Unsupervised Learning Methods for Statistical Signal Processing

vorgelegt von
Diplom Ingenieur
Roland Vollgraf
aus Berlin

Von der Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin
zur Erlangung des Akademischen Grades
Doktor der Ingenieurwissenschaften
– Dr. Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Jörg Raisch
Berichter:	Prof. Dr. rer. nat. Klaus Obermayer
Berichter:	Univ.-Prof. Dr. Sepp Hochreiter

Tag der wissenschaftlichen Aussprache: 19. September 2006

Berlin 2006

D 83

Zusammenfassung

In vielen Problemstellungen der Künstlichen Intelligenz und des Maschinellen Lernens bedient man sich statistischer Methoden, um aus einer gegeben Menge von oftmals numerischen Daten relevante Merkmale und Informationen zu extrahieren. Die dazu in Betracht kommenden Vorgehensweisen bestehen stets aus zwei Teilen. Zum Ersten sind die gegebenen Daten in den Kontext eines plausiblen statistischen Modells zu stellen, wobei die Daten eine empirische Beschreibung dieses Modells erlauben müssen. Zum Zweiten ist eine geeignete statistische Methode zu finden, die aus dem Modell, repräsentiert durch die Daten, die gewünschten Merkmale extrahieren kann.

Unter diesem Gesichtspunkt wird aus dem Titel bereits klar, dass in der vorliegenden Arbeit der Schwerpunkt auf Verfahren liegt, die die gegeben Daten als statistische Signale oder, in der Terminologie der Wahrscheinlichkeitstheorie, als stochastische Prozesse interpretieren und dass die verwendeten statistischen Methoden zu den unüberwachten Lernverfahren zählen.

Die Interpretation eines endlichen Datensatzes als eine oder mehrere Realisierungen eines stochastischen Prozesses ist durchaus nicht zwingend (und manchmal sogar unangebracht). In aller Regel ist der gegebene Datensatz mehrdimensional und numerischer Art oder lässt sich entsprechend umformen. Ein deutlich einfacheres statistisches Modell ergibt sich wenn man die Daten entlang einer Dimension als Realisierungen einer multivariaten Zufallsvariable auffasst. Dieser Ansatz liegt vielen maschinellen Lernverfahren zu Grunde, wie z.B. *Clustering, Hauptkomponentenanalyse, Projection Pursuit* oder *Independent Component Analysis (ICA)* als einige Vertreter unüberwachter Lernverfahren. Jedoch gerade am Beispiel von ICA lässt sich gut zeigen, wo das Modell einer multivariaten Zufallsvariable versagt. Als Gegenstück zu ICA kann man *Second Order Blind Source Separation* auffassen. Beide Methoden dienen der blinden Quellentrennung und resultieren in einer linearen Transformation des angenommenen statistischen Modells. Während sich jedoch ICA allein aus der Verteilung einer multivariaten Zufallsvariable ableiten lässt, benötigt man für Second Order BSS "zeitliche" Korrelationen oder Nichtstationaritäten – Größen die in dem Modell einer multivariaten Zufallsvariable nicht enthalten sind, wohl aber in dem generelleren Modell eines stochastischen Prozesses.

Dieses generellere Modell kann als roter Faden durch die vorliegende Arbeit angesehen werden. Kapitel 1 widmet sich daher der formalen Definition stochastischer Prozesse und wichtigen Eigenschaften derselben wie z.B. Stationarität und Ergodizität.

Kann man eine plausible Interpretation eines gegebenen Datensatzes als stochastischen Prozess finden, dann ist auch sofort klar, was unter statistischer Signalverarbeitung zu verstehen ist, nämlich eine jede Abbildung, die einen stochastischen Prozess in einen anderen überführt. Unter diesem Gesichtspunkt kann man überwachte und unüberwachte Lernverfahren unterscheiden. Überwachte sind solche, bei denen für eine Realisierung des einen Prozesses (der gegebene Datensatz) auch eine Realisierung des anderen verfügbar ist (Target-Datensatz), wobei eine Abbildung gesucht wird, die konsistent mit beiden Realisierungen ist, und die generell genug ist, um auch mit zukünftigen Realisierungen beider stochastischer Prozesse konsistent zu sein. Im Gegensatz dazu ist beim unüberwachten Lernen eine Abbildung gesucht, sodass der resultierende stochastische Prozess bestimmte statistische Eigenschaften aufweist, welche sich aus der Problemstellung ergeben.

Die vorliegende Arbeit beinhaltet die Ergebnisse verschiedener Arbeiten zu Problemen der statistischen Datenanalyse. Aufgrund der zahlreichen Kooperationen, die zwischen der Arbeitsgruppe "Neuronale Informationsverarbeitung" und vielen anderen Arbeitsgruppen bestehen, stammen die Aufgaben aus zum Teil recht verschiedenen Domänen. Viele kamen dabei aus dem Gebiet der biomedizinischen Datananalyse. Allen gemein ist, dass die gegebenen Daten sehr gut zu den Modellen stochastischer Prozesse passten. Bis auf Abschnitt 4.3 handelt es sich bei den dargestellten Methoden um unüberwachtes Lernen von signalverarbeitenden Abbildungen. Deren Charakteristiken entsprechend ist die vorliegende Arbeit gegliedert in *Instantane Lineare Funktionen, Optimale Lineare Filter, Nichtlineare Filter* und *Slow Feature Analyse* als allgemeine nichtlineare Methode. Dabei erheben die einzelnen Kapitel keinen Anspruch auf Vollständigkeit bezüglich der dargestellten Klassen von Methoden. Vielmehr sollen einzelne, dafür aber besonders interessante Verfahren und Ergebnisse vorgestellt werden.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

General symbols

$(\cdot)^T$, $(\cdot)^{-1}$, $(\cdot)^{-T}$	matrix transpose, inverse and transpose inverse
$\left\langle \cdot \right\rangle$	expectation
$\odot$	Hadamard (element wise) product of two matrices
$\star$	cross-correlation
$\lvert \cdot \rvert$	absolute value; number of elements in a set
$\lVert \cdot \rVert$	Euklidean length (of vectors)
$\lVert \cdot \rVert_F$	Frobenius norm
$\mathbf{1}$	square matrix with all elements equal to 1
conj$(\cdot)$	conjugate complex
det$(\cdot)$	determinant
dg$(\cdot)$	the diagonal matrix defined by the argument
$\overline{\text{dg}}(\cdot)$	the off-diagonal matrix defined by the argument
$\mathcal{F}[\cdot]$	discrete Fourier transform
$\mathcal{F}^{-1}[\cdot]$	inverse discrete Fourier transform
$\bar{\mathbf{I}}$	inverse unity matrix, $\bar{\mathbf{I}} = \mathbf{1} - \mathbf{I}$
$\mathbf{I}$	unity matrix
i	index variable
j	imaginary unit
j	index variable
$\mathbf{K}$	kernel matrix
K	natural number
k	index variable
$k(\cdot, \cdot)$	kernel function
l	index variable
λ	Lagrange multiplier
$\mathbf{\Phi}(\cdot)$	feature space projection
sign$(\cdot)$	signum function
t, τ, δt	temporal index rsp. offset
tr$(\cdot)$	matrix trace

Symbols specific to chapter 1

$\mathscr{A}$	σ-algebra defined on Ω
A, B	measurable subsets of Ω
$\mathscr{B}$	σ-algebra of all Borel subsets of E
E	state space of stochastic process

f	signal processing function
$\mathscr{H}(\mathrm{I})$	set of all non-empty, finite subsets of I
$\hat{\mathrm{I}}$	index set in finite datasets
I	temporal index set of stochastic process
J	receptive field, subset of I
Ω	arbitrary, measurable space
ω	element of Ω
P_{I}, P_{J}	probability measure
$\mathbf{r}$	channel index
t	temporal index variable
θ_τ	shift operator
$\mathbf{w}$	parameter vector of f
$(x_t)_{\mathrm{t}\in\mathrm{I}}$	stochastic process
$\mathbf{x}_{\mathrm{J}}$, $\mathbf{x}_t$	realization of stochastic process $\mathbf{x}$
$\mathcal{X}$	dataset
X_{J}, X_t	marginal event of stochastic process x

Symbols specific to chapter 2

$\left\langle \cdot \right\rangle_{\mathbf{C}}$	average over all matrices $\mathbf{C} \in C$
$\equiv$	matrix equality up to permutations and scalings
$\nabla_{\mathbf{W}}$	gradient w.r.t. $\mathbf{W}$
$\widetilde{\nabla}_{\mathbf{W}}$	natural gradient w.r.t. $\mathbf{W}$
$\mathbf{A}$	mixing matrix
$\mathbf{a}$	column vector of $\mathbf{A}$
$\mathfrak{A}$	secondary mixing matrix in Two-Step ICA
$\mathfrak{a}_{\star j}$	feature vector in two-step ICA
α, α_k	regularization and weighting coefficients
$\mathbf{b}_i$	vector
$\mathbf{B}$	arbitrary $N \times N$ matrix
$\mathbf{C}$	covariance matrix
$\mathbf{C}_0$, $\mathbf{C}_k$, $\mathbf{C}_\tau$	second order correlation matrix
C	set of correlation matrices
$\mathbf{D}$	$N \times N$ matrix used by QDIAG
$\mathscr{D}_N$	set of all non-singular, diagonal $N \times N$ matrices
$\mathfrak{I}(\mathbf{s})$	source assumptions
L	BSS objective function; Lagrangian.
$\mathbf{\Lambda}$	diagonal matrix
$\mathbf{M}$	arbitrary matrix
M	natural number, number of channels
N	natural number, number of channels
n, $\mathbf{n}$	noise signal
$\mathbf{P}$	whitening (sphering) matrix
$\pi(\cdot)$	product of the diagonal elements of the argument matrix
$\mathbf{Q}$	permutation matrix
$\mathscr{Q}_N$	set of all non-singular $N \times N$ permutation matrices
$\mathbf{s}_{\mathrm{J}}$, $\mathbf{s}_t$	realization of stochastic process $\mathbf{s}$
$\mathbf{S}_t$	patch of stochastic process $\mathbf{s}$
$\mathbf{s}_{\star j}$	j-th column vector of $\mathbf{S}$
$\mathbf{s}_{i\star}$	i-th row vector of $\mathbf{S}$

S_J, S_t	marginal event of stochastic process s
s, $\mathbf{s}$, $(\mathbf{s}_t)_{t\in\mathrm{I}}$	vector valued stochastic process, sources.
$\Sigma_\mathbf{a}$, $\Sigma_\mathbf{w}$	quadratic forms for regularization terms
$\mathscr{T}_\mathbf{W}$	tangent space at $\mathbf{W}$
$\mathbf{U}$	matrix of feature coefficients
$\mathbf{U}$	unitary matrix
$\mathbf{U}^+$	matrix of dependent feature coefficients
$\mathbf{U}^\circ$	matrix of independent feature coefficients
$\mathfrak{W}$	secondary mixing matrix in two-step ICA
$\mathbf{W}$	de-mixing matrix
$\mathbf{w}_i$	i-th row vector of $\mathbf{W}$
$\mathscr{W}$	set of equivalent solutions $\mathbf{W}$
$\mathcal{W}_{adm}$	set of admissible solutions in QDIAG
x, $\mathbf{x}$, $(\mathbf{x}_t)_{t\in\mathrm{I}}$	stochastic process, observations
y, $\mathbf{y}$, $(\mathbf{y}_t)_{t\in\mathrm{I}}$	stochastic process, outputs

Symbols specific to chapter 3

α^*	optimal value of regularization parameter
E	error value
ϵ	bound on the maximal values of the output cross-correlation functions
$\bar{\mathbf{F}}$	matrix summarizing all filters $\bar{f}$
$\bar{f}^*$	optimal filter in vector notation
$f, \boldsymbol{f}, \bar{\boldsymbol{f}}$	filter function
$\mathbf{H}$	Hessian matrix of the Lagrange equation
κ_d	cumulant of order d
(l_t)	template response
$(n_t)_{t\in\mathrm{I}}$, $(\mathbf{n}_t)_{t\in\mathrm{I}}$	noise signal
$\tilde{\boldsymbol{n}}_t$	patch vector of noise process n rsp. $\mathbf{n}$
ω	angular frequency
$q_{\{x\}}$, $q_{\{y\}}$, $q_{\{PCA\}}$	spike sorting performance measures
ρ_d	standardized cumulant of order d
σ_n^2	noise variance
$\theta(\cdot)$, $\Theta(\cdot)$	non-linear discrimination and momentum functions
$(v_t)_{t\in\mathrm{I}}$	real valued, sparse stochastic signal
$\tilde{\boldsymbol{x}}_t$	patch vector of stochastic process x rsp. $\mathbf{x}$
$\boldsymbol{\Xi}$	matrix which contains the template ξ in every row
$\xi, \boldsymbol{\xi}, \bar{\boldsymbol{\xi}}$	template, wave-form

Symbols specific to chapter 4

$\alpha_i^{(*)}$	Lagrange multiplier in SVR
$(b_\tau)_{\tau\in\mathrm{J}}$	basis of Taylor expansion
β	regularization parameter
c	model parameter in ν-SVR
ε	margin of the ε-sensitive loss in SVR
$\mathcal{F}$	feature space
f, f_i	arbitrary (filter) function
$g_{\tau_1,\ldots,\tau_d}$	tensor of partial derivatives of order d
γ	RBF-kernel parameter

ν	model parameter in ν-SVR
$(v_{\tau,n})_{\tau\in\mathrm{J},n=1\ldots N}$	tensor decomposition matrix
$\Phi(\cdot)$, $\Phi_i(\cdot)$	basis function
$\pi(\cdot,\ldots,\cdot)$	permutation of indices
Q_d	set of detected peaks
Q_{Exp}	set of manually labeled vesicles
Q_{match}	set of correctly detected vesicles
Q_{Exp}	set of detected vesicles
T_x, T_f	receptive field sizes
$(W_k)_{k=1\ldots T_x}$	Fourier coefficients of $(w_t)_{t=1\ldots T_f}$
$(X_k)_{k=1\ldots T_x}$	Fourier coefficients of $(x_t)_{t=1\ldots T_x}$
$\xi_i^{(*)}$	slack variable in SVR
$(Y_k)_{k=1\ldots T_x}$	Fourier coefficients of $(y_t)_{t=1\ldots T_x}$
$(z_t)_{t\in\mathrm{I}'}$	filtered outputs

Symbols specific to chapter 5

$\star$	set convolution
$\bar{\mathbf{1}}$	vector with all elements equal to 1
$\mathbf{a}$	linear output projection
$\boldsymbol{f}^i$	multi-channel filter in layer i
I_r	spatial index set in hierarchical SFA
I_t	temporal index set in hierarchical SFA
J_r^i	spatial receptive field of $\boldsymbol{f}^i$
J_r	overall spatial receptive field of last last layer
$\mathbf{K}$	kernel matrix
$\mathbf{k}(\mathbf{x}_t)$	row vector of the kernel matrix $\mathbf{K}$
$\dot{\mathbf{K}}$	temporal derivative kernel matrix
$\mathbf{P}$	sphering matrix
q_t	'landmark function' in hierarchical SFA experiment
$\hat{S}(\mathbf{y})$	empirical slowness value of slow feature vector $\mathbf{y}$
$S(\mathbf{y})$	slowness value of slow feature vector $\mathbf{y}$
$\mathbf{t}=(t,r)^T$	temporal/spatial index in hierarchical SFA
T	length of the dataset
$\mathbf{V}$	projection matrix in feature space
$\mathbf{W}$	$M\times N$ projection matrix in linear SFA
$\mathbf{w}^i$	parameter vector of $\boldsymbol{f}^i$
$\mathbf{w}_i$	i-th row vector of $\mathbf{W}$
$\mathbf{x}_t$	input data vector
$\dot{\mathbf{x}}_t$	temporal derivative of input data vector
$\mathbf{y}^i, tr$	slow feature vector of layer i at spatio/temporal offset t,r)
$\mathbf{y}_t$	slow feature vector
$\dot{\mathbf{y}}_t$	temporal derivative of slow feature vector
$\mathbf{z}_t$	(initial) support vector; parameter vector of explicit expansion

Acknowledgments

At first, I wish to express my gratitude to Prof. Klaus Obermayer, who was the supervisor of my research work. His lecture, 'Neural Information Processing', was when I first got in contact with the matter of neural networks and machine learning. While firstly I was somewhat overwhelmed by all that math behind the methods, during my time at Prof. Obermayer's lab I learned to understand the theories and to discover the beauty, which is behind some weird formulas. I want to thank him for constantly encouraging me to publish my results, and for the opportunities he offered to present them at several international conferences.

I wish to express my special thanks to Prof. Sepp Hochreiter, who was member of our research group during a long period of my work. It has always been a pleasure to talk and discuss with him during our daily coffee breaks. The most fruitful and original ideas often came up at these occasions. The more I'm glad that he agreed to be the second assessor of my PhD.

I wish to thank Dr. Matthias Munk and Gordon Pipa from the *MPI Brain Research, Frankfurt* for the fruitful collaboration in the development of the spike-sorting procedures. All data I used had been recorded there. Visiting their lab it was highly interesting to learn about their research and the procedures that come prior to the data analysis step.

My thanks are to Dr. Laurenz Wiskott for the exchange of ideas about Slow Feature Analysis. His work and the concept of place cells were the inspiration to focus on SFA for the visual data processing in the robotics project.

I am indebted to Dr. Marek Musial for the endeavors he made to bring the robotics project to live.

Last not least I would like to thank my colleagues at NI for not less than the good time I had during my work at the NI group.

Institutional support of the work leading to this thesis was provided by:
Wellcome Trust, grant no. 10008261 (section 2.4 and chapter 3), *Deutsche Forschungsgemeinschaft*, grant no. DFG OB 102/3-1 and *Wellcome Trust*, grant no. 050080/Z/97 (section 2.5), *Deutsche Forschungsgemeinschaft*, grant no. DFG SE 931/1-1 and DFG OB 102/3-1 and *Wellcome Trust*, grant no. 061113/Z/00 (section 2.6), *Bundesministerium für Bildung und Forschung*, grant no. 0311559 (section 4.3), and *Deutsche Forschungsgemeinschaft*, grant no. DFG OB 102/7-1 (section 4.4 and chapter 5).

Preface

In many problems of Artificial Intelligence and Machine Learning, one makes use of statistical methods in order to extract relevant features and information from a given set of often numerical data. The approaches that come into consideration therefor always comprise two parts. At first, the given data have to be put into the context of a plausible statistical model, where the data must allow to empirically describe the model. Secondly, one has to find a suitable statistical method that allows to extract the desired features from the model, which is represented by the data.

From this point of view, the title already make apparent that focus of this thesis is on methods that interpret given data as statistical signals or, in the terminology of probability theory, as stochastic processes, and that make use of statistical methods which belong to the family of unsupervised learning methods.

The interpretation of a finite dataset as one or more realizations of a stochastic process in not stringent (and even sometimes inappropriate). In general the given dataset is multi-dimensional and of numerical nature or can be transformed that way. A quite simpler model could be achieved if all data along one dimension were interpreted as the realizations of a multi-variate random variable. This approach underlies many machine learning procedures like, e.g., *Clustering*, *Principal Component Analysis*, *Projection Pursuit*, or *Independent Component Analysis (ICA)* to name some representative unsupervised learning methods. However, even with ICA it becomes apparent where the model of a multi-variate random variable fails. *Second Order Blind Source Separation* can be considered as a counterpart to ICA. Both methods approach blind source separation and lead to a linear transformation of the underlying statistical model. However, while ICA can be derived solely from a multi-variate statistical distribution, Second Order BSS requires 'temporal' correlations or non-stationarities – quantities that are not contained in the model of a multi-variate random variable, but are in the more general model of a stochastic process.

This more general model can be seen as a thread through this thesis. Therefore, chapter 1 is devoted to the formal definition of stochastic processes and their important properties like stationarity and ergodicity.

If one can find a plausible interpretation of a given dataset as a stochastic process, then immediately it becomes clear what is meant by the term 'statistical signal processing', namely any mapping that transforms one stochastic process into another one. In this respect one can distinguish supervised and unsupervised learning methods. Supervised learning methods are those for which with a realization of the one stochastic process (the given dataset), also a realization of the other stochastic process is available (a target dataset) such that a mapping from one process to the other is searched, that is consistent with both realizations and sufficiently general to be consistent with future realization of both stochastic processes. In contrast, in unsupervised learning methods a mapping is searched such that the resulting stochastic process exhibits certain statistical properties, which arise from the problem at hand.

This thesis contains the results of my work on different problems in statistical data analysis. Due to the numerous collaborations that exist between the *Neural Information Processing Group* and many other labs, the data and the associated problems originate from quite different domains. Most of them were from the field of biomedical data analysis. Common to all is the fact that they fit well into the framework of stochastic processes. Except for section 4.3 all presented methods are unsupervised learning of signal processing

mappings. According to their characteristics this thesis is organized into the chapters *Instantaneous Linear Functions, Optimal Linear Filters, Non-Linear Filtering,* and *Slow Feature Analysis* as a general non-linear method. The individual chapters do not claim to be complete in terms of classes of signal processing methods they represent. Rather some individual procedures and results that are of particular interest shall be presented.

A large part of this thesis has been published in scientific journals or conference proceedings.

The work on extraction of single sources form linear mixtures was presented as a talk at the *International Conference on Artificial Neural Networks* 2001 in Vienna and is published in Vollgraf et al. (2001). In the same year the multi-dimensional ICA approach was presented at *NIPS* and published in the conference proceedings (Vollgraf and Obermayer, 2002).

At that time I started my work on the data analysis of extracellularly recorded signals of neural activity, the so called 'spike-sorting', in a collaboration with the *MPIH, Frankfurt,* and Dr. Matthias Munck. After a long and straining review process, the approach with optimal single-channel filters eventually was published in Vollgraf et al. (2005b). Meanwhile a presentation of the results could be given at the *30th Göttingen Neurobiology Conference* (Vollgraf et al., 2005a). In the thesis this material is distributed over the sections 3.1, 3.2, and 3.5, because the work on optimal multi-channel filters (section 3.3 and 3.4) is the logical continuation of the optimal single-channel filters (although, chronologically it was done after the spike sorting application presented in section 3.5). The work of section 3.3 was published separately in Vollgraf and Obermayer (2006a).

In 2003, in the course of a collaboration with the *Laboratory of Invertebrate Neurobiology* and Dr. I.A. Meinertzhagen, the research on optimal non-linear filtering (section 4.3) was done and soon after presented rsp. published in Scholz et al. (2003) and Vollgraf et al. (2004). The insights about the decomposition of non-linear filters achieved during this study is an important fundamental for the ongoing work on Slow Feature Analysis and non-linear filtering of video data (cf. chapter 5). The problem that non-linear filtering operations still can be so much time consuming was decisive to spend some work on how sparse optimization for Kernel-SFA (and other second order kernel methods) can be achieved. The result was the *Hyperellipsoidal Conjugate Gradient Descent* algorithm (section 4.4), which will be presented at the *IJCNN 2006* workshop (Vollgraf and Obermayer, 2006c).

The development of the QDIAG algorithm (cf. section 2.4) was the consequence of a particular problem that involved the approximate diagonalization of several quite large matrices, where non of the existing algorithms could solve the problem satisfactory. As it is so often the case, the actual problem turned out to be of minor relevance, but the QDIAG algorithm proved to be a very powerful tool for general matrix diagonalizations. It is published in Vollgraf and Obermayer (2006b).

Chapter 1

Spatio/temporal data

In this chapter we will first define what is understood under a signal in the statistical sense. Then this definition will offer a quite intuitive way to distinguish different ways of statistical signal processing.

1.1 Stochastic signals

When talking about signals one usually has in mind a set of data samples x that are elements of some common state space E. At the same time these data samples exhibit a certain temporal or sometimes spatial structure, which is given by another set I, the index set.

The subject of study in this thesis will be real valued data of finite dimension,

$$x \in \mathrm{E} := \mathbb{R}^{R_1 \times \ldots \times R_{N_r}} , \quad N_r \geqslant 1, R_i \geqslant 1 , \tag{1.1}$$

i.e. the state space E can be the set of all real valued scalars, vectors, matrices, or finite dimensional arrays. Indices

$$\mathbf{r} \in \mathrm{R} := [1, R_1] \times \ldots \times [1, R_{N_r}] \subset \mathbb{N}^{N_r} \tag{1.2}$$

into E are called *channel indices*. We will consider data that have a finite dimensional, discrete 'temporal' structure, giving rise to the index set to be a finite power of the set of whole numbers,

$$t \in \mathrm{I} := \mathbb{Z}^{N_t} , \quad N_t \geqslant 1 . \tag{1.3}$$

Even for more than one dimensional signals ($N_t > 1$) we may refer to t as *temporal index* and to I as *temporal index set or temporal set*. In general we refer to signals according to equations (1.1) and (1.3) as *time discrete real signals*.

In order to access their statistical properties it is reasonable to consider signals as stochastic processes. A stochastic process, is given by the quadruple

$$(\Omega, \mathscr{A}, P, (x_t)_{t \in \mathrm{I}}) , \tag{1.4}$$

where $(\Omega, \mathscr{A}, P)$ is some probability space and $x_\mathrm{I} := (x_t)_{t \in \mathrm{I}}$ is a family of random variables associated with that probability space. These random variables may assume values in the measurable space $(\mathrm{E}, \mathscr{B})$, where $\mathscr{B}$ is the σ-algebra of all Borel subsets of E. For any given $\omega \in \Omega$ the mapping from I to E, given by $t \mapsto x_t(\omega)$ is called a *realization* or *trajectory* of the process.

The trajectories of the stochastic process (1.4) give rise to the stochastic process

$$(\mathrm{E}^\mathrm{I}, \mathscr{B}(\mathrm{E}^\mathrm{I}), P_\mathrm{I}, (x_t)_{t \in \mathrm{I}}) \quad \text{with} \quad x_t(\omega) = \omega_t , \tag{1.5}$$

which is called the canonical process belonging to (1.4). Hence, any statistical properties of the trajectories of the process are subject to the probability measure P_I. In general it

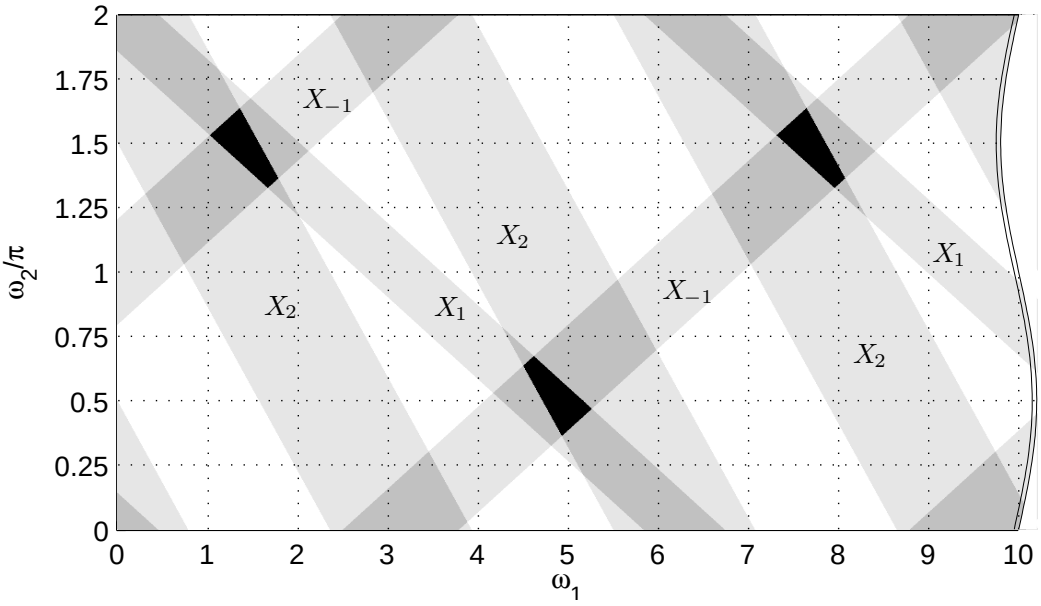


Figure 1.1: The event $X_{\{-1,1,2\}}$ with $X_{-1} = [-1, -0.8]$, $X_1 = [0.9, 1]$ and $X_2 = [0, 1]$ on the stochstic process of example 1. On measurable subsets of the displayed space Ω the probability measure P is defined. Gray stripes show the subsets of the individual events X_t. The measure P of their intersection (black areas) is the probability of occurence of event X_J.

is impossible to define P_I directly, because of the infinite dimensionality of the events generated by the stochastic process. However, it is possible to define probability measures P_J for finite marginal events

$$X_J := \prod_{t \in J} (X_J)_t \quad \text{with} \quad (X_J)_t \in \mathscr{B}(E)\,,\ J \in \mathscr{H}(I), \tag{1.6}$$

where $\mathscr{H}(I)$ is the set of all non-empty, finite subsets of I. According to the theorem of Daniell-Kolmogorov (Brauer (2002), §35.3, p. 307) P_I is uniquely defined by the projective family $(P_J)_{J \in \mathscr{H}(I)}$ of all finite marginal distributions of the process. The family $(P_J)_{J \in \mathscr{H}(I)}$ is *projective* if for any $H \subset J \in \mathscr{H}(I)$ and $X_H \in \mathscr{B}(E^H)$, $X_J \in \mathscr{B}(E^J)$ holds

$$P_H(X_H) = P_J(X_J) \tag{1.7}$$

whenever

$$X_J = \{x_J \in E^J : x_t \in (X_H)_t\,,\ t \in H\} = \prod_{t \in J} \begin{cases} (X_H)_t, & t \in H \\ E, & t \in J \backslash H \end{cases} . \tag{1.8}$$

Now, the theorem states that to any projective family there uniquely exist a probability measure P_I, the *projective limes*, so that

$$P_J(X_J) = P_I(\{x_I \in E^I : x_t \in (X_J)_t\,,\ t \in J\}) \quad \text{for all} \quad J \in \mathscr{H}(I)\,.$$

This is very convenient as it allows us to fully specify the canonical process only by its finite marginal distributions. The following two examples shall illustrate this.

Example 1

Be $\Omega = \mathbb{R}^+ \times [0, 2\pi)$, $\mathscr{A} = \mathscr{B}(\Omega)$, $E = [-1, 1]$ and $I = \mathbb{Z}$. Then the stochastic process

$$x_t(\omega) = \cos(\omega_1 t + \omega_2)\,, \quad (\omega_1, \omega_2)^T \in \Omega \tag{1.9}$$

describes all discrete time cosine signals, the frequency (ω_1) and phase (ω_2), which are subject to the distribution $(\Omega, \mathscr{B}(\Omega), P)$. For finite marginal events

$$\mathrm{X_J} = \prod_{t\in\mathrm{J}} (\mathrm{X_J})_t \,,\ \mathrm{J} \in \mathscr{H}(\mathrm{I}) \,,\ (\mathrm{X_J})_t \in \mathscr{B}(\mathrm{E}) \tag{1.10}$$

the probability measure P_J is given by

$$P_\mathrm{J}(\mathrm{X_J}) = P\left(\bigcap_{t\in\mathrm{J}} \{\omega \in \Omega : \cos(\omega_1 t + \omega_2) \in (\mathrm{X_J})_t\}\right) . \tag{1.11}$$

Because $x_t(\omega)$ is measurable, $\bigcap_{t\in\mathrm{J}}\{\omega \in \Omega : \cos(\omega_1 t + \omega_2) \in (\mathrm{X_J})_t\} \in \mathscr{B}(\Omega)$ and, hence, P_J exists. See figure 1.1 for an illustration of the event $\mathrm{X}_{\{-1,1,2\}}$.

Example 2: white noise signal

In this example the state space is the set of real numbers, $\mathrm{E} = \mathbb{R}$, and the index set is $\mathrm{I} = \mathbb{Z}^{N_t}$. A white noise signal can be seen as the quadruple $(\mathrm{E^I}, \mathscr{B}(\mathrm{E^I}), P, (x_t)_{t\in\mathrm{I}})$ where P is a probability measure so that $(E, \mathscr{B}(\mathrm{E}), P)$ is the distribution of every single x_t. To complete this example we need to define the corresponding probability measure P_I on $(\mathrm{E^I}, \mathscr{B}(\mathrm{E^I}))$.

Consider a collection of subsets of $\mathrm{E^I}$ of the form

$$\mathrm{X'_J} := \prod_{t\in\mathrm{I}} \begin{cases} \mathrm{X}_t, & t \in \mathrm{J} \\ E, & t \notin \mathrm{J} \end{cases} = \{(x_t)_{t\in\mathrm{I}} : x_t \in \mathrm{X}_t, t \in \mathrm{J}\} \,, \tag{1.12}$$

where $\mathrm{J} \in \mathscr{H}(\mathrm{I})$ and $\mathrm{X}_t \in \mathscr{B}(\mathrm{E})$. These subsets form a semi-algebra $\mathscr{S}(\mathrm{E^I})$ on $\mathrm{E^I}$, meaning that: (i) $\varnothing \in \mathscr{S}(\mathrm{E^I})$, (ii) if $\mathrm{A}, \mathrm{B} \in \mathscr{S}(\mathrm{E^I})$ than also $\mathrm{A} \cup \mathrm{B} \in \mathscr{S}(\mathrm{E^I})$, and (iii) for every $\mathrm{A} \in \mathscr{S}(\mathrm{E^I})$ there exists a finite number of sets $\mathrm{B}_i \in \mathscr{S}(\mathrm{E^I})$ so that $\mathrm{E^I}\backslash\mathrm{A} = \bigcup_{i=1}^n \mathrm{B}_i$. By construction we can easily define to all elements of $\mathscr{S}(\mathrm{E^I})$ (but not necessarily to all elements of $\mathscr{B}(\mathrm{E^I})$) the measure

$$P_\mathscr{S}(\mathrm{X'_J}) = \prod_{t\in\mathrm{J}} P(\mathrm{X}_t) \,, \tag{1.13}$$

where $\mathrm{X'_J} \in \mathscr{S}(\mathrm{E^I})$ is the extension of the event $\mathrm{X_J}$ to $\mathscr{S}(\mathrm{E^I})$ according to equation (1.12). Clearly, for any disjoint sets $\mathrm{J}, \mathrm{H} \in \mathscr{H}(\mathrm{I})$ holds

$$P_\mathscr{S}(\mathrm{X'_J} \cap \mathrm{X'_H}) = P_\mathscr{S}(\mathrm{X'_J}) P_\mathscr{S}(\mathrm{X'_H}) \,, \tag{1.14}$$

hence, disjoint events $\mathrm{X_J}$ and $\mathrm{X_H}$ are independent. $P_\mathscr{S}$ is countably additive, i.e. for any countable set of pairwise disjoint $\mathrm{B}_i \in \mathscr{S}(\mathrm{E^I})$ holds

$$P_\mathscr{S}\left(\bigcup_{i=1}^{\infty} \mathrm{B}_i\right) = \sum_{i=1}^{N} P_\mathscr{S}(\mathrm{B}_i) \,. \tag{1.15}$$

With $P_\mathscr{S}(\mathrm{E^I}) = 1$ this is sufficient for a unique probability measure P_I on $(\mathrm{E^I}, \mathscr{B}(\mathrm{E^I}))$ to exist, which extends $P_\mathscr{S}$ to $\mathscr{B}(\mathrm{E^I})$, such that all elements of $\mathscr{B}(\mathrm{E^I})\backslash\mathscr{S}(\mathrm{E^I})$ have the measure zero w.r.t. P_I. For the proof of the last statement see Walters (1982, pp. 3–6) and Kingman and Taylor (1966, p. 140).

1.2 Stationarity and ergodicity

1.2.1 Moments

Generally for a given signal and the associated stochastic process $(\Omega, \mathscr{A}, P, (x_t)_{t\in\mathrm{I}})$, it is impossible fully determine the underlying probability measure P. However, in many cases it is sufficient to determine certain interesting moments of its trajectories $x_\mathrm{I} = (x_t)_{t\in\mathrm{I}}$,

$$\left\langle g((x_t)_{t\in\mathrm{I}}) \right\rangle := \int_\Omega g((x_t)_{t\in\mathrm{I}}) \,\mathrm{d}P_\mathrm{I} = \int_\Omega g((x_t)_{t\in\mathrm{I}}) \,\mathrm{d}P \,, \tag{1.16}$$

where g is a measurable function $\mathrm{E^I} \to \mathbb{R}$.

1.2.2 Stationarity

A stochastic process $(\Omega, \mathscr{A}, P, (x_t)_{t\in \mathrm{I}})$ is called a stochastic process with the discrete shift θ_τ, $\tau \in \mathrm{I}$, if there exist a transformation $\theta_\tau : \Omega \to \Omega$ with $x_t \circ \theta_\tau = x_{t+\tau}$.

A stochastic process with shift θ_τ is called *stationary w.r.t.* θ_τ, if θ_τ is a measure preserving transformation, i.e.

$$\theta_\tau{}^{-1}(\mathrm{A}) \in \mathscr{A} \text{ and } P(\theta_\tau{}^{-1}(\mathrm{A})) = P(\mathrm{A}) \tag{1.17}$$

holds for all $\mathrm{A} \in \mathscr{A}$. The importance of stationarity assumptions results from the fact that for stationarity processes any moments are shift invariant,

$$\left\langle g(x_{\mathrm{I}}) \right\rangle = \left\langle g(x_{\mathrm{I}} \circ \theta_\tau) \right\rangle \tag{1.18}$$

which directly follows from equation (1.16).

For the stochastic process in example 1 a shift is given by

$$\theta_\tau((\omega_1, \omega_2)^T) = (\omega_1, (\tau\omega_1 + \omega_2) \mod 2\pi)^T , \tag{1.19}$$

hence

$$x_t \circ \theta_\tau = \cos(\omega_1(t + \tau) + \omega_2) . \tag{1.20}$$

θ_τ leads to a cyclic shift of ω_2, and, thus, example 1 represents a stationary process if, and only if, P is invariant on ω_2, meaning that the phase of the cosine signals is uniformly distributed over the interval $[0, 2\pi)$.

By construction, equation (1.13), the white noise process in example 2 is a stationary process.

1.2.3 Ergodicity

The shift invariance of moments of stationary processes, equation (1.18), gives rise to the idea to compute empirical moments over increasing numbers of shifts,

$$\left\langle g(x_{\mathrm{J}}) \right\rangle = \lim_{n\to\infty} \frac{1}{\#\mathrm{H}_n} \sum_{\tau\in \mathrm{H}_n} g(x_{\mathrm{J}} \circ \theta_\tau) , \tag{1.21}$$

with $\mathrm{H}_n \subset \mathrm{H}_{n+1} \in \mathscr{H}(\mathrm{I})$. The advantage of doing so is that one needs only one realization $x_{\mathrm{I}}(\omega)$ of the process for the estimation of moments. However, in order for this sequence to converge to $\left\langle g(x_{\mathrm{J}}) \right\rangle$ a further condition must be fulfilled: the process must be ergodic.

A stochastic process $(\Omega, \mathscr{A}, P, (x_t)_{t\in \mathrm{I}})$ is called *ergodic w.r.t. shifts* θ_τ if it is stationary w.r.t θ_τ, and if for all $\mathrm{A} \in \mathscr{A}$ holds

$$\theta_\tau{}^{-1}(\mathrm{A}) = \mathrm{A} \quad \Rightarrow \quad P(\mathrm{A}) \in \{0, 1\} . \tag{1.22}$$

In other words, there exist no non-trivial events A that are shift invariant. Consider a process that is stationary w.r.t. θ_τ, but not ergodic. Then, there exists an $\mathrm{A} \in \mathscr{A}$ with $0 < P(\mathrm{A}) < 1$ and $\theta_\tau{}^{-1}(\mathrm{A}) = \mathrm{A}$. Now, consider $\omega \in \mathrm{A}$ in equation (1.21). Then, however, there is a set $\mathrm{B} = \Omega\backslash\mathrm{A} \in \mathscr{A}$ with $P(\mathrm{B}) > 0$ so that

$$\left(\lim_{n\to\infty} \bigcup_{\tau\in \mathrm{H}_n} \theta_\tau(\omega) \right) \cap \mathrm{B} = \varnothing . \tag{1.23}$$

Hence, one can say that with non-zero probability $P(\mathrm{A})$ a subset $\mathrm{B} = \Omega\backslash\mathrm{A}$ with finite measure $P(\mathrm{B})$ will not be used for the computation of the empirical moment in equation (1.21), leading to incorrect results. On the other hand, it was proven with the *Birkhoff-Khinchin Ergodic Theorem* and related theorems (cf. Fröhlich et al. (1999, §2, pp. 12–)) that ergodicity is sufficient to correctly compute empirical moment according to equation (1.21).

Clearly, the stochastic process in example 1 is not ergodic unless $\omega_1 = \omega_1^*$ is constant and ω_2 is uniformly distributed, so that

$$P(\mathrm{A}) = \frac{1}{2\pi}\lambda(\{\omega_{2*} \in [0, 2\pi) : (\omega_1^*, \omega_2^*)^T \in \mathrm{A}\}) \,. \tag{1.24}$$

In all other cases one can construct sets $\mathrm{A} = \mathrm{A}_1 \times [0, 2\pi)$, $\mathrm{A}_1 \in \mathscr{B}(\mathbb{R})$ for which (1.22) does not hold.

1.3 Signal processing

Now, having signals defined as stochastic processes, it is quite intuitive to refer to any measurable function $f : \mathrm{E}^{\mathrm{I}} \to \mathrm{E}' = \mathbb{R}^{R'_1 \times \ldots \times R'_{N'_r}}$,

$$y = f((x_t)_{t\in\mathrm{I}}; \mathbf{w}) \,, \tag{1.25}$$

as *signal processing*. The vector $\mathbf{w}$ denotes the parameters of f. Because f is measurable, the resulting signal,

$$y_{\mathrm{I}} := (y_t)_{t\in\mathrm{I}} = (f((x_{t'+t})_{t'\in\mathrm{I}}; \mathbf{w}))_{t\in\mathrm{I}} \,, \tag{1.26}$$

is again a stochastic process on the same probability space $(\Omega, \mathscr{A}, P)$ that x_{I} is associated to. For real data signal processing applications, clearly one can consider only finite input functions $f : \mathrm{E}^{\mathrm{J}} \mapsto E'$, $\mathrm{J} \in \mathscr{H}(\mathrm{I})$. In this case we will call the set J the *receptive field of* f.

In the following sections a possible classification of signal processing functions according to their structure and their input set J will be presented. The organization of the subsequent chapters in this thesis is inspired by this classification.

1.3.1 Instantaneous, stationary functions

Instantaneous, stationary functions have the general structure

$$f : \mathrm{E}^{\{0\}} \mapsto \mathrm{E}' \,. \tag{1.27}$$

Thus the receptive field contains only one element, $t = 0$. This function is "instantaneous" because any y_t does only depend on x_t and not on any other $x_{t\neq t}$. The special case of a linear, instantaneous, stationary function has the form

$$(y_{\mathbf{r}',t'})_{t'\in\mathrm{I}} = (f(x_{t'}; \mathbf{w}))_{t'\in\mathrm{I}} = \left(\sum_{\mathbf{r}\in\mathrm{R}} w_{\mathbf{r}',\mathbf{r}}\, x_{\mathbf{r},t'}\right)_{t'\in\mathrm{I}} \,, \tag{1.28}$$

in which case $\mathbf{w} \in \mathbb{R}^{\mathrm{R}'\times\mathrm{R}}$.

1.3.2 Simple filter functions

A simple filter function f has the form

$$f : \mathbb{R}^{\mathrm{J}} \mapsto \mathbb{R} \,. \tag{1.29}$$

It is applied to every channel $\mathbf{r} \in \mathrm{R}$ individually,

$$(y_{\mathbf{r},t'})_{t'\in\mathrm{I}} = (f((x_{\mathbf{r},t+t'})_{t\in\mathrm{J}}; \mathbf{w}))_{t'\in\mathrm{I}} \,. \tag{1.30}$$

It may also be useful to define simple filter functions for every channel individually, in which case the channel index $\mathbf{r}$ is a parameter to f.

$$(y_{\mathbf{r},t'})_{t'\in\mathrm{I}} = (f((x_{\mathbf{r},t+t'})_{t\in\mathrm{J}}; \mathbf{w}, \mathbf{r}))_{t'\in\mathrm{I}} \,. \tag{1.31}$$

In many cases the receptive field J is defined as closed intervals around zero,

$$\mathrm{J} = [-t_1, t_1] \times \ldots \times [-t_{N_t}, t_{N_t}] \,. \tag{1.32}$$

The special case of a linear simple filter function has the form

$$(y_{\mathbf{r},t'})_{t' \in \mathrm{I}} = \left(\sum_{t \in \mathrm{J}} w_t x_{\mathbf{r},t+t'} \right)_{t' \in \mathrm{I}} \tag{1.33}$$

with $\mathbf{w} = (w_t)_{t \in \mathrm{J}} \in \mathbb{R}^{\mathrm{J}}$.

1.3.3 Multi-channel filters

Multi-channel filters represent the most general form of signal processing functions. They have the form

$$f : \mathrm{E}^{\mathrm{J}} \mapsto \mathrm{E}' \tag{1.34}$$

with

$$(y_{t'})_{t' \in \mathrm{I}} = (f((x_{t+t'})_{t \in \mathrm{J}}; \mathbf{w}))_{t' \in \mathrm{I}} \,. \tag{1.35}$$

Also for multi-channel filters often a local receptive field according to equation (1.32) is used. In a linear multi-channel filter holds $\mathbf{w} \in \mathbb{R}^{\mathrm{R}' \times \mathrm{R} \times \mathrm{J}}$ and

$$(y_{\mathbf{r}',t'})_{t' \in \mathrm{I}} = \left(\sum_{\mathbf{r} \in \mathrm{R}} \sum_{t \in \mathrm{J}} w_{\mathbf{r}',\mathbf{r},t+t'} \, x_{\mathbf{r},t+t'} \right)_{t' \in \mathrm{I}} \,. \tag{1.36}$$

It is often interesting to study cases in which a multi-channel filter can be split into an instantaneous function f_1 and simple filter functions f_2, either in this way:

$$\begin{aligned} (z_t)_{t \in \mathrm{I}} &= (f_1(x_t; \mathbf{w}_1))_{t \in \mathrm{I}} & (1.37) \\ (y_{\mathbf{r}',t'})_{t' \in \mathrm{I}} &= (f_2((z_{\mathbf{r}',t+t'})_{t \in \mathrm{J}}; \mathbf{w}_2, \mathbf{r}'))_{t' \in \mathrm{I}} \,, & (1.38) \end{aligned}$$

or in that way:

$$\begin{aligned} (z_{\mathbf{r},t'})_{t' \in \mathrm{I}} &= (f_2(x_{\mathbf{r},t+t'})_{t \in \mathrm{J}}; \mathbf{w}_2, \mathbf{r})_{t' \in \mathrm{I}} & (1.39) \\ (y_{t'})_{t' \in \mathrm{I}} &= (f_1(z_{t'}; \mathbf{w}_1))_{t' \in \mathrm{I}} \,. & (1.40) \end{aligned}$$

Apparently, a linear multi-channel filter function, in case it can be split, can be split in both ways.

1.4 Finite training data

In real applications one is provided with a finite amount of data which may be organized in a large N-dimensional array of real numbers. We call this a *dataset* and write

$$\mathcal{X} \in \mathbb{R}^{R_1 \times \ldots \times R_N} \,. \tag{1.41}$$

Thus, the data set $\mathcal{X}$ reflects a finite region of one realization of the stochastic process for one particular $\omega \in \Omega$,

$$\mathcal{X} = x_{\hat{\mathrm{I}}}(\omega) \in \mathrm{E}^{\hat{\mathrm{I}}} \,, \tag{1.42}$$

where $N = N_t + N_r$ and

$$\hat{\mathrm{I}} = [0, R_{i_1} - 1] \times \ldots \times [0, R_{i_{N_t}} - 1] \subset \mathbb{Z}^{N_t} \,, \quad E = \mathbb{R}^{R_{i_{N_t+1}} \times \ldots \times R_{i_{N_r}}} \,.$$

The temporal indices in $\hat{\mathrm{I}}$ may start with zero without loss of generality.

1.4.1 Interpretation of data sets

The main question now is: How is the data in $\mathcal{X}$ organized or what organization would be meaningful? The answer depends on the problem at hand and in many situations one has some idea about the stochastic process that generated the dataset. However, in many other situations nothing is known a priori, and it is not quite obvious how to separate the dimensions into channels and temporal indices. However, there are a few 'rules of thumb':

- Because in statistical signal processing one generally attempts to compute empirical moments of the unknown stochastic process by the truncation of equation (1.21) at sufficiently large n, one would like to treat dimensions with large R_i as temporal dimensions, which assures a large sample basis for $\left\langle g(x_{\mathrm{J}})\right\rangle$. In particular, the extension of $\hat{\mathrm{I}}$ in all its dimensions has to be large compared to the receptive field J since the number of admissible shifts is limited by the fact that the shifted receptive field $J + \tau$ has to be contained in $\hat{\mathrm{I}}$.

- One will usually consider dimensions for which the stationarity assumption cannot be well justified as channel dimensions.

- Dimensions for which a continuation is not meaningful or that have no intrinsic order of the elements (eg. the channels of an EEG measurement) are channel dimensions.

- Of course, there must be at least one temporal dimension. Otherwise the whole dataset $\mathcal{X}$ would represent one single data point of dimension $\prod_{i=1}^{N} R_i$ and considering statistics of a single data point is clearly not meaningful.

Consider the following examples

- $N = 1$, *Time series.* Obviously, in this case one has no other choice than to consider the indices into $\mathcal{X}$, which is a vector, as temporal indices. Whether this time series is the trajectory of an stationary rsp. ergodic process must be decided by other means.

