gaussianized (c).

different class label. Therefore, this type of Gaussianization is rather unsuited for practical applications because, even if the original distribution is factorial, it is difficult to properly unmix the components and link Gaussian modes to classes.

3.1.2 Gaussianization for non-factorial distributions

The reminder of this chapter is dedicated to describing a Gaussianization method that works also for the case when no independent components are available and that is not holistic. At the core of this approach is an elastic transform between nonparametric and parametric pdf estimates of a training sample. Free of any type of productive model, we will concentrate on making the input data as Gaussian as possible, with as few assumptions as possible. In a way it may be said that we work with a *descriptive model* as the accent lies now on gaussianizing the data given the training set, irrespective of how this data was generated, e.g., even if it was not Gaussian to begin with. This approach will be demonstrated on both real and synthetic pattern-recognition problems.

3.2 Elastic transform-based multiclass Gaussianization

In the supervised multiclass Gaussianization proposed here, the pdf of the available labeled training data is first estimated nonparametrically, then parametrically as a GMM with one component per class, as described in Section 3.2.1. Then, an elastic transform is computed such that the nonparametric estimate is "morphed" on the GMM, minimizing the sum of squared differences (SSD) between the two functions. The displacement field corresponding to the elastic transform defines the way the input data should be modified such that its distribution is a GMM. Clearly the displacement field is properly defined only over a region Ω close to the support of the training sample. We extend this to $\mathbb{R}^N$ by means of the identity transform, such that data points outside this support remain unchanged. The elastic transform is discussed in Section 3.2.2.

In contrast to other Gaussianization methods, pdf-distance measures inspired from the performance analysis of kernel density estimators [183] are used here. However, no explicit precaution to conserve probability is taken – or in the elastic-transform terminology, mass conservation issues in the sense of Monge-Kantorovich [154] are ignored – even if the obtained density is properly scaled (i.e., such that it integrates to one over $\mathbb{R}^N$). This strategy keeps the computational burden at a level acceptable for practical applications.

Probability conservation issues. As shown in equation (3.1), the density in the transformed space depends also on the Jacobian of the used transformation, which follows from the *probability conservation constraint*. The probability conservation constraint takes care that the probability of a certain event remains constant even if the way the event is expressed changes. These issues will be discussed next with the help of a scalar random variable, the extension to vectorial random variables being straightforward.

Next, it is assumed that there exist two random variables x and y that are related by an injective function $g(\cdot)$ such that $y = f(x)$ and $f^{-1}(x)$, the inverse of $f(\cdot)$ exists. Now, in light of the relationship among the random variables, we would like to find the relationship among their densities. As $f(\cdot)$ is an injective function, we start from the observation that, by

the probability conservation constraint $P\left\{x \in \left[\mathfrak{x} - \frac{d\mathfrak{x}}{2}, \mathfrak{x} + \frac{d\mathfrak{x}}{2}\right]\right\} = P\left\{y \in \left[\mathfrak{y} - \frac{d\mathfrak{y}}{2}, \mathfrak{y} + \frac{d\mathfrak{y}}{2}\right]\right\}$. In other words, the probability of x taking values in the interval

$$\mathfrak{x} - \frac{d\mathfrak{x}}{2} < x \leq \mathfrak{x} + \frac{d\mathfrak{x}}{2}$$

equals the probability of y taking values in the interval

$$\mathfrak{y} - \frac{d\mathfrak{y}}{2} < y \leq \mathfrak{y} + \frac{d\mathfrak{y}}{2}$$

We may thus write

$$|p_x(\mathfrak{x})d\mathfrak{x}| = |p_y(\mathfrak{y})d\mathfrak{y}|$$

from which it follows that

$$\begin{aligned} p_y(\mathfrak{y}) =& p_x(\mathfrak{x}) \left|\frac{d\mathfrak{x}}{d\mathfrak{y}}\right| \\ =& p_x(\mathfrak{x}) \frac{1}{\left|\frac{d\mathfrak{y}}{d\mathfrak{x}}\right|} \\ =& p_x(\mathfrak{x}) \frac{1}{|f'(\mathfrak{x})|}, \end{aligned}$$

with $\mathfrak{x} = f^{-1}(\mathfrak{y})$. As it can be noticed, to conserve the probability between the original and the transformed space, the derivative of the function $f(\cdot)$ needs to be handled. Equivalently, for vectorial random variables, the determinant of the Jacobian matrix of the transform needs to be handled. In our case, such a course of action would strongly increase the computational complexity of the transform, thus severely hampering the practical usability of the method. Therefore, the probability conservation constraint is ignored here.

3.2.1 Estimating the probability density function

The pdf is estimated from the available sample (see Section 2.1.1) both nonparametrically and parametrically under the Gaussian assumption. This procedure works irrespective of the number of classes. An example is shown in Figure 3.4. In a multiclass scenario, the nonparametric estimate is carried out for the entire training set, ignoring the class labels, whereas the parametric estimate considers the class information.

Nonparametric estimation

For nonparametric estimation the Parzen estimation procedure is used. The estimate $\tilde{p}_O(\mathbf{x})$ is computed using the N vectors of the training sample with the help of equation (2.17). With $\gamma(\cdot)$ the Gaussian isotropic kernel and m the size of the feature space $\tilde{p}_O(\mathbf{x})$ can be rewritten according to equation (2.18) as:

$$\tilde{p}_O(\mathbf{x}) = \frac{1}{N \cdot h^m} \sum_{i=1}^{N} \gamma\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right).$$

Now we need to establish the bandwidth, which, with the kernel choice from before, is just a

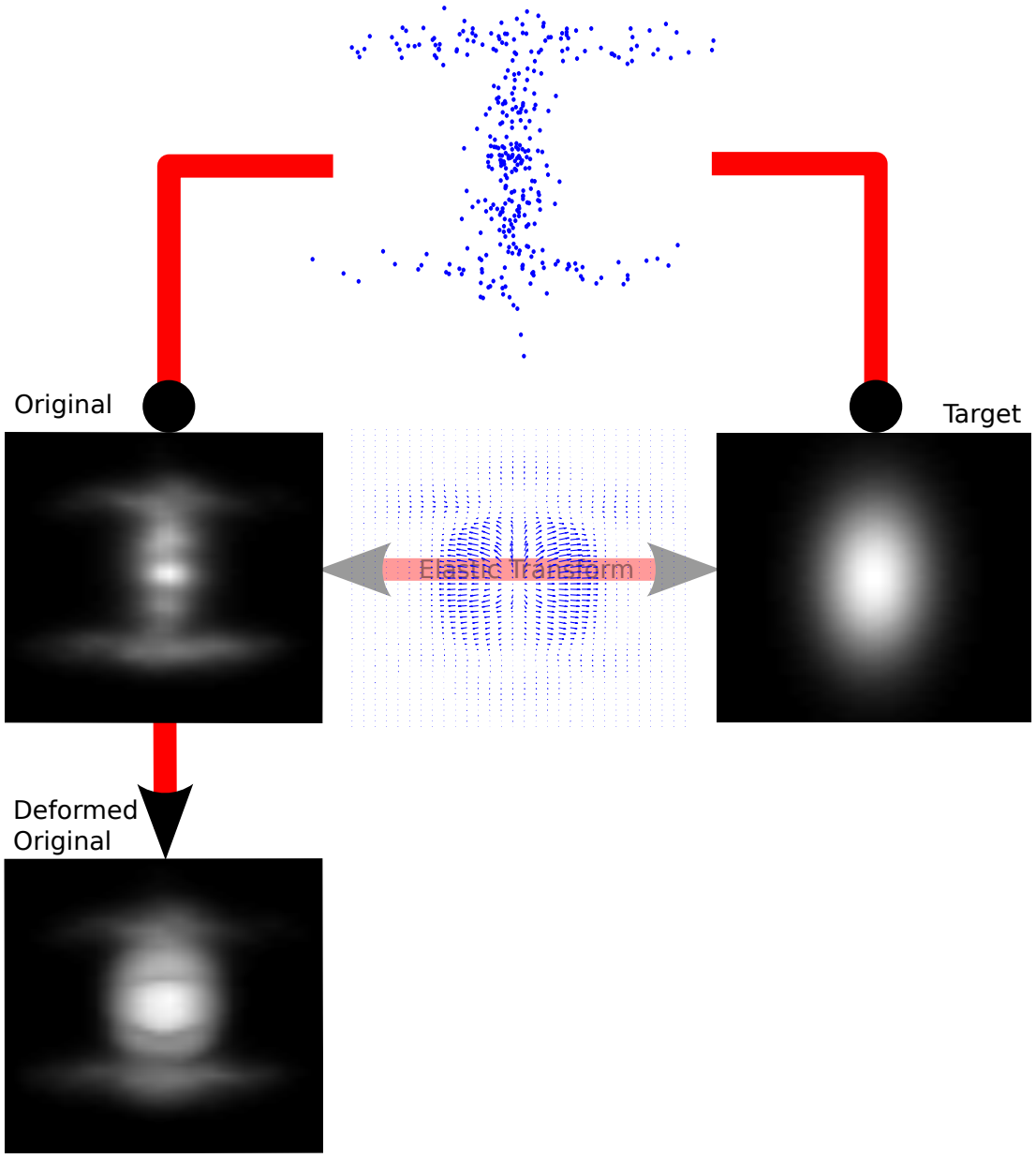


Figure 3.4: Example of non-Gaussian 2D data. The Original (nonparametric pdf estimation) and the Target (Gaussian, parametric pdf estimation) are computed. The displacement field of the elastic transform (defined over the fixed transform grid) is also shown as well as the deformed template.

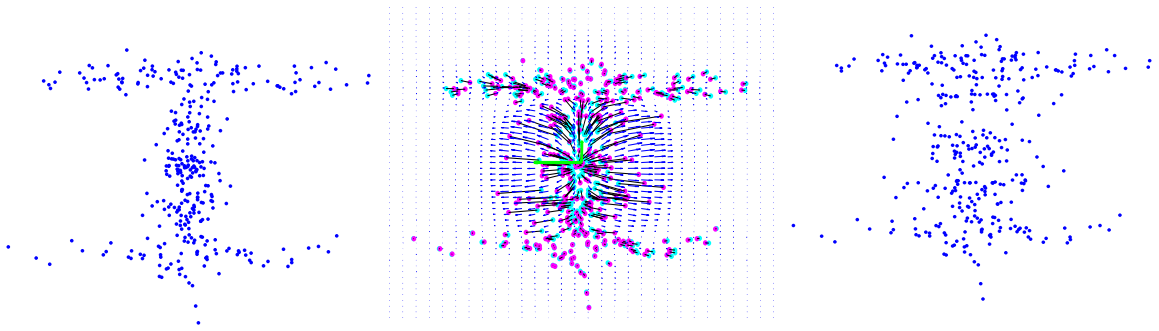


Figure 3.5: Shown here are first the original non-Gaussian data, then the transform grid defined over the target region, with initial positions of the data points (cyan) and final position after applying the transform (magenta). The displacement vectors are also marked (black) as well as the principal components (light green). Finally the transformed data is shown.

scalar value. As discussed in Section 2.1.1, for scalar random variables, the bandwidth parameter h – which is again a scalar value – is computed (see Equation (2.15)) as

$$h = \left[\frac{8 \cdot \sqrt{\pi} \cdot R(\gamma)}{3 \cdot \mu_2^2(\gamma) \cdot N}\right]^{\frac{1}{5}} \hat{\sigma} \tag{3.2}$$

with $\hat{\sigma}$ a parametric estimate under the Gaussian assumption. In Reference [183] it is shown that a better choice for $\hat{\sigma}$ is

$$\hat{\sigma} = \frac{SIQR}{\Phi^{-1}\left(\frac{3}{4}\right) - \Phi^{-1}\left(\frac{1}{4}\right)} \tag{3.3}$$

with $SIQR$ the *Sample Interquartile Range*, and $\Phi^{-1}(p) = \sqrt{2} \cdot \mathrm{erf}^{-1}(2 \cdot p - 1)$ for $p \in (0, 1)$ the quantile function of the univariate Gaussian distribution, a choice that minimizes the estimator error for kernels with fixed bandwidth. Conversely, for multivariate kernel based estimates a good choice for the bandwidth of a Gaussian anisotropic kernel depends on the standard deviation of each component, as shown in Equation (2.19). Here, with Gaussian *isotropic* kernel the parameter h is computed as described in equation (3.2) and $\hat{\sigma}$ is approximated with the help of equation (3.3) computed for the component of $\mathbf{z}$ with the largest variance. $R(\gamma)$ is defined as $R(\gamma) = \int \gamma^2(\mathbf{z}) d\mathbf{z}$ and $\mu_2(\gamma) = \int \|\mathbf{z}\|^2 \cdot \gamma(\mathbf{z}) d\mathbf{z}$.

Parametric estimation

The parametric estimate for the multiclass case is computed as

$$\tilde{p}_T(\mathbf{x}) = \sum_{l=1}^{L} P_l \cdot p(\mathbf{x}|\omega_l),$$

where L is the number of classes. The values $P_l = N_l/N$, with N_l being the number of training vectors in class ω_l, are the a priori probabilities of the classes, and the class-conditional likelihoods $p(\mathbf{x}|\omega_l)$ are multivariate Gaussians. We use the maximum-likelihood estimate $\mu_l = \frac{1}{N_l} \sum_{k=1}^{N_l} \mathbf{x}_k^l$ and the unbiased estimate $\mathbf{\Sigma_l} = \frac{1}{N_l - 1} \sum_{k=1}^{N_l} (\mathbf{x}_k^l - \boldsymbol{\mu}_l)(\mathbf{x}_k^l - \boldsymbol{\mu}_l)^T$ for the class means and covariance matrices.

3.2.2 Transformation

The nonlinear Gaussianization transform is computed in a variational approach from the elastic transform [141] that modifies the nonparametric pdf estimate such that it becomes as similar as possible to the Gaussian parametric one (see Figure 3.4). As discussed before, probability conservation effects are ignored to keep the computational complexity manageable. The computation of the elastic transform relies on the pdf estimation. The pdf is estimated at the knots of a grid laid over the support of the training sample in the feature space (i.e., a discretization of Ω).

Elastic transform

Next the nonparametric estimate $p_O(\mathbf{x})$ will be denoted with $O(\mathbf{x})$ (for original) and the parametric pdf estimate $p_T(\mathbf{x})$ with $T(\mathbf{x})$ (for target). The sought transformation is $\phi : \mathbb{R}^d \to \mathbb{R}^d$, with d the dimension of the input. The transformed original $O(\phi(\mathbf{x}))$ is as similar as possible to the target $T(\mathbf{x})$. The transformation $\phi = \mathbf{x} - u(\mathbf{x})$ has two parts: the identity $\mathbf{x}$, and the displacement $u(\mathbf{x})$.

Given T and O the displacement u is sought such that

$$\mathcal{I}[u] = \mathcal{D}[T, O; u] + \alpha \mathcal{S}[u] \to \min, \tag{3.4}$$

where $\mathcal{D}[T, O; u]$ is the distance between T and O with respect to u, $\mathcal{S}[u]$ is a regularizing term and α is a positive real constant. α controls the influence of the data term on the solution of the optimization problem in relation to the constraint. Clearly, the higher α the less influence the data term has. A usual value for α is $\alpha = 1$. The distance measure used here is the SSD, computed as

$$\mathcal{D}[T, O; u] = \frac{1}{2} \| O(\phi(\mathbf{x})) - T \|^2_{\ell_2(\Omega)},$$

with Ω being the region under consideration. The linearized elastic potential

$$\mathcal{S}[u] = \int_\Omega \frac{\mu}{4} \sum_{j,k=1}^{d} (\partial_{x_j} u_k + \partial_{x_k} u_j)^2 + \frac{\lambda}{2} (\text{div } u)^2 d\mathbf{x},$$

with λ and μ being the Lamé constants [141] is used as regularizing term. These constants control the rigidity of the elastic deformation, such that high values lead to increased rigidity. The linearized elastic potential takes care that the solution remains smooth.

The solution of the optimization problem (3.4) is obtained by numerically solving the corresponding Euler-Lagrange equations

$$f = \mu \triangle u + (\lambda + \mu) \nabla \text{div}(u), \tag{3.5}$$

with[2] f the force term, which is related to the distance measure $\mathcal{D}$ and depends thus also on the displacement u. For this purpose equation (3.5) is rewritten in the form

$$f(u) = \mathcal{A}[u], \tag{3.6}$$

[2] $\triangle$ is the Laplace operator, ∇ is the gradient operator and div is the divergence.

with $\mathcal{A}[u] = \mu\triangle u + (\lambda + \mu)\nabla\text{div}(u)$ a partial differential operator related to the regularizer $\mathcal{S}$. To solve this, a fixed-point iteration scheme is used

$$\mathcal{A}[u^{k+1}](\mathbf{x}) = f(\mathbf{x}, u^k(\mathbf{x})), \tag{3.7}$$

with $\mathcal{A}[u^{k+1}](\mathbf{x}) = \mathcal{A}[u^{k+1}(\mathbf{x})]$, $\mathbf{x} \in \Omega$ and $k \in \mathbb{N}$. For stability purposes the displacement u may be made time dependent and the sought minimizer is obtained as the steady-state solution of the corresponding time-dependent partial-differential equation[3] [141]:

$$\begin{aligned} \frac{\partial}{\partial t}u = f(u) - \mathcal{A}[u] \Leftrightarrow \\ \frac{u^{k+1}(\mathbf{x}) - u^k(\mathbf{x})}{\rho} = f(\mathbf{x}, u^k(\mathbf{x})) - \mathcal{A}[u^k](\mathbf{x}) \Leftrightarrow \\ u^{k+1}(\mathbf{x}) = u^k(\mathbf{x}) + \rho\left(f(\mathbf{x}, u^k(\mathbf{x})) - \mathcal{A}[u^k](\mathbf{x})\right) \end{aligned}$$

The transform thus obtained is diffeomorphic, for appropriate values[4] of the Lamé constants.

Transform grid

To ensure numerical tractability, the elastic transform is computed over a discretization[5] of the target region Ω. This discretization is called the *transform grid*. In practice the grid is in the form of a hyperrectangle whose sides are double the standard deviations of the training sample in the corresponding directions. The grid is centered on the training sample and it thus spreads over the margins of the sample.

As previously discussed, the grid is fixed for the standard Gaussianization and adaptive for the accelerated Gaussianization. A fixed grid with displacement vectors is shown in Figure 3.4 and superimposed on the training data in Figure 3.5. An adaptive grid is shown in Figure 3.6. The gird is defined at integer positions, the displacement u is computed at non integer positions with the help of interpolation, like for example bilinear interpolation.

For the standard Gaussianization, the size δ of the grid (i.e., the distance between two grid points along an axis of the feature space) is an important parameter. It is related to the total variance of the training data. Empirically it has been found that $\delta = \log_{10}(\text{tr}(\Sigma))$, where Σ is the covariance matrix of the training data.

For the accelerated Gaussianization, the adaptive grid is comped with the help of a heuristic, as described in Section 3.3.2.

3.3 Gaussianization speed up

The density of grid points in the analyzed feature space is set in relation to the variance of the available training data and can be very high. This, corroborated with the dimension of

[3]This is equivalent ot finding the extremum point of the function $g(u) = \int f(u(t)) - \mathcal{A}u(t)\ dt$ by applying a gradient method.

[4]Assuming $\alpha = 1$, if the values of both constants are chosen to small in relation to f, it may lead to a transform that is not diffeomorphic.

[5]This is equivalent to say that the involved derivatives are approximated with finite differences and solved numerically.

the training space (i.e., the hyper volume of the grid), leads in many cases to an extremely large number of grid points. For a feature space with d dimensions and with g grid points per dimension we have d^g grid points. Therefore, even for relatively small (for a pattern-recognition problem) dimensions of the feature space, like for example 15, and with a low density of grid points of 10 per dimension, we have large numbers of grid points – in our example $\sim 5.76 \cdot 10^{11}$ grid points.

It is then clear that the multiclass Gaussianization described above is computationally demanding and thus slow, having to determine the displacement field at each grid point. For practical purposes, we need thus to increase the speed of the standard Gaussianization.

To speed up the Gaussianization there are in principle three main possibilities:

1. Apply a dimensionality reduction technique, before the Gaussianization.
2. Come up with a faster numerical solver to the system of equations (3.7).
3. Reduce the computational burden by decreasing the number of parameters.

The most straight forward way to reduce the complexity of the Gaussianization is to compute the transform on a feature space with less dimensions. In this case we need a suitable projection operator from the original high-dimensional feature space to a low-dimensional one. In other words, we need to find a transformation to be plugged in before the Gaussianization to enable us to reduce the dimensionality, while keeping the information we are interested in from the original feature space.

The majority of dimensionality reduction methods, like the linear PCA or the nonlinear Locally Linear Embedding [156], work properly only under a set of assumptions that in many cases go against the entire argumentation that followed here. These assumptions are in general stricter for linear dimensionality reduction than non-linear dimensionality reduction methods. However, the latter are usually tailored to certain problems, being specifically designed for them.

For these reasons, dimensionality reduction before Gaussianization should be considered only as a last resort solution, to be used when nothing else works. Nevertheless, this solution is always available and it would be interesting to find out which transformations are best suited to be used in this role.

Next we will concentrate on the remaining two ways to speed up the Gaussianization. There are several ways to devise a faster numerical solver. In the context of partial differential equations, these include also multigrid methods and *adaptive grid* methods. Although adaptive grid methods are a subset of the multigrid methods, they will be treated separately here, as they represent the inspiration for the *accelerated Gaussianization*, proposed here as a solution to the problem of increasing the speed of the standard Gaussianization. Thus, we discuss next in more detail faster numerical solvers in Section 3.3.1, including multigrid methods and then introduce in Section 3.3.2 the accelerated Gaussianization, which is an adaptive grid-based approach to reduce the computational burden of the multiclass Gaussianization by decreasing the number of parameters.

3.3.1 A faster numerical solver

Solving the Euler-Lagrange equation (3.5) involves computing the at each iteration k (see equation (3.7)) a solution to a linear system of equations of the form $\varphi = A\nu$ that results from the

finite differences approximation of the partial differential equation over the transform grid.

In general, depending upon how complicated a system of equations is, we may chose different methods to solve it. In practice we need to use numerical methods (i.e., algorithms), and thus our choice is really between direct and iterative methods. *Direct methods* like Gaussian elimination return the true solution in a certain number of computation steps. *Iterative methods* like the Jacobi method and the Gauss-Seidel method, improve at each iteration step an initial poor approximation of the true solution. In general, the iterative methods would need a very large number of steps to reach the true solution, however approximate solutions are available from the very first iteration. This leads to the following consideration: if the system to be solved is very complicated, such that a direct method would take too long to compute, we use an iterative method with a number of steps n chosen such that it takes less time to compute than the direct method and the approximation at the final step is satisfactory. The Conjugate Gradient (CG) method [94] is an example for a numerical method that is direct, but admits also iterative implementations. By a complicated system of equations we meant until now a large system of equations, with a high number of individual equations. If the matrix of the system of equations is not rectangular or it has a low rank, then we have another type of complications in that we either have no or to many solutions. There are several ways to proceed in these cases as well: use the pseudoinverse, or SVD, or again use iterative methods. It appears thus that iterative methods represent a general solution for many types of problems related to systems of equations.

For the multiclass Gaussianization, even if considering only the sheer size of this system of equations, it seems that the only option is to search the solution in an iterative manner and settle for a close approximation of the true solution. This is in general true for dimensions $d > 3$ and all but the smallest systems. For $d = 2, 3$ and with periodic boundary conditions, there exists a FFT-based technique [72] able to efficiently diagonalize the matrices arising from the finite difference discretization of the Euler-Lagrange equations that is faster and more accurate than iterative solutions. Nevertheless, in our case, typically we have $d > 3$, and thus we need to proceed with the iterative strategy. However, to reach a satisfactory coarse approximations to the true solution, the standard iterative methods will simply take too much time to be of practical use for the large systems that typically appear in our case. Therefore, the speed of iterative methods needs to be increased, which usually means reducing the number of iterations that are needed to reach the satisfactory solution, while the procedure that enables us to do so requires less computations than those needed for the saved iterations. Multigrid approaches allow us to do just that.

Multigrid approaches

The main problem of the iterative methods is the initialization. A poor initialization (i.e., the initial solution is very far from the true one) leads to a large number of iterations that need to be computed to reach the vicinity of the true solution. Thus, to speed up such algorithms a good initialization is needed. Multigrid considerations [185, 177] are instrumental on this path, in our context of numerical solutions to partial differential equations (PDE). With multigrid methods, we would first find a direct solution to a reduced system of equations that we obtain by simply disregarding some equations (and equivalently some unknowns). Then, we would extrapolate this reduced solution to the entire system and use this as initialization for the iterative scheme. The difficulty with this approach is related to the frequency characteristics of the error. If the error is smooth, then everything is alright, as the multigrid methods can cope with low-

frequency errors. Conversely high-frequency errors cannot be well approximated on a coarser grid. However, there are solutions that work well with high-frequency errors and bad with low-frequency errors, showing thus a complementary behavior to the naïve multigrid approaches. State of the art multigrid methods combine both these strategies.

Dealing with low-frequency errors. Within the PDE context, the multigrid approach comes naturally when considering discrete approximations, like for example the method of finite differences. Discrete approximations are either used to approximate the solution numerically, with the help of a computer, or they arise naturally for example in the field of digital signal processing, as in the case of image registration or our nonlinear Gaussianization.

For a coarse grid Ω_C, we have the system of equations $A_C\nu_C = \varphi_C$. If we go to a finer grid Ω_f we have the system of equations:

$$A_f\nu_f = \varphi_f. \tag{3.8}$$

The solution ν_f that we compute here is a better approximation of the true solution ν^T, but is achieved with a more complicated system of equations. Next we seek to improve the approximation while keeping a manageable complexity.

With ν_f^A some approximative solution on the fine grid Ω_f, we have:

$$A_f(\nu_f^A - \nu_f) = A_f\nu_f^A - \varphi_f.$$

If we define the defect

$$d_f = A_f\nu_f^A - \varphi_f \tag{3.9}$$

and the error $e_f = \nu_f^A - \nu_f$ we obtain:

$$A_f e_f = d_f. \tag{3.10}$$

Computing the error by $e_f = A_f^{-1}d_f$ gives us the possibility to compute $\nu_f = \nu_f^A - e_f$, but solving equation (3.10) is in no way more simple than solving equation (3.8). The breakthrough is achieved when we project the error back on the coarse grid Ω_C with the help of a restriction operator $r : \Omega_f \to \Omega_C$ and solve there the less complicated system of equations:

$$A_C e_C = r(d_f). \tag{3.11}$$

The coarse error

$$e_C = A_C^{-1}r(d_f) \tag{3.12}$$

allows us to compute an improved approximation of the true solution on the fine grid as

$$\nu_f^I = \nu_f^A - p(e_C), \tag{3.13}$$

where $p : \Omega_C \to \Omega_f$ is the prolongation operator, that projects the coarse error e_C (that can be understood now as a correction term) on the fine grid Ω_f. We achieve thus a good approximation with reduced complexity. Considering that usually an error has both high-frequency and low-frequency components, this method can successfully cope with the low-frequency components.

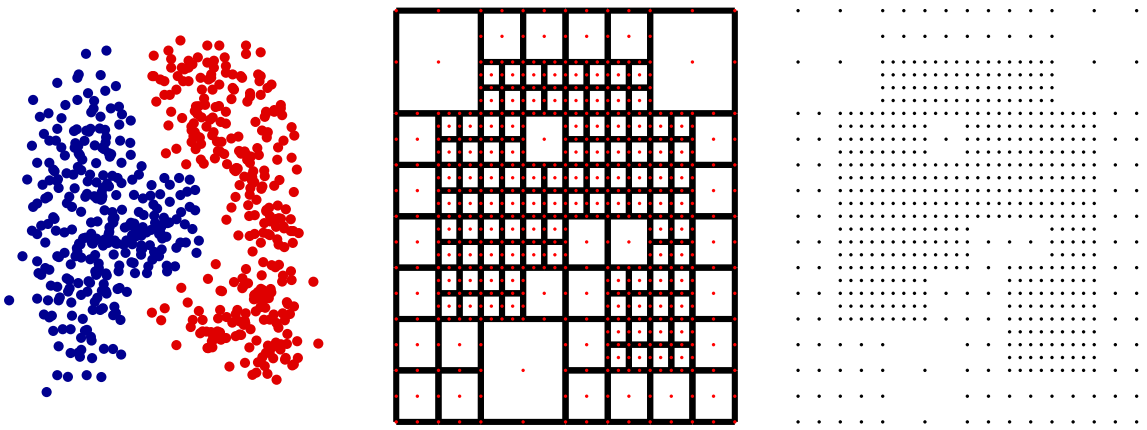


Figure 3.6: Shown here are: first the original non-Gaussian data (the class affiliation is color coded), then the adaptive grid as red dots with the subregions R_i, drawn with black lines (the smaller subregions are not drawn for display purposes) and finally the adaptive grid.

Dealing with high-frequency errors. For a useful solution strategy for systems of equations, we need now to find some way to reduce the high-frequency component of the error as well. However, the methods we employ should not add more complexity than what we have gained by working on the coarse grid. It can be proven that standard iterative methods like the Jacobi method (see Chapter A) are proficient at reducing the high-frequency components of the error but have problems with the low-frequency components. At the same time, the additional complexity brought by executing a few steps from such an iterative method is small, smaller than what we have gained by working on Ω_C.

We are now ready to introduce the two-grid approach [90] as an iterative sequence of several steps aimed at rapidly finding a good solution for the system of equations (3.8):

1. Smoothing: On Ω_f apply a few steps of an iterative method for numerically solving a system of equations, to compute ν_f^A.
2. Defect: Compute the defect with equation (3.9).
3. Restriction: Use the restriction operator to compute $d_C = r(d_f)$.
4. Coarse error: Compute the coarse error with equation (3.12).
5. Prolongation: Use the prolongation operator to compute $e_f = p(e_C)$
6. Correction: Compute the improved estimate of the true solution with equation (3.13).

Repeat these steps until $\left\| A_f \nu_f^I - \varphi_f \right\|^2 < \epsilon$, for a small tolerance ϵ or for a predefined number of steps. We go from two-grid to multigrid approaches, by applying the two-grid approach to compute the coarse error in step four.

3.3.2 Adaptive multigrid methods for accelerated Gaussianization

Adaptive grids are usually employed to decrease the computation burden (but also to increase the accuracy) of multigrid methods applied to the solving of PDEs. The core idea is to spend

computational power only where it is needed, i.e, around the areas of interest, where the change occurs. There are two main ways to implement this idea [177], in the form of (i) static (or predefined) adaptive grids and (ii) dynamic (or self-adaptive) adaptive grids. In the former case, the structure of the grid is defined before the computation starts and in the latter case, the grid modifies its structure during the computation.

Adaptive grid numerical methods represent efficient and general ways to solve systems of equations and were used here as inspiration for an accelerated Gaussianization. The idea followed now is simple, being based on the observation that in the training set the data points are not uniformly distributed over Ω. Therefore, instead of computing the displacement field over a fixed grid, they will be computed over a variable grid that is dense where the data is dense and sparse where the data is sparse. The grid size δ becomes now the lower bound of an adaptive grid. To compute the multiclass Gaussianization transform, the fixed-point iteration scheme (3.7) is adjusted to the adaptive grid.

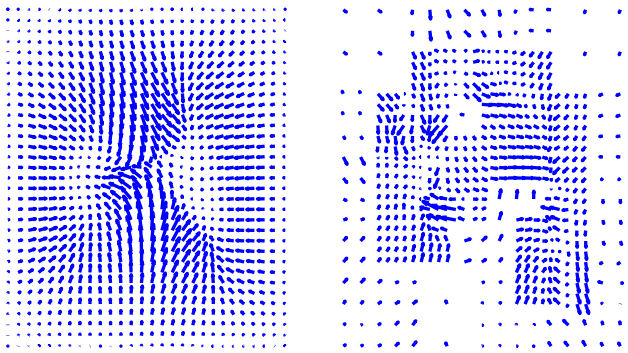
The iteration

For the purpose of Gaussianization a predefined adaptive grid is introduced next. The grid is defined in such a way that in the regions where the data is sparse, the grid is sparse as well, while in the regions where the data is maximally concentrated, the density of grid points remains constant in comparison to the fixed grid. As a consequence the total number of grid points decreases, while still ensuring a proper link to the feature space. We start by constructing the adaptive grid, for which purpose we determine the granularity of the grid in non-overlapping hypersquare-sub-regions of Ω. Grid points are positioned at the corners of these regions. Then, this adaptive grid is used with the fixed point iteration (3.7). The iteration is conducted first at the coarsest grid, over the entire Ω and then hypersquare-wise at finer grids, using as initialization the extrapolated result from the previous coarser grid. At each iteration, the corresponding systems of equations are solved with the CG. Once a hypersquare-sub-region reaches its finest granularity, no further computations are conducted there. A accelerated Gaussianization displacement field is shown in Figure 3.7(b).

The adaptive grid

There is no standard solution for the problem of defining an adaptive grid in this context. Here a heuristic is used that is derived from the field of image segmentation. The heuristic stems from the region splitting image-segmentation algorithm [81]. This is a region-based procedure that makes use of a homogeneity criterion $H(R)$ to find out if a certain region R_i of the input image is part of the object or of the background. As a result, the image I is divided into non-overlapping regions $R_i,\ i = 1, \ldots, N_R$, such that $\bigcup_{i=1}^{N_R} R_i = I$, while $\forall\, i :\ R_i \subseteq I$ and $\forall\, i \neq j :\ R_i \cap R_j = 0$. Each region satisfies $H(R_i)$. For a 2D image, region-splitting procedure begins by considering the entire image one region. If $H(R)$ is not fulfilled, then R is divided into four new regions, by halving each side of the initial region. This division step is repeated as long as $H(R_i) = false$. A data structure similar to a quad tree is thus obtained.

To implement the variable grid, the region-splitting procedure needs to be adapted. It should separate the data into foreground and background. The foreground is there where the data is concentrated, while the background is there where the data is sparse. Therefore, the homogeneity criterion has to measure how concentrated the data is in a region, and to do so $H(R_i)$ is



(a) (b)

Figure 3.7: Displacement field for the standard, static-grid transform (a) and for the fast, adaptive-grid transform (b), for the two-class input data from Figure 3.6

defined as

$$H(R_i): \begin{cases} \text{true}, & \text{if } \#\{R_i\} \leq \tau \\ \text{false}, & \text{otherwise} \end{cases}$$

where $\#\{R_i\}$ is the number of data points in R_i and τ is a threshold. τ is defined with the help of δ, the size of the static transform grid, as the maximum of the number of data points that can be found in a hypercube of side δ. This definition is used such as to ensure that at its finest granularity, the adaptive grid is similar to the static grid and thus a true reduction of the number of grid points is achieved in comparison to the standard Gaussianization. This procedure is illustrated in Figure 3.6.

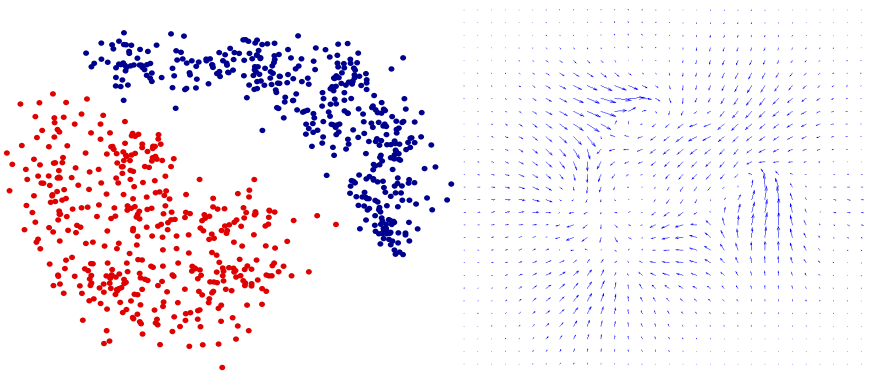
3.4 Experiments and discussion

The multiclass Gaussianization was tested on synthetic and real data. The tests on synthetic data in Section 3.4.1 are meant to demonstrate exemplary the benefits and limitations of the approach described above. The real data, used for the tests whose results are shown in Section 3.4.2, is Fischer's "Iris" dataset [18].

During testing the available data was randomly divided into a training and a test set, each containing 50% of the initial data points. The Gaussianization transform (both standard and accelerated) is computed using the training set and then applied on the test set.

The experiments were conducted with binary classifiers. The data was classified before (Org.) and after both the standard (Gauss.) and the accelerated (acc. Gauss.) Gaussianization with five types of classifiers: (i) a white Gaussian Bayesian classifier, computed under the assumption of equal, unit class covariance matrices (W); (ii) a linear Gaussian Bayesian classifier, computed under the assumption of equal class covariance matrices (L); (iii) a nonlinear Gaussian Bayesian classifier (nL); (iv) a SVM with a Radial Basis Function (rbf) kernel (SVM); and (v) a linear perceptron (P).

The elastic transform has several parameters, some are computed from the training data while others need to be set in advance. From among the latter, the magnitudes of λ and μ are related to the magnitude of the force term f. Here, the elastic transform is used to register densities that must integrate to one over the definition domain, and thus their values at every grid point are much smaller than one. Accordingly, the magnitudes in the force term will also be



(a) (b)

Figure 3.8: Standard Gaussianization: Linearly-separable data (a) and corresponding displacement field (b). The class affiliation is color coded.

small as will the values of the two parameters[6]. All experiments were satisfactorily conducted with the same parameter choice ($\alpha = 1$, $\lambda = 4 \cdot 10^{-5}$ and $\mu = 16 \cdot 10^{-5}$), established by six-fold cross validation on the linearly separable data set.

In all experiments the number of grid points for both the standard and the accelerated Gaussianization was also computed and the training time, i.e., the time needed to compute the parameters of the transform, was measured. The results are shown in Section 3.4.3. Finally in Section 3.4.4 the results obtained are discussed.

3.4.1 Experiments on synthetic data

Tests have been conducted on data sets with dimensions of up to 10 and with various numbers of classes between two and five. Here the findings are summarized on 2D examples, a dimension that is most convenient for visualization. There are two classes present. For each experiment 800 manually generated data points were used, 400 per class. The synthetic data is separable, differentiating between linearly separable data and nonlinearly separable data. The separable case was chosen for the tests such as to have a reference in the sense that error-free classification is possible there.

Linearly separable data

The linearly-separable data and the displacement field of the standard Gaussianization transform are shown in Figure 3.8. The result of standard Gaussianization on the test set is shown in Figure 3.9. The elastic transform parameters were $\delta = 2.1$ and $h = 1.6$. The same h was used for the accelerated Gaussianization as well. The classification results for both the standard and the accelerated Gaussianization are shown in Table 3.1. Only after Gaussianization does the linear Gaussian classifier find the separating surface. The accelerated Gaussianization clusters the data better thus yielding a transformed space with better Gaussian-separability properties, that are accordingly exploited by the classifiers.