- $N = 2$, *Multi-channel time series vs. image.* A matrix of data can be interpreted in two ways: a time series of vectors, or a two dimensional series of scalar values. However, in most cases this is easy to decide. For an image it is often suitable to think of it as a finite section of an infinitely spread out 2d process. On the other hand, for something like a stereo signal it is rather clear that the dimension that subscribes the channels is short ($R_i = 2$), is not necessarily stationary, and has no meaningful extension (to values $R_i > 2$), and, hence, is no temporal dimension.

- $N = 3$, *Time series of a sensor matrix vs. multi-channel image vs. video vs. 3d-scan.* Things get more ambiguous with $N = 3$, as one can see from the number of possible interpretations of an 3d data set; although, according to the above rules, for many applications it should be quite clear what the data set represents. However, there are cases that can be interpreted in various ways all of which have their meaning. Consider a video with the number of frames being in the same order of magnitude as the frame dimensions. It may be interpreted as a matrix of time series ($N_r = 2$, $N_t = 1$) or a 2d vector field ($N_r = 1$, $N_t = 2$).

- $N \geqslant 4$, *multi-channel video, 3d-sequence ...* With increasing N the interpretation of the data set may get even more ambiguous. However, there are two cases for $N = 4$ which may be of some particular importance. These are multi-channel 2d-sequences (e.g. RGB color videos, stereo videos) for which at least one dimension is obviously a channel dimension, and 3d-sequences (e.g. fMRI sequences) with 3 physical space dimensions and one physical time dimension which can be interpreted in either way.

1.4.2 Considerations to stationarity and ergodicity

For one given dataset $\mathcal{X}$ alone there is no way to detect whether or not the underlying process is ergodic. Ergodicity, however, must be assumed in order to compute empirical moments according to equation (1.21). Thus, in many applications one has to content oneself with the statement that it is 'reasonable to assume ergodicity'.

If, however, a number $R_k > 1$ of datasets $(\mathcal{X}_k)_{k=0\ldots R_k-1}$ is given, then, of course, one can consider them as multiple realizations of the same stochastic process, and check for ergodicity by just comparing the resulting empirical moments, (1.21). But anyway, even if one finds ergodicity, any averaging should not be done solely on shifts over the temporal dimension but also over the R_k datasets. Hence, non-ergodicity can be overcome by multiple realizations of the stochastic process. In particular, in such a situation it would seem to be appropriate to perform cross-validation on the set of trajectories.

On a single dataset it is sometimes meaningful and advantageous to take one of the channel dimensions as different trajectories of the process. This should be done if one has reason to believe that the individual channels along this dimension are iid., i.e.

$$P_{\hat{\mathrm{I}}} = \prod_{k=0}^{R_k-1} P_{\hat{\mathrm{I}}_k}\,, \quad P_{\hat{\mathrm{I}}_{k_1}} = P_{\hat{\mathrm{I}}_{k_2}} \text{ for all } \quad k_1, k_2 \in [0, R_k - 1]\,, \tag{1.43}$$

where $\hat{\mathrm{I}}_k$ is the k-th slice of $\hat{\mathrm{I}}$ along this dimension. An example could be given by medical time series (e.g. EEG) of common length recorded from R_k subjects in individual independent experiments.

There are cases in which the stationarity assumption is not well founded but one has reason to believe that the measures P_{J} change slowly when shifted over I. In such situations one can split $\hat{\mathrm{I}}$ into R_k smaller pieces in which the stationarity assumption is better justified. Correspondingly, $\mathcal{X}$ is divided into R_k realizations of one stochastic process for which now stationarity (but not ergodicity!) can be assumed.

Chapter 2

Instantaneous linear functions

2.1 Instantaneous linear mixtures

For stochastic signals as we have them defined in chapter 1, the most basic form of signal processing is by means of stationary, instantaneous, linear functions. Consider a family of vector valued random variables $(\mathbf{s}_t)_{t\in \mathrm{I}}$, where

$$\mathbf{s} \in \mathrm{E} := \mathbb{R}^N \tag{2.1}$$

and

$$t \in \mathrm{I} := \mathbb{Z}^{N_t} \, . \tag{2.2}$$

We refer to $(\mathbf{s}_t)_{t\in \mathrm{I}}$ as *real valued, stochastic multi-channel signal* together with the associated canonical stochastic process

$$(\mathrm{E}^{\mathrm{I}}, \mathscr{B}(\mathrm{E}^{\mathrm{I}}), P_{\mathrm{I}}, (\mathbf{s}_t)_{t\in \mathrm{I}}) \, . \tag{2.3}$$

In the following $(\mathbf{s}_t)_{t\in \mathrm{I}}$, or shortly $\mathbf{s}$, denotes the whole stochastic process, while $\mathbf{s}_t$ denotes one particular random vector at index t rsp. the value of its realization. The i-th channel of $\mathbf{s}_t$ is itself a stochastic process $(s_{i,t})_{t\in \mathrm{I}}$. Signal processing with stationary, instantaneous, linear functions leads to linear combinations of $s_{i,t}$ rsp. $(s_{i,t})_{t\in \mathrm{I}}$. Be $(\mathbf{x}_t)_{t\in \mathrm{I}}$ the result of these linear combinations: another stochastic multi-channel signal with

$$(\mathrm{E}'^{\mathrm{I}}, \mathscr{B}(\mathrm{E}'^{\mathrm{I}}), P'_{\mathrm{I}}, (\mathbf{x}_t)_{t\in \mathrm{I}}) \tag{2.4}$$

and $\mathrm{E}' = \mathbb{R}^{N'}$. We call $(\mathbf{x}_t)_{t\in \mathrm{I}}$ an *instantaneous, linear mixture of* $(\mathbf{s}_t)_{t\in \mathrm{I}}$ with the *mixing matrix* $\mathbf{A} \in \mathbb{R}^{N' \times N}$ if for all finite marginal events

$$\mathrm{X}_{\mathrm{J}} = \prod_{t\in \mathrm{J}} \mathrm{X}_t \, , \; \mathrm{X}_t \in \mathscr{B}(\mathrm{E}') \tag{2.5}$$

holds

$$P'_{\mathrm{J}}(\mathrm{X}_{\mathrm{J}}) = P_{\mathrm{J}} \left(\{ \mathbf{s}_{\mathrm{J}} \in \mathrm{E}^{\mathrm{J}} : \; \mathbf{A}\mathbf{s}_t \in \mathrm{X}_t \, , \; t \in \mathrm{J} \} \right) , \tag{2.6}$$

where $\mathbf{s}_{\mathrm{J}} := (\mathbf{s}_t)_{t\in \mathrm{J}}$ and P_{I} and P'_{I} are the projective limes of $(P_{\mathrm{J}})_{\mathrm{J}\in \mathscr{H}(\mathrm{I})}$ and $(P'_{\mathrm{J}})_{\mathrm{J}\in \mathscr{H}(\mathrm{I})}$, respectively. We call $(s_{i,t})_{t\in \mathrm{I}}$ the *sources* of the mixture.

Signal processing with instantaneous, linear functions always can be interpreted as a linear mixture of the channels of a stochastic multi-channel signal. In this framework unsupervised learning methods aim on finding optimal instantaneous, linear functions such that either the resulting mixtures exhibit certain 'interesting' statistical properties or such that a previous instantaneous linear function is reverted. These two concepts are often called *projection pursuit* and *source separation*. Both are equivalent if that what is an 'interesting property' is something that is inherent only to the stochastic source signals $\mathbf{s}$. In the following, the focus will on source separation.

2.2 Blind source separation

From now on we assume that marginal densities $p_{\mathrm{J}}(\mathbf{s}_{\mathrm{J}}) \geqslant 0$ exist such that

$$\int_{\mathrm{S_J}} p(\mathbf{s}_{\mathrm{J}})\, d\mathbf{s}_{\mathrm{J}} = P_{\mathrm{J}}(\mathrm{S_J}) \tag{2.7}$$

for all $\mathrm{S_J} \in \mathscr{B}(\mathrm{E}^{\mathrm{J}})$ and $\mathrm{J} \in \mathscr{H}(\mathrm{I})$. The stochastic process (2.4) is connected to the stochastic process (2.3) by the relation

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t \tag{2.8}$$

for all $t \in \mathrm{I}$. In the following we will consider square $N \times N$ matrices $\mathbf{A}$ with full rank. Then for the marginal densities of $\mathbf{x}$ holds

$$p(\mathbf{x}_t) = |\det \mathbf{A}|^{-1} p(\mathbf{s}_t) \quad \text{and} \quad p(\mathbf{x}_{\mathrm{J}}) = |\det \mathbf{A}|^{-|\mathrm{J}|} p(\mathbf{s}_{\mathrm{J}})\,. \tag{2.9}$$

In the most general form we can characterize blind source separation as the task of recovering the original source signal $\mathbf{s}$ only from the observed mixtures $\mathbf{x}$ without knowledge about $\mathbf{A}$. Clearly, if $\mathbf{A}$ was known, the sources could be easily separated by inverting it, yielding source estimates

$$\mathbf{s}_t = \mathbf{y}_t = \mathbf{A}^{-1}\mathbf{x}_t\,. \tag{2.10}$$

However, even if the mixing matrix is completely unknown (except the assumptions about its size and rank), statistical properties of the sources can be used to estimate them together with the mixing matrix "blindly" from the observations alone. Practically an estimate $\mathbf{W}$ for the inverse mixing matrix is searched that allows to compute source estimates

$$\hat{\mathbf{s}}_t = \mathbf{W}\mathbf{x}_t\,.$$

This procedure is called *blind source separation (BSS)*.

2.2.1 Permutation ambiguity

Depending on the available knowledge about the sources and the type of BSS algorithm, the demands on the solution can or must be somewhat relaxed from the true one, $\mathbf{W} = \mathbf{A}^{-1}$:

$$\mathbf{WA} = \mathbf{I} \qquad \Rightarrow \qquad \mathbf{WA} \in \mathscr{D}_N \qquad \Rightarrow \qquad \mathbf{WA} \in \mathscr{Q}_N\,, \tag{2.11}$$

where $\mathscr{Q}_N$ is the set of all non-singular $N \times N$ permutation matrices, and $\mathscr{D}_N \subset \mathscr{Q}_N$ is the set of all non-singular diagonal matrices. Often the order and the scaling of the sources are immaterial, and one is content with the *permutation ambiguity* of a solution $\mathbf{WA} \in \mathscr{Q}_N$. In the following all considerations are made in this respect.

We define the relation "$\equiv$" of two non-singular $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$ to be equal up to permutations as

$$\mathbf{A} \equiv \mathbf{B} \quad \Leftrightarrow \quad \mathbf{A}^{-1}\mathbf{B} \in \mathscr{Q}_N\,. \tag{2.12}$$

Obviously this relation is (i) reflexive, (ii) symmetric, and (iii) transitive and, hence, is an equivalence relation. (i) follows from the fact that the unit matrix is a permutation matrix, (ii) follows from the fact that the inverse of a permutation matrix is a permutation matrix, and (iii) follows from the fact that the product of two permutation matrices is a permutation matrix. Thus, any solution $\mathbf{W} \equiv \mathbf{A}^{-1}$ of the blind source separation problem spans a whole class of equivalent solutions, which we may call $\mathscr{W}$. The neighborhood of $\mathbf{W}$ in $\mathscr{W}$ is given by all possible scalings of rows of $\mathbf{W}$. Thus $\mathscr{W}$ is locally linear. Be $\mathscr{T}_{\mathbf{W}}$ the the tangent space of $\mathscr{W}$ at $\mathbf{W}$. Because $\mathscr{W}$ is locally linear, it locally coincides with its tangent space, which is a N dimensional subspace of $\mathbb{R}^{N^2}$. See figure 2.1 for an illustration of $\mathscr{W}$ and $\mathscr{T}_{\mathbf{W}}$. A basis to $\mathscr{T}_{\mathbf{W}}$ is given by N matrices $(\mathbf{B}_n)_{n=1\ldots N}$ with $B_{n,ij} = \delta_{i,n} W_{ij}$,

$$\mathscr{T}_{\mathbf{W}} = \left\{ \sum_{n=1}^{N} d_n \mathbf{B}_n\,,\ d_n \in \mathbb{R} \right\}\,. \tag{2.13}$$

The core of any BSS algorithm is a suitable objective function $L(\mathbf{W}; (\mathbf{x}_t)_{t\in\mathrm{I}})$. Under the assumption of stationarity and ergodicity, the dependence of L from $(\mathbf{x}_t)_{t\in\mathrm{I}}$ is usually in terms of empirical moments and higher order statistics of $\mathbf{W}\mathbf{x}_t$, which are computed from the finite set of observations (Cardoso, 1998; Hyvarinen et al., 2001).

The objective function be such that under certain assumptions about the sources, $\mathfrak{I}((\mathbf{s}_t)_{t\in\mathrm{I}})$, the gradient of the objective function w.r.t. $\mathbf{W}$ falls into the tangent space if, and only if, $\mathbf{W}$ is a solution of the BSS problem,

$$\mathfrak{I}((\mathbf{s}_t)_{t\in\mathrm{I}}) \quad \Rightarrow \quad \left(\nabla_{\mathbf{W}} L \in \mathscr{T}_{\mathbf{W}} \quad \Leftrightarrow \quad \mathbf{W} \equiv \mathbf{A}^{-1}\right) . \tag{2.14}$$

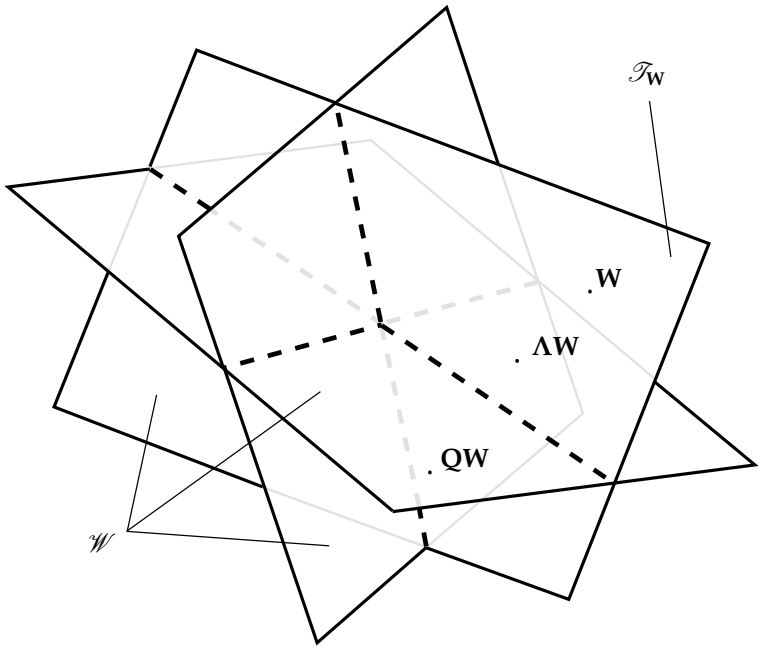


Figure 2.1: Illustration of the space $\mathscr{W} = \{\mathbf{W} : \mathbf{WA} \in \mathscr{Q}_N\}$. Equivalent solutions are any multiplications from the left of $\mathbf{W}$ with a non-singular permutation matrix $\mathbf{Q}$. If a permutation matrix $\mathbf{\Lambda}$ happens to be diagonal, then $\mathbf{\Lambda W}$ does not leave the local tangent space $\mathscr{T}_{\mathbf{W}}$. The intersections of the individual tangent spaces (dashed lines) are not in $\mathscr{W}$ because $\mathbf{W}$ is singular there.

2.2.2 Natural gradient

If $\mathbf{W}$ is a solution for observations $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$, then, of corse, $\mathbf{WA}$ would be a solution for observations $\mathbf{s}_t$, and vice versa. The value of the objective function depends on $\mathbf{W}$ through the recovered sources $\hat{\mathbf{s}}_t = \mathbf{W}\mathbf{x}_t = \mathbf{WA}\mathbf{s}_t$. Thus, one would expect that with $\nabla_{\mathbf{W}} L \in \mathscr{T}_{\mathbf{W}}$ also $\nabla_{\mathbf{WA}} L \in \mathscr{T}_{\mathbf{WA}}$ holds true. However, this is in general not the case, because the gradient $\nabla_{\mathbf{W}} L$, taken for a vector, is a contravariant quantity while the tangent space is covariant. Under a transformation of the coordinate system from $\mathbf{W}$ to $\mathbf{WA}$ it holds

$$\nabla_{\mathbf{WA}} L = \nabla_{\mathbf{W}} L \mathbf{A}^{-T} , \tag{2.15}$$

whereas

$$\mathscr{T}_{\mathbf{WA}} = \mathscr{T}_{\mathbf{W}}\mathbf{A} = \left\{ \sum_{n=1}^{N} d_n \mathbf{B}_n \mathbf{A} , \; d_n \in \mathbb{R} \right\} .$$

Thus, the relation

$$\nabla_{\mathbf{WA}} \in \mathscr{T}_{\mathbf{WA}} \quad \Leftrightarrow \quad (\nabla_{\mathbf{W}} L)\mathbf{A}^{-T}\mathbf{A}^{-1} = (\nabla_{\mathbf{W}} L)\mathbf{W}^T\mathbf{W} \in \mathscr{T}_{\mathbf{W}}$$

gives rise to define

$$\widetilde{\nabla}_{\mathbf{W}} L := (\nabla_{\mathbf{W}} L)\mathbf{W}^T\mathbf{W} \tag{2.16}$$

as a contravariant vector so that (2.14) becomes consistent with transformations of the coordinate system. $\widetilde{\nabla}_{\mathbf{W}} L$ is called the *natural gradient*. It was introduced by Amari (1998) as gradient in Riemannian parameter spaces. From a geometrical perspective $\mathbf{W}^T\mathbf{W}$ defines a contravariant metric tensor that transforms the covariant vector $\nabla_{\mathbf{W}} L$ into the contravariant form $\widetilde{\nabla}_{\mathbf{W}} L$. For problems regarding statistical models, e.g. for L being a likelihood function, the natural gradient was shown to yield Fisher efficient estimators. A very similar concept was independently introduced by Cardoso (1998) with the notion of the *relative gradient*. The relative gradient reflects the variations of an objective function with respect to changes of the outputs. It is given by equation (2.15) for objective functions that depend on $\mathbf{W}$ only through $\mathbf{WA}$. For algorithms that optimize the relative gradient it was shown that they exhibit uniform performance, i.e. the 'hardness' of a problem, and hence the performance of the algorithms, does not depend on the mixing matrix $\mathbf{A}$ anymore. Moreover, as we will see below, it is often difficult or impossible to construct BSS algorithms, i.e. to define $\Im((\mathbf{s}_t)_{t\in\mathrm{I}})$ and L such that (2.14) holds independently from $\mathbf{A}$ when the direct gradient is used rather than the natural gradient.

2.2.3 BSS algorithms

The relation (2.14) represents the most general form to define a BSS algorithm by means of an objective function L together with necessary assumptions $\Im$ so that the right equivalence holds. One can categorize existing BSS algorithms by the type of their objective function. There are those that make use of higher order, instantaneous statistics

$$L(\mathbf{W};\mathbf{x}) = f\left(\left\langle l(\mathbf{W};\mathbf{x}_t)\right\rangle\right) ,$$

for example likelihood functions or higher order cumulants of $\mathbf{W}\mathbf{x}_t$. These algorithms usually work under the general assumption of instantaneous independence[1],

$$\Im(\mathbf{s}) \;=\; \text{“}\, p(\mathbf{s}_t) = \prod_{i=1}^{N} p_i(s_{i,t}) \,\text{”} , \tag{2.17}$$

giving rise to the notion *Independent Component Analysis (ICA)*. However, for a given objective function it is usually difficult, if not impossible, to define the least restrictive $\Im$ for (2.14) to hold true. For example, in many ICA algorithms at most one source is allowed to be normally distributed, hence (2.17) would not be restrictive enough. On the other hand, (2.17) is too restrictive because any ICA algorithm can lead to recovered sources that are not independent.

In the following sections the focus will be on second order BSS algorithms, i.e. those that exclusively make use of second order statistics in the definition of the objective function. Interestingly, it will turn out that, at the same time, $\Im(\mathbf{s})$ may involve higher order statistics.

2.3 Approximate matrix diagonalization for second-order BSS methods

Approximate matrix diagonalization – also called *jojnt diagonalization* – is the problem of simultaneously diagonalizing a given set of square matrices of common size with a single bilinear transformation. Joint diagonalization problems often occur naturally if second order quantities like correlation matrices or second derivatives are a measure for optimality because they transform bilinearly under a linear transformation of the coordinate system. For a given set of matrices $C \subset \mathbb{R}^{N\times N}$ a linear transformation $\mathbf{W}$ is found such that the matrices in the set $\tilde{C} = \left\{\mathbf{W}\mathbf{C}\mathbf{W}^T : \mathbf{C} \in C\right\}$ are as diagonal as possible. Exact joint diagonalizations

[1] where the necessary densities may exist and $\mathbf{s}$ be stationary.

are in general possible only if C contains not more than two matrices. In this case the solution is achieved by an eigenvalue problem (Jolliffe, 1986) rsp. a generalized eigenvalue problem (Molgedey and Schuster, 1994; Pham and Garat, 1997). If C contains more than two matrices, however, then in general a method for approximate joint diagonalization is necessary.

One of the most prominent applications of joint diagonalization is in BSS problems. There are many approaches to construct a suitable set C from the observed signals that allows to achieve source separation by means of joint diagonalization. C can contain delayed correlation matrices (Belouchrani et al., 1997), certain optimal linear combinations of them (Vollgraf et al., 2000; Ziehe et al., 2000), correlation matrices of different epochs of non-stationary signals (Matsuoka et al., 1995; Choi and Cichocki, 2000), or combinations of them. Source separation can also be achieved through the diagonalization of higher order cumulant tensors (Cardoso and Souloumiac, 1993; Cardoso, 1998). It was shown that the diagonalization of the full 4th order cumulant tensor can be achieved efficiently with the diagonalization of the set of cumulant matrices, which are 2d parallel slices through the 4th order tensor (cf. also Wax and Sheinvald (1997)), thus leading to a joint diagonalization problem.

2.3.1 Second order moments

The basic ingredient to all joint diagonalization BSS algorithms are matrices of second order moments. A second order moment of two channels i and j of a multi-channel signal $\mathbf{s}$ is given in the most general form as

$$C_{ij}(\mathbf{s}) = \left\langle f_1(s_{i,\mathrm{J}_1}) f_2(s_{j,\mathrm{J}_2}) \right\rangle - \left\langle f_3(s_{i,\mathrm{J}_3}) \right\rangle \left\langle f_4(s_{j,\mathrm{J}_4}) \right\rangle \,, \tag{2.18}$$

where $f_1 \dots f_4$ are linear functions $f_i : \mathbb{R}^{|\mathrm{J}_i|} \mapsto \mathbb{R}$. $\left\langle \cdot \right\rangle$ denotes the expectation over $p(\mathbf{s}_{\mathrm{J}_1 \cup \mathrm{J}_2 \cup \mathrm{J}_3 \cup \mathrm{J}_4})$. If $\mathbf{s}$ is stationary, then so is $C_{ij}(\mathbf{s})$, i.e. it does not change when all J_i are shifted to $\mathrm{J}_i + t$ for any $t \in \mathrm{I}$. If $\mathbf{s}$ is moreover ergodic, then $C_{ij}(\mathbf{s})$ can be estimated by means of those shifts. It is often convenient to consider for a given configuration $\{f_1, \mathrm{J}_1, \dots, f_4, \mathrm{J}_4\}$ the moments between all pairs of channels summarized in the matrix $\mathbf{C}(\mathbf{s})$.

A nice property of matrices of second order moments is that they transform bilinearly when the signal is linearly transformed according to (2.8),

$$\mathbf{C}(\mathbf{x}) = \mathbf{A}\mathbf{C}(\mathbf{s})\mathbf{A}^T \,, \tag{2.19}$$

and

$$\mathbf{C}(\hat{\mathbf{s}}) = \mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T \,. \tag{2.20}$$

Given that

$$f_1 = f_3,\ \mathrm{J}_1 = \mathrm{J}_3, \quad f_2 = f_4,\ \mathrm{J}_2 = \mathrm{J}_4 \,,$$

equation (2.18) is the second order cross cumulant of $f_1(s_{i,\mathrm{J}_1})$ and $f_2(s_{j,\mathrm{J}_2})$ and is always zero for $i \neq j$ if the source channels are statistically independent.

The basic principle of all second order blind source separation algorithms is to assume in $\mathfrak{I}(\mathbf{s})$ that all second order matrices $\mathbf{C}_k(\mathbf{s})$, for $k = 1 \dots K$ different configurations, are diagonal. We will see below that this assumption can be somewhat relaxed, depending on the objective function. Together with an appropriate objective function L, namely one for which $\nabla_{\mathbf{W}} L \in \mathscr{T}_{\mathbf{W}}$ is equivalent to all $\mathbf{W}\mathbf{C}_k(\mathbf{x})\mathbf{W}^T$ being diagonal, the blind source separation algorithm is almost complete. Almost, because it still must be shown that the K chosen configurations are 'sufficiently non-trivial' to hold the relation (2.14).

Following this principle, a number of second order blind source separation algorithms have been proposed, which can be classified by the used configurations of the second order moments.

Principal Component Analysis (PCA)

In principal component analysis there is only one configuration,

$$f_1 = f_2 = f_3 = f_4 = 1\,, \quad J_1 = J_2 = J_3 = J_4 = \{t\}\,, \tag{2.21}$$

and, hence, one matrix $\mathbf{C}(\mathbf{x})$ to be diagonalized. In conjunction with

$$\mathfrak{I}(\mathbf{s}) = \quad " \left\langle \mathbf{s}_t \mathbf{s}_t^T \right\rangle - \left\langle \mathbf{s}_t \right\rangle \left\langle \mathbf{s}_t \right\rangle^T = c\mathbf{I},\ c \in \mathbb{R}^+ "$$

an objective function

$$L(\mathbf{W}; \mathbf{x}) = \mathrm{tr}\left(\left(\mathbf{W} \left(\left\langle \mathbf{x}_t \mathbf{x}_t^T \right\rangle - \left\langle \mathbf{x}_t \right\rangle \left\langle \mathbf{x}_t \right\rangle^T \right) \mathbf{W}^T \right)^2 \right)$$

constitutes a quasi-BSS algorithm in the respect that

$$\mathfrak{I}(\mathbf{s}) \quad \Rightarrow \quad \left(\tilde{\nabla} L_{\mathbf{W}} \in \mathscr{T}_{\mathbf{W}} \quad \Leftrightarrow \quad \mathbf{WA} = \mathbf{\Lambda U} \right)\,,$$

where $\mathbf{\Lambda}$ is a positive diagonal matrix, and $\mathbf{U}$ is an orthonormal matrix. Thus, the sources can be recovered only up to arbitrary rotations. Therefore, PCA is actually no proper source separation algorithm. In order to avoid the rotation ambiguity at least two configurations of second order matrices are necessary.

The above formulation of PCA was intended to show how PCA fits into the second order BSS framework. In practice one usually doesn't solve PCA via an objective function. The solution can be more easily achieved form an eigen-decomposition of $\mathbf{C}(\mathbf{x})$, where the row vectors of $\mathbf{W}$ are the resulting eigenvectors.

Temporal decorrelation

To avoid the permutation ambiguity, two or more configurations of the second order matrices (2.18) are required. For sources that have pairwise uncorrelated channels, but that have temporal correlations in the individual channels, several matrices $\mathbf{C}_k(\mathbf{s}),\ k = 1 \ldots K$, can be defined as

$$f_1 = f_2 = f_3 = f_4 = 1, \quad \mathrm{J}_1 = \mathrm{J}_3 = \{t\}, \quad \mathrm{J}_2 = \mathrm{J}_4 = \{t + \delta t_k\}\,,$$

where δt_k are appropriately chosen temporal shifts. Molgedey and Schuster (1994) proposed to simultaneously diagonalize two matrices by the solution of the generalized eigenvalue problem imposed by $\mathbf{C}_1(\mathbf{x})$ and $\mathbf{C}_2(\mathbf{x})$. The choice of the shift δt_k is crucial in many cases, in particular in situations where the assumption of zero cross-correlations holds only approximately. However, there is no general strategy how to chose it unless specific knowledge about the source auto-correlations is known. It has been reported several times that the robustness and accuracy of temporal decorrelation based BSS algorithms can be considerably improved when more than two matrices are to be diagonalized approximately (Ziehe and Müller, 1998; Schöner et al., 2000).

If knowledge about the source auto-correlations is known, the temporal decorrelation approach allows to influence the behavior of the BSS algorithm w.r.t. the source signals. If the shifts δt are set exclusively at positions where the auto-correlation function of one source vanishes, this source is virtually ignored in the algorithm. This is of particular interest for white noise corrupted observations. The noise can be considered as additional sources which, however, have no auto-correlations for $\delta t \neq 0$. The proper choice of shifts can also help to focus a second order BSS algorithm that separates a single source (cf. section 2.5) to the desired one.

Convolutive blind source separation

If one interprets a shifted signal as the special case of the convolution with a delta function, then the idea arises to consider the correlations of arbitrarily convolved signals for second

order BSS (Vollgraf et al., 2000). In the framework of equation (2.18) the second order moments of convolved source channels are avchieved with configurations

$$f_1 = f_3 = \bar{f}_k, \quad f_2 = f_4 = \bar{g}_k, \quad \mathrm{J}_1 = \mathrm{J}_2 = \mathrm{J}_3 = \mathrm{J}_2 = \mathrm{J}_k \,.$$

Without loss of generality the convolution kernels $\bar{f}_k, \bar{g}_k \in \mathbb{R}^{|\mathrm{J}_k|}$ may have the same receptive field J_k. With $\bar{f}_k = \bar{g}_k$ symmetric matrices $\mathbf{C}_k(\mathbf{x})$ can be guaranteed, which is important for some diagonalization algorithms. Similar to the choice of shifts in the temporal decorrelation approach, in the convolutive decorrelation approach the choice of the convolution kernels is crucial for the accuracy of the BSS algorithm. However, if knowledge about the source auto-correlations is known, the algorithm can be influenced to a higher degree by the modification of general convolution kernels than by the modification of simple shifts.

Non-stationarities

Besides temporal correlations second order BSS algorithms can also make use of non-stationarities (Matsuoka et al., 1995; Choi and Cichocki, 2000). Under this assumption correlation matrices

$$\mathbf{C}_k(\mathbf{s}) := \left\langle \mathbf{s}_{t_k} \mathbf{s}_{t_k}^T \right\rangle - \left\langle \mathbf{s}_{t_k} \right\rangle \left\langle \mathbf{s}_{t_k} \right\rangle^T$$

can be defined, for example. These would reflect the non-stationarities of the variance of $\mathbf{s}$ at different time t_k. Clearly, because of the non-stationarity also ergodicity is not provided. Hence, any empirical expectations must be computed over multiple realizations of the same stochastic process. Alternatively, if not sufficiently many independent realizations are available, it can be sometimes justified to assume the non-stationary variations to be slow and to compute empirical expectations over k disjunctive and sufficiently small subsets of I.

2.3.2 Approximate matrix diagonalization

The assumption that sources are uncorrelated and, hence, are independent may be inappropriate in some problems. Often it is rather the case that they are weakly correlated, but over a wide range of temporal lags τ, for example. Thus, it may be impossible to find any two matrices of second order moments that are truly diagonal in the sources and that allow to estimate $\mathbf{W}$ from the generalized eigenvalue problem. This gave rise to the idea to take a large number of matrices $(\mathbf{C}_k)_{k=1\ldots K}$, which transform according to equations (2.19) and (2.20), and which are approximately diagonal in the sources and 'less diagonal' in the mixtures in terms of a suitable diagonalization measure. The hope is that this measure constitutes a valid objective function.

In the following two possible measures for approximate diagonality will be presented, and the necessary source assumptions $\Im(\mathbf{s})$ will be derived under which these measures constitute a objective function according to (2.14). The later of both measures was implemented in the powerful *QDIAG* algorithm (Vollgraf and Obermayer, 2006b), which will be described in length in section 2.4.

Diagonality measure involving the determinant

This is a measure for approximate diagonality that is applicable for positive definite matrices $(\mathbf{C}_k)_{k=1\ldots K}$. According to Hadamard's inequality, for any positive semi-definite $N \times N$ matrix $\mathbf{A}$ holds

$$\det \mathbf{A} \leqslant \prod_{i=1}^{N} A_{ii} \,. \tag{2.22}$$

If at the same time $\mathbf{A}$ is positive definite, then equality is attained if, and only if, $\mathbf{A}$ is diagonal. This gives rise to establish a measure

$$L(\mathbf{x}; \mathbf{W}) = \left\langle \det(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T) - \pi(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T) \right\rangle_{\mathbf{C}} \,, \tag{2.23}$$

where $\pi(\cdot)$ is the product of the diagonal elements of it's argument, and $\left\langle \cdot \right\rangle_{\mathbf{C}}$ denotes the average of the terms inside the bracket over all $(\mathbf{C}_k)_{k=1\ldots K}$.

It remains to show under what conditions $\Im(\mathbf{s})$ equation (2.23) is a valid source objective function according to the relation (2.14). It will turn out that with the gradient $\nabla_{\mathbf{W}} L$ equation (2.23) is no source objective function unless $\Im(\mathbf{s})$ states that all matrices $\mathbf{C}(\mathbf{s})_k$ are diagonal. To use equation (2.23) as a source objective function even for approximate diagonal matrices $\mathbf{C}_k(\mathbf{s})$, the natural gradient (Amari, 1996; S.Amari and H.Nagaoka, 2001),

$$\tilde{\nabla}_{\mathbf{W}} L = (\nabla_{\mathbf{W}} L)\mathbf{W}^T\mathbf{W}\,, \tag{2.24}$$

has to be taken instead. We need to show that

$$\tilde{\nabla}_{\mathbf{W}} L \in \mathscr{T}_{\mathbf{W}} \quad \Leftrightarrow \quad \mathbf{W} \equiv \mathbf{A}^{-1}\,, \tag{2.25}$$

and all assumptions about $\mathbf{C}_k(\mathbf{s})$ that are therefore necessary constitute $\Im(\mathbf{s})$. We decompose (2.23) into

$$\begin{aligned} L_1 &:= \det(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T) = \det(\mathbf{Q})^2 \det(\mathbf{C}(\mathbf{s}))\,, & (2.26)\\ L_2 &:= \pi(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T) = \pi(\mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q}^T) & (2.27) \end{aligned}$$

so that $L = \left\langle L_1 - L_2 \right\rangle_{\mathbf{C}}$, where and $\mathbf{Q} = \mathbf{W}\mathbf{A}$. Consider now the natural gradient

$$\left\langle \tilde{\nabla}_{\mathbf{W}} L_1 \right\rangle_{\mathbf{C}} = 2\det(\mathbf{Q})^2 \left\langle \det(\mathbf{C}(\mathbf{s})) \right\rangle_{\mathbf{C}} \mathbf{W}. \tag{2.28}$$

This is $\mathbf{W}$ multiplied with a scalar and, hence, always element of $\mathscr{T}_{\mathbf{W}}$ regardless of $\mathbf{C}(\mathbf{s})$. Under the assumption that $\mathbf{Q} = \mathbf{W}\mathbf{A} \in \mathscr{Q}_N$ we derive the natural gradient

$$\begin{aligned} \left\langle \tilde{\nabla}_{\mathbf{W}} L_2 \right\rangle_{\mathbf{C}} &= \left\langle 2\, L_2\, \mathrm{dg}(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T)^{-1}\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T \right\rangle_{\mathbf{C}} \mathbf{W} & (2.29)\\ &= 2\,\pi(\mathbf{Q}\mathbf{Q}^T)\,(\mathbf{Q}\mathbf{Q}^T)^{-1}\mathbf{Q} \left\langle \pi(\mathbf{C}(\mathbf{s}))\,\mathrm{dg}(\mathbf{C}(\mathbf{s}))^{-1}\,\mathbf{C}(\mathbf{s}) \right\rangle_{\mathbf{C}} \mathbf{Q}^T\,\mathbf{W}\,. \end{aligned}$$

This is $\mathbf{W}$ multiplied from the left with a diagonal matrix and, hence, element of $\mathscr{T}_{\mathbf{W}}$ whenever $\left\langle \pi(\mathbf{C}(\mathbf{s}))\,\mathrm{dg}(\mathbf{C}(\mathbf{s}))^{-1}\,\mathbf{C}(\mathbf{s}) \right\rangle_{\mathbf{C}}$ is a diagonal matrix. In the above equation we further made use of the fact that the matrices $\mathbf{C}(\mathbf{s})_k$ are positive definite and symmetric. Thus, the assumptions that have to be made for $\Im(\mathbf{s})$ are

$$\Im(\mathbf{s}) \;=\; \text{“}\quad \left\langle \pi(\mathbf{C}(\mathbf{s}))\,\mathrm{dg}(\mathbf{C}(\mathbf{s}))^{-1}\,\mathbf{C}(\mathbf{s}) \right\rangle_{\mathbf{C}} = \textit{diag.} \quad \text{and} \quad \mathbf{C}_k(\mathbf{s}) = \textit{pos. def.} \quad \text{”}\,. \tag{2.30}$$

If the sources fulfill $\Im(\mathbf{s})$, then it holds

$$\mathbf{W} \equiv \mathbf{A}^{-1} \quad \Rightarrow \quad \tilde{\nabla}_{\mathbf{W}} L = \left\langle \frac{\partial L_1}{\partial \mathbf{W}} - \frac{\partial L_2}{\partial \mathbf{W}} \right\rangle_{\mathbf{C}} \mathbf{W}^T\mathbf{W} = \mathbf{\Lambda}\mathbf{W}\,, \tag{2.31}$$

where $\mathbf{\Lambda}$ is a diagonal matrix. For any solution of the blind source separation problem the natural gradient necessarily falls into the tangent space $\mathscr{T}_{\mathbf{W}}$. Unfortunately, sufficiency, i.e. the reversal of the implication (2.31), could not be shown by the above derivation. So, there still may exist spurious attractors or other stationary points of L for which the left hand side of (2.25) holds, but not the right. Showing sufficiency appears to be a difficult task. It would require to prove

$$\left\langle \pi(\mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q})\,\mathrm{dg}(\mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q}^T)^{-1}\,\mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q} \right\rangle_{\mathbf{C}} \text{ is diagonal} \quad \Rightarrow \quad \mathbf{Q} \in \mathscr{Q}_N\,,$$

which possibly needs further restrictions to $\Im(\mathbf{s})$ to be made. On the other hand one should mention that for most machine learning methods the existence of spurious local minimum rsp. maximum solutions cannot be excluded.

Least squares measure

The measure is given by the average squared sum of all off-diagonal elements of the diagonalized matrices,

$$L(\mathbf{W};\mathbf{x}) = \left\langle \sum_i \sum_{j\neq i} \left(\sum_k \sum_l W_{ik} W_{jl} C_{kl}(x) \right)^2 \right\rangle_{\mathbf{C}} . \tag{2.32}$$

A more compact expression is achieved in matrix notation:

$$L(\mathbf{W};\mathbf{x}) = \left\langle tr\left(\overline{\mathrm{dg}}(\mathbf{W}\mathbf{C}^T\mathbf{W}^T)\, \mathbf{W}\mathbf{C}^T\mathbf{W}^T \right) \right\rangle_{\mathbf{C}} . \tag{2.33}$$

As well as the diagonality measure (2.23) also the least squares measure is an objective function in conjunction with the natural gradient (2.24). It remains to show what conditions $\Im(\mathbf{s})$ must be fulfilled so that at a solution of the blind source separation problem the natural gradient falls into $\mathscr{T}_{\mathbf{W}}$. Under the assumption that $\mathbf{Q} = \mathbf{W}\mathbf{A} \in \mathscr{Q}_N$ is a permutation matrix and the fact that

$$\mathbf{Q} \in \mathscr{Q}_N \quad \Rightarrow \quad \overline{\mathrm{dg}}(\mathbf{Q}\mathbf{C}\mathbf{Q}^T) = \mathbf{Q}\overline{\mathrm{dg}}(\mathbf{C})\mathbf{Q}^T ,$$

the natural gradient is given by

$$\begin{aligned} \widetilde{\nabla}_{\mathbf{W}} L &= 4\Big\langle \left(\overline{\mathrm{dg}}(\mathbf{W}\mathbf{C}(\mathbf{x})\mathbf{W}^T)\right) \mathbf{W}\mathbf{C}^T(\mathbf{x}) \;+ \left(\overline{\mathrm{dg}}(\mathbf{W}\mathbf{C}^T(\mathbf{x})\mathbf{W}^T)\right) \mathbf{W}\mathbf{C}(\mathbf{x}) \Big\rangle_{\mathbf{C}} \mathbf{W}^T\mathbf{W} \\ &= 4\Big\langle \left(\overline{\mathrm{dg}}(\mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q}^T)\right) \mathbf{Q}\mathbf{C}^T(\mathbf{s})\mathbf{Q}^T \;+ \left(\overline{\mathrm{dg}}(\mathbf{Q}\mathbf{C}^T(\mathbf{s})\mathbf{Q}^T)\right) \mathbf{Q}\mathbf{C}(\mathbf{s})\mathbf{Q}^T \Big\rangle_{\mathbf{C}} \mathbf{W} \\ &= 4\mathbf{Q}\Big\langle \overline{\mathrm{dg}}(\mathbf{C}(\mathbf{s}))\, \mathbf{Q}^T\mathbf{Q}\, \mathbf{C}^T(\mathbf{s}) \;+ \overline{\mathrm{dg}}(\mathbf{C}^T(\mathbf{s}))\, \mathbf{Q}^T\mathbf{Q}\, \mathbf{C}(\mathbf{s}) \Big\rangle_{\mathbf{C}} \mathbf{Q}^T\mathbf{W} . \end{aligned} \tag{2.34}$$

The terms left of $\mathbf{W}$ must yield a diagonal matrix for any $\mathbf{Q} \in \mathscr{Q}_N$. However, the innermost term $\mathbf{Q}^T\mathbf{Q}$ cannot be eliminated and remains in

$$\Big\langle \overline{\mathrm{dg}}(\mathbf{C}(\mathbf{s}))\, \mathbf{Q}^T\mathbf{Q}\, \mathbf{C}^T(\mathbf{s}) + \overline{\mathrm{dg}}(\mathbf{C}^T(\mathbf{s}))\, \mathbf{Q}^T\mathbf{Q}\, \mathbf{C}(\mathbf{s}) \Big\rangle_{\mathbf{C}} = diag. \tag{2.35}$$

This depends on $\mathbf{Q}$ and, hence, is no general assumption about the sources regardless of $\mathbf{A}$. If, however, a matrix diagonalization algorithm restricts the set of admissible solutions to those that yield constant $\mathbf{Q}^T\mathbf{Q}$, then the source assumptions can be established independently from the actual mixture and separation. For example, $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ can be guaranteed when there is a matrix $\mathbf{C}_0(\mathbf{s})$ with diagonal elements equal to one, and $\mathbf{W}$ is constrained to yield $\mathbf{W}\mathbf{C}_0(\mathbf{x})\mathbf{W}^T = \mathbf{Q}\mathbf{C}_0(\mathbf{s})\mathbf{Q}^T$ with unit diagonal elements as well. This is exactly what the QDIAG algorithms described in the next section does. There, the matrix $\mathbf{C}_0(\mathbf{s})$, which usually represents the variance of the sources, is used to define the column norm of $\mathbf{Q}$.

Thus, we could show that

$$\Im(\mathbf{s}) \;=\; \text{“ } \Big\langle \overline{\mathrm{dg}}(\mathbf{C}(\mathbf{s}))\, \mathbf{C}^T(\mathbf{s}) + \overline{\mathrm{dg}}(\mathbf{C}^T(\mathbf{s}))\, \mathbf{C}(\mathbf{s}) \Big\rangle_{\mathbf{C}} = diag. \text{”} \tag{2.36}$$

is a sufficient condition for an appropriate matrix diagonalization algorithm to have a stationary point at the true solution (the right-to-left implication of (2.25)). This is obviously a weaker condition than the special case that all $\mathbf{C}(\mathbf{s})$ are diagonal, which is included because the empty matrix is a diagonal matrix. However, like for the determinant measure (2.23) also for the least squares measure the left-to-right part of (2.25) could not be shown, so spurious solutions cannot be excluded to this end.

2.4 The QDIAG algorithm

In the following a new algorithm for approximate matrix diagonaliztion, which is called QDIAG, is presented. It splits the overall optimization problem into a sequence of simpler second order sub-problems. There are no restrictions imposed on the transformation matrix, which may be non-orthogonal, indefinite, or even rectangular, and there are no restrictions,

except for one, imposed on the matrices to be diagonalized, regarding their symmetry or definiteness. In section 2.4.4 the new method will be applied to second order BSS. It will be shown that the algorithm convergences fast and reliably. It allows for an implementation with a complexity independent of the number of matrices and, therefore, is particularly suitable for problems dealing with large sets of matrices.

2.4.1 Introduction

There have been proposed many joint diagonalization algorithms in the past (Cardoso et al., 1996; Vollgraf et al., 2000; Ziehe et al., 2000; Pham, 2001; van der Veen, 2001; Joho and Rahbar, 2002; Yeredor, 2002; Ziehe et al., 2004: 2003). They differ in how the actual optimization problem is solved and what restrictions to $\mathbf{W}$ and the matrices in C this implies. For some algorithms, in particular those that use Jacobi angles for optimization, $\mathbf{W}$ is restricted to be orthogonal or unitary (Cardoso et al., 1996; Belouchrani et al., 1997). In BSS applications, however, the assumption of orthogonal mixing and de-mixing matrices can often be inappropriate. Clearly, a general joint diagonalization problem can be reduced to an orthogonal problem by sphering w.r.t. some matrix $\mathbf{C}_0$. As a result, however, this matrix is always perfectly diagonalized, which may be a too strong assumption and inappropriate for approximate diagonalizations. In such situations methods that work with non-orthogonal matrices should be preferred.

Joint diagonalization algorithms can be also categorized by whether they require definiteness of the matrices in C. Positive definite matrices, for example, allow for interesting information theoretic measures of diagonality (Pham, 2001). In some applications (e.g. Matsuoka et al. (1995); Pham (2001)) definiteness is given by construction of C. However, in other cases (e.g. in de-correlation based BSS) positive definiteness cannot be assumed in general (Belouchrani et al., 1997; Vollgraf et al., 2000). Thus, algorithms that are able to diagonalize indefinite matrices are desirable.

For joint diagonalization algorithms that are based on a least squares measure of diagonality, equation (2.33), it is necessary to restrict the set of admissible solutions in order to avoid the trivial or singular solutions. The way these constraints are implemented is crucial for the outcome of the joint diagonalization, in particular when only an approximate diagonalization is achievable. As mentioned above, constraining $\mathbf{W}$ to unitary matrices may be too tight, and bias is introduced by sphering. A less specific constraint is achieved when $\mathbf{W}$ is kept invertible (Ziehe et al., 2003: 2004). However, the resulting matrices can become almost singular, which can lead to 'unbalanced' results, where certain rows and columns of the diagonalized matrices are implicitly weighted much less than others. In section 2.4.4 an example of the problems that can arise with unbalanced solutions will be shown and discussed. A more 'goal oriented' constraint is achieved when the the rows $\mathbf{W}$ are fixed to unit quadratic norm wrt. some positive definite matrix $\mathbf{C}_0(\mathbf{x})$. In a BSS application, for example, one would chose $\mathbf{C}_0(\mathbf{x})$ to be the covariance matrix of the observations, thus leading to source estimates with unit variance and a well balanced diagonalization. Note that, in contrast to sphering, $\mathbf{C}_0(\mathbf{x})$ is not necessarily diagonalized perfectly and may even be excluded from C.

An interesting property of a joint diagonalization algorithm is whether or not it is invariant under invertible, affine transformations of the coordinate system. For many applications it is desirable that, with some contra-variant, bilinear transformation of all $\mathbf{C}_k(\mathbf{x})$, the rows of a BSS solution $\mathbf{W}_{opt}$ undergo the same but covariant transformation (McCullagh, 1987). This holds for algorithms according to (2.33) as long as $\mathbf{W}$ stays in $\mathbf{W}_{adm}$,

$$\mathcal{W}_{adm} = \left\{ \mathbf{W} : \left(\mathbf{W}\mathbf{C}_0(\mathbf{x})\mathbf{W}^T \right)_{ii} = 1 \right\} , \tag{2.37}$$

after the transformation, which is clearly the case for the constraint (2.37). One can easily verify that affine invariance is not provided for algorithms like those of van der Veen (2001) and Yeredor (2002), which solve the inverse problem

$$\mathbf{A}_{opt} = \arg \min_{\mathbf{A}, \{\Lambda^{(k)}\}} \sum_k \left\| \mathbf{C}_k(\mathbf{x}) - \mathbf{A}^T \Lambda^{(k)} \mathbf{A} \right\|_F^2 \tag{2.38}$$

w.r.t. to the matrix $\mathbf{A}$ and a set of diagonal matrices $\Lambda^{(k)}$, and then set $\mathbf{W}_{opt} = \mathbf{A}_{opt}^{-1}$.

In the following we will develop the *Quadratic Diagonalization (QDIAG)* algorithm, which efficiently solves the joint diagonalization problem given by equations (2.33) and (2.37). Except for $\mathbf{C}_0$, which is used to establish the constraint (2.37), QDIAG does not make any assumptions on the definiteness or symmetry of the matrices in C. Besides (2.37), no further restrictions are imposed on $\mathbf{W}$. In fact, $\mathbf{W} \in \mathbb{R}^{M\times N}$ does not even have to be square, in which case undercomplete diagonalizations, $(M < N)$, are certainly of more practical interest. They can offer a performance gain if the algorithm is able to optimize few rows of $\mathbf{W}$ instead of searching the whole matrix and then discarding $N - M$ rows.

A performance comparison between QDIAG and two other methods, ACDC (Yeredor, 2002) and FFDIAG (Ziehe et al., 2003: 2004), will be given because, in terms of the above mentioned criteria, these are the two methods most closely related to QDIAG. Both algorithms do not restrict the definiteness of the matrices in C and allow for non-orthogonal diagonalization matrices. They differ from QDIAG, however, in that ACDC is not invariant under affine transformations and FFDIAG requires square matrices $\mathbf{W}$.

2.4.2 Derivation of the QDIAG algorithm

In order to find a matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ that diagonalizes the set of the matrices $C = \{\mathbf{C}_k\}$ in a least squares sense we consider a cost function given by the squared sum of all off-diagonal elements of $\mathbf{W}\mathbf{C}_k\mathbf{W}^T$,

$$L(\mathbf{W}) = \sum_k \alpha_k \sum_i \sum_{j\neq i} (\mathbf{W}\mathbf{C}_k\mathbf{W}^T)_{ij}^2 = \sum_k \alpha_k \sum_i \sum_{j\neq i} \left(\mathbf{w}_i^T \mathbf{C}_k \mathbf{w}_j\right)^2 . \tag{2.39}$$

$\mathbf{w}_i$ is a vector containing the elements of the i-th row of $\mathbf{W}$, and $\sum_k \alpha_k = 1$ are factors that weight the importance of the individual matrices $\mathbf{C}_k$. Here and whenever there are no ambiguities, in the following $\mathbf{C}_k$ stands for $\mathbf{C}_k(\mathbf{x})$, for convenience. Also explicit indication of the range of the summation indices will be omitted. They are always determined by the indexed variables (i.e. $i, j = 1 \ldots N$, $k = 0 \ldots K$).

In order to prevent $\mathbf{W}$ from converging to the trivial solution $\mathbf{W} = \mathbf{0}$, we use constraint (2.37) for some positive definite matrix $\mathbf{C}_0$. In a BSS framework, for example, $\mathbf{C}_0$ could be the covariance matrix of the observations, thus leading to source estimates of unit variance. We then obtain the optimization problem with objective function (2.39):

$$\textbf{minimize} \quad L(\mathbf{W}) \qquad \textbf{subject to} \quad \mathbf{w}_i^T \mathbf{C}_0 \mathbf{w}_i = 1 \tag{2.40}$$

for all $i = 1, \ldots N$. The matrix $\mathbf{W}$ occurs to the 4th power in the cost function, which makes minimization a rather complex task. However, equation (2.39) is not a full 4th order problem because every row vector of $\mathbf{W}$ appears only quadraticly in $L(\mathbf{W})$, and the 4th order terms are always mixed terms between two different row vectors. Therefore, $L(\mathbf{W})$ can be minimized for one $\mathbf{w}_i$ at a time, which reduces the 4th order problem to a series of quadratic subproblems. Equation (2.39) can be written in the form

$$L(\mathbf{W}) = \sum_i \sum_{j\neq i} D_{ij} , \tag{2.41}$$

where

$$D_{ij} := \sum_k \alpha_k \left(\mathbf{w}_i^T \mathbf{C}_k \mathbf{w}_j\right)^2 . \tag{2.42}$$

The elements of $\mathbf{D}$ that are influenced by $\mathbf{w}_i$ are those in the i-th row and in the i-th column.

They can be summarized by the quadratic function

$$\begin{aligned} l(\mathbf{w}_i) &= \sum_{j\neq i} \left(D_{ij} + D_{ji}\right) \\ &= \sum_k \alpha_k \sum_{j\neq i} \left(\mathbf{w}_i^T \mathbf{C}_k \mathbf{w}_j \mathbf{w}_j^T \mathbf{C}_k^T \mathbf{w}_i \; + \mathbf{w}_i^T \mathbf{C}_k^T \mathbf{w}_j \mathbf{w}_j^T \mathbf{C}_k \mathbf{w}_i\right) \\ &= \mathbf{w}_i^T \left(\sum_k \alpha_k \left(\mathbf{C}_k \left(\sum_{j\neq i} \mathbf{w}_j \mathbf{w}_j^T\right) \mathbf{C}_k^T \; + \; \mathbf{C}_k^T \left(\sum_{j\neq i} \mathbf{w}_j \mathbf{w}_j^T\right) \mathbf{C}_k\right)\right) \mathbf{w}_i \\ &=_{def.} \quad \mathbf{w}_i^T \mathbf{M} \mathbf{w}_i \; . \end{aligned} \tag{2.43}$$

Note that till now we made no assumption about the symmetry of $\mathbf{C}_k$, so that generally $D_{ij} \neq D_{ji}$, and both terms have to be considered. We can now minimize $L(\mathbf{W})$ by sequentially solving the quadratic constrained minimization problems

$$\textbf{minimize} \quad l(\mathbf{w}_i) \qquad \textbf{subject to} \quad \mathbf{w}_i^T \mathbf{C}_0 \mathbf{w}_i = 1 \tag{2.44}$$

for one $\mathbf{w}_i$ after another. For every iteration step the cost function

$$L(\mathbf{W}) = l(\mathbf{w}_i) + \sum_{j\neq i} \sum_{l\neq i} D_{jl} \tag{2.45}$$

is decreased by the amount for which $l(\mathbf{w}_i)$ is reduced. Thus, the sequence of minimization steps eventually converges to a value L_{min} which is a local minimum of (2.40) w.r.t. all $\mathbf{w}_i$ individually. Note that w.r.t. $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)^T$ L_{min} is not necessarily a local minimum, but could be a saddle point instead. The experiments, however, did not give empirical evidence for a convergence to saddle points[2], hence this case is probably rare.

For every iteration, we now have to solve a quadratic optimization problem of the form $\mathbf{w}^T\mathbf{M}\mathbf{w}$ under the quadratic constraint $\mathbf{w}^T\mathbf{C}_0\mathbf{w} = 1$. $\mathbf{M}$ is positive semi-definite by construction (see equation (2.43)), and $\mathbf{C}_0$ is positive definite by definition. The situation is illustrated in figure 2.2.A for a two dimensional example. For positive definite $\mathbf{M}$ the cost function is strictly convex (cf. the elliptic iso-lines). For positive definite $\mathbf{C}$ constraints are fulfilled for all points on a single ellipse (bold line). The optimization problem (2.44) can be transformed into an equivalent problem through a coordinate transformation $\mathbf{w}' = \mathbf{P}^{-1}\mathbf{w}$ such that

$$\mathbf{P}^T \mathbf{C}_0 \mathbf{P} = \mathbf{I} \; , \tag{2.46}$$

where $\mathbf{P}$ can, for example, be calculated by a Principle Component Analysis of $\mathbf{C}_0$. The cost function then becomes

$$l = \mathbf{w}^T \mathbf{M} \mathbf{w} = \mathbf{w}'^T (\mathbf{P}^T \mathbf{M} \mathbf{P}) \mathbf{w}' \; , \tag{2.47}$$

and the constraint holds for all points on the hypersphere $\mathbf{w}'^T\mathbf{w}' = 1$, as shown in figure 2.2.B. The vector $\mathbf{w}'$ with unit norm that minimizes l is given by the eigenvector of $\mathbf{P}^T\mathbf{M}\mathbf{P}$ corresponding to the smallest eigenvalue (bold arrow). It is unique as long as the smallest eigenvalue is unique. The back transformation $\mathbf{w} = \mathbf{P}\mathbf{w}'$ yields the solution of the primary optimization problem.

It is rather inefficient to perform a coordinate transformation for every iteration. Instead, the whole problem can be transformed once at the beginning,

$$\mathbf{C}_0 \quad \longleftarrow \quad \mathbf{P}^T \mathbf{C}_0 \mathbf{P} = \mathbf{I} \; , \tag{2.48}$$

$$\mathbf{C}_k \quad \longleftarrow \quad \mathbf{P}^T \mathbf{C}_k \mathbf{P} \; , \tag{2.49}$$

and minimization is performed under the constraint

$$\mathbf{w}_i^T \mathbf{w}_i = 1 \qquad \forall i = 1 \ldots N \; . \tag{2.50}$$

After convergence the desired solution is obtained via

$$\mathbf{W} \quad \longleftarrow \quad \mathbf{W}\mathbf{P}^T \; . \tag{2.51}$$

[2] $\mathbf{W}$ is a saddle point solution if there is a unitary matrix $\mathbf{U}$ so that for the transformed joint diagonalization problem ($\mathbf{W}' = \mathbf{W}\mathbf{U}^T$, $\mathbf{C}_k' = \mathbf{U}^T\mathbf{C}_k\mathbf{U}$) $l(\mathbf{w}_i')$ can be further decreased for some $\mathbf{w}_i'$.

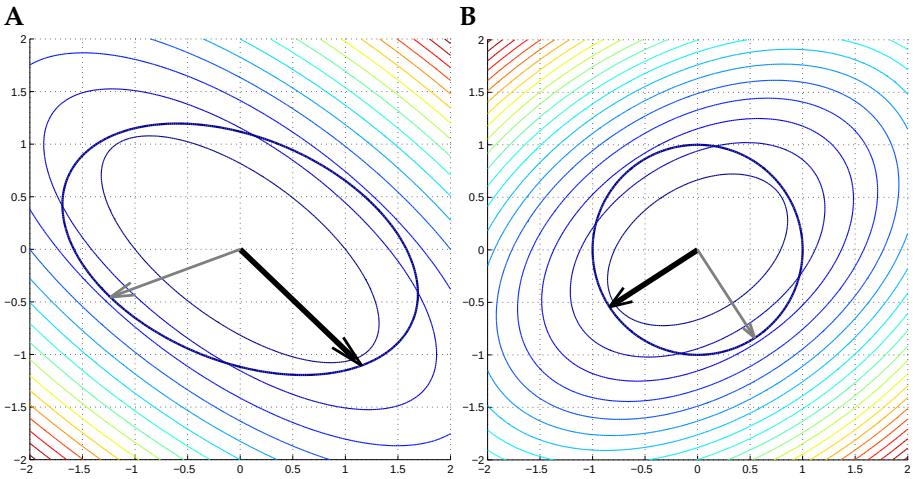


Figure 2.2: Illustration of the quadratic optimization problem (2.44) with quadratic constraints. Iso-cost contours are indicated by the thin lines. The 'thick' ellipse indicates the line on which the constraint is fulfilled. The figure shows the situation before (**A**) and after (**B**) sphering on the matrix $\mathbf{C}_0$. After sphering the constraint is fulfilled for all points lying on a unit hypersphere. Arrows show the eigenvectors of the quadratic cost function after sphering (**B**) and their back projection (**A**). The solution to the constrained optimization problem is given by the point on the constraint line in direction of the eigenvector with the smallest eigenvalue (bold arrow).

2.4.3 Computational complexity of QDIAG

In an efficient implementation, the smallest eigenvector of a $N \times N$ matrix can be calculated with a complexity of $O(N^2)$. In every iteration of QDIAG, this has to be repeated for N row vectors of $\mathbf{W}$, thus leading to $O(N^3)$. However, the computation of the matrix $\mathbf{M}$ involves K iterations (cf. (2.43) and (2.47)), each of which has a complexity of at least $O(N^2)$. Although the complexity of the eigenvector computation has a higher impact than the computation of $\mathbf{M}$, in the limit of large numbers of matrices in C, the complexity of QDIAG is $O(KN^3)$. Algorithm 1 shows pseudocode of an $O(KN^3)$ implementation of QDIAG. If the matrices in C are explicitly symmetrized or symmetric by construction, the steps which involve computations with $\mathbf{m}_2$ and $\mathbf{M}_2$ are obsolete and the computational load approximately reduces by a factor of 2.

QDIAG also allows an implementation with a complexity of $O(N^5)$ per iteration. Although the power of N has raised by 2, the complexity does no longer depend on the number of matrices K. This implementation is useful for joint diagonalization problems with a large number of small matrices. The influence of K on the complexity per iteration can be eliminated by rearranging the summation in (2.43) such that the sum over k stands in the innermost loop and does not depend on $\mathbf{w}_i$. Thus, all $\mathbf{C}_k$ can be collected in a 4-d array, which can be computed prior to the start of the main loop. This is shown in pseudo code in Algorithm 2, which has complexity $O(N^5)$. The operators $vec()$ and $mat_{N\times N}()$ rearrange the elements of their argument into a vector and into an $N \times N$ matrix, respectivly. Both, of course, must follow the same storage order.

2.4.4 Numerical experiments

Description of the datasets

In order to evaluate the performance of QDIAG and to provide benchmark results, experiments with the following three datasets were performed:

1. **Fully diagonalizable data.** A set $C(\mathbf{s})$ of $K+1$ diagonal matrices $\mathbf{C}_k(\mathbf{s})$ was constructed. The diagonal elements where drawn iid from the standard normal distribution for

Algorithm 1 The QDIAG-algorithm with complexity $O(KN^3)$

function QDIAG($\mathbf{C}_0$, $\{\mathbf{C}_k\}$)
  compute $\mathbf{P}$ so that $\mathbf{P}^T\mathbf{C}_0\mathbf{P} = \mathbf{I}$
  for $k = 1 \ldots K$ **do**
    $\mathbf{C}_k \longleftarrow \mathbf{P}^T\mathbf{C}_k\mathbf{P}$
  end for
  initialize W_{ij} randomly according to $\mathcal{N}(0;1)$
  $\mathbf{M} \longleftarrow 0^{N\times N}$
  for $k = 1 \ldots K$ **do**
    $\mathbf{M}_1 \longleftarrow \mathbf{C}_k\mathbf{W}^T$
    $\mathbf{M}_2 \longleftarrow \mathbf{C}_k^T\mathbf{W}^T$
    $\mathbf{M} \longleftarrow \mathbf{M} + \alpha_k \left(\mathbf{M}_1\mathbf{M}_1^T + \mathbf{M}_2\mathbf{M}_2^T\right)$
  end for
  repeat
    for $i = 1 \ldots N$ **do**
      for $k = 1 \ldots K$ **do**
        $\mathbf{m}_1 \longleftarrow \mathbf{C}_k\mathbf{w}_i$
        $\mathbf{m}_2 \longleftarrow \mathbf{C}_k^T\mathbf{w}_i$
        $\mathbf{M} \longleftarrow \mathbf{M} - \alpha_k \left(\mathbf{m}_1\mathbf{m}_1^T + \mathbf{m}_2\mathbf{m}_2^T\right)$
      end for
      $\mathbf{w}_i \longleftarrow$ smallest eigenvector of $\mathbf{M}$
      for $k = 1 \ldots K$ **do**
        $\mathbf{m}_1 \longleftarrow \mathbf{C}_k\mathbf{w}_i$
        $\mathbf{m}_2 \longleftarrow \mathbf{C}_k^T\mathbf{w}_i$
        $\mathbf{M} \longleftarrow \mathbf{M} + \alpha_k \left(\mathbf{m}_1\mathbf{m}_1^T + \mathbf{m}_2\mathbf{m}_2^T\right)$
      end for
    end for
  until convergence
  $\mathbf{W} \longleftarrow \mathbf{W}\mathbf{P}^T$
  return W
end function

$0 < k \leqslant K$. $\mathbf{C}_0(\mathbf{s})$ was the unit matrix. Then, a matrix $\mathbf{A}$ with elements also drawn from the standard normal distribution was used to generate a set of correlation matrices $\mathbf{C}_k(\mathbf{x}) = \mathbf{A}\mathbf{C}_k(\mathbf{s})\mathbf{A}^T$. Clearly, these matrices can be jointly diagonalized with any matrix, that differs from $\mathbf{A}$ only by a permutation of the rows of $\mathbf{A}^{-1}$, or by scale factors multiplied to them. For every dataset $K + 1 = 15$ matrices of size 10×10 were used.

2. **Approximately diagonalizable data.** This data set was generated the same way as the fully diagonalizable data, except that every $\mathbf{C}_k(\mathbf{x})$ was computed with an individual matrix $\mathbf{A}_k$ which slightly deviated from $\mathbf{A}$. The elements $A_{k,ij}$ were drawn independently from the normal distribution with $\mu = A_{ij}$; $\sigma = 10^{-2}$. For every data set $K + 1 = 15$ matrices of size 10×10 were used.

3. **Linear mixture of audio data.** Here six different channels of audio data (music and speech, zero DC) sampled at 44.1 kHz (10.000 samples) were used. The cross correlation matrices $\mathbf{C}_k(\mathbf{s})$,

$$C_{k,ij}(\mathbf{s}) = \frac{1}{T}\sum_t s_i(t)s_j(t+k)\,, \tag{2.52}$$

were computed for time lags $k = 0, 5, 10, \ldots, 95, 100$. Negative lags can be ignored, because $\mathbf{C}_{-k}(\mathbf{s}) = \mathbf{C}_k(\mathbf{s})^T$. Then, a matrix $\mathbf{A}$ with elements drawn from the standard normal distribution was used to generate a set of correlation matrices $\mathbf{C}_k(\mathbf{x}) = \mathbf{A}\mathbf{C}_k(\mathbf{s})\mathbf{A}^T$.

Algorithm 2 The QDIAG-algorithm with complexity $O(N^5)$

function QDIAG($\mathbf{C}_0$, $\{\mathbf{C}_k\}$)
  compute $\mathbf{P}$ so that $\mathbf{P}^T\mathbf{C}_0\mathbf{P} = \mathbf{I}$
  for $k = 1 \ldots K$ **do**
    $\mathbf{C}_k \longleftarrow \mathbf{P}^T\mathbf{C}_k\mathbf{P}$
  end for
  initialize W_{ij} randomly according to $\mathcal{N}(0;1)$
  $\mathbf{\Gamma} \longleftarrow 0^{N\times N\times N\times N}$
  for $i, j, g, h = 1 \ldots N$ **do**
    for $k = 1 \ldots K$ **do**
      $\mathbf{\Gamma}_{ijgh} = \mathbf{\Gamma}_{ijgh} + \alpha_k (\mathbf{C}_k)_{ig} (\mathbf{C}_k)_{jh}$
      $\mathbf{\Gamma}_{ijgh} = \mathbf{\Gamma}_{ijgh} + \alpha_k (\mathbf{C}_k)_{gi} (\mathbf{C}_k)_{hj}$
    end for
  end for
  $\mathbf{\Gamma} \longleftarrow mat_{N^2\times N^2}(\mathbf{\Gamma})$
  $\mathbf{\Omega} \longleftarrow vec\left(\mathbf{W}\mathbf{W}^T\right)$
  repeat
    for $i = 1 \ldots N$ **do**
      $\mathbf{\Omega} \longleftarrow \mathbf{\Omega} - vec\left(\mathbf{w}_i\mathbf{w}_i^T\right)$
      $\mathbf{M} \longleftarrow mat_{N\times N}(\mathbf{\Gamma}\mathbf{\Omega})$
      $\mathbf{w}_i \longleftarrow$ smallest eigenvector of $\mathbf{M}$
      $\mathbf{\Omega} \longleftarrow \mathbf{\Omega} + vec\left(\mathbf{w}_i\mathbf{w}_i^T\right)$
    end for
  until convergence
  $\mathbf{W} \longleftarrow \mathbf{W}\mathbf{P}^T$
  return W
end function

These matrices are in general not exactly diagonalizable, because the $\mathbf{C}_k(\mathbf{s})$ are only approximately diagonal.

In all experiments, the weighting factors were set equally to $\alpha_k = \frac{1}{(K+1)}$.

Benchmark results

The performance of QDIAG was compared with *Fast Frobenius Diagonalization (FFDIAG)* (Ziehe et al., 2003) and the *Alternating Columns/Diagonal Centers (ACDC)* algorithm proposed by Yeredor (2002). Algorithms that are limited to orthogonal matrices $\mathbf{W}$ or require positive definite correlation matrices were not considered. First, the convergence behavior in terms of the diagonalization error as a function of the number of iterations was investigated.

Every diagonalization matrix $\mathbf{W}$ found by QDIAG fulfills the constraint (2.37), and the diagonalization error

$$E := \frac{1}{(N^2 - N)} L(\mathbf{W}) \tag{2.53}$$

can be used as a performance measure. $N^2 - N$ is the number of off-diagonal elements of each correlation matrix. The rows $\mathbf{w}_i$ of the diagonalization matrices obtained by the FFDIAG and ACDC methods were normalized according to $\left(\mathbf{w}_i^T\mathbf{C}_0\mathbf{w}_i\right)^{-\frac{1}{2}}\mathbf{w}_i$ before evaluating equation (2.53).

The first experiment was performed with **fully diagonalizable data**. All three methods were applied to 5 randomly generated data sets. Figure 2.3 shows the diagonalization error vs. the number of iterations. As expected, all methods achieve an error of almost zero[3], which for ACDC (dash-dotted lines), however, required more than the 150 shown iterations. The convergence rates of ACDC and QDIAG (solid lines) are both linear, but

[3]The final errors of approx. 10^{-30} are due to the limited precision of the `double` data type, that had been used.

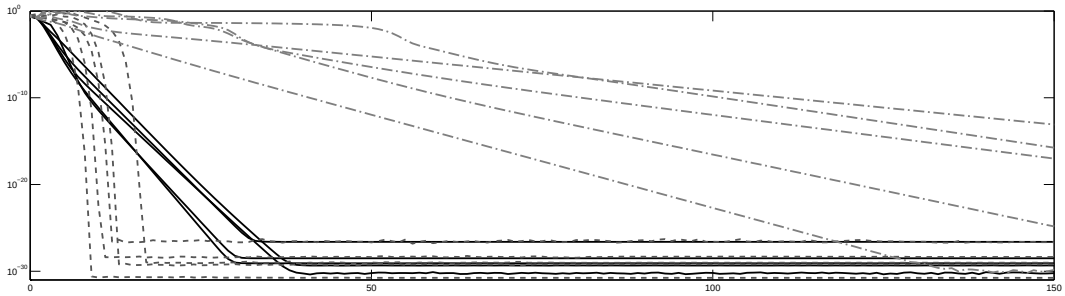


Figure 2.3: Diagonalization error E as a function of the number of iterations for QDIAG (solid lines), FFDIAG (dashed lines), and ACDC (dash-dotted lines) for 5 different 'fully diagonalizable' data sets.

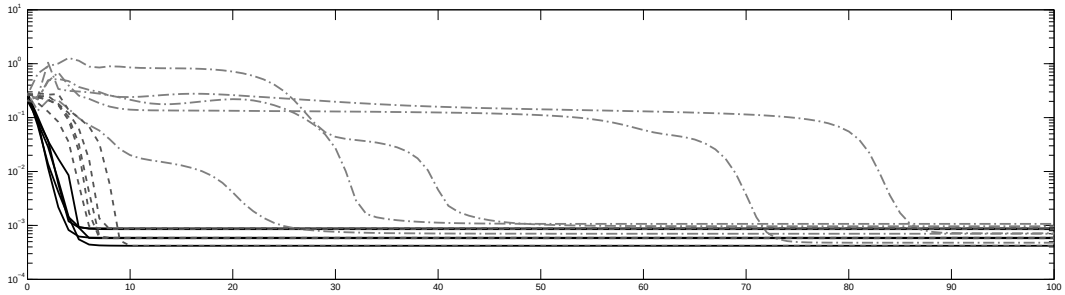


Figure 2.4: Diagonalization error E as a function of the number of iterations for QDIAG (solid lines), FFDIAG (dashed lines), and ACDC (dash-dotted lines) for 5 different 'approximately diagonalizable' data sets.

the speed of convergence of ACDC is noticeably slower than that of QDIAG. This behavior may be due to the fact that ACDC solves the inverse problem and has to estimate both, $\mathbf{W}^{-1}$ and the diagonal matrices $\mathbf{C}_k(\mathbf{s})$. Thus, there exists an M-dimensional space of equivalent solutions spaned by arbitrary scalings of the columns of $\mathbf{W}^{-1}$, which are compensated by the appropriate (inverse) scaling of the diagonal elements of $\mathbf{C}_k(\mathbf{s})$ (see Yeredor (2002) for further details). In QDIAG, however, the solution is unique (up to permutations). FFDIAG (dashed lines) exhibits the quadratic convergence rate mentioned in Ziehe et al. (2003) and Ziehe et al. (2004). Thus, it is slow initially, but becomes fast as soon as the linear approximation becomes valid (cf. figure 2.3). Compared to QDIAG, FFDIAG needs more iterations up to diagonalization errors of 10^{-8}, but only needs roughly half the number of iterations until full convergence. The effect becomes important when the set of matrices is not fully diagonalizable, and errors less than 10^{-8} are never reached. This is illustrated in a second experiment with **approximately diagonalizable data**. QDIAG, FFDIAG, and ACDC were applied to 5 datasets, randomly generated as described above. Figure 2.4 shows the resulting diagonalization errors vs. the number of iterations. For all 5 datasets, QDIAG converged faster than FFDIAG and considerably faster than ACDC. The initial increase in the error traces of ACDC is because the error which is shown here is not the one which is minimized by ACDC. Otherwise it would be monotonically decreasing (cf. also figure 2.11).

In a third experiment, conducted with the **linear mixture of audio data**, FFDIAG converged slightly faster than QDIAG for 3 out of 5 different mixtures (cf. figure 2.5), but also here the convergence of QDIAG was faster at the beginning. The convergence of ACDC is about one order of magnitude slower than the other two methods. Similar to the previous experiment, long periods of very slow convergence at the beginning can be observed. Because the data in this experiment are actually 5 different affine transformations of one and the same problem, QDIAG and FFDIAG converge to the same error value in every 5 trials. The final errors of ACDC are in general different in every trial.

In order to evaluate the computational efficiency of QDIAG, both, an $O(KN^3)$ and an

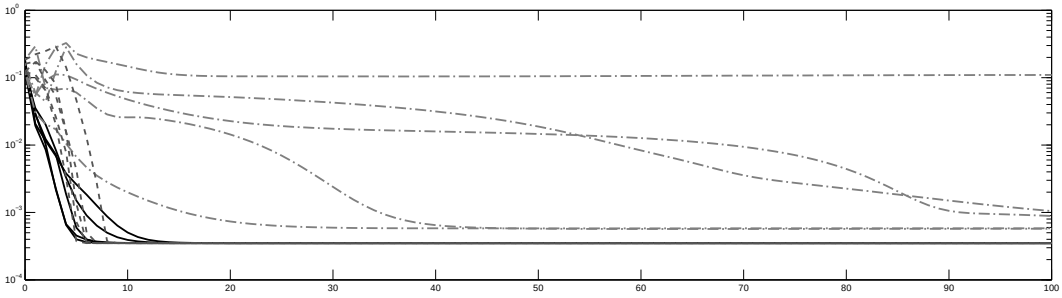


Figure 2.5: Diagonalization error E as a function of the number of iterations for QDIAG (solid lines), FFDIAG (dashed lines), and ACDC (dash-dotted lines) for 5 different 'linear mixtures of audio data'.

$O(N^5)$ implementation of QDIAG, and implementations of FFDIAG and ACDC were used. All programming has been done in MATLAB. Figure 2.6 shows the computation time per iteration for different values of N and K. The values have been averaged over 1000 iterations for evey setup. Although ACDC has the same complexity as the $O(KN^3)$ implementation, it is 40-60% slower than QDIAG, in terms of the computation load per iteration. FFDIAG is in general faster, but the $O(N^5)$ implementation can outperform FFDIAG in problems with many but small matrices. At first glance, this is surprising since FFDIAG is $O(KN^2)$ for moderate matrix sizes[4]. However, the impact of the N^5 term in QDIAG is small and can be neglected for small N, which leads to a much smaller actual efficiency: the slopes of the curves in 2.6.C are approximately like $O(N^{1.4})$ at $N = 2$ and like $O(N^2)$ at the other end, at $N = 20$. As expected, the computation time per iteration does not change in this implementation of QDIAG when the number of matrices increases, but considerably does so for the other methods. Thus, the break even point could be used to switch between the two implementations of QDIAG.

Unbalanced solutions and singular diagonalization matrices

On several occasions it could be observed that the final diagonalization error (2.53) that was achieved by FFDIAG was higher than the error obtained by QDIAG (cf. also figure 2.11). This effect occurs in particular for sets of matrices that can be poorly diagonalized only and is a result of the different ways trivial solutions are avoided. In FFDIAG $\mathbf{W}$ is initialized as the unit matrix and is multiplicatively updated with a non-singular matrix so that $\mathbf{W} \neq \mathbf{0}$ is guaranteed[5]. But this procedure cannot prevent single rows of $\mathbf{W}$ converging to a norm close to zero and $\mathbf{W}$ converging to a matrix close to a singular matrix. A small row norm has the consequence that the corresponding rows and columns of the correlation matrices have small elements, which leads to a small diagonalization error at the cost of almost vanishing diagonal elements. In this sense the solution is unbalanced because these rows and columns are weighted lower than others.

In figure 2.7 such an unbalanced solution is illustrated. The top row displays the correlation matrices after diagonalization with FFDIAG. The diagonalization matrix was taken from an experiment with the 'approximately diagonalizable data', which had large final error in FFDIAG, but low error in QDIAG. The off-diagonal elements have small values, but also some diagonal elements are almost zero. When the rows of $\mathbf{W}$ are scaled so that the constraint (2.37) holds, the off-diagonal elements become larger (center row). Hence, compared to the matrices diagonalized by QDIAG (bottom row), the unbalanced solution leads to the observed poor diagonalization error. For poor diagonalizations also ACDC can yield a higher error value (2.53) than QDIAG. In these cases prominent unbalanced diagonalizations were not observed, but otherwise poor solutions. Note that these can be

[4]For large N FFDIAG approaches $O(KN^3)$ due to the multiplicative updates in every iteration.

[5]Note that $\mathbf{W}$ here always denotes the diagonalizing matrix rather than the update matrix, as in Ziehe et al. (2004).

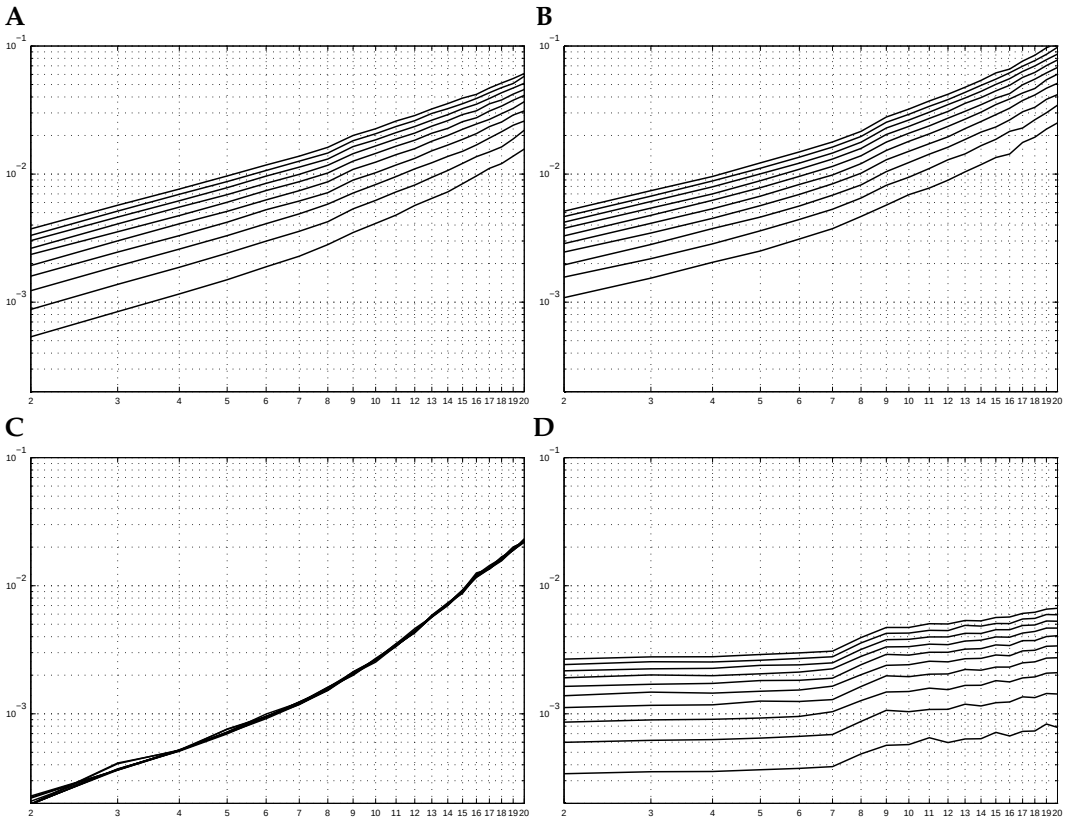


Figure 2.6: Average computation times per iteration of **A:** QDIAG (Algorithm 1), **B:** ACDC, **C:** QDIAG (Algorithm 2), and **D:** FFDIAG, as functions of the matrix size $N = 2, \ldots, 20$ (x-axis) and different numbers of matrices, $K = 5, 10, 15, \ldots, 50$ (individual lines). For moderate matrix sizes the $O(N^5)$ implementation of QDIAG can outperform all other methods in terms of costs per iteration.

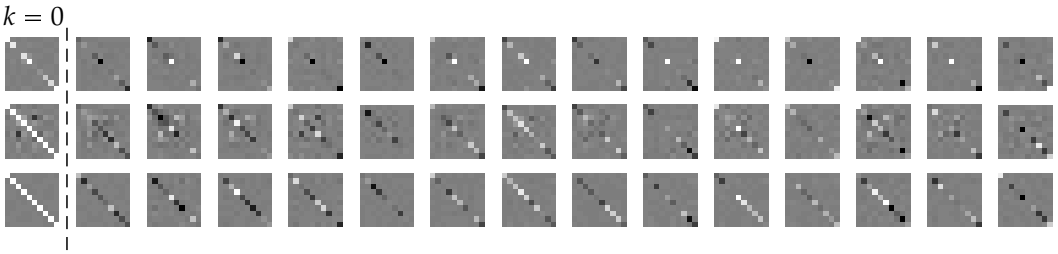


Figure 2.7: Example of an 'unbalanced' solution to the diagonalization problem. The figure shows the 15 correlation matrices (left ⟶ right), where the value of every matrix element is coded by the gray value of its respective pixel. *Top row:* Diagonalization result obtained by FFDIAG. The off-diagonal elements are small, but also some diagonal elements vanish. *Center row:* Result shown in the top row, scaled to hold the constraint (2.37). Note that the absolute values of the off-diagonal elements become larger. *Bottom row:* Results for QDIAG. Gray scale top row: white → black corresponds to 6 → -6; gray scale center and bottom row: white → black corresponds to 1 → -1.

quite good solutions of the inverse problem, which is natively optimized by ACDC.

Singular diagonalization matrices can also occur when QDIAG is applied because the constraint (2.37) is not violated when two or more rows of $\mathbf{W}$ are identical. This behavior can be observed mostly for problems for which the correlation matrices cannot be well diagonalized simultaneously and large diagonalization errors remain. Let two identical rows $\mathbf{w}_i = \mathbf{w}_j$ fulfill (2.37). Then, the off-diagonal elements $(\mathbf{W}\mathbf{C}_0\mathbf{W}^T)_{ij} = (\mathbf{W}\mathbf{C}_0\mathbf{W}^T)_{ji} = 1$ increase the diagonalization error (provided $\mathbf{C}_0 \in \mathcal{C}$). But this can still be the smallest error if at the same time the solution has no significant diagonal elements $(\mathbf{W}\mathbf{C}_k\mathbf{W}^T)_{ii} = (\mathbf{W}\mathbf{C}_k\mathbf{W}^T)_{jj}$ for the majority of the correlation matrices. Then, the off-diagonal elements $(\mathbf{W}\mathbf{C}_k\mathbf{W}^T)_{ij} = (\mathbf{W}\mathbf{C}_k\mathbf{W}^T)_{ji}$, which are equal to the diagonal elements, also have small absolute values and keep the diagonalization error low. In such a situation merging rows of $\mathbf{W}$ can be avoided when more weight is put onto $\mathbf{C}_0$ or other $\mathbf{C}_k$ that have large 'diagonal content', i.e. the corresponding α_0 rsp. α_k are raised compared to the others. In the limit of infinite weight on $\mathbf{C}_0$, QDIAG would perfectly diagonalize it, i.e. perform a sphering w.r.t. $\mathbf{C}_0$ and allow for an additional unitary transformation to approximately diagonalize the remaining $\mathbf{C}_k$.

Indefinite and asymmetric correlation matrices

QDIAG does not impose restrictions on the symmetry or the definiteness of the correlation matrices, except for the matrix $\mathbf{C}_0$. Interestingly, it could be observed that the speed of convergence is noticeably higher for matrices that have some negative eigenvalues. Figure 2.8 shows the diagonalization error as a function of the number of iterations for the **fully diagonalizable data** in comparison with sets of positive definite matrices constructed the same way, but with positive diagonal matrices $\mathbf{C}_k(\mathbf{s})$. The 5 leftmost and rightmost curves represent the results for the indefinite and the positive definite matrices, respectively. The reduced convergence speed was also observed for strictly negative definite $\mathbf{C}_k(\mathbf{x})$, and also for not fully diagonalizable matrices.

Non-symmetric correlation matrices frequently occur in applications of second order blind source separation techniques because of their asymmetric way of construction (2.52). The empirical correlation matrices can be split into a symmetric and an anti-symmetric part,

$$\mathbf{C}_k = \frac{1}{2}\left(\mathbf{C}_k + \mathbf{C}_k{}^T\right) + \frac{1}{2}\left(\mathbf{C}_k - \mathbf{C}_k{}^T\right) \ .$$

The anti-symmetric part is a result of finite source cross correlations and does not contain any information that helps for the source separation. Therefore, and because many joint diagonalization methods cannot handle asymmetric matrices, it is usually neglected. Thus, it may be more of academic interest to discuss how QDIAG is able handle the anti-symmetric part. Because an anti-symmetric matrix remains anti-symmetric after any bilinear transformation, diagonalization of an anti-symmetric matrix is equivalent to making that matrix as

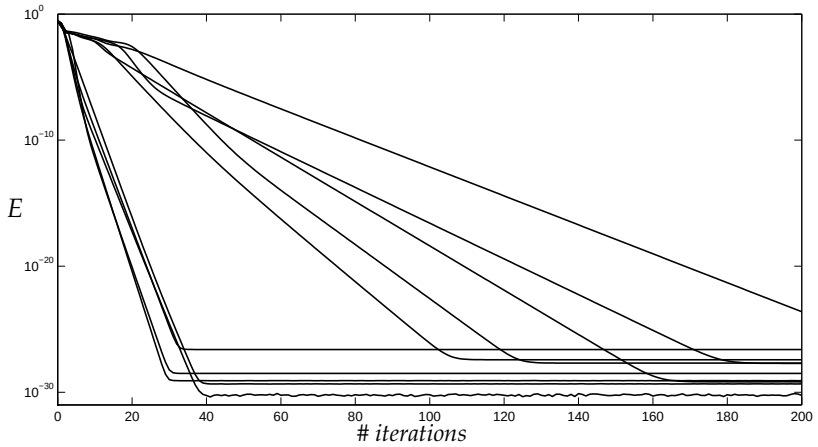


Figure 2.8: Diagonalization error as a function of the number of iterations of QDIAG for fully diagonalizable, indefinite (left group of curves) and positive definite (right group of curves) correlation matrices.

close as possible to zero. Clearly, this is only achievable with a singular matrix $\mathbf{W}$. But then the minimal diagonalization error must always be larger than zero if $\mathbf{C}_0$ is element of C. With singular $\mathbf{W}$ also $\mathbf{W}\mathbf{C}_0\mathbf{W}^T$ is singular. Under the constraint (2.37) this implies the existence of finite off-diagonal elements of $\mathbf{W}\mathbf{C}_0\mathbf{W}^T$ and, hence, finite diagonalization error. Figure 2.9 shows an example of an anti-symmetric diagonalization. A set of 4 anti-symmetric matrices was constructed, with the elements $C_{k,ij}$ drawn from the standard normal distribution for $j > i$ and $C_{k,ij} = -C_{k,ji}$ else. $\mathbf{C}_0$ was the unit matrix. These matrices are shown on the left panel of the figure. The right panel shows the result of a diagonalization with QDIAG.

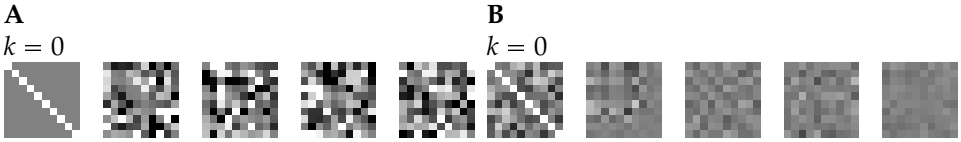


Figure 2.9: Diagonalization of anti-symmetric matrices. **A:** The unit matrix $\mathbf{C}_0$ together with 4 anti-symmetric matrices $\mathbf{C}_k$. **B:** The result of the 'diagonalization'. Color scale: white: 1, black: -1

Thus, the off-diagonal elements of the anti-symmetric matrices can be reduced only at the cost of increasing off-diagonal elements of $\mathbf{W}\mathbf{C}_0\mathbf{W}^T$. This trade-off is controlled by the weighting parameters α_k. Provided α_0 is not too small, QDIAG is able to converge to a non-singular (but generally poor) solution $\mathbf{W}$, even if all $\mathbf{C}_{k>0}$ are pure anti-symmetric. If $\mathbf{C}_0 \notin C$, however, then the algorithm will usually converge to a trivial solution, which has zero error and $\mathbf{w}_i = \mathbf{w}_j$ for all i, j.

Non-square diagonalization matrices

An interesting property of QDIAG is that it is able to optimize equations (2.44) even for non-square matrices $\mathbf{W} \in \mathbb{R}^{M\times N}$. There are two cases to be distinguished: incomplete diagonalization ($M < N$) and overcomplete diagonalization ($M > N$). In the case of **incomplete diagonalization**, equations (2.44) are minimized w.r.t. a subset of the off-diagonal elements, and the final diagonalization error is always lower or equal than the error for the corresponding complete diagonalization. Table 2.4.4 shows the diagonalization errors of undercomplete diagonalizations for 20 trials with the **approximately diagonalizable data**.

Incomplete diagonalizations can be applied when a full diagonalization does not lead to satisfactory results. In applications of blind source separation, for example, incomplete diagonalization allows to separate groups of sources that exhibit cross correlations among

M	$\log_{10}(E)$
2	-12.15 ± 12.89
3	-5.16 ± 0.57
4	-4.48 ± 0.51
5	-4.14 ± 0.53
6	-3.77 ± 0.71
7	-3.48 ± 0.76
8	-3.14 ± 1.07
9	-2.84 ± 1.27
10	-2.47 ± 1.66

Table 2.1: Diagonalization error as a function of M for incomplete diagonalizations ($M < N$) for 20 trials with the approximately diagonalizable data. The average decadic log of the error and the 95% probability range (3 times standard deviation) are shown. For $M = 2$ the tendency of QDIAG could be observed, to get stuck in local minima, thus leading to the large standard deviation.

each other, but not between the groups. This is illustrated with the help of a little example. A **linear mixture of audio data**. However, instead of 6 individual audio signals, three 2-channel stereo audio signals were used as sources of the linear mixture. Thus, there were three groups of two sources, with neglectable cross correlations between different groups and considerable cross correlations between the two sources in each group. The order of the sources was $\mathbf{s} = (s_{1,left}, s_{1,right}, s_{2,left}, s_{2,right}, s_{3,left}, s_{3,right})^T$. For the blind source separation achieved through complete diagonalization, the result in terms of the multiplication of $\mathbf{W}$ with the mixing matrix $\mathbf{A}$ was

$$\mathbf{WA} = \begin{pmatrix} \mathbf{2.45} & \mathbf{-2.49} & 0.00 & -0.02 & -0.01 & -0.01 \\ \mathbf{0.50} & \mathbf{0.52} & -0.03 & 0.02 & 0.02 & -0.02 \\ -0.02 & 0.03 & \mathbf{-0.80} & \mathbf{-0.48} & 0.01 & -0.01 \\ 0.04 & -0.04 & \mathbf{-0.65} & \mathbf{0.89} & 0.00 & -0.01 \\ 0.00 & -0.01 & 0.01 & -0.02 & \mathbf{-0.49} & \mathbf{-0.57} \\ -0.01 & 0.01 & 0.01 & 0.00 & \mathbf{1.49} & \mathbf{-1.43} \end{pmatrix},$$

which would be a diagonal (rsp. permutation) matrix for perfect separation. However, one can see that, as expected, the algorithm was not able to resolve the stereo channel pairs (cf. the bold faced sub-matrices) because the assumption of vanishing cross correlations was not valid for them. If for the application at hand it is only necessary to separate the sources, regardless of what linear combinations of the individual stereo channels come out, then the computation cost could be reduced by an incomplete diagonalization from $O(KN^3)$ to $O(KMN^2)$ rsp. from $O(N^5)$ to $O(MN^4)$. With $M = 3$ in this experiment the joint diagonalization led to

$$\mathbf{W}_{3\times 6}\mathbf{A} = \begin{pmatrix} \mathbf{-2.23} & \mathbf{1.59} & 0.02 & 0.05 & 0.01 & 0.00 \\ -0.01 & 0.01 & \mathbf{-1.00} & \mathbf{0.34} & -0.00 & -0.01 \\ -0.01 & 0.02 & 0.00 & 0.01 & \mathbf{1.54} & \mathbf{-1.06} \end{pmatrix}.$$

The incomplete diagonalization separated the signals well up to linear combinations of the individual stereo channels. The shown results were manually corrected for the permutation in the rows of $\mathbf{W}$.

In the case of **overcomplete diagonalization** a perfect diagonal solution in general does not exist. The matrices $\mathbf{WC}_k\mathbf{W}^T$ are of size $M \times M$, but have a rank of at most N. They can only be diagonal if at least $M - N$ diagonal elements are zero. With $\mathbf{W}$ holding the constraint (2.37), this is impossible for positive definite $\mathbf{C}_k$. But also for indefinite $\mathbf{C}_k$ such a solution will not exist in general. Therefore, the $M > N$ case may not be of practical interest. QDIAG can handle this case correctly. However, diagonalization errors of overcomplete diagonalizations are usually rather poor. Figure 2.10 shows an example for an overcomplete diagonalization. A set of 15 fully diagonalizable matrices (cf. figure 2.3) of size 10×10 was constructed, and a 12×10 matrix $\mathbf{W}$ was optimized to diagonalize them. For the first trial (top row) all weighting parameters were set to $\alpha_k = \frac{1}{15}$. The algorithm has to 'distribute' the unavoidable error over the off-diagonal elements, which leads to identical rows $\mathbf{w}_8$ and $\mathbf{w}_9$. In the next experiment more weight was put on $\mathbf{C}_0$ by setting $\alpha_0 = 10\alpha_{k\neq 0}$. As expected, the algorithm takes more care of diagonalizing $\mathbf{C}_0$, at the cost of slightly larger off-diagonal elements in the other matrices.

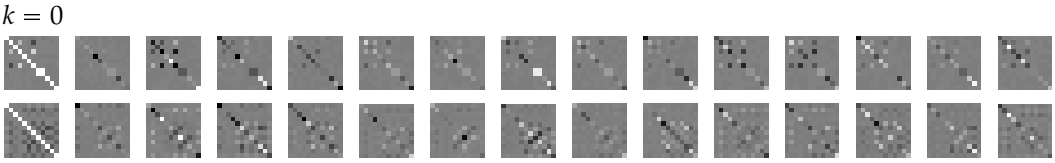


Figure 2.10: Overcomplete diagonalization. A fully diagonalizable set of 15 10×10 matrices was diagonalized by a 12×10 matrix. *Top row:* The same weight on all matrices. *Bottom row:* The weight on $\mathbf{C}_0$ is 10 times larger than on the other matrices.

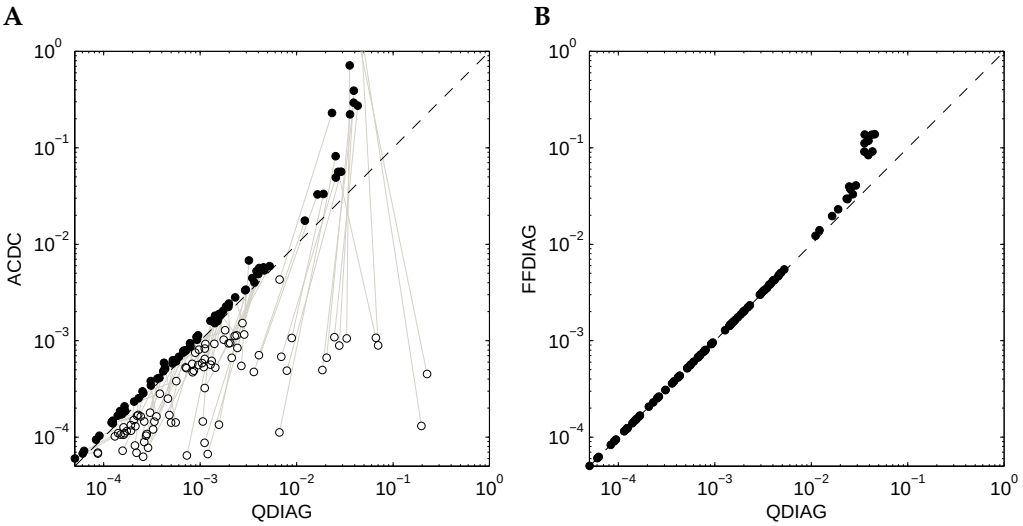


Figure 2.11: Diagonalization errors for 100 different 'approximately diagonalizable' data sets. **A:** The final error of QDIAG plotted against the final error of ACDC. **Black dots** indicate the final error in terms of error measure (2.53), **white dots** in terms of the error which is minimized by ACDC (cf. Yeredor (2002)). Lines connect corresponding results. **B, black dots:** The final error of QDIAG against FFDIAG in terms of error measure (2.53). Because the error which is minimized by FFDIAG is not accessible outside the FFDIAG iterations, it cannot be plotted for comparison.