[6]By comparison, in the case of image registration, assuming for example eight-bit images, the values at each and every grid point and the corresponding magnitudes in f will clearly be larger. Accordingly larger values for λ and μ are customary in this case. This should be understood as another proof that each problem has its own

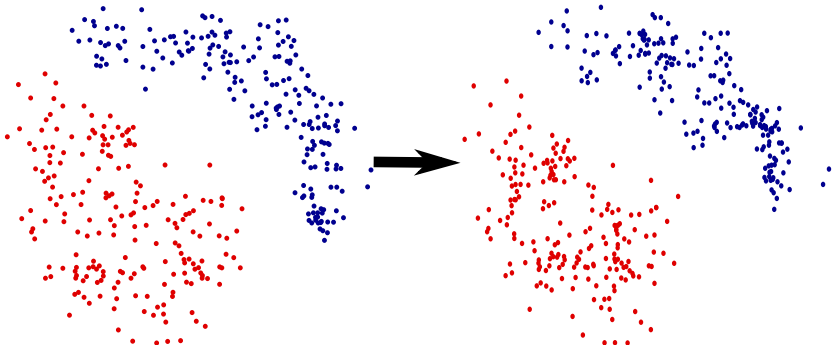


Figure 3.9: Standard Gaussianization: Linearly-separable test dataset before and after transformation.

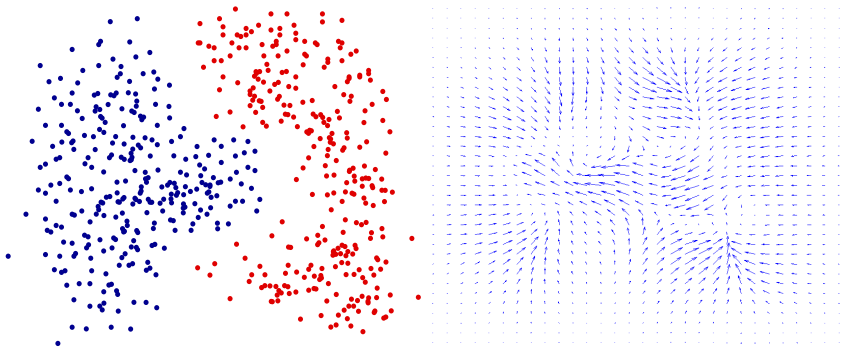
	W	L	nL	SVM (SVs)	P
Org.	1.75	0.75	0.75	0 (15)	0.25
Gauss.	0.25	0	0	0 (11)	0
acc. Gauss.	0.25	0	0	0 (12)	0

Table 3.1: Error rates (%) on the linearly separable dataset.

Nonlinearly separable data

We have also generated nonlinearly separable data, shown in Figure 3.10. The transform parameters for the standard Gaussianization were $\delta = 1.8$ and $h = 1.4$. The same h was used for the accelerated Gaussianization as well. The classification results for this case are shown in Table 3.2. The quadratic and linear Gaussian classifiers provide identical results, because the class-covariance matrices are very similar, as it can be seen in Figure 3.11.

particular solution even if the solution strategy is similar.



(a) (b)

Figure 3.10: Standard Gaussianization: Nonlinearly separable data (a) and corresponding displacement field (b). The class affiliation is color coded.

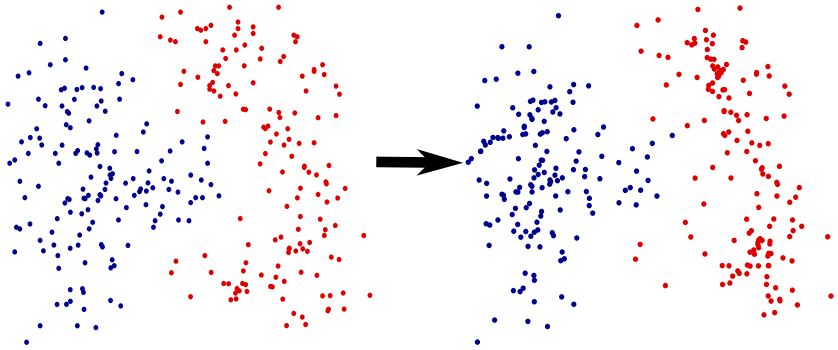


Figure 3.11: Standard Gaussianization: Nonlinearly separable test dataset before and after Gaussianization.

	W	L	nL	SVM (SVs)	P
Org.	5.25	4.25	4.25	1.25 (17)	3.75
Gauss.	2.25	1.75	1.75	0.25 (12)	1.75
acc. Gauss.	2.75	2	2	0.25 (14)	1.75

Table 3.2: Error rates (%) on the nonlinearly separable dataset.

3.4.2 Experiments on real data

The "Iris" dataset contains 150 feature vectors from three classes, with 50 vectors each. The classes are not separable. There are four features per vector. To adapt the binary classifiers to a multiclass scenario, a majority voting rule was used. The results are shown in Table 3.3. The elastic transform parameters for the standard Gaussianization were $\delta = 0.7$ and $h = 0.6$. The same h was used for the accelerated Gaussianization as well.

3.4.3 Complexity reduction with multigrid methods

The accelerated Gaussianization works by reducing the size of the grid where the elastic transform is computed and offering better initialization locally for the CG solver, but the parameters of several gird points (i.e, those also present on the coarser previous grid) are recomputed each time the grid turns finer in the respective region.

To investigate the decrease in computational complexity of the Gaussianization when using the adaptive grid, the number of grid points (the size of each grid) was computed for each of the tested datasets: linearly separable (2D-lin.), nonlinearly separable (2D-nlin.), and the "Iris" dataset (4D). The results are shown in Table 3.4.3. The time needed for each type of Gaus-

	W	L	nL	SVM (SVs)	P
Org.	8	2.66	2.67	1.33 (20)	10.66
Gauss.	10.66	1.33	0	0 (17)	14.66
acc. Gauss.	8	1.33	0	0 (15)	8

Table 3.3: Error rates (%) on the "Iris" dataset.

Dimension	Type	Grid size
2D-lin.	std.	1056
	acc.	563
2D-nlin.	std.	924
	acc.	589
4D	std.	2520
	acc.	2157

Table 3.4: Grid complexity for various datasets.

sianization was measured under MATLAB on a dual-core Opteron 8222 machine at 3GHz with 16GB of RAM in each scenario. The results show a decrease in computational time of at most 10%, even if the number of grid points is halved. The computation of the displacement field of the standard Gaussianization for the 4D "iris" dataset takes approximatively two minutes.

3.4.4 Discussion

The experiments on synthetic data show that after standard Gaussianization, the tested classifiers work better, in the sense that they make less errors. Also in comparison to other classifiers, the improvement of Gaussian classifiers—in terms of reduction of the error rate—is larger. For rbf-SVMs the number of support vectors diminishes. In the case of real data, the perceptron and the white Gaussian classifier misclassify three, respectively two more vectors after Gaussianization, but the nonlinear Gaussian classifier perfectly separates the data. Thus, the Gaussianization transform brings improvements only for the more powerful classifiers.

The same observations can be made for the accelerated Gaussianization as well. As in the case of the standard Gaussianization, the accelerated Gaussianization effectively makes the data more Gaussian, as shown by the improved performance of Gauss-related classifiers in the transformed space. The small increases of the error rate on the nonlinearly separable data set are due to the fact that in the regions where the adaptive grid is sparse, the displacement vectors of the nonlinear transform are larger – which follows from the very way the transform is computed. Test-set points falling in regions of the feature space covered by a coarse section of the adaptive grid will tend to travel further away, potentially over the linear separation surface, but they do so in a grouped manner, such that in the case of the SVM, new SVs placed there lead to the group being correctly classified. Still the number of SVs is smaller than for the original data, as it may be observed in Table 3.2. Conversely the same behavior works to our advantage on the "Iris" dataset.

The main purpose of the accelerated Gaussianization is to reduce the training time (i.e., the time needed to compute the parameters of the elastic transform), such as to be able to apply the Gaussianization to features spaces of higher dimension. The accelerated Gaussianization works by reducing the size of the grid where the elastic transform is computed and offering better initialization locally for the conjugate-gradient solver. On the other hand, the parameters of several gird points (i.e, those also present on the coarser previous grid) are recomputed each time the grid turns finer in the respective region. Furthermore, with the adaptive grid some time is spent during the computation of the transform with the generation of the adaptive grid and then with the management of the adaptive grid. Therefore, the time reduction can not be

linear in the number of grid points. Nevertheless, as a whole, the time needed to train the Gaussianization is reduced, because, even if some grid points are recomputed several times, as a total, a smaller number of equations needs to be solved. Furthermore, the CG solver coverages in a smaller number of steps due to the improved initialization. The process can be sped up even further if solvers faster than the CG are used.

The transformation has several parameters. The grid size δ for the standard Gaussianization, the grid sizes (i.e., structure of the adaptive grid) for the accelerated Gaussianization, as well as the kernel size h can be computed from the training data. δ is a parameter to which the standard Gaussianization is particularly sensitive. The formula proposed here for this parameter was established empirically, therefore better choices are possible. The rest of the parameters, i.e., $\alpha,\ \lambda$ and μ, should be established by cross-validation. As the experiments show, the method is largely insensitive to the these three parameters, which is to be expected, because always only densities are registered.

3.5 Conclusions, outlook, and summary

A nonlinear multiclass Gaussianization transform was described in this section. Its purpose is to support the Gaussian assumption often made in many classification problems. The multiclass Gaussianization is computed as the displacement field of an elastic transform that makes the nonparametric pdf estimate of the training data as similar as possible, with respect to the SSD between the two functions, to a GMM with one component per class. The Gaussianization transform is defined at discrete positions over a region of interest centered on the training sample. Data points outside this region are left unmodified. The analysis is conducted only over the support of the available training set. Interpolation is used to compute the displacement at positions between the knots of the grid where the transform is defined. Two methods have been discussed: the standard Gaussianization that works on a fixed grid, and the accelerated Gaussianization (see also Section 3.5.1) that decreases the computational burden during training by working on an adaptive grid.

The classification strategy implicitly proposed here with the introduction of the Gaussianization includes a nonlinear transform followed by a relatively simple classifier. In the experiments only binary classifiers were used. Still, Gaussian discriminant function-based multiclass classifiers can be also easily used in the transformed feature space. This strategy is arguably similar to that used by a SVM, however the Gaussianization described here is an explicit transform, as opposed to the kernel trick and is not limited to binary classifiers.

A nonlinear transform has complete control over the original feature space and theoretically, with such a transform perfect Gaussianization is possible. As discussed in Section 3.5.2, the algorithmic choices made here make perfect Gaussianization impossible, but arguably, in this case perfect, reasonable Gaussianization is anyway impossible, thus nothing is lost while obtaining acceptable Gaussianization and improved usability.

In summary, here for the first time a complete nonlinear multi-class Gaussianization solution was described and demonstrated on several examples. The practice will show if the precise algorithms used to implement the Gaussianization represent truly the best choices. Even if the general steps of the Gaussianization (i.e., parametric and non-parametric density estimation, nonlinear, diffeomorphic transform and adaptive grid) are universally valid, the precise implementation of each step can be optimized for specific problems.

3.5.1 Accelerated Gaussianization

A major issue with the multiclass Gaussianization is the very long training time, related to the dimension of the processed feature space. For Gaussianization it is possible to reduce the computation burden and thus the training time, by making use of problem-specific information in the sense that we use the fact that the training data does not have a constant density everywhere over its support. Following this train of thought, the accelerated Gaussianization that uses an adaptive grid, was introduced. Then, the number of positions where the vectors of the displacement field need to be computed decreases, being correlated to the data density as this data density can be inferred from the training set. A predefined adaptive grid was used here, whose structure is found with a heuristic derived from the field of image segmentation. The adaptive grid is computed such as to ensure that the same number of training-set vectors is present in each hyperrectangle with grid-points at its corners. The adaptive grid accelerated Gaussianization effectively makes the input data more Gaussian, while reducing the computational complexity. The accelerated Gaussianization is an iterative multigrid method. There are also direct methods – developed in the context of image registration – to solve the system of equations (3.6) that, at least for relatively small sizes (up to 1024^2 and dimension $d = 2, 3$) decrease the computation time approximately by a factor of three in comparison to a multigrid approach [141]. In principle, at least for large dimensions of the processed data (as often encountered in pattern recognition as opposed to image registration) an adaptive grid method will outperform a direct method as a direct method will pointlessly spend computation power on parts of the feature space where the training data is sparse. A last resort solution to achieve a significant reduction in complexity for problems of very large size would be to make use of a dimensionality-reduction transform before Gaussianization. It remains to be investigated if this is a viable solution in this context.

3.5.2 Perfect Gaussianization

A most important fact that should be take into account is that on the way towards Gaussianization, we need to pay attention not to lose the important data-contained information in which we are interested. For example, in the case of classification and multiclass Gaussianization that were discussed here, this important information is about the differences between classes, because these differences allow us to shatter the feature space and conduct classification. If we completely destroy data/feature-space cohesion only to reach Gaussianity, we may find out that we nolonger can conduct classification in the Gaussian feature space. More precisely, the problem is that there where two classes overlap, we simply don't know in which direction to push to make each class Gaussian. Hence the decision to make the smallest possible modifications such that each class becomes more Gaussian than before. In other words, in order to keep data cohesion a regularizer is used that favors such a behavior, namely the linearized elastic potential. Probability conservation would also play a role to alleviate such problems but in turn would also increase computational complexity and thus decrease usability. This issue is also related to the fact that the class labels of the training data are not explicitly considered while computing the Gaussianization.

As already mentioned, with a nonlinear transform, should another regularizer been chosen, the data could possibly have been turned into a perfect GMM. However, even if we have had the same number of modes in the mixture as the number of classes and the proper priors, we

probably could not have guaranteed that one mode is responsible for just one class, without explicitly considering the class-label information while computing the transform. An unreasonable Gaussianization could have thus been obtained, were a class would have been generated by several different modes.

New means and variances for the transformed data could have also been imposed with the aim of increasing separability. However, this practically destroys the cohesion of the data with negative influences on the classification performance. At the same time, the transform is generally diffeomorphic and the elastic constraint used supports data cohesion in the transformed space. In other words, the transform works well when each class in the original space is close to Gaussian and the new class means and covariances are close to the old ones. The more dissimilar the original class-conditional distributions are to the target Gaussian distributions, the larger the displacements the data term requires to minimize the SSD and the stronger the opposition of the elastic potential regularizer. On top of this, probability conservation issues are ignored and the multiclass Gaussianization is susceptible to the curse of dimensionality, as it relies on pdf estimation. It is thus clear that the methods described here can not achieve perfect Gaussian class-conditional pdfs. However, this is not the purpose here, but rather it is desired to make these class-conditional pdfs more Gaussian than before, such that Gaussian methods are more appropriate on the new data. This goal is achieved, as shown by the superior performance of Gaussian methods after Gaussianization.

Chapter 4

Improved hysteresis classification

Classification in a skewed feature space, where there is a strong disproportion between the number of representatives of each class, is a challenging task in particular for high-accuracy classifiers, which strive to achieve a minimal number of false decisions. Classification in the presence of strong overlap is difficult irrespective of the chosen method, however, good results can be achieved if prior knowledge is used. A set of classifiers will be introduced here that share the same principled approach constituting together a classification paradigm able to effectively deal with these types of problems. This paradigm will be demonstrated on the problem of retinal-image segmentation, as it represents a good example where both these types of problems arise. Furthermore retinal-image segmentation is needed in retina-based person authentication, as discussed in Section 5.1.2. Within the paradigm, the accent will be set on new, more powerful classification methods.

The purpose of image segmentation is to separate objects from background, where by objects one understands items of interest in the analyzed images. As a binary pattern classification problem, where each pixel needs to be assigned to either the background or the object class, image segmentation is usually afflicted by class skew, as there are often much more background pixels than object ones. Also, in many cases, there is a strong overlap between object and background in the pixel feature space, usually due to the variations of the object and background-defining properties. Therefore, a paradigm is introduced next, which makes explicit use of the prior knowledge about the connectivity of objects, to provide a framework for easily and successfully designing binary classifiers for such purposes. These classifiers return good results despite large class skew and overlap. The accent is set here on binary classification, as in a large number of cases segmentation implies the separation of a single object – or class of objects – from the background. Should several object classes be present, binary classifiers can still be used, e.g., one for each class.

A particularly challenging problem is the segmentation of retinal vessels in photographies of the retina. Photographies of the retina showing its vasculature are used both in medical and security applications. For such purposes, the retinal vessels need to be segmented to compute measures like vessel area and length, vessel width, normal and abnormal branching, and also to provide a localization of vascular structures. The contrast of vessels in the analyzed images is related to the quantity of blood found therein. Hence, small vessels have a weak contrast. Also differences between vessels and background pixels are localized in the vicinity of the vessel. The background is usually inhomogeneous and can be locally similar to the vessels. All these lead to poor separability between vessels and background. Conversely vessels are connected

structures, as the blood flows from the large vessels through smaller ones to the capillaries.

Vessel segmentation in 2D-projection images, including among others retinal images, is a topic of high interest in the machine-vision community [112]. Owing to the typical separability problems of vessel images, the first step in extracting the vessels is to enhance them, i.e., increase their contrast/separability to the background. For this purpose a large variety of techniques have been proposed including matched filters [150, 98], Hessian measures [74] the wavelet transform [34, 169], line detectors [155], often this methods include multiscale computations to compensate for the large variety of vessel sizes [133]. Particularly successful have been approaches where various vessel measures, aiming at different vessel properties have been combined to generate multidimensional pixel-feature vectors [172, 40, 155] or new vesselness measures [170, 117].

Vessels can be then segmented in a supervised manner based on a pre-labeled set of examples, e.g., using k-nearest neighbor classification [172], hysteresis classification [38], centerline detectors [170], Gaussian Mixture Models-based Bayesian classifiers [169], SVMs [155] or in an unsupervised manner by, e.g., tracking [34], clustering [40, 39], centerline detection and region growing [138, 133], as well as by unsupervised hysteresis classification [37, 143] and multi-thresholding methods [105, 98]. In particular such multi-threshold methods can achieve good results being especially designed to deal with the large overlap of the vessel and background pixel classes.

The solution to the vessel segmentation problem, which is proposed here is the *Hysteresis Classification Paradigm* that makes use of the connectivity of vessels to return fast and accurate classifiers. The methods described here are used to label each pixel individually. Most vessel segmentation algorithms are unsupervised and semiautomatic [148], but there are some important applications in which supervised methods are well suited and automatic methods are needed, like e.g., retina-based person identification and screening for diabetic rethinopathy [113]. The hysteresis paradigm can be used to generate both types of methods for either scalar or vectorial inputs. It uses two classifiers:

1. *The pessimist*, which is characterized by high specificity, working with a practically zero false-positive rate, which with overlapping classes implies a high false-negative rate.

2. *The optimist*, which is characterized by high sensitivity, working with a practically zero false-negative rate and a high false-positive rate.

Then, using the connectivity property of vessels, the pessimist classification can be used to select true vessels from among the optimist classification. This last connectivity-based step is in a way similar to region growing techniques, where the seeds are generated by the pessimist and the optimist helps define the logical predicate needed for the growing process. However, the hysteresis methods are significantly faster.

Next, a hysteresis classification method for scalar inputs will be denoted as a *hysteresis threshold*, while a hysteresis classification method for vectorial inputs will be denoted as a hysteresis classifier. By design the hysteresis paradigm yields for scalar inputs two thresholds [41] and it is thus a multi-threshold method [163]. Typical multi-threshold methods are actually local thresholds, where various image regions (varying in size from large portions of the input image to one pixel [2] [1], [105]) get different thresholds when conducting segmentation, as a way to compensate for varying background. The hysteresis threshold however always generates only two global thresholds, the final segmentation being then generated based on the connectivity

relationships among object pixels. The discussion here is centered on hysteresis classifiers that by comparison to hysteresis thresholds yield better results.

Hysteresis methods are not new to neither image processing nor vessel segmentation, however, no consideration was given before to the way the two classifiers should be chosen in relation to the available data. The hysteresis classification paradigm includes methods to properly establish the parameters of hysteresis classifiers by training.

For image segmentation in general, the hysteresis-classification paradigm yields two types of classifiers the absolute and the relative hysteresis classifiers that will be described in more detail next. The absolute hysteresis classifier has already been proposed in Reference [36], while relative hysteresis classifiers, which as shown in Section 4.3 yield improved results, are new developments, being proposed for the first time in References [49] and [50]. For the specific problem of retinal vessel segmentation, hysteresis classifiers return results that are comparable or slightly better but computed faster than some of the most powerful state-of-the-art methods specially designed for this problem [98], [105], [169], [172], [138], [155], [117].

Next, in Section 4.1 the hysteresis classification paradigm is discussed together with its application to vessel segmentation. In Section 4.2 a feature extraction process is described that results in the computation of a feature vector for each pixel of an analyzed image. In Section 4.3 various hysteresis classifiers are tested and compared to state of the art methods. Finally Section 4.4 gives a critical review of the proposed methods.

4.1 A hysteresis binary-classification paradigm applied to image segmentation

The concept of hysteresis thresholding for image analysis is not new, being used successfully for edge segmentation by Canny [28] and even before this to construct motion masks [52]. With respect to the design of the hysteresis threshold, Canny only mentions that the high-confidence threshold should be some two times larger than the low-confidence one, as in the edge map the edges are always brighter than the background. Such considerations are characteristic for the way the hysteresis threshold is currently used. The two thresholds usually stay in a fixed, predetermined relationship to one another. The hysteresis classification paradigm introduced here provides means for the design of flexible hysteresis methods that can adapt to the analyzed data and return thus better results.

The set of concepts related to the hysteresis paradigm is described beginning with Section 4.1.1. Particular implementations of these concepts lead to various types of hysteresis classifiers. Hysteresis classifiers consist of a high- and a low-confidence classifier and may be used for both scalar and vectorial inputs, assuming that a condition about connectivity of the object-class pixels holds. A major hysteresis concept is that of percentile-based classification, which is discussed in Section 4.1.2. This leads to highly accurate and robust relative hysteresis classifiers. In Section 4.1.3 the hysteresis framework will be used to construct classifiers for the particular problem of image segmentation, which are then applied to the segmentation of vessels. In Section 4.1.4 a set of rules is introduced that permits the training of hysteresis classifiers.

4.1.1 Hysteresis classification

In the hysteresis classification paradigm, the high- and low-confidence classifiers are called the pessimist and the optimist, respectively, and they represent the base classifiers. When applying the paradigm to image segmentation, connectivity is defined over neighborhood relationships among pixel sites. A border-separability constraint is introduced that is supposed to ensure that the connectivity condition leads exclusively to the selection of object pixels. For image segmentation, the paradigm yields two types of classifiers, the absolute one, taking into consideration only individual pixels, and the relative one, considering also the analyzed image. Finally, for image segmentation it is shown how to conduct feature extraction using object maps. A flowchart of the hysteresis-classification paradigm is shown in Figure 4.1.

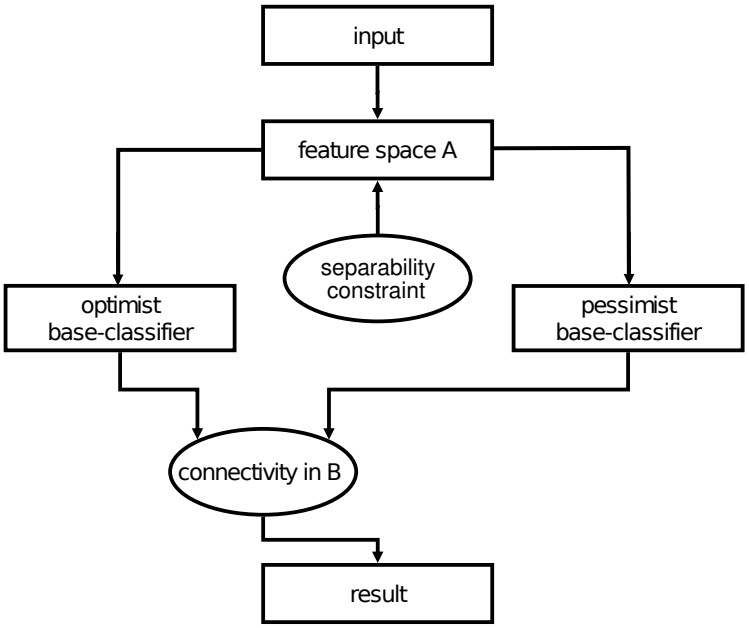


Figure 4.1: Flowchart of the hysteresis classification paradigm.

Base classifiers. If the two classes of a binary classification problem overlap strongly but not completely in some feature space (denoted next by A), then error-free classification is impossible there. If the components of one class do exhibit some type of connectivity in a different feature space (denoted next by B), where there is no overlap (i.e., the classes are disjoint), then the hysteresis paradigm is used to design methods that may achieve error-free classification. Two classifiers working in feature space A (i.e., the pessimist and the optimist), coupled over the connectivity constraint in B, build a *hysteresis classifier*. However, if the connectivity and disjointness in B can be described by some numerical features, then these should be included in A, where a standard classifier can then be used to achieve error-free classification. Thus, those cases are considered in which the connectivity and disjointness in feature space B cannot be easily described numerically.

Hysteresis classification for image segmentation. For the particular case of image segmentation, the features in feature space A are computed from the intensity levels of the analyzed

image, such that for each pixel there is a point in the feature space. The feature set B is given by the set of pixel sites $S = \{s_1, s_2, \ldots, s_N\}$ with $N = m \times n$ for an image with m lines and n columns. Each pixel site is represented by a 2D position vector $s_i = [x_i, y_i]^T$ with $i \in \{1 \ldots N\}$. A neighborhood system $\mathcal{N}^k = \{\mathcal{N}_s, s \in S\}$ is introduced as a collection of *pairs* of pixel sites $\mathcal{N}_s$. For a site s_A, a pair is defined as $\mathcal{N}_{s_A} = \{s_A, s_B\}$, such that $s_B \in \mathcal{N}_{s_A} \Leftrightarrow s_A \in \mathcal{N}_{s_B}$ and the distance between them satisfies $d(s_A, s_B) \leq d_{\mathcal{N}}$, with $d(s_A, s_B) = [(x_A - x_B)^2 + (y_A - y_B)^2]^{\frac{1}{2}}$ being the Euclidean distance. If $d_{\mathcal{N}} = 1$, then we have a four-points neighborhood $\mathcal{N}^4$, as there are four pairs for each site. If $d_{\mathcal{N}} = \sqrt{2}$ we have an eight-points neighborhood $\mathcal{N}^8$.

Each base classifier returns a sets of labels $L = \{l_s, s \in S\}$ with $l_s \in \{0, 1\}$, for a total of two different label configurations $L_{optimist}$ and $L_{pessimist}$. With respect to a set of labels, B fulfills a disjointness condition, as each site receives one of two possible and mutually exclusive labels: object ($l_s = 1$), or background ($l_s = 0$). The object connectivity in B is expressed as a proximity relationship, i.e., object points are neighboring other object points according to their labels and the neighborhood system $\mathcal{N}^k$.

For vessel segmentation, vessels are considered to be connected objects, with each vessel-pixel site being linked over an $\mathcal{N}^8$ neighborhood to another vessel-pixel site. Therefore an image is segmented by selecting from $L_{optimist}$ all those vessel-pixel sites connected over a chain of $\mathcal{N}^8$ neighborhoods to a vessel-pixel site from $L_{pessimist}$.

The border-separability constraint. In order to select all object points from A, assuming overlapping classes, the "optimist" will falsely classify many background points as object points. For the hysteresis classification to work without without making any errors, these false objects must be situated in B at a distance larger than the chosen threshold $d_{\mathcal{N}}$ from true objects. In other words, they may not be connected to true objects, and more precisely, they may not be in the neighborhood of true objects. Therefore, the borders of objects have to be sufficiently separable from the surrounding background such that there exists a classifier which yields a segmentation result where true object pixels are separated from false object pixels by $d_{min} > d_{\mathcal{N}}$ in B. This property is called *the border-separability constraint*. In practice, of course, the border-separability constraint will not always hold, and it should be enforced prior to segmentation in the feature-extraction process.

Relative and absolute hysteresis classification. For image segmentation, the hysteresis classification paradigm returns two types of classifiers: the absolute hysteresis classifier, and the relative hysteresis classifier. For the absolute hysteresis classifier, the training returns the parameters for two separation surfaces, one for each base classifier. During testing and operation, these surfaces are used to decide for each pixel, irrespective of the image it comes from, whether it is a background or an object pixel. For the relative hysteresis classifier, during training two percentile values are found. These values are used during testing and operation to compute for each analyzed image two individual separating surfaces that are in turn used to decide for each pixel in the respective image whether it belongs to the background or to the object class.

Feature extraction: the object map. In the case of image segmentation, feature extraction implies a chain of processing steps aimed at improving the separability between the two pixel classes, like, for example, improving the contrast and the homogeneity of the pixel-intensity-level representation of objects and background. For hysteresis segmentation, special care has

to be taken during feature extraction to preserve and enhance the objects' borders, in support of the border-separability constraint. To avoid undesired bias, the features should be normalized. The result of feature extraction is an object map, in which each pixel has an intensity value attached as feature.

4.1.2 Relative hysteresis classification and percentiles

To design the base classifiers for relative hysteresis classification, percentiles will be used extensively. Percentiles will be introduced in this context starting from a significance test on scalar inputs, i.e., in the case of a hysteresis threshold.

The *k*'th percentile is defined as that value of a 1D random variable, which is larger than *k* percent of all other realizations in the available sample. As we are on the real axis, it is self-evident that the two margins of the sample are the maximal and the minimal value. Therefore, the percentile spans the real axis between these two extreme values, defining thresholds to select in uniform steps percentages of the number of realizations in the sample. For scalar feature spaces, each base classifier can be such a threshold. These concepts will be used to adapt the notion of percentile for vectorial inputs. Thus, for multidimensional feature spaces, we will discuss two ways to design the base classifiers: (i) by extending the notion of percentile with the help of a linear classifier, and (ii) by applying a feature extraction transform that maps the original feature space onto a line, where we can again work with the usual percentiles.

Percentile-based hysteresis threshold. We would like to determine the pessimist and the optimist for a hysteresis threshold. This is achieved by means of significance testing.

For the pessimist, the null hypothesis is that the pixel under investigation belongs to the background class. Hence, it is imposed that $P(x_b < t_p) = \alpha$ with x_b being an intensity value in the background class, t_p a threshold, and α the significance. We then have

$$P(x_b < t_p) = \sum_{i=v_{bmin}}^{t_p} \frac{n_{bi}}{N_b} = \alpha, \tag{4.1}$$

where v_{bmin} represents the minimum intensity value on the histogram (see Section 2.1.1) of the background intensities, n_{bi} denotes the number of background pixels with intensity value i, and N_b is the total number of background pixels in the image. The value t_p is then the α'th quantile of the histogram of the background's pixel-intensity levels.

The histogram of the image is the discrete approximation of the mixture of vessel and background class-conditional probability density functions (pdfs). Therefore, t_p is also a quantile of the histogram of the image and can be found via

$$P(x < t_p) = \sum_{i=v_{min}}^{t_p} \frac{n_i}{N} = \alpha_{im} \tag{4.2}$$

where x is a pixel intensity level in the image, v_{min} is the minimum intensity level on the histogram, n_i is the number of pixels with intensity level i, and N denotes the total number of pixels in the image. The threshold t_p is then the α_{im}'th quantile of the histogram of the image, and it should be chosen such that it selects practically *only* vessel pixels.

Similarly, the optimist is computed using the object's class-conditional pdf. This time, it is hypothesize that the pixel under consideration is an object pixel. To compute the threshold, again a small significance level β is imposed,

$$P(x_o > t_o) = \sum_{i=t_o}^{v_{omax}} \frac{n_{oi}}{N_o} = \beta, \tag{4.3}$$

where x_o is a pixel-intensity level in the object class, v_{omax} is the maximum intensity level on the histogram of the object's pixel-intensity levels, n_{oi} is the number of object pixels with intensity level i, and N_o is the total number of object pixels in the image. The value t_o is some quantile of the histogram of the object's pixel-intensity levels and it is also a quantile of the histogram of the image. It can be found from

$$P(x < t_o) = \sum_{i=v_{min}}^{t_o} \frac{n_i}{N} = \beta_{im}. \tag{4.4}$$

The threshold t_o is then the β_{im}'th quantile of the histogram of the image, and it should be chosen such that it selects practically *all* vessel pixels. For the purpose of hysteresis classification percentiles (i.e., 100'th quantiles) are used. An example showing the thresholds t_p and t_o and the two class-conditional pdfs can be seen in Figure 4.2 (a).

The linear-classifier percentile. The linear-classifier percentile (LCP) is introduced next for the 2D case. An extension to more dimensions is straightforward, with the LCP turning from a line into a hyperplane.

To define the LCP, one should first establish the margins of the sample. Optimally, these are along the axis of largest separability. Then, one should also define a way to select percentages of the total number of realizations in the sample. For this second purpose, we need a type of separating surface. A "linear" percentile is obtained when this separating surface is a line. Thus, a LCP is defined by a linear separating surface and by its position on an axis perpendicular to it, i.e., the axis of largest separability.

A linear separating surface

$$h_l(\mathbf{x}) = \mathbf{b}^T\mathbf{x} + c = 0 \tag{4.5}$$

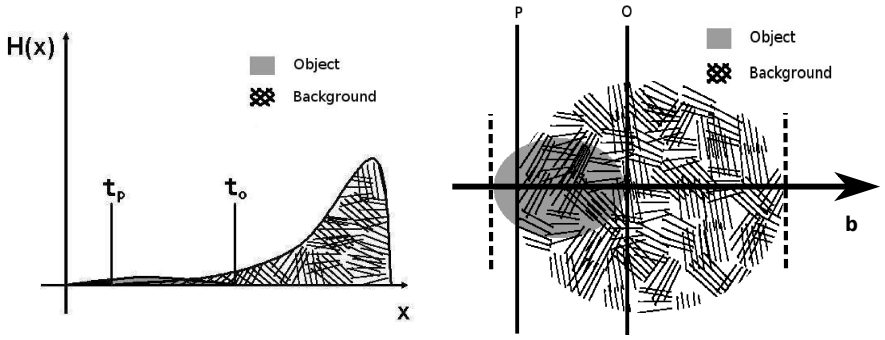
is defined in 2D by the vector of weights $\mathbf{b} = [b_1, b_2]^T$ and the position c. By modifying c, the separating surface is moved over certain distances on the axis defined by $\mathbf{b}$, such that it selects percentages of the available sample in unit steps. The scalar product $\mathbf{b}^T\mathbf{x}$ can also be seen as applying a transformation $\mathbf{b}$ to the data vector $\mathbf{x}$ that maps it to a scalar value (see also Section 1.1.1).

We need now to define also a direction on the axis, i.e., an orientation of the separating surface. This is defined from the mean of the object class towards the mean of the background class, which is equivalent in the 1D case to considering objects dark. Therefore, the k'th LCP separates k percent of the data from the rest, and for a relatively small k, most of this data will belong to the object. Then, the pessimist and the optimist are chosen from the set of decision surfaces given by the LCPs from zero to 100. The weight vector $\mathbf{b}$, encodes both the axis of largest separability and the orientation of the separating surface. An example of the LCP is shown in Figure 4.2 (b). Depending on the way $\mathbf{b}$ is computed, various types of LCPs may be constructed, as discussed later in Section **The relative hysteresis classifier** in more detail.

The transformation-based percentile. In an alternative setup, the notion of percentile is extended to vectorial inputs in an indirect manner, by means of a transformation. Thus, in the beginning a feature extraction transform is applied, such that a scalar random variable is obtained, where the usual percentiles may be used again. If a linear transform is used, we are in a setup similar to that of the LCP.

4.1.3 Hysteresis classifiers for vessel segmentation

In this section the absolute and the relative hysteresis classifiers are discussed in detail. While a hysteresis threshold works directly on an object map, which for vessel segmentation is called a vessel map, a hysteresis classifier works on a *pixel feature space* (see Section 4.2), which represents a collection of vessel maps such that each pixel is described by a vector. For the purposes followed here, a training set represents a sample from such a pixel feature space, corresponding to some or all pixels from the labeled images typically available for supervised image segmentation.



(a) (b)

Figure 4.2: Schematic representation of an object-map histogram with the pessimist and optimist thresholds, t_p and t_o respectively (a) and schematic representation of a 2D pixel feature-space with the axis of largest separability $\mathbf{b}$ (that ideally goes through the two class means), the margins of the sample (dashed line) and the optimist O and pessimist P classifiers (b).

The absolute hysteresis classifier

The absolute hysteresis classifier is a supervised, automatic method. It needs a labeled set of examples from the two classes, viz. vessels and background. This set is extracted from the pixel feature space and it is used to compute the parameters of the hysteresis classifier. These parameters remain constant for all analyzed images.

Inputs: all images in the training set, `number_iterations`, `classifier_type`
$idx_{max} \leftarrow$ `number_iterations`;
$idx \leftarrow 0$;
pessimist(idx)$\leftarrow 0$;
classifier_set = RAW_CLASSIFIER($l = 0$ to 100, `classifier_type`);
$ROC \leftarrow$ BUILD-ROC(classifier_set);
optimist(idx) $\leftarrow \arg\max_{l}$(DISTANCE-TO-ROC-BASELINE(ROC, l));
while $idx \leq idx_{max}$ **do**
  $idx \leftarrow idx + 1$;
  pessimist_classifier_set = RAW_CLASSIFIER($k = 0$ to 100, `classifier_type`);
  $hROC \leftarrow$ BUILD-HYSTERESIS-ROC(pessimist_classifier_set, optimist($idx - 1$));
  pessimist(idx) $\leftarrow \arg\max_{k}$(DISTANCE-TO-ROC-BASELINE($hROC, k$));
  optimist_classifier_set = RAW_CLASSIFIER($m = 0$ to 100,`classifier_type`);
  $hROC \leftarrow$ BUILD-HYSTERESIS-ROC(optimist_classifier_set, pessimist(idx));
  optimist(idx) $\leftarrow \arg\max_{m}$(DISTANCE-TO-ROC-BASELINE($hROC, m$));
  if pessimist(idx) == pessimist($idx - 1$) & optimist(idx) == optimist($idx - 1$)
    exit loop
  end if
end while
return hysteresis-classifier(pessimist,optimist)

Figure 4.3: Pseudo-code for the iterative training algorithm. For `classifier_type` = "absolute classifier", the RAW_CLASSIFIER(d,`classifier_type`) is a threshold. For `classifier_type` = "relative classifier", the RAW_CLASSIFIER(d,`classifier_type`) is a percentile. RAW_CLASSIFIER(d,`classifier_type`) denotes a classifiers that separates $d\%$ of data from the rest $(100 - d)\%$. By varying d from 0 to 100 we obtain a set of classifiers.