2.4.5 Concluding remarks

The QDIAG algorithm could be shown to be fast and able to handle arbitrary diagonalization matrices (including non-orthogonal, non-square matrices) and indefinite correlation matrices. The performance evaluation showed that the new method is competitive with FFDIAG and considerably faster than ACDC in terms of the convergence rate.

In the benchmarks for the convergence rates the performance was compared using the error measure (2.53), which is the objective function to be optimized in QDIAG, but not in the other two methods. This can lead to final errors of FFDIAG and ACDC in their native measures that are smaller than the QDIAG error measure (2.53) and is the reason for the error of ACDC to be not monotonically decreasing in the experiments presented in figures 2.3–2.5. Figure 2.11 shows the QDIAG diagonalization errors (2.53) (black dots) that were achieved by all three methods for 100 trials with 'approximately diagonalizable' datasets. Since it is possible to compute the error which is minimized by ACDC also for results achieved with other methods[6], one can additionally compare the results of QDIAG and ACDC in terms of this error measure (figure 2.11.A, white dots). In particular for poor diagonalizations, ACDC may reach significantly lower final error values compared to QDIAG in terms of its own error measure. From figure 2.11 one can also see, however, that the QDIAG error measure is quite consistent in so far that all three methods converge to

[6] In contrast to the FFDIAG error, which can be only accessed together with the FFDIAG iterations.

very similar error values for fair diagonalizations. Note that different error measures do not affect the number of iterations, that are required for full convergence.

In general the error measure has to be chosen according to the application at hand. In QDIAG, $\mathbf{C}_0$ defines the 'balance' of the solution, and certainly there are applications for which it is not meaningful to treat one of the matrices in C different than the others. However, for the blind separation of sources, $\mathbf{C}_0$ can naturally be assigned to the covariance matrix of the signals in order to avoid trivial or unbalanced solutions. Fortunately, unbalanced solutions, for which some rows of $\mathbf{W}$ have almost vanishing norm, tend to occur only in problems that are not properly diagonalizable and have comparable large minimal diagonalization errors. If such solutions are undesired for the given problem, a major advantage of QDIAG is the possibility to explicitly avoid them by using the constraint on $\mathbf{C}_0$.

2.5 Extraction of single sources

As shown in section 2.4.4 the QDIAG algorithm is capable of separating less sources than observations (or a less 'source groups' that observations, to be precise). This concept is meaningful as long as there are at least two outputs. For the blind separation of single sources from linear mixtures another approach is necessary, which will be presented in the following. As well as QDIAG also this algorithm is driven by the concept of approximate matrix diagonalization.

2.5.1 Cost function for extracting single components

In order to separate a single source from a linear mixture of M sources plus additive noise,

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \mathbf{n}_t \ ,$$

one has to find an N-dimensional vector $\mathbf{w}$ that yields zero for all components except one when it is multiplied with the unknown and rectangular $N \times M$ mixing matrix $\mathbf{A}$,

$$\mathbf{w}^T\mathbf{A} = (c, 0, \ldots, 0) \ . \tag{2.54}$$

Without loss of generality $\mathbf{w}$ may be constrained to unit length and the first component of $\mathbf{w}^T\mathbf{A}$ may be the one that is different from zero. Thus $\mathbf{w}$ separates the first source

$$\mathbf{w}^T\mathbf{x}_t = c\, s_{1,t} + \mathbf{w}^T\mathbf{n}_t$$

from the mixture, while $\mathbf{A}$ is not known. Attempts are not only to fulfill (2.54), but also to achieve a large value c to get the variance of the recovered source large compared to the noise.

Let us consider a matrix $\mathbf{M}$,

$$\mathbf{M} := \mathbf{C}_\tau(\mathbf{s})\mathbf{A}^T\,(\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_{N-1}) \ . \tag{2.55}$$

$\mathbf{C}_\tau(\mathbf{s})$ is the shifted source cross correlation matrix, $\mathbf{a}$ is the normalized first column vector of $\mathbf{A}$, and the $\mathbf{b}_i$ are $N-1$ orthogonal vectors of unit length that span the subspace orthogonal to $\mathbf{a}$,

$$\mathbf{b}_i^T\mathbf{b}_j = \delta_{ij}, \qquad \mathbf{b}_i^T\mathbf{a} = 0 \ .$$

Then a multiplication from the right of (2.54) with $\mathbf{M}$ keeps its form unchanged, i.e.

$$\mathbf{v}^T := \mathbf{w}^T\mathbf{A}\mathbf{M} = (\bar{c}, 0, \ldots, 0) \ .$$

Using $\mathbf{C}_\tau(\mathbf{x}) = \mathbf{A}\mathbf{C}\tau(\mathbf{s})\mathbf{A}^T$, a cost function is obtained, which is the squared sum of all

elements of $\mathbf{v}$ except the first.

$$\begin{aligned} E &= \sum_{\tau} \sum_{i=2}^{N} \left(\mathbf{w}^T \mathbf{C}_\tau(\mathbf{x}) \left(\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_{N-1}\right)\right)_i^2 \\ &= \sum_{\tau} \Big(\left(\mathbf{w}^T \mathbf{C}_\tau(\mathbf{x}) \left(\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_{N-1}\right)\right) \left(\mathbf{w}^T \mathbf{C}_\tau(\mathbf{x}) \left(\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_{N-1}\right)\right)^T - \\ &\quad - \mathbf{w}^T \mathbf{C}_\tau(\mathbf{x}) \mathbf{a}\mathbf{a}^T \mathbf{C}_\tau(\mathbf{x})^T \mathbf{w} \Big) \\ &= \sum_{\tau} \mathbf{w}^T \mathbf{C}_\tau(\mathbf{x}) \left(\mathbf{I} - \mathbf{a}\mathbf{a}^T\right) \mathbf{C}_\tau(\mathbf{x})^T \mathbf{w} \end{aligned} \tag{2.56}$$

E has to be minimized with respect to $\mathbf{w}$ and $\mathbf{a}$ under the constraint $\|\mathbf{w}\| = \|\mathbf{a}\| = 1$. In estimating a single source this way, the number of parameters to be estimated could be reduced from N^2 for a full demixing matrix $\mathbf{W}$ to $2N$ for the vectors $\mathbf{w}$ and $\mathbf{a}$.

2.5.2 Prior knowledge for a and w

Successful minimization of (2.56) will lead to the separation of one source from the mixture, but there is no control yet about which source is being extracted. Therefore, some prior knowledge is needed that allows to add appropriate regularization terms to the cost function (2.56),

$$E = E_{(2.56)} + E_{\mathbf{w}}(\mathbf{w}) + E_{\mathbf{a}}(\mathbf{a}) \; . \tag{2.57}$$

Although it is in general possible to use any functional form for the regularization terms that reflects the knowledge about $\mathbf{w}$ and $\mathbf{a}$ respectively, one is interested in using such that are computationally convenient. The cost function (2.56) is quadratic in $\mathbf{w}$ and quadratic in $\mathbf{a}$ (but biquadratic in $\mathbf{w}$ and $\mathbf{a}$ together). This particularly simple form of the cost function is preserved when quadratic regularization terms are used,

$$E_{\mathbf{a}}(\mathbf{a}) = \mathbf{a}^T \Sigma_{\mathbf{a}}^{-1} \mathbf{a} \quad \text{and} \quad E_{\mathbf{w}}(\mathbf{w}) = \mathbf{w}^T \Sigma_{\mathbf{w}} \mathbf{w} \; . \tag{2.58}$$

$\Sigma_{\mathbf{a}}$ is chosen to have its largest eigenvalue in direction of $\boldsymbol{\mu}_{\mathbf{a}}$, the most probable value of $\mathbf{a}$. The other eigenvalues have to be small compared to the largest one. The smallest value of $E_{\mathbf{a}}$, when $\mathbf{a}$ is constrained to unit length, is obtained for $\mathbf{a}$ in direction of $\boldsymbol{\mu}_{\mathbf{a}}$. $E_{\mathbf{w}}$ must be constructed in a different way, because one usually has prior information only about the mixing process and not its inverse. We know from (2.54) that $\mathbf{w}$ has to be orthogonal to all those column vectors of $\mathbf{A}$ that represent the mixing process of the unwanted components. If prior knowledge is available for any of them, quadratic functions

$$E_{\bar{\mathbf{a}}_i}(\mathbf{a}) = \mathbf{a}^T \Sigma_{\bar{\mathbf{a}}_i}^{-1} \mathbf{a}$$

can be defined just like $E_{\mathbf{a}}$. Then, $E_{\mathbf{w}}$ must yield large values if $E_{\bar{\mathbf{a}}_i}$ is small and vice versa. Hence the quadratic form must be

$$\Sigma_{\mathbf{w}} = \left(\sum_i \Sigma_{\bar{\mathbf{a}}_i} \right) \; , \tag{2.59}$$

where the sum is over all columns of $\mathbf{A}$ that prior knowledge is available for. Both, $\Sigma_{\mathbf{a}}$ and $\Sigma_{\mathbf{w}}$, should be positive semi-definite in order to guarantee non-negative error values of the regularized error function . The strength of regularization directly depends on the determinants of $\Sigma_{\mathbf{a}}$ and $\Sigma_{\bar{\mathbf{a}}_i}$ and has to be set to reasonable values.

Due to the choice of regularization terms, the error function is still quadratic in $\mathbf{w}$ and in $\mathbf{a}$ and biquadratic in $\mathbf{w}$ and $\mathbf{a}$ together. It has to be minimized with respect to $\mathbf{w}$ and $\mathbf{a}$ under the constraint $\|\mathbf{w}\| = \|\mathbf{a}\| = 1$. This can be efficiently done by alternating optimizations with respect to $\mathbf{w}$ and $\mathbf{a}$:

(i) update $\mathbf{w}$ to the smallest eigenvector of

$$\mathbf{C}_\tau(\mathbf{x})(\mathbf{I} - \mathbf{a}\mathbf{a}^T)\mathbf{C}_\tau(\mathbf{x})^T + \Sigma_{\mathbf{w}} \; , \tag{2.60}$$

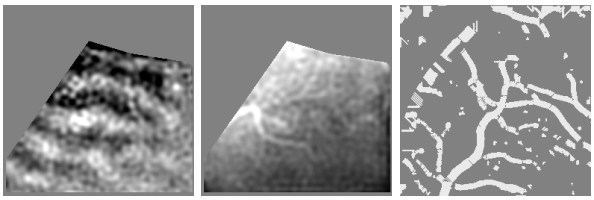


Figure 2.12: The toy datasets used for simulation. These are reconstructed sources from an optical recording experiment.

(ii) update $\mathbf{a}$ to the smallest eigenvector of

$$\Sigma_{\mathbf{a}}^{-1} - \mathbf{C}_\tau(\mathbf{x})^T(\mathbf{w}\mathbf{w}^T)\mathbf{C}_\tau(\mathbf{x}) \; . \tag{2.61}$$

until convergence.

2.5.3 Extraction of additional sources

The algorithm can be applied iteratively to separate additional sources. After one source is successfully extracted using $\hat{s}_{i,t} = \mathbf{w}^T\mathbf{x}_t$, it is also possible to remove that source from the remaining mixtures using the vector $\mathbf{a}$. To do that, one has to construct a $(N-1 \times N)$-dimensional Matrix

$$\mathbf{W}' = \left(\mathbf{w}'_1, \ldots, \mathbf{w}'_{N-1}\right)^T \; .$$

The $\mathbf{w}'_i$ are chosen to be pairwise orthogonal and to span the subspace orthogonal to $\mathbf{a}$. For the separated source $\mathbf{a}$ is the corresponding column vector of the mixing matrix $\mathbf{A}$. Thus, in $\mathbf{W}'\mathbf{A}$ this column is equal to zero. Hence, $\mathbf{x}' = \mathbf{W}'\mathbf{A}\mathbf{s}$ is a mixture of $\mathbf{s}$, that does not contain the separated source. $\mathbf{x}'$ can now be used as input to separate the next source. It is important to note that with every step also the priors have to be projected into the subspace of $\mathbf{x}'$ using $\mathbf{W}'$.

2.5.4 Experiments

In the following the capabilities of this algorithm are demonstrated on the basis of toy data with prototype signals and time courses that are similar to those obtained in experiments with optical imaging of neural activity.

In optical imaging a cortical area of interest is illuminated with monochromatic light of wavelengths usually between 500 - 800 nm. This area is then recorded with a sensitive CCD- or video camera. Changes in reflectance of this light from the cortex are mainly due to variations in the light scattering properties of the tissue and to variations in the local concentrations of deoxigenated and oxygenated hemoglobin. Typically these changes are very small and do not exceed 0.1% of the reflected light (Blasdel and Salama, 1986). Because of these small signal intensities, the signal to noise ratios are of the order of 0 dB.

It could be shown that for this data second order BSS algorithms are superior to those that rely on factorizing source distributions (Schöner et al., 2000). To evaluate the properties of the single source separation procedure, 20 mixtures of the sources shown in figure 2.12 were generated, using a 20×3-mixing matrix $\mathbf{A}$. Each column represents the time course of a single source (cf. figure. 2.13.D, left) and was motivated by results of optical imaging experiments (Malonek and Grinvald, 1996). Finally, white noise with approximately 3dB SNR was added to the mixture. In three experiments $\mathbf{w}$ and $\mathbf{a}$ where estimated using a set of 8 cross correlation matrices with shifts τ arranged in a star like pattern. In figure 2.13.A, the first column of $\mathbf{A}$ was used as the prior for $\mathbf{a}$ and the third column was used as the orthogonal prior for $\mathbf{w}$. The eigenvalues of the matrices $\Sigma_{\mathbf{a}}$ and $\Sigma_{\bar{\mathbf{a}}_i}$ where chosen as 1 for the largest one and 0.1 for the remaining. The separated source is displayed next to the estimated values of $\mathbf{w}$ and $\mathbf{a}$. The algorithm performs well, even if the shape of the

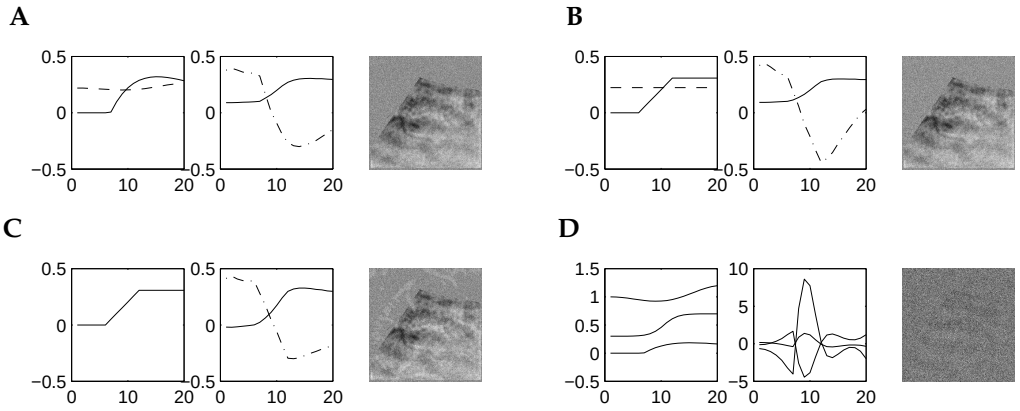


Figure 2.13: Results on a 20 × 3 mixture of the sources of figure 2.12. **A-C:** one separated source using different priors. left: prior for **a** (solid), to **w** orthogonal prior (dashed), middle: estimated course of **a** (solid) and **w** (dash-dotted), right: the separated source. **D:** left: course of the column vectors of the mixing matrix **A**, middle: course of the row vectors of the pseudo inverse of **A**, right: the separated source using the first row of the pseudo inverse

priors only roughly corresponds to columns of **A** (figure 2.13.B). If the orthogonal prior was omitted, the result was only slightly worse (figure 2.13.C). This shows that the priors just select on which source the algorithm focuses. The particular shape of **w** is not much influenced by the priors.

It is important to note, that even exact knowledge of **a** is not sufficient to recover the source, since no **w** can be derived from **a** alone. Only the knowledge about mixing process of all sources, i.e. the whole matrix **A** would allow to compute it's pseudo inverse. However, source reconstruction using the first row of the pseudo inverse of **A** fails completely (figure 2.13.D). Although the pseudo inverse would perform a perfect separation, it dramatically amplifies the noise and yields a poor reconstruction. The single source separating algorithm, however, is able to exploit information from the higher number of mixtures to achieve a much better SNR in the separated source.

2.6 Multi-dimensional ICA

Let us consider a stochastic N-channel source signal $(\mathbf{s}_t)_{t\in \mathrm{I}}$, and assume that for every $\mathrm{J} \in \mathscr{H}(\mathrm{I})$ a probability density $p(\mathbf{s}_{\mathrm{J}+t})$ exists. The source signals shall be stationary, thus p is invariant on t. For ease of notation we introduce the $N \times M$ random matrix $\mathbf{S}_t$, the j-th column of which equals $\mathbf{s}_{t+\tau_j}$, where τ_j is the j-th element of the ordered set J and $M = |\mathrm{J}|$. We call $\mathbf{S}_t$ a patch of the signal $(\mathbf{s}_t)_{t\in \mathrm{I}}$. The sources are linearly combined by an unknown mixing matrix **A** of full rank to produce a set of N observations $(\mathbf{x}_t)_{t\in \mathrm{I}}$,

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t \quad \text{rsp.} \quad \mathbf{X}_t = \mathbf{A}\mathbf{S}_t. \tag{2.62}$$

The mixing process is assumed to be stationary, i.e. that the mixing matrix **A** is independent of t. Hence, in the following we may drop the subscript t from random vectors and matrices, as long as we consider their statistical properties. Further, we need to introduce the following two notations:

$$\mathbf{s}_{\star j} = (S_{1j}, \ldots, S_{Nj})^T$$

be a vector containing the elements of the j-th column of **S**, and

$$\mathbf{s}_{i\star} = (S_{i1}, \ldots, S_{iM})$$

be a row vector containing the elements of the i-th row of **S**.

The goal of linear blind source separation is to find an appropriate demixing matrix $\mathbf{W}$ which, when applied to the observations $\mathbf{x}_t$, recovers good estimates,

$$\hat{\mathbf{s}}_t = \mathbf{W}\mathbf{x}_t \approx \mathbf{s}_t, \tag{2.63}$$

of the original source signals (up to a permutation and scaling of the sources). Since the mixing matrix $\mathbf{A}$ is not known, it's inverse $\mathbf{W}$ has to be detected blindly, i.e. only properties of the sources which are detectable in the mixtures can be exploited. In contrast to many other ICA algorithms, let us consider here a set of sources which are not independent, in the sense that not even instantaneous independence, equation (2.17), holds, i.e.

$$p(\mathbf{s}) = p(\mathbf{s}_{\star j}) \neq \prod_{i=1}^{N} p(S_{ij}) \; . \tag{2.64}$$

These dependencies among the sources make most ICA algorithms fail. However, things are not completely lost if the dependencies are of a certain structure, which will be described in the following:

Every row of $\mathbf{S}$ be a superposition of fixed, but unknown, feature vectors $\mathfrak{a}_{\star j}^T$,

$$\mathbf{s}_{i\star} = \sum_{j=1}^{M} U_{ij} \mathfrak{a}_{\star j}^T \; , \tag{2.65}$$

or, in matrix notation,

$$\mathbf{S} = \mathbf{U}\mathfrak{A}^T \; , \tag{2.66}$$

where $\mathfrak{a}_{\star j}$ is the j-th column of the $M \times M$ matrix $\mathfrak{A}$, and $\mathbf{U}$ is a random matrix containing the coefficients U_{ij}. The features may be linear independent so that $\mathfrak{A}$ has full rank. With that the distribution of the source patches $\mathbf{S}$ is given by the distribution of the coefficients $\mathbf{U}$,

$$p(\mathbf{S}) = |\det(\mathfrak{A})|^{-N} p(\mathbf{U}) \; . \tag{2.67}$$

The features $\mathfrak{a}_{\star j}$ may fall into two categories: those that contribute to the source dependencies and those that don't. Because the features $\mathfrak{A}$ are fixed, this is fully determined by the distribution of their coefficients. We partition the random matrix $\mathbf{U}$ into M° columns, the corresponding features of which do not contribute to the source dependencies, and M^+ columns of dependent feature coefficients,

$$\mathbf{U} = \left(\mathbf{U}^\circ, \mathbf{U}^+\right) \; . \tag{2.68}$$

So it may hold

$$p(\mathbf{u}_{\star j}^\circ) = \prod_{i=1}^{N} p(U_{ij}^\circ) \; , \qquad p(\mathbf{u}_{\star j}^+) \neq \prod_{i=1}^{N} p(U_{ij}^+) \; . \tag{2.69}$$

The basic idea behind the multi-dimensional ICA algorithm is to estimate first the feature matrix $\mathfrak{A}$ rsp. it's inverse $\mathfrak{W} = \mathfrak{A}^{-1}$, together with the coefficients $\mathbf{U}$. Once this is done the features belonging to $\mathbf{U}^\circ$ have to be detected and discarded in order to eliminate the source dependencies. The mean for detecting the features is the assumption that they occur independently from each other. For this to function the coefficients in every row of $\mathbf{U}$ have to be independently distributed and non-Gaussian,

$$p(\mathbf{U}) = \prod_{j=1}^{M} p(\mathbf{u}_{\star j}) \; . \tag{2.70}$$

Without loss of generality the coefficients may be normalized to zero mean and unit variance,

$$\left\langle U_{ij} \right\rangle = 0 \; , \qquad \left\langle U_{ij}^2 \right\rangle = 1 \; . \tag{2.71}$$

Equations (2.66) and (2.62) constitute linear operations, which can be arbitrarily interchanged,

$$\mathbf{X} = \mathbf{AS} = \mathbf{AU}\mathfrak{A}^T = \mathbf{V}\mathfrak{A}^T \,. \tag{2.72}$$

Consider random vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{v}}$ which are uniform mixtures of all rows of $\mathbf{X}$ rsp. $\mathbf{V}$,

$$p(\tilde{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{x}_{i\star}^T) \,, \qquad p(\tilde{\mathbf{v}}) = \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{v}_{i\star}^T) \,. \tag{2.73}$$

From equation (2.72) follows that $\tilde{\mathbf{x}} = \mathfrak{A}\tilde{\mathbf{v}}$ and, hence,

$$p(\tilde{\mathbf{x}}) = |\det \mathfrak{A}|^{-1} p(\tilde{\mathbf{v}}) \,. \tag{2.74}$$

If we can show that the elements of the $\tilde{\mathbf{v}}$ are independent, then we can estimate $\mathfrak{A}$ or $\mathfrak{W} = \mathfrak{A}^{-1}$ by means of ICA on realizations of $\tilde{\mathbf{x}}$. From the independence assumption (2.70) follows

$$p(\mathbf{V}) = \prod_{j=1}^{M} p\left(\mathbf{v}_{\star j}\right) \,, \tag{2.75}$$

and therefrom

$$p(\mathbf{v}_{i\star}) = \prod_{j=1}^{M} p(V_{ij}) \,. \tag{2.76}$$

We assume all $\mathbf{x}_{i\star}$ and, hence, all $\mathbf{v}_{i\star}$ have zero mean and equal covariance. According to the considerations given in appendix A, this assures independence up to fourth order for the elements of $\tilde{\mathbf{v}}$. Thus, $\mathfrak{W}$ can be estimated from $\tilde{\mathbf{x}}$ using an appropriate fourth order ICA algorithm, e.g. FastICA (Hyvärinen and Oja, 1997).

Once $\mathfrak{W}$ is known one can compute the matrix

$$\mathbf{V}_t = (\mathbf{V}_t^\circ, \mathbf{V}_t^+) = \mathbf{X}_t\mathfrak{W}^T \,, \tag{2.77}$$

where, without loss of generality, $\mathfrak{W}$ may be partitioned as

$$\mathfrak{W} = \begin{pmatrix} \mathfrak{W}^\circ \\ \mathfrak{W}^+ \end{pmatrix} = (\mathfrak{A}^\circ, \mathfrak{A}^+)^{-1} \,. \tag{2.78}$$

The decision what rows of $\mathfrak{W}$ belong to $\mathfrak{W}^\circ$ and what to $\mathfrak{W}^+$ can made with the following heuristic: Features that carry dependencies between the sources must be present with large entropy, otherwise the dependencies would have been low. In the first approximation these are the features with large variance, which in turn correspond to rows of $\mathfrak{W}$ with small Euclidean norm, Thus, the heuristic is to put rows of small norm into $\mathfrak{W}^+$ and discard them. Then, the columns of $\mathbf{V}^\circ$ and $\mathbf{U}^\circ$ constitute mixture vectors $\tilde{\mathbf{v}}'$ and $\tilde{\mathbf{u}}'$, respectively, which are related as $\tilde{\mathbf{v}}' = \mathbf{A}\tilde{\mathbf{u}}'$. Under the assumptions of appendix A the elements of $\tilde{\mathbf{u}}'$ are independent and $\mathbf{W}$ can be estimated from $\tilde{\mathbf{v}}'$ using ICA.

2.6.1 The Two-Step algorithm

The two step multi-dimensional ICA algorithm can be summarized as follows:

- Collect samples $(\mathbf{X}_t)_{t\in \mathrm{I}'\subset \mathrm{I}}$ of the observations, the row vectors of which, $(\mathbf{x}_{t,i\star}^T)_{t\in \mathrm{I}', i\in[1,N]}$, are taken as realizations of a random vector $\tilde{\mathbf{x}}$.
- **1. Step:** Use an appropriate ICA algorithm to estimate the matrix $\mathfrak{W}$ from the realizations of $\tilde{\mathbf{x}}$ so that the elements of $\mathfrak{W}\tilde{\mathbf{x}}$ are as independent as possible.
- Discard rows of $\mathfrak{W}$ with small norm and summarize the remaining M° rows in the matrix $\mathfrak{W}^\circ$.
- For all $t \in \mathrm{I}'$ compute $\mathbf{V}_t = \mathbf{X}_t\mathfrak{W}^{\circ T}$. The column vectors $(\mathbf{v}_{t,\star i})_{t\in \mathrm{I}', i\in[1,M^\circ]}$ are taken as the realizations of the random vector $\tilde{\mathbf{v}}'$.
- **2. Step:** Use an appropriate ICA algorithm to estimate the demixing matrix $\mathbf{W}$ from the realizations of $\tilde{\mathbf{v}}'$.

2.6.2 Experiments

The derivation of the multi-dimensional ICA algorithm requiered a number of assumptions about the source signals, for which it may be difficult to decide whether, or to what degree, they are fulfilled. Thus, in this section some experiments on two datasets will be shown.

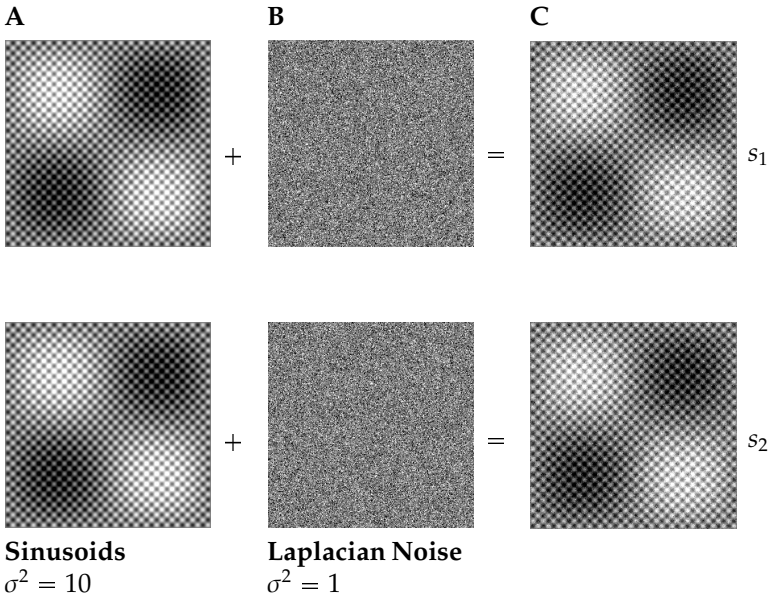


Figure 2.14: The 'noisy sinosoids' dataset. Two dependent source images s_1 and s_2 (**C**) where generarated by adding a sinosoidal signal with two spatial frequencies (**A**) to Laplacian white noise (**B**). Signal (**A**) was identical and signals (**B**) where independent in both sources. The dependent signal component had 10 times larger variance than the independent component.

The first one, the 'noisy sinosoids' dataset, consists of two dependent source images of 256×256 pixels size, zero mean and unit variance. They are a superposition of a dependent and an independent component (figure 2.14). The dependent component was an image containing two sinosoids with spatial frequencies $\omega_1 = (2\pi, 2\pi)$ and $\omega_2 = (20\pi, 20\pi)$ and was equal in both sources (figure 2.14.A). The independent component were two Laplacian white noise images with one tenth of the variance of the dependent component (figure 2.14.B).

These data where used to test the first ICA step estimating $\mathfrak{W}$ and to see in how far equations (2.68) and (2.69) hold. $5 \cdot 10^4$ patches of size 6×6 where taken from both source images at random positions to produce samples for the random vector $\tilde{\mathbf{s}}$ on which *FastICA* was performed to estimate $\mathfrak{W}$. Actually $\tilde{\mathbf{s}}$ is an overcomplete mixture of independent components because at least 36 components are contributed already by the white noise component, and an unknown number are additionally contributed by the sinosoidal signal part. ICA, however, can reconstruct at most 36 components. ICA succeds to separate all 5 components coming form the dependent source part[7] from the majority of the low variance components coming from the independent source part, which have to 'share' the remaining 31 components. Figure 2.15.C shows the Euklidean length of the rows of $\mathfrak{w}_{j\star}$ together with the mutual information $I(U_{1j}, U_{2j})$, which was estimated by means of an histogram estimator. One can see that 5 components with small norm carry high mutual information, where the remaining 31 components have nearly no dependencies.

In the second dataset the sources were 8 passport photographs of 150×150 pixels size, normalized to zero mean and unit variance. They are shown in figure 2.16.A . These are interesting data, because here the source dependencies are not generated artificially as

[7] When taking patches only form the dependent source component, figure 2.14.A, then these span a 5 dimesional space.

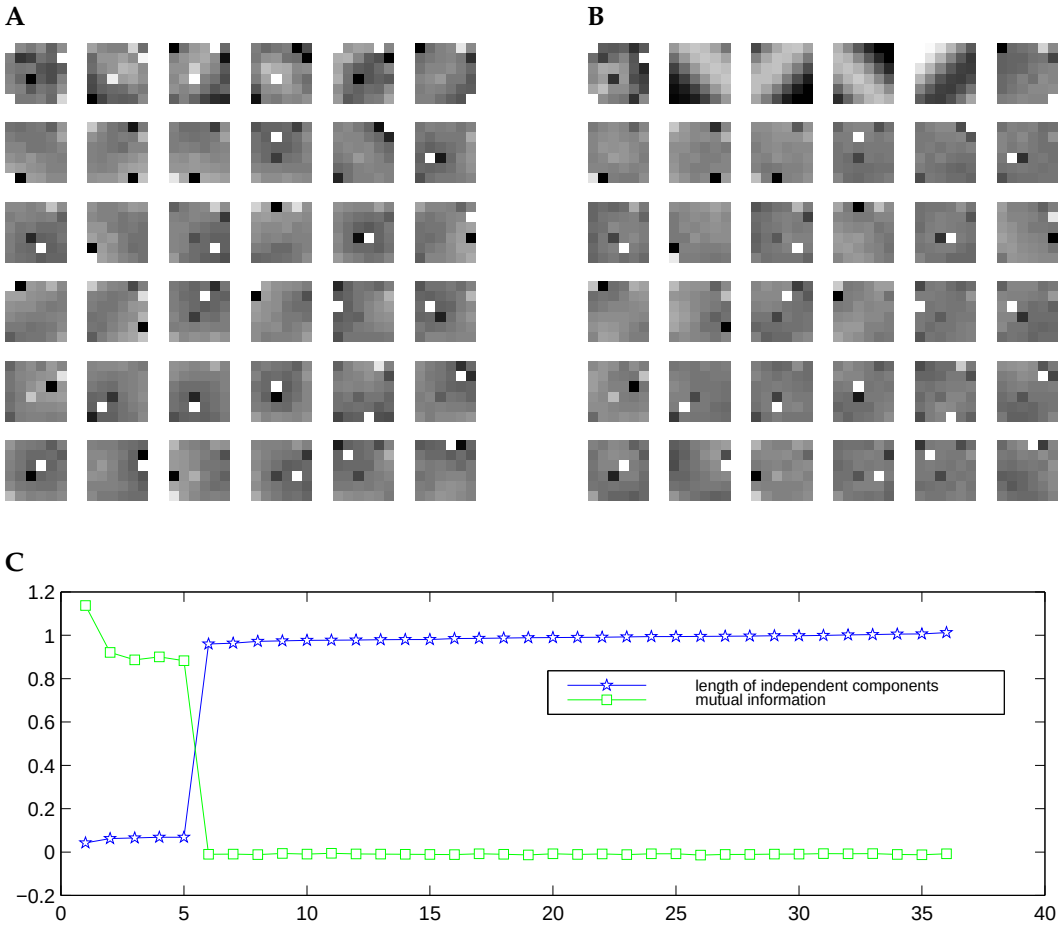


Figure 2.15: **A:** Rows of the matrix $\mathfrak{W}$ estimated using FastICA from $5 \cdot 10^4$ patches of size 6×6 of the 'noisy sinosoids' dataset (figure 2.14.C). The rows are sorted by increasing Euklidean norm from left/top to right/bottom. **B:** The corresponding columns of the matrix $\mathfrak{W}^{-1}$. **C:** Euklidean length of $\mathfrak{w}_{j\star}$ (stars) and the mutual information $I(U_{1j}, U_{2j})$ (squares) for all $j = 1 \ldots 36$. The components with small norm $\mathfrak{w}_{j\star}$ are those with large inter-source dependencies.

in the 'noisy sinosoids' dataset, but due to similar brightness distributions. Correlation coefficients between the source images were in the range from 0.4 to 0.9. The images were linearly mixed using a matrix $\mathbf{A}$, the elements of which were drawn randomly from a normal distribution with mean zero and variance one. The mixing matrix had a condition number of 80. The mixed sources are shown in figure 2.16.B. The source dependencies make standard ICA methods failed to recover the sources: figure 2.16.C shows the results of FastICA. Figure 2.16.D shows the result of the Two-Step multidimensional ICA described in section 2.6.1, which achieved fairly good source separation. For better comparison the images were inverted manually to appear positive.

In the first step $\mathfrak{W}$ was estimated using FastICA on 10^5 patches, 6×6 pixels in size, which were taken with equal probability from random positions from all mixtures. The results of the first ICA are displayed in figure 2.17 analogously to figure 2.15 (see above). The mutual information in panel C was estimated exemplarily for sources 1 and 7 because these were the two with the strongest correlations. Note that in this experiment $\mathfrak{W}$ was estimated from the mixtures. In order to estimate the mutual information for panel C, the second ICA step was anticipated using $\mathbf{U} = \mathbf{A}^{-1}\mathbf{V}$. In the 'passport photo' dataset there is only one component with large mutual information, which resembles the overall brightness of the image patches. The other components are very similar to contrasts in various directions,

Figure 2.16: **A:** The 'passport photos' dataset: 8 images of 150 × 150 pixels size, normalized to zero mean and unit variance, are used as sources. **B:** The sources linearly mixed. **C:** Separation results using the FastICA Matlab package (FastICA,v2.5, 2005). **D:** Separation results using multidimensional ICA (For explanation see text).

which recently have been found by the application of ICA or Sparse Coding to patches of natural images, and are believed to be the coding strategy in V1 (Bell and Sejnowski, 1997; Olshausen and Field, 1996: 1997).

For the second ICA step the first column $\mathbf{v}_{t,\star 1}$ of $\mathbf{V}_t$ was discarded. Then, from the remaining components $\mathbf{v}_{t,\star j}$, 10^4 were chosen randomly and with equal probability, and used as input for FastICA to estimate $\mathbf{W}$. A comparison between figures 2.16.A and D shows that all sources were successfully recovered.

In the next experiment the influence of selecting columns $\mathbf{v}_{\star j}$ prior to the second ICA was examined. Figure 2.18 shows the reconstruction error (cf. appendix B) that could be achieved with the second ICA when only a single component $\mathbf{v}_{\star j}$ served as input. From the previous experiment we have seen that only the first component has considerable dependencies. As expected, only the first one yields poor reconstruction error.

Figure 2.19 shows the reconstruction error vs. M^+ for the case that the M^+ smallest norm rows of $\mathfrak{W}$ (rsp. the corresponding columns $\mathbf{v}_{\star j}$) are discarded. One can see that for all values a good reconstruction is achieved ($re < 0.6$). Even if no row is discarded the result is only slightly worse than for one or two discarded rows. The dependencies of the first component are 'averaged' by the vast majority of components that carry no dependencies in this case. The conspicuous large variance of the error for larger numbers M^+ might be due to convergence instabilities or close to Gaussian distributed columns $\mathbf{u}_{\star j}$. In either case it gives rise to discard as few components as possible.

patch size M	μ_{re}	σ_{re}
2×2	0.4361	0.0383
3×3	0.2322	0.0433
4×4	0.1667	0.0263
5×5	0.1408	0.0270
6×6	0.1270	0.0460

Table 2.2: Separation result of the Two-Step algorithm performed on a set of 8 correlated passport images (cf. figure 2.16, top row). The table shows the average reconstruction error μ_{re} and it's standard deviation σ_{re} calculated from 9 different mixtures.

To evaluate the influence of the patch size M, the Two-Step algorithm was applied

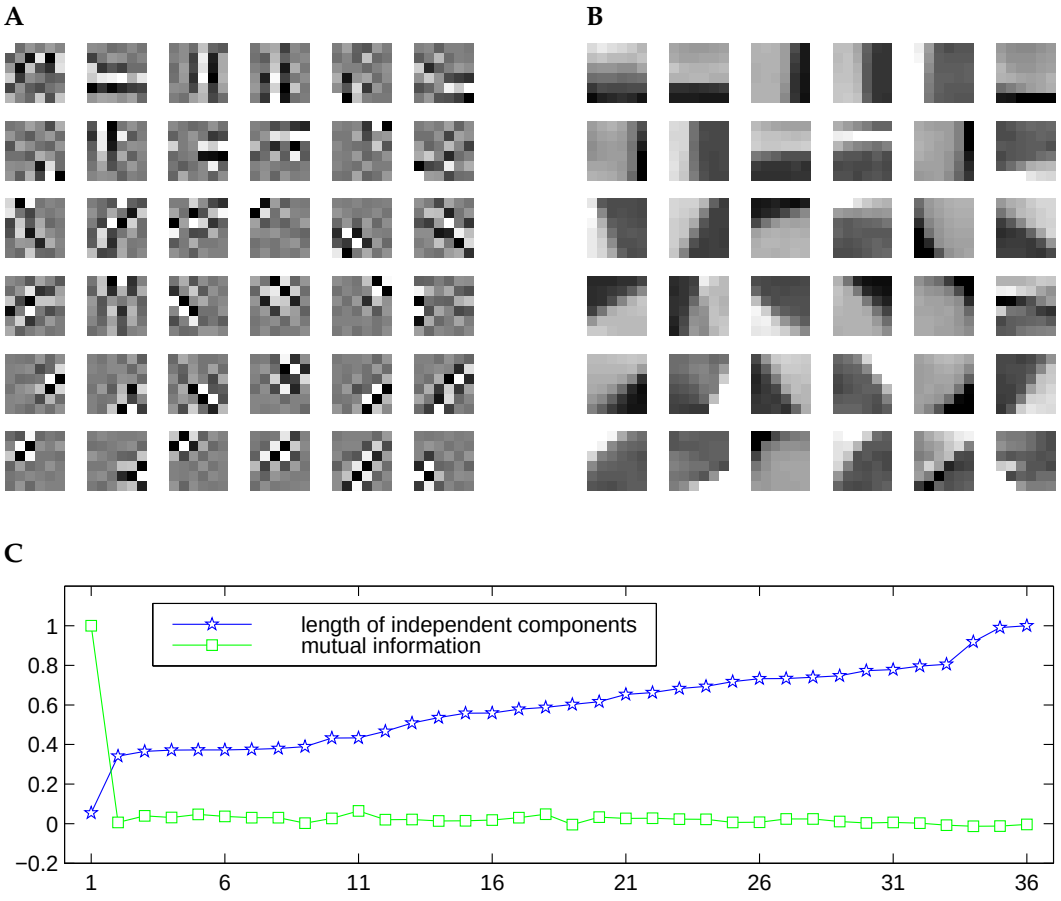


Figure 2.17: **A**: Rows of the matrix $\mathfrak{W}$ estimated using FastICA from 10^5 patches of size 6×6 of a linear mixture of the 'passport photos' dataset (figure 2.16.B). The rows are sorted by increasing Euklidean norm from left/top to right/bottom. **B**: The corresponding columns of the matrix $\mathfrak{W}^{-1}$. **C**: Euklidean length of $\mathfrak{w}_{j\star}$ (stars) and the mutual information $I(U_{1j}, U_{7j})$ (squares) for all $j = 1 \ldots 36$. There is only one component with small norm that carries most of the dependencies between the sources 1 and 7.

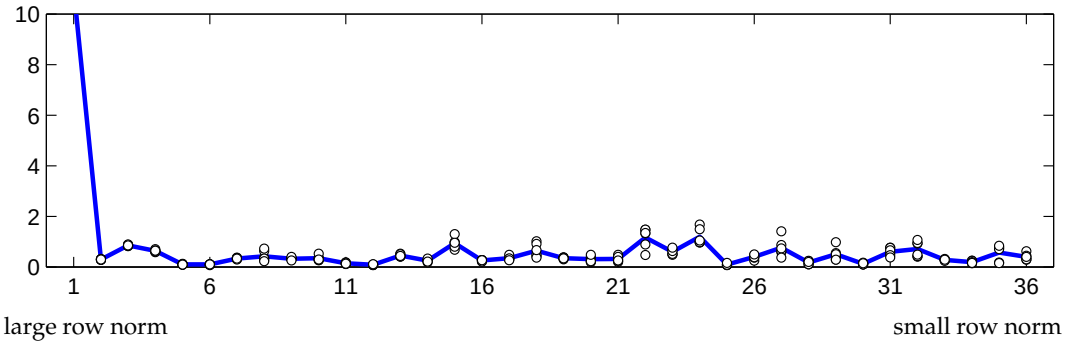


Figure 2.18: Single components $\mathbf{v}_{\star j}$ of the passport photo dataset used to generate input for the second ICA. Only the first (smallest norm) component causes bad reconstruction error for the second ICA step. The reconstruction error (line) was averaged over 6 individual trials (circles) with randomly initialized mixing matrices **A**.

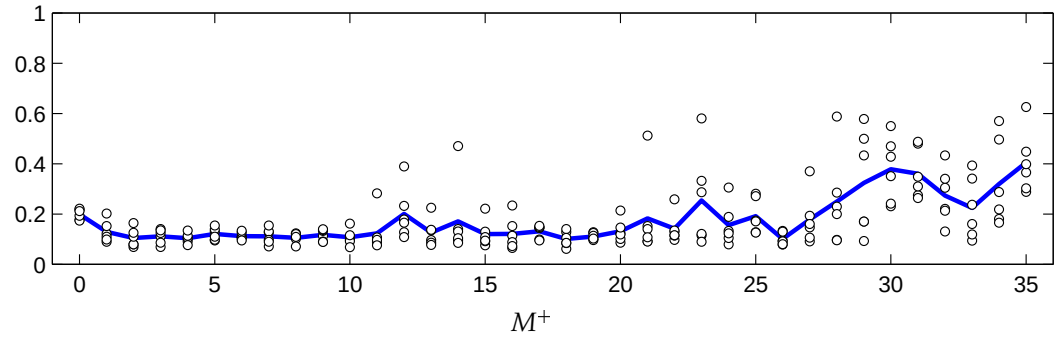


Figure 2.19: The M^+ rows with the smallest norm were discarded. All values of M^+ provide good reconstruction errors in the second step. Note the slightly worse result for $M^+ = 0$. The reconstruction error (line) was averaged over 6 individual trials (circles) with randomly initialized mixing matrices $\mathbf{A}$.

to 9 different mixtures of the passport photo sources using patch sizes between $M = 2 \times 2$ and $M = 6 \times 6$. Table 2.2 shows the mean and standard deviation of the achieved reconstruction error. The elements of the mixing matrix $\mathbf{A}$ were randomly drawn from a normal distribution with mean zero and variance one. FastICA was used for both steps, where 10^5 sample patches were used to extract the optimal features and $2.5 \cdot 10^4$ samples were used to estimate $\mathbf{W}$. The smallest row of $\mathfrak{W}$ was always discarded. The algorithm shows a quite robust performance, and even for patch sizes of 2×2 pixels a fairly good separation result is achieved.

Chapter 3

Optimal Linear Filters

In chapter 2 we defined instantaneous linear functions on real valued stochastic signals and had a closer look at various algorithms for instantaneous linear signal processing. It turned out that it can be quite useful to consider temporal structure and dependencies in the stochastic signal, even if the function to be estimated is an instantaneous one.

Thus, the idea which suggests itself is to incorporate temporal structure also into the signal processing function leading to non-instantaneous functions, so called *filter functions*. This chapter is devoted to the unsupervised learning of optimal linear filters and multi-channel filters.

3.1 Optimal filtering for template detection

Template detection aims on finding time steps t, together with amplitudes v_t, at which the stochastic signal $(x_t)_{t\in\mathrm{I}}$ approximately exhibits a given template $(\xi_\tau)_{\tau\in\mathrm{J}}$ scaled by v_t such that for $\tau \in \mathrm{J}_\xi$

$$x_{t+\tau} \approx v_t \xi_\tau \,.$$

If $(x_t)_{t\in\mathrm{I}}$ is generated by the model

$$x_t = \sum_{\tau\in\mathrm{J}} v_{t-\tau}\xi_\tau + n_t \,, \tag{3.1}$$

where $(v_t)_{t\in\mathrm{I}}$ is a real valued, sparse stochastic signal, and $(n_t)_{t\in\mathrm{I}}$ is a noise signal, then template detection amounts to detecting $(v_t)_{t\in\mathrm{I}}$ from the observed stochastic signal x_t. This procedure is called *deconvolution*.

In the following we assume scalar valued signals, $\mathrm{E} = \mathbb{R}$, and discrete, one-dimensional index sets, $\mathrm{I} = \mathbb{Z}$. The results, however, can be easily generalized for vector valued, multi-channel signals. The later case is explicitly addressed in section 3.3 with the derivation of optimal multi-channel filters for template discrimination.

With regard to the spike sorting applications which are presented below, we assume the occurrence of the templates to be sparse, which means that $v_t = 0$ (rsp. v_t is close to zero, compared to it's variance) with probability close to 1.

The problem of deconvolution can be addressed by the application of an appropriate linear filter. We try to find a filter that responds to the template with a narrow impulse. This way after filtering even partially overlapping templates can be detected because they are transformed into partially overlapping narrow impulses, the amplitudes of which are less distorted.