Absolute hysteresis classification and LDA. The absolute hysteresis classifier uses the Linear Discriminant Analysis (LDA) [78] [124]. In [38] a supervised absolute hysteresis classifier is described, where the pessimist and the optimist are two Fisher classifiers with parameters $(\mathbf{w}, T_p)$ and $(\mathbf{w}, T_o)$ respectively. For the binary classification problem here, $\mathbf{w}$ defines an LDA transformation from a multidimensional pixel-feature space to a 1D feature space. $T_p,\ T_o$ represent thresholds in this 1D space. $\mathbf{w}, T_p,$ and T_o represent the parameters of the hysteresis classifier.

During LDA, one looks for a transformation such that in the transformed space, the separability criterion

$$F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{4.6}$$

where $\mu_1,\ \mu_2$ and $\sigma_1,\ \sigma_2$ are the means and variances in the transformed space, is optimized. Using a labeled input feature space, the transformation weights are

$$\mathbf{w}^T = (\mathbf{m}_1 - \mathbf{m}_2)^T \left(\frac{n_1}{n} \mathbf{\Sigma}_1 + \frac{n_2}{n} \mathbf{\Sigma}_2 \right)^{-1} \tag{4.7}$$

where $\mathbf{m}_1,\ \mathbf{m}_2$ and $\mathbf{\Sigma}_1,\ \mathbf{\Sigma}_2$ are the class-conditional means and covariance matrices, respec-

tively. n_1 and n_2 are the numbers of components in each class and n is the total number of components. $\mathbf{w}$ is computed in the pixel-feature vector space, as described above, with $\mathbf{m}_1 = \mathbf{m}_b$ the mean of the background class and $\mathbf{m}_2 = \mathbf{m}_o$ the mean of the object class. T_p and T_o are found during hysteresis training on the pixel-feature vector space that includes all pixels from the available labeled images.

The relative hysteresis classifier

The relative hysteresis classifier is defined with the help of percentiles. Each image i is considered independently, hence the sample over which the percentiles are computed is given by all pixels in one image at a time. Thus, even if the percentile values obtained during training remain constant for all analyzed images, the parameters of the corresponding separation surfaces change from image to image, thus better adapting to the analyzed data and ultimately providing better results.

Two LCPs are used as base classifiers. Thus, for an image i we have

$$\begin{aligned} h_p^i &= \mathbf{b}^T\mathbf{x} + c_p^i \\ h_o^i &= \mathbf{b}^T\mathbf{x} + c_o^i \end{aligned} \tag{4.8}$$

The parameters c_p and c_o represent the positions of the corresponding linear separation surfaces along the axis defined by $\mathbf{b}$ such that they separate certain percentages of the available sample from the rest.

In the following, two types of relative classifiers will be described in detail. Depending on the way $\mathbf{b}$ is computed, the first uses the Gaussian LCP [49], and the second uses the LDA percentile [50]. The percentiles are found during hysteresis training.

The Gaussian linear-classifier-percentile-based relative classifier. In the case of the Gaussian LCP, to compute $\mathbf{b}$ it is assumed that in each image i of the training set the class-conditional pdfs of the object and the background are Gaussian, with equal covariance matrices. Starting from the likelihood ratio

$$l(\mathbf{x}) = \frac{p(\mathbf{x}|\omega_o)}{p(\mathbf{x}|\omega_b)}$$

we obtain under this assumption the decision rule

$$\mathbf{b}^{iT}\mathbf{x} + c \underset{\omega_o}{\overset{\omega_b}{\gtrless}} T,$$

which is based on the linear separation surface $\mathbf{b}^{iT}\mathbf{x} + c = 0$, where

$$\mathbf{b}^i = 2\mathbf{\Sigma}^{-1}(\mathbf{m}_b^i - \mathbf{m}_o^i)$$

with $\mathbf{m}_b^i$ being the mean of the background class and $\mathbf{m}_o^i$ denoting the mean of the object class. $\mathbf{\Sigma}$ describes the variance in both classes and can be for example the average scatter matrix, the minimal scatter matrix, etc. The vector $\mathbf{b}$ is then computed as

$$\mathbf{b} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{b}_i, \tag{4.9}$$

where M denotes the number of images in the training set. The two percentiles that yield c_p^i and c_o^i for each image are found during hysteresis training based on the images of the training set.

LDA-percentile-based relative classifier. For the LDA percentile, the LDA is used to compute the axis of largest separability. As shown in equation (4.7), the LDA returns a vector $\mathbf{w}$ as separability-optimal transform for two-class problems. The vector $\mathbf{w}_i$ is computed for each image i in the training set, and then, $\mathbf{b}$ is computed as the mean over all $\mathbf{w}_i$:

$$\mathbf{b} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{w}_i. \tag{4.10}$$

The base classifiers are now Fisher classifiers, but defined in a relative manner, as the corresponding thresholds $T_p^i \equiv c_p^i$ and $T_o^i \equiv c_o^i$ are computed from percentile values that are set during hysteresis training on the images of the training-set.

Considering the implicit Gaussian homoscedastic assumption of LDA [124], the LDA-based percentile can be considered a type of Gaussian percentile. More precisely, the LDA percentile is the Gaussian percentile where $\mathbf{\Sigma}$ was chosen to be the average scatter matrix.

4.1.4 Hysteresis training

The purpose of training is to establish the parameters of the base classifiers. A set of labeled examples is needed for training. The training is thus supervised. Assuming that $\mathbf{b}$, or $\mathbf{w}$ have already been computed, we will discuss here how to compute the other parameters. For absolute hysteresis classifiers, two thresholds have to be determined, and for relative hysteresis classifiers, two percentile values are needed.

The training algorithm is shown as pseudo code in Figure 4.3. For better understanding, the training algorithm is described in detail separately for the absolute and the relative hysteresis classifiers respectively. Nevertheless, the principles of hysteresis automatic training are common to both classifier types.

Training absolute classifiers. To compute the two thresholds the *Receiver Operating Characteristic* (ROC) is used. For the beginning a ROC is determined by varying the threshold T of a Fisher classifier $(\mathbf{w}, T)$ between the minimum and the maximum values in the transformed 1D space and computing each time the percentage of true positives or correct classifications (TP) and that of false positives (FP) on the pre-labeled training set. The thresholds T_p and T_o are then determined the following way:

1. Starting from a standard Fisher classifier used as "optimist" – with T corresponding to the ROC-point that is most distant to the baseline linking the ROC-points with FP= 0% and FP= 100% (i.e., the straight diagonal baseline across the ROC graph, corresponding to random performance) – try all "pessimists" by varying T_p between the minimum and the maximum value, building thus a hysteresis-ROC (this is the BUILD-HYSTERESIS-ROC (pessimist_classifier_set, optimist(i)) method in Figure 4.3). Using the hysteresis-ROC, choose the "pessimist" threshold corresponding to the point most distant to the baseline.

2. Apply the same procedure – this time keeping the "pessimist" constant (this is the BUILD-HYSTERESIS-ROC (optimist_classifier_set, pessimist(i)) method in Figure 4.3) – to find the "optimist".

3. Repeat for a predetermined number of steps, or until the thresholds do not change anymore.

Training relative classifiers. Training is conducted in this case in a manner similar to that used for absolute classifiers. The difference being that now, a percentile is used instead of a threshold. To compute the two percentiles, ROCs of percentile-based decisions are used now. A percentile-based decision implies running all percentiles from zero to 100, each time segmenting the available images. The corresponding ROC is constructed from the FP and TP rates of each percentile. The training procedure has then the following steps:

1. First, the ROC of a percentile-based decision is used to initialize the optimist as that percentile corresponding to the point which is most distant from the baseline. Then, the ROC of a hysteresis decision is built, using the previously established optimist and all possible pessimist classifiers corresponding to the percentiles from zero to 100 (this is the BUILD-HYSTERESIS-ROC (pessimist_classifier_set, optimist(i)) method in Figure 4.3). The pessimist corresponding to the point that is most distant from the baseline is selected.

2. The procedure is repeated, this time to find the optimist (this is the BUILD-HYSTERESIS-ROC (optimist_classifier_set, pessimist(i)) method in Figure 4.3).

3. Iterate for a predetermined number of steps, or until the base classifiers remain unchanged for two consecutive iterations.

4.2 Feature extraction for retinal-vessel segmentation

The purpose of feature extraction is to compute vessel maps (see Section 4.2.1) in which the separability of the vessel and object pixel classes is improved, while obeying the border-separability constraint. In the vessel map, each pixel has an intensity value attached as feature.

As discussed in Section 4.2.2, several maps can be used together in a pixel-based multidimensional description of vessels. In this case feature selection may be conducted to find out which vessel maps should be used.

4.2.1 Vessel maps

Five vessel maps [39], [40] have been used during experiments. These are:

1. The Bothat: the result of a Bothat transform, used to select contrasted structures of a certain size;

2. The Hessian single scale: the first eigenvalue of the Hessian matrix, to select elongated structures;

3. The Hessian multiscale: the result of the analysis of the eigenvalues of the Hessian matrix extended in a multiscale approach to also reach the small vessels and improve the homogeneity of the vessel class;

4. The Band-pass filter: the result of a band-pass filter, to select only vessels based on their size;

5. The Laplacian pyramid: the result of a multi-resolution analysis using the Laplace pyramid, to select only fine-detail vessel structures and improve the homogeneity of the vessel pixel class.

In each vessel map, the vessel pixels are characterized by smaller intensities than the background. Each vessel map has been processed to have unit variance and smallest positive mean, such that all pixel intensities are larger than zero. This has been achieved by normalizing first to zero mean and unit variance and then subtracting the smallest value. An example showing the vessel maps computed for an input image is depicted in Figure 4.4.

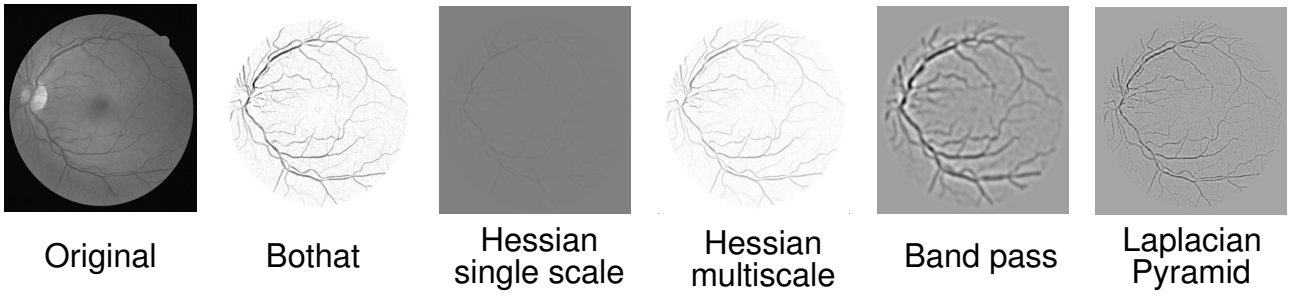


Figure 4.4: Retinal image and corresponding vessel maps.

Enforcing the separability constraint

Each vessel map depends on a set of parameters. These parameters were chosen using quality measures defined specifically for hysteresis-based vessel segmentation. These quality measures were designed such as to enforce the separability constraint, i.e., they get larger values for vessel maps where the vessels can be better separated from their immediate vicinity. An example of such a measure is the background-less partial area under the ROC (background-less pAROC). To compute this quality measure for a vessel map, a labeled ground-truth image is needed. First a ground-truth mask is defined by morphologically dilating the ground-truth image, thus being able to select only the vessels and their immediate vicinity. Next, the vessel map is segmented by a set of thresholds corresponding to the intensity-level percentiles from zero to 100, and a modified ROC (mROC) curve is built. The false-positive rates of the mROC are computed only from the region selected by the ground-truth mask. The quality measure is given by the pAROC, computed using a 2% bound on the false-positives rate. The pAROC was chosen because for "good" vessel maps, it is expected that the true-positives rate increases more rapidly at small false-positives rates in the direct vicinity of a vessel. More details and a discussion over various such measures can be found in [36].

4.2.2 Pixel-based multidimensional description of vessels

A multidimensional feature space is obtained by combining the results of several different vessel-enhancement methods. For each pixel, a feature vector is built by ordering its scalar features (i.e., intensities) in each object map into a vector [40, 38].

In each object map the separability between the background and object pixel-classes is increased. If possible, the border-separability constraint is also enforced. The strategy in this case is to combine the results of several different enhancement methods, in the hope that together they constitute a more separable representation of objects and background than any of them taken alone. It is believed that a multidimensional pixel feature space is better than a single vessel map, as it includes more information about vessels, acquired from different perspectives.

A schematic representation of the way a pixel-based feature vector space is computed is shown in Figure 4.5.

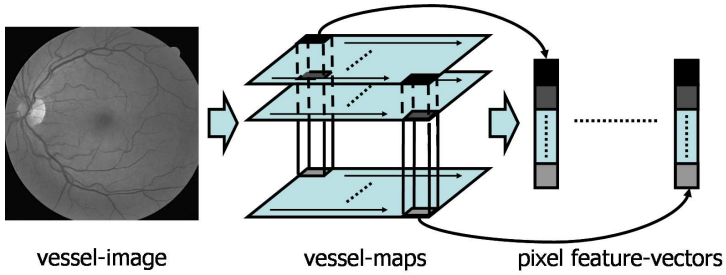


Figure 4.5: Schematic representation of the method to achieve a pixel-based multidimensional description of vessel and background.

Feature selection. The ROC of various decision rules can be used to characterize the feature space. Clearly, the larger the area under the ROC (AROC), the more separable the feature space is. In the limit the two classes are linearly separable when AROC is one, i.e 100% TP for 0% FP. If several features have been computed, then those which build the best feature space will also yield the largest AROC.

In the case of hysteresis classification we are interested also in the derivative of the ROC curve, especially in the region where the FP rate is small. A hysteresis classifier trained on a ROC curve with a large integral, but a comparatively mild increase over the region with small FP rate, will yield rather poor results, because the "pessimist" classifier will select too few true vessels. Thus, it is better to consider only a pAROC. During experiments only the AROC bounded by a 30% FP rate was computed.

Several strategies can then be followed. For example, a full search strategy where all possible combinations of features are investigated, or a sequential search strategy [151], where first the single best feature is selected, then the best combination between that feature and another one and so on, until the optimal feature set is found.

4.3 Experimental evaluation

Two publicly available databases were used during experimental validation: the DRIVE database [172], and the STARE database [98]. The DRIVE database contains 40 images, divided into a training and a test set. Each of these two sets contains 20 images. For the test images there are two sets of hand-labeled ground-truth images, marked as first and second observer, respectively, as they were generated by different groups of persons. For the training images, there is just one set of ground-truth images, marked first observer. The first-observer set was used as ground truth during the experiments.

The STARE database contains 20 images. As there is no training and test set, the segmentation performance on this database is computed by means of the leave-one-out method. The STARE database contains two sets of hand-labeled ground-truth images, again marked as first and second observer respectively. The first-observer set was used as ground truth during experiments.

Using the training set of the DRIVE database, the parameters of the enhancement methods used to generate the vessel maps were computed such as to optimize a quality criterion for vessel maps [36]. This quality criterion considers both the border-separability constraint and the separability between the two pixel classes. The same parameters were used for the STARE database as well.

For feature selection, the pixel-feature space built from the training images of the DRIVE dataset was used, and the ROC is computed as in the case of training absolute hysteresis classifiers. The ROC is determined by varying a threshold between the minimum and the maximum values in the transformed 1D space and computing each time the percentages of TP and FP on the pre-labeled training set. Having only five features to choose from, a full-search strategy was employed. After feature selection, only the Hessian multiscale, the Hessian single scale and Laplacian-pyramid based vessel maps remained. This type of feature vector was used on the STARE images as well.

For the Gaussian LCP-based relative hysteresis classifier, $\mathbf{\Sigma}$ was computed as the smallest of the object and background scatter matrices.

The performance of the classifiers was measured by the AROC. The corresponding ROC is computed by fixing the pessimist and modifying the optimist such that it assigns to the vessel class between 0% and 100% of the available test samples. Accuracy, sensitivity and specificity were computed as well. The *accuracy* is defined as

$$Ac = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FN} + N_{TN} + N_{FP}}, \tag{4.11}$$

with N_{TP} being the number of true positives, N_{TN} the number of true negatives, N_{FP} the number of false positives, and N_{FN} the number of false negatives. The *sensitivity* is defined as:

$$S_e = \frac{N_{TP}}{N_{TP} + N_{FN}}. \tag{4.12}$$

The *specificity* as:

$$S_p = \frac{N_{TN}}{N_{TN} + N_{FP}}. \tag{4.13}$$

The performance measures rely on the manual ground truth. For image segmentation in general [22], and for vessel segmentation in particular, such a ground truth is difficult to compute. While conducting hand-labeling experiments in a larger group, including students, image processing experts, and physicians, it has been noticed that the human observer usually tends to ignore very thin vessels and to enlarge thick vessels. Therefore, some false positives may still be true vessels, and some false negatives may actually be background. However, such problems are at least partially alleviated by using publicly available databases to conduct comparisons among vessel-segmentation methods.

4.3.1 Results

Table 4.1 and Table 4.2 contain the results for the two databases. All results are average values over all test images in the respective database. Some classification examples are shown in Figure 4.6. Table 4.1 contains results obtained by some state-of-the-art vessel-segmentation methods on the DRIVE database, as reported in the respective articles. The sensitivity and specificity entries for the other methods (except for [131]) have been taken from the work of

Classifier type	AROC	A_c	S_e	S_p
Hysteresis absolute	0.9642	0.9484	0.9053	0.9517
Hysteresis relative LCP	0.9713	0.9509	0.9086	0.9580
Hysteresis relative LDA	**0.9726**	0.9516	**0.9094**	0.9591
Soares et al.(2006) [169]	0.9614	0.9466	-	-
Staal et al.(2004) [172]	0.9520	0.9441	0.7194	0.9773
Jiang and Mojon(2003) [105] (from [117])	0.9327	0.8911	-	-
Mendonca and Campilho(2006) [138]	-	0.9463	0.7315	0.9781
Ricci and Perfetti(2007) [155]	0.9633	**0.9595**	-	-
Lam et al.(2010) [117]	0.9614	0.9472	-	-
Marin et al.(2011) [131]	0.9588	0.9452	0.7067	**0.9801**
Second observer	-	0.9473	0.7761	0.9725

Table 4.1: Results achieved on the DRIVE database by different segmentation methods. Values written with bold characters represent best results.

Mendonca and Campilho [138]. The specificity is not directly mentioned there, but instead the false positives fraction (FPF), defined as the fraction of pixels erroneously classified as vessel points that is interpreted as $FPF = 1 - S_p$.

4.3.2 Discussion

In the following, the various hysteresis classifiers are discussed. Relative hysteresis methods show better results than absolute hysteresis methods. The LDA-based relative hysteresis classifier shows the best performance on the test images. On the DRIVE database, with respect to sensitivity at a significance level $\alpha = 0.02$ and with respect to specificity at a significance level $\alpha = 0.005$, it is significantly (see Appendix B) better than the LCP-based relative classifier. The observations with respect to the AROC and the accuracy remain valid on the STARE database as well. However on this database, the LDA-based relative hysteresis classifier is more specific and less sensitive than the LCP-based relative hysteresis classifier. This means that it will yield a smaller number of correct classifications but also less false positives. The LCP-based relative hysteresis classifier has a slightly more pronounced tendency to oversegmentation in comparison to the LDA-based relative classifier. This is interpreted as an indication that the explicit Gaussianity assumption made in the case of the former classifier is less likely to hold by comparison to the implicit Gaussianity assumption made in the case of the latter classifier. Also, considering that for the LCP-based relative classifier the smallest scatter matrix is used to describe the variance in the both classes, this shows that the class-conditional covariance matrices are dissimilar.

As the results in Table 4.1 show, the hysteresis methods have the best sensitivity from among the automatic segmentation methods, but they tend to trade this in for specificity, thus being more likely to return slightly oversegmented results. Nevertheless, they show a very good accuracy, which means that the tradeoff is worth it. The hysteresis classifiers perform better than other vessel-segmentation methods with respect to the AROC. Only the SVM-based method by Ricci et al. [155] has an improved accuracy, but a smaller AROC. This shows that the training algorithm is not yet optimal – during experiments the training was halted after a predetermined

Classifier type	AROC	A_c	S_e	S_p
Hysteresis absolute	0.9650	0.9569	0.8879	0.9652
Hysteresis relative LCP	0.9757	0.9574	*0.8907*	0.9654
Hysteresis relative LDA	**0.9791**	0.9595	0.8902	0.9673
Soares et al.(2006) [169]	0.9671	0.9480	-	-
Staal et al.(2004) [172]	0.9614	0.9516	0.6970	0.9810
Jiang and Mojon(2003) [105] (from [117])	0.9298	0.9009	-	-
Mendonca and Campilho(2006) [138]	-	0.9479	0.7123	0.9758
Ricci and Perfetti(2007) [155]	0.9680	**0.9646**	-	-
Lam et al.(2010) [117]	0.9739	0.9567	-	-
Marin et al.(2011) [131]	0.9767	0.9526	0.6944	**0.9819**
Second observer	-	0.9351	**0.8949**	0.9390

Table 4.2: Results achieved on the STARE database by different segmentation methods. Values written with bold characters represent best results. Values written with italic characters represent best results achieved without human intervention.

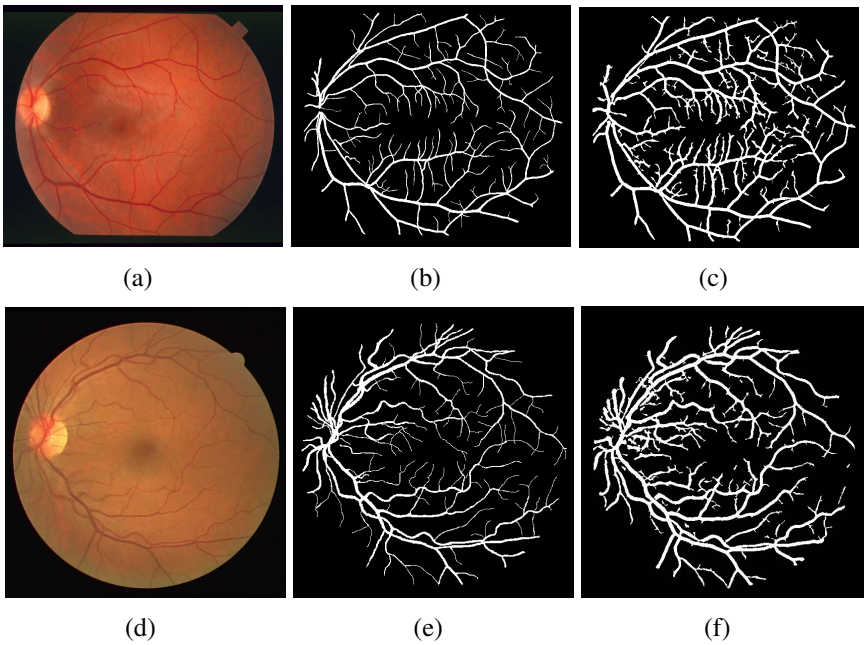
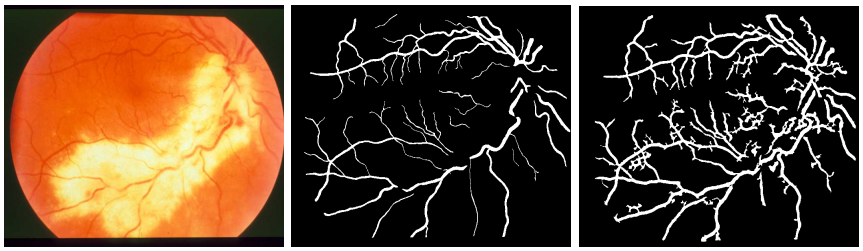


Figure 4.6: Image of the STARE database (a), the corresponding ground truth (b) and segmentation result achieved by the relative hysteresis classifier (c). Image from the DRIVE database (d), ground truth (e) and result of segmentation by the relative hysteresis classifier (f).



(a) (b) (c)

Figure 4.7: Result on a pathological image: input image (a), manual ground truth (b) and segmentation result computed with the relative LCP-based hysteresis classifier (c).

number of steps. However, the strategy based on computing pixel-features from vessel maps is most promising, because theoretically, the achieved results could be further improved by improving the separability in the feature space, i.e., designing better vessel maps.

The STARE database includes relatively many images showing pathological retinas. The vessel-map based hysteresis approach is not designed to handle such images. Nevertheless, it is relatively successful in some cases. Such an example is shown in Figure 4.7. The strong background variation caused by the large bright spot is reduced during feature extraction to a level that does not afflict the segmentation. Large bright background structures can be eliminated in vessel maps computed for example with the help of morphological image processing algorithms.

On a machine with a Core 2 Duo E6700 processor and 2 GB memory under MATLAB, the training time for a relative LDA-based classifier was one and a half hours on the DRIVE data set. The time needed to reach a result by the same classifier in the test and operation phase is 0.8 seconds per image from the DRIVE database. The time needed to compute the pixel-feature vector set for the same image is six seconds. Therefore a new image is segmented every 6.8 seconds. The absolute hysteresis classifier is a few milliseconds faster. Hysteresis methods yield classifiers that are significantly faster than other state-of-the-art methods. An overview of running times for various vessel segmentation methods is given in Table 4.3.

Supervised vessel segmentation can be achieved also by a general-purpose classifier like an SVM working on the pixel-feature space described in detail in Reference [36]. However, the strong overlap has various negative consequences. First, there will be a rather large number of support vectors, therefore, it should take a relatively long time to segment an image. There is also the danger of overfit, particularly if the size of the sample used for training is too small. Conversely, if the entire available training database is used (i.e., all pixels from all images in the training set), the time needed for training is very long, which makes designing the SVM a tedious job. Ricci et al. [155] successfully use a linear SVM to conduct vessel segmentation (on a different feature space). For training they used only 20000 pixels randomly chosen from all images in the training set. There is no mention of the time needed to segment one image in their paper, except for a comment that the method should be very fast, being based on a linear SVM. Marin et al. [131] propose neural networks (NNs) to segment vessels on a newly for this purpose designed feature space. Their feature space bares resemblance to ours, as it assigns to each pixel a feature vector consisting of intensity-level-based features, but also of moment-invariants-based features. For training they use a set of 30000 manually selected pixels from the DRIVE database, fairly divided between vessels and background.

	Training time (h)	Execution time (s)	PC configuration	Software
relative LDA	**1.5**	**6.8**	Core 2 Duo, 2.66GHz, 2GB RAM	MATLAB
Soares et al. [169]	9	180	Athlon XP, 2.17GHz, 1GB RAM	MATLAB
Staal et al. [172]	–	900	P3, 1GHz, 1GB RAM	–
Jiang and Mojon [105]	–	∼19	P3, 600MHz	C
Mendonca et al. [138]	–	150	P4, 3.2GHz, ∼1GB RAM	MATLAB
Lam et al. [117]	–	780	Core 2 Duo, 1.83GHz, 2GB RAM	MATLAB
Marin et al. [131]	–	93	Core 2 Duo, 2.13GHz, 2GB RAM	–

Table 4.3: Running times for different vessel segmentation methods on the DRIVE database. Values written with bold characters represent best results.

4.4 Conclusions and summary

In this chapter, the hysteresis classification paradigm for binary classification was discussed in a unitary manner and new powerful relative hysteresis classifiers were introduced. The hysteresis paradigm returns methods that are well suited to solve problems afflicted by large class skew and strong overlap. For this it makes use of available prior knowledge. The paradigm is rooted in the field of image segmentation and analysis, and was demonstrated successfully for the segmentation of retinal vessels. In this case there is a large class skew, because there are a lot more background pixels than vessel pixels and there is a strong overlap as the vessels and the background are inhomogeneous. The paradigm successfully uses the prior knowledge that the vessels are connected structures.

Hysteresis segmentation can successfully segment objects of inhomogeneous intensity-level representation found on an inhomogeneous background, as long as there is a slight difference between object and background at a local level around the object's borders, and the supports of the two classes in the pixel feature space do not overlap completely. These conditions should be enforced during feature extraction also. For the particular problem of vessel segmentation, the feature-extraction process was designed such that the vessels are not too thick when segmented by the optimist. This was done mainly by taking care to improve the contrast of vessels at their borders.

The case of pathological retinal images is not treated separately here. In general the performance of the methods described above on pathological images is worse than on normal images. However, the hysteresis paradigm can be used to handle such problems as well. In such a case, special vessel maps need to be devised to improve vessel separability on images with specific pathologies. This represents an indication of the versatility of the vessel-map-based feature-extraction process. The feature extraction is aimed at ensuring the border separability constraint and improving the overall separability.

For retinal vessel segmentation, the hysteresis methods tend to trade specificity for sensi-

tivity. However the good accuracy obtained shows that it was a good trade and that the danger of oversegmentation is small. Furthermore, these considerations need to be weighted by the fact that manually labeled gold-standard images used to compute these quality measures may themselves by afflicted by errors.

Requiring the pessimist to have a zero false positives rate is practically equivalent, in the case of a linear classifier, to selecting that region in the feature space A that intersects with the support of the object class and it can be separated from the support of the background class by a linear hypersurface. If only nonlinear separation is possible, then a nonlinear classifier is needed as pessimist.

Sometimes, the set of separating surfaces computed by percentiles is too sparse such that the optimist and the pessimist can not be defined correctly. For the pessimist, this means that there is no percentile-based separating surface to select only object points, although such a separating surface exists. For the optimist, this means that there is no separating surface to fulfill the border-separability constraint, although such a separating surface exists. Then, instead of using percentiles, one should use other quantiles (larger than 100) which yield a sufficiently dense set of separating surfaces.

Even though it has been demonstrated here only for vessel segmentation, the hysteresis paradigm is a more general method, readily applicable to other types of segmentation and binary-classification problems as well.

Chapter 5

Person identification and event detection

After introducing in the previous chapters new methods related to the two main parts of any pattern recognition system, namely, feature extraction and classification, we are now ready to dwell into the "raison d'etre" of such a system, i.e., *the applications*. Pattern recognition systems are designed for specific applications and in this work the accent lies on the particular case of security applications, more precisely person identification and event detection.

In the case of person identification two applications are discussed, which differ with respect to the biometric characteristic used to conduct the identification, namely: fingerprints and vascular nets, more precisely retina vessels. When conducting person identification using vascular nets, the relative LDA-based hysteresis classifier described in Section 4 is used for the purpose of vessel segmentation.

In the context of security applications, event detection is mostly encountered while conducting surveillance. Accordingly, two simulated scenarios are discussed, which are related to: person surveillance, and traffic surveillance. Besides complete algorithmic solutions to such system applications, novel general-purpose event-detection algorithms are introduced as well. Thus, several novel event-detection algorithms of different complexity levels will be introduced with the help of an exemplary medicine-related surveillance application.

Next, the focus will be set on two main topics: the sparse classifier, and stochastic methods for event detection. In Section 5.1 it is argued that specifically the sparse classifier is very well suited for security applications and it is shown how to use it for the purpose of both person identification and event detection. In Section 5.2 we will then concentrate on the analysis of random signals for the purpose of detecting events. In this context, besides new significance-test-based methods, a novel linear-predictors mixture is described and the conditional random fields are adapted for event detection.

5.1 Sparse classifiers for identification and surveillance

The sparse classifier is particularly robust to noise afflicting a large portion of the feature vector. This type of disturbance is often encountered in biometric applications, where, for example, a person may use sun glasses to mask his eyes and make the task of recognizing his identity more difficult for a face-recognition system working appearance-based[1] [187]. This property remains

[1] An appearance-based system uses the image directly as feature vectors. It uses thus only the raw features, renouncing any transform-based feature extraction.

useful for systems using a feature extraction transform as well, in particular when this is a unitary transform and the disturbance is such that it cannot be captured in just a few transform coefficients. Such is the case of poor fingerprints collected from various items, as for example a fingerprint collected from the pages of a book. By its ability to deal with noise, but also by its other characteristics, like robustness in particular with respect to a small-size training set, the sparse classifier is well suited for person identification tasks and we show in Section 5.1.1 how it can be applied to fingerprint-based person identification and in Section 5.1.2 to retina-based person identification.

At the same time, the sparse classifier has the inbuilt ability to detect outliers with the help of the SCI (see Section 2.3.2), which makes it well suited for the task of event detection. Furthermore, events are by definition sparse and it seems only natural to use sparse methods to detect them, as described in Section 5.1.3.

5.1.1 Fingerprint-based person identification

The pattern of ridges and valleys of the skin covering the interior side of distal phalanges (at the tip of a finger) represents a fingerprint. The fingerprints represent unique characteristics of each human being and as such have been the biometric feature of choice for a long time. Currently they receive attention from the machine vision community in the quest for automatic person-identification systems to be used for various purposes ranging from access control to electronic banking.

A typical fingerprint identification algorithm consists of a feature extraction step followed by a classification step. During classification, the query fingerprint is assigned to one of the available classes – one for each person in the target group – of enrolled fingerprints. The fingerprints are considered to be available in the form of digital images coming either from a sensor or from digitized latent fingerprints. The latent fingerprints are collected by forensic modalities from various items. Person identification with respect to a target group/database is achieved only if the classification can be conducted with sufficient confidence.

There are several types of features that can be extracted from a fingerprint [104]. Level-1 features are related to general characteristics of the fingerprint, like the location of singular points (e.g., points characterized by large ridge curvature). Level-2 features include more particular characteristics, these so called minutiae features are, for example, the locations of ridge bifurcations and endings. Level-3 features are related to fingerprint details like ridge width, edge contours or pores.

The classification step is usually conducted with the help of either image correlation, phase matching, skeleton matching or minutiae matching [104]. Minutiae-based approaches are most commonly used [30], [29]. They mimic to a certain extent the way a human expert usually goes to work on classifying a fingerprint.

It is considered that for fingerprint identification Level-2 and Level-3 features are needed [104], and under such circumstances, there are many effective solutions to the problem of fingerprint identification [130]. However, the problem space is not yet fully covered, as there are still some unanswered questions that lead to new research opportunities [104]. Major difficulties are encountered when working with poor quality fingerprints, yet another difficulty is represented by the small overlapping area between a query fingerprint and the enrolled fingerprints. Fingerprints are additionally afflicted by nonlinear distortions and in the case of latent fingerprints, a complex background. Under such circumstances, the extraction of fingerprint

features of Level-2 and Level-3, and hence fingerprint identification by methods based on these types of features, becomes challenging.

Here, a fingerprint identification method is proposed that works despite the above mentioned difficulties. However, it is assumed that for each enrolled user, at least a few fingerprints of one finger are available.

From the above-mentioned feature categories, only one Level-1 feature is used, i.e., the location of the *core point*. The core point is the most central point of the fingerprint, around which the ridge orientation changes rapidly [102], [123]. The core point is used as reference to select around it a region of interest (ROI) of the fingerprint image. The ROI serves the purpose of concentrating the analysis on the same area of the fingerprint, irrespective of how it was positioned over the imaging sensor. From the ROI, a feature vector is extracted that includes a certain selection of Discrete Cosine Transform (DCT) coefficients. The feature vector concentrates information that is less-likely to be afflicted by image noise, difficult backgrounds or scars and scratches. Such information is related to the global pattern of a fingerprint.

A query fingerprint is then assigned to a specific finger from the target database, with the help of a sparse classifier that works well despite the availability of a very small number of training samples per class, which is usually the case for fingerprint identification. The sparse classifier has been previously used in the context of fingerprint analysis, but for the purpose of pore-matching in high-quality fingerprints [122]. The sparse classifier naturally offers the possibility to compute the confidence in the computed result. If this confidence is not high enough, the identification of the respective fingerprint should be conducted – in the case of poor-quality fingerprints – by a human expert.

To correctly identify a finger despite small fingerprint overlap area, or the availability of a partial or partially corrupted fingerprint, the abilities of both the sparse classifier and the feature vector to handle occlusions are harnessed. To handle distortions, it is assumed that, in general, the training set includes fingerprints with the usual linear and nonlinear geometric transformation caused by the acquisition procedure. At the same time, the feature extraction is designed to offer a feature vector with additional invariance properties to some geometric transformations but also to point-transformations of the fingerprint-image's gray levels.

The novel concepts introduced in this section are the following: first, a new definition of the core point is given and methods are described to detect it; second, a transform-based feature vector is defined that captures basic fingerprint-information that is available even in the most difficult fingerprint images; and third, the sparse classifier of Section 2.3.2 is introduced for the problem of fingerprint identification.

The methods described here can be combined to construct an automatic fingerprint identification system for low-quality fingerprints, like for example those acquired in forensic applications. When combined in such a system they are meant to support a human expert at identifying a fingerprint with the help of a database of fingerprints. By tuning the confidence rate of the system, a human operator can decide what percentage of the available database should be investigated automatically at a certain accuracy.

Feature extraction

For each fingerprint image, from a ROI placed at the core point, a feature vector is computed. The sparse classifier then assigns a feature vector to the class whose training vectors span the subspace closest to it. For sparse classification, the dimension of the feature space is also

(a) (b) (c)

Figure 5.1: Various types of fingerprints: loop (a), whorl (b) and plain arch (c).

important. It needs to be chosen in relation to the number of training samples and the number of representatives per class (see Section 2.3.2).

To detect the core point, the orientation of the ridges is used. The features used are some DCT coefficients of the core-point-centered ROI. The particular choice of DCT coefficients offers mild rotation and translation invariance and at the same time robustness to changes in the mean of the fingerprint image. To find out which DCT features to include in the feature vector a feature selection procedure [84] is applied.

The detection of the core point. The core point is currently defined as the point of maximal curvature of the concave ridges of a fingerprint [102]. Initial approaches to detecting the core point were based on the Poincaré index [110]. They work well only for good-quality fingerprints. To detect the core point in poor-quality fingerprints, robust methods are needed. Such methods are proposed in [102], [11], [123] but they have difficulties with arch-type fingerprints (see Figure 5.1), due to the definition used for the core point.

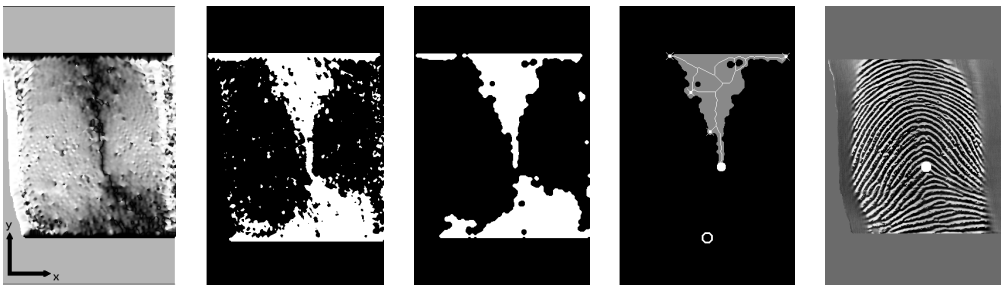
Under such circumstances, a new definition of the core point is proposed next. Under the assumption that the available fingerprints are approximately vertically oriented, it may be observed that starting from the top of the fingerprint and going down, the ridges are less and less flat – whereby flat means similar to a horizontal line (see Figure 5.1). The core point is then defined as the point where such ridge flatness becomes minimal.