Figure 3.1.A shows a signal which was generated according to the data model (3.1). The template ξ was linearly superimposed at several times t with $v_t \in \{0,\ 0.5,\ 0.75,\ 1.0\}$. There is no noise in this example. We can see noticeable overlapping templates near time steps 500 and 900, which make it hard to detect them individually and estimate their amplitudes. The output of the optimal filter in this study is shown on panel **B**. The responses to the

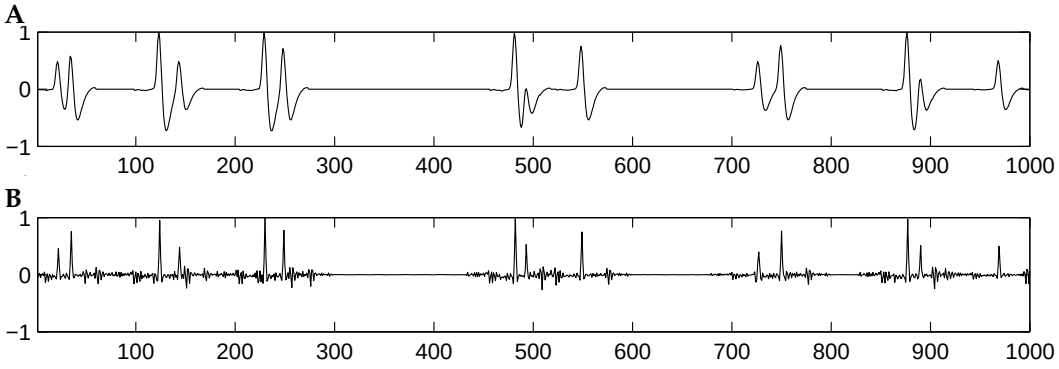


Figure 3.1: **A**: A signal containing spikes with the shape of the template shown in figure 3.4.A. **B:** response of the optimal noiseless filter (cf. figure 3.2, top row, $\alpha = 0$)

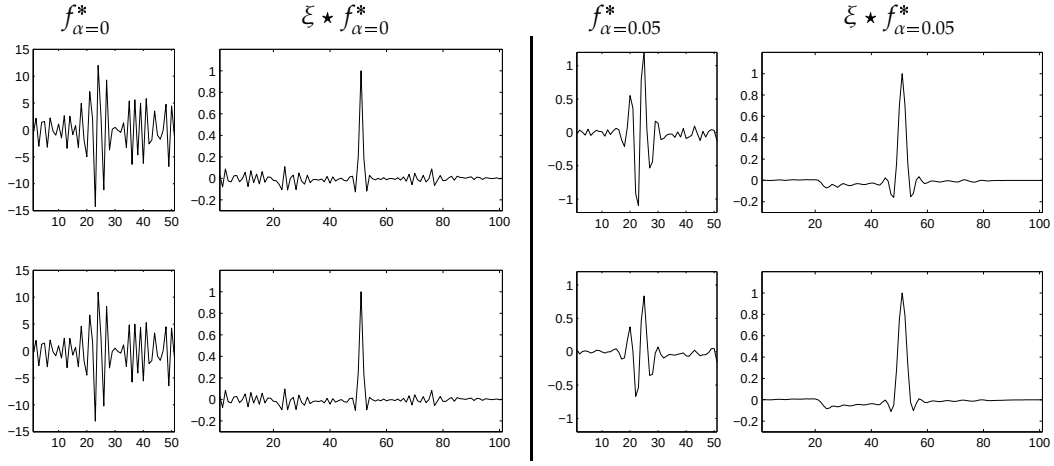


Figure 3.2: Optimal filters to the template shown in figure 3.4.A achieved from constraint optimization (3.6) (*top row*) and from (3.23) (*bottom row*) for values of the regularization parameter $\alpha = 0$ (*left half*) and $\alpha = 0.05$ (*right half*).

templates are sharp peaks, the amplitude of which can easily be detected also at the regions of overlap.

The way in which this filter can be derived in situations where the template is known beforehand, will be the subject of this section. It will provide insight into the relations between the optimal filter and the template. Using this insight we will then develop an algorithm which estimates the optimal filter together with the template using the recorded signal alone.

3.1.1 Optimal filter for a given template

In this section we derive the optimal linear filter f^* that would transform the template ξ into an impulse with minimal width.

The response of a filter $(f_t)_{t\in \mathrm{J}_f}$ to the template $(\xi_t)_{t\in \mathrm{J}_\xi}$ is given by

$$l_\tau = (\xi \star f)_\tau = \sum_{\substack{t\in \mathrm{J}_f \\ t+\tau \in \mathrm{J}_\xi}} \xi_{t+\tau} f_t \,, \tag{3.2}$$

where $\star$ denotes the cross-correlation. ξ may be defined over the closed interval $\mathrm{J}_\xi = [-T_\xi, T_\xi]$ and f over $\mathrm{J}_f = [-T_f, T_f]$. Outside these intervals we assume ξ and f to be zero. Without loss of generality be $\mathrm{J}_\xi = \mathrm{J}_f$. We will also use vector notation for the template and

for the filter, and we indicate vectors with the symbols

$$\bar{\xi} = (\xi_{T_\xi}, \ldots, \xi_{-T_\xi})^T \quad \text{and} \quad \bar{f} = (f_{T_f}, \ldots, f_{-T_f})^T \,.$$

We now require the response l_τ to be as close to zero as possible for all τ and to be equal to 1 at $\tau = 0$. This is a constrained optimization problem and can be solved using the technique of Lagrange multipliers. Minimizing the squared norm of l under the constraint $l_0 = 1$, the corresponding Lagrange equation becomes

$$L = \sum_{\tau=-T_l}^{T_l} l_\tau^2 + \lambda \cdot (l_0 - 1), \qquad T_l = T_f + T_\xi \,. \tag{3.3}$$

The unconstrained optimization problem is quadratic and positive definite while the constraint is linear in f. Thus, a unique solution exists at the stationary point of the Lagrangian and can be found in one step of a Newton iteration (cf. appendix D for $\alpha = 0$). In vector notation the optimal filter is given by

$$\bar{f}^* = \frac{\mathbf{H}^{-1}\bar{\xi}}{\bar{\xi}^T \mathbf{H}^{-1} \bar{\xi}} \,, \tag{3.4}$$

where

$$H_{kl} = 2(\xi \star \xi)_{k-l}$$

is the Hessian of the Lagrange equation w.r.t. f. It is a matrix containing the autocorrelation function of ξ shifted along the main diagonal.

3.1.2 Problems with white Gaussian noise

The amplitude of the filter can be several orders of magnitude higher than the amplitude of the wave form. This effect becomes apparent when the signal is corrupted with noise. Figure 3.3.A shows the signal of figure 3.1.A, but corrupted with white Gaussian noise with standard deviation of just 2% of the maximum peak amplitude. Figure 3.3.B shows the corresponding response after application of the optimal filter according to equation (3.4). Whereas in the noiseless case (figure 3.1.B) one obtains nicely pronounced, narrow peaks for every spike, a little bit of noise already gives rise to strong artifacts such that the response gets useless. In the course of filtering additive, white Gaussian noise n_t becomes

$$(n \star f)_t = \sum_{\tau=-T_f}^{T_f} n_{t+\tau} f_\tau \,.$$

Because the noise is white, the variance $\sigma^2_{n\star f}$ of the noise after filtering is

$$\sigma^2_{n\star f} = \sigma_n^2 \sum_{\tau=-T_f}^{T_f} f_\tau^2 \,. \tag{3.5}$$

The variance $\sigma^2_{n\star f}$ grows linearly with the squared Euclidean norm of $\bar{f}$. In order to keep noise artifacts small, we add a regularization term to the Lagrangian, equation (3.3), that introduces a bias in favor of vectors $\bar{f}$ with a small norm. The Lagrangian becomes

$$L = \sum_{\tau=-T_l}^{T_l} l_\tau^2 + \alpha \sum_{t=-T_f}^{T_f} f_t^2 \quad + \lambda(l_0 - 1) \,, \tag{3.6}$$

where α is the non-negative regularization parameter, which allows to trade low variance $\sigma^2_{n\star f}$ in the output noise for the sharpness of the peak in the filter response. The optimal filter is given by equation (3.4), but with 2α added to the diagonal of $\mathbf{H}$,

$$H_{kl}(\alpha) = 2(\xi \star \xi)_{k-l} + 2\alpha\, \delta_{k-l} \,. \tag{3.7}$$

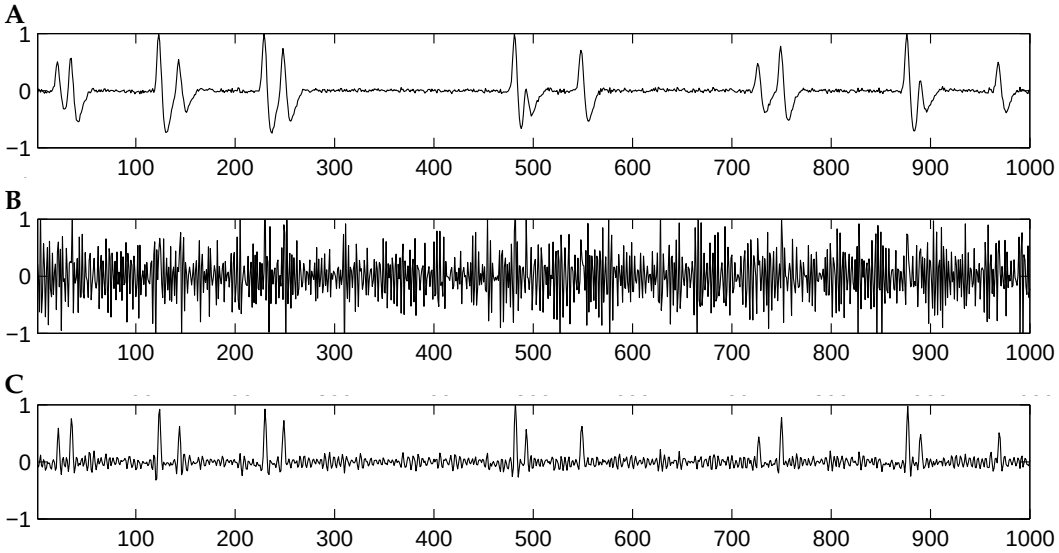


Figure 3.3: Influence of noise to the response of the optimal filter. **A:** Signal corrupted with white, Gaussian noise with $\sigma_n = 0.02$. **B:** Response of the optimal filter (figure 3.2, top row, $\alpha = 0$). **C:** Response of the optimal filter that takes noise into account (figure 3.2, top row, $\alpha = 0.05$).

For the derivation see appendix D.

The trade off between the level of noise and the concentration of the filter response is illustrated in figure 3.4.B . The norm $|\bar{f}|$ of the optimal filter for the template shown in figure 3.4.A and the deviation of l from a delta pulse, $\sum_\tau l_\tau^2 - 1$, are plotted as functions of the regularization parameter α. If α is close to zero, the filter reaches its maximally peaked response, but the level of noise is high. For large values of α, $\mathbf{H}$ is almost diagonal, and the filter approaches the pseudo inverse $\bar{f} = (\bar{\xi}^T\bar{\xi})^{-1}\bar{\xi}$ of $\bar{\xi}$. The pseudo inverse provides the maximum signal to noise distance in its output, but the response of the filter is not specifically concentrated at $\tau = 0$ in this case.

For the noisy signal in figure 3.3, we computed the optimal filter with $\alpha = 0.05$. This amplifies the noise only by a factor of ≈ 2.5 and still yields narrow peaks (cf. figure 3.3.C). Figure 3.2, top row, shows this filter, together with the filter for $\alpha = 0$. The regularized filter is smoother and has lower amplitude, than the noiseless optimal filter.

3.2 Optimal filtering for an unknown template

In the previous section we showed how an optimal filter can be detected if the actual template ξ is known. In practice, however, this is usually not the case. In the following section we will show how the Hessian $\mathbf{H}$ can be estimated from the data alone making use of the particular statistical properties of the recorded signal. Then, in sections 3.2.2 and 3.2.3 an unsupervised learning method will be presented that is able to find an almost optimal filter without knowing the template by means of ICA techniques. From this filter, together with $\mathbf{H}$, an approximate template can be inferred.

3.2.1 Estimating the Hessian from the data

From the data generated according to equation (3.1) one can estimate $\mathbf{H}$ by means of the autocorrelation function of the recorded signal. This fact is summarized in the following theorem:

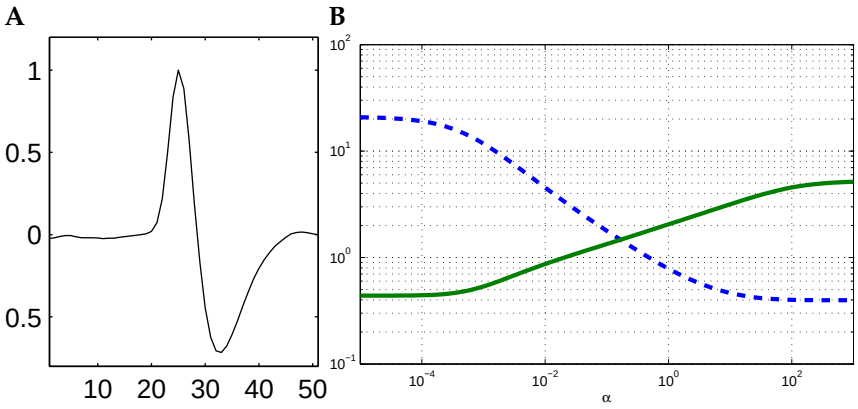


Figure 3.4: **A:** The template wave form that was used to produce the toy examples in figures 3.1 and 3.3. **B:** Influence of the regularization parameter α to the optimal filter. Optimal filters to the template shown left have been computed for various values of α. *Dashed line:* Euclidean length of the optimal filter. *Solid line:* $\sum_\tau l(\tau)^2 - 1$.

Theorem 3.1 *For data generated according to the model*

$$x_t = \sum_\tau v_{t-\tau}\xi_\tau + n_t \,,$$

where $(v_t)_{t\in\mathrm{I}}$ *is a stationary, white signal with variance* $\sigma_v^2 = \left\langle v_t^2 \right\rangle - \left\langle v_t \right\rangle^2$*, the following holds*

1. *For the regularization parameter* α *assuming the value*

$$\alpha = \alpha^* = \frac{\sigma_n^2}{\sigma_v^2} \,, \tag{3.8}$$

the Hessian given by equation (3.7) is

$$H_{kl}(\alpha^*) = 2\frac{\left\langle x_{t_k} x_{t_l} \right\rangle - \left\langle x_{t_k} \right\rangle \left\langle x_{t_l} \right\rangle}{\sigma_v^2} \,, \quad t_k - t_l = k - l \,. \tag{3.9}$$

2. *The filter*

$$\bar{f}(\alpha^*) = \frac{\mathbf{H}^{-1}(\alpha^*)\bar{\xi}}{\bar{\xi}^T \mathbf{H}^{-1}(\alpha^*)\bar{\xi}} \tag{3.10}$$

yields the filtered signal $x \star f$ *with the smallest variance.*

Note: This theorem states that it is possible to derive the Hessian for one particular value α^*, equation (3.8), from the data alone and that α^* yields the optimal tradeoff between noise robustness and peak sharpness (cf. figure 3.4.B) in terms of the output variance of the filtered signal[1]. Minimal output variance is the criterion for the optimal trade off value α for the following reason: When for a filter $\bar{f}(\alpha)$ only α is changed, according to the constraint in equation (3.6), the peak amplitudes of the filtered templates remain constant. Thus, in the signal with the smallest variance, the peaks are easiest to detect. Taking α too small would increase the output variance due to large noise artifacts. Taking α to large, however, would lead to coarse peaks and, hence, to large values beside the peaks also increasing the variance.

Proof: Let us consider the mixed second central moments of the signal x. If the noise n has zero mean and is uncorrelated to the intrinsic signal v, we obtain

[1]Note that equation (3.10) is not yet sufficient to estimate f from the data since it still depends on ξ. An estimator for f will be derived in the following sections.

$$\begin{aligned}\left\langle x_{t_k}x_{t_l}\right\rangle - \left\langle x_{t_k}\right\rangle\left\langle x_{t_l}\right\rangle &= \left\langle \left(\sum_{\tau_1} v_{t_k-\tau_1}\xi_{\tau_1} + n_{t_k}\right)\left(\sum_{\tau_2} v_{t_l-\tau_2}\xi_{\tau_2} + n_{t_l}\right)\right\rangle \\ &\quad - \left\langle \sum_{\tau_1} v_{t_k-\tau_1}\xi_{\tau_1} + n_{t_k}\right\rangle\left\langle \sum_{\tau_2} v_{t_l-\tau_2}\xi_{\tau_2} + n_{t_l}\right\rangle \\ &= \sum_{\tau_1}\sum_{\tau_2}\xi_{\tau_1}\xi_{\tau_2}\left(\left\langle v_{t_k-\tau_1}v_{t_l-\tau_2}\right\rangle - \left\langle v_{t_k-\tau_1}\right\rangle\left\langle v_{t_l-\tau_2}\right\rangle\right) \\ &\quad + \left\langle n_{t_k}n_{t_l}\right\rangle - \left\langle n_{t_k}\right\rangle\left\langle n_{t_l}\right\rangle\end{aligned}$$

because the mixed covariances of v and n vanish. The term

$$\left\langle v_{t_k-\tau_1}v_{t_l-\tau_2}\right\rangle - \left\langle v_{t_k-\tau_1}\right\rangle\left\langle v_{t_l-\tau_2}\right\rangle$$

is the auto covariance of v at delays $t_k - \tau_1 - (t_l - \tau_2)$. Because the intrinsic signal is temporally white, this term vanishes for all $t_k - \tau_1 \neq t_l - \tau_2$. The noise covariance term vanishes for $t_k \neq t_l$. We obtain

$$\begin{aligned}\left\langle x_{t_k}x_{t_l}\right\rangle - \left\langle x_{t_k}\right\rangle\left\langle x_{t_l}\right\rangle &= \sum_{\tau}\xi_{\tau}\xi_{t_l-t_k+\tau}\left(\left\langle v^2_{t_k-\tau}\right\rangle - \left\langle v_{t_k-\tau}\right\rangle^2\right) + \sigma_n^2\delta_{t_k-t_l} \\ &= \sigma_v^2\cdot(\xi\star\xi)(t_l-t_k) + \sigma_n^2\delta_{t_k-t_l}\,, \qquad (3.11)\end{aligned}$$

Putting equation (3.11) into equation (3.9), the proof of the first part of the theorem is complete.

Be $\bar{\boldsymbol{x}}_t = (x_{t+T_f},\ldots,x_{t-T_f})^T$ a clip of the length of the filter $\bar{f}$ taken from the signal x at time t. Using equation (3.4) we can write the variance of the filtered signal as

$$\begin{aligned}\left(\left\langle(\bar{\boldsymbol{x}}_t^T\bar{f})^2\right\rangle - \left\langle\bar{\boldsymbol{x}}_t^T\bar{f}\right\rangle^2\right) &= \left\langle\left(\bar{\boldsymbol{x}}_t^T\frac{\mathbf{H}^{-1}\bar{\xi}}{\bar{\xi}^T\mathbf{H}^{-1}\bar{\xi}}\right)^2\right\rangle - \left(\left\langle\bar{\boldsymbol{x}}_t^T\right\rangle\frac{\mathbf{H}^{-1}\bar{\xi}}{\bar{\xi}^T\mathbf{H}^{-1}\bar{\xi}}\right)^2 \\ &= \frac{\bar{\xi}^T\mathbf{H}^{-1}\left(\left\langle\bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^T\right\rangle - \left\langle\bar{\boldsymbol{x}}_t\right\rangle\left\langle\bar{\boldsymbol{x}}_t^T\right\rangle\right)\mathbf{H}^{-1}\bar{\xi}}{\left(\bar{\xi}^T\mathbf{H}^{-1}\bar{\xi}\right)^2}\,, \qquad (3.12)\end{aligned}$$

where we made use of the fact that $\left\langle\bar{\boldsymbol{x}}_t\right\rangle$ is a vector of constants. According to theorem C.1 the minimum of equation (3.12) is achieved for equation (3.9), i.e. for $\alpha = \alpha^*$.

□

With theorem 3.1 we have an easy way to estimate the Hessian up to a constant factor containing the unknown quantity σ_v^2. This factor, however, cancels in equation (3.4) and, hence, has not to be estimated. In a practical application we may have the recordings already corrected by their means due to the recording channel. Thus, we can simply compute

$$\hat{\mathbf{H}} = \frac{1}{T_x}\sum_{t=T_f}^{T_x+T_f-1}\bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^T$$

as an estimate for the Hessian.

We now are able to estimate $\mathbf{H}$, which represents the connection between f and ξ. In the next section we will show how f can be derived from the data alone using ICA techniques. Then, together with $\mathbf{H}$, from f ξ can be derived.

3.2.2 ICA and optimal filters

With the optimal filter we try to cancel (as well as possible) the effect of the convolution of the intrinsic signal v with the template ξ, equation (3.1). Consider vectors containing small

signal clips

$$\begin{aligned}\bar{\boldsymbol{x}}_t &= \left(x_{t+T_f}, \ldots, x_{t-T_f}\right)^T , \\ \bar{\boldsymbol{v}}_t &= \left(v_{t+T_\xi+T_f}, \ldots, v_{t-T_\xi-T_f}\right)^T , \\ \bar{\boldsymbol{n}}_t &= \left(n_{t+T_f}, \ldots, n_{t-T_f}\right)^T .\end{aligned}$$

With that we can write the convolution as the matrix multiplication

$$\bar{\boldsymbol{x}}_t = \boldsymbol{\Xi}\bar{\boldsymbol{v}}_t + \bar{\boldsymbol{n}}_t , \tag{3.13}$$

where $\boldsymbol{\Xi}$ is a $(2T_f + 1) \times (2T_\xi + 2T_f + 1)$ matrix which contains the template ξ in every row, shifted by one from one row to the next, $\Xi_{kl} = \xi_{l-k-T_\xi}$. Equation equation (3.13) is the paradigm of *over-complete, noisy ICA*. According to our data model the mixed intrinsic spike trains are temporally white. Hence, the elements of $\bar{\boldsymbol{v}}_t$ can be interpreted as statistically independent sources. The elements of $\bar{\boldsymbol{n}}_t$ can be interpreted as additional sources,

$$\bar{\boldsymbol{x}}_t = \left(\boldsymbol{\Xi}\ \mathbf{I}_{2T_f+1}\right)\begin{pmatrix}\bar{\boldsymbol{v}}_t \\ \bar{\boldsymbol{n}}_t\end{pmatrix} ,$$

where $\mathbf{I}_{2T_f+1}$ is the unity matrix of size $2T_f + 1$ and $\left(\boldsymbol{\Xi}\ \mathbf{I}_{2T_f+1}\right)$ is the concatenation of the two matrices. Thus we have a mixture of $4T_f + 2T_\xi + 2$ sources ($2T_\xi + 2T_f + 1$ from $\bar{\boldsymbol{v}}_t$ and $2T_f + 1$ from $\bar{\boldsymbol{n}}_t$) giving rise to $2T_f + 1$ observations.

It is well known that for over-complete, noise corrupted mixtures a perfect source separation in general does not exist Lewicki and Sejnowski (2000); Lee et al. (1999). For a perfect separation $\bar{\boldsymbol{f}}^T\bar{\boldsymbol{x}}_t = v_t$ would hold. This would mean that the vector $\bar{\boldsymbol{f}}$ is orthogonal to all columns of $\boldsymbol{\Xi}$, but the middle, and orthogonal to all(!) columns of the unity matrix. Clearly such $\bar{\boldsymbol{f}}$ is not realizable. An intuitive measure for how close $\bar{\boldsymbol{f}}$ is to the hypothetic optimal one is the squared sum of its scalar product with all columns of $\boldsymbol{\Xi}$ and the unity matrix under the constraint that the scalar product with the middle column of $\boldsymbol{\Xi}$ (which is $\bar{\boldsymbol{\xi}}$) equals 1. We weight the columns of the unity matrix by some α which shall reflect the strength of the noise compared to the variance of v_t. The Lagrangian of this problem is

$$\begin{aligned}L &= \bar{\boldsymbol{f}}^T\boldsymbol{\Xi}\boldsymbol{\Xi}^T\bar{\boldsymbol{f}} + \alpha\bar{\boldsymbol{f}}^T\bar{\boldsymbol{f}} + \lambda\left((\bar{\boldsymbol{f}}^T\bar{\boldsymbol{\xi}})^2 - 1\right) \\ &= \sum_k\sum_l(\bar{f}_k\Xi_{kl})^2 + \alpha\sum_k\bar{f}_k^2 + \lambda\left(\sum_k(\bar{f}_k\bar{\xi}_k)^2 - 1\right) \\ &= \sum_{\tau=-T_f}^{T_f}\sum_{t=-T_f-T_\xi}^{T_f+T_\xi}(f_{-\tau}\xi_{t-\tau})^2 + \alpha\sum_{\tau=-T_f}^{T_f}f_\tau^2 + \lambda\left(\sum_{\tau=-T_f}^{T_f}(f_\tau\xi_\tau)^2 - 1\right) ,\end{aligned}$$

where $\bar{f}_k = f_{T_f-k+1}$ is the k-th element of $\bar{\boldsymbol{f}}$ and $\bar{\xi}_k = \xi_{T_f-k+1}$ is the k-th element of $\bar{\boldsymbol{\xi}}$. But this is exactly equation (3.6), and we note that the best possible solution to the over-complete, noisy ICA is also the optimal filter. Thus, we can hope to find the optimal filter using ICA on clips of the recorded signal.

3.2.3 Maximum skewness

We will now take a closer look at an ICA-algorithm which is related to *FastICA* Hyvärinen and Oja (1997). We consider at first the noise-free case and in the next section the case where the recordings are corrupted by Gaussian, white noise.

According to our assumptions about the occurrence of the template, v_t is temporally white, i.e. iid. with respect to t. Further, v_t be subject to a non-Gaussian distribution $p(v_t)$ that does not depend on t. Then, there exists a cumulant $\kappa_d[v_t]$ of order $d > 2$ which is different from zero. We assume $\kappa_d[v_t]$ to be finite and positive, without loss of generality.

After the convolution with the template ξ, we obtain

$$\kappa_d[x_t] = \sum_{\tau=-T_\xi}^{T_\xi} \xi_\tau^d \kappa_d[v_t] \,. \tag{3.14}$$

To be invariant to affine transformations, we consider the standardized cumulant $\rho_d := \kappa_d/\kappa_2^{\frac{d}{2}}$. Standardized cumulants of third or higher order can be interpreted as measures of deviation from normality of the underlaying distribution because they are zero for Gaussian distributions. From theorem C.2 follows

$$\left(\sum_\tau |\xi_\tau|^2\right)^{\frac{1}{2}} \geqslant \left(\sum_\tau |\xi_\tau|^d\right)^{\frac{1}{d}} \Longrightarrow \left|\sum_\tau \xi_\tau^2\right|^{\frac{1}{2}} \geqslant \left|\sum_\tau \xi_\tau^d\right|^{\frac{1}{d}} \tag{3.15}$$

$$\Longleftrightarrow \left|\frac{\kappa_d[v_t]}{(\kappa_2[v_t])^{\frac{d}{2}}}\right| \geqslant \left|\frac{\kappa_d[v_t]\sum_\tau \xi_\tau^d}{\left(\kappa_2[v_t]\sum_\tau \xi_\tau^2\right)^{\frac{d}{2}}}\right| = \left|\frac{\kappa_d[x_t]}{(\kappa_2[x_t])^{\frac{d}{2}}}\right| \,. \tag{3.16}$$

The absolute value of the standardized cumulant must decrease as a result of the convolution. Therefore, the objective for the blind deconvolution is to maximize the absolute value of the standardized cumulant of the filtered signal

$$y_t := \sum_{\tau=-T_f}^{T_f} x_{t+\tau} f_\tau = \bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}} \tag{3.17}$$

with respect to the filter $\bar{\boldsymbol{f}}$. Assuming zero mean for the recordings, the corresponding objective function is given by

$$L = |\rho_d[y_t]| = \frac{|\kappa_d[y_t]|}{\kappa_2[y_t]^{\frac{d}{2}}} = \frac{|\kappa_d[y_t]|}{\left(\bar{\boldsymbol{f}}^T \left\langle \bar{\boldsymbol{x}}_t \bar{\boldsymbol{x}}_t \right\rangle \bar{\boldsymbol{f}}\right)^{\frac{d}{2}}} = \frac{|\kappa_d[y_t]|}{\left(\bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}}\right)^{\frac{d}{2}}} \,. \tag{3.18}$$

However, it is more convenient to see this as a constrained optimization problem. Since we are not interested in the norm of $\bar{\boldsymbol{f}}$ and since the above cost function is invariant to it, we are free to chose the norm of $\bar{\boldsymbol{f}}$ such that $\bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}} = 1$ and the denominator in equation (3.18) vanishes. The Lagrangian for this constrained optimization problem is

$$L = |\kappa_d[y_t]| - \lambda(\bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}} - 1) \,. \tag{3.19}$$

In practical applications the cases $d = 3$ and $d = 4$ are the most interesting ones. There the standardized cumulants ρ_3 and ρ_4 represent the skewness (a measure for the deviation from the symmetry) and the kurtosis (a measure for the sparseness) of the underlaying distribution, respectively. For normalized mean and variance, $\kappa_1 = 0$, $\kappa_2 = 1$, the third- and fourth-order cumulants are given by third-, respectively fourth-order central moments alone (up to a constants). This is important for the algorithm as the computation of sample cumulants reduces to the computation of sample moments. Taking derivatives of sample moments is simple and in principle allows for online algorithms. In contrast, for $d > 4$, products of sample moments occur in the computation of sample cumulants, making online implementations difficult.

The solution of the Lagrangian is found at the stationary point, i.e. the gradients w.r.t. $\bar{\boldsymbol{f}}$ and λ vanish. For $d \in \{3, 4\}$ this is given with

$$0 = \frac{\partial L}{\partial \bar{\boldsymbol{f}}} = d \left\langle (\bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}})^{d-1} \bar{\boldsymbol{x}}_t \right\rangle \pm \lambda \mathbf{H} \bar{\boldsymbol{f}} \,, \qquad 0 = \frac{\partial L}{\partial \lambda} = \bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}} - 1 \,.$$

The solution is a fixed point, which can be rapidly found by the repeated iteration of

$$\bar{\boldsymbol{f}} \leftarrow \mathbf{H}^{-1} \left\langle (\bar{\boldsymbol{f}}^T \bar{\boldsymbol{x}}_t)^d \bar{\boldsymbol{x}}_t \right\rangle \,, \tag{3.20}$$

$$\bar{\boldsymbol{f}} \leftarrow \frac{\bar{\boldsymbol{f}}}{\sqrt{\bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}}}} \,. \tag{3.21}$$

For $d = 4$ this is in principle the same fixed point iteration as in Hyvärinen and Oja (1997), and a convergence proof is given there. The only difference is that no sphering is done here as a preprocessing step, so $\mathbf{H}$ appears in the iteration formula.

3.2.4 Relation to optimal filters

After the iteration of equations (3.20) and (3.21) has converged, we obtain a filter f which, for example, maximizes the skewness in it's output. We now have to investigate how f relates to the desired optimal filter and what happens if sensor noise is present. In other words, we have to check whether f really minimizes equation (3.6).

The second- and d-th-order cumulants of the signal y after filtering are given by

$$\kappa_2[y_t] = \kappa_2[v_t] \sum_\tau l_\tau^2 + \sigma_n^2 \sum_\tau f_\tau^2 \,, \qquad \kappa_d[y_t] = \kappa_d[v_t] \sum_\tau l_\tau^d \,.$$

Because white Gaussian noise has no higher order (> 2) cumulants, there is no such term for the noise. The objective function equation (3.18) can now be written as

$$L = \frac{\kappa_d[v_t] \sum_\tau l_\tau^d}{\left(\kappa_2[v_t] \sum_\tau l_\tau^2 + \sigma_n^2 \|\bar{\boldsymbol{f}}\|^2\right)^{\frac{d}{2}}} \,.$$

Since we expect a positive standardized cumulant ρ_d, L is positive at least in the neighborhood of its maximum, and we can omit the $|\cdot|$ function. Maximizing L is equivalent to minimizing

$$L' = \frac{\kappa_d[v_t]^{\frac{2}{d}}}{\kappa_2[v_t]} L^{-\frac{2}{d}} = \frac{\sum_\tau l_\tau^2 + \frac{\sigma_n^2}{\kappa_2} \|\bar{\boldsymbol{f}}\|^2}{\left(\sum_\tau l_\tau^d\right)^{\frac{2}{d}}} \,. \tag{3.22}$$

Because this function is invariant under multiplication of $\bar{\boldsymbol{f}}$ with any scalar, we can restrict the set of solutions to those that yield a denominator equal to 1. Thus, we have to

$$\text{minimize} \qquad \sum_\tau l_\tau^2 + \alpha \|\bar{\boldsymbol{f}}\|^2 \qquad \text{s.t.} \quad \sum_\tau l_\tau^d = 1 \,, \tag{3.23}$$

where $\alpha = \sigma_n^2 / \kappa_2[v_t]$. The cost function has the same form as equation (3.6). Also the value for the regularization parameter α is the optimal one (cf. theorem 3.1). Only the constraint $\sum_\tau l_\tau^d = 1$ is different from the constraint $l_0 = 1$, but due to the power of d it is quite similar for a narrow impulse response (small l_τ for $\tau \neq 0$).

Figure 3.2 shows the optimal filters for the template of figure 3.4.A computed for two different values of α from equation (3.23) with $d = 3$ and from equation (3.6). Equation (3.23) was solved indirectly through minimizing equation (3.22) using gradient descent. For comparison, the resulting filters have been scaled to yield a peak value of $l_0 = 1$ rather than $\sum_\tau l_\tau^3 = 1$. One can see that for $\alpha = 0$ the filters and also the impulse responses are almost identical. For $\alpha = 0.05$ the filters are still very similar. But we note that the maximum skewness filter has a somewhat smaller amplitude for the same value of α. This is an effect of the constraint in equation (3.23), which allows for slightly broader peaks.

Although it is actually the solution of equation (3.6) and not of equation (3.23), $\bar{\boldsymbol{f}} = (\mathbf{H}^{-1}\bar{\boldsymbol{\xi}})/(\bar{\boldsymbol{\xi}}^T \mathbf{H}^{-1} \bar{\boldsymbol{\xi}})$ can be used as a connection between f and ξ. Simple matrix calculus yields

$$\bar{\boldsymbol{\xi}} = \frac{\mathbf{H}\bar{\boldsymbol{f}}}{\bar{\boldsymbol{f}}^T \mathbf{H} \bar{\boldsymbol{f}}} \,. \tag{3.24}$$

When this is inserted into equations (3.20) and (3.21), a fixed point iteration for $\boldsymbol{\xi}$ is obtained

$$\bar{\boldsymbol{\xi}} \;\leftarrow\; \left\langle \left(\bar{\boldsymbol{\xi}}^T \mathbf{H}^{-1} \bar{\boldsymbol{x}}_t\right)^2 \bar{\boldsymbol{x}}_t^T \right\rangle \tag{3.25}$$

$$\bar{\boldsymbol{\xi}} \;\leftarrow\; \frac{\bar{\boldsymbol{\xi}}}{\sqrt{\bar{\boldsymbol{\xi}}^T \mathbf{H} \bar{\boldsymbol{\xi}}}} \,. \tag{3.26}$$

Note that the constraint in equation (3.23) does not guarantee the peak of the filter response to be in the center. According to the permutation ambiguity of ICA, the algorithm yields the solution for an arbitrary column of Ξ, which contain the template shifted by an arbitrary amount. In the conducted numerical experiments however it was observed that the templates were always more or less covered by the resulting filters, i.e. the filters were not clipped due to boundary effects.

3.3 Optimal multi-channel filters for template discrimination

3.3.1 Template discrimination

In section 3.1 we derived a noise robust, optimal linear filter that responds to a given template with a narrow impulse and, hence, is suitable to detect the occurrence of the template. The question arises whether such filters could also allow to distinguish between different templates in the sense that every filter responds to its 'tuned-in' template with a narrow impulse and suppresses all other templates (and possibly any noise).

A little experiment shows that the single-template filters according to equation (3.4) obviously do not fulfill this task. Consider the two different templates shown in figure 3.5.A. The panels below **B** show the shape of the two filters that optimally detect the corresponding template under noise conditions ($\alpha = 0.05$). From looking at the panels below **C**, which show the responses of all filters to all templates, one can see that both filters respond to their own template with a sharp impulse of unit amplitude, but hardly suppress the other templates.

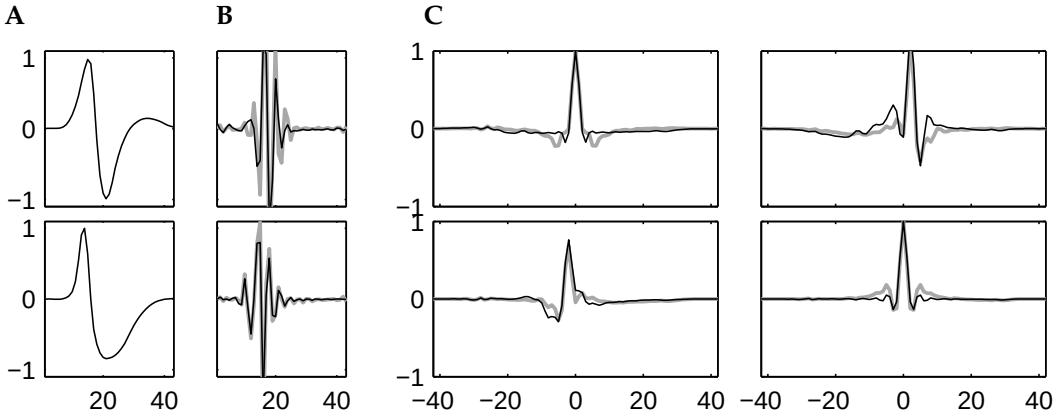


Figure 3.5: Attempt to discriminate two different templates with linear filters. **A:** Two slightly different templates. **B:** Individual optimal filters according to equations (3.4) and (3.7) with $\alpha = 0.05$. **C:** The response of the filters to the corresponding templates (diagonal) and non-corresponding templates (off-diagonal). Both filters cannot discriminate the two templates. *Underlaying gray line:* Optimal discrimination filters derived according to equation (3.46) and their responses. For the explanation of the bad performance see text.

Thus, the derivation of the filters must be adopted so that they do not only suppress shifted versions of it's own template, but also shifted and unshifted versions of all other templates. Two different approaches therefore are given in the following sections.

The derivation of the optimal filter for a single template in section 3.1 was done for convenience, but without true loss of generality, for scalar valued signals, $x_t \in \mathrm{E} = \mathbb{R}$. At this point one should note that template discrimination by means of linear filters usually does not yield satisfactory results for scalar valued signals and templates. The gray lines in panels **B** and **C** of figure 3.5 are the results for optimal discriminating filters as they will be derived in the following (according to equation (3.46)). They are hardly better than the single-template filters. This can be explained as follows: Imagine there are M different templates, one of which the filter shall respond to when it is exactly aligned, the

others shall be fully suppressed. M templates give rise to M intrinsic signals $(v_t^i)_{t\in\mathrm{I},\, i\in[1,M]}$ which determine the occurrence of the templates. In relation to equation (3.13) and the argumentation below, the number of columns of $\boldsymbol{\Xi}$ increases to $M(2T_f + 2T_\xi + 2) - 1$ while the dimensionality of the source separation problems remains $2T_f + 1$. Thus, the problem gets harder.

If, however, we have vector-valued signals, $\mathbf{x}_t, \boldsymbol{\xi}_t, \boldsymbol{f}_t \in \mathrm{E} = \mathbb{R}^N$, then the 'overcompleteness' of the source separation problem relatively reduces with increasing N. The ratio between the number of observations and the number of sources,

$$\frac{M(2T_f + 2T_\xi + 1) + (2NT_f + 1) - 1}{2NT_f + 1}$$

gets closer to one.

In the following we will refer to vector-valued signals as *multi-channel* signals, where every vector component is one channel. We indicate the presence of multiple channels with bold faced symbols, for example $\mathbf{x}$ or $(\mathbf{x}_t)_{t\in\mathrm{I}}$, where $(x_{k,t})_{t\in\mathrm{I}}$, or shortly x_k, denotes the k-th channel of $\mathbf{x}$. Different multi-channel signals are distinguished with superscripts, for example $(\boldsymbol{\xi}_\tau^i)_{\tau\in\mathrm{J}_\xi}$, or shortly $\boldsymbol{\xi}^i$, denotes the i-th template, where ξ_k^i or $(\xi_{k,t}^i)_{t\in\mathrm{J}_\xi}$ is the k-th channel of the i-th template.

3.3.2 Filter for given templates – derivation in the frequency domain

The first approach to optimal linear multi-channel filters for template discrimination goes back to Roberts and Hartline (1975). They made use of the convolution theorem in order to compute the response of the j-th filter to the i-th template in the frequency domain,

$$\hat{l}^{ij}(\omega) = \sum_k \hat{\xi}_k^i(\omega) \left(\hat{f}_k^j(\omega)\right)^C . \tag{3.27}$$

The hat symbol denotes the corresponding quantity in the frequency domain, achieved through discrete Fourier transformation, e.g.

$$\hat{\xi}_k^i(\omega) = \frac{1}{T_\xi} \sum_{\tau=0}^{T_\xi - 1} \xi_{k,\tau}^i \exp(-2\pi \mathrm{j}\omega) .$$

$(\cdot)^C$ denotes a complex conjugate. $\hat{l}^{ij}(\omega)$ is the Fourier transform of a sum of cyclic cross correlations and implies that $\boldsymbol{\xi}^i$, $\boldsymbol{f}^j$ and l^{ij} have common length T. The objective for optimization is to minimize the power of the response l^{ij} for $i \neq j$, together with the average power of the response to a stationary, ergodic noise process $\mathbf{n} = (n_{k,t})_{t\in\mathrm{I},k\in[1,N]}$. At the same time the filters shall provide unit amplitude of the response for $i = j$.

According to Parseval's theorem, the power of l^{ij} can be calculated in the time domain as well as in the frequency domain.

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1} l^{ij}(t)^2 &= \frac{1}{T^2}\sum_{\omega=0}^{T-1} \left|\hat{l}^{ij}(\omega)\right|^2 \\ &= \frac{1}{T^2}\sum_{\omega=0}^{T-1}\sum_{kl} \left(\hat{f}_k^j(\omega)\right)^C \hat{\xi}_k^i(\omega) \left(\hat{\xi}_l^i(\omega)\right)^C \hat{f}_l^j(\omega) \end{aligned} \tag{3.28}$$

Similarly, the noise power can be calculated in the frequency domain.

$$\begin{aligned} \sigma_j^2 &= \left\langle \frac{1}{T^2}\sum_{\omega=0}^{T-1} \left|\sum_k \hat{n}_k(\omega) \left(\hat{f}_k^j(\omega)\right)^C\right|^2 \right\rangle \\ &= \frac{1}{T^2}\sum_{\omega=0}^{T-1}\sum_{kl} \left(\hat{f}_k^j(\omega)\right)^C \hat{C}_{kl}(\omega) \hat{f}_l^j(\omega) , \end{aligned} \tag{3.29}$$

where $\hat{C}_{kl}(\omega) = \left\langle \hat{n}_k(\omega) \left(\hat{n}_l(\omega)\right)^C \right\rangle$ is the average cross power spectrum of the noise signals n_k and n_l. Minimizing (3.28) and (3.29) leads to the following constrained optimization problem:

$$\min \quad \sum_{\omega=0}^{T-1} \sum_{kl} \left(\hat{f}_k^j(\omega)\right)^C \left(\sum_{i \neq j} \hat{\xi}_k^i(\omega) \left(\hat{\xi}_l^i(\omega)\right)^C + \alpha \hat{C}_{kl}(\omega) \right) \hat{f}_l^j(\omega) \tag{3.30}$$

$$\text{s. t.} \quad l^{ii}(0) = \frac{1}{T} \sum_{\omega=0}^{T-1} \sum_{k} \hat{\xi}_k^i(\omega) \left(\hat{f}_k^i(\omega)\right)^C = 1 \,. \tag{3.31}$$

The value of α controls the balance between template discrimination and noise suppression. The solution of ((3.30)) and ((3.31)) is found using the technique of Lagrange multipliers and is given by

$$\hat{f}_k^j(\omega) = \lambda_j \sum_l \left(\sum_{i \neq j} \hat{\xi}_k^i(\omega) \left(\hat{\xi}_l^i(\omega)\right)^C + \alpha \hat{C}_{kl}(\omega) \right)^{-1}_{kl} \hat{\xi}_l^j(\omega) \,, \tag{3.32}$$

$$\lambda_j = \frac{T}{\sum_{kl} \hat{\xi}_k^j(\omega) \left(\sum_{i \neq j} \hat{\xi}_k^i(\omega) \left(\hat{\xi}_l^i(\omega)\right)^C + \alpha \hat{C}_{kl}(\omega) \right)^{-1}_{kl} \hat{\xi}_l^j(\omega)} \,, \tag{3.33}$$

where $(\cdot)^{-1}_{kl}$ denotes the kl-th element of the inverse of the matrix within the brackets.

3.3.3 Filter for given templates – derivation in the time domain

The drawback of the approach in the frequency domain is that (3.27) is not valid when a long signal is filtered which contains the the template at an arbitrary position. In this situation, the correct response of the filters to the templates is achieved from the non-cyclic cross correlation functions,

$$l_\tau^{ij} = \sum_{k=1}^{N} \sum_{\substack{t \in \mathrm{J}_f \\ t+\tau \in \mathrm{J}_\xi}} \xi_{k,t+\tau}^i f_{k,t}^j \,. \tag{3.34}$$

For the temporal index t, the summation can be assumed over the interval $t = -\infty \ldots \infty$, and $\xi(t) = 0$ for $t < 0$ or $t \geqslant T_\xi$, and $f(t) = 0$ for $t < 0$ or $t \geqslant T_f$. Unlike the approach in the frequency domain, the filter length is not restricted to $T_f = T_\xi$. In contrast to (3.34) the cyclic cross correlation (3.27) leads to the response

$$l_\tau^{ij} = \sum_{k=1}^{N} \left(\sum_{t=0}^{T-\tau-1} \xi_{k,t+\tau}^i f_{k,t}^j + \sum_{t=T-\tau}^{T-1} \xi_{k,t+\tau-T}^i f_{k,t}^j \right) \,, \tag{3.35}$$

where $0 \leqslant \tau < T$, and $(\xi_{k,t}^i)_{t \in \mathrm{J}_\xi}$ and $(f_{k,t}^j)_{t \in \mathrm{J}_f}$ are defined over $\mathrm{J}_\xi = \mathrm{J}_f = [0, T-1]$.

In order to achieve template discrimination, the response of the optimal filter to non-matching templates shall be as small as possible, while the response to the matching template shall be a peak of unit amplitude. At the same time, the filter must not be too noise sensitive. The variance of the response of the j-th filter to a stationary, ergodic, and zero DC noise process $n_k(t)$ is

$$\begin{aligned} \sigma_j^2(\tau) &= {\sigma_j}^2 = \left\langle \left(\sum_k \sum_t n_k(t+\tau) f_k^j(t) \right)^2 \right\rangle \\ &= \sum_{t_1 t_2} \sum_{kl} \left\langle n_k(t_1) n_l(t_2) \right\rangle f_k^j(t_1) f_l^j(t_2) \end{aligned} \tag{3.36}$$

regardless of the shift τ. Thus, the optimal filter in the time domain can be obtained from the solution of the following constrained optimization problem:

$$\begin{aligned} &\text{Minimize} \quad \frac{1}{T_l}\sum_i\sum_{j\neq i}\sum_\tau \left(l^{ij}(\tau)\right)^2 + \alpha\sum_i {\sigma_i}^2 \\ &\text{subject to} \quad l^{ii}(0) = 1\,. \end{aligned} \tag{3.37}$$

Because $l^{ij}(\tau) = 0$ for $|\tau| > \frac{1}{2}(T_\xi + T_f)$, we set $T_l = T_\xi + T_f - 1$ and weight the influence of the noise term by a factor α. The optimization problem (3.37) leads to the Lagrange equation

$$L = \frac{1}{T_l}\sum_i\sum_{j\neq i}\sum_\tau \left(l^{ij}(\tau)\right)^2 + \alpha\sum_i {\sigma_i}^2 + \sum_i \lambda^i\left(\sum_{t,k}\xi^i_k(t)f^i_k(t) - 1\right)\,. \tag{3.38}$$

The solution is found at a stationary point of the Lagrangian, i.e. where the partial derivatives of L with respect to $f^i_k(t)$ and λ^i vanish. We now chose a more convenient vector notation. From the i-th filter, the i-th template and the noise process we construct the vectors

$$\bar{\boldsymbol{f}}^i = (f^i_{1,-T_f}, \ldots, f^i_{1,T_f}, \ldots, f^i_{N,-T_f}, \ldots, f^i_{N,T_f})^T \tag{3.39}$$

$$\bar{\boldsymbol{\xi}}^i_\tau = (\xi^i_{1,-T_f-\tau}, \ldots, \xi^i_{1,T_f-\tau}, \ldots, \xi^i_{N,-T_f-\tau}, \ldots, \xi^i_{N,T_f-\tau})^T \tag{3.40}$$

$$\bar{\boldsymbol{n}} = (n_{1,-T_f}, \ldots, n_{1,T_f}, \ldots, n_{N,-T_f}, \ldots, n_{N,T_f})^T\,, \tag{3.41}$$

with $\xi^i_{k,t} = 0$ for $t < 0$ or $t \geqslant T_\xi$. All three vectors, $\bar{\boldsymbol{f}}^i$, $\bar{\boldsymbol{\xi}}^i(\tau)$, and $\bar{\boldsymbol{n}}$, are of the same length, NT_f. We further define the matrix $\mathbf{H}^i$,

$$\mathbf{H}^i := \frac{1}{T_l}\sum_\tau \bar{\boldsymbol{\xi}}^i_\tau \left(\bar{\boldsymbol{\xi}}^i_\tau\right)^T = \begin{pmatrix} \mathbf{H}^i_{11} & \cdots & \mathbf{H}^i_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{H}^i_{N1} & \cdots & \mathbf{H}^i_{NN} \end{pmatrix}, \tag{3.42}$$

where the matrices $\mathbf{H}^i_{kl}$ summarize the cross-correlation functions of ξ^i_k and ξ^i_l,

$$\begin{aligned} \left(\mathbf{H}^i_{kl}\right)_{t_1t_2} &= \frac{1}{T_l}\sum_\tau \xi^i_{k,t_1-\tau}\xi^i_{l,t_2-\tau} \\ &= \frac{1}{T_l}(\xi^i_k \star \xi^i_l)_{t_1-t_2} \end{aligned}$$

which are shifted along the main diagonals. The symbol $\star$ denotes the non-cyclic cross correlation function. In a similar way we define the noise covariance matrix

$$\mathbf{C} := \left\langle \bar{\boldsymbol{n}}\bar{\boldsymbol{n}}^T \right\rangle = \begin{pmatrix} \mathbf{C}_{11} & \cdots & \mathbf{C}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{N1} & \cdots & \mathbf{C}_{NN}\,, \end{pmatrix} \tag{3.43}$$

where the (t_1, t_2)-th element of the matrix $\mathbf{C}_{kl}$ is given by

$$\begin{aligned} (\mathbf{C}_{kl})_{t_1t_2} &= \left\langle n_{k,t_1} n_{l,t_2} \right\rangle \\ &= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} n_{k,t+t_1-t_2} n_{l,t}\,. \end{aligned}$$

We can compute the empirical expectation over time, because we assumed n_k to be ergodic and stationary.

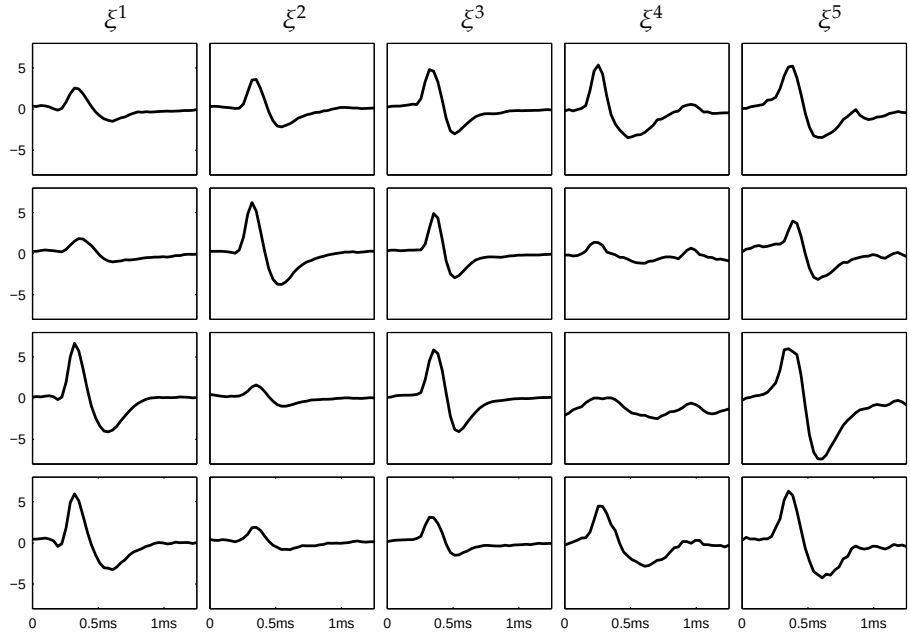


Figure 3.6: Wave form templates averaged from manually identified events from a real tetrode recording. Every column shows the four channels of one of the five templates $\xi^1 \dots \xi^5$.

The Lagrangian can now be formulated in matrix/vector notation,

$$L = \sum_i \left(\vec{f}^i\right)^T \left(\sum_{j \neq i} \mathbf{H}^j + \alpha \mathbf{C}\right) \vec{f}^i + \sum_i \lambda^i \left(\left(\vec{\xi}_0^i\right)^T \vec{f}^i - 1\right) .$$

The partial derivatives with respect to $\vec{f}^i$ and λ^i yield the set of equations

$$\frac{\partial L}{\partial \vec{f}^i} = 2 \left(\sum_{j \neq i} \mathbf{H}^j + \alpha \mathbf{C}\right) \vec{f}^i + \lambda^i \vec{\xi}_0^i = \mathbf{0} , \tag{3.44}$$

$$\frac{\partial L}{\partial \lambda^i} = \left(\vec{\xi}_0^i\right)^T \vec{f}^i - 1 = 0 , \tag{3.45}$$

from which the solution is obtained as

$$\vec{f}^i = \frac{1}{2} \lambda^i \left(\sum_{j \neq i} \mathbf{H}^j + \alpha \mathbf{C}\right)^{-1} \vec{\xi}_0^i , \tag{3.46}$$

$$\lambda^i = \frac{1}{2} \frac{1}{\left(\vec{\xi}_0^i\right)^T \left(\sum_{j \neq i} \mathbf{H}^j + \alpha \mathbf{C}\right)^{-1} \vec{\xi}_0^i} . \tag{3.47}$$

3.3.4 Performance: Frequency domain vs. time domain

In the following both, the frequency domain approach (3.32) and the time domain approach (3.46) to optimal multi-channel filters shall be compared in a spike-sorting application. Therefore, from a small piece extracellular tetrode recordings spike events were manually detected and classified. The data are described more detailed in section 3.5.1. All experiments regarding multi-channel filters were done on the monkey dataset[2] (figure 3.18). The

[2]This is not because of special properties of these data, but just a matter of the availability of data.

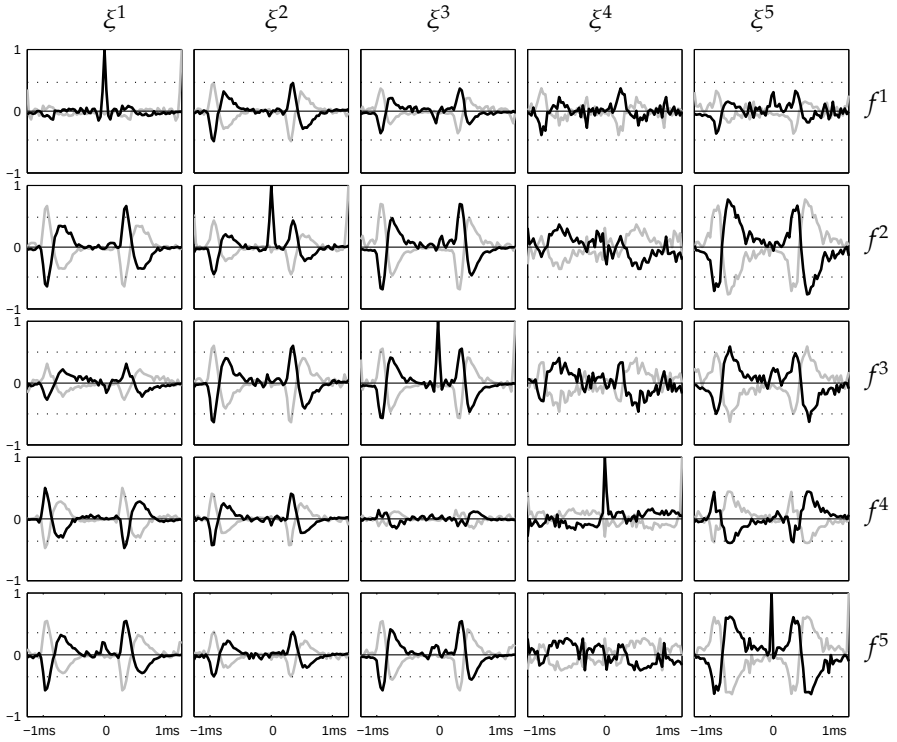


Figure 3.7: Time domain responses (3.34) of the optimal filters derived in the frequency domain to the templates shown in figure 3.6. The panel in the i-th row and j-th column shows the course of $l^{ij}(\tau)$, the response of the j-th filter to the i-th template. Dotted lines indicate the three times standard deviation interval, $\pm 3\sigma_j$, of the noise after filtering. Gray lines show the responses cyclically shifted by T samples in order to illustrate the symmetry of the artifacts that can occur in the frequency domain approach.

detected events were used to estimate five different averaged wave form templates, which are shown in figure 3.6. There were from 13 (ξ^4) up to 110 (ξ^2) sample wave forms available, each of which had a length of $T_\xi = 40$ samples (1.28 ms at 31.250 Hz sampling rate). The noise covariance matrices $\mathbf{C}$ rsp. $\hat{C}$ were estimated from a piece of the recordings with no foreground neural activity.

For better comparison both, the time and frequency domain filters, were computed with $T = T_f = T_\xi = 40$ samples. The parameter α would determine the output noise variance. However, the functional relation between α and σ_i is not known analytically. Thus, in both experiments α was adjusted numerically so that for the output noise held $\max_i \sigma_i(\alpha) = \frac{1}{6}$. Hence, the peak amplitude in the response of matching templates was twice as large as the 3σ interval of the output noise, and peaks could be reliably detected. The adjustment of α was achieved through a simple line search optimization procedure.

Figures 3.7 and 3.8 show the responses calculated according to (3.34) for the filters derived in the frequency and in the time domain, respectively. Due to the constraint the responses to the matching templates equal one at $\tau = 0$. The responses to all other templates shall be minimized. While the filter responses near $\tau = 0$ are essentially the same in both approaches, the frequency domain filter yields much larger responses left and right to the centers. These incorrect values, $\Delta l^{ij}(\tau)$, have the symmetry

$$\Delta l^{ij}(\tau) = -\Delta l^{ij}(\tau \pm T) \tag{3.48}$$

as indicated by the gray lines in figure 3.7. One can see that the incorrect values are shifted by T samples and have opposit signs. This can be understood from equation (3.35). The optimal filter derived in the frequency domain minimizes the sum of the response and the shifted response and, hence, is insensitive for this type of artifacts.

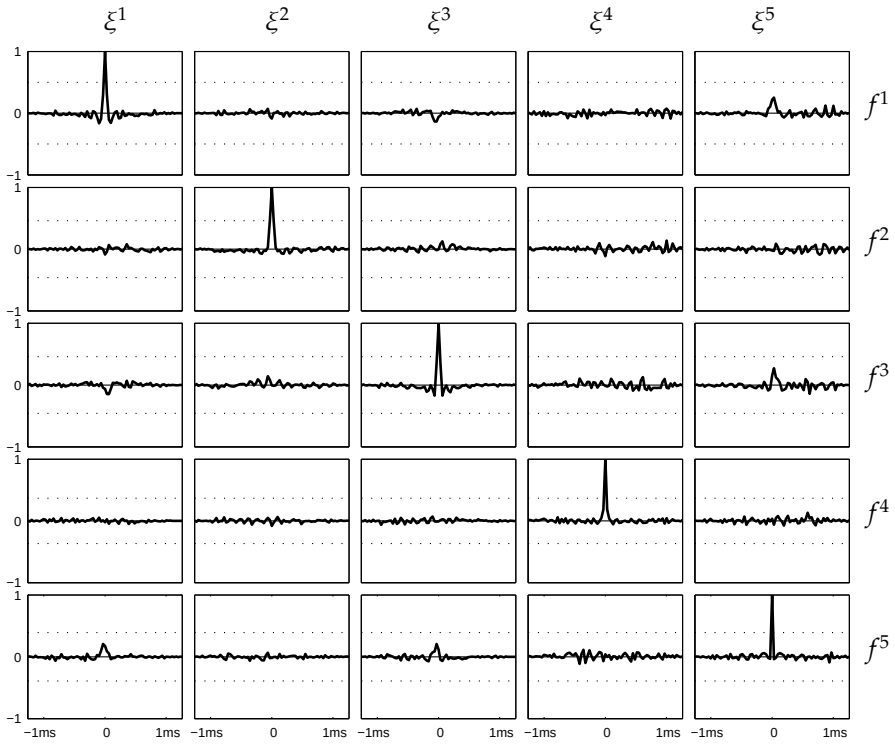


Figure 3.8: Time domain responses (3.34) of the optimal filters to the templates shown in figure 3.6. Dotted lines indicate the three times standard deviation interval, $\pm 3\sigma_j$, of this filters noise output.

In contrast to the frequency domain filters, the time domain filters can have a length different from the templates, $T_f \neq T_\xi$. For the frequency domain approach only longer filters can be derived by zero padding the templates to the desired length. In the following experiment the performance of both approaches for different filter length was compared. Filters with T_f in the range from 10 to 120 samples (0.32 . . . 3.84ms) rsp. 40 to 120 samples were computed with the time and frequency domain methods. For the frequency domain filters, zero padding was done uniformly at both sides of the template. Again, the parameter α was adjusted such that every filter set produced output noise with $\max_j 3\sigma_j = 0.5$. As a measure of performance, the maximum 'non-matching' amplitude and the average 'non-matching' energy were computed from the filter responses,

$$E_1 = \max_{i \neq j, \tau} \left(\left| l^{ij}(\tau) \right| \right) , \quad E_2 = \frac{1}{M(M-1)} \sum_{i \neq j, \tau} l^{ij}(\tau)^2 , \tag{3.49}$$

which are shown in figure 3.9 as functions of T_f. As expected, with increasing filter length the performance of the frequency domain filters approaches the time domain filters. The same performance, however, can always be achieved with much shorter time domain filters.

While the artifacts that are introduced by the cyclic operations in the frequency domain approach can be reduced with appropriate zero padding of the templates, this is at the expense of longer filters. It could be shown that at the same performance the time domain filters can be much shorter – even shorter than the templates. This may be of particular interest for a real time application as the computation cost of the filtering operation is proportional to the filter length. The length of the frequency domain filter can only be reduced when the length of the template is reduced, which discards information contained in the template. For increasing filter length, with the according zero padding, the performance of time and frequency domain filters come close. But it is important to note that the case $T_f = 2T_\xi$ is not at all the case where both approaches are equivalent, which directly follows from (3.34) and (3.35) (cf. also $T_f \geqslant 80$ in figure 3.9). An equivalent solution could be only

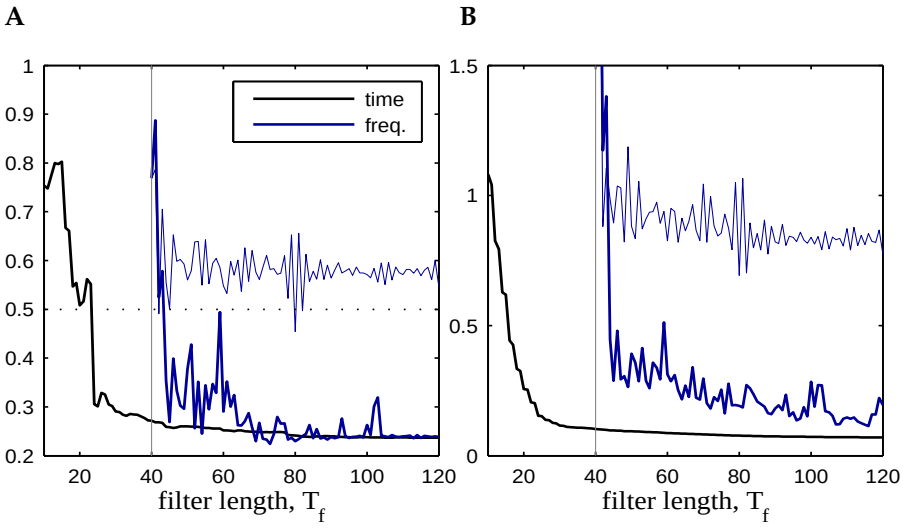


Figure 3.9: The errors **A:** E_1 and **B:** E_2 (cf. equations (3.49)) plotted as functions of the filter length T_f of the time domain filter (black) and the frequency domain filter (blue). For filters longer than $T_\xi = 40$, in the frequency approach the wave form templates were zero padded evenly at both sides (bold line) or from the left side only (thin line) to match the length of the filter T_f. At better performance the time domain approach allows for significant shorter filters than the frequency domain approach.

achieved when also f could be zero padded. This, however is impossible in the derivation (cf. equation (3.32)) of the frequency domain filters. Interestingly, it is crucial how the zero padding of the templates is done. There was almost no performance gain for the frequency domain filter, when the templates were enlarged at one side only, instead of padding them uniformly at both sides (cf. thin blue lines in figure 3.9).

Because of the fact that filters derived in the time domain always show a better performance and can be considerably shorter, one should prefer this approach whenever optimal filters for template discrimination are needed.

3.4 Multi-channel blind deconvolution

We have seen that the optimal multi-channel filter for template discrimination is achieved from the templates in a way very similar to the single-channel filter for template detection (compare equations (3.46) and (3.4)). In section 3.2 we have seen that optimal filters can be derived even for unknown templates. The same would be desirable also for the extension to multiple channels and to multiple templates. We call such an unsupervised learning process *Multi-Channel Blind Deconvolution (MCBDC)*. The term deconvolution is with respect to an assumed data model like in theorem 3.2.1 or, in the multi-channel, multiple templates case,

$$x_{k,t} = \sum_i \sum_\tau v^i_{t-\tau} \xi^i_{k,\tau} + n_{k,t} \,. \tag{3.50}$$

The observed data are the result of convolutions of temporarily white intrinsic signals v^i with multi-channel templates $\boldsymbol{\xi}^i$ and deconvolution is the reverse operation.

3.4.1 Fixed point iteration approach: template detection

In section 3.2 we have seen that the optimal filter for an unknown template can be found by means of an unsupervised fixed point iteration. The question arises whether this is also possible for optimal template discriminating filters. Assuming the generative data model,

equation (3.50), we will first derive a fixed point iteration equivalent to the single template case. Be

$$\bar{\boldsymbol{x}}_t = \left(x_{1,t-T_f}, \ldots, x_{1,t+T_f}, \ldots, x_{N,t-T_f}, \ldots, x_{N,t+T_f}\right)^T .$$

a vector of data patches taken at position t from the input signal. With that a given filter $\boldsymbol{f}^i$ can be used to estimate the corresponding template $\boldsymbol{\xi}^i$ (up to scalings) as the expectation over t

$$\bar{\boldsymbol{\xi}}_0^i \propto \left\langle \theta\left(\bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}}^i\right) \bar{\boldsymbol{x}}_t \right\rangle . \tag{3.51}$$

The template is the weighted sum of the data at those positions t at which a template has been detected. The function θ is a non-linear discrimination function that emphasizes large responses of $\boldsymbol{f}^i$. We assumed, without loss of generality, that $T_f = T_\xi$ so that the vector $\bar{\boldsymbol{\xi}}_0^i$ gives an estimate for all elements of the template $\boldsymbol{\xi}^i$. The new templates are used to recompute the matrices $\mathbf{H}^i$ according to equation (3.42), which in turn allow to compute new filters $\boldsymbol{f}_i$ with (3.46). However, the normalization of the length of $\bar{\boldsymbol{f}}^i$ according to equation (3.47) can spoil the stability of the solution and prevent convergence. The filter and the template are scale invariant except for the condition ${\bar{\boldsymbol{\xi}}_0^i}^T \bar{\boldsymbol{f}}^i = 1$. A larger template is compensated with a smaller filter. With smaller filters, however, the impact of the noise term in (3.37) decreases. Thus, the iteration may yield solutions that are biased to larger templates or even may diverge.

Therefore, it is necessary to do the normalization of $\boldsymbol{f}^i$ and $\boldsymbol{\xi}^i$ not with respect to each other, but with respect to some global quantity. Be $\boldsymbol{f}^i$ normalized to yield unit output variance,

$${\bar{\boldsymbol{f}}^i}^T \mathbf{C}_0 \bar{\boldsymbol{f}}^i = 1 ,$$

where

$$\mathbf{C}_\tau := \left\langle \bar{\boldsymbol{x}}_t \bar{\boldsymbol{x}}_{t+\tau}^T \right\rangle - \left\langle \bar{\boldsymbol{x}}_t \right\rangle \left\langle \bar{\boldsymbol{x}}_{t+\tau}^T \right\rangle . \tag{3.52}$$

Then, the fixed point iteration would include repeated execution of the following steps simultaneously for all i:

$$\bar{\boldsymbol{\xi}}_0^i \leftarrow \left\langle \theta\left(\bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}}^i\right) \bar{\boldsymbol{x}}_t \right\rangle \tag{3.53}$$

$$\mathbf{H}^i \leftarrow \frac{1}{T_l} \sum_\tau \bar{\boldsymbol{\xi}}_\tau^i \left(\bar{\boldsymbol{\xi}}_\tau^i\right)^T \tag{3.54}$$

$$\bar{\boldsymbol{f}}^i \leftarrow \left(\sum_{j \neq i} \mathbf{H}^j + \alpha \mathbf{C}\right)^{-1} \bar{\boldsymbol{\xi}}_0^i \tag{3.55}$$

$$\bar{\boldsymbol{f}}^i \leftarrow \left({\bar{\boldsymbol{f}}^i}^T \mathbf{C}_0 \bar{\boldsymbol{f}}^i\right)^{-\frac{1}{2}} \bar{\boldsymbol{f}}^i \tag{3.56}$$

However, this approach can – and usually does – converge to spurious solutions. These are solutions where all filters, or many of them, are equal, in which case also the corresponding templates are equal. In the present form there is no means that distract the iterations from such fixed points.

Equal filters and templates lead to identical and, hence, 100% correlated outputs. Thus, a step in the right direction would be to modify the scaling such that uncorrelated outputs are guaranteed. Instead of (3.56) the appropriate scaling would be

$$\bar{\mathbf{F}} \leftarrow \bar{\mathbf{F}} \left(\bar{\mathbf{F}}^T \mathbf{C}_0 \bar{\mathbf{F}}\right)^{-\frac{1}{2}} , \tag{3.57}$$

where

$$\bar{\mathbf{F}} := \left(\bar{\boldsymbol{f}}^1 \ldots \bar{\boldsymbol{f}}^M\right) .$$

Consider figure 3.10 for an example of the fixed point approach with scaling rule (3.57). The iterations were performed simultaneously for $M = 5$ templates/filters on a 16s piece of

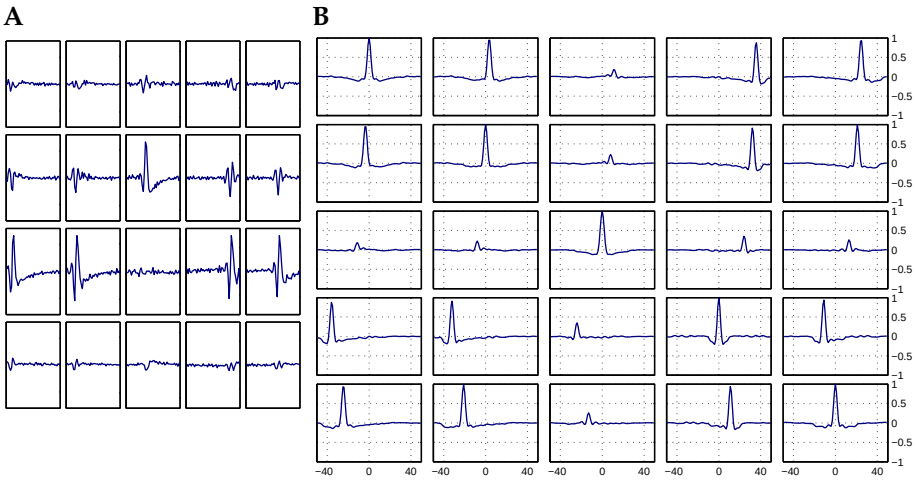


Figure 3.10: Multi-channel filters that emerge in a fixed point iteration in conjunction with the scaling rule (3.57). **A:** The resulting filters. The time course of the k-th channel of the i-th filter is displayed in row k, column i. The filters had a length of 50 samples. **B:** Cross-correlation functions of the filtered signals for shifts $\tau \in [-50, 50]$. The cross-correlation function between outputs i and j is displayed in row i, column j. Although the filtered signals are uncorrelated at $\tau = 0$, filters for one and the same template can emerge several times, shifted by a few samples.

extracellular tetrode recordings (cf. section 3.5.1, figure 3.19). The function θ was the third power of the absolute value of its argument. The expectation in (3.53) was approached by the empirical expectation over $3 \cdot 10^5$ signal clips $\bar{\boldsymbol{x}}_t$ taken from random positions t of the tetrode signal. The matrix $\mathbf{C}_0$ was the empirical covariance matrix of 10^6 such clips. The filters $\vec{\boldsymbol{f}}^i$ and the signal clips had a length of 50×4 samples. Starting with iid., normally distributed elements of $\vec{\boldsymbol{f}}^i$, the whole procedure converged in less than 20 iterations.

One can see in the figure that, although this approach guarantees uncorrelated outputs at $\tau = 0$, the same filters rsp. templates can emerge several times, just shifted by a few samples. Correspondingly, the filtered signals have large cross-correlations at these shifts. In the current example filters for not more than two different templates emerge (cf. columns 1,2,4,5 and column 3 in figure 3.10.A) between them the outputs have small cross-correlation functions (cf. the 3rd row and column in figure 3.10.B). So we may conclude that the fixed point iteration has its focus on the detection of templates rather than their discrimination.

3.4.2 Decorrelation approach: template discrimination

To solve the before mentioned problems, measures must be taken that the filtered signals are not only instantaneously uncorrelated, but also in a sufficiently large range of shifts $\tau \neq 0$. We must relax this a little to 'approximately uncorrelated' because it is in general infeasible to achieve exactly uncorrelated outputs for more than two different shifts. As we have seen in section 2.3.2, approximate decorrelation leads to simultaneous matrix diagonalization problems, which can be efficiently solved by means of the QDIAG algorithm introduced in section 2.4. For the current problem this would involve the diagonalization of the matrices $\mathbf{C}_\tau$ for $\tau = [-T_l, T_l]$.

This approach is illustrated in figure 3.11 with the result of an experiment with the tetrode data. The matrices $\mathbf{C}_\tau$ were the empirical covariance matrices estimated from 10^6 data clips $\bar{\boldsymbol{x}}_t$ rsp. $\bar{\boldsymbol{x}}_{t+\tau}$ of length 50×4 samples. The shifts were in the range $\tau \in [-50, 50]$. QDIAG was used to simultaneously diagonalize the matrices, where $\mathbf{C}_0$ served as the constraint matrix. This way output signals with unit variance were guaranteed.

In figure 3.11.B one can see that with matrix diagonalization almost perfectly uncorrelated output signals can be achieved. However, this solution must be considered as a

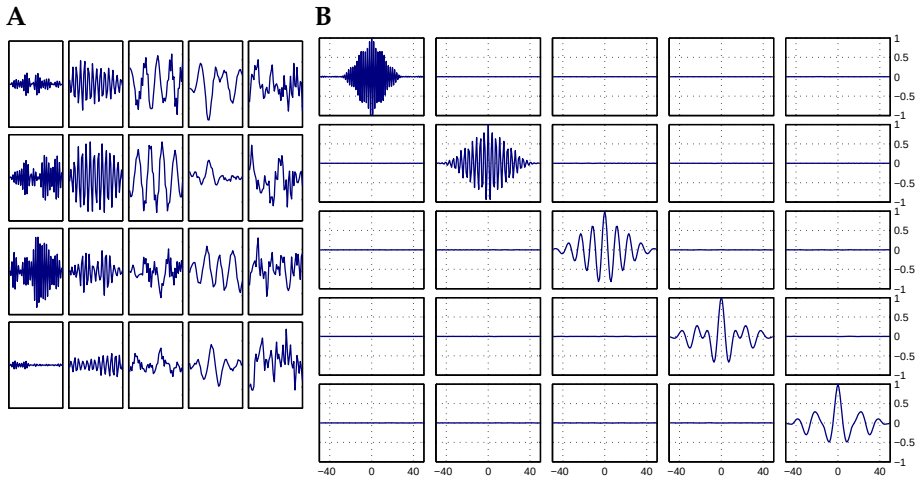


Figure 3.11: Multi-channel filters that emerge when approximate simultaneous matrix diagonalization is performed on the matrices $\mathbf{C}_\tau$ for shifts $\tau \in [-50, 50]$. Diagonalization was done using QDIAG. **A:** Time course of the resulting multi-channel filters. **B:** Cross-correlation functions of the filtered signals. (cf. figure 3.10). Outputs are almost perfectly uncorrelated, but template detection capabilities are low.

trivial one in the context of the current problem. This can be explained as follows: Signals with vanishing cross-correlation functions have disjunctive spectra. With increasing length, linear filters can yield increasingly complex frequency responses. Thus, one can expect that from a certain length on, the filters tune to disjunctive frequency responses, in which case arbitrary input signals would lead to decorrelated output signals.

In fact, from looking at the filters in figure 3.11.A, and the auto-correlation functions in the diagonal of figure 3.11.B, one can notice that different frequencies are predominant in different filters. One can perform a very simple experiment to verify this: Apply the filter to an arbitrary signal, e.g. white Gaussian noise. In this case the filters of figure 3.11.A would still yield signals with small cross-correlation functions not exceeding 14% of the output signals variance.

3.4.3 MCBDC

We have seen that the fixed point iteration approach emphasizes the detection of templates, while the decorrelation approach virtually cares about nothing else than decorrelation – hence discrimination. Thus, to unsupervisedly learn optimal filters for template discrimination, both aspects must be taken into account.

Consider the following constraint optimization problem:

$$\max_{\bar{\mathbf{F}}} \sum_{i=1}^{M} \left\langle \Theta\left(\bar{x}^T \bar{f}^i\right) \right\rangle \quad \text{s.t.} \quad |\bar{f}^{i\,T} \mathbf{C}_\tau \bar{f}^j| \leqslant \epsilon \quad \text{and} \quad |\bar{f}^{i\,T} \mathbf{C}_0 \bar{f}^i| \leqslant 1 \,. \tag{3.58}$$

The first constraint bounds the cross-correlation functions of the outputs to the interval $[-\epsilon, \epsilon]$. It applies to all $i \neq j$ and τ over a sufficiently large range $[-T_l, T_l]$, which covers the central part of the cross-correlation functions. This constraint covers the 'discrimination part' of the problem.

The objective function and the second constraint are responsible for the 'detection part' of the problem. These two alone lead to a solution very similar to the one obtained by the fixed point iteration of section 3.4.1, as we will see shortly.

Consider the sightly modified solution of equations (3.46) and (3.47)

$$\bar{f}^i = \frac{1}{\bar{\xi}_0^{i\,T} \mathbf{H}^{-1} \bar{\xi}_0^i} \mathbf{H}^{-1} \bar{\xi}_0^i \,, \tag{3.59}$$

where

$$\mathbf{H} := \sum_{i=1}^{M} \mathbf{H}^i + \alpha \mathbf{C} \,.$$

In terms of the objective function (3.37) this means that also the responses of every filter to their corresponding templates are minimized. Because the centers of these responses are fixed due to the quadratic constraints $l^{ii}(0) = 1$, minimization leads to peaked responses. Although this was not the primary goal in template discrimination, it can be considered as a well acceptable feature of the algorithm.

The advantage of the summation over all $\mathbf{H}^i$ is that for data that are subject to the generative model (3.50), the matrix $\mathbf{H}$ can be estimated by the empirical covariance matrix of data clips $\bar{\boldsymbol{x}}$, i.e.

$$\mathbf{H} \propto \mathbf{C}_0 \tag{3.60}$$

This follows from the extension of theorem 3.1 to multiple channels and multiple templates. The extension to multiple channels is straightforward. The extension to multiple templates needs the assumption that all intrinsic signals v^i are independent. Then the matrix $\mathbf{H}$ of a multi-template signal is the linear superposition of the individual matrices $\mathbf{H}^i$, because these are second order cumulants of independent single-template signals[3]. However, since we take $\mathbf{H}$ as the equally weighted sum of all $\mathbf{H}^i$, it is implied that all intrinsic signals v^i have equal (w.o.l.o.g. unit) variance $\sigma^2_{v^i}$ (cf. equation (3.9)). In other words: the scale of the template $\boldsymbol{\xi}^i$ is a feature of it, and not one of the intrinsic signals v^i. In fact, for differently scaled templates, equations (3.46) and (3.47) usually yield differently shaped filters.

Equation (3.59) has another advantage over the original form (3.46). It is that there is (up to scalings) a linear relation between $\bar{\boldsymbol{f}}^i$ and $\bar{\boldsymbol{\xi}}^i_0$ which is determined by $\mathbf{C}_0$, the data covariance matrix. If this is inserted into the iterations they reduce to

$$\begin{aligned} \bar{\mathbf{F}} &\leftarrow \mathbf{C}_0^{-1} \left\langle \bar{\boldsymbol{x}}_t \, \theta(\bar{\boldsymbol{x}}_t^T \bar{\mathbf{F}}) \right\rangle \\ \bar{\mathbf{F}} &\leftarrow \bar{\mathbf{F}} \left(\bar{\mathbf{F}}^T \mathbf{C}_0 \bar{\mathbf{F}} \right)^{-\frac{1}{2}} \end{aligned} \tag{3.61}$$

Such iterations are well known from projection pursuit methods that maximize the deviation of its outputs from a Gaussian distribution. In fact, if we do a coordinate transformation with $\mathbf{C}_0^{-\frac{1}{2}}$ and substitute $\bar{\boldsymbol{x}}_t$ with $\mathbf{C}_0^{\frac{1}{2}} \bar{\boldsymbol{x}}_t$ and $\bar{\mathbf{F}}$ with $\mathbf{C}_0^{-\frac{1}{2}} \bar{\mathbf{F}}$ we obtain exactly the FastICA fixed point equations for spherical data (cf. Hyvärinen and Oja (1997)). However, also FastICA cannot provide decorrelated outputs for shifts $\tau \neq 0$.

Therefore, the cross-correlation function constraint was introduced in (3.58). The objective function is nonlinear, and in general non-quadratic in $\bar{\mathbf{F}}$. It has to be maximized/minimized with respect to numerous inequality constraints which are quadratic in $\bar{\mathbf{F}}$. This constitutes a rather difficult optimization problem. However, it turns out that there is exactly one fully quadratic constraint for every $\bar{\boldsymbol{f}}^i$, all others are bilinear in $(\bar{\boldsymbol{f}}^i, \bar{\boldsymbol{f}}^j)$ for $i \neq j$. Thus, if all other $\bar{\boldsymbol{f}}^j$, $j \neq i$, are kept fixed while optimization is done with respect to $\bar{\boldsymbol{f}}^i$, then this results in non-linear optimization under one quadratic and a number of linear inequality constraints, which is a much simpler problem. It's Lagrangian is given by

$$L^i = \left\langle \Theta(\bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}}^i) \right\rangle + \sum_{j \neq i, \tau} \lambda^i_{j,\tau} \left(\pm \bar{\boldsymbol{f}}^{iT} \mathbf{C}_\tau \bar{\boldsymbol{f}}^j - \epsilon \right) + \lambda^i_{i,0} \left(\bar{\boldsymbol{f}}^{iT} \mathbf{C}_0 \bar{\boldsymbol{f}}^i - 1 \right) \tag{3.62}$$

with Lagrange multipliers $\lambda^i_{j,\tau}, \lambda^i_{i,0} \leqslant 0$. The solution is found at a stationary point, where the gradient with respect to $\bar{\boldsymbol{f}}^i$ vanishes,

$$0 = \frac{\partial L^i}{\partial \bar{\boldsymbol{f}}^i} = \left\langle \theta(\bar{\boldsymbol{x}}_t^T \bar{\boldsymbol{f}}^i) \bar{\boldsymbol{x}}_t \right\rangle + \sum_{j \neq i, \tau} \pm \lambda^i_{j,\tau} \mathbf{C}_\tau \bar{\boldsymbol{f}}^j + \lambda^i_{i,0} \mathbf{C}_0 \bar{\boldsymbol{f}}^i \,.$$

[3] The noise is considered as an independent separate signal.

Assume that there is a solution which provides unit output variance. Then $\lambda^i_{i,0}$ is true negative, and we can rearrange the above equation to

$$\bar{f}^i_0 \quad \propto \quad \mathbf{C}_0^{-1} \left\langle \theta(\bar{x}_t^T \bar{f}^i) \bar{x}_t \right\rangle + \sum_{j \neq i,\tau} \pm \lambda^i_{j,\tau} \mathbf{C}_0^{-1} \mathbf{C}_\tau \bar{f}^j \,. \tag{3.63}$$

Comparing this equation with equation (3.61), one can see that additional terms arise for $\lambda^i_{j,\tau} \neq 0$, i.e. for those shifts τ and signal pairs (i, j) at which the cross-correlation function attaches the bound $\pm\epsilon$. One can interpret this as follows: Whenever $\bar{f}^i$ gets similar to $\bar{f}^j$ for a certain shift τ, the output cross-correlation function attaches ϵ for that shift[4], the Lagrange multiplier $\lambda^i_{j,\tau}$ assumes a negative value, and a fraction of $\bar{f}^j$ is substracted from the unconstraint estimate of $\bar{f}^i$. $\mathbf{C}_0^{-1}\mathbf{C}_\tau$ is nothing else than a shift operator that provides the corresponding shift of $\bar{f}^j$ by τ samples.

However, one must note that equation (3.63) cannot serve as an replacement for (3.61) because the correct values of the Lagrange multipliers neither are known beforehand, nor they would remain constant during the iterations. To actually solve the constraint optimization problem other means must be found.

3.4.4 MCBDC optimization

As mentioned above the task of finding a solution of (3.58) can be considerably simplified when it is split-up into subsequent optimization episodes of a single $\bar{f}^i$ while all other $\bar{f}^j$ remain fixed. During the optimization w.r.t. $\bar{f}^i$, from the point of view of $\bar{f}^j$, the linear constraints that are functions of $\bar{f}^i$ change. However, if $\bar{f}^j$ fulfills all it's constraints before the optimization of $\bar{f}^i$ started, then afterwards it will do so as well, provided $\bar{f}^i$ does not violate any constraints that are functions of $\bar{f}^j$. Thus, we require that after every optimization episode all constraints are fulfilled.

For fixed $\bar{f}^j$, $j \in \{1 \ldots M\} \backslash \{i\}$, the set of admissible points for $\bar{f}^i$ is a convex, closed subset of $\mathbb{R}^{N(2T_f+1)}$. It has the shape of a hyper-ellipsoid defined by $\mathbf{C}_0$ which is centered around zero and which is cut off at the planes defined by the linear constraints. Obviously zero is an admissible point.

In the following the optimization algorithm used for MCBDC is described. It performs general function minimizations under one positive definite quadratic constraint and several linear inequality constraints (objective functions that have to be maximized must be inverted accordingly). Let's ignore the quadratic constraint for the moment, and assume that the linear constraints alone constitute a closed set of admissible points. Without loss of generality we want to minimize a cost function $E(\bar{f})$ under the given constraints. E may be smooth and differentiable. Starting from an admissible point $\bar{f}_A$ we can do a linear approximation, E_{lin}, of the cost function by it's gradient at that point, $\nabla\bar{f}_A$, which may be different from zero. By means of standard *Linear Programming* methods (Press et al., 1992), the global minimum of the linearized objective function under the linear constraints can be easily found. It may be attained at the admissible point $\bar{f}_B$. It now may happen that $E(\bar{f}_B) > E(\bar{f}_A)$ although $E_{lin}(\bar{f}_B) < E_{lin}(\bar{f}_A)$. In such a case we know that every point on the line between $\bar{f}_A$ and $\bar{f}_B$ is admissible, because the set of admissible points is convex. Thus we can search for the point

$$\bar{f}_C = (1-\alpha)\bar{f}_A + \alpha\bar{f}_B \,, \quad \alpha \in [0,1] \,, \tag{3.64}$$

on the line between $\bar{f}_A$ and $\bar{f}_B$ that has the smallest cost function value. It can be found with standard line-search methods, e.g. *Golden Section Search*. Eventually, at the new point $\bar{f}_B$ rsp. $\bar{f}_C$ the whole procedure is repeated.

It remains to extend the described procedure so that it is also subject to the quadratic constraint. This can be achieved by using the tangent of the quadratic constraint as a

[4] w.o.l.o.g. the correlations may be positive

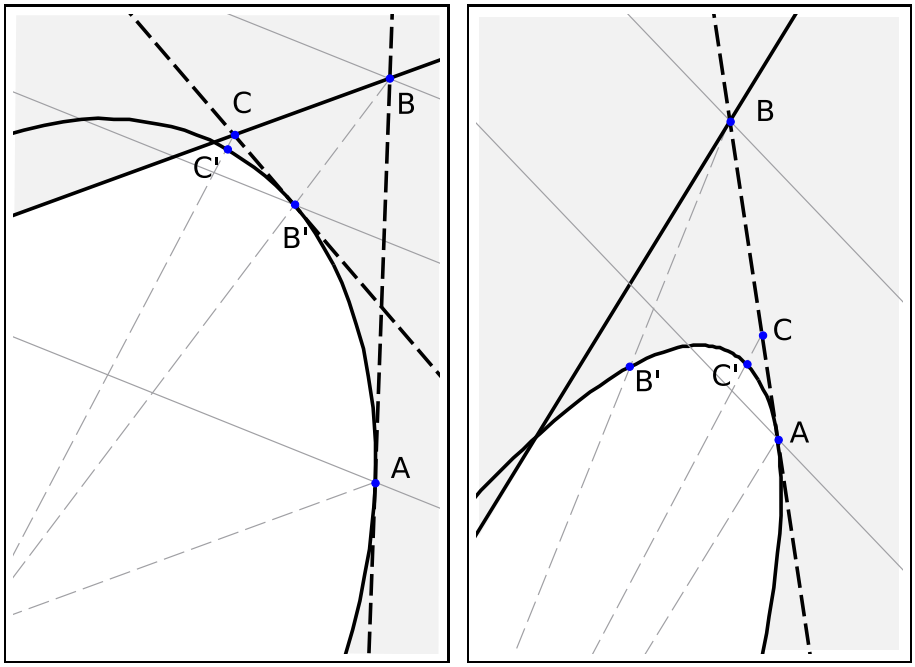


Figure 3.12: Illustration of the MCBDC optimization procedure. Bold black lines indicate the quadratic and linear constraints, respectively. In every iteration the quadratic constraint is approximated by it's tangent at the current point displayed with dashed black lines. The region of admissible solutions is displayed white. Solid gray lines are the iso-lines of the (linearized) objective function. Dashed gray lines show the connection of the points A, B, B′, C and C′ to zero. For detailed explanation see text.

'temporary' linear constraint. Consider as an example the left panel of figure 3.12. The region of admissible points is the white area. The linear and quadratic constraints are the solid black lines. Without loss of generality the current point may attach the boundary of the quadratic constraint at **A**. There the tangent constraint is established (dashed black line going through **A**). The linear programming procedure is started and yields **B** as the constrained minimum of E_{lin} (iso-lines of E_{lin} at **A** are indicated by thin, solid gray lines in the figure). It now may happen that **B** violates the quadratic constraint[5]. In this case **B** is projected in dircetion of zero onto the boundary of the quadratic constraint. If the cost function at this point, **B'**, is smaller than at **A** this iteration step is complete. The next turn starts at **B'** with a new tangent constraint, a new gradient, and a new E_{lin}, leading to **C** and to **C'** and so on. It may, however, happen that the cost function evaluated at **B'** yields a larger value than at **A**. This is illustrated on the right panel of figure 3.12. In this case we search that point **C** on the line between **A** and **B**, the projection **C'** of which yields the smallest cost function value. The listing Algorithm 3 summarizes the the optimization procedure in pseudocode.

It is necessary to make some statements that constitute the correctness of the method. We need to show that every iteration cycle (i) leads to an admissible point and (ii) does not increase the value of the cost function.

(i) $\bar{f_A}$ is an admissible point. With respect to only the linear and the tangent constraint, the result of the linear program, $\bar{f_B}$, is an admissible point. Because the set of admissible points is convex, any point $\bar{f_C}$ according to (3.64), and any point on the line between zero and $\bar{f_C}$ is admissible. In particular

$$\bar{f_{C'}} = \min(1; (\bar{f_C}^T \mathbf{C}_0 \bar{f_C})^{-\frac{1}{2}}) \bar{f_C}$$

is an admissible point also with respect to the quadratic constraint.

[5] If it does not, then one proceeds as described above, where there was no quadratic constraint.

Algorithm 3 MCBDC optimization.

function OPT_MCBDC($\bar{f}, E, \mathbf{C}_0, C, \epsilon$)
▷ Returns $\bar{f}$ with minimal value $E(\bar{f})$ under the constraints $\bar{f}^T \mathbf{C}_0 \bar{f} \leqslant 1$ and $|\bar{f}^T \mathbf{c}| \leqslant \epsilon$ for all $\mathbf{c} \in C$. C is usually computed as $C = \bigcup_{j \neq i,\, \tau \in [-T_l, T_l]} \{\mathbf{C}_\tau \bar{f}^j\}$.

$\bar{f}_A \leftarrow \bar{f}$
$C \leftarrow \{\frac{1}{\epsilon}\mathbf{c} : \mathbf{c} \in C\} \cup \{-\frac{1}{\epsilon}\mathbf{c} : \mathbf{c} \in C\}$
repeat
$\nabla \bar{f}_A \leftarrow \left.\frac{\partial E(\bar{f})}{\partial \bar{f}}\right|_{\bar{f}=\bar{f}_A}$
if $\|\nabla \bar{f}_A\| < \eta_1$ **then**
return $\bar{f}_A$ ▷ Stopping criterion 1
end if
$\mathbf{c}_A \leftarrow \mathbf{C}_0 \bar{f}_A$ ▷ The tangent constraint
$\bar{f}_B \leftarrow$ LINPROG($\nabla \bar{f}_A, C \cup \{\mathbf{c}_A\}$)
$\bar{f}_{B'} \leftarrow$ BACKPROJ($\bar{f}_B, \mathbf{C}_0$)
if $E(\bar{f}_{B'}) > E(\bar{f}_A)$ **then**
$\alpha \leftarrow \operatorname{argmin}_{\alpha \in [0,1]} E\left(\text{BACKPROJ}((1-\alpha)\bar{f}_A + \alpha \bar{f}_B), \mathbf{C}_0\right)$
▷ Performed by some standard line search method
$\bar{f}_{B'} \leftarrow$ BACKPROJ($(1-\alpha)\bar{f}_A + \alpha \bar{f}_B$)
end if
if $E(\bar{f}_A) - E(\bar{f}_{B'}) < \eta_2$ or $\|\bar{f}_A - \bar{f}_{B'}\| < \eta_3$ **then**
return $\bar{f}_{B'}$ ▷ Stopping criteria 2 and 3
end if
until max. number of iterations exceeded ▷ Stopping criterion 4
return $\bar{f}_{B'}$
end function

function BACKPROJ($\bar{f}, \mathbf{C}_0$)
▷ Projects $\bar{f}$ in direction of zero if necessary, to fulfill the quadratic constraint

$\bar{f} \leftarrow \min\left(1; \bar{f}^T \mathbf{C}_0 \bar{f})^{-\frac{1}{2}}\right) \bar{f}$
return $\bar{f}$
end function

function LINPROG($\mathbf{g}, C$)
▷ Standard linear programming method. Returns $\bar{f}$ that yields the minimal value of $\bar{f}^T \mathbf{g}$ under the constraint $\bar{f}^T \mathbf{c} \leqslant 1$ or all $\mathbf{c} \in C$.
end function

(ii) If $E(\bar{f}_{B'}) > E(\bar{f}_A)$ we do a line search minimization from $\bar{f}_A$, $\alpha = 0$, to $\bar{f}_{B'}$, $\alpha = 1$. We need to show that the directional derivative of E on the trajectory described by $\bar{f}_{C'}$ has a negative slope at $\bar{f}_{C'} = \bar{f}_A$ (at $\alpha = 0$). If $\bar{f}_A^T \mathbf{C}_0 \bar{f}_A < 1$ or $\bar{f}_B \mathbf{C}_0 \bar{f}_A < 1$, then there is an α' with $0 < \alpha' \leqslant 1$ so that for $0 < \alpha < \alpha'$ holds $\bar{f}_C^T \mathbf{C}_0 \bar{f}_C <= 1$. That means that in the neighborhood around $\alpha = 0$ no projection toward zero is necessary and

$$\bar{f}_{C'} = \bar{f}_C \,.$$

In this case the derivative

$$\left.\frac{dE(\bar{f}_{C'})}{d\alpha}\right|_{\alpha=0} = (\bar{f}_B - \bar{f}_A)^T \nabla \bar{f}_A = E_{lin}(\bar{f}_B) - E_{lin}(\bar{f}_A) < 0 \,.$$

is always below zero.

If $\bar{f}_A^T \mathbf{C}_0 \bar{f}_A = \bar{f}_B^T \mathbf{C}_0 \bar{f}_A = 1$, i.e. we start on the tangent and end on the tangent, it holds that $\bar{f}_C^T \mathbf{C}_0 \bar{f}_C > 1$ for $\alpha > 0$. Thus, back projection in direction of zero is necessary:

$$\bar{f}_{C'} = (\bar{f}_C^T \mathbf{C}_0 \bar{f}_C)^{-\frac{1}{2}} \bar{f}_C \,.$$

The derivative has the form

$$\left.\frac{dE(\bar{f}_{C'})}{d\alpha}\right|_{\alpha=0} = (\bar{f}_B - \bar{f}_A)^T \nabla \bar{f}_A - (\bar{f}_B^T \mathbf{C}_0 \bar{f}_A - 1)\,(\bar{f}_A^T \nabla \bar{f}_A) \,.$$

The second term, however, is always zero so that also here the slope of $E(\bar{f}_{C'})$ at $\alpha = 0$ is always negative.

Problems can arise if the linear constraints together with the tangent constraint do not constitute a closed set of admissible points, in which case the linear program can diverge. This can be helped with additional box constraints in suitable distance around $\bar{f}_A$. Such box constraints also have the advantage that they limit the distance between $\bar{f}_B$ and $\bar{f}_A$ so that the approximated cost function $E_{lin}(\bar{f}_B)$ becomes not too different from $E(\bar{f}_B)$. In the experiments an increase of the performance speed could be observed if suitably chosen box constraints where used.