Next ridge-flatness is measured by the sine of the angle between the orientation vector and the x axis in a fingerprint image (see Figure 5.2 (a)). The orientation vector is evaluated from a local neighborhood Ω at each pixel as the eigenvector of the orientation tensor corresponding to the minimal eigenvalue. The orientation tensor is computed as [3]

$$\mathbf{J} = \begin{bmatrix} B(R_y \cdot R_y) & B(R_x \cdot R_y) \\ B(R_x \cdot R_y) & B(R_x \cdot R_x) \end{bmatrix},$$

with B being a smoothing kernel whose impulse response is related to Ω, and R_i being the directional derivative in the direction i. Therefore, the grey-level variation along the orientation vector is minimal.

The sine image (with the values of the sine function of the orientation angle at each pixel) is considered to exhibit two classes, one for orientation angles close to $\frac{\pi}{2}$ and one with orientation angles close to zero. Using Otsu's method for unsupervised binary classification [147] – that



(a) (b) (c) (d) (e)

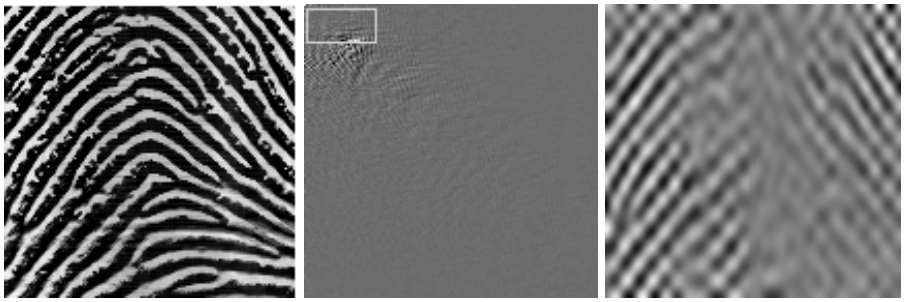
Figure 5.2: Processing chain for core-point detection: sine image (a), binarization result (b), breaking at the core-point region (c), upper central object-part with skeleton and detected terminations – the core point is marked by a disc and the reference position by a circle (d), final result (e).

optimizes a separability measure – the sine image is binarized, defining labels such that the class of pixels with orientation close to zero represent the object and the rest the background (see Figure 5.2 (b)).

The binarized sine image is then morphologically processed [65] to detect the core point. First, it is broken into two parts at the region of the core point by eroding with an appropriate disc-like structuring element of diameter d_1 and then dilating with a similar structuring element but with a diameter $d_2 < d_1$ (see Figure 5.2 (c)). Then the object region in the upper central part of the image is selected. For this purpose the object pixels in the upper quarter (along the y axis) of the image are used as seed points. As the pixels may be divided among several object structures in the upper quarter, only those pixels belonging to the structure closest to the image center are considered. All object points linked to these over an eight-neighborhood (i.e., all neighboring pixels situated at an Euclidian distance $d \leq \sqrt{2}$) are selected. Then, the morphological skeleton of the result is computed and the termination points (see Figure 5.2 (d)) are detected. The (usually) lowest most central termination point is the sought core point. We find this point as the termination point closest to the empirically established reference position given by $\left[\frac{5m}{6}, \frac{n}{2}\right]$, with m the number of lines and n the number of columns in the image.

The feature space. The purpose followed here is to design a system robust to partial occlusions of the fingerprint as well as to partial and total corruptions (like e.g., overlays). Therefore, the feature vector must also exhibit such properties. At the same time, we would like to reduce the number of dimensions with respect to the size of the training sample to avoid additional problems related to the curse of dimensionality. Therefore, in contrast to other appearance-based methods [102] transform features will be used next. LDA and other related methods are not appropriate, because we want a method unrelated to the number of classes – i.e., the number of fingers in the query database. The PCA is of limited use, as there are hardly any common high-eigenvalue variation modes, considering how dissimilar the various types of fingerprints are (see Figure 5.1). Furthermore, with PCA we would theoretically need to recompute the transformation matrix with each new finger enrolled in the database. Thus the DCT will be used. The precise DCT coefficients to use, are found by feature selection.

The feature vector is computed from an ROI centered at the core point (see Figure 5.3 (a)).



(a) (b) (c)

Figure 5.3: ROI (a), DCT coefficients – the marked region gives the feature vector (b) and ROI reconstructed with the selected coefficients (c).

The size of the ROI should be chosen in relation to the resolution of the fingerprint imaging device. For some fingerprint images the ROI may go over the boundaries of the image. Then, the corresponding region is filled with zeros, being thus treated as a total occlusion. The vector contains DCT coefficients of the ROI from a certain region of the DCT space (see Figure 5.3 (b)).

By eliminating the coefficients corresponding to DC and very low frequencies, the feature vector is invariant to changes in the mean of the gray-levels of the fingerprint image. The specific choice of DCT coefficients represents a sub-sampled, alias-pledged representation (see Figure 5.3), that in comparison to the original ROI is largely invariant to translation and has mild rotation invariance properties, as shown below. The feature vector contains thus mainly information on the global pattern of a fingerprint, the most general and often available fingerprint information.

Feature selection. Assuming a small labeled data set is available, it may be used to conduct fast feature selection, as described in Reference [84]. The objective function is the squared training error of a set of linear discriminant functions with one function per class. The procedure starts from a vector with k DCT coefficients randomly selected from the DCT space with a total of $N \gg k$ coefficients. For each selected coefficient, the objective function for a feature vector including it with that obtained for a feature vector excluding it are compared. At the same time, the coefficients are ranked by the value of the difference between the two objective functions. The least relevant coefficient is eliminated and replaced with another one randomly chosen from the remaining $N - k$ coefficients. The procedure is repeated q times. After q repetitions, the better coefficients will be selected more often (see Figure 5.5). The final feature vector is computed from that region of the DCT space, which has the coefficients most often selected during feature selection. This region of feature concentration is selected, instead of using precisely the selected features, because it can not be assumed that the data sample used for feature selection is optimally representative for the random variable feature vector.

Mild rotation invariance. Under mild rotation invariance it is understood here that for relatively small rotations, the selected DCT coefficients of the original and the rotated image differ insignificantly. To better understand what this means, the analogy of a fingerprint to a planar wave will be used. It can be observed that a planar wave of frequency ω_1 is less variant

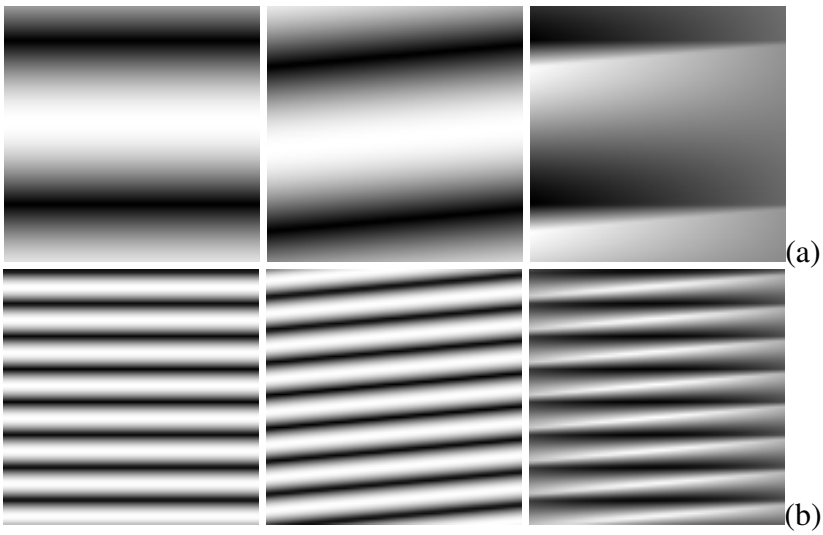


Figure 5.4: Image (first column), its rotation (second column) and their difference (third column) for a low-frequency planar wave (a) and for a higher-frequency planar wave (b).

to rotations than a planar wave of frequency $\omega_2 > \omega_1$, when both are considered within the same finite area. This is illustrated in Figure 5.4 for a rotation of five degrees (with the center of the coordinate system at the center of the image). The squared error between the original and the rotated image increases with the frequency of the corresponding planar wave.

A similar phenomenon takes place as well in the case of the DCT-based feature vector. It contains a low-frequency representation of the ROI, similar to the low-frequency planar wave form above, while the original ROI behaves similar to the higher-frequency planar wave.

Experiments and discussion

In the last decade fingerprint-based biometric applications have received a lot of attention from the machine vision community. Fingerprint Verification Contests (FVCs) have been organized in 2000 [129], 2002, 2004 [30] and 2006 [29]. These are mainly geared towards fingerprint verification and not fingerprint identification/recognition as is the case with this contribution. Therefore, a direct comparison is not possible.

Furthermore, the sparse classifier offers the possibility to default a decision in favor of a higher-accuracy classifier or of a human observer, if the confidence is not high enough. This has a major influence on the way the algorithm described above is used. The test setup is designed to take into account such particularities.

Experiments are conducted on the DB3 database. This database has been used within the FVC 2004 [30] and is currently available online. The DB3A database contains 800 fingerprints, eight fingerprints for each of 100 different fingers and is meant for test purposes. The DB3B database contains 80 fingerprints, eight fingerprints for ten different fingers. They have been acquired with a thermal sweeping sensor (Atmel FingerChip), at a resolution of 512 dpi and an image size of 300×480 pixels. To test the algorithm the data from DB3A is divided into eight equal parts, each part containing 100 different fingerprints and an eight-fold cross validation is conducted. The establishment of various parameters for the method has been done with the help of DB3B. The size of the ROI was 141×141 pixels.

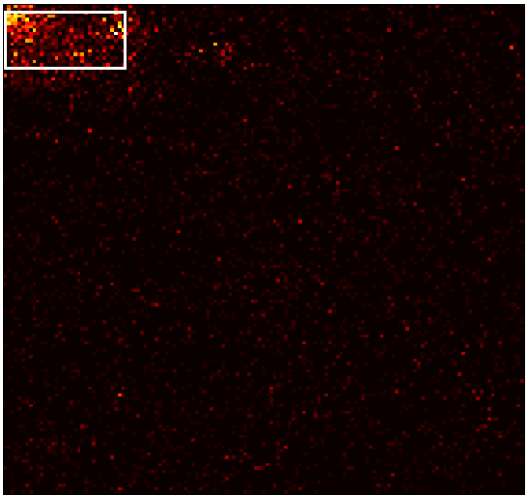


Figure 5.5: Results of feature selection. The chosen coefficients are those in the white rectangle. The brighter a pixel is the more often the DCT at the corresponding position has been selected.

For the algorithm described here, the enrollment time for a fingerprint is approximately 0.4 seconds. This includes 0.39 seconds for detecting the core point and 0.016 seconds for computing the feature vector. The match time for a fingerprint with respect to a database containing 700 fingerprints, i.e., seven fingerprints per finger and 100 different fingerprints is 1.29 seconds, including about 0.88 seconds for classification. The experiments were conducted under MATLAB R2010b on a Core 2 Duo E6600 (2.66 GHz) machine with 4 GB of RAM.

Core-point. The parameters of the core-point detection method are: $\Omega = 11$, $d_1 = 5$, and $d_2 = 4$. The filters implementing the directional derivative had a length $l_d = 9$. For testing a manual ground-truth of core points [45] has been used. To account for human imprecisions, a core point is found successfully if it falls into a region of 21×21 pixels centered at the manual core point. Within this setup, the core point is successfully detected in 720 cases from 800.

Feature vector. To find out which DCT coefficients are optimally suited, feature selection has been conducted using DB3B. The number of potentially available DCT coefficients is $N = 141^2$. For feature selection, $k = 500$ coefficients, and the number of iterations was $q = 1500$ times. The results are shown in Figure 5.5.

The brighter the pixels, the more often the respective DCT coefficient was selected. The DCT region of the feature vector corresponds with the concentration region of the most selected DCT coefficients. The region R_b has the size 17×35. Thus, a feature vector with $d_{R_b} = 595$ dimensions is obtained. Experiments have also been conducted with the DCT coefficients from a smaller region R_s of the size 17×17, such that it covers approximately the left half of R_b. With the resulting vector of $d_{R_b} = 289$ dimensions, similar identification results are obtained. For the experiments, a number of $n = 700$ fingerprints, corresponding to 100 fingers with $t = 7$ fingerprints per finger have been used, therefore in both cases (i.e., for both R_b and R_s) d respects the bound (2.77) on the minimal number of features in relation to the number of example per class in the training set needed with a sparse classifier (see Section 2.3.2).

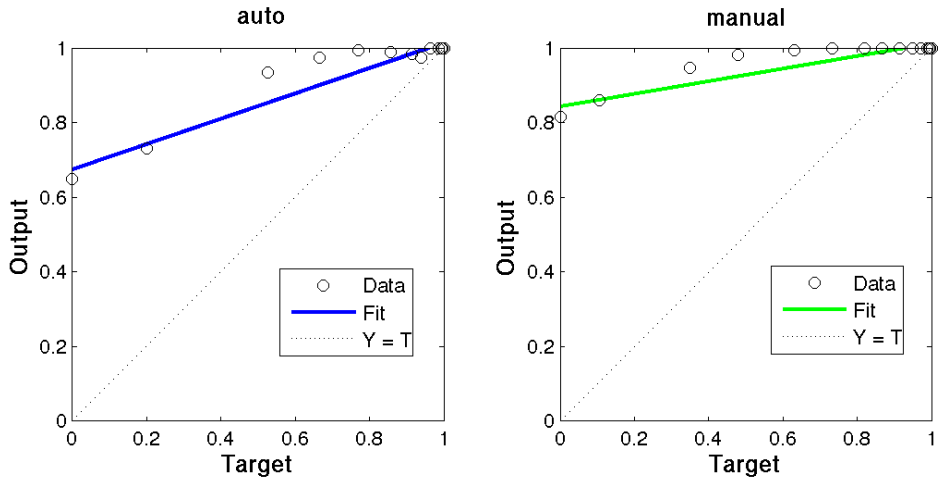


Figure 5.6: Left: the ROC for automatically detected core-points and right: the ROC for manually detected core-points. Target gives the *"uncertain"* rate and output the *"correct decision"* rate.

Finger identification. During eight-fold cross validation, the data has been divided into eight chunks each of 100 different fingerprints. 700 fingerprints are used for training and the remaining 100 fingerprints for testing. The procedure is repeated eight times, each time with different sets, until each set has been used at least once as test set. A type of ROC is computed by varying the confidence threshold τ between zero and one in steps of 0.01 and computing for each threshold the mean rate of correct decisions and the mean rate of "uncertain" decisions, where the means were taken over the cross-validation results

To investigate the dependancy of this method on an accurate detection of the core-points, two ROCs have been computed. For the first, features computed based on the set of automatically detected core points are used and for the second those based on the manually selected core points. The results are shown in Figure 5.6. As it can be seen, the accuracy of the detection of the core point influences the results up to an "uncertain" rate of 0.5.

80% correct decisions is considered here to be the minimal rate for successful fingerprint identification from poor-quality fingerprints. Using the automatic core points, 80% correct decisions is obtained for a 30% "uncertain" rate. This means that from the 70% of the data for which a decision is obtained, 80% are correct decisions. With manual core-point detection, the 80% correct decision rate is reached for 0% "uncertain" rate.

To simulate bad fingerprints, the images have been low-pass filtered using a Gaussian kernel. The performance of the method described here remains unchanged when using filters up to a minimal 3dB bandwidth of $\frac{\pi}{6}$. The method can handle occlusions up to 30% of the ROI. Such occlusions are encountered in the used database for those fingerprints whose core points were placed near the borders of the analyzed image, such that only about 70% of the ROI was filled with fingerprint information, the rest being filled with zeros. As long as the core point was correctly detected, all these images were correctly classified.

Besides the DCT, several other feature extraction methods have been tested on DB3B, these are: the LDA, the PCA, and also a feature vector that contains a downsampled (by a factor of three) ROI. As expected, none of them worked well, with correct-decision rates of under 15%

for $\tau = 0$.

Summary, conclusions, and outlook

In this section, a fingerprint-identification framework was discussed, which is designed to work with poor-quality fingerprints. Under these circumstances only one of the most basic fingerprint features is used, namely the core point. At the same time, a feature vector is built that captures only part of the information found in the ridge-pattern of a finger, namely that part that is imprinted on a grabbed item under most difficult conditions for the subsequent fingerprint acquisition. Therefore, the feature-extraction process yields a feature vector that is not particularly rich in information, and to obtain satisfactory results, we need to compensate for this in the classification phase. By its properties the sparse classifier is optimally suited to work under such conditions. It works well with a less-informative feature vector (as long as the size of the feature space is well chosen and the training set covers most of the variability to be expected in the test sample) and with a small number of training examples per class. The system proposed here can handle occlusions or corruptions of large parts of the analyzed fingerprint.

The novel definition of the core point proposed here is able to deal with all types of fingerprints, i.e., including arch-type fingerprints, however, the particular method used for the detection of the core point leads to slightly imprecise results. This makes the core-point detection method described here not optimally suited to detect core points for fingerprint alignment (with respect to rotation and translation). The algorithms discussed here profit from the ability to work with arch-type fingerprints and are designed to be robust to such imprecision. While the method is not heavily influenced by small errors in positioning the core point, a complete failure definitely leads to a wrong decision for the analyzed fingerprint. Furthermore, the automatic core-point detection method proposed here may still be improved, as shown by the superior performance achieved when using manually-selected core points. Improved core-point detection methods and enrollment-failure detection methods that evaluate if the core point can be detected at all in a given image need to be investigated.

The feature vector contains actually a downsampled band-pass representation of the fingerprint ROI. The low frequencies related to average gray level and large, low-frequent structures, like, e.g., a cut or bruise, are ignored as are high-frequency noise structures. At the same time the ridge pattern is blurred (and undersampled). The autocorrelation of the fingerprint ROI remains high over larger rotations and/or shifts after the blurring as opposed to before the blurring. Therefore the algorithms described here are insensitive to both small translations of the ROI due to imprecise core-point detection and small rotations due to differences in placing the fingerprint over the sensor. The precise choice of DCT coefficients has been established by feature selection.

In contrast to many previous approaches, here it is implicitly assumed that several fingerprints of the same finger are available. This assumption becomes increasingly valid the more ubiquitous fingerprint-based systems become. Even if in the beginning a finger gets just one impression stored, the more often the respective person uses the system, the more impressions of the same finger – under various transformations/influences – become available. Clearly this information should be harvested. Until now it has been assumed that the query fingerprint is bad, but the enrolled fingerprints are relatively good. Should bad fingerprints also be available we expect an improvement of the results of this method.

The “uncertain“ decision, which comes naturally for the sparse classifier, should be inter-

preted as there is not enough information to make a decision with enough confidence, thus, in this case, a human observer should analyze the respective fingerprint. This is the setup for which this algorithm has been designed. Clearly, under such circumstance the combined correct decision rate of the fingerprint-based person identification framework described here will be greatly improved.

5.1.2 Retina-based person identification

The pattern of vessels supplying blood to the retina is a unique feature in each eye and can be used to authenticate an individual [168, 96]. This feature is impossible to forge and the blood vessels decay too fast to allow the eye of a deceased person to be used to deceive the system. Furthermore, the vascular pattern is virtually constant over the entire life span of the enrolled individual, the most often exceptions being pathological cases like diabetic retinopathy. Even though it enjoys such desirable features in comparison to other biometric traits, it is by far the least used. The reason for this is the acquisition procedure that is considered intrusive and requires a relatively high degree of user cooperation (for example eyeglasses must be removed). From a historical perspective there has been a tradeoff between the quality of the acquired retinal scans and the amount of data analysis needed for the purpose of identification. The first commercial retina-based person identification systems acquired high-quality images in a tightly controlled environment, for which purpose they used visible light to illuminate the retina [95]. This procedure was very uncomfortable for the user and as a consequence a near-infrared light source replaced the visible-light source shortly afterwards. Later, the amount of energy radiated by this source, as well as the acquisition time, was decreased more and more with each new retinal-scan system on the market to improve acceptance. However, as a direct consequence, the quality of the obtained images deteriorated. Thus, the improved acceptance generated the need for more powerful data analysis tools to accomplish the intended identification task. Besides a decreasing signal-to-noise ratio (SNR), the acquired retinal scans may be afflicted by geometric transforms like rotation and translation but also a small amount of scaling due to the eye movement or head placing with respect to the sensor. Clearly, retina-based person identification must be invariant to such disturbances. Additional difficulties are encountered with persons suffering from astigmatism and under some circumstances with person wearing contact lenses.

With high-quality retinal scans, conducting person identification is usually a relatively easy task. The first solution, described in Reference [96], simply used the Fourier transform for feature extraction, to deal with some of the imaging-related issues, followed by simple correlation to measure the similarity between two images. More modern approaches still use correlation [108, 132, 79, 69], but the preprocessing is different. Geometric distortions due to the image acquisition process are dealt with using polar coordinates [108], or image registration [79, 132] with vessel parts as cues. In Reference [69], a more complicated feature extraction process is used, with a polar transform followed by a wavelet-based multiresolution analysis of segmented vessels, also modified correlation coefficients over several scales are used together to reach a decision. In other approaches [8], vessel segmentation together with a simple orientation analysis on the segmentation result using an angular partition are used for feature extraction. The feature vectors thus obtained are compared with the help of the Manhattan distance. In [146, 145], two retinal images are matched based on a set of feature points of the vessel pattern. These feature points are anatomically interesting points of the vascular network like bifurcations and crossovers, that can be easily detected despite low SNR. After extracting them, the optimal transform

that registers/aligns the two feature-points clouds is found. Then, the number of landmark-pairs (i.e., pairs of feature points that are considered to be the representation of the same anatomical point in each image) is untied to compute the similarity between two images. A similar feature point-based strategy is used in Reference [119] as well.

The approach followed here is based also on feature points, extracted from the vessel segmentation result computed with a hysteresis classifier (see Chapter 4), that is particularly robust to noise. From each image one feature vector is generated. The extracted features exhibit all the invariance properties required by retina-based person identification. Unlike other approaches, the feature extraction process lifts the need to align a query image with a database image to find out if they come from the same eye, which clearly reduces the computational burden. Together with the subsequent sparse classifier based decision (see Section 2.3.2), which is very well suited considering the typical small-size training set encountered in person-identification applications, this results in a particularly robust algorithm.

SIFT-based point-cloud features for retinal vascular networks

For each retina image, a cloud of feature points is computed. The subsequent feature extraction process has as its aim to provide a vector-based description of this point cloud. Then, person identification is conducted in this feature space with the help of the sparse classifier.

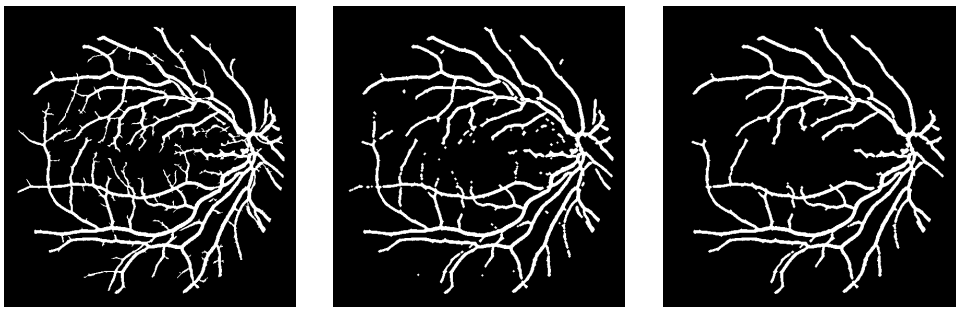
The feature-point cloud should offer a unique and compact description of the target retinal vessels. Highly informative points in this context are the vessel bifurcations, as they are related to anatomical characteristics of the vascular network [36], but also the vessel crossings, i.e., the points where a retinal vessel goes over or under another one. At the same time, such points offer a set of invariance properties, like rotation invariance, that makes them well suited for our purposes here.

Therefore, the feature points used here are the vessel bifurcations and crossings. These are detected on binary images with vessel centerlines. The vessel centerlines are computed with the help of morphological image processing methods from segmented vessels. The vessels are segmented with the help of a hysteresis classifier. After detecting the feature points, from small neighborhoods centered at the location of a feature point the corresponding Scale Invariant Feature Transform (SIFT) features [125] are computed. The SIFT feature vectors of each component of a feature-points cloud are then used to empirically estimate the corresponding covariance matrix. The final retina-image feature vector is computed from the corresponding log-covariance descriptor.

Vessel bifurcations and crossings, together with many more other interesting feature points, can be detected by applying SIFT directly to the original retinal image. However, in this case we obtain besides the sought points a large set of other points whose relationship to the vascular network is questionable (e.g., points in the background), even though their relationship to each analyzed image is strong. This in turn decreases the descriptive power of the covariance-based feature vector. This will be related more to the background than to the vascular network, and thus each retinal-vessel network will appear to be similar to any other one (see also the discussion on the "Ugly-duckling" theorem in Section 1.2). We are therefore interested in a feature-point selection procedure that is rather specific with respect to crossings and bifurcations.

Computing the feature-point cloud. As shown in Chapter 4, the hysteresis classifier together with the corresponding feature extraction procedure successfully segments the retina vessels for medical applications. In this case, special care is taken to segment small, barely detectable vessels. For security applications our interest shifts from detecting small vessels to reliably detecting crossings and bifurcations (i.e., we want to detect only points where we are highly confident that they are at a vessel crossing or bifurcation). Thus, the hysteresis classier is used in a modified feature space in comparison to the one discussed in Section 4.2. Now, instead of concentrating on small vessels, the feature extraction process is conducted with the aim of increasing the separability of mid and large-size vessels, where crossings and bifurcations can be reliably detected (see Section 4.2). Additionally, the segmentation result is morphologically processed such as to eliminate small vessels, before detecting crossings and bifurcations, again with the help of binary-morphological image processing methods.

Binary-morphological processing of segmented retinal vascular networks. Starting from a segmentation result, our aim is to obtain the feature-point cloud, i.e., the locations of crossings and bifurcations. The first step is to open the original segmentation with a disk-like structuring element of a radius larger than the smallest vessels (see Figure 5.7 (b)). After this we need to select only the main vessels. To this end successive erosion steps are applied to the result of the opening until all vessel points are eliminated and then the vessel points from the last but one erosion result are used as markers for the large vessels. More precisely, these points are used to select from the opening result only the large vessels. For this purpose all points linked to the markers over an eight-neighborhood (i.e., the 3×3 region of interest centered at the keypoint) are selected (see Figure 5.7 (c)).



(a) (b) (c)

Figure 5.7: Original vessel segmentation (a), result of the opening (b) and final result after elimination of the small vessels (c)

After eliminating the small vessels, the large ones are thinned until they are one-pixel thick. On the thinned vessels, the crossings and bifurcations are easily detected by counting the object points in the eight-neighborhood of each object pixel. If the count is larger or equal to three, then we detect a feature point. A feature-point cloud, with the corresponding vascular network, for one of the images used during experiments, is shown in Figure 5.8.

The log-covariance matrix of SIFT features. After computing the locations of the points in the feature-point cloud, the corresponding SIFT descriptor is extracted for each component

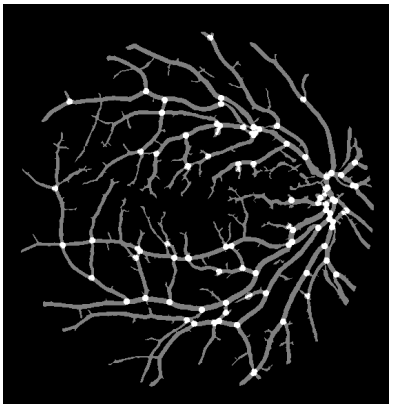


Figure 5.8: Feature-point cloud (white) and the corresponding vascular network (gray).

of the cloud. For this purpose the SIFT is applied to the retinal image and from the detected keypoints only the descriptors of those in the feature-point cloud are selected. The SIFT descriptor provides a unique and largely invariant representation of the local neighborhood of the corresponding point [126]. This representation is based on the local orientation [3] including gradient-vector magnitudes and angles with respect to the Cartesian coordinate vectors, encoding also angles relative to each of the present orientations. By local orientation we understand that the orientation is analyzed in a certain region of interest centered at the respective keypoint. The size of this region of interest is related to the scale at which the keypoint can be optimally described.

The final feature vector is related to the statistical properties of the SIFT-descriptor sample in the feature-point cloud. Working this way leads to robustness with respect to potential outliers, for example, in the form of cloud components that are neither bifurcations nor crossings, or spurious cloud components, like those appearing when new vessels are formed due to pathological reasons (e.g., diabetic retinopathy).

The size of the final feature vector depends on the size of SIFT descriptor. Depending on the maximal number of examples per class, i.e., the maximal number of available retinal scans of an enrolled person, the minimal size of the final feature vector can be established such as to ensure the appropriateness of the the sparse-classifier framework for this problem. According to the theory of the sparse classifier (see Section 2.3.2) the number of examples per class c is related to the dimension of the feature vector m, while at the same time the $m \times n$ training matrix $\mathbf{T}$ usually needs to be underdetermined, such that $n > m$. As the SIFT descriptor is rather large, its dimension is reduced with the help of PCA. This has the benefit of both an improved fit in the sparse-classifier framework and of a smaller computational burden. The PCA is computed form all SIFT descriptors from all images in the training set only once.

The final feature vector is computed from the diagonal of the log-covariance matrix [85] that is estimated from the available PCA-transformed feature-point cloud SIFT descriptors. The log-covariance matrix is the reconstruction of the covariance matrix from its eigen decomposition where the eigenvalues have been replaced with their natural logarithms. This procedure is needed to ensure that the feature space is a true vector space, such that linear combinations of vectors from this feature-space lead to a vector that is itself a component of the feature space, i.e., the feature space has the property of being closed under linear combinations, and the sparse

classifier may be used here. This type of feature extraction offers an additional set of invariance properties to some simple transformations of the SIFT descriptors.

Experiments and discussion

The qualities of this retina-based person identification system are demonstrated on two databases. The first database is a publicly available database, the VARIA [100], the second one was constructed on purpose for such applications from the DRIVE database [172] (see also Section 4.3).

The VARIA database has 233 images from 139 different individuals, out of which 59 had two or more samples. The optic-disc centered images have been acquired over a period of several years with a TopCon NW-100 model non-mydriatic[2] retinal camera at a resolution of 768x584. These images have a high variability in contrast and illumination.

The DRIVE for Retinal Authentication (DRIVERA) database [42] contains 280 images. These were generated from the 20 test-set images of the DRIVE database, with a resolution of 576×560 pixels. For this purpose for each DRIVE image 14 more images have been created while applying various types of distortions. These distortions are supposed to simulate different image acquisition-related problems that may appear when the same retinal vasculature is imaged at different times.The distortions were divided into three categories: person-related, optics related, and sensor related. The person-related distortions are small rotations, translations and scalings of the original image, the optics-related distortion are blurring, barrel and pincushion transforms applied to the original image and the sensor-related distortions are changes in illumination, white and "salt&pepper" noise. The new images were generated by randomly applying these distortions, such that an image may be affected by one or by several such distortions. The original DRIVE images are all color images, in the DRIVERA database there are only gray-level images obtained by selecting and then processing only the green channel of each original image. The way the DRIVERA database was generated is depicted in Figure 5.9.

During morphological processing, the diameter of small vessels is needed, such as to eliminate only these from the original segmentation. This depends on the hardware used for image acquisition. For example for the images of the STARE database, this was two pixels. To compute the SIFT descriptors of the feature-point cloud, the SIFT transform is applied to the entire analyzed image and only those SIFT keypoints are selected that correspond to points in the cloud. Usually, the positions of feature points in the cloud can be found among the SIFT keypoints, should this not be the case, we simply take the closest SIFT keypoint to a cloud point.

To validate both the feature extraction and the classification process, tests have been conducted in three scenarios: (i) with manually selected crossings and bifurcations (manual), (ii) with SIFT keypoints (SIFT), and (iii) with segmentation-based, automatically selected crossings and bifurcations (segmentation). In each scenario three types of classification algorithms have been applied: the first one is based on the sum of squared differences between the scale-aligned query image and the train-set image (SSD); the second one is based on the number of landmark pairs between the query image and the train-set image, similar to the algorithm used in [145] (RANSAC); and finally the third one is the sparse classifier of Section 2.3.2 using the covariance-based feature vector (Sparse). The leave-one-out cross-validation results are shown in Table 5.1 for the DRIVERA database and in Table 5.2 for the VARIA database.

As previously discussed, for the sparse classifier, the dimension of the feature space must be chosen in accordance with the maximal number of examples per class in the training set. For the

[2]Non-mydriatic cameras can image the retina without induced dilation of the pupil.

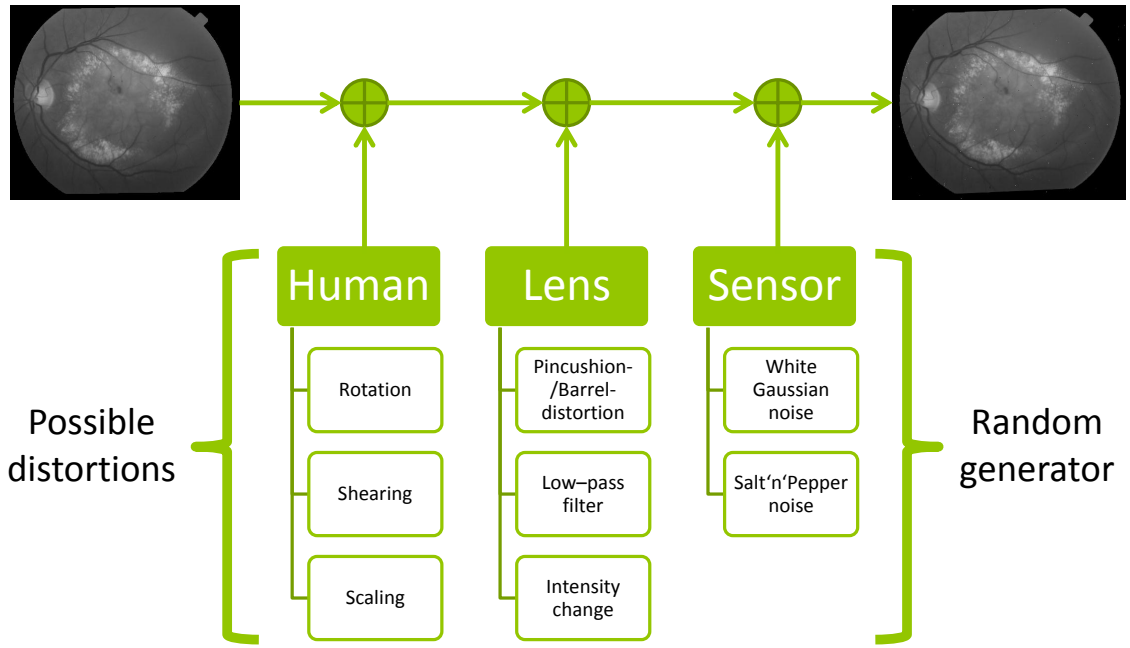


Figure 5.9: Generation procedure for the DRIVERA database.

VARIA database this is $c = 7$ and for the DRIVERA database this is $c = 14$. At the same time the dimension of the feature space must be smaller then the total number of examples in the training set. Cross-validation experiments on half of the DRIVERA database led to choosing the dimension of the feature space to be $m = 57$ that was also used for the VARIA database. All classifiers were tested on this feature space.

For the first and second classifiers landmark pairs are needed, to compute the parameters of the scaling and the maximal number of landmarks respectively. The landmark pairs are found with the help of a procedure similar to the RANSAC algorithm [73]. In the beginning, sets of matched SIFT keypoints points from the two images [12] are found. Assuming we find more than four matches, all possible sets of four matches are used to register the two images with a scaling transform. The number of landmarks for this transform is counted. Landmarks are feature points pairs that after the transformation have positions that are very similar, the Euclidian distance between their position vectors being smaller than a small threshold. The number of landmarks describes the quality of the transform. The optimal transform is the one with the largest number of landmarks. Each image pair where more than four matches were found is described by a certain transform and thus a number of shared landmarks. Image pairs where less than four matches were found are discarded.

As the sparse classifier needs several images to work properly, only the four individuals from the VARIA database with five or more images were used for testing. To ensure proper

Setup	Classifier	correct (%)
manual	differences	78
	RANSAC	100
	Sparse	100
SIFT	differences	70
	RANSAC	74
	Sparse	82
segmentation	differences	74
	RANSAC	83
	Sparse	91

Table 5.1: Results on the VARIA database.

Setup	Classifier	correct (%)
manual	differences	92
	RANSAC	100
	Sparse	100
SIFT	differences	88
	RANSAC	93
	Sparse	94
segmentation	differences	90
	RANSAC	95
	Sparse	99

Table 5.2: Results on the DRIVERA database.

deployment of the sparse classifier however, all available examples were considered when computing the sparse vector. For the DRIVERA database experiments have been conducted with the sparse classifier in two scenarios: with seven images per eye, and with all 14 images. The results improved from 90.71% to 99.29% correct decisions when using more images.

On the DRIVERA database, when the sparse classifier works on all SIFT keypoints it achieves 94.64% correct classifications, while when working only on the anatomic-relevant feature points it achieves 99.29% for the automatically segmented feature points and 100% for the manual ones. A similar behavior has been observed on the VARIA database. This shows that it is indeed advantageous to conduct person identification using only anatomically-relevant feature points, thus completely ignoring the background. It also shows that there is room for improvement with respect to the computation of the feature-point cloud.

The entire system is very fast, it returns a decision in six seconds, in comparison the RANSAC-based algorithm needs 14 seconds, while the SSD-based classification 66 seconds on an Intel Core i5 (3.1GHz) machine with 16GB RAM.

Conclusions and outlook

As it can be seen in Figure 5.8, some feature-point cloud components are neither crossings nor bifurcations. However, this is far from critical considering that the final feature vector is

computed from the log-covariance matrix of all SIFT descriptors. Finally, only a majority of cloud points need to be true crossings and bifurcations.

The feature vector is related to the second-order statistical properties of the feature vectors corresponding to the feature-point cloud. Clearly, other statistical descriptors could be used. It remains to be investigated if relating the feature vector to higher-order statistics, or using other types of statistical descriptions then moment-based, represents an improvement for this type of feature extraction.

A feature vector based on anatomical feature points related to the vascular network of a retina is very well suited for this task [146, 119]. The problem until now was that a reliable vessel segmentation would take a rather long time. The relative hysteresis classifier is fast enough to render such an approach feasible. Also by concentrating on the mid and large vessels and extracting the final feature vector with the help of the log-covariance descriptor (that is robust to outliers, meaning that a few additional crossings and bifurcations will not lead to significant changes), the algorithm is robust to pathological changes of the retinal vessels such as those encountered in the case of diabetic retinopathy [5].

The results obtained with the sparse classifier improve the more images are available for each eye in the database. For a practically deployed retina-based identification system, one should either acquire several images during enrollment, when setting up the system, or start with one image per eye and use, e.g., the classifier based on the number of landmark-pairs shared by two images to conduct classification, recording new images each time the respective eye is imaged, until enough images have been acquired to be able to sensefully use the sparse classifier.