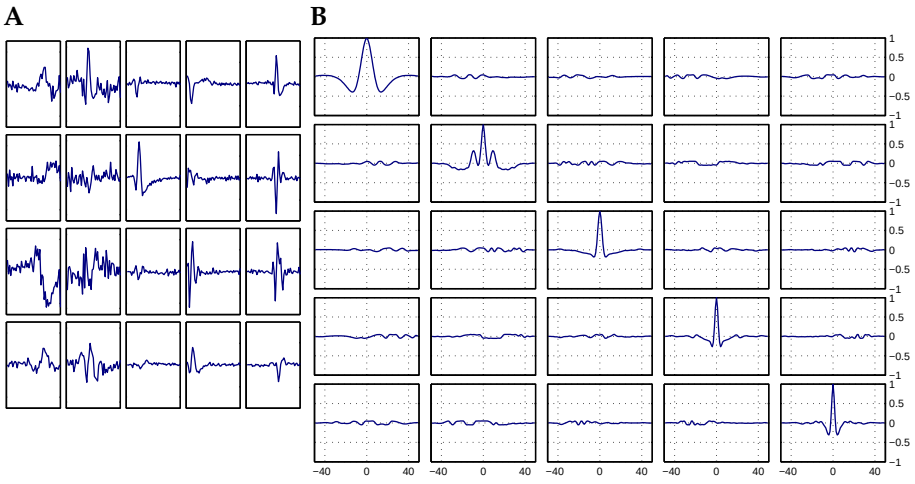


Figure 3.13: **A:** Optimal multi-channel filters, learned with MCBDC maximizing $\left\langle \text{sign}(\bar{x}_t^T \bar{f}^i)\, |\bar{x}_t^T \bar{f}^i|^4 \right\rangle$ as the objective function. **B:** Cross-correlation functions of the filtered signals (cf. figures 3.10 and 3.11). Constraints provided that the cross-correlation functions between two different signals do not exceed ± 0.05.

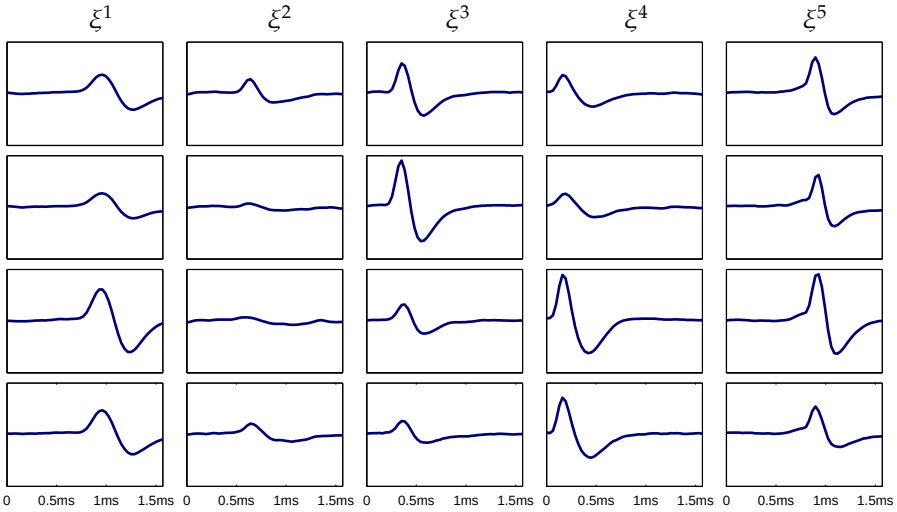


Figure 3.14: The corresponding templates estimated with $\left\langle 4|\bar{x}_t^T \bar{f}^i|^3 \bar{x}_t \right\rangle$. Compare the templates with the manually detected ones in figure 3.6.

Figures 3.13 and 3.14 show the results of the proposed MCBDC algorithm applied to the same tetrode dataset used in the previous experiments. There were five multi-channel filters to be estimated from the data. The objective was to maximize the fourth order 'skew' moments,

$$\left\langle \Theta(\bar{x}^T \bar{f}^i) \right\rangle = \left\langle \mathrm{sign}(\bar{x}_t \bar{f}^i)(\bar{x}_t \bar{f}^i)^4 \right\rangle \ , \tag{3.65}$$

under the constraints that limit the output variance to a value of at most one and the cross-correlation functions to absolute values below or equal to $\epsilon = 0.05$. The skew variant of the fourth order moment was chosen because in the tetrode data the templates occur with fixed polarity (ideally even with fixed amplitude). Thus, the distribution of v^i (cf. equation (3.50)) is a sparse and asymmetric one, and the choice of Θ accounts for that. The setup was otherwise the same as described in the sections 3.4.1 and 3.4.2. The optimization was performed sequentially for one $\bar{f}^i$ after another in altogether 100 cycles of $M = 5$ episodes. An episode contained the execution of the Opt_MCBDC function (cf. Algorithm 3) for the corresponding $\bar{f}^i$. In order to achieve more or less uniform updates of all $\bar{f}^i$, inside Opt_MCBDC only one iteration per function evaluation was performed. The convergence speed could be increased to some degree, when the linear program was performed with box constraints in all dimensions at the distance of ± 0.05 from $\bar{f}^i$. The whole implementation was done in MATLAB and the linear programming and line search routines where taken from MATLAB's Optimization Toolbox.

Figure 3.13.A shows the resulting filters. One can see that they neither they are shifted copies of themselves as in figure 3.10, nor they are just frequency splitting as in figure 3.11. The corresponding cross-correlation functions of the filtered signals are shown in figure 3.13.B for shifts $\tau \in [-50, 50]$. All of them are in the prescribed range $[-0.05, 0.05]$. The auto-correlation functions (panels on the diagonal) have a peak value of one meaning unit variance of the filtered signals as required. From the filtered signals templates can be estimated according to equation (3.51) with the empirical expectation

$$\bar{\xi}^i = \left\langle 4|\bar{x}_t^T \bar{f}^i|^3 \bar{x}_t \right\rangle \ .$$

These are shown in figure 3.14. The fact that they are very similar to those that were manually detected and averaged, figure 3.6, shows that the MCBDC algorithm performs correctly.

However, one must note that optimal linear multi-channel filters as such – whether they are computed from previously known templates or unsupervisedly estimated using

MCBDC – are usually not sufficient to perform high accuracy spike sorting tasks. In real applications problems usually arise from the fact that the data model (3.50) is a too strong idealization. In real data the wave form templates are known to be not constant over time, but changing with respect to the inter-spike interval time. Further, a drifting of the electrode and, hence, a change of the multi-channel template shape can never be totally excluded. Thus, a linear spike-sorting algorithm must be able to cope with short-time template changes due to the last ISI and must be able to adapt to slow changes due to electrode drifts. Also the limited capacity of linear filters can give rise to problems, as discussed in section 3.2.2. In the presented experiment (figures 3.13 and 3.14) , for example, it is quite hard to detect template $\boldsymbol{\xi}^1$ with a linear filter under the presence of template $\boldsymbol{\xi}^5$, which is rather similar but occurs with larger amplitude in the data. The filters must be sharply tuned to discriminate the templates (and provide small cross-correlation functions), but cannot be arbitrary sharply tuned because of the presence of background noise. Looking closer to the auto-correlation functions (panels (1,1) and (5,5) in figure 3.13.B) one can see that MCBDC uses a 'trick' to provide small cross-correlations between the two similar templates $\boldsymbol{\xi}^1$ and $\boldsymbol{\xi}^5$: It is that, roughly speaking, $\vec{\boldsymbol{f}}^1$ focuses on low frequencies and $\vec{\boldsymbol{f}}^5$ on higher frequencies, which can be recognized from the smoothness of the auto-correlation functions. In this experiment it was just possible to detect $\boldsymbol{\xi}^1$ only with low frequencies, but it reflects in small values of the objective function (3.65). With linear multi-channel filters alone, more than five templates would be hardly detectable in these data.

On the other hand, linear filters offer the advantage that, once they have been learned, they can be applied on current hardware in principle in real time. A real time experiment, however, could allow to modify the electrode placement so that the diversity of the templates is maximized and optimized for the detection and discrimination with linear filters. Thus, further research on this area is quite meaningful.

3.4.5 MCBDC for video data

The MCBDC algorithm is not limited to signals that are time sequences. It is in principle applicable to any stochastic signals $(\mathbf{x}_t)_{t\in \mathrm{I}}$ with temporal indices in $\mathrm{I} = \mathbb{Z}^{N_t}$. In this section an experiment with MCBDC applied to video data will be presented. The input dataset is a video consisting of K frames, each of which is $T_1 \times T_2$ pixels in size. Every pixel is a gray value, hence, the whole dataset is a $T_1 \times T_2 \times K$ array of real numbers (cf. section 1.4.1). However, in order to process motion features in the video, we consider it as a series of $K - N + 1$ multi-channel frames by processing every N adjacent frames together as one N-channel frame. Thus, the setup can be summarized as the stochastic process (cf. section 1.1)

$$(\mathrm{E}^{\mathrm{I}}, \mathscr{B}(\mathrm{E}^{\mathrm{I}}), P_{\mathrm{I}}, (\mathbf{x}_t)_{t\in \mathrm{I}}) \quad \text{with} \quad \mathbf{x}_t \in \mathrm{E} = \mathbb{R}^N \;\; \mathrm{I} \subseteq \mathbb{Z}^2 \,.$$

The given K frames in the video stack can be interpreted as $K - N + 1$ realizations of the stochastic processes. One usually finds that for this process the assumption of ergodicity is not well founded. Thus, expectations and empirical moments are computed over all available realizations and all t, or over a uniformly chosen subset therefrom (cf. section 1.4.2 for a discussion on ergodicity of finite datasets).

Leaving this aside, the extension of MCBDC to higher dimensional index sets is straightforward. If we want to estimate filters $(\boldsymbol{f}_t)_{t\in \mathrm{J}}$ and templates $(\boldsymbol{\xi}_t)_{t\in \mathrm{J}}$ with receptive fields J (w.o.l.o.g. the filters and templates may have the same receptive field), we need to define the $\mathbb{R}^{N|\mathrm{J}|}$ vectors

$$\begin{aligned}
\vec{\boldsymbol{f}}^i &= (f^i_{1,t_0}, f^i_{1,t_1}, \ldots, f^i_{N,t_0}, f^i_{N,t_1}, \ldots)^T \,, \quad t_j \in \mathrm{J} \\
\vec{\boldsymbol{\xi}}^i &= (\xi^i_{1,t_0}, \xi^i_{1,t_1}, \ldots, \xi^i_{N,t_0}, \xi^i_{N,t_1}, \ldots)^T \,, \quad t_j \in \mathrm{J} \\
\bar{\boldsymbol{x}}_t &= (x_{1,t+t_0}, x_{1,t+t_1}, \ldots, f_{N,t+t_0}, f_{N,t+t_1}, \ldots)^T \,. \quad t_j \in \mathrm{J}\,, t + t_j \in \mathrm{I}
\end{aligned}$$

To compute the constraints, we further need the matrices $\mathbf{C}_\tau$ computed as empirical expectations of equation (3.52) over $\tau \in \mathrm{J}$. With that the MCBDC algorithm can be applied just as described in section 3.4.4 .

The data for this experiment was a b/w video sequence of $K = 15,000$ frames, recorded at 16 fps, with a resolution of 320 × 240 pixels. The original video can be found on the accompanying CD under the name `floor_+2.avi` (cf. also sections 5.3 and 5.4.2, in which this video is also used; the recording setup is described there). For this experiment the left image was discarded, and every frame was 2-fold down sampled to 160 × 120 pixels size. Every $N = 4$ adjacent frames were grouped together to form a multi-channel frame. Altogether there were $K - N + 1 = 14,997$ multi-channel frames then. Therefrom $4 \cdot 10^6$ samples $\vec{x}$ were taken uniformly distributed over the whole video. The receptive field was 10×10 pixels and cross-correlations for lags up to $T_l = 9$ pixels horizontally and vertically should be considered in $\mathbf{C}_\tau$, thus

$$\mathrm{J} := [0,9] \times [0,9]\,, \quad \mathrm{J}_l := [-9,9] \times [-9,9]\,.$$

The learning procedure was performed sequentially by solving the Lagrangian (3.62) for one $\vec{f}^i$ after another. That means that for $\vec{f}^1$ only the quadratic constraint was applied, while for all $j > i$ the optimization of $\vec{f}^j$ was also subject to the linear ones. The output cross-correlation functions were limited not to exceed values of $\pm\epsilon = 0.025$. The objective was to maximize the kurtosis of the outputs subject to the constraints, thus Θ was the fourth power.

Figure 3.15.A shows the resulting first 16 4-channel filters. The panel in the k-th row and i-th column displays the k-th channel of the i-th filter, $(f^i_{k,t})_{t\in\mathrm{J}}$. The kurtosis values of the filter outputs were

$$\begin{aligned}\left(\left\langle\Theta(\vec{x}^T\vec{f}^i)\right\rangle\right)_{i=1\ldots16} &= (\ 178.3,\ 95.5,\ 79.9,\ 52.3,\ 64.0,\ 153.0,\ 155.7,\ 149.3,\ \ldots\\ &\qquad 92.6,\ 216.0,\ 158.9,\ 41.5,\ 59.0,\ 84.9,\ 203.5,\ 184.4)\,,\end{aligned}$$

which shows that quite sparse outputs were achieved. The corresponding templates obtained from the relation (cf. equations (3.59) and (3.60))

$$\vec{\xi}^i = \mathbf{C}_0\vec{f}^i\,. \tag{3.66}$$

are displayed in Figure 3.15.B . Figure 3.15.C shows that at the same time the constraints were not violated. The (i,j)-th panel is the average output cross-correlation function $(l^{ij}_\tau)_{\tau\in\mathrm{J}_l} = \left((\vec{f}^i)^T\mathbf{C}_\tau\vec{f}^j\right)_{\tau\in\mathrm{J}_l}$. The quadratic constraints lead to peak values in the main diagonal equal to one, the linear constraints provide values in the off diagonal panels close to zero ($\lesseqgtr \pm\,\epsilon$). Interestingly, the outputs do not need to have disjunctive spectra in order to have small cross-correlation functions, as one can see from the shape of the average auto-correlation functions. For example, the filters $\vec{f}^8$, $\vec{f}^{11}$ and $\vec{f}^{12}$ lead to output signals with nearly identical power spectra (cf. 8th, 11th and 12th panel on the main diagonal). Their cross-correlation functions, however, are close to zero. This is possible because the linear filters perform multi-channel operations. To understand this somewhat surprising result consider the spectrum $(Y^i_\omega)_{\omega\in\mathrm{I}_\omega}$ of the i-th filter output. For some $i \neq j$ the slightly simplified condition[6]

$$|Y^i_\omega|^2 = |Y^j_\omega|^2 \gtrless 0\,, \quad Y^i_\omega\,\mathrm{conj}(Y^j_\omega) = 0\,, \tag{3.67}$$

may hold, where the power spectra must be non-zero for at least one $\omega \in \mathrm{I}_\omega$. The cross-power spectra of the outputs can be written in terms of the input cross-power spectra and the spectra of the filters,

$$Y^i_\omega\,\mathrm{conj}(Y^j_\omega) = \sum_{k=1}^{N}\sum_{l=1}^{N}\left\langle\mathrm{conj}(X_{k,\omega})X_{l,\omega}\right\rangle F^j_{k,\omega}\mathrm{conj}(F^i_{l,\omega})\,,$$

where $(X_{k,\omega})_{\omega\in\mathrm{I}_\omega}$ is the spectrum of the k-th channel of $(\mathbf{x}_t)_{t\in\mathrm{I}}$, and $(F^i_{k,\omega})_{\omega\in\mathrm{I}_\omega}$ is the spectrum of the k-th channel of the i-th filter. In matrix notation (3.67) is

$$\begin{aligned}c_\omega\,\delta_{ij} &= Y^i_\omega\,\mathrm{conj}(Y^j_\omega)\\ &= (F^j_{1,\omega},\ldots,F^j_{N,\omega})\,\left\langle\mathrm{conj}(X_{1,\omega},\ldots,X_{N,\omega})^T(X_{1,\omega},\ldots,X_{N,\omega})\right\rangle\,\mathrm{conj}(F^i_{1,\omega},\ldots,F^i_{N,\omega})^T\,,\end{aligned}$$

[6] In MCBDC only approximate disjunctive cross power spectra are required.

with $c_\omega \neq 0$ for at least one $\omega \in \mathrm{I}_\omega$. Obviously, this cannot be the fulfilled in a single-channel scenario with $N = 1$. More generally speaking, at most N filters can fulfill (3.67) together because the matrix $\left\langle \mathrm{conj}(X_{1,\omega}, \ldots, X_{N,\omega})^T (X_{1,\omega}, \ldots, X_{N,\omega}) \right\rangle$ has a rank of at most N.

For the above mentioned filters on can find that all three detect edges in one and the same orientation, but with different directions of motion. Figure 3.17 gives an illustration of the outputs these filters for three similar multi-channel frames. One can see that all respond to edges of the same orientation. What not can be seen in the figure is that in **A** the scene was moving to the right, in **B:** to the left, and in **C:** slightly to the right. This type of motion can be distinguished only with multi-channel filters. The whole filtered video can be found on the CD (file `filt_8_11_12.avi`).

For comparison, in figure 3.16 the filters, templates, and cross-correlation functions are shown that arise when iterations (3.61) are carried out instead of MCBDC. One can see that also these filters detect directional and sparse features. The kurtosis values of the filter outputs were

$$\left(\left\langle \Theta(\vec{x}^T \vec{f}^{\,i}) \right\rangle \right)_{i=1\ldots16} = (80.9,\ 64.3,\ 63.6,\ 97.4,\ 255.4,\ 158.4,\ 73.3,\ 62.6,\ \ldots \\ 97.2,\ 96.6,\ 65.1,\ 202.3,\ 79.7,\ 65.2,\ 95.5,\ 67.5)\ .$$

However, trivially shifted versions of the filters and the templates occur leading to large cross-correlations. Only for $\tau = 0$ (center of the panels in 3.16.C) zero cross-correlations are guaranteed.

3.4.6 Discussion

Two experiments from different domains were presented to illustrate the capabilities of the MCBDC optimization procedure. The derivation was done on the basis of the generative data model (3.50). In this respect MCBDC can be seen as a convolutive BSS algorithm because the data model is nothing else than a convolutive mixture of independent and white sources corrupted with white, Gaussian noise. This model may have been more appropriate to the tetrode data in section 3.4.1 and 3.4.2 than to the video data in section 3.4.5. In the spike sorting experiment there is a rather intuitive interpretation of the intrinsic signals (and the filtered signals) as sources, which are the actual causes of the observations. For the video data this is less intuitive. Assuming that edges in various orientations are the ingredients of natural images, one would not expect that they are spread iid. over the image. Strong local dependencies that occur, for example, in very elongated edges are ignored by the model (3.50). However, MCBDC is meaningful for such data anyway if it is seen as a projection pursuit method. Locations in images at which a sparse projection assumes large values may have an interpretation as 'interesting points in the scene' rather than a linear source. Here the decorrelation constraint provides a means to extract sufficiently different projections.

In this respect non-linear projections seem to be particularly promising. A non-linear version of MCBDC is in principle possible. Therefore it is important that the constraints can be efficiently evaluated. This can be achieved without too much overhead by a non-linear expansion of all data samples $\vec{x}$ into some (certainly high-dimensional) feature space. In that feature space the linear MCBDC algorithm is executed. Here, as well as in other methods that base non-linear expansions of linear methods, the problem arises what a good non-linear expansion would be and how it should be parameterized. This is crucial because the expansion is chosen ad hoc and not changed during optimization. See also sections 5.1.2 and 5.1.3 .

With increasing complexity, e.g. with increasingly complex non-linearities, the multi-channel filters can be assumed to yield increasingly sparse outputs. This, leads to the probably most severe problem with MCBDC in large scale applications: any empirical estimates are subject to increasing variance, when the sparseness of the underlying distribution increases. Hence, the closer one gets to the optimal, sparseness maximizing projection, the more difficult it becomes to actually estimate it without overfitting. Already the linear

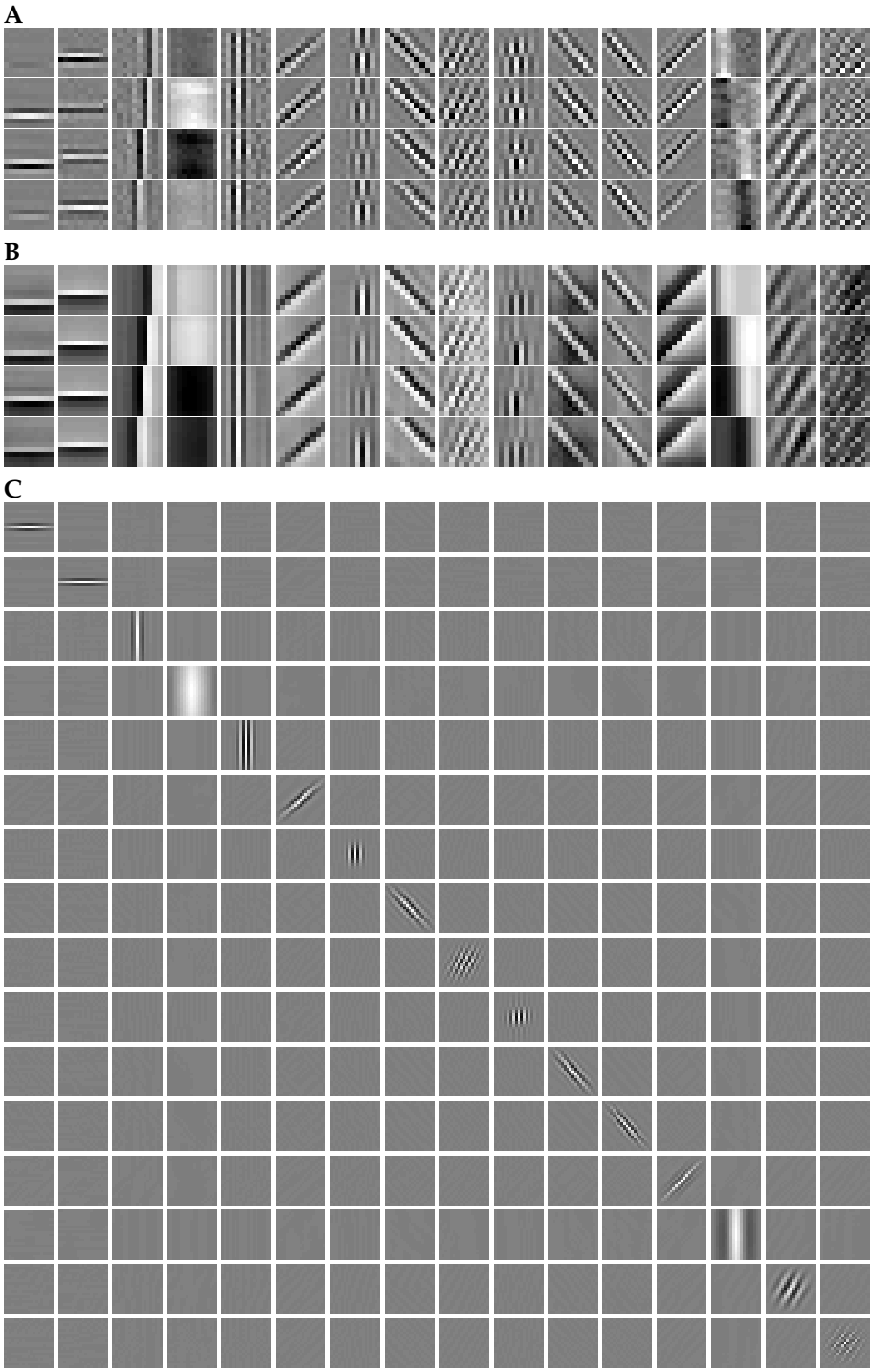


Figure 3.15: **A:** Multi-channel filters achieved through MCBDC on video data. The panels in j-th column show the four channels of the j-th filter. **B:** The corresponding multi-channel templates according to equation (3.66). Gray values are mapped to $\pm \max |\vec{f}^j|$ rsp. $\pm \max |\vec{\xi}^j|$ individually for every column. **C:** Cross-correlation functions of the filter outputs. The i, j-th panel shows the values of $(l_\tau^{ij})_{\tau \in \mathsf{J}_l}$ as gray values (black: -1, white: 1).

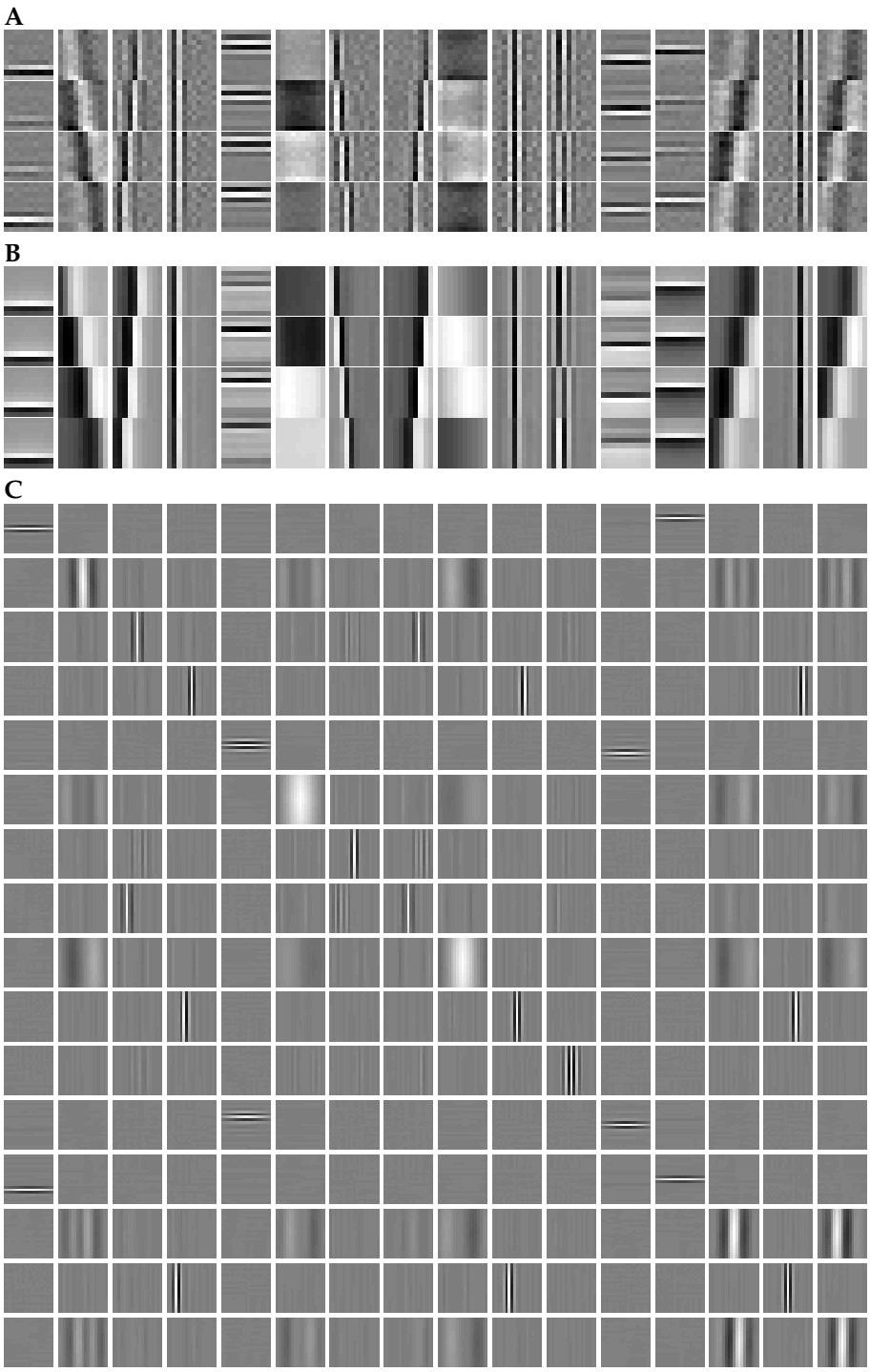


Figure 3.16: The equivalent to figure 3.15 when the filters are derived through the iterations (3.61). These filters provide uncorrelated responses for $\tau = 0$ (centers of the panels in **C**), but trivially shifted filters frequently occur.

Figure 3.17: Three 4-channel frames of the video filtered with $\vec{f}^{8}$ (red), $\vec{f}^{11}$ (green), and $\vec{f}^{12}$ (blue). Points for which $\pm y_t^i > 10$ are displayed in the corresponding color. The displayed image is the the fourth channel, $(x_{4,t})_{t\in\mathrm{I}}$, of the input. The filters become active when edges of certain orientation move **A:** to the right, **B:** to the left, and **C:** slightly to the right.

projections in the video data experiment required as much as $4 \cdot 10^6$ training examples. A non-linear variant of the algorithm would further increase the necessary training dataset by (i) the increase of the dimensionality of the data samples in feature space and (ii) by the amount of additional data samples that are necessary for reliable estimates under increasing sparseness. Thus, MCBDC may be better suited to small scale problems. For large scale problems the temporal coherence principle (Stone, 1996) can avoid such problems – chapter 5 is devoted to non-linear projection pursuit methods based on this principle.

3.5 Application: Spike-sorting

In extracellular recordings of neural activity, potential differences in the order of 1-100μV caused by changing extracellular currents are measured in the tissue. Whereas extracellular recordings were at first done with single electrodes, nowadays the technique of multiple simultaneous recordings from the same local area by means of *tetrodes* or *multitrodes* McNaughton et al. (1983); Reece and O'Keefe (1989) becomes more and more common. Because of the very high density of neurons, even in the direct neighborhood of the very fine electrode tips, there are still few cells, who's extracellular currents superimpose at the recording site. In order to analyse the activity of the individual neurons, the recorded signal has to be decomposed into its individual components by means of an appropriate spike sorting method, and therefore the additional information provided by multiple recording channels has been proven to be useful Gray et al. (1995).

Spike sorting is usually done three steps: event detection, feature selection, and clustering Lewicki (1998); Sahani (1999). All three steps have been subject to intensive research and development in the past, in order to improve the the performance and to benefit from multi-channel recordings.

The event detection is usually done by a simple thresholding operation. Surprisingly, it is not straight forward to extend this to multiple channels. Most simply, the thresholding can be performed on all channels individually, which, however, leads to an unintuitive rectangular thresholding surface in the space of amplitudes. A more sophisticated approach is to apply the threshold to the Mahalanobis distance of the amplitude vector, w.r.t. the covariance of the recorded signals Rebrik et al. (1999), leading to a hyperellipsoidal thresholding surface. In this approach, however, the distinction between positive and negative peaks is impossible, i.e. some additional heuristics is needed, to detect only one peak of the usually biphasic wave forms. The polarity of the peaks can be considered, however, by a combination of hyperellipsoidal and linear thresholding surfaces, as described in Sahani (1999).

From the neighborhood of the detected spikes in the recorded signals, a feature vector is computed, which must provide sufficient information to distinguish spikes from different cells. What features are used can be defined beforehand (e.g. peak-to-peak amplitudes, duration of the positive and negative phases, etc.) or can be learned from the data. For the

later case linear dimension reduction by means of principal component analysis or linear discriminant analysis are widely used techniques. The selection of the features is crucial for the overall performance of the spike sorting because the neighborhood of a detected spike does not contain the original wave form of the action potential, but its superposition with noise, background activity, and other action potentials. Thus, features must be robust against these types of distortions. The recording with multiple channels provides a very simple and powerful feature: the typical amplitude pattern of an action potential in the recording channels.

By means of a suitable clustering algorithm, the detected spikes, which are described by their feature vectors, are assigned to the individual source neurons. Beside manual cluster cutting in principle any automatic clustering algorithm can be applied. Gaussian mixture models or modifications thereof are widely used, even though widely tailed distributions have been reported to more accurately capture the multivariate statistics of the clusters Shoham et al. (2003). The question of how many sources are expected and, hence, how many clusters to choose was addressed in a number of publications Fee et al. (1996); Nguyen et al. (2003). It seems to be advantageous to set the number of clusters larger than the number of expected foreground neurons, on the one hand to be able to model more complex cluster distributions and, on the other hand, to pickup mistakenly detected spikes in separate clusters serving as 'garbage collectors'. This, of course, requires some strategy to put together all the clusters of the same neuron, and split off spare clusters Fee et al. (1996). The spare clusters may contain overlapping events, and procedures to decompose them into single events have been proposed Takahashi et al. (2003). However, the problem that overlapping events may not cross the threshold and remain undetected can not be solved by this means.

Alternatively, in a Bayesian frame work a posterior probability for the number clusters together with the posterior assignment probabilities can be given Nguyen et al. (2003), and the inter-spike interval distributions and firing statistics can be incorporated into the clustering model Pouzat et al. (2004). Theoretically, the entire recorded signals including the times of occurrence of the individual spikes could be modeled probabilistically similar to the Bayesian deconvolution approach in Andrieu et al. (2001). However, in all Bayesian approaches, the necessary posterior sampling by means of Markov chain Monte Carlo methods is known to be computationally rather expensive and, hence, may be suitable only for short signal length in an off-line environment.

The event detection alone can be a difficult task, in particular in situations where the action potentials are distorted by noise and large background activity and where overlap between spikes occurs. In such situation optimal linear filters that respond to the biphasic wave form of an action potential with a narrow impulse can help to improve the event detection accuracy. Because of the linearity, partially overlapping wave forms are transformed into partially or even non-overlapping narrow impulses, the peak values of which are less distorted. Thus, in the filtered recordings, the amplitude patterns of the recorded units are less sensitive against overlaps. Hence, the amplitudes of the impulses are more reliable estimates for the amplitude of the action potentials than the peaks of the original biphasic wave forms. With the help of real and simulated tetrode data, it will be demonstrated that by means of optimal filtering the amplitudes of the action potentials can be estimated with sufficiently high accuracy so that the four dimensional amplitude vectors can be used as sole features for the clustering step.

3.5.1 Tetrode data

There were two datasets of four channel extra-cellular tetrode recordings. One was recorded in 2002 from the visual cortex of an anesthetized and paralyzed cat during spontaneous activity. This one was used for the spike-sorting experiments presented in this section. The other one was recorded 1995 from the prefrontal cortex of an awake monkey during a visual task experiment Pipa et al. (2003). These data were mainly used to perform the experiments with optimal multi-channel filters in sections 3.3 and 3.4. Note, however, that there is nothing particular that binds the successful application of any of the described

method to either one of the datasets.

All electrophysiological recordings and preparations were performed at the *MPI for Brain Res., Frankfurt/M, Germany* according to the German Law for the Protection of Experimental Animals. The procedures also conform to the regulations issued by the NIH and the Society for Neuroscience.

The electrodes were prepared by pulling the multicore fiber and grinding it under a binocular microscope until the tip has a nice pencil-like shape. The impedance of the contacts is optimally between 0.3 and 0.8MΩ (measured at 1kHz) and was adjusted by electrolytic plating Pezaris et al. (1995). After mounting into the microdrive, the electrodes were driven through the intact meninges (dura mater) into the neuropil with micrometer precision. The first somatic action potentials are typically isolated after a 200–300 micrometer penetration depth, i.e. in the upper layers of the cortex and are most likely derived from pyramidal cells. The electrical signals were fed through a preamplifier and subsequently through filters (0.5-3kHz, 3db/octave) and digitized at 31.25kHz (1401plus, Spike2 Version4, CED Cambridge, UK). After mechanical pressure due to the initial penetration of the dura has ceased, fine positioning of the electrode is continued until several different action potentials can be seen in the simultaneously monitored signals on a multi-channel oscilloscope. Figures 3.18 and 3.19 show 5s of both datasets together with a 100ms closeup. The data have been normalized for zero mean and unit variance individually in every channel.

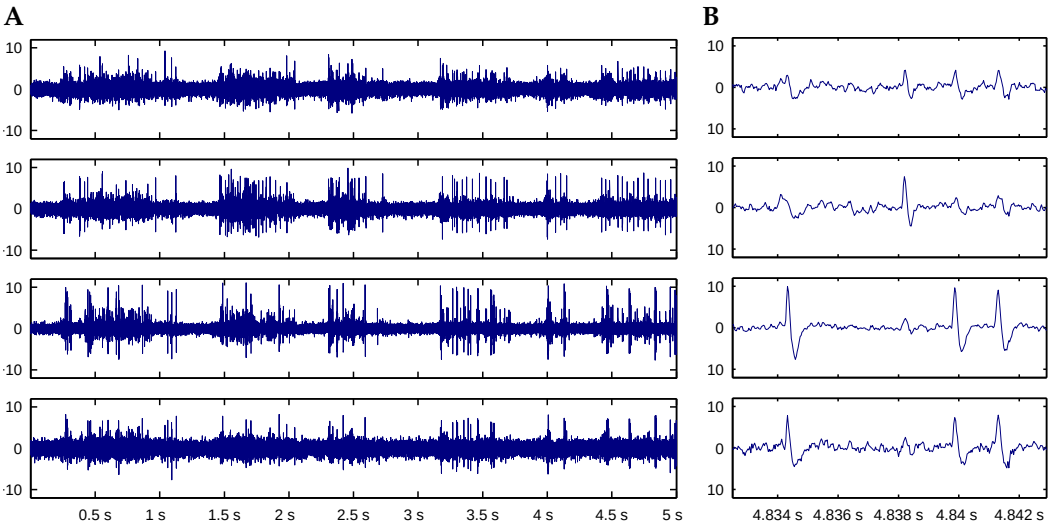


Figure 3.18: **A:** 5s of extra-cellular tetrode recordings from the prefrontal cortex of an awake monkey during a visual task experiment. **B:** 100ms closeup.

3.5.2 Spike-sorting based on optimal filtering

The following steps constitute a spike-sorting algorithm based on optimal filtering as preprocessing:

Step 1: High pass filtering and correction for the mean: Raw recordings of extra-cellular activities are often contaminated by low frequency fluctuations, that can be several orders of magnitude higher than the recorded action potentials. If there is no high pass filtering performed already in the recording system, it has to be done prior to all other processing steps. Cutoff frequencies for filtering are typically in the range of 400–800 Hz. A side effect of the high pass filtering is a zero mean in the data, provided the zero frequency component is sufficiently suppressed. If not, the mean has to be corrected explicitly. Without high pass filtering the optimal filter obtained from maximizing skewness has a high pass characteristic on its own. The data model, however, is no longer true. In particular (3.9) would no longer hold, and the connection between f and ξ would not be valid.

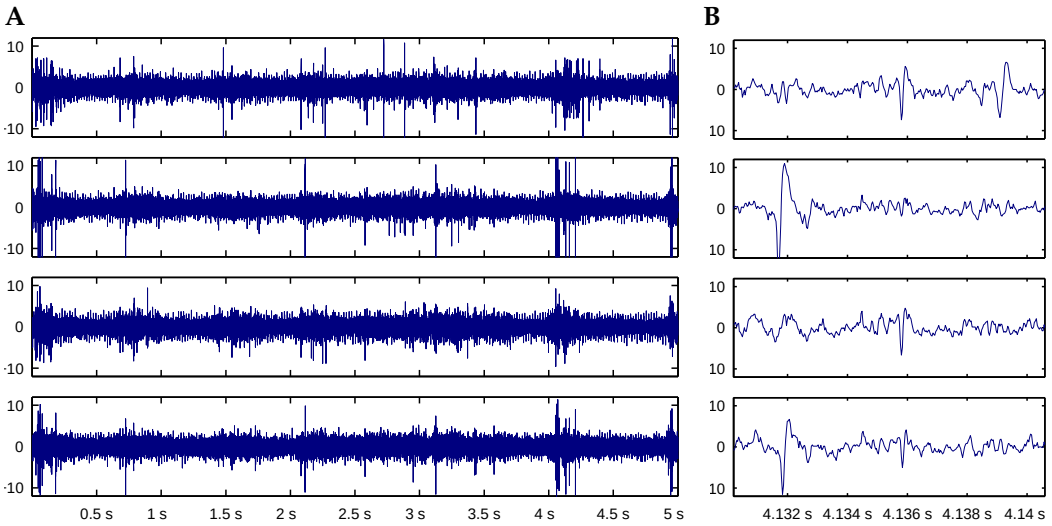


Figure 3.19: **A:** 5s of extra-cellular tetrode recordings from the visual cortex of an anesthetized and paralyzed cat during spontaneous activity. **B:** 100ms closeup.

Step 2: Hessian matrix H: The Hessian is estimated by computing the empirical mixed second moments of small signal clips $\bar{\boldsymbol{x}}_i(t)$ either using all possible time steps t and channels or a sufficiently large number of randomly taken clips. The estimated Hessian is given by

$$\hat{\mathbf{H}} = \frac{1}{NT_x} \sum_{i=1}^{N} \sum_{t=T_f}^{T_x+T_f-1} \bar{\boldsymbol{f}}_i(t) \bar{\boldsymbol{f}}_i(t)^T \,. \tag{3.68}$$

Step 3: Estimation of the optimal filter: The optimal filter is obtained from the fixed point iterations equations (3.20) and (3.21). The expectation is replaced by the empirical mean over all time steps and all channels. The average over a sufficiently large number of uniformly randomly drawn clips $\bar{\boldsymbol{f}}_i$ can be taken instead. One obtains

$$\boldsymbol{f} \leftarrow \mathbf{H}^{-1} \sum_{i=1}^{N} \sum_{t=T_f}^{T_x+T_f-1} (\boldsymbol{f}^T \bar{\boldsymbol{f}}_i(t))^2 \bar{\boldsymbol{f}}_i(t) \tag{3.69}$$

$$\boldsymbol{f} \leftarrow \frac{\boldsymbol{f}}{\sqrt{\boldsymbol{f}^T \mathbf{H} \boldsymbol{f}}} \,. \tag{3.70}$$

Step 4: Filtering: The cross correlation of the raw recordings x_i with the optimal filter f is calculated using the *Fast Fourier Transform (FFT)*. The spectrum of x_i is multiplied with the conjugate complex spectrum of f. f has to be expanded with zeros to the length of x_i before. The back transform of the product yields y_i. To avoid shifts introduced by the filtering, the expansion of f to the size of x_i should be done as follows:

$$f_{T_x} = (f(0), \ldots, f(T_f), \underbrace{0, \ldots, 0,}_{(T_x - 2T_f + 1)\times} f(-T_f), \ldots, f(-1)) \,. \tag{3.71}$$

This assures that the peak in y_i occurs at the middle of the wave form in x_i and not on its sides.

Step 5: Covariance matrix of the filtered recordings: The covariance matrix $\mathbf{C}$ of the filtered recordings, which is needed for peak detection, is computed from their empirical second moments,

$$C_{ij} = \frac{1}{T_y} \sum_{t=0}^{T_y-1} y_i(t) y_j(t)^T \,. \tag{3.72}$$

Step 6: Peak detection value: Following Rebrik et al. (1999) a hyper-ellipsoidal thresholding surface in the N-dimensional space of the filtered recordings is used. However, peaks with negative polarity shall be excluded (although after filtering most of the peaks are already positive). The non-negativity can be defined in the Mahalanobis metric Sahani (1999), which leads to

$$\mathbf{y}_+(t) = \max(\mathbf{C}^{-\frac{1}{2}}\mathbf{y}(t); 0) \,, \tag{3.73}$$

and the peak detection value $r(t)$,

$$r(t) = \mathbf{y}_+(t)^T\mathbf{y}_+(t) \,. \tag{3.74}$$

Step 7: Local maxima in peak detection value: The peaks in $r(t)$ still have an extension of a few samples. To get time discrete events, local maxima in $r(t)$ are determined, i.e. those values that exceed all others in the neighborhood $[-T_r, T_r]$,

$$r'(t) = \begin{cases} r(t) & : \quad r(t) > r(t+\tau) \quad \text{for all} \quad \tau \in [-T_r, T_r] \\ 0 & : \quad \text{else} \end{cases} \,. \tag{3.75}$$

Step 8: Clustering: The vectors $\mathbf{y}(t)$ for which the $r'(t)$ exceeds a preset threshold are used as input for clustering to finally make the assignment of the events to individual units. In principle any clustering algorithm can be applied. In the experiments below a *Gaussian Mixture Model* was used because it turned out to be robust, fast, and achieved good results.

3.5.3 Experiments

The spike-sorting method is based on fairly strong assumptions about the processes generating the data, in particular the existence of a unique wave form and the assumption that there were no delays between the recording channels. In order to demonstrate that it works correctly under real recording conditions, the performance of the method was tested on tetrode recordings, for which above assumptions would hold only approximately.

Figure 3.19 shows a period of 5s of the raw tetrode signals used for the following experiments. The entire recording was of approximately 100 seconds duration. Prior to further processing each signal was normalized for it's mean and variance. From these data a total amount of $5 \cdot 10^4$ samples of length 40 (1.28ms) for $\tilde{\boldsymbol{f}}_i(t)$ were selected randomly from all channels at random time positions to estimate the covariance matrix $\mathbf{H}$. Subsequently, the optimal filter f was computed using the fixed point iterations equation (3.20) and (3.20). The 40 elements of f were initialized randomly iid. according to the $\mathcal{N}(0;1)$ normal distribution. The procedure converged in less than 45 iterations.

In the bottom-left panel of figure 3.20 the resulting optimal filter is shown. The corresponding wave form (figure 3.20, bottom-center) was obtained from (3.24). Figure 3.20, bottom-right, shows the response of the filter to the wave form. The width of the impulse did not decrease very much compared to the width of the (negative) peak of the wave form because of the noise in the data. Hence, spikes closer than 0.2ms cannot be well separated. For larger distances separation quality has increased because the undershoots of the impulse response are much lower than in the original wave form. This leads to a higher accuracy of the peak amplitude values and to more pronounced clusters. The filtered data are illustrated in figure 3.20, center, for a close-up of the 3rd recording channel (For the reason of limited space only one out of four channels is shown). One can see now positive impulses that correspond to the former negative peaks in the raw data. Note that the average amplitude of the undershoots is minimized – as expected there is maximum skewness in this signal.

The filtering is indispensable in situations where the original wave form has no single prominent peak. This can happen, for example, if the recording channel has a poor frequency characteristic, and the wave form is distorted to a wavelet like shape with several maxima or minima. This can be simulated with the application of a narrow band pass filter to the original recordings. A closeup of channel 3 of the resulting raw recordings is shown in figure 3.21, top panel. Individual peaks and their amplitudes can hardly be detected. The

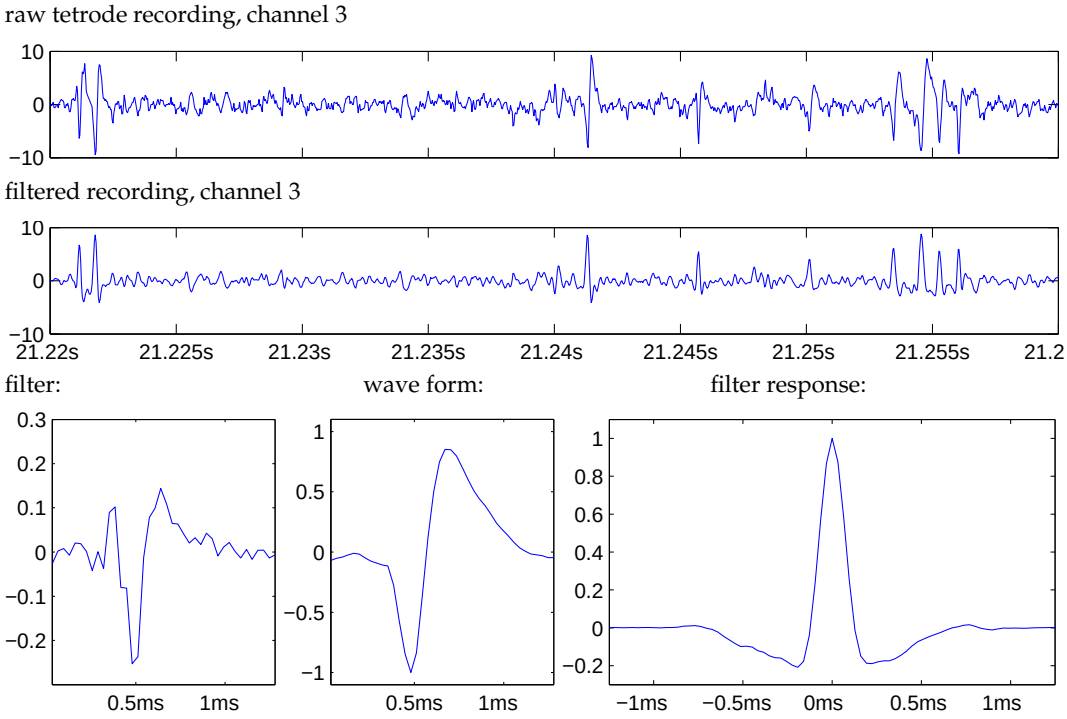


Figure 3.20: *Top panel:* A closeup of channel 3 of the raw tetrode recordings. *Center panel:* A closeup of channel 3 of the filtered tetrode recordings. *Bottom, left:* The optimal filter, learned from $5 \cdot 10^4$ small signal clips (1.28ms, 40 samples) taken randomly from all channels of the raw tetrode recordings. *Bottom, center:* The typical wave form derived from the filter using (3.24). *Bottom, right:* The response of the filter to the typical wave form.

optimal filter learned from this data together with the wave form and the filter response is shown on the bottom panels of the figure. After optimal filtering one obtains narrow peaks again (figure 3.21, center), which are easy to detect and the amplitudes of which correspond to those of figure 3.20, center.