Using the sparse classifier offers an easy way to deal (while the system is in use) with slow changes in the anatomical structure of the retinal vasculature (should these appear) or other changes in the acquired images occurring over a long period of time, due to various uncritical problems with the image-acquisition hardware. To compensate for this, one should simply update the train set of a certain eye by adding new images to the train set recorded at fixed time intervals.

5.1.3 Sparse classifiers for event detection

In the context of security and surveillance applications, often, unusual human behavior needs to be detected. The purpose of this Section is to prove that the sparse classifier is well suited for such tasks and to describe the specific requirements that need to be met along the way. Although sparse representations have been already used for video/signal analysis, the link to event detection is new [51, 193] and here [47] it is shown for the first time how to use the sparse classifier for this purpose. In comparison to standard, model-based approaches to action recognition [188] and event detection [137] the modeling step is no longer needed in this case, as only raw data will be processed. In the case of the sparse classifier, similar to [21], when analyzing new data, the training data is searched for similar instances.

The underlying assumption here is that there exists a set of allowed behaviors or action types for which sufficient video material is available for training in the form of action sequences showing a person conducting one of the allowed types of actions. With the sparse classifier, a test video is represented as a linear combination of the training data, (i.e., of the various actions in the training data) and the action it shows is recognized if the representation contains mostly data from one single training action. For event detection, if all actions in the training set are

more or less equally present in this representation it may be concluded that an event has been observed. Therefore, the sparse classifier works by recognizing normal actions and detecting events when no normal action is recognized with sufficient confidence. An entire new setup is needed to use the sparse classifier for event detection – beside its use for action recognition.

Also needed is a suitable feature extraction process. The feature-extraction process proposed here is designed from the very beginning to generate a feature space with a set of properties that are useful for event detection. These are mainly properties of invariance with respect to anthropomorphic changes, but also to Euclidian motion in the image plane. At the same time, as with all methods using the sparse classifier, the features need to be designed such that they exhibit a set of mathematical properties that make them suitable for sparse-representations-based event detection.

Since the main purpose is to prove that the sparse classifier is suited for event detection and to set the frame for further research in this direction, the data sets on which the algorithms are tested is less challenging than usual, but nevertheless related to real practical scenarios [83].

An event-detection setup for sparse classifiers

The sparse classifier as described in Section 2.3.2 will be used here for event detection. For this purpose it needs to be adapted. These adaptations are described next.

Features. For each frame during feature extraction, a sequence-feature vector is extracted from the chunk of video consisting of the analyzed and the last $R-1$ frames. Therefore, a single feature vector is extracted for each frame of video. This feature vector is computed on the basis of the contours of the silhouettes of the acting person. As we work with an overlap of $R-1$, the labeling, may start only with the R'th frame.

The "unsure" decision. With the help of the sparse classifier we can assign class labels to vectors. As described above, each frame generates one feature vector. If a vector cannot be assigned with sufficient confidence to any of the available classes, then it receives the label "unsure". This label is assigned with the help of the SCI, as defined in equation (2.76), in Section 2.3.2.

For this decision we need the parameters l and τ. The parameter l should be larger than one and $SCI(\mathbf{x}) \leq 1$. Usually, τ is chosen such that $\tau \in [0,1]$ and l is set accordingly. For a fixed τ, the larger l the higher the specificity of the algorithm, i.e., only "obvious" events, when the weights in $\mathbf{x}$ are equally distributed among the entries will be detected.

Labels for vectors and vector sequences. A vector can be labeled either with one of the k labels corresponding to the classes in the training set or with the "unsure" label, when the confidence in the classification result is low.

However, the data analyzed here consists of a sequence of vectors (as it comes from a video sequence) and the decision-making process needs to be adapted accordingly. In this context, time windows of L consecutive vectors need to be labeled with one of the labels that each frame may get. Therefore, a vector-sequence of length L is assigned the label of the class that yields a majority among the vectors composing it, while ignoring "unsure" decisions. If all frame-decisions in a sequence are "unsure", then the sequence is classified as 'unsure". Therefore,

working frame based provides the possibility to refine the decision for an action, which usually extends over many frames, by considering several frame decisions.

The detection of events. With the sparse classifier trained to recognize the normal case and its various components, the sparsity concentration index is used to detect events. Should a vector sequence be labeled as "unsure", then this is equivalent to detecting an event starting at the first vector of the L-length sequence.

Building the training matrix for the sparse classifier. The sparse classifier labels frames. To compute the training matrix, various numbers of frames per class can be used. For the sparse classifier, the number of frames per class has to be set in relation to the dimension of the feature space. Furthermore, in this particular case, there are two possibilities to construct the training set: (i) either by randomly selecting frames from different action sequences of the same action type, or (ii) by taking consecutive frames from one or more action sequences of the same action type. The former case is equivalent to using the sparse classifier to classify directly into action types, i.e., with a number of classes equal to the number of action types. In the latter case, the sparse classifier is used to classify first into action sequences, i.e., with a number of classes equal to the number of different action sequences. The frame in question is finally labeled with the action-type label of the action sequence to which it was classified by the sparse classifier.

Labeling action sequences, while having several sequences at our disposal for training, has several advantages when using the sparse classifier for event-detection purposes (see also Section 2.3.2). It has the advantage of increasing the sparseness of the sought coefficients vector. Furthermore, event detection may be conducted while not using the sparse classifier as a one-class classifier, even in the case when only a single type of action is defined as normal – in such a case we simply construct the training set from several sequences of the same action type.

Invariant sequence feature extraction

The features are based on Fourier descriptors (FD) [6] computed from the contour of the acting person. The final sequence feature vector that corresponds to one frame, contains information from a set of R consecutive frames, including the current one. These features are chosen such that they achieve a set of invariances needed for event detection. A feature vector is computed for every frame of video starting with the R'th.

Feature extraction has to be devised to support the basic assumptions of sparse-representation based classification about the linear relationship of vectors of the same class. The obtained sets of features need to form a vector space [85]. Furthermore, for human-action analysis the features need to be invariable to anthropometric changes like height, gait, hair style, clothing, face traits, etc. [165]. Ideally they should also be invariant to scaling and viewpoint changes. However, here this is achieved during classification rather than during feature extraction, by adding sequences obtained under various viewpoints and at various scales to the training set.

Contour extraction and Fourier descriptors. The features describe human actions. These actions take place under various illumination conditions, and are conducted by persons wearing differently-textured clothes. To achieve invariance over such conditions the features are computed from the contour of the acting person.

The contour-detection procedure used here assumes the human is the only object moving in the analyzed video and the background is available. It starts with the computation of a binary motion mask B by subtracting the background from the current frame and comparing the result with a threshold [147]. The contour $\mathcal{C} = [(x_1, y_1), \ldots, (x_{n_C}, y_{n_C})]^T$, with n_C contour points, is obtained by subtracting the eroded motion mask from the original. For erosion a cross-like structuring element A_+ [65] is used.

$$\mathcal{C} = B - (B \ominus A_+) \tag{5.1}$$

The FDs for the contour are computed as

$$\vartheta(\omega_p) = \mathcal{F}(\mathcal{C}^c) = \sum_{i=1}^{n_C} u_i e^{-j\omega_p i}, \tag{5.2}$$

where $\mathcal{C}^c = [x_1 + jy_1, \ldots, x_{n_C} + jy_{n_C}]^T$ is the complex representation of the contour and $\mathcal{F}$ is the Fourier operator. As features for frame f we keep only the magnitudes of Q descriptors – half for the negative and half for the positive frequencies, excluding the DC component:

$$\boldsymbol{\vartheta}^f = [|\vartheta_1|, \ldots, |\vartheta_Q|]^T. \tag{5.3}$$

The FD magnitudes are invariant to starting point, rotation and reflection.

Invariant sequence features. Viewpoint changes, scale variations but foremost the anthropometric characteristics of various persons lead to changes in the acquired contours. Invariance needs to be achieved with respect to such changes. The needed invariances, are introduced with the help of invariant integration [161]. Invariant integration returns features invariant to the actions of a group of transformations on an input signal. For this purpose a feature function $f(\cdot)$ is defined on the signal space and integrated over the group of actions. In Reference [162] it is shown that the set of monomials $m(\cdot)$ is a good choice for $f(\cdot)$. For a D-dimensional input $\mathbf{t} = [t_1, \ldots, t_D]$, the monomials are defined as

$$m(\mathbf{t}) = \prod_{d=1}^{D} t_d^{b_d}, \tag{5.4}$$

where $b_d \in \mathbb{N}$. The number of product components from equation (5.4) that are different from one gives the order of the monomial.

A model for anthropometric changes. The group of transformations to which invariance is desired is defined in relation to the effects that anthropometric changes have on the contour of the person. At this stage, anthropometric changes are modeled by the convolution of the contour with pairs of Dirac pulses, where the distance between the two pulses is variable. Thus, we would like to achieve invariance to sinusoids modulating the FDs. Multiplication of the FDs with a sinusoid is equivalent to a shift of the Fourier coefficients of the FDs. Therefore, we need to compute features invariant to shifts of the Fourier transform of the FDs. To begin with, the Fourier transform of the columns of FDS is computed, obtaining thus

$$\mathcal{S} = \mathcal{F}_c(FDS) = [\boldsymbol{\varphi}^1, \ldots, \boldsymbol{\varphi}^R],$$

with $\boldsymbol{\varphi} = [\phi_1, \ldots, \phi_Q]^T$. Next, invariant integration is applied on the columns of $\mathcal{S}$ and only the magnitudes are taken into consideration. By integrating over all shifts from 1 to Q, while enforcing suitable boundary conditions, invariance is achieved with respect to a set of modulating sinusoids of various frequencies.

The feature function. Here, monomials of order two are used. Thus, $b_d = 0$, for $d \in \{1, 2, \ldots, Q\} \backslash \{q_1, q_2\}$ and $b_d \neq 0$ for $d \in \{q_1, q_2\}$. For better separability, various monomial features (i.e., different values for q_1 and q_2) are used, obtaining for each column of $\mathcal{S}$ an invariant feature vector with one entry per monomial. Here, we need a scalar feature for each column from $\mathcal{S}$, thus, the feature function $\mathcal{M}(\cdot)$ is defined as a linear combination of monomials. To obtain class-conditional distributions and thus a feature space that is better suited for the sparse classifier, we need $b_d \in \mathbb{R}$, instead of $b_d \in \mathbb{N}$.

The feature function consists of a linear combination of three monomials $m_i,\ i = 1 \ldots 3$, from various entries along the columns $\boldsymbol{\varphi}^r,\ r = 1, \ldots, R$ of $\mathcal{S}$. For m_1, we have $q_1 = q$ and $q_2 = q - 1$ with $b_{q_1} = b_{q_2} = 1.7$, for m_2 we have $q_1 = q$ and $q_2 = q - 3$ with $b_{q_1} = b_{q_2} = 1.5$ and for m_3 we have $q_1 = q$ and $q_2 = q - 5$ with $b_{q_1} = b_{q_2} = 1.3$. The index q runs over the entries of $\boldsymbol{\varphi}^r$. The feature function is then

$$\mathcal{M}(\boldsymbol{\varphi}^r; q) = (\phi^r_q \phi^r_{q-1})^{1.7} + (\phi^r_q \phi^r_{q-3})^{1.5} + (\phi^r_q \phi^r_{q-5})^{1.3},\ q = 1, \ldots, Q,$$

with periodic boundary conditions.

The sequence feature may now be computed by integrating over this feature function. Integration is numerically approximated by summation, which is in turn unnormalized mean computation. For increased robustness order statistics are used here instead of integration. The sequence-feature vector is computed by taking the 25'th percentile over the combinations of monomials:

$$v_r = p_{25}(\mathcal{M}(\boldsymbol{\varphi}^r; q)).$$

Therefore, for the entire $\mathcal{S}$, an R-dimensional sequence-feature vector $\mathbf{v} = [v_1, \ldots, v_R]^T$ is obtained, which is invariant to a set of anthropometric variations. The parameters of the feature function and the precise percentile need to be established empirically using the available dataset.

It is interesting to note the relationship that exists between invariant integration and the log-covariance matrix-based feature extraction, described in Section 5.1.2, where, assuming the mean is zero, we build with the help of the arithmetic mean as an expectation estimate, a robust type of monomials. However in this case, we do not integrate anymore over the monomials, but rather gather each monomial in the final feature vector.

Experiments

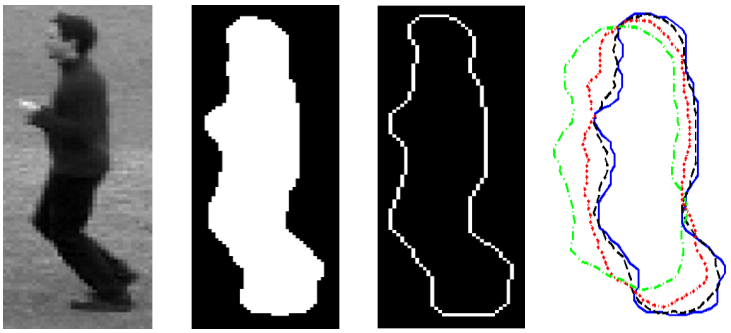
Next it will be assumed that the video data has 24 fps. This value will be used for several parameter choices, which may otherwise appear random. The methods described above have been tested on part of the KTH action database [160]. This database contains six types of human actions (walk, jogg, run, box, hand wave and hand clap) performed by 25 persons, from which the walk (W), jogg (J) and run (Rn) actions have been used. Each person performs the action four times: three times outdoors, and one time indoors. Only the outdoor sequences where the person moves parallel to the camera have been used. Therefore, for each action type there are 25 action sequences (one for each person) in the data set.

The experiments are divided into: feature extraction, action-sequence recognition, and point-event detection. For action-sequence recognition, the query action sequence is already in the training set. For point-event detection, the analyzed action sequence is not in the training set. To compute the SCI, the parameter l had the value ten. For the "unsure" decision $\tau = 0.4$ has been used.

Since a step during walking takes some ten frames, the number of columns of the FDS is chosen to be $R = 10$. Accordingly, the maximal number of training-set vectors per class needs to be five.

Feature extraction. The feature extraction process is demonstrated here with the help of an example, illustrated in Figure 5.10 and Figure 5.11.

After motion detection, a motion mask is obtained, as shown in Figure 5.10 (b). This mask is then used to detect the contour of the moving person as it can be seen in Figure 5.10 (c). The



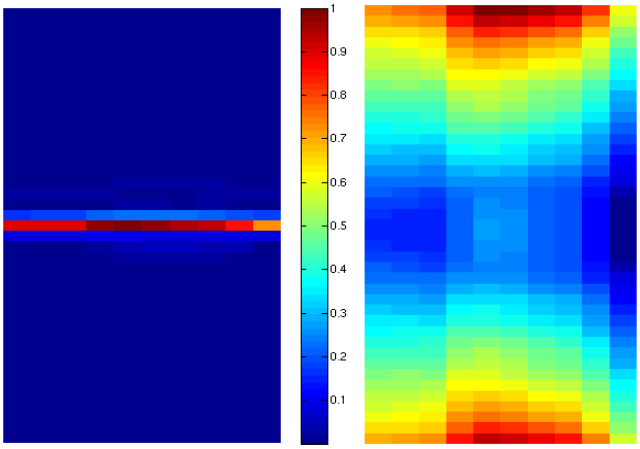
(a) (b) (c) (d)

Figure 5.10: Motion region in a frame (a), motion mask (b) and its contour (c). Original contour (continuous blue line) and and variations to which we are invariant after invariant integration (interrupted lines), applied to the $\mathcal{S}$ (d)

contour is further used to compute the FDs with $Q = 40$ and the FDS with R=10, which, for our example, is depicted in Figure 5.11 (a). That yields in turn the $|\mathcal{S}|$ shown in Figure 5.11 (b), and after the invariance transform, the feature vector that is shown in Figure 5.12 (b). By using the magnitudes of the FDs we are already invariant to a set of variations of the original contour. The invariance transform achieves that the corresponding feature vector is invariant to several more contour variations. Some of these additional variations are shown in Figure 5.10 (d).

Action recognition. The action-recognition performance of this algorithm is demonstrated with the help of two experiments: in the first experiment it is investigated how is each video frame classified, while in the second the frame decisions are used to classify action sequences. In the latter case a decision is needed for video sequences more than L frames long, but showing the same action. For this purpose, the vector sequence formed from the first $L = 24$ frames of an action sequence is labeled as described in Paragraph **Labels for vectors and vector sequences** and this label is awarded to the analyzed action sequence. The results were computed by means of the leave-one-out procedure.

After each experiment, permutation matrices have been generated. These matrices should be read along lines, e.g., for the action of jogging, the first column contains correct decisions, the second wrong decisions in favor of the class labeled “Running” and the third column wrong decisions in favor of the class labeled “Walking”. For the frame experiment, on average 32% of all frames in a sequence are labeled as “unsure”. In Table 5.3 (a) results are shown that include the “unsure” frames. In Table 5.3 (b) only valid decisions are considered, thus ignoring “unsure” frames. Therefore, in this table, for example, only 75% of all “Running” frames have



(a) (b)

Figure 5.11: FDS (a) and $|\mathcal{S}|$ (b)

(a) Results including "unsure" frames

(%)	J	Rn	W	unsure
J	36.79	3.28	18.99	40.94
Rn	0	75	0	25
W	2.72	0.98	65.38	30.92

(b) Results ignoring "unsure" frames

(%)	J	Rn	W
J	62.36	5.56	32.19
Rn	0	100	0
W	3.84	1.42	94.75

Table 5.3: Permutation matrices for frame decisions.

been considered, namely those for which a label other than "unsure" has been awarded; and as it may be observed, all these valid decision have been correct for this class. For the sequence experiment, the results are shown in Table 5.4. For the sequences labeled as "unsure", all first 24 frames were labeled as 'unsure'.

Event detection. To demonstrate that the sparse classifier with invariant sequence features is well suited for the purpose of event detection, two experiments have been conducted: for the first one, two types of actions are used to simulate the normal case and the event was the third (e.g., running and walking were normal and jogging was an event); and for the second one, one type of action is used as normal case and the other two were the event. Similar to the action-recognition case, action sequences that extend over more than L frames need to be analyzed. In this case again, an action sequence is labeled with the label of the vector sequence generated from its first $L = 24$ frames.

(%)	J	Rn	W	unsure
J	86.67	0	13.33	0
Rn	0	60	0	40
W	0	0	100	0

Table 5.4: Permutation matrix for action-sequence decisions.

Normal	J & Rn	J & W	Rn & W
Event	W	Rn	J
(%)	100	73.33	86.67

Table 5.5: Event-detection results for the first experiment.

Normal	J	Rn	W
Event	Rn & W	W & J	Rn & J
(%)	96.67	100	86.67

Table 5.6: Event-detection results for the second experiment.

The results for the first experiment are shown in Table 5.5 and for the second, in Table 5.6. These tables show the percentage of action sequences correctly classified as event. The results are obtained by a modified type of five fold cross-validation. At each iteration, the training matrix is composed of five different "normal" action-sequences and the testing event set is given by all 25 "event" action-sequences. In the end we compute the average of the detection rates.

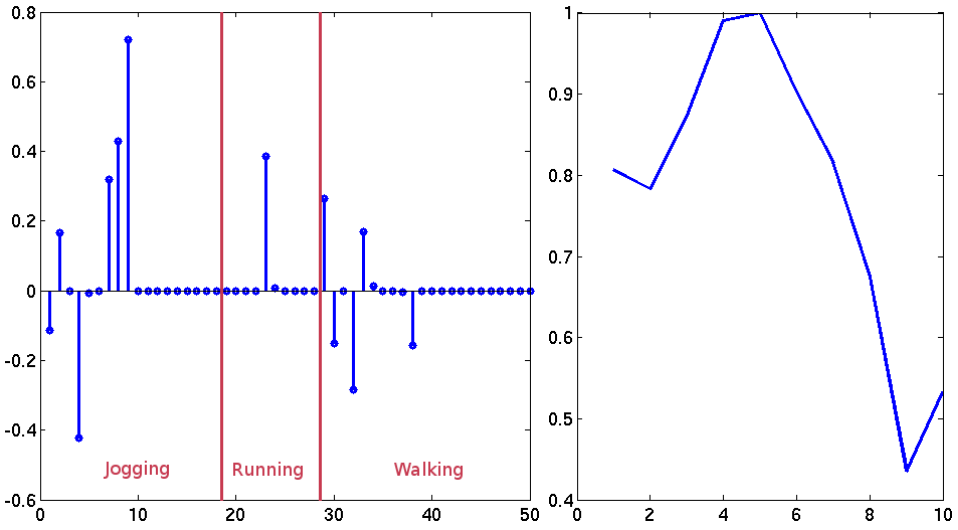
Discussion

For each of the $L = 24$ frames needed to take a vector-sequence decision, a feature vector is extracted from ten frames (nine previous frames and the current one). Thus, a decision for an action sequence is taken after 34 frames have been recorded. Clearly, to analyze an action sequence in the current setup, a minimum of at least ten frames is needed, in which case the decision for the 11'th frame is the decision for the entire sequence. Deciding for an action sequence based on a majority of frame-decisions from the first 24 frames proved well suited for the current data set. Alternative decision strategies can also be used, like for example averaging decisions over several overlapping or non-overlapping vector sequences. As a rule of the thumb, the more frames are considered, the better the sequence-decision. The algorithm can also be used online, in which case it would label chunks of video of 24 frames that can be taken with or without overlap, depending on the application.

Extracting the sequence features from ten frames is well suited for the current data. The number of frames to be considered for a sequence-feature vector should be chosen in relation with the frame rate of the analyzed video and the length of an atomic part of the analyzed action (in this case one step of the person executing the action) and validated on the training data.

During feature extraction, $Q = 40$ FDs left and right from zero are used and hence a minimal contour length for a certain image resolution is implicitly assumed. This number of FDs is well suited for the analyzed data, but it should be chosen according to these considerations in practice. All parameters with no rules for determining them were established by six-fold cross validation on a separated data set.

The feature vector is invariant to several variations of the person's contour, some corresponding to anthropometric changes, however, some can be thought of as corresponding to viewpoint changes and to scale changes and thus we obtain also a mild viewpoint invariance in the feature vector.



(a) (b)

Figure 5.12: The coefficient vector for a frame from a jogging sequence with decision regions (a) and the feature vector (b).

As described in Paragraph **Building the training matrix for the sparse classifier**, the sparse classifier is used to label each frame with an action-sequence label and not with an action-type label. Finally, as discussed before, each frame is still classified into three different action types, but this is now achieved by taking over the label of the sequence to which it was assigned by the sparse classifier. This has several consequences. For the example illustrated in Figure 5.12 (a), for a training space with two action sequences per action type and using for jogging the first nine frames of each sequence, for running the first five of each sequence and for walking the first eleven of each sequence, there are the following numbers of feature-space vectors per class: $c_{walk} = 22$, $c_{run} = 10$ and $c_{jogg} = 18$. If we assume in this example that we have one class per action type and therefore the sparse classifier works directly with three classes, the minimal dimension R of the feature space needs to be $R = c_{walk} \cdot 2 = 44$ such as to use the sparse classifier properly. However, with one class per action sequence, the sparse classifier works with six classes and the minimal R needs to be $R = 22$ such as to use the sparse classifier properly. In this case, besides being able to work with a smaller R, the sought coefficients vector is also more sparse (i.e., 11-sparse in comparison to 22-sparse in the former case).

By assigning a frame first action-sequence labels and not directly action-types labels, the sparse classifier can even be used as a one class classifier for action types, of course provided several action sequences are available for the one action type. Experiments have been conducted in such a scenario to test the limits of the sparse classifier for event detection. More precisely, it has been attempted to detect events, while the normal case was given by one action type and there were just two normal-case sequences in the training set. Therefore, considering that the maximal number of training-set vectors per action sequence is five and the dimension of the feature space is ten, there is an equal numbers of dimensions in the feature space and vectors in the training set, i.e., the matrix of the corresponding system of equations is square. The experiments were successful, which should be understood in relation with the fact that the

components of the feature vector are correlated. Indeed, other practical applications of the sparse classifier (like for example some of those discussed in Reference [187]) have shown that sparse approximate solutions may still be found even for overdetermined systems. It appears that what matters here is the rank of the training-space matrix that needs to be lower than the number of training-set vectors.

Yet another interesting aspect is that the sparse classier works well enough in this case even if the maximal number of training-set vectors per class is half of the dimension of the feature space and not less than a third, as suggested by the equation (2.77). It seems that there is a large-enough (even if not "overwhelming") probability to recover the correct solution efficiently even in such cases.

A decision for a sequence of around 50 frames is available after 47 seconds under MATLAB on a 2.66 GHz dual-core machine. However, many of the algorithmic steps can be conducted in a parallel manner.

Conclusions and summary

The sparse classification paradigm can be used for action recognition and to detect all sort of point events. While not directly suited for context and collective events, it may represent an action-recognition building block for such algorithms, other blocks being necessary for analyzing the chains of individual actions [134]. The particular feature extraction process used here is specific for the analysis of human behavior. The analysis is concerned with the behavior of a person in a single track and the current feature extraction is adapted for this case. A prerequisite for deploying these methods in more complicated scenarios is a successful tracking, irrespective of the number of cameras used.

By using sparse representations, training-related advantages are obtained: the method described here can be easily trained, as this is just a matter of gathering raw data; and in connection with this it can be also easily extended, should the normal case change, by just adding corresponding raw data to the training set.

The algorithm can be seen of consisting of two parts: feature extraction, and sparse classification. The feature extraction is targeted to certain invariances and tailored to the sparse classifier. Sparse classification offers a set of advantages over other methods for the problem of action recognition and event detection, being robust, adaptive and intuitive. The sparse classifier can not be used directly as a one-class classifier. However, in the event-detection setup, provided the sub-cases of the normal case are available, the sparse classifier can detect events by attempting to classify into these sub-cases and detecting an event when an out-of-training-set vector is observed.

It has been shown that sparse classification as introduced in Reference [187] is well suited for human event detection. In this context, the focus is now set on the extraction of suitable features to enable the usage of such methods. Furthermore, even if the issue of invariance can be addressed at the classifier level, when using sparse classifiers, many of the desirable invariance properties that characterize a good human action recognition/event detection method should be obtained by means of the feature extraction process. We have also discussed how to use the invariant integration [161] to extract such features from the contour of the acting person.

A complete, sparse-classifier-based event-detection framework, has been thus introduced, which follows largely the requirements for algorithms used in security applications, as described in Section 1.3.3.

5.2 Novel statistical approaches to event detection

The statistical framework has the advantage of a solid research foundation stretching over several decades that led not only to a sound mathematical setup but also to a large number of general yet powerful methods. This framework is the most often used in the event-detection context, generating a true field of *statistical event detection* expressed in a large number of publications. Here, this field is expanded by introducing in Section 5.2.1 a novel linear-predictors mixture and by adapting the LCCRFs for event detection in Section 5.2.2.

5.2.1 A linear-predictors mixture

As an event represents a deviation from the normal case, one possibility to detect events – in particular for cases where a time component is present – is to build a model of the normal case and look for deviations from the model. The model should be able to predict what would happen at a future time point under the normal-case assumption. As long as the model is successful and its prediction is similar to the observation made, it may be assumed that the normal case is observed. When the prediction is very different from what is really observed, it is assumed that an event has occurred.

To implement such an event detection setup, a linear-predictors mixture will be introduced next. This method has several similarities with both GMMs and HMMs, in a way combining the advantages of both. Assuming a sample from the analyzed signal has been observed, GMMs use the position of the sample in its corresponding feature space to distinguish the "normal case" from the "events", as the probability of the normal case, given the normal-case-trained trained GMM is larger. The GMMs ignore thus any type of temporal relationship among observations (as long as this is not explicitly present in the observation). They throw thus away a lot of context, as observations that are close in time are usually related (see Section 2). The decision is made based on the affinity of the observation to a certain component of the mixture, affinity that is actually measured with the help of the variance-weighted distance to that component's mean. In this case, the decision is made again with respect to the affinity of the observation to a certain mixture component. However, now the mixture components are linear predictors and thus the affinity is measured based on the success with which the observation is predicted based on the training samples used to train the predictor. Thus, their temporal relationship is used implicitly. It is interesting to note the conceptual relationship between the linear predictors mixture and the sparse classifier. In both cases we use a linear function of a set of observations from the training set to explain a new observation and decide based on how "good" this explanation is. In comparison to HMMs (see Section 2.2.1), this method is simpler to implement while not being compelled to follow the Markovian assumption. It is able to model more complex relationships between observations, as the temporal connection is used directly, not over the hidden states. The method described here concentrates on the observations, and in a certain sense the HMM states are replaced by linear predictors thus going around the restrictions related to the discrete state space.

In the context of filter theory, assuming the input signal is not stationary, adaptive filters [88] are needed. However, linear adaptive filters include several strong assumptions with respect to the observed data, like Gaussianity and linearity. As a bridge gap solution, linearity is assumed over short intervals. This leads to methods like the extended Kalman filter. Here an alternative is proposed, in the form of a mixture of linear one step predictors. This method has several advan-

tages, for example less training vectors are needed to achieve comparable results. Furthermore, for event detection, should what we define as the normal case change with time, this method can be easily adapted. Of course, should the underlying linearity assumptions fail, we still need to resort to more powerful methods such as the particle filter described in Section 2.2.1.

Linear predictor mixtures

The parameters of one-step linear predictors are computed from the Yule-Walker equations [181, 190] (see Section 2.1.2). Next it is described how to build a mixture of several predictors and use it for event detection. To construct the mixture the prediction errors of the individual predictor components will be used. The parameters of the mixture are estimated with the help of some training data that shows only the normal case.

Linear predictors and their errors. As described in Section 2.1.2, there is a strong relationship between Linear Predictors (LPs) and AR models, a linear predictor working optimally when the input random signal is an autoregressive process of the same order [23]. For a sequence of observations $\mathbf{x}(i) \in \mathbb{R}^N, \forall i \in \{1, \ldots, t-1\}$, a linear predictor $\mathbf{a} = [a(0), a(1), \ldots, a(p)]^\top$ will estimate the next observation $\mathbf{x}(t)$ as

$$\hat{\mathbf{x}}(t) = \sum_{i=1}^{p} a(i)\mathbf{x}(t-i) + a(0)\mathbf{e}_N, \tag{5.5}$$

with $\mathbf{e}_N = [1, 1, \ldots, 1]^\top$. This follows also from $\hat{\mathbf{x}}(t) = E\{\mathbf{x}(t)\}$, where

$$\mathbf{x}(t) = \sum_{i=1}^{p} a(i)\mathbf{x}(t-i) + a(0)\mathbf{e}_N + \mathbf{v}(t), \tag{5.6}$$

is a linear combination of p predecessors and $\mathbf{v}(t)$ a Gaussian error term such that $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \Sigma)$.

With $\mathbf{X}(t) = [\mathbf{e}_N, \mathbf{x}(t-1), \mathbf{x}(t-2), \ldots, \mathbf{x}(t-p)]$, we can rewrite equation (5.5) in matrix notation as

$$\hat{\mathbf{x}}(t) = \mathbf{X}(t) \cdot \mathbf{a}.$$

Therefore, for a sequence $\mathbf{y}_n(t) = \left[\mathbf{x}(t)^\top, \mathbf{x}(t-1)^\top, \ldots, \mathbf{x}(t-n)^\top\right]^\top$ of n time-consecutive vectors, we obtain

$$\hat{\mathbf{y}}_n(t) = \mathbf{Y}(t) \cdot \mathbf{a},$$

where we have used the sequence of matrices $\mathbf{Y}(t) = [\mathbf{X}(t), \ldots, \mathbf{X}(t-n)]$. From here, we can compute an approximation of the linear predictor [23, 181, 190] at time t as

$$\mathbf{a}(t) = \left[\mathbf{Y}(t)^\top \cdot \mathbf{Y}(t)\right]^{-1} \mathbf{Y}(t)^\top \hat{\mathbf{y}}_n(t). \tag{5.7}$$

The quadratic prediction error of $\mathbf{a}(t)$ at time step s is then

$$\hat{\epsilon}_t(s)^2 = (\mathbf{x}(s) - \hat{\mathbf{x}}(s))^\top (\mathbf{x}(s) - \hat{\mathbf{x}}(s)), \tag{5.8}$$

including in $\hat{\mathbf{x}}(s)$ the approximation uncertainties for $\mathbf{a}(t)$ as well. Essentially, next this error term is going to be used for event detection, such that if the prediction error is high, an event

is declared in an approach similar to the filter-based change detection, as discussed in Section 1.3.3. However, with the Linear Predictor Mixture model (LPM) a new type of model is introduced, which is able to better accommodate the normal case.

Using a matrix representation of the linear predictor

$$\mathbf{A}(t) = [\mathbf{I}, -\mathbf{I} \cdot a(0), -\mathbf{I} \cdot a(1), \ldots, -\mathbf{I} \cdot a(p)]$$

and denoting $\eta(s) = [\mathbf{x}(s)^\top, \mathbf{e}_N^\top, \mathbf{y}_p(s-1)^\top]^\top$, with p the length of the predictor, equation (5.8) can be rewritten with the help of equation (5.5) as

$$\begin{aligned} \hat{\epsilon}_t(s)^2 &= (\mathbf{A}(t)\eta(s))^\top(\mathbf{A}(t)\eta(s)) \\ &= \eta(s)^\top \mathbf{H}(t)\eta(s), \end{aligned}$$

with $\mathbf{H}(t) = \mathbf{A}(t)^\top \mathbf{A}(t)$, which will prove useful when introducing the LPM next.

Mixture model and detection of events. The LPM has similarities to GMMs (see Section 2.1.3). For GMMs we have $p(\mathbf{x}) = \sum_{i \in I} w(i) g_i(\mathbf{x})$, where $g_i(\mathbf{x})$ is a Gaussian distribution, I is a set of indices, each referring to one Gaussian mode, and $w(i)$ are weights with $\sum_{i \in I} w(i) = 1$, $w(i) \geq 0$. In the same manner, the LPM is a mixture of several linear prediction error filters, and therefore an approximation to complex time series irrespective if this is stationary or non stationary.

In order to introduce the LPM event detector, the *exponential representation* of the error $f_t(\eta(s))$ defined as

$$\begin{aligned} f_t(\eta(s)) &= \exp(-\hat{\epsilon}_t(s)^2) \\ &= \exp\left(-\eta(s)^\top \mathbf{H}(t)\eta(s)\right) \end{aligned}$$

is used in a weighted sum (similar to GMMs). The error of the LPM for the observation $\mathbf{x}(s)$ is computed as

$$F(\eta(s)) = \sum_{t \in T} w(t) f_t(\eta(s)), \tag{5.9}$$

where T is a set of time indices that refers to a training set and $\sum_{t \in T} w(t) = 1$, $w(t) \geq 0$. An event is detected if the score F is below a threshold θ with $0 \leq \theta \leq 1$. Clearly, $0 < f_t(\eta(s)) \leq 1$ and the smaller $f_t(\eta(s))$, the larger the estimated prediction error $\hat{\epsilon}_t(s)$.

Parameter estimation. The parameters of the LPM are a set of several linear predictors $\mathbf{a}(t)$, and the corresponding weights $w(t)$. These are computed from a training set in a procedure involving several steps, as described next.

Let $\mathbf{x}_0(i), i \in \{1, \ldots, l\}$ be a training set of l observations, with $l \gg p$. At a certain time point $t^{(\tau)} > n, \forall \tau = 1, 2, \ldots, \tau^{max}$, equation (5.7) returns a unique linear predictor $\mathbf{a}(t^{(\tau)})$. At $\tau = 1$, the first step of the procedure (usually such that $t^{(1)} = n + 1$), we initialize the set T of linear predictors of the LPM with the predictor corresponding to time index $t^{(1)}$, and thus $T \leftarrow \{t^{(1)}\}$. At each step predictors are added for those observations from the training set that are not well predicted by any of the predictors already available in T. Thus, at iteration $\tau > 1, \tau \in \mathbb{N}$, a linear predictor $\mathbf{a}(t^{(\tau)})$ is added such that this offers a better prediction for the

training sample for which the prediction error of the predictors available in T at iteration $\tau - 1$ is maximal. Hence, we set

$$t^{(\tau)} = \arg\min_{\tilde{t} \notin T} \sum_{i=1}^{\tau-1} f_{t^{(i)}}(\eta(\tilde{t})) \tag{5.10}$$

and $T \leftarrow T \cup \{t^{(\tau)}\}$. New predictors are added to the mixture for a fixed number of steps, until a maximal preset size of the LPM, τ^{max} is reached.

After finding the predictors, the weights in equation (5.9) are computed by

$$w(t^{(i)}) = \frac{\sum_s f_{t^{(i)}}(\eta(s))}{\sum_{t^{(j)} \in T} \sum_s f_{t^{(j)}}(\eta(s))} \tag{5.11}$$

for each $t^{(i)} \in T$, with $s \in \{p+1, \ldots, l\}$. The estimated linear predictors and the weights define the LPM, see equation (5.9).

Experiments and discussion

The LPM is tested on both real and synthetic data. The tests on synthetic data demonstrate that the LPM is a viable improvement over both GMMs and HMMs with Gaussian state-conditional distributions, for the purpose of event detection. This shows that the LPM is not that bounded to the Gaussian assumption (see Section 1.1.1). The test on real data shows that the LPM successfully detects events in real-life scenarios as well.

Experiments on synthetic data. To generate 5D synthetic data, a dynamic system was used. It is defined as

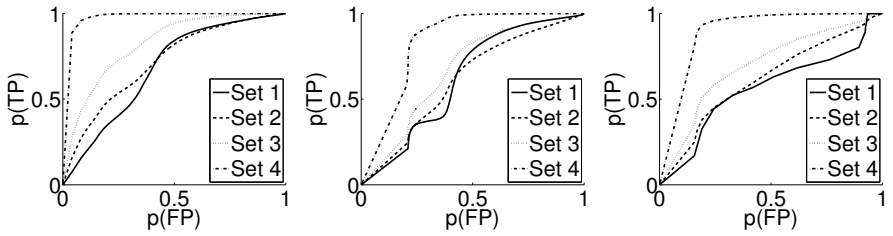
$$\mathbf{x}(t) = \mathbf{m}(\psi) + \sum_{i=1}^{3} a_\psi(i) \left(\mathbf{x}(t-i) - \hat{\mu}(t)\right) + \mathbf{v}(t),$$

where $\hat{\mu}(t) = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}(t-i)$ and $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \Sigma)$. The parameters of the dynamic system are the coefficients $a_\psi(i), i \in \{1,2,3\}$ and the offset $\mathbf{m}(\psi)$. The time-dependent variable ψ represents the state of the system, and it is modeled as a first-order Markov chain.

The normal-case data was generated with the help of this dynamic system, using a predefined set of parameters. To generate the event data, the parameters were changed. Three event data sets were generated by changing the offsets $\mathbf{m}(\psi)$ (Set 1), using different coefficients $a_\psi(i) + r(t,i)$ with $r(t,i) \sim \mathcal{N}(0,1)$ (Set 2) and $r(t,i) \sim \mathcal{N}(0,2)$ (Set 3) respectively, and finally by generating $a_\psi(i)$ directly from a Gaussian distribution (Set 4). For each of these five data sets (i.e., one "normal-case" and four "event") 50000 observations were generated.

For testing, a LPM with $\tau^{max} = 50$ predictors is built. A number of $p = 10$ previous observations is used to predict the following one, and in the training, 15 observations are used to estimate one predictor, i.e., $n = 14$.

The LPM is compared on this data with a GMM and with a Gaussian HMM (GHMM). In the case of the GMM, an event is detected if its probability given the GMM is smaller than a threshold. The GMM is trained with the help of the EM algorithm (see Section 2.1.3) on some "normal" data. To find out the optimal number M of mixture components, we have employed a full-search strategy between five an 100 components. The best results have been obtained with $M = 10$. The HMM (see Section 2.2.1) has ten Gaussian states. It has been also trained on



(a) LPM (b) GMM (c) GHMM

Figure 5.13: ROCs of (a) the LPM, (b) the GMM, (c) the GHMM.

"normal" data. To detect events the evaluation problem is solved for new data, if the probability of the observations given the model is less than a certain threshold, an event is detected.

In each and every case, for training half of the observations available are used as the "normal" data. The tests were then conducted on the rest of the "normal" data and on the "event" data.

Results. As in the end event detection is a binary classification problem, the Receiver Operating Characteristic (ROC) is used to compare the methods described above. To compute the ROC, two probabilities are needed: $p(TP)$, the probability of detecting a true positive; and $p(FP)$, the probability of detecting a false positive. These are computed while varying θ between zero and one. $p(TP)$ is the probability of correctly detecting an event, defined as:

$$p(TP) = \frac{\#(\text{Detected, simulated events})}{\#(\text{Simulated events})}.$$

$p(FP)$ is the probability that a normal observation is falsely classified as an event, defined as:

$$p(FP) = \frac{\#(\text{Falsely detected events})}{\#(\text{Normal observations})}.$$

Figure 5.13 shows the performance of each of the tested methods on each "event" dataset. Clearly Set 1 is the most difficult one, which suggests that the LPM is rather sensible to offsets, even though not as much as the GHMM and the GMM. Nevertheless, this is a normal behavior in a setup where the amplitude of the observation is the feature mainly used to decide on the label. As it may be seen from the results achieved for Set 2 and Set 3, the more different the event is from the normal case, the better the event-detection results achieved by the LPM. This is the normal behavior of any event-detection algorithm, as illustrated by the results achieved by the GHMM and the GMM.

In Figure 5.14, the overall performance (results using all event datasets as one set) of each model is shown. Comparing the GHMM and the GMM, the GHMM performs better, but the LPM outperforms both methods, showing that there are event detection problems where the LPM can be successfully applied, and they perform better than GMMs or GHMMs.

Experiments on real data. The real data represents a video sequence recorded with a web cam in a fixed position. A radio-controlled car is driven in the surveyed area. The car performs several actions (see also Figure 5.2.1). The "normal case" consists of any combination of normal

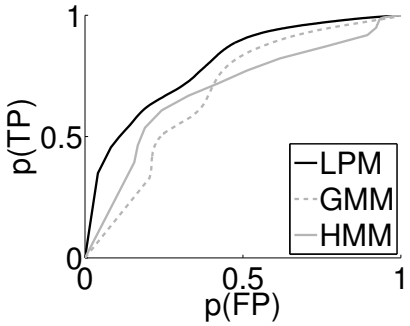
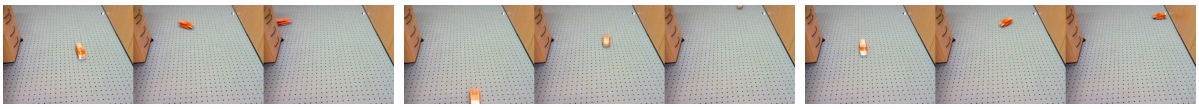


Figure 5.14: Comparison among various methods over all data sets.



(a) Turning left (b) Straight (c) Turning right

Figure 5.15: Frames from the training data: the car turns left (a), drives straight (b) and turns right (c).

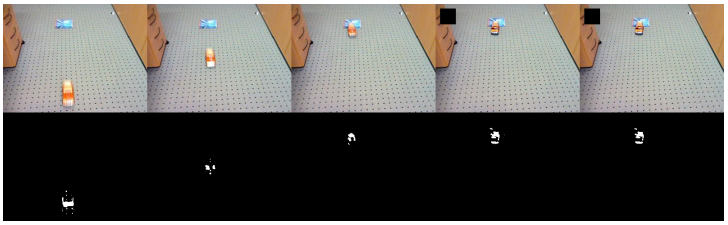
movements (driving straight, turning left or right), while an "event" is an arbitrary action that differs from the "normal" actions, as for example, when the car hits an object.

As the car is the only thing moving in the analyzed video, the tracking is very simple, needing only motion detection. For motion detection a background subtraction algorithm [149] is used. This estimates for every image of a video sequence the foreground and updates a background model. It uses one threshold for each pixel. The algorithm is applied directly to color images. The result of motion detection is a blob of moved pixels corresponding to the car. Some motion detection examples can be seen in Figure 5.16.

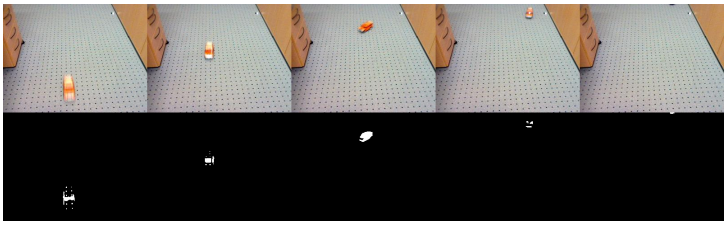
The observations of the LPM are 2D position vectors of the center of mass of the detection blob in each frame of video. As the normal case includes three actions, the LPM has three predictors: going straight, turning left, and turning right. If the score F is lower than the threshold θ, an event is detected. In general, θ is an arbitrary threshold with $0 < \theta < 1$. For this experiment the value of θ was set to 0.4 by analyzing the normal data (see Figure 5.17).

Results. In Figure 5.16 (a), several frames of one of the analyzed video sequences are shown: first, the frame where a moving object has been detected for the first time, then the frame immediately before an accident, followed by the frame that has been captured during the accident, the following one, and at the last frame of the sequence. The mark on the upper left corner of the video frames denotes an event. The event is detected right after the accident. In Figure 5.16 (b) the car performs an S-bend. This action is correctly classified as normal activity, and no event is detected.

In Figure 5.17, we can see the score F of several normal movements and an event. The score of the normal activities is above the threshold. The same is true for the event video until the accident happens, than the score drops below the threshold and keeps at this low level. The separability in this F-space is good enough such that a simple threshold successfully separates the normal case from the event.



(a) Accident: the black square marks event frames.



(b) Normal case

Figure 5.16: Several frames from video and the binary motion detection result for an accident (a) and for the normal activity of the car driving in an S-bend (b).

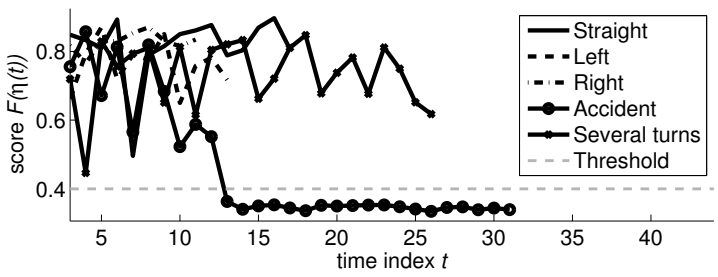


Figure 5.17: The score F computed for various analyzed sequences.

Conclusions

An event detection method was described, which is based on a mixture of linear predictions. As shown above, this model outperforms a GMM and a GHMM in a set of tests. In contrast to GMMs, the LPM uses time dependencies for an improved decision. Furthermore, the LPM is a discriminative model, while the GMM and the GHMM are generative ones. However, LPMs and GMMs have the same simplicity in the inference. In comparison to both GMM and GHMMs, training a LPM is fairly easy and robust to a small size (i.e., number of observations) of the training set, mainly because we do not need to estimate covariances. The ease of training makes the adaptation of the LPM to changes of the normal case also easy.

Some problems with the LPM arise from the solution of the Yule Walker equations. For example, in the presence of outliers, the accuracy of the predictor estimation decreases, and if the variance in the data is too low, the number of values to estimate a linear predictor increases. Solutions to these problems are available within the frame of the Yule Walker equations. Because the LPM builds upon these equations, these solutions are available for the LPMs as well.

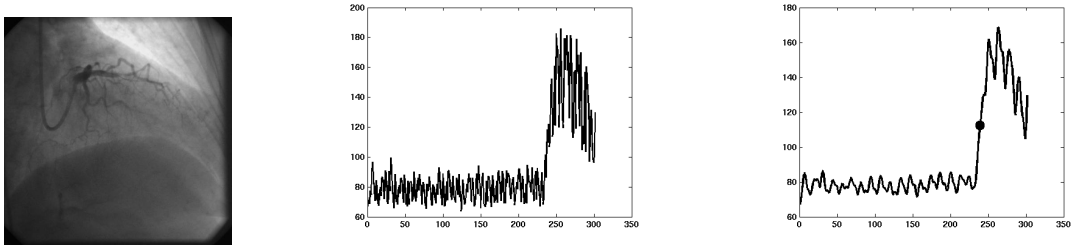
5.2.2 Kernel density estimation and linear-chain conditional random fields for event detection

The LCCRFs described in Section 2.2.2 are a type of LLM. They represent powerful stochastic models that will be adapted here for the purpose of event detection. For an exemplary point event detection application, starting with the simplest algorithm that may constitute a solution we will work our way up to more complicated and powerful ones, culminating with LCCRFs. Thus, the discussion begins by describing the point event detection application, followed by a rather simple solution in the form of a significance test and concludes with the adaptation of the LCCRFs to event detection.

Contrast injections in catheter interventions

Coronary arteries disease occurs when the vessels supplying oxygenated blood to the heart muscle narrow as a consequence of plaque buildup. Treatment is facilitated by Percutaneous transluminal coronary angioplasty. During such an intervention, the narrowing is eliminated usually by inflating a balloon at that position in the vessel. The balloon is brought in place with the help of a fluoroscopic imaging system. The imaging system functions in imaging sessions, which usually differ from one another by the position of the imaging device. To make the vessels visible under X-ray for a while, a bolus of radio-opaque contrast agent is injected through a catheter positioned into the vessels. Algorithms have been developed in Reference [36] to use the contrasted images, also called coronary angiograms (see Figure 5.18 (a)), to build a dynamic vessel roadmap such that the physician still sees the vessels even after the contrast agent has washed out, and can thus navigate better. In this context, it is needed to detect the moment when contrast agent first appears in the fluoroscopic images and turns vessels visible during an imaging session. Considering that previous to that moment only images without vessels (i.e., without contrast agent) have been recorded, this is a point-event detection problem.

The vessel area is measured in each frame of a fluoroscopic image sequence by a vessel-area-related feature that reaches its maximal value as soon as two percents of all pixels in the analyzed image are vessel pixels. As in the beginning of an analyzed sequence there is no



(a) (b) (c)

Figure 5.18: Coronary angiogram (a), vessel-feature curve (b) and filtered vessel-feature curve (c). The vessel event is marked on the filtered feature curve by a bullet.

contrast agent present, the typical appearance of the vessel-feature curve, showing the value of the vessel feature over frame index, includes a region with small values in the beginning followed by a region with higher values, once contrast agent appears. The appearance of vessels is thus usually accompanied by a jump on the feature curve. An example of a typical feature curve is shown in Figure 5.18 (b). Before looking for events, the feature curve is filtered by a recursive low-pass filter, as shown in Figure 5.18 (c). This is supposed to eliminate outliers that are sudden large variations of the feature due to other reasons than the appearance of contrast agent, like for example, due to spurious vessels or the influence of the AGC.

This data has some interesting properties that make it challenging and representative for other event detection problems. It depends on patient, machine settings and properties and it may be influenced by the way the surgeon uses the fluoroscopic device. In particular the interactions of the surgeon, but also patient motion may lead to strong changes in contrast over a relatively small number of frames, and as a consequence to an atypical feature curve. The contrast is set with the help of the automatic gain control unit (AGC). The AGC adapts the contrast when for example, the field of view of the fluoroscopic device changes from one position to another with respect to the body of the patient, or when the patient moves, such that areas of the body with different X-ray absorption coefficients appear in the field of view. Thus, the feature curve corresponding to each sequence may have different unknown statistical properties.

Setup As all images recorded until the vessels appear show only the background, the problem of detecting the appearance of vessels – and implicitly the injection of contrast agent – with the help of the feature curve is reformulated like an event-detection problem. It is considered that an image showing no vessels represents the normal case, and an image with vessels represents the event. Usually there are 10 to 15 seconds from the moment a recording begins until vessels appear. At a frame rate of at least 12 fps, this gives quite a large sample of background images, i.e., a "normal" training set for event detection algorithms.

The database for this application includes 23 video sequences. The sequences were recorded during 10 different interventions. All sequences were acquired under similar conditions, i.e., constant X-ray dosage, while patient and table movement was possible. The patients were mostly allowed to breath freely, but in some cases they were required to hold their breath. Some patients had also an open-heart surgery, and the sewing wires were visible in the analyzed sequences. The analyzed sequences contained images with a resolution of 512×512 pixels and were quantized to eight bits. To compute the vessel-area-related feature, the same parameters

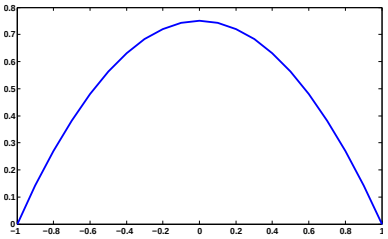


Figure 5.19: The Epanechnikov kernel.

set was used for all sequences.

Nonparametric density estimation for improved point-event detection

In Reference [36] a method has been described to detect the first image of a contrast agent injection by means of a significance test. As shown next, this method may be improved by estimating the likelihood of the null hypothesis by means of nonparametric pdf estimation (see Section 2.1.1), rather than parametric under the Gaussian assumption.

$p(y(n)|H_0)$ is estimated from the samples acquired during the first seconds of an imaging session. Instead of assuming this pdf to be Gaussian and estimate it parametrically, next it will be estimated nonparametrically by kernel smoothing. For this purpose the optimal *Epanechnikov kernel* will be used. The Epanechnikov kernel is defined as [68]:

$$K_E = \begin{cases} \frac{3}{4} \cdot (1 - x^2) & \text{for} \quad |x| \leq 1 \\ 0 & \text{else} \end{cases} . \tag{5.12}$$

A plot of the kernel is known in Figure 5.19.

The nonparametric estimate of $p(y(n)|H_0)$ is then computed according to equation (2.13) [183]. The bandwidth is fixed and is computed using equation (2.15), with $\tilde{\sigma}$ estimated with the help of the SIQR as in equation (3.3).

The event-threshold T is determined such that the probability of $y(n)$ exceeding T given H_0 is α, the significance level. A typical value for α is $\alpha = 1 \cdot 10^{-4}$. T is computed by inverting $\Pr(y(n) > T|H_0) = \alpha$ based on $p(y(n)|H_0)$. As soon as the vessel-area-related feature $y(n)$ exceeds this threshold, the event is detected at the corresponding image.

Experiments and discussion. The significance level used during the experiments was set to $\alpha = 3 \cdot 10^{-3}$. The size of the kernel was $h = 2.3684$. These values were established empirically with the help of six sequences. During testing, α and h were held constant for all sequences. Alternatively we have also conducted experiments where h was computed for each analyzed sequence independently. The classification results do not change, however such an approach is more robust, as the algorithm adapts better to the data.

For the learning phase 72 images were used, which are acquired over a period of six seconds, with a frame rate of 12fps. This is justified by the need to sample at least one complete respiration cycle, considering that under normal circumstances, a human breathes between 15 to 20 times a minute.

The algorithm described above gives the solution to a classification problem – i.e. for each new image decide if it shows any vessels. It is evaluated here using discrepancy methods [31], [192], by analyzing the frequency of correct and incorrect decisions relative to a golden standard [171]. The golden-standard was achieved after detecting per expert-visual analysis the first image which shows vessels. Taking into consideration the variability of such a visual segmentation, it is considered that the automatic detection was successful if it falls in an interval of plus/minus twelve images from the expert-reference. In 19 sequences the detection was successful. In the remaining four sequences, the algorithm failed to detect the contrast burst correctly. This was due to the fact that the feature curve did not show the typical behavior as previously discussed.

By comparison to the algorithm described in Reference [36], where the likelihood of the normal case is estimated parametrically under the Gaussian assumption, the new algorithm represents a 19% improvement, as the old one successfully detected the event in only 15 sequences.

Conclusions. This event detection algorithm works under a set of assumptions. The most important one is that the first six seconds of an imaging session show no vessels. As in this interval the "training" data is collected, it should include a good sample of the normal case and cover a few heart beats and a full respiration cycle. This is particularly needed for projection angles which permit also the visualization of the diaphragm, as in this case, the vessel-area-related feature follows the variation of the projected diaphragm area. This is a consequence of the AGC unit trying to compensate the variation of the image brightness caused by the moving diaphragm.

The contrast agent that reaches the vessels mixes with blood and is carried away by this, thus at the end of the imaging session, there is usually no contrast agent left into the vessels. However, the contrast agent present in the catheter is not washed away by blood. If a new imaging session begins with the catheter with contrast agent in place, then there will be something similar to vessels visible from the very beginning. By handling each imaging session independently, the algorithm described here is also robust against this case and can detect the moment when contrast agent reaches the vessels, even if some parts of the catheter with contrast agent in it are visible in the analyzed sequence.

Here the pdf of the feature given the normal case is estimated nonparametrically. Nonparametric estimation provides a better, robuster statistical model for the specific problem we are dealing with. This leads to better results in comparison to a parametric approach. Using the Epanechnikov kernel, the estimation result does no longer have an infinite support. Furthermore, this kernel was proven to fulfill some optimality criteria for nonparametric density estimation [183], as it minimizes the mean squared error of the estimator.

Filtering the feature curve by a low-pass filter eliminates outliers and returns the analyzed feature curve to its typical appearance. Nevertheless, there are cases when this is not enough, and the feature curve still maintains an untypical appearance. Then, more powerful methods, able to work with even less assumptions, or better said despite the lack of much prior information, are needed.

Conditional random fields applied to event detection

Assuming the feature curve is "nice", meaning that it shows the expected behavior of rising very fast to a maximum and then decreasing slowly over time, relatively simple methods like the

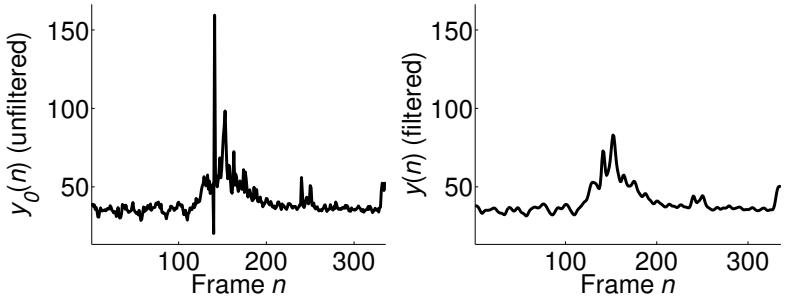


Figure 5.20: Original vessel feature curve (left hand side) and the filtered histogram feature curves (right hand side) not entirely showing a typical behavior.

significance-testing based event detection algorithm of the previous section will work successfully. Nevertheless, there are instances when, the feature curves are more difficult to analyze, as they do not show the expected behavior. Such examples are shown in Figure 5.20 and also in Figure 5.24 (e). The event may still be detectable, but not according to the previous intuition with respect to the feature curve. Thus, significance-test based methods are inappropriate and more powerful methods are needed. Such methods are discussed in this section. They work on feature vectors that are generated from the vessel-feature curve. For this purpose, the feature curve is windowed and from each window a feature vector is generated. The feature curve is thus transformed into a sequence of feature vectors.

LCCRFs and MEMMs, are finite state models [87, 182, 116, 136] used for classification problems [136, 116]. Neither the MEMMs nor the LCCRFs can be used directly for event detection, as they cannot accomodate a state not present in the training set. Next the LCCRFs will be adapted to event detection, in the form of the Event Detection Log-Linear Model (edLLM). Consequently, the edLLM is placed within the LLM [182, 116] framework and is related besides the LCCRFs also to the MEMMs [136] that were discussed in Section 2.2.1. It will be shown that the edLLM is able to properly detect events even in the more difficult contrast-agent sequences sequences, where significance-based methods fail.

The path towards an event-detection LLM. When using a MEMM, the probability to observe a certain state depends only on the states before, while in LCCRFs, it depends on the states before and after the actual state. Hence, in MEMMs, we pass forward information, while in LCCRFs, the information is passed forward and backward. This has two important consequences: (i) the LCCRFs have potentially a higher accuracy because they use more information for each time step and (ii) they cannot be applied directly for online problems, because the whole data sequence needs to be measured before the inference. At the same time, similar to MEMMs, the LCCRFs can use non-independent relationships extending over several observations.

LCCRFs need a labeled data set for training. The trained model is applied to new data sets to generate a corresponding label sequence. All labels that are to be assigned during the operation phase must be present in the training set as well. For event detection problems, as there are no labeled event-samples in the training set, the usual training algorithms for LCCRFs [87] cannot be used. There exist methods that are able to work with partly labeled data [106], but these still assume that at least some labels are available for each class. Next, it will be described how to extend the training such that it works when one label is not at all present in the training set. Furthermore, new methods will be introduced to conduct the inference. This represents then

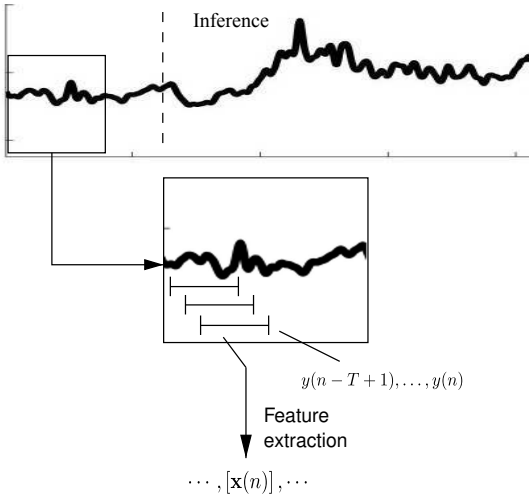


Figure 5.21: Schematic representation of the feature extraction process. Short sequences of the vessel feature curve are selected, using a sliding window with $T-1$ overlap. From each such batch consisting of $y(n+1), y(n+2)\ldots, y(n+T)$, we extract a normalized feature vector $\mathbf{x}(n)$. This results in a sequence of feature vectors, which we analyze with the edLLM.

the edLLM. As the "normal" case can include several labels, this provides the opportunity to correctly model a more complicated scenario, where several potentially very different things are considered normal.

Feature extraction. As already pointed out, now the data is analyzed batch-wise. From each batch a feature vector is extracted. For the contrast-agent detection problem, each batch considers several vessel features corresponding to a specific portion of the feature curve. This portion is selected by means of a sliding window of length T with $T-1$ overlap. Therefore, one batch – given by the set of vessel feature corresponding to the frames $y(n), y(n+1), \ldots, y(n+T-1)$ – corresponds to one frame of fluoroscopic video, and for each frame a feature vector $\mathbf{x}(n)$ is computed.

The feature vector is related to the mean, curvature and slope in each batch. An ordered set of feature vectors constitutes a *feature vector sequence*. A decision is returned for each feature vector and thus for each frame of the video sequence, starting at frame T.

In the following, the components of the feature vector are introduced by describing the functions used to compute the mean, curvature and slope. These features are general enough to be applicable to many different time series and for our application describe local properties of the vessel feature curve.

The feature vectors are normalized to reduce the influence of outliers. As previously discussed, a vessel feature is an outlier if it is either very large or very small in comparison to both, the previous and next vessel feature on the feature curve. A schematic representation of the feature extraction process can be seen in Figure 5.21.

Mean-related components of the feature vector. The first two components of the feature vector are related to the mean of the feature curve. Let α and β be two scalars with $\alpha, \beta \in (0,1)$. Then $\mu(n+1) = \alpha\mu(n) + (1-\alpha)\sum_{m=0}^{T-1} y(n+m)$ is a slowly adapting mean value, and

$\nu(n+1) = \beta\nu(n)+(1-\beta)\mu(n)$ a fast adapting one. These are called "mean values" because for stationary signals, the adapting mean values converge to the mean of the signal with increasing T. α and β are adaptation rates and depend on the frame rate of the analyzed video. Further, $d_n(m) = y(m) - \mu(n)$ and $\tilde{d}_n(m) = y(m) - \nu(n)$ are differences between those mean values and a vessel feature.

As our intention is to compare the mean value of a batch to the adapting mean values, the first two components of the feature vector are

$$\begin{aligned}\xi_1(n) &= \left(1 - \sqrt{\frac{\sum_{m=0}^{T-1} d_n(m+n)^2}{2 \cdot T \cdot \hat{\sigma}^2}}\right) \cdot 10,\\ \xi_2(n) &= \left(1 - \sqrt{\frac{\sum_{m=0}^{T-1} \tilde{d}_n(m+n)^2}{5 \cdot T \cdot \hat{\sigma}^2}}\right) \cdot 10,\end{aligned}$$

where $\hat{\sigma}^2$ is the estimated variance of the training data. These functions are positive if the mean of the batch is close to the adapting mean values, and negative otherwise. The scaling within the roots control what is called "close". The factor 10 is to emphasize these features, because they act in a manner similar to the statistical test from the previous section.

Curvature-related components of the feature vector. The next components of the feature vector are related to the curvature. These features are computed as the difference of each element of a batch to the mean of a batch. The next T components of the feature vector are then

$$\xi_{2+k+1}(n) = y(n+k) - \frac{1}{T}\sum_{m=0}^{T-1} y(n+m),$$

for $k = 0, 1, 2, \ldots, T-1$.

Slope-related components of the feature vector. The last L components of the feature vector are used to describe the slope of a batch. The slope of a batch is defined as the slope of its linear regression, that is $\hat{b}(n) = \arg\min_{b\in\mathbb{R}} \sum_{m=0}^{T-1} \left(y(n+m) - y(n) - b \cdot m\right)^2$. With its help, L new components of the feature vector are computed. For this purpose a set of L *sample slopes*, are used. A sample slope w_l, $l = 1, 2, \ldots, L$, is defined as $w_l = \frac{L-l}{L-1} b_{min} + \frac{l-1}{L-1} b_{max}$. b_{min} is the minimal slope that is observed in the training, "normal-case" batches, and b_{max} the maximal slope.

Then, the slope-related components of the feature vector are computed as

$$\xi_{2+T+l}(n) = 1 - \left(\frac{\hat{b}(n) - w_l}{w_{l+1} - w_l}\right)^2$$

for $l = 1, 2, \ldots, L$. This function is positive if a new slope is close to the respective sample slope, and negative otherwise.

Normalized feature vectors. For each batch $y(n), y(n+1), \ldots, y(n+T-1)$, a vector $\xi(n) = [\xi_i(n)]_{i=1}^N$ with $N = 2 + T + L$ is defined. The functions $\xi_1, \xi_2, \ldots, \xi_N$ describe properties of each batch, but are affected by outliers. Let $v(n) = \sum_{m=1}^{T-1}(y(n+m) - y(n+m-$

$1))^2$ be the (unnormalized) *local variation*. Then, the normalized outlier-robust feature vector is $\mathbf{x}(n) = \frac{1}{v(n)}\xi(n)$. $\mathbf{x}(n)$ is then called the *feature vector* at lag n.

The local variation increases in the presence of large outliers. This normalization does not remove the possibility to detect events in the feature vector sequence, because any real event is visible for several successive frames.

Event detection LLM. The edLLM is somewhat similar to multiclass logistic regression (see Reference [20] pp. 209). The LCCRF part consists in the way we compute the function Φ, which relates states and observations at a specific time index. The idea of the edLLM is to label a feature-vector sequence of a certain length at once. As each feature vector is computed from a batch of vessel features, a larger context is thus used for an improved decision. The probability of a state sequence, given a sequence of feature vectors is determined similar to the case of LCCRFs [87]. The algorithm works with sequences of feature vectors that are defined again with the help of a sliding window. Thus, it may be also used online.

The training data is considered to represent a single sequence of feature vectors, thus no sliding window is used during training. Training is conducted similar to LCCRFs and MEMMs, except for a penalty term that is defined such as to allow the introduction of a state that is not present in the training data. Thus, we may train for event detection on "normal-case" data.

Let $\mathbf{x}_i = \{\mathbf{x}(t_i+1), \mathbf{x}(t_i+2), \ldots, \mathbf{x}(t_i+M)\}$ with $\mathbf{x}(n) \in \mathbb{R}^N, \forall n$ be a feature vector sequence of length $M \in \mathbb{N}$, then the corresponding state sequence is $\mathbf{s}_i = \{s(t_i+1), s(t_i+2), \ldots, s(t_i+M)\}$, with $s(n) \in \{\zeta_0, \zeta_1, \zeta_2, \ldots, \zeta_K\}$. Next it is assumed that $\zeta_1, \zeta_2, \ldots, \zeta_K$ are the states that describe the normal case, and ζ_0 is the event state, which is related in our case to the occurrence of contrast agent. The training data includes no events.

Log-linear model. The following notation is used here: $\mathbf{x}_i = [\mathbf{x}(t_i+m)]_{m=1}^M$ represents a feature vector sequence of length M, starting at $t_i \in \mathbb{N}_0$. The training data represents a single sequence of feature vectors; for the training data we have $i = 0$, therefore the training data is contained in $\mathbf{x}_0$. For a shorter notation $\mathbf{x}_i(m) = \mathbf{x}(t_i+m)$ is used. Similarly, for the states $\mathbf{s}_i = [s(t_i+m)]_{m=1}^M$ and $s_i(m) = s(t_i+m)$ are used.

The probability of $\mathbf{s}_i$, given $\mathbf{x}_i$ and $s_i(0)$ is defined by [87]

$$p(\mathbf{s}_i|\mathbf{x}_i, s_i(0)) = \frac{\exp\left(\sum_{m=1}^{M} \boldsymbol{\lambda}^\top \Phi(s_i(m-1), s_i(m), \mathbf{x}_i, m)\right)}{Z(\mathbf{x}_i)}, \tag{5.13}$$

where $Z(\mathbf{x}_i)$ is a normalization value such that $p(\mathbf{s}_i|\mathbf{x}_i, s_i(0))$ is a probability.

The function Φ establishes the relationship between the feature vectors and the states. It is defined by

$$\Phi(s_1, s_2, \mathbf{x}_i, n) = \begin{bmatrix} [\mathbf{x}_i(n) \cdot [\![s_2 = \zeta_k]\!]]_{k=1}^K \\ \left[[[\![s_1 = \zeta_j]\!] \cdot [\![s_2 = \zeta_k]\!]]_{k=1}^K\right]_{j=1}^K \\ \mathbf{x}_i(n) \cdot [\![s_2 = \zeta_0]\!] \end{bmatrix}. \tag{5.14}$$

with $\mathbf{x}_i(n) \in \mathbb{R}^N$, $\Phi(s_1, s_2, \mathbf{x}_i, n) \in \mathbb{R}^{(N+1)K+K^2}$ and $[\![\cdot]\!]$ the Iverson bracket [87].

$\boldsymbol{\lambda}$ is a weighting vector. This vector contains information about: (i) feature-vector components per state (including the event state), and (ii) state transitions. The entries in this vector are established during training. In general, the entries in $\boldsymbol{\lambda}$ may be characterized as follows:

- The first block of N entries is related to the first state.
- The second block of N entries is related to the second state, and so on until the K'th block of N entries, which is related to the last possible state.
- The next K^2 states describe the transition probabilities between normal-case states.
- The last block of N entries describes the event state.

For each sequence i, an initial label $s_i(0) = s(t_i)$ is needed. For the training data $\mathbf{x}_0$, there is no information about $s_0(0) = s(0)$, so an arbitrary symbol is used, such that $s(0) \notin \{\zeta_0, \zeta_1, \ldots, \zeta_K\}$ [87]. For $i = 1$, the initial label is $s_1(0) = s(t_1)$ is the label of the last feature vector from $\mathbf{x}_0$. For $i > 1$, $s_i(0) = s(t_i)$ the label of the last feature vector of the previous sequence.

Example. To illustrate the concepts described above, it is assumed that a sequence of length $M = 2$ of 2D feature vectors ($N = 2$) has to be analyzed. First we will assume we are in the trainig phase, thus $i = 0$. Then we will asume we are in the operative phase with $i > 1$. In each case we will discuss several possible state sequences of length two for the observation vectors $\mathbf{x}_i(1) \equiv \mathbf{x}(t_i + 1) = [a, b]^\top$ and $\mathbf{x}_i(2) \equiv \mathbf{x}(t_i + 2) = [c, d]^\top$. Further, it is also assumed that $K = 2$, that is, $s_i(n) \in \{\zeta_0, \zeta_1, \zeta_2\}$, $n = 1, 2$. With this parameters, $(K + 1)^M = 9$ sequences are possible.

In the training phase, the initial state is $s_0(0) = \zeta_x \notin \{\zeta_0, \zeta_1, \zeta_2\}$. We will discuss two state sequences.

- For the state sequence $[\zeta_2, \zeta_1]^\top$ at lag one (i.e., for the first vector of the training sequence) the transitions are ignored and the entries from five to eight are all zero, thus $\Phi(\zeta_x, \zeta_2, \mathbf{x}_0, 1) = [0, 0, a, b, 0, 0, 0, 0, 0, 0]^\top$. At lag two we have that $\Phi(\zeta_2, \zeta_1, \mathbf{x}_i, 2) = [c, d, 0, 0, 0, 0, 1, 0, 0, 0]^\top$. The first two entries are $\mathbf{x}_i(2)$. The seventh entry is one due to the transition from ζ_2 to ζ_1.
- For the state sequence $[\zeta_1, \zeta_1]^\top$ at lag one $\Phi(\zeta_x, \zeta_1, \mathbf{x}_0, 1) = [a, b, 0, 0, 0, 0, 0, 0, 0, 0]^\top$. Then $\Phi(\zeta_1, \zeta_1, \mathbf{x}_0, 2) = [c, d, 0, 0, 1, 0, 0, 0, 0, 0]^\top$. The first two entries are again $\mathbf{x}_i(2)$. The fifth entry is one as the state is kept.

In the operative phase, assuming that the label of the initial state is $s_i(0) = \zeta_1$, we discuss four different state sequences. In comparison to the training phase, things are different at lag one and similar starting at lag two. Furthermore, we have the possibility of observing an event.

- For the state sequence $[\zeta_1, \zeta_2]^\top$, at lag one $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 1) = [a, b, 0, 0, 1, 0, 0, 0, 0, 0]^\top$. At lag two, the state changes and $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 2) = [0, 0, c, d, 0, 1, 0, 0, 0, 0]^\top$, where the third and fourth entry of $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 2)$ are $\mathbf{x}_i(2)$ and the sixth entry is one due to the transition from ζ_1 to ζ_2.
- For the state sequence $[\zeta_1, \zeta_1]^\top$, at lag one $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 1) = [a, b, 0, 0, 1, 0, 0, 0, 0, 0]^\top$. At lag two the state is preserved and $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 2) = [c, d, 0, 0, 1, 0, 0, 0, 0, 0]^\top$. The first two entries are $\mathbf{x}_i(2)$. The fifth entry of $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 2)$ indicates a "transition" within the investigated feature vector sequence from state ζ_1 to ζ_1 (i.e., the systems stays in state one).

- For the state sequence $[\zeta_2, \zeta_1]^\top$ at lag one $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 1) = [0, 0, a, b, 0, 1, 0, 0, 0, 0]^\top$. The third and fourth entries are $\mathbf{x}_i(1)$. The sixth entry is one symbolizing a change of state from ζ_1 to ζ_2. At lag two, the state changes and we have $\Phi(\zeta_2, \zeta_1, \mathbf{x}_i, 2) = [c, d, 0, 0, 0, 0, 1, 0, 0, 0]^\top$. The seventh entry is one due to the transition from ζ_2 to ζ_1.
- For the state sequence $[\zeta_1, \zeta_0]^\top$, thus assuming an event at lag two (that is $s(2) = \zeta_0$), we have $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 1) = [a, b, 0, 0, 1, 0, 0, 0, 0, 0]^\top$ at lag one and at lag two $\Phi(\zeta_1, \zeta_0, \mathbf{x}_i, 2) = [0, 0, 0, 0, 0, 0, 0, 0, c, d]^\top$.

The unnormalized probability is computed according to equation (5.13) as

$$\tilde{p}(\mathbf{s}|\mathbf{x}_i, s_i(0)) = \exp\left(\sum_{m=1}^{2} \boldsymbol{\lambda}^\top \Phi(s(m-1), s(m), \mathbf{x}_i, m)\right). \tag{5.15}$$

In the training phase, with the weighting vector $\boldsymbol{\lambda} = [\lambda(j)]_{j=1}^{10} = [1, -1, -1, 1, 1, 0.5, 0.5, 1, 1, 1]^\top$ we have that

$$\begin{aligned}\tilde{p}([\zeta_1, \zeta_1]^\top|\mathbf{x}_0, s_0(0)) &= \exp\left[\lambda^\top \Phi(\zeta_x, \zeta_1, \mathbf{x}_0, 1) + \lambda^\top \Phi(\zeta_1, \zeta_1, \mathbf{x}_0, 2)\right] \\ &= \exp(a - b + c - d + 1) \\ &= \exp(a - b) \cdot \exp(c - d + 1)\end{aligned}$$

and

$$\begin{aligned}\tilde{p}([\zeta_2, \zeta_1]^\top|\mathbf{x}_0, s_0(0)) &= \exp(-a + b + c - d + 0.5) \\ &= \exp(b - a) \cdot \exp(c - d + 0.5).\end{aligned}$$

We analyze next the first sequence of observations in the operative phase. Under the same conditions as above, with the same $\mathbf{x}_1, \boldsymbol{\lambda}$ and with $s_1(0) = \zeta_1$ we have for the state sequence $[\zeta_1, \zeta_2]^\top$:

$$\begin{aligned}\tilde{p}([\zeta_1, \zeta_2]^\top|\mathbf{x}_1, s_1(0)) &= \exp\left[\lambda^\top \Phi(\zeta_1, \zeta_1, \mathbf{x}_1, 1) + \lambda^\top \Phi(\zeta_1, \zeta_2, \mathbf{x}_1, 2)\right] \\ &= \exp(a - b + 1 + d - c + 0.5) \\ &= \exp(a - b) \cdot \exp(d - c + 1.5).\end{aligned}$$

Unde the same conditions as above we assume now we observe the state sequence $[\zeta_1, \zeta_0]^\top$. Then, the unnormalized probability of an event at lag two in the operative phase is:

$$\begin{aligned}\tilde{p}([\zeta_1, \zeta_0]^\top|\mathbf{x}_1, s_1(0)) &= \exp(a - b + 1 + c + d) \\ &= \exp(a - b) \cdot \exp(c + d + 1).\end{aligned}$$

Depending on the values in each of the two observed feature vectors and on $\boldsymbol{\lambda}$, a certain sequence of states will have the highest probability. Clearly the weighting vector $\boldsymbol{\lambda}$ regulates which state sequence is the most probable. In this example, we can interpret $\boldsymbol{\lambda}$ as follows:

- The first $N = 2$ entries of $\boldsymbol{\lambda}$ describe the state ζ_1. This means: (i) a positive value "prefers" a corresponding positive value in $\mathbf{x}_i$ (if an entry in $\boldsymbol{\lambda}$ is positive and the corresponding one in $\mathbf{x}_i$ too, the probability for this state is higher than otherwise), (ii) a negative value in $\boldsymbol{\lambda}$ "prefers" a negative value in $\mathbf{x}_i$, and (iii) a zero in $\boldsymbol{\lambda}$ can be interpreted as "no influence" on the probability.

- The next $N = 2$ entries describe the state ζ_2.
- The next $K^2 = 4$ entries describe the transition probabilities between the normal case states.
- The last $N = 2$ entries describe the event.

Training. The edLLM labels each observed feature vector as either normal or event. The simplest thing to do would be to design a model with just two states, viz., normal and event. Nevertheless, the normal case is usually rich enough to need more than just one label to be able to achieve a satisfactory description for event-detection purposes. Therefore, the edLLM includes several "normal" states and one "event" state.

Assuming that the training set includes data that is described as "normal" without additional labels for the sub-cases of the normal case, in a first step the states corresponding to the "normal" case need to be assigned to the training data. This is an unsupervised classification problem, where each feature vector from the training set receives one label from the set $\{\zeta_1, \zeta_2, \ldots, \zeta_K\}$. Within the context of the contrast-agent-detection application, let y_{min} be the minimum vessel feature in the training data and y_{max} be the maximum one. Defining a_q as $a_q = \frac{K-q+1}{K} y_{min} + (1 - \frac{K-q+1}{K}) y_{max}$ for $q = 1, 2, \ldots, K+1$, the training data is labeled as $s(n) = \zeta_q$ if $a_q \leq \frac{1}{T} \sum_{m=0}^{T-1} y(n+m) < a_{q+1}$.

After assigning the labels of the normal case, the training phase includes the estimation of the rest of the model parameters. The weighting vector $\boldsymbol{\lambda}$ is estimated in the training phase. It effectively selects the feature-vector components that are important for a successful description of the normal case and provides also a description of the event state. During training, $p(\mathbf{s}_0|\mathbf{x}_0, s(0))$ is maximized with respect to $\boldsymbol{\lambda}$, given the training data $\mathbf{x}_0$ and a corresponding state sequence $\mathbf{s}_0$.

Let t_0 be the start index of the training data (i.e., $t_0 = 0$) and M_0 the number of training feature vectors, then

$$\Phi(\mathbf{s}_0, \mathbf{x}_0) = \sum_{m=1}^{M_0} \Phi(s_0(m-1), s_0(m), \mathbf{x}_0, m).$$

During training $L_{\boldsymbol{\lambda}}$, which is the penalized log-likelihood of $p(\mathbf{s}_0|\mathbf{x}_0, s(0))$, with $s(0)$ some initial state, is optimized. $L_{\boldsymbol{\lambda}}$ is defined as

$$\begin{aligned} L_{\boldsymbol{\lambda}} &= \log\left(p(\mathbf{s}_0|\mathbf{x}_0, s(0))\right) - F(\boldsymbol{\lambda}) \\ &= \boldsymbol{\lambda}^\top \Phi(\mathbf{s}_0, \mathbf{x}_0) - \log Z(\mathbf{x}_0) - F(\boldsymbol{\lambda}), \end{aligned} \tag{5.16}$$

where $F(\boldsymbol{\lambda}) = \frac{\|\mathbf{D}\boldsymbol{\lambda}\|^2}{2} - \tilde{\mathbf{e}}^\top \boldsymbol{\lambda}$ with $\tilde{\mathbf{e}} = [\tilde{e}_k]_{k=1}^{(N+1)K+K^2}$, and

$$\tilde{e}_k = \begin{cases} \sum_{i=1}^{M_0} [\![s(i) \neq \zeta_j]\!] & \text{if } (j-1) \cdot N < k \leq j \cdot N, \\ 0 & \text{otherwise.} \end{cases}$$

The special penalty term $F(\boldsymbol{\lambda})$ is necessary for two purposes: to avoid overfitting [87], and to *adapt the training to the event detection setup*. For the former purpose, $\tilde{\mathbf{e}}$ compensates the

different numbers of training vectors for each state. For the latter purpose, the *penalty matrix* $\mathbf{D}$ is defined as $\mathbf{D} = [D(i,j)]_{i,j=1}^{(N+1)K+K^2}$ with

$$D(i,j) = \begin{cases} 1 & \text{if } i = j;\ \ i,j \leq N \cdot K + K^2 \\ \frac{N}{2N+1} & \text{if } i = j;\ \ i,j > N \cdot K + K^2 \\ \frac{1}{2N+1} & \text{if } i \neq j;\ \ i,j > N \cdot K + K^2 \\ \frac{1}{N} & \text{if } i = j + N \cdot K + K^2 - N \cdot (k-1);\ \ k = 1,2,\ldots,K;\ \ i > N \cdot K + K^2 \\ 0 & \text{otherwise.} \end{cases}$$

As it may be observed, some components $\mathbf{D}$ have values different from both one and zero. These values were established empirically.

With this penalty matrix the event state ζ_0 can be accommodated (i.e., compute the corresponding entries of $\boldsymbol{\lambda}$), even though the training data $\mathbf{x}_0$ does not include any events, that is $s_0(m) \neq \zeta_0$, $m = 1,2,\ldots,M_0$.

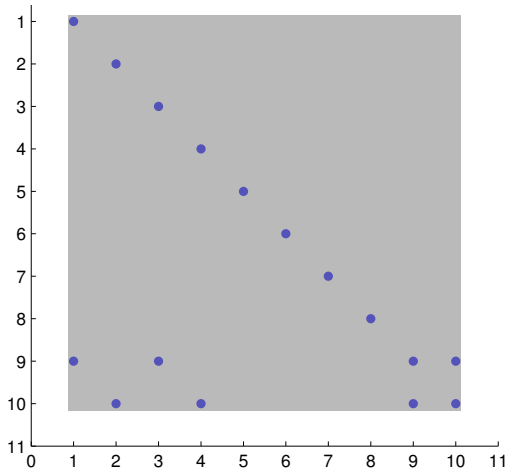


Figure 5.22: The penalty matrix $\mathbf{D}$ for our example, with $N = 2$ and $K = 2$. Only the entries different from zero are marked.

While computing the optimal $\boldsymbol{\lambda}$ the following happens: the first $N \cdot K + K^2$ entries of the weight vector are related to the normal case and will be established according to the training data, while the last N entries that are related to the event will be determined by the previous entries (i.e., the normal-case entries) and the condition that the penalty term must be chosen such that the objective function is optimized. The penalty matrix for our example is shown in Figure 5.22.

Because we want to determine $\boldsymbol{\lambda}$ such that the penalized log-likelihood (see equation (5.16)) is maximized, the training of the edLLM is an optimization problem. The gradient of the penalized log-likelihood is

$$\nabla L_{\boldsymbol{\lambda}} = \Phi(\mathbf{s}_0, \mathbf{x}_0) - E(\mathbf{s}|\mathbf{x}_0, \boldsymbol{\lambda}) - \mathbf{D}\boldsymbol{\lambda} + \tilde{\mathbf{e}},$$

where $E(\mathbf{s}|\mathbf{x}_0, \boldsymbol{\lambda})$ is the expected value of the sequence $\mathbf{s}$, given $\mathbf{x}_0$ and $\boldsymbol{\lambda}$. An efficient algorithm to compute $E(\mathbf{s}|\mathbf{x}_0, \boldsymbol{\lambda})$ can be found in [87]. The training proceeds in several iterations. At iteration τ, $\boldsymbol{\lambda}$ is updated by $\boldsymbol{\lambda}^{(\tau+1)} = \boldsymbol{\lambda}^{(\tau)} + 0.3\nabla L_{\boldsymbol{\lambda}^{(\tau)}}$. The constant 0.3 in the update of $\boldsymbol{\lambda}$ is selected empirically.

Inference. With a trained model, the probability of a state sequence for a new batch/sequence of feature vectors needs to be computed. For the event detection, the most interesting task is to measure the probability to detect an event, that is the probability that the state ζ_0, appears in the state sequence corresponding to a certain feature vector batch.

As described before, $i > 0$ indicates the i-th feature vector batch of length M, with t_i as starting point. The whole sequence of feature vectors $\mathbf{x}$ is analyzed batchwise to improve the classification results. The batches $\mathbf{x}_i$ overlap by d vectors, and d vectors are re-labeled each time. The first batch $\mathbf{x}_1$ is initialized to include several features that belong to the training set. Each feature vector corresponds to a frame of the X-ray images. This procedure is illustrated in the example from Figure 5.23.

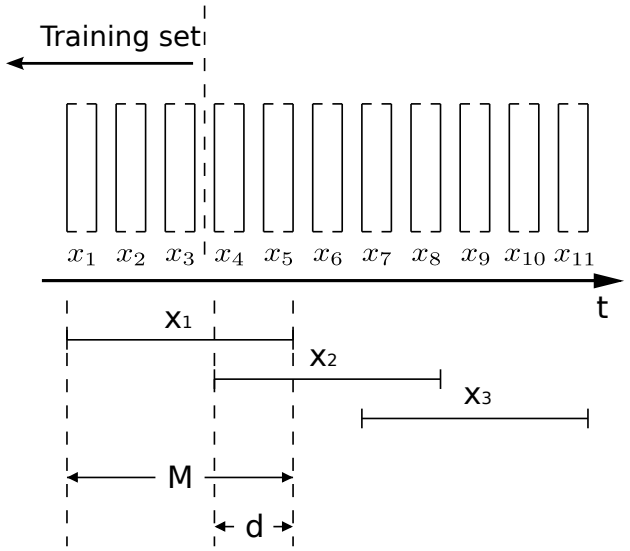


Figure 5.23: In this example the inference is applied on feature vector batches $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ of length $M = 5$ with an overlap of $d = 2$. The firs batch includes vectors from the training set.

For each batch $\mathbf{x}_i$, we would want to estimate a state sequence $\mathbf{s}_i$. An exhaustive search for each possible state sequence corresponding to this feature vector batch is not feasible, because we would have to test $(K+1)^M$ different state sequences. To circumvent this problem, a forward path is defined recursively [87] as

$$\alpha_{i,m}(k) = \frac{1}{Z_\alpha} \sum_{j=0}^{K} \alpha_{i,m-1}(j) \cdot \exp(\boldsymbol{\lambda}^\top \Phi(\zeta_j, \zeta_k, \mathbf{x}_i, m)),$$

and a backward path as

$$\beta_{i,m}(j) = \frac{1}{Z_\beta} \sum_{k=0}^{K} \beta_{i,m+1}(k) \cdot \exp(\boldsymbol{\lambda}^\top \Phi(\zeta_j, \zeta_k, \mathbf{x}_i, m+1)),$$

where $m = 1, 2, \ldots, M$, Z_α and Z_β are constants such that $\sum_{j=1}^{K} \alpha_{i,m}(j) = \sum_{k=1}^{K} \beta_{i,m}(k) = 1$. In this case, $\alpha_{i,m}(k)$ is the probability that ζ_k is observed at lag m in the batch $\mathbf{x}_i$, given the previous states and $\beta_{i,m}(j)$ is the probability of state ζ_j at lag m in the batch $\mathbf{x}_i$, given the

following states. The probability of state ζ_j at time step m in batch $\mathbf{x}_i$ is given by

$$p\left(s_i(m) = \zeta_j | \mathbf{x}_i\right) = \frac{\alpha_{i,m}(j) \cdot \beta_{i,m}(j)}{Z_{\alpha,\beta}}, \tag{5.17}$$

where $Z_{\alpha,\beta}$ is a normalization constant such that $p\left(s_i(m) = \zeta_j | \mathbf{x}_i\right)$ is a probability. This forward-backward algorithm maximizes the information that is used in the end for labeling and therefore the reliability of the method. Also, working with the overlap both past and future relationships with respect to the labeled feature vector are used. The price to pay for this is a delay in obtaining the sought label.

To further detail this aspect of using information both from the past and the future, it is assumed now that $\alpha_{i,m}(k)$ and $\beta_{i,m}(j)$, $j, k = 1, 2, \ldots, K$ have been computed for some batch $\mathbf{x}_i$. To initialize the inference on the next batch $\mathbf{x}_{i+1}$, the forward path initialization is defined as $\alpha_{i+1,0}(k) = p\left(s_i(M-d) = \zeta_k | \mathbf{x}_i\right)$ and the backward path as $\beta_{i+1,M+1}(j) = \frac{1}{K+1}$. With this initialization, inference may be conducted on the batch $\mathbf{x}_{i+1}$. Thus labels are assigned to the feature vectors from $\mathbf{x}_{i+1}$ (including the relabeling of the first d vectors) using for this purpose information from all previous feature vectors in the entire analyzed sequence, as this information is captured in the transition from the feature vector at position $M - d$ in $\mathbf{x}_i$ to the first[3] feature vector of $\mathbf{x}_{i+1}$ as well as to some extent in the labels of the first vectors from $\mathbf{x}_{i+1}$ that were established taking into consideration also the previous $M - d$ vectors.. Conversely, the feature vectors in the overlap region between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are now relabeled using also information from a larger number of "future" feature vectors, (i.e., those at positions $d+1, \ldots, M$ in $\mathbf{x}_{i+1}$) while their previous labels, assigned when analyzing $\mathbf{x}_i$, used information from a smaller number of future vectors[4]. To use this in a meaningful manner, d should be larger than $\frac{M}{2}$, otherwise, some vectors (in the middle of the batch) will be labeled using less information than the others.

With respect to Figure 5.23, to label the vectors $x_7, \ldots, x_{11}$ of the batch $\mathbf{x}_3$, the transition from s_6 is used. Conversely, to relabel x_7 and x_8, while conducting the inference on $\mathbf{x}_3$, now also x_9, x_{10} and x_{11} are used, which represents more information than that used to label x_7 and x_8, while conducting the inference on $\mathbf{x}_2$ – in which case for x_8 we used no information from the "future".

For event detection, $p(s_i(m) = \zeta_0 | \mathbf{x}_i)$ the probability of the event state ζ_0 (see equation (5.17)) needs to be computed. In a straightforward manner, an event is detected if $p(s_i(m) = \zeta_0 | \mathbf{x}_i) > \theta$ for some $0 < \theta < 1$.

CUSUM test. The straightforward method from above is very sensitive to noise in the probability of the event state, which can occur if the number of training feature vectors is limited or the variation in the data is high. A more robust method is discussed next.

The probability for the observation of the event state ζ_0 in a feature vector batch $\mathbf{x}_i$ at lag m is $p(s_i(m) = \zeta_0 | \mathbf{x}_i)$. For a robust decision, a CUSUM test (see Section 1.3.3) is used to establish if an event has occurred. For this purpose, $c(m)$ is defined as

$$c(m) = c(m-1) + p\left(s_i(m) = \zeta_0 | \mathbf{x}_i\right) - c_0,$$

[3]Because of the overlap, the first feature vector of $\mathbf{x}_{i+1}$ is at the same time the feature vector at position $M - d + 1$ in $\mathbf{x}_i$.

[4]To the limit, for the feature vector at position M in batch $\mathbf{x}_i$, no information from future feature vectors is used.

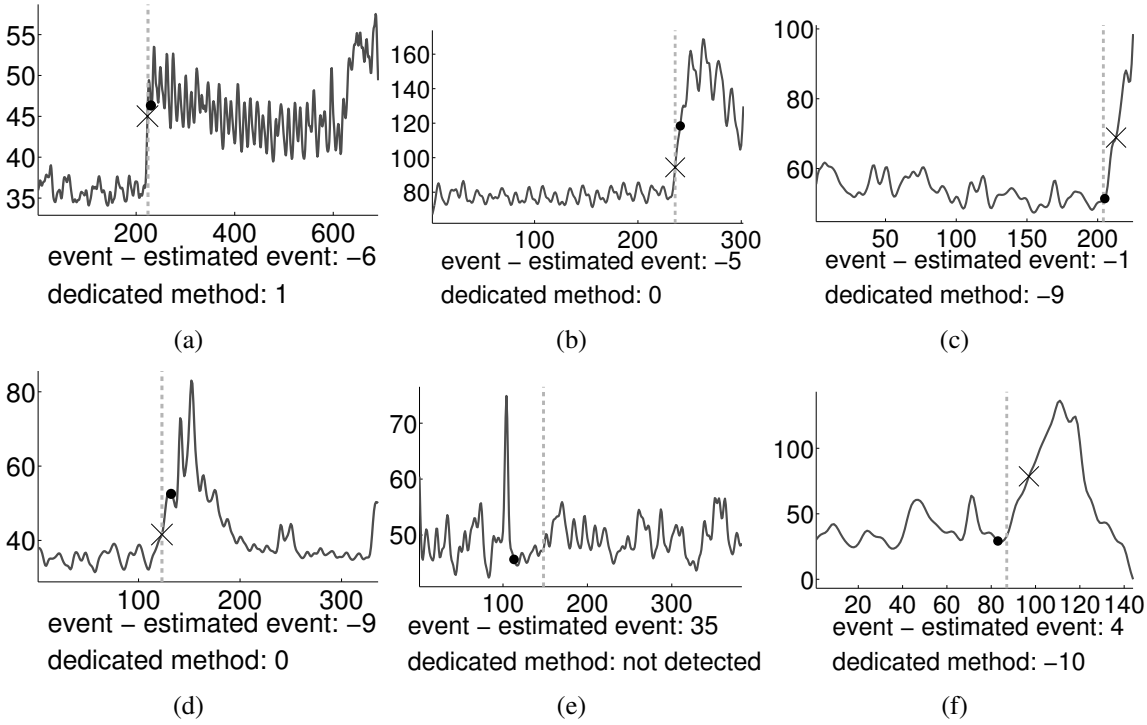


Figure 5.24: Shown here are six vessel-feature curves, together with the manual ground truth (dashed gray line), the edLLM result (black dot) and significance-test result (black cross).

where c_0 is estimated such that $c(m) \leq 1$ for $m \leq M_0$, that is, no event is detected in the training data. An event has occurred if $c(m) > 1$. $c(m)$ is set to zero either if an event has occurred or if $c(m) \leq 0$.

Experiments, results, and discussion. The algorithm is tested on the 23 video sequences available in the contrast-agent-detection dataset. Some results are shown in Figure 5.24. As it may be seen, the edLLM is a more powerful (i.e., general) event detection algorithm which can successfully analyze very different time series.

To establish the parameters needed to apply the edLLM, three randomly chosen sequences have been used as training set. Using the leave-one-out cross validation different parameters (L from 3 to 9, T from 10 to 50) have been tested without noticing any significant change in the computed results. Therefore a sliding window of length $T = 36$ and $L = 3$ sample slopes w_l have been used, obtaining thus a $N = 41$ dimensional feature vector. An edLLM with $K = 5$ is trained on the first $M_0 = 64$ feature vectors. Therefore, the training set consists of the first 99 images of an analyzed sequence (8.25 seconds). Different parameters for the inference have been tested the best results being obtained with $M = 10$ and a delay $d = 2$.

The choice of features discussed here is well suited to detect events on vessel-feature curves, including both those with a typical and an atypical behavior. In general, the choice of features is application dependent and it should be conducted with a thorough understanding of the problem at hand.

In Figure 5.24, several detection results are shown. Displayed are the manually labeled critical point and the measured ones, with both the significance test using non-parametric pdf estimation from the previous section and the edLLM. Mentioned is also the distance in video

frames between: (i) the manual label and the edLLM result, and (ii) the manual label and the significance test result (i.e., the dedicated method).

The ability of the edLLM to make use of the information found in the connections between consecutive components of the feature curve (i.e., the relationship between the level of contrast agent in video frames that are close to one-another in time) leads to the edLLM correctly detecting the event in two more sequences in comparison to the non-parametric estimation-based method. The feature curve corresponding to these sequences is rather different from the typical appearance (as shown in Figure 5.24 (e)). In two sequences the edLLM fails to correctly detect the events, as it also does the significance-based method. Those two sequences are heavily contaminated by noise, such that the vessel-area-related feature extraction fails.

Summary and conclusions. The edLLM is a powerful yet general event detection method that was successfully applied to the specific problem of detecting coronary contrast agent injections. Due to the decision to analyze batches of frames rather than each frame independently, it is not possible to detect the contrast agent immediately, but after a few frames, however, this delay is below the human reaction time.

The edLLM works on a set of features. These have to be determined in advance and represent the link between the practical problem that needs to be solved and the edLLM framework. Depending on the application this set of features can be increased and modified as deemed necessary. The particular set of features used here is suited for chunks of 1D signals, being adapted to the problem of detecting contrast bursts using feature curves.

The delay d included in the algorithm is used to increase the precision of the labeling and is chosen such that the classification is almost instantaneous. Setting $d = 0$ eliminates the delay, but decreases the precision with which an event is detected. However, this does not necessary afflict the detection of events.

Comparing the edLLM with LCCRFs and MEMMs from a theoretical point of view, there are a lot of similarities. In a way, the edLLM combines a LCCRF with a MEMM, as we can obtain an algorithm similar to a LCCRF but also to a MEMM by altering the length of the window of feature vectors M and the delay d in the inference. If M is equal to the whole sequence of feature vectors and $d = 0$, we have a LCCRF, that is, the state sequence can be represented by an undirected graph. If $M = 1$ and $d = 0$, we have a MEMM. The edLLM combines thus the advantages of both these methods.

Chapter 6

Summary

Biometrics and surveillance are two of the main domains of the field of information forensics and security. In the form of person identification and event detection respectively they represent the clear focus of this work. Nevertheless, the type of problems encountered here, are shared by a majority of security applications, thus the scope of this work extends beyond these two domains. Pattern recognition clearly plays a pivotal role over the entire field, irrespective of whether we talk about biometrics, computer and network security, cryptography [164] or data hiding. Conversely, the main approach to pattern recognition is based on statistics.

In this context, we have gone in this work through all design steps of a pattern-recognition system, while keeping in mind that this system is supposed to solve a security-related application.

Therefore, both new methods have been proposed and established ones have been adapted to security-related problems, covering theoretical approaches as well as systems applications. In a building-related analogy, both new bricks were designed and these as well as older designs have been used for new types of houses.

This way, it can be confirmed (yet again) that each problem has its own particular optimal solution, which is a very good thing, because it means that generations of signal-processing specialists that are currently studying will definitely be able to earn a living with this type of knowledge in the future.

The main steps of any pattern recognition system are feature extraction and classification. We differentiate here between feature extraction and raw-feature generation. The raw-features are the result of various measuring processes applied to the investigated reality, while the features are the result of transformations applied to the raw features designed for the purpose of constructing a classification-optimal feature space. With respect to *feature extraction* this work (i) introduced a novel method that is called multiclass Gaussianization, and also (ii) described custom-made feature extraction methods for various security applications. For the *classification* step there are several contributions: (i) the introduction of a new hysteresis classifiers, and also (ii) the linear-predictors mixture, and (iii) the adaptation of the sparse classifier, as well as (iv) the log-linear models – like the conditional random fields and the maximum-entropy Markov models – to biometric identification and event detection applications.

The research presented here in a unitary manner has found its expression in several conference [43, 49, 50, 45, 134, 135, 47, 48, 42] and journal [44, 46] publications as well as a

government-founded project [99]. This demonstrates the interest of both the community but also the society at large on such methods and their applications.

Throughout this entire work, as much attention as possible was paid to giving clear and simple explanations such as to emphasize its didactic side as well. Thus it not only describes cutting-edge research results, but puts them in the proper frame for better understanding and also to ensure a proper focus. Furthermore, care has been given to introducing new concepts starting from simple basics, while putting them in the proper historical context and underlining their connections, such as to make this accessible to a larger public.

6.1 Gaussianization for feature extraction

Considering the ubiquitousness of data-analysis methods that work under the Gaussian assumption, it is only natural to ask how appropriate these are for a certain problem, in light of their underlying assumption. Taking also into account the advantages of working under the Gaussian assumption, there is a high interest to ensure that Gaussian methods are appropriate. There are also methods that are free of the Gaussian assumption, but depending on the application, the cost of using such methods may be higher than the cost of adapting the problem setup to the Gaussian assumption and using Gaussian methods. Within this context, the multiclass Gaussianization described here is a feature extraction method aimed at changing the distribution of the data in the transformed space such that classifiers with explicit or implicit Gaussian assumptions can optimally shatter the transformed feature space. While Gaussianization transforms have been proposed previously, here we have discussed for the first time a Gaussianization transform for classification purposes, i.e., such that each class conditional pdf is Gaussianized.

A majority of the Gaussianization methods are iterative algorithms, closely linked to projection pursuit density estimation [76, 75] that have difficulties to work in a multiclass scenario. Inspired by quality measures for nonparametric density estimators, the multiclass Gaussianization is conducted with the help of an elastic transform that is otherwise successfully applied in the field of image registration. For image registration, the dimensions of the transformed space is usually very small, like two or three, while in a classification setup, the dimension is usually large. The computational complexity of the elastic transform depends on the size of the input space, and to promote the practical deployment of the multiclass Gaussianization, adaptive grids have been introduced, which effectively reduce the computational complexity, such as to be able to apply it with the same effectiveness also in feature spaces with a larger number of dimensions. The adaptive-grid Gaussianization is called “accelerated Gaussianization”.

Nevertheless, the purpose of the multiclass Gaussianization is not to achieve perfect Gaussian classes, but to make the classes more Gaussian than in the original space, such as to ensure satisfactory results with comparatively simple Gaussian classifiers. The experiments conducted show that this purpose was achieved, for both the standard and the fast Gaussianization.

Under these conditions, the multiclass Gaussianization is on the best way from cutting-edge research to practical applications.

6.2 Binary classification with hysteresis methods

Binary classification is an important special case of the classification problem, if we are only to consider the example of the support vector machines [53]. It is only natural that a lot of

effort has been invested into the development and improvement of binary classifiers. In the case of overlapping classes, prior knowledge is instrumental in ensuring that the respective classification problem can be solved with enough accuracy. Here, concepts such as the linear-classifier percentile and the relative hysteresis classifier helped completing the hysteresis binary classification framework – that was introduced for the first time in Reference [36] – a framework that purposefully uses prior knowledge. The resulting hysteresis classifiers offer modalities to introduce prior problem-domain knowledge in the feature-space analysis that is implicitly conducted for the purpose of labeling. The hysteresis classifier can be also seen as a classifier pair, or the smallest possible committee where the classifiers generate and combine their outputs in such a way as to make optimal use of the additional prior information that can not be captured in the form of features.

The concept of hysteresis processing is not new, and hysteresis segmentation [28] is known at a graduate level. However, here this concept was taken and transformed into a true classification framework that gives highly efficient and accurate algorithms for difficult classification problems, as shown in the retinal-vessel segmentation experiments. Vessel segmentation is a topic of high interest not only in the medical community, but also for biometrics, as pointed out in this work as well.

Hysteresis classification is now a mature technology, ready to be deployed into praxis. It has been demonstrated here to full extent on the retinal-vessel segmentation problem, where the the hysteresis vessel segmentation is one of the most accurate methods while being the fastest.

6.3 Sparse classification for security applications

Sparsity as a data-analysis tool is a hot topic in current research. Impressive results have been obtained with compressed sampling [58], but more and more we are discovering or rediscovering sparsity as an universal principle – Occam's razor being just one of its early instantiations. For pattern recognition, similar ideas found their expression in concepts like minimum message length and minimum description length [66], and increasingly in the sparse classifier [187] that enjoys a clear link to the k-NN classifier as shown here as well. Through its k-NN link, the sparse classifier is also a statistical pattern recognition method.

The sparse classifier has a set of characteristics that make it particularly well suited for security applications, like for example the ability to work satisfactory despite the curse of dimensionality, as well as the robustness to noise in the feature vector or even to cropping of the feature vector. These advantages were put to use here in the design of robust biometric systems, working with fingerprints and with retinal vascular networks. The sparse classifier framework also includes simple and efficient outlier-detection methods that were used here successfully to introduce the sparse classifier to the field of event detection.

For biometric applications the application pool of the sparse classifier was simply extended to fingerprints and retinal-vascular networks – together with the necessary feature extraction methods. In the case of event detection, the sparse classifier framework was introduced here for the first time to this application; this being also among the first contributions using sparsity as a data-analysis tool in this context. The validity of such an approach was demonstrated and also the stage was set for further research in this direction.

6.4 Event detection with statistical models

Time series can be modeled very well with statistical models, and as event detection often implies the analysis of a times series, statistical models are often used for event detection. General-purpose methods like hidden Markov models and standard adaptive filters are already well suited for such tasks. Still, the particularities of event detection (in principal those related to the applications where the normal case evolves during time) require new approaches.

In this context, besides introducing a novel linear-predictor mixture model, here for the first time log-linear models like the maximum entropy Markov models and the conditional random fields were adapted for the purpose of event detection in the form of the edLLM. As in the case of the hysteresis classification, these statistical models are able to return better results by processing more information, and by incorporating it in the decision-making process. These methods were tested on an exemplary real-life medical problem, but also on artificially generated data and some toy-examples, to demonstrate their validity. Besides proving the validity of these approaches, directions for further research in these fields were also identified.

Overall, a novel linear predictor mixture was introduced for the analysis of random signals and in particular for event detection; and the log-linear models were adapted, and it has been demonstrated that they can be effectively used for the same purpose of event detection.

Appendix A

Formulae

The Fisher information matrix

The Fisher information matrix $\mathbf{J}$ is defined as the variance of the score

$$\mathbf{J} = E\left\{\mathbf{f}\mathbf{f}^T\right\},$$

with the score $\mathbf{f}$ being

$$\mathbf{f} = \frac{d}{d\mathbf{a}} \ln p(\mathbf{r}|\mathbf{a}),$$

where $\mathbf{a}$ is the vector to be estimated and $\mathbf{r}$ the observation.

LDA for more than two classes

The extension of the Fisher discriminant to N_ω classes is straightforward, for $\mathbf{x} = \mathbf{A}\mathbf{y}$, we have

$$J(\mathbf{A}) = \frac{\mathbf{A}\mathbf{S}_B\mathbf{A}^T}{\mathbf{A}\mathbf{S}_W\mathbf{A}^T}, \tag{A.1}$$

with $\mathbf{S}_B = \sum_{i=1}^{N_\omega} P(\omega_i)(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}})^T$ the between-class scatter matrix and $\mathbf{S}_W = \sum_{i=1}^{N_\omega} P(\omega_i)\boldsymbol{\Sigma}_i$ the within-class scatter matrix, while $\bar{\boldsymbol{\mu}} = \sum_{i=1}^{N_\omega} P(\omega_i)\boldsymbol{\mu}_i$.

Minimizing the trace of the error's correlation matrix

We have the following objective function:

$$\begin{aligned} J(\mathbf{A}) &= \text{tr}\left\{E\left\{[\mathbf{A}\mathbf{r} - \mathbf{a}]\,[\mathbf{A}\mathbf{r} - \mathbf{a}]^H\right\}\right\} \\ &= \text{tr}\left\{E\left\{[\hat{\mathbf{a}}(\mathbf{r}) - \mathbf{a}]\,[\hat{\mathbf{a}}(\mathbf{r}) - \mathbf{a}]^H\right\}\right\} \\ &= \text{tr}\left\{E\left\{\mathbf{e}(\mathbf{r})\mathbf{e}^H(\mathbf{r})\right\}\right\} \\ &= \text{tr}\left\{\mathbf{R}_{ee}\right\}, \end{aligned}$$

for which we have to compute:

$$\tilde{\mathbf{A}} = \arg\min_{\mathbf{A}} J(\mathbf{A})$$

The objective function can be rewritten as

$$\begin{aligned}
J(\mathbf{A}) &= \text{tr}\left\{\mathbf{R}_{aa} - \mathbf{A}\mathbf{R}_{ar} - \mathbf{R}_{ra}\mathbf{A}^H + \mathbf{A}\mathbf{R}_{rr}\mathbf{A}^H\right\} \\
&= \text{tr}\{\mathbf{R}_{aa} - \mathbf{A}\mathbf{R}_{ar} - \mathbf{R}_{ra}\mathbf{A}^H + \mathbf{A}\mathbf{R}_{rr}\mathbf{A}^H + \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\} \\
&= \text{tr}\{\mathbf{R}_{aa} - \mathbf{A}\mathbf{R}_{rr}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{rr}\mathbf{A}^H + \mathbf{A}\mathbf{R}_{rr}\mathbf{A}^H + \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{rr}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar} \\
&\quad - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\} \\
&= \text{tr}\{\mathbf{R}_{aa} + \mathbf{A}\mathbf{R}_{rr}\left(\mathbf{A}^H - \mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\right) - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{rr}\left(\mathbf{A}^H - \mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\right) \\
&\quad - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\} \\
&= \text{tr}\{\mathbf{R}_{aa} + \left(\mathbf{A}\mathbf{R}_{rr} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{rr}\right)\left(\mathbf{A}^H - \mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\right) - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\} \\
&= \text{tr}\{\mathbf{R}_{aa} + \left(\mathbf{A} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\right)\mathbf{R}_{rr}\left(\mathbf{A}^H - \mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\right) - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\}
\end{aligned}$$

As it can be easily seen, the only term that depends on $\mathbf{A}$ in the equation above is:

$$\text{tr}\left\{\left(\mathbf{A} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}\right)\mathbf{R}_{rr}\left(\mathbf{A}^H - \mathbf{R}_{rr}^{-1}\mathbf{R}_{ar}\right)\right\}.$$

This term is a sum of squares and thus always positive. Therefore $J(\mathbf{A})$ is minimized when

$$\mathbf{A} - \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1} = 0$$

which is equivalent to

$$\tilde{\mathbf{A}} = \mathbf{R}_{ra}\mathbf{R}_{rr}^{-1}.$$

Conditional Gaussians

Joint distributions. For the random variable

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix},$$

if $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is Gaussian distributed, then both marginals $p(\mathbf{x})$ and $p(\mathbf{y})$ are also Gaussian. Furthermore, with $\boldsymbol{\Lambda} := \boldsymbol{\Sigma}^{-1}$ and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{xx}} & \boldsymbol{\Sigma}_{\mathbf{xy}} \\ \boldsymbol{\Sigma}_{\mathbf{yx}} & \boldsymbol{\Sigma}_{\mathbf{yy}} \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{\mathbf{xx}} & \boldsymbol{\Lambda}_{\mathbf{xy}} \\ \boldsymbol{\Lambda}_{\mathbf{yx}} & \boldsymbol{\Lambda}_{\mathbf{yy}} \end{pmatrix}$$

we have that

$$p(\mathbf{x}|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}, \boldsymbol{\Lambda}_{\mathbf{xx}}^{-1}) \tag{A.2}$$

with $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{x}} - \boldsymbol{\Lambda}_{\mathbf{xx}}^{-1}\boldsymbol{\Lambda}_{\mathbf{xy}}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})$, and

$$p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{xx}}) \tag{A.3}$$

Bayes theorem and extensions for Gaussian variables. According to the Bayes theorem we have that:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \tag{A.4}$$

Given two dependent Gaussian variables $\mathbf{x}$ and $\mathbf{y}$ related by $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ and the densities

$$p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

and

$$p(\mathbf{y}|\mathbf{x}) \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{K})$$

we need to compute $p(\mathbf{y})$ and $p(\mathbf{x}|\mathbf{y})$. For this purpose, we first compute the joint density $p(\mathbf{x}, \mathbf{y})$ and obtain $p(\mathbf{y})$ by marginalizing it, as:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{K} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T). \tag{A.5}$$

$p(\mathbf{x}|\mathbf{y})$ is then computed with the help of equation (A.4) and making use of equation (A.7) [20] as

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{L}[\mathbf{A}^T\mathbf{K}^{-1}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}], \mathbf{L}), \tag{A.6}$$

with $\mathbf{L} = (\boldsymbol{\Sigma}^{-1} + \mathbf{A}^T\mathbf{K}^{-1}\mathbf{A})^{-1}$.

Matrix properties

Schur complement. Assuming a square matrix $\mathbf{M}$ can be partitioned into

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}$$

then its inverse $\mathbf{M}^{-1}$ can be computed as

$$\mathbf{M}^{-1} = \begin{pmatrix} \mathbf{S} & -\mathbf{S}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{S} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{S}\mathbf{B}\mathbf{D}^{-1} \end{pmatrix} \tag{A.7}$$

where $\mathbf{S} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$ is the Schur complement.

Other matrix identities. The following matrix identities exist:

$$(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1} \tag{A.8}$$

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}. \tag{A.9}$$

Appendix B

Methods

Principal Component Analysis

With $\mathbf{x} = [x_1, \ldots, x_n]^T$, where $\mathbf{x} \in \mathbb{R}^n$ is the realization of a stochastic process, we would like to find the set of parameters $\mathbf{u}_i$ that allow us to approximate $\mathbf{x}$ by $\tilde{\mathbf{x}} = \sum_{i=1}^{m} \alpha_i \mathbf{u}_i$ with $m < n$. $\mathbf{u}_i$ are some orthonormal vectors, and $\alpha_i = \langle \mathbf{x}, \mathbf{u}_i \rangle$ such that

$$\begin{aligned} \boldsymbol{\alpha} &= \left(\mathbf{U}^T\right)^{-1} \mathbf{x} \\ &= \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_n^T \end{bmatrix} \mathbf{x} \\ &= \mathbf{U}\mathbf{x}, \end{aligned} \tag{B.1}$$

with $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T$ and $\mathbf{U}$ an orthonormal transform, such that $\mathbf{x} = \mathbf{U}^T\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$. The error in representing $\mathbf{x}$ by $\tilde{\mathbf{x}}$ is:

$$\varepsilon = \mathbf{x} - \tilde{\mathbf{x}} = \sum_{i=m+1}^{n} \alpha_i \mathbf{u}_i. \tag{B.2}$$

The problem is now to find an orthonormal basis $\mathcal{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, such that the MSE $\mathcal{E} = E\{\|\varepsilon\|^2\}$ is as small as possible. From equation (B.2), the MSE can be written as

$$\mathcal{E} = \sum_{j=m+1}^{n} \mathbf{u}_j^T \mathbf{R}_{xx} \mathbf{u}_j,$$

where $\mathbf{R}_{xx}$ is the correlation matrix of $\mathbf{x}$. Therefore, to find the basis vectors $\mathbf{u}_i$, we need to minimize the composed criterion function

$$\mathcal{E}_l = \sum_{j=m+1}^{n} \mathbf{u}_j^T \mathbf{R}_{xx} \mathbf{u}_j + \sum_{j=m+1}^{n} \lambda_j (1 - \mathbf{u}_j^T \mathbf{u}_j), \tag{B.3}$$

where we have introduced the orthonormality condition over the Lagrange multipliers λ. The solution to the constraint optimization problem (B.3) is given by

$$\mathbf{R}_{xx} \mathbf{u}_j = \lambda_j \mathbf{u}_j, \tag{B.4}$$

which means that $\mathbf{u}_j$ are the eigenvectors of the correlation matrix $\mathbf{R}_{xx}$. For the eigenvalue problem in (B.4) there are always n orthogonal eigenvectors. An orthonormal basis is obtained if the vectors are scaled by their norm. The mean square error is then

$$\mathcal{E} = \sum_{j=m+1}^{n} \lambda_j$$

and is minimized if we select in the representation of $\tilde{\mathbf{x}}$ those eigenvectors $\mathbf{u}$ that correspond to the largest eigenvalues. Hence the sought basis is made of the eigenvectors corresponding to the largest m eigenvalues.

As $\alpha_i = \langle \mathbf{x}, \mathbf{u}_i \rangle = \mathbf{x}^T \mathbf{u}_i = \mathbf{u}_i^T \mathbf{x}$ it follows that

$$\begin{aligned} E\{\alpha_i \alpha_j\} &= \mathbf{u}_i^T E\left\{\mathbf{x}\mathbf{x}^T\right\} \mathbf{u}_j \\ &= \mathbf{u}_i^T \mathbf{R}_{xx} \mathbf{u}_j \\ &= \mathbf{u}_i^T \lambda_j \mathbf{u}_j \\ &= \lambda_j \delta_{ij}. \end{aligned}$$

The correlation matrix for $\boldsymbol{\alpha}$ is then:

$$\boldsymbol{\Lambda} = E\left\{\boldsymbol{\alpha}\boldsymbol{\alpha}^T\right\} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix}. \tag{B.5}$$

Hence, the components of the representation of $\mathbf{x}$ with respect to the found basis are not correlated. They are called principal components.

Whitening

The purpose of the Whitening transform is not only to decorrelate the components $\boldsymbol{\alpha}$, as in the case of the PCA but also to ensure that their variances are equal. By this transformation a colored random process is transformed into a white random process. Whitening will be defined next as a further processing step after PCA.

We look for a linear transform $\mathbf{T}$ such that for $\tilde{\mathbf{n}} = \mathbf{T}\mathbf{n}$, the covariance matrix of $\tilde{\mathbf{n}}$ is:

$$\mathbf{K}_{\tilde{n}\tilde{n}} = \mathbf{T}\mathbf{K}_{nn}\mathbf{T}^T = \mathbf{I}$$

It is clear that $\mathbf{T}$ is not unique, as multiplication of $\mathbf{T}$ by an orthonormal matrix has no influence on the relation above.

Applying the PCA, to the random vector $\mathbf{n}$, the covariance matrix in the transformed space will be:

$$\mathbf{U}\mathbf{K}_{nn}\mathbf{U}^T = \boldsymbol{\Lambda}.$$

It follows that

$$\mathbf{K}_{nn} = \mathbf{U}^T \boldsymbol{\Lambda} \mathbf{U},$$

with $\mathbf{U}$ the PCA transform matrix. At this stage we observe that $\boldsymbol{\Lambda}$ can be written as $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$, with:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_N} \end{bmatrix}.$$

Therefore we have that

$$\mathbf{K}_{nn} = \mathbf{U}^T \mathbf{\Sigma}^T \mathbf{\Sigma} \mathbf{U}$$

and thus

$$\mathbf{\Sigma}^{-1} \mathbf{U} \mathbf{K}_{nn} \mathbf{U}^T \mathbf{\Sigma}^{T-1} = \mathbf{I}$$

and the transfomation $\mathbf{T} = \mathbf{\Sigma}^{-1}\mathbf{U}$ is a whitening transform.

The orthogonality principle

Generally, we obtain a minimum error estimate $\hat{\mathbf{x}}$ of the signal $\mathbf{x}$ when the error vector $\hat{\mathbf{x}} - \mathbf{x}$ is perpendicular on the estimate. This is shown in Figure B.1.

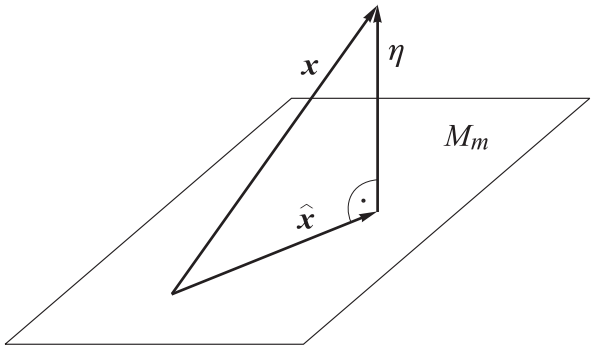


Figure B.1: Illustration of the orthogonality principle.

For random signals, this means that a minimum mean error estimate of the parameter vector $\mathbf{a}$ verifies the orthogonality relation:

$$\begin{aligned} E\left\{[\hat{\mathbf{a}} - \mathbf{a}]\hat{\mathbf{a}}^H\right\} &= E\left\{\hat{\mathbf{a}}\hat{\mathbf{a}}^H\right\} - E\left\{\mathbf{a}\hat{\mathbf{a}}^H\right\} \\ &= \mathbf{A}\mathbf{R}_{rr}\mathbf{A}^H - \mathbf{R}_{ar}\mathbf{A}^H \\ &= \left[\mathbf{A}\mathbf{R}_{rr} - \mathbf{R}_{ar}\right]\mathbf{A}^H \\ &= 0 \end{aligned}$$

where we have used $\hat{\mathbf{a}} = \mathbf{A}\mathbf{r}$. It follows that:

$$\mathbf{R}_{ar} = \mathbf{A}\mathbf{R}_{rr}. \tag{B.6}$$

The same relationship can be also derived from (2.22) and leads us to concluding that the MMSE estimate verifies the orthogonality relation.

Equation (B.6) can be expressed equivalently as

$$E\left\{\mathbf{a}\mathbf{r}^H\right\} = \mathbf{A}E\left\{\mathbf{r}\mathbf{r}^H\right\}$$

and using again $\hat{\mathbf{a}} = \mathbf{A}\mathbf{r}$ we obtain

$$E\left\{\mathbf{a}\mathbf{r}^H\right\} = E\left\{\hat{\mathbf{a}}\mathbf{r}^H\right\}$$

which means that

$$E\left\{\mathbf{a}\mathbf{r}^H\right\} - E\left\{\hat{\mathbf{a}}\mathbf{r}^H\right\} = 0$$

and thus

$$E\left\{[\hat{\mathbf{a}} - \mathbf{a}]\mathbf{r}^H\right\} = 0 \tag{B.7}$$

The relationship in (B.7) describes **the orthogonality principle** . By the orthogonality principle, we obtain a MMSE estimator when the estimation error is not correlated with the observation $\mathbf{r}$ that were used to compute the estimate $\hat{\mathbf{a}}(\mathbf{r})$.

Derivation of the Wiener-Hopf equations. In the case of the Wiener filter, the solution to the minimization problem $E\left\{|e(n)|^2\right\} \rightarrow \min$ must follow the orthogonality principle. In our current setup – where we denote the observations by $x(n)$, the true signal by $d(n)$ and the estimate by $y(n) = \sum_{i=0}^{p-1} h(i)x(n-i)$ – we have that:

$$E\left\{\left[d(n) - \sum_{i=0}^{p-1} h(i)x(n-i)\right] x^*(n-j)\right\} = 0, \quad j = 0, 1, \ldots, p-1. \tag{B.8}$$

Assuming a stationary process, we obtain from (B.8) the *Wiener-Hopf Equations*

$$\sum_{i=0}^{p-1} h(i) r_{xx}(j-i) = r_{xd}(j), \quad j = 0, 1, \ldots, p-1,$$

with

$$r_{xx}(m) = E\left\{x^*(n)x(n+m)\right\},$$
$$r_{xd}(m) = E\left\{x^*(n)d(n+m)\right\}.$$

Significance testing for classifier comparison

In its initial formulation, a significance test is conducted to get an idea of how likely it is that the desired outcome observed at the output of a system is due to chance. If this is unlikely, we conclude that the system is significantly better than chance at producing this desired outcome. Significance testing is based on the binomial distribution, however, the probability of success should be computed from the available data about the investigated system.

A significance test has four major steps:

- Build the null hypothesis $\mathbf{H}_0 \rightarrow$ The system is no better than chance.

  As a consequence of intuitively imposing completeness, you get as well the alternative hypothesis $\mathbf{H}_1 \rightarrow$ The system is better than chance.

- Establish P_{chance}, the probability that the desired system outcome is due to chance (i.e, the probability of succeeding by chance).

- Set the significance level α that is related to how confident you would like to get about the result of the test.

- Find out if the null hypothesis can be rejected. For this purpose, we make use of the random variable X, representing the number of times the desired outcome has been observed (i.e., the number of successes) in N trials. We obtain the following rule:

  Reject $\mathbf{H}_0$ if we observe an event ξ and the probability of this event under the null hypothesis is very small. With $\xi = X > C$, we would reject $\mathbf{H}_0$ if

  $$P_{\mathbf{H}_0}(X > C) \leq \alpha,$$

  with α very small. $P_{\mathbf{H}_0}(X > C) = 1 - F(C, N, P_{chance})$ is probability of observing $X > C$ under $\mathbf{H}_0$ and it represents thus the Type I error probability, i.e., the probability of falsely rejecting the null hypothesis. Furthermore, we have

  $$F(C, N, p) = P(X \leq C) = \sum_{i=0}^{X} \left(\begin{array}{c} N \\ i \end{array} \right) p^i (1-p)^{n-i}, X \in \mathbb{N}$$

  the binomial cumulative distribution function, with $C \in \mathbb{N}$ the threshold on X corresponding to α.

If $\mathbf{H}_0$ is rejected, then this does not automatically mans accepting $\mathbf{H}_1$. All we can say is that available data does not support accepting $\mathbf{H}_0$ at the given significance level. $\mathbf{H}_1$ is accepted only by way of consequence.

Example 1 25 data points need to be classified into four classes. Find out if a classification algorithm is better than chance for this task.

With four classes, we have that $P_{chance} = \frac{1}{4}$. Setting $\alpha = 0.01$, we compute C as the solution to the equation $1 - F(C, 25, \frac{1}{4}) = 0.01$. We obtain in this case $C = 11$. This means that the probability of our classifier correctly deciding by chance 12 or more times in 25 trials is 0.01. If this nevertheless happens, we conclude that it is unlikely it happened by chance because of the small probability of this event under the "'chance"' assumption. Therefore, if our classifier decides 12 or more times correctly on the 25 data points it is significantly better than chance at a significance level of $\alpha = 0.01$

Example 2 25 data points need to be classified into four classes. We have two classifiers A and B at our disposal. Find out if the two classifiers are significantly different.

We construct the null hypothesis that "A is no better than B". With the probability of a correct decision by B being $P_B = \frac{1}{2}$, and a significance level $\alpha = 0.01$, the solution of the equation $1 - F(C, 25, \frac{1}{2}) = 0.01$ is $C = 18$. This mens that if A decides correctly 19 or more times, it is significantly better than B at a significance level of $\alpha = 0.01$.

Iterative solutions of a system of equations

We are looking for the solution to the system of equations

$$\mathbf{Ax} = \mathbf{b}$$

We would like to search for the solution in an iterative manner, such as to be able to step-wise improve an initial solution until we reach the true solution.

The Jacobi method. We observe that the matrix $\mathbf{A}$ can be written as $\mathbf{A} = \mathbf{D} + \mathbf{R}$, i.e., the sum of a diagonal matrix $\mathbf{D}$ and a rest matrix $\mathbf{R}$. Then it follows that

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} && \Leftrightarrow \\ (\mathbf{D} + \mathbf{R})\,\mathbf{x} &= \mathbf{b} && \Leftrightarrow \\ \mathbf{D}\mathbf{x} &= \mathbf{b} - \mathbf{R}\mathbf{x}. \end{aligned}$$

which leads to the iterative solution $\mathbf{x}_t = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x}_{t-1})$. The iterative algorithm converges always if the matrix $\mathbf{A}$ is diagonally dominant, meaning that all entries in a row are smaller than the diagonal element of the row. Otherwise, convergence is not guaranteed.

The least-squares method. In this case, we build the quadratic form $F(\mathbf{x}) = (\mathbf{A}\mathbf{x} - \mathbf{b})^2$ and look for its minimum to obtain the desired solution. We obtain two types of methods: (i) the direct method, and (ii) the iterative method.

(i) Direct algebraic computations involve computing the derivative of the quadratic form as $\nabla F(\mathbf{x}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$ and solving over $\mathbf{x}$ the equation obtained by setting this derivative to zero. In this case the solution is simply $\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$.

(ii) We search for the optimum of the quadratic form in an iterative manner, starting at some initial location on the surface $F(\mathbf{x})$ and going in the direction of the gradient within the gradient descent framework or in a direction conjugate to the gradient[1] within the framework of the CG method [166]. When using the gradient descent, the iterative solution is $\mathbf{x}_t = \mathbf{x}_{t-1} - \gamma\nabla F(\mathbf{x})$. When using the CG, the matrix $\mathbf{A}$ needs to be symmetric and positive semidefinite, otherwise we should solve $\mathbf{A}^T\mathbf{A} = \mathbf{A}^T\mathbf{b}$ instead.

[1] Two vectors $\mathbf{m}$ and $\mathbf{n}$ are conjugate with respect to a matrix $\mathbf{K}$ if $\mathbf{m}^T\mathbf{K}\mathbf{n} = 0$.

Bibliography

[1] T. Aach and A. P. Condurache. Transformation of adaptive thresholds by significance invariance for change detection. In *Proceedings of the 13'th Workshop on Statistical Signal Processing (SSP)*, pages 637 – 642, Bordeaux, France, July 17–20 2005. IEEE.

[2] T. Aach and A. Kaup. Bayesian algorithms for adaptive change detection in image sequences using Markov random fields. *Elsevier Signal Processing: Image Communication*, 7:147 – 160, 1995.

[3] T. Aach, I. Stuke, C. Mota, and E. Barth. Estimation of multiple local orientations in image signals. In *Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages III 553–556, Montreal, Canada, May 17–21 2004. IEEE.

[4] A. Adam and E. Rivlin. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):555–560, 2008.

[5] A. P. Adamis, J. W. Miller, M. T. Bernal, D. J. D'Amico, J. Folkman, T. K. Yeo, and K. T. Yeo. Increased vascular endothelial growth factor levels in the vitreous of eyes with proliferative diabetic retinopathy. *American Journal of Ophthalmology*, 118(4):445–450, 1994.

[6] K. Arbter, W. E. Snyder, H. Burkhardt, and G. Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:640 – 647, 1990.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174 – 188, 2002.

[8] W. Barkhoda, F. A. Tab, and M. D. Amiri. Rotation invariant retina identification based on the sketch of vessels using angular partitioning. In *Proceedings of International Multiconference on Computer Science and Information Technology (IMCSIT)*, pages 3–6, Margowo, Poland, October 12–14 2009. IEEE.

[9] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search with applications to pattern recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, MN, US, 17-22 June 2007. IEEE.

[10] M. Basseeville and Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*, page 35ff. Prentice-Hall, 1993.

[11] A. M. Bazen and S. H. Gerez. Systematic methods for the computation of the directional fields and singular points of fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:905–919, July 2002.

[12] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1000–1006, 1997.

[13] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Comput.*, 7(6):1129–1159, 1995.

[14] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 21(6):1601 – 1621, 2009.

[15] Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. *Large-scale kernel machines*, 34:1 – 41, 2007.

[16] A. L. Berger, S. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[17] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192 – 236, 1974.

[18] J. C. Bezdek, J.M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal. Will the real iris data please stand up? *IEEE Transactions on Fuzzy Systems*, 7(3):368–369, 1999.

[19] P. Billingsley. *Probability and measure*. John Wiley & Sons, 1986.

[20] C. M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer, 2009.

[21] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[22] S. Bouix, M. Martin Fernandez, L. Ungar, M. Nakamura, M.S. Koo, R.W. McCarley, and M.E. Shenton. On evaluating brain tissue classifiers without a ground truth. *NeuroImage*, 36(4):1207–1224, 07 2007.

[23] M. A. Burock and A. M. Dale. Estimation and detection of event-related fMRI signals with temporally correlated noise: A statistically efficient and unbiased approach. *Human Brain Mapping*, 11 Issue 4:249–260, 2000.

[24] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis?: Recovering low-rank matrices from sparse errors. In *Proc. IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 201–204, Jerusalem, Israel, October 4 – 7 2010.

[25] E. J. Candès, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[26] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[27] Emmanuel Candés. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[28] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[29] R. Cappelli, M. Ferrara, and D. Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2128–2141, 2010.

[30] R. Cappelli, D. Maio, D. Maltoni, J.L. Wayman, and A.K. Jain. Performance evaluation of fingerprint verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:3–18, 2006.

[31] J. S. Cardoso and L. Corte-Real. Toward a generic evaluation of image segmentation. *IEEE Transactions on Image Processing*, 14(11):1773–17872, 2005.

[32] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[33] S. Chen and R. Gopinath. Gaussianization. In *Proceedings of Neural Information Processing Systems (NIPS)*, Denver, USA, 2000.

[34] Z. Chen and S. Molloi. Multiresolution vessel tracking in angiographic images using valley courses. *Optical Engineering*, 42:1673–1682, 2003.

[35] P. Comon. Independent component analysis: A new concept? *Signal Process.*, 36(3):287 – 314, 1994.

[36] A. P. Condurache. *Cardiovascular biomedical image analysis: methods and applications.* GCA-Verlag, Waabs, 2008.

[37] A. P. Condurache and T. Aach. Vessel segmentation in angiograms using hysteresis thresholding. In *Proceedings of Machine Vision Applications (MVA)*, pages 269–272, Tsukuba, Japan, May 16–18 2005.

[38] A. P. Condurache and T. Aach. Vessel segmentation in 2D-projection images using a supervised linear hysteresis classifier. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, volume 1, pages 239–243, Hong Kong, China, August 20–24 2006. IEEE.

[39] A. P. Condurache, T. Aach, S. Grzybowski, and H.-G. Machens. Imaging and analysis of angiogenesis for skin transplantation by microangiography. In *Proceedings of International Conference on Image Processing (ICIP)*, pages II/1250 – II/1253 (also on CD–ROM: ISBN 07803 9135–7), Genoa, Italy, September 11–14 2005. IEEE.

[40] A. P. Condurache, T. Aach, S. Grzybowski, and H.-G. Machens. Vessel segmentation and analysis in laboratory skin transplant micro-angiogram. In *Proceedings of the 18'th IEEE Symposium on Computer Based Medical Systems (CBMS)*, pages 21–26, Dublin, Ireland, June 23–24 2005. IEEE.

[41] A. P. Condurache, T. Aach, S. Grzybowski, and H.-G. Machens. Vessel segmentation for angiographic enhancement and analysis. In *Proceedings of Bildvearabeitung für die Medizin (BVM)*, pages 173–177, Heidelberg, Germany, March 13–15 2005. Springer.

[42] A. P. Condurache, J. Kotzerke, and A. Mertins. Robust retina-based person authentication using the sparse classifier. In *Proceedings of European Signal Processing Conference (EUSIPCO)*, pages 1514–1518, Bucharest, Romania, 27-31 August 2012. IEEE.

[43] A. P. Condurache and A. Mertins. A point-event detection algorithm for the analysis of contrast bolus in fluoroscopic images of the coronary arteries. In *Proceedings of European Signal Processing Conference (EUSIPCO)*, pages 2337–2341, Glasgow, August, 24–28 2009.

[44] A. P. Condurache and A. Mertins. Elastic-transform based multiclass gaussianization. *IEEE Signal Processing Letters*, 18(8):482–485, 2011.

[45] A. P. Condurache and A. Mertins. Robust core-point-roi based fingerprint identification using a sparse classifier. In *Proc. of Digital Image Computing: Techniques and Applications (DICTA)*, pages 487–493, Noosa, Queensland, Australia, 6-8 December 2011. IEEE.

[46] A. P. Condurache and A. Mertins. Segmentation of retinal vessels with a hysteresis binary-classification paradigm. *Computerized Medical Imaging and Graphics*, 36(4):325–335, June 2012.

[47] A. P. Condurache and A. Mertins. Sparse representations and invariant sequence-feature extraction for event detection. In *Proceedings of Computer Vision Theory and Applications (VISAPP)*, Rome, Italy, 24-26 February 2012. Springer.

[48] A. P. Condurache and A. Mertins. Fast nonlinear gaussianization for feature extraction. In *Submitted to the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, Barcelona, Spain, 15-18 February 2013. Springer.

[49] A. P. Condurache, A. Mertins, and T. Aach. Supervised, hysteresis-based segmentation of retinal images using the linear-classifier percentile. In *Medizinische Bildverarbeitung und Mustererkennung (GI Jahrestagung)*, volume P154 of *Lecture Notes in Informatics*, pages 1285–1293, Lübeck, October 2009. GI.

[50] A. P. Condurache, F. Müller, and A. Mertins. An LDA-based relative hysteresis classifier with application to segmentation of retinal vessels. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, pages 4202–4205, Istanbul, Turkey, 2010. IEEE.

[51] Y. Cong, J. Yuan, and J. Liu;. Sparse reconstruction cost for abnormal event detection. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA, 20 – 25 June 2011.

[52] D. J. Connor, B. G. Haskell, and F. W. Mounts. A frame-to-frame picturephone coder for signals containing differential quantizing noise. *The Bell System Technical Journal*, 52(1):35–51, 1973.

[53] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines.* Cambridge University Press, 2000.

[54] A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation of sparse PCA using semidefinite programming. *SIAM Review*, 49(3), 2007.

[55] Michael Davies and Rémi Gribonval. Restricted isometry constants where ℓ^p sparse recovery can fail for $0 < p \leq 1$. *IEEE Transactions on Information Theory*, 55(5):2203–2214, 2009.

[56] T. M. Dias, R. Attux, J. M. Romano, and R. Suyama. Blind source separation of post-nonlinear mixtures using evolutionary computation and gaussianization. In *Proceedings of Independent Component Analysis (ICA)*, pages 235–242. Springer, 2009.

[57] D. Donoho. For most large underdetermined systems of equations, the minimal ℓ_1 -norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[58] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289 – 1306, 2006.

[59] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1-minimization. *Proceedings of the National Academy of Science of the US*, 100(5):2197–2202, 2003.

[60] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

[61] D. L. Donoho and J. Tanner. Neighborliness of randomly projected simplices in high dimensions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(27):9452–9457, 2005.

[62] D. L. Donoho and J. Tanner. Counting faces of randomly projected polytopes when the projection radically lowers dimension. *Journal of the American Mathematical Society*, 22:1–53, 2009.

[63] D. L. Donoho and Y. Tsaig. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. Technical Report 18, Stanford University Department of Statistics, 2006.

[64] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice (Statistics for Engineering and Information Science)*. Springer, 2001.

[65] E. R. Dougherty. *An introduction to morphological image processing.* SPIE Optical Engineering Press, 1992.

[66] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* J. Willey & Sons Inc., New York, 2000.

[67] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[68] V. Epanechnikov. Nonparametric estimation of a multivariate probability density. *Theory of Probability and its Applications*, 14:153–158, 1969.

[69] H. Farzin, H. A. Moghaddam, and M.-S. Moin. A novel retinal identification system. *EURASIP Journal on Advances in Signal Processing*, 2008:1–10, 2008.

[70] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[71] A. Fischer and C. Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47:25 – 39, 2014.

[72] B. Fischer and J. Modersitzki. Fast inversion of matrices arising in image processing. *Numerical Algorithms*, 22:1–11, 1999.

[73] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *ACM Communications*, 24:381–395, June 1981.

[74] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998.

[75] J. H. Friedman, W. Stuetzle, and A. Schroeder. Projection pursuit density estimation. *Journal of the American Statistical Association*, 79:599–608, 1984.

[76] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23:881–890, 1974.

[77] M. Frydenberg. The chain graph markov property. *Scandinavian Journal of Statistics*, 17(4):333 – 353, 1990.

[78] K. Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, 1990.

[79] K. Fukuta, T. Nakagawa, Y. Hayashi, Y. Hatanaka, T. Hara, and H. Fujita. Personal identification based on blood vessels of retinal fundus images. In *Proceedings of Medical Imaging: Image Processing (MI)*, pages 69141V–69141V–9, San Diego, US, February 16–21 2008. SPIE.

[80] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribtions, and the Bayesian restauration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[81] R. C. Gonzales and R. E. Woods. *Digital image processing. Third edition.* Pearson Education, 2008.

[82] R. A. Gopinath. Maximum likelihood modeling with gaussian distributions for classification. In *Proceedings of International Conference on Acousitc Speech and Signal Processing (ICASSP)*, pages 661–664, Seattle, U.S.A., 1998.

[83] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[84] T. Gramss. Word recognition with the feature finding neural network (FFNN). In *Proc. IEEE Workshop Neural Networks for Signal Processing*, pages 289–298, Princeton, NJ, USA, Oct. 1991.

[85] K. Guo, P. Ishwar P., and J. Konrad. Action recognition using sparse representation on covariance manifolds of optical flow. In *Proceedings of Advanced Video and Signal Based Surveillance (AVSS)*, pages 188–195, 2010.

[86] R. M. Gupta and Y. Chen. *Theory and Use of the EM Algorithm (Foundations and Trends in Signal Processing)*. Now Publishers Inc, 2011.

[87] R. M. Gupta and S. Sarawagi. Conditional Random Fields. Technical report, KReSIT, IIT Bombay, 2005.

[88] F. Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley and Sons, Inc., 2000.

[89] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3:1157–1182, 2003.

[90] W. Hackbusch. *Iterative solution of large sparse systems of equations*. Springer-Verlag (New York), 1993.

[91] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2008.

[92] S. Haykin. *Adaptive filter theory*. Prentice Hall, fourth edition, 2002.

[93] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of Computer Vision and Pattern Recognition (CVPR (2))*, pages 695 – 702, Washington, DC, US, 27 June – 2 July 2004.

[94] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409 – 436, 1952.

[95] R. B. Hill. Apparatus and method for identifying individuals through their retinal vascular patterns. Technical report, US patent - nr. 4109237, 1978.

[96] R. R. Hill. *Retina identification*. Biometrics: Personal Identification in Networked Society. Kluwer Academic Press, Boston, 1999.

[97] R. Hogg, A. Craig, and J. McKean. *Introduction to mathematical statistics.* Prentice Hall, 6 edition, 2004.

[98] A. Hoover, V. Kouznetzova, and et al. Locating blood vessels in retinal images by picewise threshold probing of a matched filter response. *IEEE Transactions on Medical Imaging*, 19(3):203–210, 2000.

[99] http://www.isip.uni luebeck.de/?id=636. Event detection for ambient assisted living (SmartAssist).

[100] http://www.varpa.es/varia.html. VARPA Retinal images for authentication (VARIA).

[101] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley and Sons, 2001.

[102] A. K. Jain and P. W. Duin. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[103] A. K. Jain, P. J. Flynn, and A. Ross, editors. *Handbook of biometrics*. Springer, 2007.

[104] A.K. Jain, J. Feng, and K. Nandakumar. Fingerprint matching. *IEEE Computer*, pages 36–44, February 2010.

[105] X. Jiang and D. Mojon. Adaptive local thresholding by verification based multithreshold probing with application to vessel detection in retinal images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):131–137, 2003.

[106] F. Jiao, S. Wang, C.-H. Lee, R. Greinerl, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 209–216, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[107] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183 – 233, 1999.

[108] S. M. Raiyan Kabir, R. Rahman, M. Habib, and M. R. Khan. Person identification by retina patern matching. In *Proceedings of International Conference on Electrical and Computer Engineering (ICECE)*, pages 522–525, Dhaka, Bangladesh, December 28–30 2004.

[109] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Transactions of the ASME. Series D, Journal of Basic Engineering*, 83:95–107, 1961.

[110] K Karu and A. K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[111] P. Kidmose. Adaptive filtering for non-gaussian processes. *Proceedings of International Conference on Acousitc Speech and Signal Processing (ICASSP)*, pages 424–427, 2000.

[112] C. Kirbas and F. K. H. Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys*, 36(2):81–121, 2004.

[113] D. C. Klonoff and D. M. Schwartz. An economic analysis of interventions for diabetes. *Diabetes Care*, 23(3):390–404, 2000.

[114] K. Kroschel. *Statistische Informationstechnik*. Springer, 2004.

[115] K. Labusch, E. Barth, and T. Martinetz. Robust and fast learning of sparse codes with stochastic gradient descent. *IEEE Transactions on Selected Topics in Signal Processing*, 5(5):1048 – 1060, 2011.

[116] J. D. Lafferty, A. Mccallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of Intl. Conf. on Machine Learning (ICML)*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[117] B. S. Y. Lam, Y. Gao, and A. W.-C. Liew. General retinal vessel segmentation using regularization-based multiconcavity modeling. *IEEE Transactions on Medical Imaging*, 29(7):1369–1381, 2010.

[118] V. Laparra, G. Camps-Valls, and J. Malo. Iterative gaussianization: From ICA to random rotations. *IEEE Transactions on Neural Networks*, 22:534–549, 2011.

[119] L. Latha, M. Pabitha, and S. Thangasamy. A novel method for person authentication using retinal images. In *Proceedings of ICICT-2010*, pages 1–6, Tamil Nadu, India, February 12–13 2010.

[120] D. S. Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:827–832, May 2005.

[121] S.Z. Li, K.L. Chan, and C. Wang. Performance evaluation of the nearest feature line method in image classification and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1335–1349, 2000.

[122] F. Liu, Q. Zhao, L. Zhang, and D. Zhang. Fingerprint pore matching based on sparse representations. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, pages 1630–1633, Istanbul, Turkey, August 2010.

[123] M. Liu, X. Jiang, and A.C. Kot. Fingerprint reference-point detection. *EURASIP Journal on Advances in Signal Processing*, 4:498–509, 2005.

[124] M. Loog and M. P. W. Duin. Liniar dimensionality reduction via a hetereoscedastic extension of lda: the chernoff criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(26):732–739, 2004.

[125] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, Kerkyra , Greece, July 27–31 1999. IEEE.

[126] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, November 2004.

[127] S. Lyu and E. P. Simoncelli. Nonlinear extraction of independent components of natural images using radial gaussianization. *Neur. Comput.*, 21:1485–1519, June 2009.

[128] S. A. Willsky M. D. Malioutov, K. J. Johnson. Walk-sums and belief propagation in gaussian graphical models. *Journal of Machine Learning Research*, 7:2031–2064, 2006.

[129] D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain. Fvc2000: Fingerprint verification competition. *IEEE Transactions on PAMI*, 24(3):402–412, 2002.

[130] D. Maltoni, D. Maio, A.K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2009.

[131] D. Marin, A. Aquino, M.E. Gegundez-Arias, and J.M. Bravo. A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features. *IEEE Transactions on Medical Imaging*, 30(1):146–158, 2011.

[132] C. Marino, M. G. Penedo, M. Penas, M. J. Carreira, and F. Gonzalez. Personal authentication using digital retinal images. *Pattern Analysis and Applications*, 9(1):21–33, 2006.

[133] M. Elena Martinez-Perez, Alun D. Hughes, Simon A. Thom, Anil A. Bharath, and Kim H. Parker. Segmentation of blood vessels from red-free and fluorescein retinal images. *Medical Image Analysis*, 11(1):47 – 61, 2007.

[134] D. Matern, A. P. Condurache, and A. Mertins. Linear prediction based mixture models for event detection in video sequences. In *Proceedins of the Iberian Conference, on Pattern Recognition and Image Analysis (IbPRIA)*, pages 25 – 32, Gran Canaria, Spain, 2011.

[135] D. Matern, A. P. Condurache, and A. Mertins. Event detection using log-linear models for coronary contrast agent injections. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 172 – 179, Vilamoura - Algarve, Portugal, 2012.

[136] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Confernece on Machine Learning (ICML)*, pages 591–598, 2000.

[137] G. G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):873–889, 2001.

[138] A. M. Mendonca and A. Campilho. Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction. *IEEE Transactions on Medical Imaging*, 25(9):1200–1213, 2006.

[139] A. Mertins. *Signaltheorie: Grundlagen der Signalbeschreibung, Filterbänke, Wavelets, Zeit-Frequenz-Analyse, Parameter- und Signalschätzung*. Vieweg+Teubner, 2. Auflage, 2010.

[140] I. Mezghani-Marrakchi, G. Mahé, M. Jaïdane-Saïdane, S. Djaziri-Larbi, and M. Turki-Hadj-Allouane. "Gaussianization" method for identification of memoryless nonlinear audio systems. In *Proceedings of European Signal Processing Conference (EUSIPCO)*, pages 2316 – 2320, 2007.

[141] J. Modersitzki. *Numerical methods for image registration*. Oxford university press, 2004.

[142] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[143] A. Niemistö, V. Dunmire, O Yli-Harja, W. Zahng, and I. Shmulevich. Robust quantification of in vitro angiogenesis through image analysis. *IEEE Transactions on Medical Imaging*, 24(4):549–553, 2005.

[144] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607 – 609, 1996.

[145] M. Ortega and M. G. Penedo. Retinal vessel tree as biometric pattern. *Biometrics*, http://www.intechopen.com/articles/show/title/retinal-vessel-tree-as-biometric-pattern:113–138, 2011.

[146] M. Ortega, M. G. Penedo, J. Rouco, N. Barreira, and M. J. Carreira. Retinal verification using a feature points based biometric pattern. *EURASIP Journal on Advances in Signal Processing*, 2009(2):1–13, 2009.

[147] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-9(1):62–66, 1979.

[148] D. L. Pham, C. Xu, and J. L. Prince. Survey of current methods in medical image segmentation. *Anual Review of Biomedical Engineering*, 2:315–337, 2000.

[149] M. Piccardi. Background subtraction techniques: a review. In *Proceedings of the Conference on Systems, Man and Cybernetics (CSMC)*, volume 4, pages 3099–3104, 2004.

[150] R. Poli and G. Valli. An algorithm for real time vessel enhancement and detection. *Computer Methods and Programs in Biomedicine*, 52(1):1–22, 1997.

[151] P. Pudil, J. Novovicova, and J. Killtler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[152] A. Quattoni, S. Wang, L. P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852, 2007.

[153] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE Proceedings*, 77(2):257–286, 1989.

[154] S. T. Rachev. The Monge-Kantorovich mass transference problem and its stochastic applications. *SIAM Theory of Probability and its Applications*, 29(4):647 – 676, 1985.

[155] E. Ricci and R. Perfetti. Retinal blood vessel segmentation using line operators and support vector classification. *IEEE Transactions on Medical Imaging*, 26(10):1357–1365, 2007.

[156] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[157] G. Saon, S. Dharanipragada, and D. Povey. Feature space gaussianization. In *Proceedings of International Conference on Acousitc Speech and Signal Processing (ICASSP)*, pages I – 329–332, 2004.

[158] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen. Maximum likelihood discriminant feature spaces. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, 5 – 9 June 2000.

[159] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Intl. conf. on Artificial Neural Networks (ANN)*, 1997.

[160] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the International Conference on Computer Vision (ICPR)*, volume 3, pages 32 – 36, Cambridge, UK, 23 – 26 August 2004.

[161] H. Schulz-Mirbach. On the existence of complete invariant feature spaces in pattern recognition. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, volume 2, pages 178–182, The Hague, 1992.

[162] H. Schulz-Mirbach. Algorithms for the construction of invariant features. In *DAGM Symposium*, volume 5, pages 324–332, Wien, 1994.

[163] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.

[164] S. O. Sharif, L. I. Kuncheva, and S. P. Mansoor. Classifying encryption algorithms using pattern recognition techniques. In *Proc. Int. Conf. Information Theory and Information Security (ICITIS)*, pages 1168–1172, 2010.

[165] Y. Sheikh, M. Sheikh, and M. Shah. Exploring the space of a human action. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 144–149, 2005.

[166] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[167] B.W. Silverman. *Density Estimation*. London: Chapman and Hall, 1986.

[168] C. Simon and I. Goldstein. A new scientific method of identification. *New York State Journal of Medicine*, 35:901–906, 1935.

[169] J. V. B. Soares, J. J. G. Leandro, R. M. Cesar Jr., H. F. Jelinek, and M. J. Cree. Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification. *IEEE Transactions on Medical Imaging*, 25(9):1214–1222, 2006.

[170] M. Sofka and C.V. Stewart. Retinal vessel centerline extraction using multiscale matched filters, confidence and edge measures. *IEEE Transactions on Medical Imaging*, 25(12):1531–1546, 2006.

[171] M. Sonka and J. M. Fitzpatrick. *Handbook of Medical Imaging.* SPIE Press, 2000.

[172] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004.

[173] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[174] C. W. Therrien. *Decision, estimation and classification.* J.Wiley & Sons Inc., New York, 1989.

[175] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[176] H. L. Van Trees. *Detection, Estimation, and Modulation Theory, Band 1*. John Wiley & Sons, 2001.

[177] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[178] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *EEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1473–1488, 2008.

[179] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[180] M. A. Vicente, P. O. Hoyer, and A. Hyvärinen. Equivalence of some common linear feature extraction techniques for appearance-based object recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):896–900, 2007.

[181] G. Walker. On periodicity in series of related terms. In *Proceedings of the Royal Society of London*, volume 131, pages 518–532, 1931.

[182] H. M. Wallach. Conditional Random Fields: An introduction. CIS Technical Report MS-CIS-04-21, University of Pensilvania, 2004.

[183] M. Wand and M. Jones. *Kernel Smoothing.* Chapman and Hall, 1995.

[184] S. Watanabe. *Knowing and Guessing: A Quantitative Study of Inference and Information.* John Wiley & Sons Inc., 1969.

[185] P. Wesseling. *An introduction to multigrid methods*. John Wiley & Sons, 1992.

[186] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *Proc. of Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, US, December 7 – 10 2009.

[187] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[188] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 379–385, Champaign, IL, 1992.

[189] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 30(5):893–908, 2008.

[190] G. Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. In *Philosophical Transactions of the Royal Society of London*, volume 226, pages 267–298, 1927.

[191] J. Zhang, S.Z. Li, and J. Wang. Nearest manifold approach for face recognition. In *Proceedings of Automated Face and Gesture Recognition (AFGR)*, pages 223–228, Seoul, Korea, 2004.

[192] Y. J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346, 1995.

[193] B. Zhao, L. Fei-Fei, and E. P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA, 20 - 25 June 2011.

[194] Z. L. Zhou and K. K. Chee. The nearest feature midpoint - a novel approach for pattern classification. *International Journal of Information Technology*, 11(1):1–15, 2005.

[195] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson. Acoustic fall detection using Gaussian mixture models and GMM supervectors. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 0, pages 69 – 72, Taipei, Taiwan, 19 – 24 April 2009. IEEE Computer Society.

Index