From the filtered recordings the peak detection value was computed as the Mahalanobis distance to the origin, (3.74), (see figure 3.22, top panel). The time stamps of the peaks were extracted (figure 3.22, bottom panel) by taking the local maxima in a 15 samples neighborhood that exceed a threshold of 20. The amplitudes of the filtered recordings at the discrete points in time that mark an event peak form clusters in the 4d space. For clustering a *Gaussian Mixture Model (GMM)* with 8 Gaussian components was used, each with full covariance matrix. One has to mention that there may exist more sophisticated clustering algorithms that could be applied here, in particular those that are able to adapt to the number of clusters (cf. Duda et al. (2001)). After training of the GMM using *Expectation Maximization* (Dempster et al., 1977), the Gaussian components cover the individual clusters. Figure 3.23 shows a 2d projection of the 4d spike amplitudes (after optimal filtering) of the whole 100s recordings. The ellipses illustrate the components of the GMM as a projection of their unit standard deviation hyper-surfaces. Even in this projection there is noticeable margin between the clusters. The formation of well defined clusters can be taken as a performance indicator. Clearly, clusters with large overlap would not yield reliable separation. To quantify the margin between the clusters, the overlap of every Gaussian with all others was computed. This value is 1 minus the probability that for a data point belonging to one component, this component is the one with the largest posterior,

$$1 - Pr\{\arg\max_{\{c'\}}(P(c'|x)) = c\} \qquad \text{where} \quad x \sim \mathcal{N}(\mu_c; \Sigma_c)\,. \tag{3.76}$$

Coarse clusters with large overlaps indicate poor performance. For every cluster c the overlap values were computed numerically by drawing 10^5 random samples from the

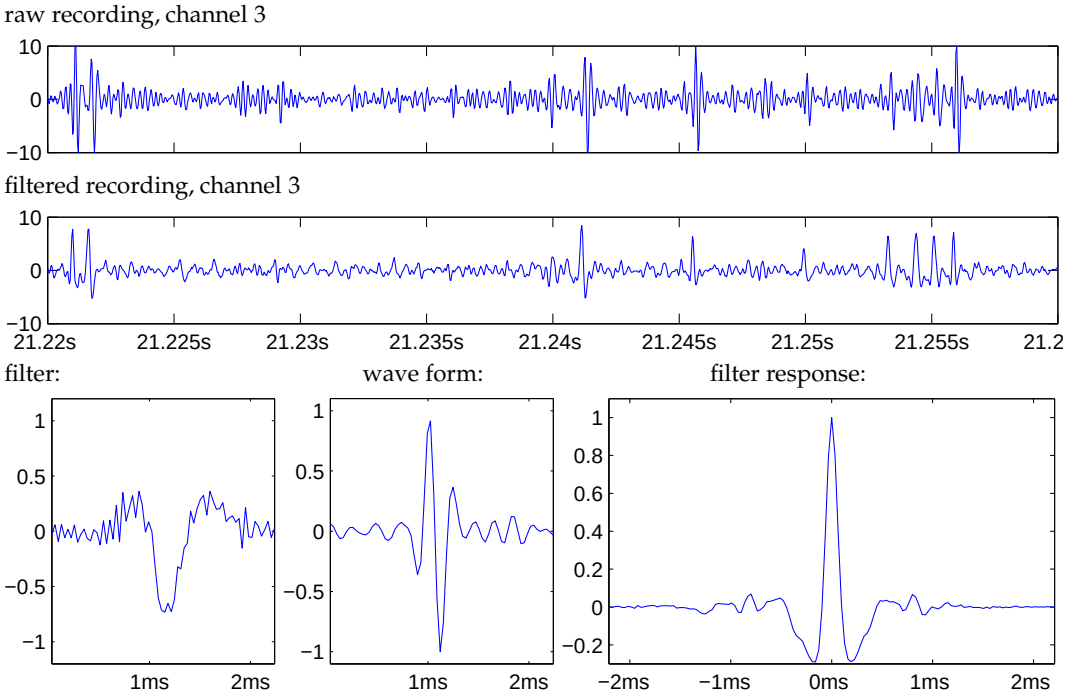


Figure 3.21: Performance of the filter for wave forms which have no single prominent peak. In the raw recordings (top row), peaks can hardly be detected. After filtering there are nice peaks (center row). Compare the peak amplitudes with those of figure 3.20, center.

$\mathcal{N}(\mu_c; \Sigma_c)$ normal distribution and replacing the probability Pr with the empirical probability $\hat{Pr}$. The overlap values are shown in the table next to figure 3.23. Apart from clusters 5 and 7 all others have an overlap of less than 4% in terms of the probability of misclassification. Note that for cluster 1 the overlap value is exactly zero because of the finite set of samples used to compute the empirical probability. The true overlap value would always be above zero.

Figure 3.24 shows the wave form templates of the detected and separated spikes for all 8 clusters (columns). It is not surprising that the spikes assigned to cluster 1 exhibit wave forms with the largest amplitudes and smallest fluctuations. Cluster 7 also yields large amplitude wave forms, however the overlap of cluster 7 was the largest. Note the large variance in the amplitude of the wave form (channel 1) and that the templates of clusters 7 and 3 are quite similar (beside the amplitude). This gives rise to the assumption that both clusters actually could be one, the spikes of which mostly occur in bursts with strong amplitude fluctuations. The same may apply to clusters 4 and 5.

In figure 3.25 the same templates but taken from the filtered recordings are shown. One can see that the wave forms now correspond to the filter response (cf. figure 3.20, bottom,right) indicating that the neuron individual wave forms have not been too different.

3.5.4 Experiments with realistic artificial data

In order to quantitatively evaluate the accuracy of the spike detection and classification, the true times of occurrence and the emitting units must be known for all foreground spikes in the recordings. Therefore realistic, artificial data sets were constructed, the basis of which is a four channel background signal. From segments of the real tetrode recordings shown in figure 3.19 which before had been identified to contain no foreground spikes the cross power spectral density (CPSD) matrix $\mathbf{C}_{PSD}(\omega)$ was estimated using standard methods. Then, a four-channel white, Gaussian noise signal was filtered with a four-channel filter matrix the frequency response of which was given by $\mathbf{F}(\omega) = \mathbf{C}_{PSD}^{\frac{1}{2}}(\omega)$. Thus, the artificial

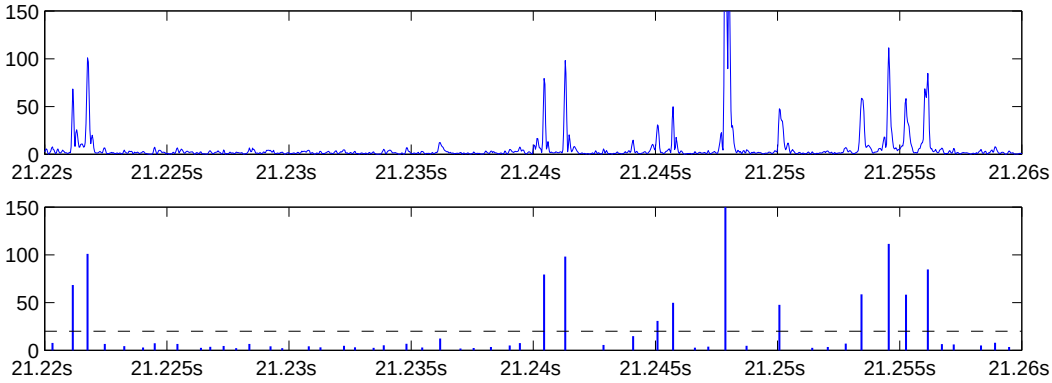


Figure 3.22: *Top panel:* The peak detection value (3.74) for the same clip as in figure 3.20, center, but derived from all four recording channels. *Bottom panel:* a threshold value of 20 (dashed line) is applied to those values, that are a local maximum in a 15 samples neighborhood (solid bars).

background has the same CPSD as the real background.

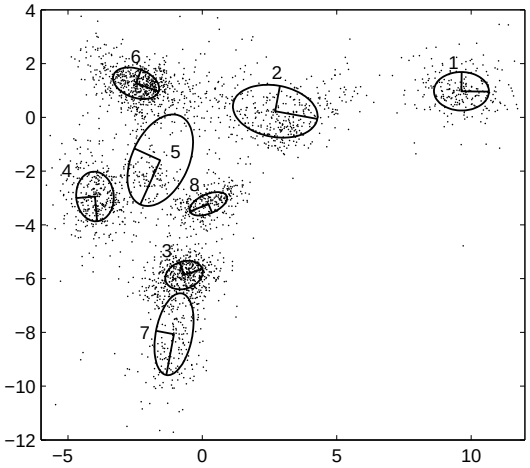
To construct the foreground signal, at first $M = 5$ intrinsic binary spike trains were generated. Although the derivation of the optimal filter required the intrinsic spike trains to be temporarily white and, hence, the inter-spike interval times exponentially distributed, the more realistic approach that the inter-spike interval times are subject to the $(\lambda; c)$ Gamma distribution was used here, where λ is a dimensionless shape parameter and c is a scale parameter carrying the dimension s. If for λ and c holds $\lambda = (\eta_i c)^{-1}$, then in the limit of large intervals, the resulting spike train has a spike rate of η_i spikes per sample interval (rsp. $31.250 \cdot \eta_i$ spikes per second). Holding this constraint the parameters λ and c were jointly adjusted such that the probability of two spikes being closer than a given minimal inter-spike interval time assumes a small value ($Pr(t_{ISI} < 0.64ms) = 10^{-4}$ in this case). Eventually, the spike times were discretized by rounding them to the closest sampling time.

Two basic types of foreground signals were used: one having unique wave forms and one having wave forms extracted from real recordings. Additionally a dataset with foreground wave forms that are a smooth interpolation between the real and the unique wave forms was used .

For the unique wave form data the binary spike trains were convolved with the wave form shown in figure 3.20. After mixing with a 4×5 mixing matrix, the elements of which were drawn from a $(0; 7)$ uniform distribution, the resulting foreground spike trains were added to the background signal. A 100ms example of the unique wave form toy data is shown in figure 3.26. One can see that the foreground signals have a rather low amplitude compared to the background. The mixing coefficients were intentionally chosen no larger to achieve a hard and challenging problem. For the experiments signals of 160s rsp. 5,000,000 samples length were used.

For the real wave form foreground signals the wave forms were used that had been extracted from the real tetrode recordings in the previous experiment (cf. figure 3.24). For every spike in the binary spike trains one of the detected spikes of the corresponding cluster in the real data experiment was randomly selected. Then, the 1.6ms neighborhood of that spike from the real tetrode recordings we added to the artificial background signals. The used spikes were those in the clusters 1, 6, 4, 2 and 7 respectively. Figure 3.27 shows a 100ms example of this type of toy data. The signal traces look very similar to the real recordings. This data is particularly challenging because beside the normal variability of the wave forms, also the background of the real recordings next to the spike appears in the toy spike train. With other words, it happens occasionly that a heavily distorted wave form is assigned to a spike in the toy data.

By means of these two types of artificial data it should be examined (i) whether the proposed filtering is advantageous in terms of the accuracy of the spike detection, (ii) how



Cluster	$1 - \hat{Pr}\{\arg\max_c(P(c\|x)) = c\}$
1	0.0000
2	0.0315
3	0.0230
4	0.0216
5	0.1047
6	0.0037
7	0.1840
8	0.0122

Figure 3.23: *Left:* Scatter plot of a 2d projection of the originally 4d spike amplitudes. Clusters where fitted by a 4d GMM with 8 components. The ellipses are the projections of the unit standard deviation hyper-surfaces of every Gaussian component. *Right:* The table shows how well every cluster is separated from the remaining ones (computed in the original 4d space). Values close to zero indicate best separation, values close to one indicate poor separation. For explanation see text.

the clustering of amplitude patterns performs in comparison to the widely used technique of clustering of principal component features of the recorded wave forms, and (iii) the influence of deviations from the assumptions of unique wave forms on the performance of the method.

In order to quantify the results two values were defined: the *true positives rate*, q^+, and the *false positives rate*, q^-. q^+ is the ratio of the number of correctly detected spikes to the total number of spikes. q^- is ratio of the number of mistakenly detected spikes to the number of detected spikes. One can easily see that $0 \leqslant q^+, q^- \leqslant 1$, where in the case of no spike detected, $q^- = 1$ was set. Clearly, for good performance q^+ should be close to 1, and q^- should be close to zero. Since there were no preferences how to trade q^+ against q^-, an overall performance measure,

$$q = q^+(1 - q^-) \,, \tag{3.77}$$

was defined. It still remains to define when a spike is correctly detected. Even on a discrete time scale, it may happen that in the course of processing the detected spike is assigned to a sampling interval shortly before or after the sampling interval where the original spike occurred. Thus, a detected spike was defined to be correctly detected if it has a binary source spike in a neighborhood of $\tau = \pm 4$ samples. This is the neighborhood considered for the computation of local maxima in the peak detection value (cf. section 3.5.2, step 7).

To evaluate the performance of the clustering procedure, also the values $q_i = q_i^+(1 - q_i^-)$ were calculated for every cluster i, where the true and the false positive rates, q_i^+ and q_i^-, were determined using the spikes of the cluster i only. Hence, a spike from source i that is correctly detected, but classified to unit j, increases q_j^- and decreases q_i^+, a spike from source i which remains undetected decreases q_i^+, and a spike that is mistakenly detected increases q_i^- for the cluster i this spike is assigned to. The clustering performance is summarized by an average 'clustering accuracy value'

$$\bar{q} = \sum_{i=1}^{M} q_i \,. \tag{3.78}$$

All performance measures clearly depend on the value of the threshold in the spike detection, where lower thresholds will increase the false positives rate and higher thresholds will decrease the true positives rate. In order to choose the threshold value, the values q and

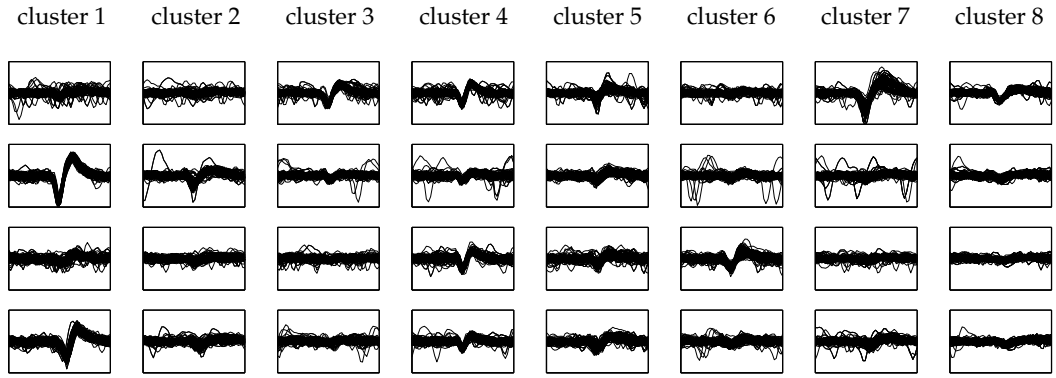


Figure 3.24: Composite plot of the 4-channel wave form templates according to the clusters shown in figure 3.23. For every detected spike a 1.6ms clip of all recording channels was plotted into the column of the corresponding cluster. The amplitude scale was identical for all plots.

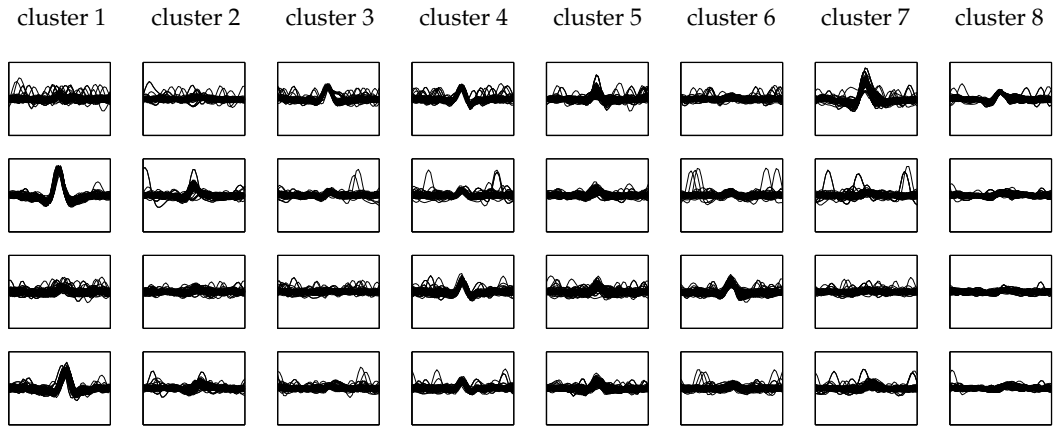


Figure 3.25: The same plot as in figure 3.24, but with the clips taken from the filtered recordings. The the wave form of the templates now is the filter response (figure 3.20, bottom,right) in different scalings.

$\bar{q}$ were computed for several thresholds in the range of 1 to 50, and the one with the best performance was taken. The clustering was done by means of Gaussian mixture models with full covariance matrices. They had 9 clusters (instead of 5, which is the number of sources) in order to have some 'spare' clusters available, that can collect false positive spikes, in particular in situations where the threshold is small. Clearly, the value $\bar{q}$ depends on the assignment of the 9 clusters to the 5 sources, so the assignment that maximized the value of $\bar{q}$ was always considered. It was found by solving a small combinatorial optimization problem by simulated annealing. Clearly, for a real application the assignment must be computed by other means, e.g. the cluster distances, minimal inter-spike times interval Fee et al. (1996), or stimulus related properties.

In the following experiments the influence of certain parameters of the spike generating process and the spike sorting algorithm was examined. In all experiments the following performance measures were computed:

$q_{\{x\}}, q_{\{y\}}$ Spike detection performance for the unfiltered, $q_{\{x\}}$, and the filtered, $q_{\{y\}}$, recordings. Steps 5 to 7 of section 3.5.2 were performed with the unfiltered and the filtered recordings. Since the wave form has negative polarity in the unfiltered recordings, in $q_{\{x\}}$ the peak detection value (3.74) was computed for $-\mathbf{x}(t)$.

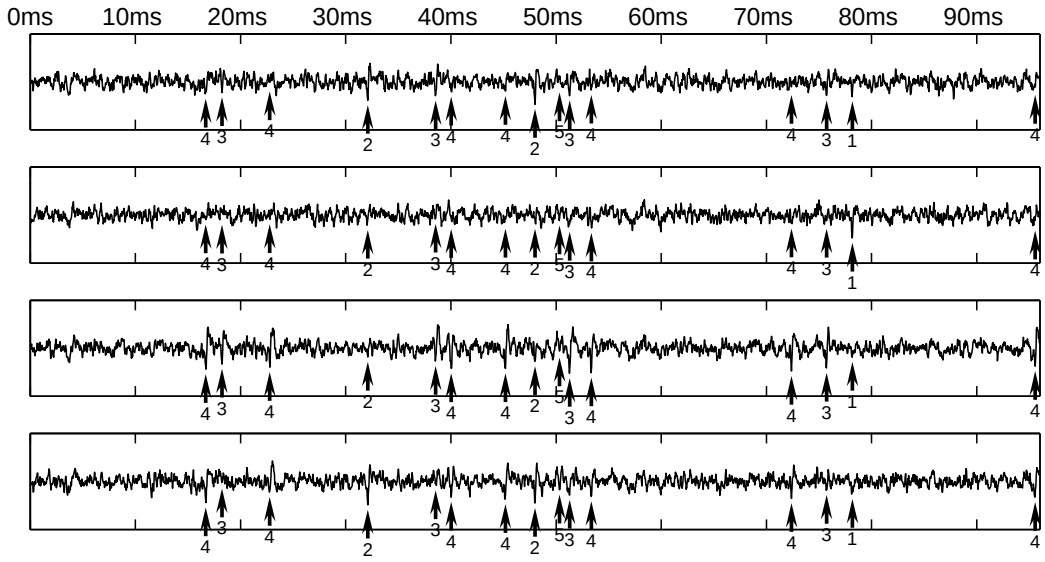


Figure 3.26: A 100ms example of the artificial data with unique wave forms for all neurons. Arrows indicate the individual foreground spikes and the number of the corresponding source.

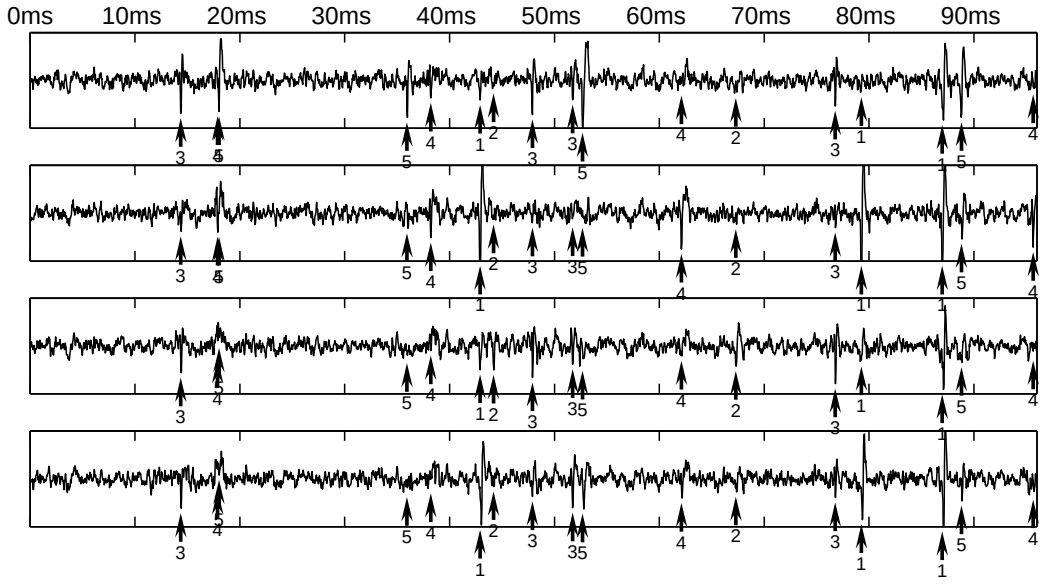


Figure 3.27: A 100ms example of the artificial data with realistic wave forms. Arrows indicate the individual foreground spikes and the number of the corresponding source.

$\bar{q}_{\{x\}}, \bar{q}_{\{y\}}$ Average clustering accuracy, (3.78), for the unfiltered rsp. the filtered recordings. For $\bar{q}_{\{x\}}$ the amplitudes of the raw recordings $\mathbf{x}(t)$ were clustered for the spikes detected from the raw recordings. For $\bar{q}_{\{y\}}$ the amplitudes of the filtered recordings $\mathbf{y}(t)$ were clustered for the spikes detected from the filtered recordings.

$\bar{q}_{\{x,PCA\}}$, $\bar{q}_{\{y,PCA\}}$ Average clustering accuracy, when principal components are clustered rather than amplitudes. For every detected spike a small window ranging from 10 samples before to 9 samples after the spike time is taken from all channels. All these 4×20 windows are projected onto their 7 largest principal components, and the resulting projection vectors are clustered with a Gaussian mixture model. This value was computed, because clustering principal components is a widely used technique the presented method should be compared with. Spikes were detected and windows were taken from the unfiltered recordings for $\bar{q}_{\{x,PCA\}}$ and from the filtered recordings for $\bar{q}_{\{y,PCA\}}$. The window was 4×20 samples, and 7 principal components were used because these were the settings which showed best performance.

In a first experiment the influence of the spike rate η_i on the performance was investigated for both types of foreground signals, the unique wave form data and the real wave form data. The left panel of figure 3.28 shows the results for the toy data with unique wave forms averaged over 5 independent trials with different mixing matrices. The right panel shows the results with real wave forms, where there was only one setup. For all spike rates and both types of data, the detection accuracy and the clustering performance are apparently better when the optimal filter is applied. For the filtered data clustering of the the spike amplitudes, $\bar{q}_{\{y\}}$, performs better than clustering of principal components, $\bar{q}_{\{y,PCA\}}$, in particular for high spike rates. Interestingly, for low spike rates it happens that $\bar{q}_{\{x,PCA\}} > q_{\{x\}}$. This result is only possible, because there were more clusters than sources, and in the assignment procedure some clusters were allowed to be not assigned to any source. This way these clusters could pickup false positives and, hence, decrease the q^- rates. Note, however, that also in these situations $\bar{q}_{\{x,PCA\}}$ was apparently worse than both types of clustering procedures with the filtered data.

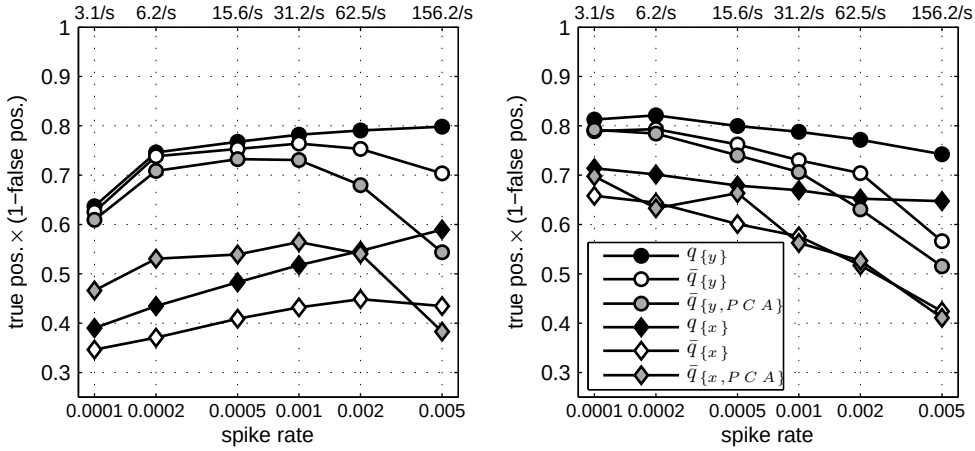


Figure 3.28: Influence of the spike rate η (labeled in spikes per sample at the bottom and spikes per second at the top axis) for toy data with unique wave forms (left) and real wave forms (right). The spike rates were modified for all sources simultaneously. For the unique wave form data, 5 independent trials were averaged. For the real wave form data there was only one trial. In all cases the detection performance is better with the filtered data, where the clustering of amplitudes performs better than the clustering of principal components.

In the next experiment was explored how the performance of the new method depends on the variability of the individual spike wave forms. Therefore a third data set was

constructed, in which the foreground signal could be continuously morphed from the unique wave form to the real wave form setting, controlled by a parameter $\beta \in [0, 1]$. For the limit $\beta = 0$ the wave forms were composed with the unique ξ such that they best represent the average of all real wave forms of the corresponding cluster in terms of squared differences. For $\beta > 0$ a linear superposition of unique and real wave forms weighted by $1 - \beta$ rsp. β was used. Figure 3.29, left, shows the performance measures q as functions of the wave form variability. The rate of the source spike trains was $\eta = 0.001$ (31.25 spikes/s). In general one can state that the performance of all methods is negatively influenced by increasing wave form variability, but one can also see that clustering the filtered amplitudes, $\bar{q}_{\{y\}}$, is over the whole range of β superior to the other methods. Also in this experiment, for small β and in particular for the unfiltered recordings, the clustering accuracy was larger than the detection accuracy. For the filtered data this effect is less apparent because here the detection accuracy was already considerably improved.

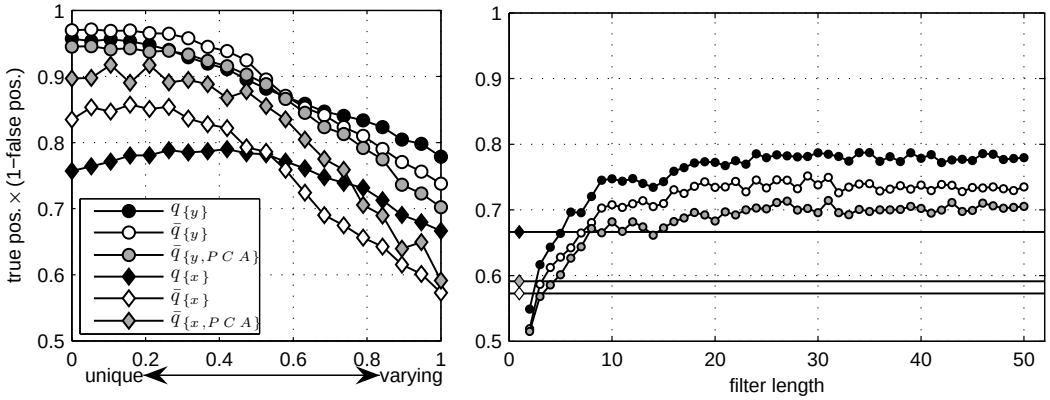


Figure 3.29: *Left:* Spike sorting performance as functions of the value of the wave form variability parameter β. For explanations see text. *Right:* Spike sorting performance as functions of the filter length for the real wave form dataset. For filters longer than 5 samples, the detection and classification performance improves compared to the unfiltered recordings. For all other experiments the filter length 40 samples was used.

Figure 3.29, right, shows the influence of the filter length on the performance for the real wave form data set. The values $q_{\{x\}}$, $\bar{q}_{\{x\}}$, and $\bar{q}_{\{x,PCA\}}$ were computed from the unfiltered data and $q_{\{y\}}$, $\bar{q}_{\{y\}}$, and $\bar{q}_{\{y,PCA\}}$ for filters in the range of 2 to 50 samples length. Starting with filters longer than 5 samples, the analysis of the filtered data yield better detection and classification results than the analysis of the unfiltered data. Above a length of approximately 20 samples, increasing filter length do not further improve the spike sorting performance. Again, clustering of amplitudes outperforms clustering of principal components.

3.5.5 Outlook

A optimal filtering method for spike sorting of extra-cellular neural activities based on the 'stereo effect' that can be achieved with multi-site electrodes was demonstrated. It could be shown that the amplitude pattern of the action potentials is a powerful feature to discriminate the individual neurons. Further performance gain can be expected by the use of *heptodes*, 7-cores multi-fiber electrodes Thomas RECORDING Gmbh (2002). The space of recorded amplitudes, i.e. the space in which the clustering is eventually done, would be larger by three dimensions. In general the performance will behave approximately like the ratio of recording channels (i.e. dimensions) to the number of foreground spiking units. Thus, for closely spaced electrode tips, the higher dimensionality provides more space between the individual clusters allowing to separate them robustly. Deviations from the model assumption, which generally lead to less pronounced clusters, can then be better tolerated.

For the future one could think of an online implementation of the algorithm, with the objective to analyse tetrode recordings until the clustering in real time. This is certainly a demanding, but still realistic aim. For long experiments, it is not feasible to save the raw recordings continuously. Thus, it would be highly desirable to have filtering and peak detection at hands in real time. It would then be possible to save only data of the event times together with the amplitude vector. Clustering could then be performed off-line to reconstruct the binary separated spike trains. Also it would be nice to have a scatter plot projection like in figure 3.23 online, because the instant feedback about the 'cluster situation' in the space of event amplitudes would help to place the tetrode or correct the electrode position when it starts sliding through the tissue during recording. Spike sorting would be most reliable when this plot exhibits pronounced clusters already. Once the optimal filter had been found, a linear filtering operation, a squared form of order 4 (the Mahalanobis distance), and a maximum search over a small window (peak detection) had to be performed for every sample. These operations are cheap and could be implemented with the help of digital signal processors. The quantities that are needed to support these operations, c, $\mathbf{H}$, and f, could be computed in parallel. As these quantities are more or less stationary, it would not be necessary to perform an update step for every sample. At worst these quantities would need some time to settle, still allowing to adapt to slow changes, e.g. in the location of the tetrode.

Chapter 4

Non-linear filtering

In many signal processing applications, filters according to equations (1.30) or (1.35) arise. In this chapter we will investigate what the computational overhead of the calculation of the filter function is. To keep the considerations as simple as possible we will, without loss of generality, consider only one-dimensional, scalar valued signals and simple filter functions at first.

Clearly the computational load of a filter function can be evaluated meaningfully only for a finite piece of the stochastic signal, which we assume to be the interval $[0, T_x - 1] \subset \mathbb{Z}$. Thus one observation of the stochastic process is given by a family of T_x realizations of the random variable x_t,

$$(x_t)_{t \in [0, T_x - 1]} \in \mathbb{R} \,.$$

Without loss of generality we consider only filters for which the receptive field J is a closed interval, just like equation (1.32). Other receptive field shapes J′ can be expressed by the smallest interval J that contains all elements of J′, together with the corresponding filter function that is invariant to changes of x_t for $t \in \mathrm{J} \backslash \mathrm{J}'$. For convenience of notation we start the interval at 0. Thus, for the one-dimensional case we have

$$\mathrm{J} = [0, T_f - 1] \,. \tag{4.1}$$

Because in this setup one can obtain values y_t in the interval $t \in [0, T_x - T_f]$, one usually assumes the receptive field to be much smaller than the length of the given signal observation,

$$T_f \ll T_x \,. \tag{4.2}$$

4.1 Linear filters and the convolution theorem

For the linear filter the response

$$y_t = \sum_{\tau=0}^{T_f} w_\tau x_{t+\tau} \tag{4.3}$$

has to be computed for $T_x - T_f + 1$ indices t. This results in a computational load of approximately $O(T_x T_f)$.

Equation (4.3) is the cross correlation between x and w, and one can make use of the convolution theorem to compute this as a direct product in the frequency domain,

$$Y_k = X_k \,\mathrm{conj}(W_k) \,, \tag{4.4}$$

where

$$y_t = \mathcal{F}^{-1}[Y_k] = \frac{1}{T_x} \sum_{k=0}^{T_x - 1} Y_k \exp\left(-\frac{2\pi \mathrm{j} k t}{T_x}\right)$$

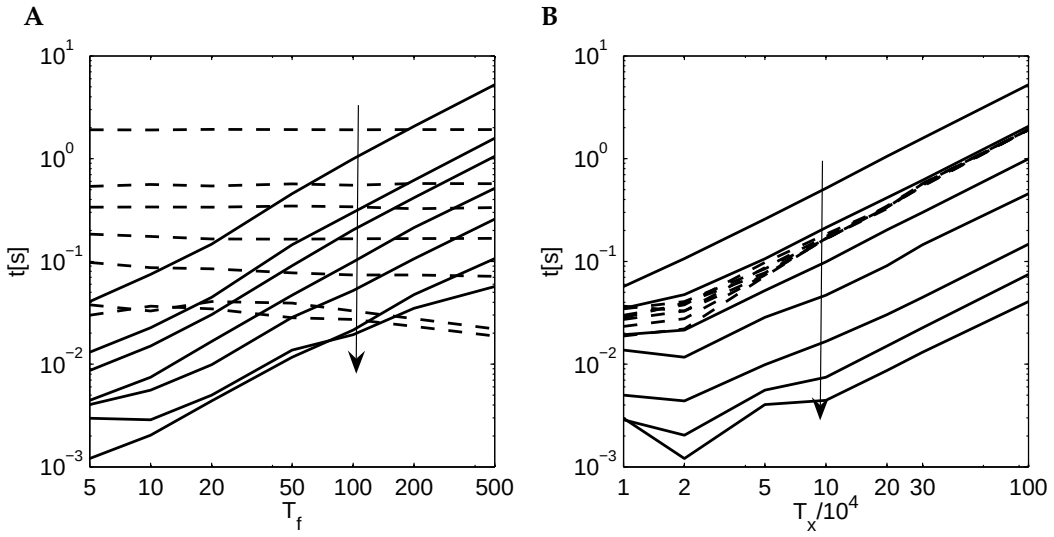


Figure 4.1: Average computation time for the computation of cross correlation functions in the time domain (solid lines) and in the frequency domain (dashed lines). **A:** Computation time as a function of T_f. The arrow indicates decreasing values of T_x form 10^6 to 10^4. **B:** Computation time as a function of T_x. The arrow indicates decreasing values of T_f form 500 to 5.

is the inverse Discrete Fourier Transform (DFT) of Y_k, and

$$X_k = \mathcal{F}[x_t] = \sum_{t=0}^{T_x-1} x_t \exp\left(\frac{2\pi \mathrm{j} k t}{T_x}\right)$$
$$W_k = \mathcal{F}[w_t] = \sum_{t=0}^{T_f-1} w_t \exp\left(\frac{2\pi \mathrm{j} k t}{T_x}\right)$$

are the DFTs of x_t and w_t, respectively. Note that with equation (4.4) a cyclic cross correlation is actually performed so that for $t > T_x - T_f$ the values of y_t are subject to errors. However, these values do not at all occur in the direct computation and, hence, can be discarded. For $t \leqslant T_x - Tf$ the results of equations (4.3) and (4.4) are identical. However the computational load for the computation in the frequency domain is approximately $O(T_x \log T_x)$, thus this approach can become advantageous when the filter length T_f is large.

One must note that the break-even point, at which both methods perform equally efficient, is not necessarily given by $T_f = \log T_x$ because in the overhead estimations any constants and factors are omitted (and we did not even specify the base of the logarithm). For a particular performance comparison consider figure 4.1. There, the average computation times for the computation of cross correlation functions in the time domain and in the frequency domain are shown for values of $(T_x, T_f) \in \{10^4, 2\cdot10^4, 5\cdot10^4, 10^5, 2\cdot10^5, 5\cdot10^5, 10^6\} \times \{5, 10, 20, 50, 100, 200, 500\}$. All computations where programmed in C using FFTW[1] for the computation of the DFT. One can see that the computation cost for the time domain method grows linearly with T_x and T_f, while in the frequency domain approach it is constant in T_f and virtually grows linearly with T_x. In this example the break even between point is at $T_f \approx 150 \ldots 200$.

4.2 Non-linear filters

The convolution theorem and the computation of linear filters in the frequency domain can lead to a strong performance gain, in particular for filters with large receptive fields. So

[1] available as free software at `http://www.fftw.org/`

the question comes close whether a similar approach exists for non-linear filter functions. Unfortunately, there is no equivalent to the convolution theorem for other than linear operations. However, in many situations it is possible to reduce a non-linear filter to an instantaneous, non-linear function of a finite number of linear filters,

$$(y_t)_{t\in\mathrm{I}} = (f(y'_{1,t},\ldots,y'_{N,t};\mathbf{w}'))_{t\in\mathrm{I}}\,, \qquad y'_{n,t} = \sum_{\tau\in\mathrm{J}} w_{n,\tau}x_{t+\tau}\,, \tag{4.5}$$

or at least approximate it as such. The instantaneous non-linear functions then can be computed with $O(T_x)$, which may lead to the desired significant performance gain.

4.2.1 Taylor expansion

If the computation of the filter function itself is expensive, then it may be sometimes advantageous to consider it's Taylor expansion (or truncated Taylor expansion) and see if terms vanish or can be simplified. An analytic filter function f can be represented by the series

$$f((x_{t+\tau})_{\tau\in\mathrm{J}}) = f((b_\tau)_{\tau\in\mathrm{J}}) + \sum_{d=1}^{\infty}\frac{1}{d!}\sum_{(\tau_1,\ldots,\tau_d)\in\mathrm{J}^d} g_{\tau_1,\ldots,\tau_d}\prod_{i=1}^{d}(x_{t+\tau_i}-b_{\tau_i})\,, \tag{4.6}$$

where

$$g_{\tau_1,\ldots,\tau_d} = \frac{\partial}{\partial x_{\tau_1}}\cdots\frac{\partial}{\partial x_{\tau_d}} f((x_\tau)_{\tau\in\mathrm{J}})\bigg|_{x=b}\,. \tag{4.7}$$

The d-th term of the series involves the multilinear multiplication of the vector $(x_{t+\tau}-b_\tau)_{\tau\in\mathrm{J}}$ with the tensor g in d dimensions. A straight forward computation of this term would contain $1+T_f+\ldots+T_f^d$ multiplications and $(T_f-1)(1+T_f+\ldots+T_f^{d-1})$ additions. Thus, the computation of a non-linear filter by means of a taylor approximation up to the degree d has a computational complexity of $O(T_x T_f^d)$ and a storage demand of $O(T_f^d)$. This may become intractable already for small d.

However, in some special cases the taylor series can have a particularly simple structure so that the computational complexity can be greatly reduced. If, for example, the d-th tensor g is diagonal,

$$g_{\tau_1,\ldots,\tau_d} = g_{\tau_1,\ldots,\tau_d}\delta_{\tau_1,\ldots,\tau_d}\,, \tag{4.8}$$

then the multilinear multiplication reduces to

$$\begin{aligned}\sum_{(\tau_1,\ldots,\tau_d)\in\mathrm{J}^d} g_{\tau_1,\ldots,\tau_d}\prod_{i=1}^{d}(x_{t+\tau_i}-b_{\tau_i}) &= \sum_{\tau\in\mathrm{J}} g_{\tau,\ldots,\tau}(x_{t+\tau}-b_\tau)^d \\ &= \sum_{i=0}^{d}\binom{d}{i}\sum_{\tau\in\mathrm{J}} x_{t+\tau}^i\,(-b_\tau)^{d-i}g_{\tau,\ldots,\tau}\,.\end{aligned} \tag{4.9}$$

The right hand side of this equation represents $d+1$ cross correlation functions (of instantaneous powers of x and b), which can be computed according to equation (4.4) with $O(d\,T_x\log T_x)$ complexity. If the origin of the expansion, b, equals zero, just one cross correlation function of x^d with the diagonal vector of g remains.

More often, however, it will be the case that g is not diagonal. But by construction it is always symmetric with respect to permutations π of it's indices,

$$g_{\tau_1,\ldots,\tau_d} = g_{\pi(\tau_1,\ldots,\tau_d)}\,. \tag{4.10}$$

Thus, there exists a $T_f\times N$ matrix with elements $(v_{\tau,n})_{\tau\in\mathrm{J},\,n\in 1\ldots N}$ and a vector $\boldsymbol{\lambda}\in\mathbb{R}^N$ so that g can be decomposed as

$$g_{\tau_1,\ldots,\tau_d} = \sum_{n=1}^{N}\lambda_n\prod_{i=1}^{d}v_{\tau_i,n}\,. \tag{4.11}$$

This is the special case for super-symmetric tensors of the *CANDECOMP–PARAFAC* decomposition (CANonical DECOMPosition or PARAllel FACtors model), which was independently proposed in Carroll and Chang (1970) and Harshman (1970) (cf. also Lathauwer et al. (2004)). The decomposition allows to compute the tensor multiplikation as

$$\begin{aligned}\sum_{(\tau_1,\ldots,\tau_d)\in\mathsf{J}^d} g_{\tau_1,\ldots,\tau_d}\prod_{i=1}^{d}(x_{t+\tau_i}-b_{\tau_i}) &= \sum_{n=1}^{N}\lambda_n\sum_{(\tau_1,\ldots,\tau_d)\in\mathsf{J}^d}\prod_{i=1}^{d}v_{\tau_i,n}(x_{t+\tau_i}-b_{\tau_i}) \\ &= \sum_{n=1}^{N}\lambda_n\left(\sum_{\tau\in\mathsf{J}}v_{\tau,n}(x_{t+\tau}-b_\tau)\right)^d . \end{aligned} \tag{4.12}$$

Equation (4.12) requires the computation of N cross correlation functions of x with all column vectors of v and subsequently taking it to the instantaneous power of d. Again, the cross correlation function can be computed in the frequency domain so that equation (4.12) leads to a complexity of $O(NT_x \log T_x)$.

Unfortunately, the smallest possible number of linear forms, N, required to decompose general tensors is only known for some small values of T_f and d (cf. Comon and Mourrain (1996) and references therein). It has been shown, anyhow, that

$$N \leqslant \binom{T_f + d - 2}{d-1} \tag{4.13}$$

for arbitrary values of T_f and d, which, however, does not mean that a decomposition with N holding that bound can always be found.

For $d = 2$ equation (4.11) constitutes the eigenvalue decomposition of the matrix g, and N is at most T_f provided g has full rank. Further, there are efficient algorithms to compute the eigenvalue decomposition and the memory requirements to store $T_f \times T_f$ matrices is more or less moderate compared to higher order tensors. Thus, quadratic functions are of outstanding importance for non-linear filtering, as it will be shown in the following sections.

4.2.2 Linear and radial basis function networks

We have seen that the Taylor expansion of an arbitrary filter function may not always yield satisfactory results in terms of computational costs. However, one can always construct non-linear filter functions that can be efficiently computed if they have the general form of a basis function network,

$$y_t = f\left(f_1((x_{t+\tau})_{\tau\in\mathsf{J}};\mathbf{w}_1),\ldots,f_N((x_{t+\tau})_{\tau\in\mathsf{J}};\mathbf{w}_N)\right) , \tag{4.14}$$

and if the basis functions f_i can be efficiently computetd. The output function f, wheter it is linear or non-linear, is instantaneous and has a complexity of only $O(T_x)$. Thus, the overall complexity is dominated by the basis functions. According to the results of section 4.2.1 usually only linear,

$$f_i((x_{t+\tau})_{\tau\in\mathsf{J}};w_i) = \phi_i\left(\sum_{\tau\in\mathsf{J}}x_{t+\tau}w_{i,\tau}+w_{i,0}\right) , \tag{4.15}$$

or quadratic,

$$f_i((x_{t+\tau})_{\tau\in\mathsf{J}};w_i,g_i) = \phi_i\left(\sum_{(\tau_1,\tau_2)\in\mathsf{J}^2} g_{i;\tau_1,\tau_2}(x_{t+\tau_1}-w_{i,\tau_1})(x_{t+\tau_2}-w_{i,\tau_2})\right) , \tag{4.16}$$

functions can be considered as basis functions.

Fortunately, these two classes of basis function networks cover most types non-recurrent function approximators used in mashine learning (Haykin, 1999):

- **Multi Layer Perceptrons (MLP):** Equation (4.15) constitutes the first layer of an MLP with N neurons in the first hidden layer. The squashing function $\phi = \phi_i$ is usually one and the same for all hidden neurons. Often the tangens hyperbolicus, the sigmoid function, or the signum functions are used. $w_{i,\tau}$ and $w_{i,0}$ are the learned weigths rsp. biases of the first layer. Subsequent hidden layers and the output layer are expressed by means of the function f in equation (4.14).

- **Radial Basis Function (RBF) Networks:** The first layer of this type of function approximators has the form of equation (4.16). Usually ϕ_i is the exponential function, and often g_i is multiple of the unit matrix, $g_i = -\frac{1}{2\sigma_i}\mathbf{I}$, leading to a spherical basis function and computation according to equation (4.9). In RBF networks usually there are no subsequent hidden layers. Thus f is a linear function.

- **Support Vector Mashines (SVM)**: A trained SVM for regression has the form of equation (4.14) if N equals the number of support vectors with non-zero coefficients, and the kernel has the form (4.15) or (4.16). With the commonly used RBF kernels a SVM is equivalent to a RBF network. The also commly used polynomial kernels have the form (4.15) where ϕ_i is the power of the kernel. Usually SVM make use of one and the same kernel function $\phi = \phi_i$ for all support vectors w_i , and the ouput function f is always linear.

4.3 Application: electron microscopy data

This section describes an application of non-linear filtering in the frequency domain coming from the field of biomedical image analysis. In electron micrograph images of photoreceptor terminals of the fruit fly, *Drosophila*, synaptic vesicles containing neurotransmitter shall be detected and labeled automatically. Hand written labels provided by human experts are used to learn an RBF filter using Support Vector Regression with Gaussian kernels. The calculation of the RBF filter output is usually very expensive. However, the Gaussian basis functions can be decomposed into a set of linear filters that can be computed efficiently in the frequency domain yielding dramatic improvement in speed.

The results show that the resulting nonlinear filter solves the task to a degree of accuracy, which is close to what can be achieved by human experts. This allows the very time consuming task of data evaluation to be done automatically or computer assisted.

4.3.1 Introduction

Using filters for image processing can be understood as a supervised learning method for classification and segmentation of certain image elements. A given training image would contain a target that should be approximated by some filter at every location. In principle, any kind of machine-learning techniques could be employed to learn the mapping from the input receptive field of the filter to the target value. The most simple filter is linear mapping. It has the advantage that it can be very efficiently computed in the frequency domain. However linear filters may not be complex enough for difficult problems. The complexity of nonlinear filters is in principle unlimited (if one leaves generalization issues aside), but the computation of the filter output can be very time consuming. However, for nonlinear filters, that are linear superpositions of Gaussian radial basis functions, there exists a decomposition into linear filters allowing the filter output to be computed in reasonable time. This sort of nonlinear filtering is for example obtained, when Support Vector Machines (SVM) with a Gaussian kernel are used for learning. SVM have proved to yield good performance on many applications. This and the ability to compute the filter output in an affordable time make SVM interesting for nonlinear filtering in image processing tasks. In the following such filters are applied to the evaluation of electron micrograph images taken from the visual system of the fruit fly, Drosophila, as a means to screen new genetic mutants.

Genetically manipulable organisms such as Drosophila provide means to address many current questions in neuroscience. The action, even of lethal genes, can be uncovered in photoreceptors by creating homozygous whole-eye mosaics in heterozygous flies Stowers and Schwarz (1999). Mutant synaptic phenotypes are then interpretable from detailed ultra-structural knowledge of the photoreceptor terminals R1-R6 in the lamina Fabian-Fine et al. (2003). Electron microscopy (EM) alone offers the resolution required to analyze sub-cellular structure, even though this technique is tedious to undertake. As representative datasets showing the feasibility of the proposed method, there were two datasets from wild type Drosophila, `ter01` for training and `ter04` for performance evaluation (cf. figures 4.3 and 4.4, top, respectively), and one from a visual system mutant, `ter08`, also for performance evaluation (cf. figure 4.4, bottom). In Drosophila genetics hundreds of mutants of the visual system have been isolated, many even from a single genetic screen. The task of analyzing each of these mutants manually is simply not feasible, hence reliable automatic (computer assisted) methods are needed. The focus here is just to count the number of synaptic vesicles, but in general the proposed method could be extended to the analysis of other structures as well.

4.3.2 Learning the RBF Filter

Given an image $(x_t)_{t\in \mathrm{I}}$, where $\mathrm{I} = [0, T_{x_1} - 1] \times [0, T_{x_2} - 1] \subset \mathbb{Z}^2$ and $x_t \in \mathbb{R}$, we want to find a simple filter function f the output of which is closest to a target image $(y_t)_{t\in \mathrm{I}'}$, in terms of some suitable distance measure.

The filter is constrained to some receptive field $\mathrm{J} \subset \mathbb{Z}^2$ so that it's output would be formulated in the most general form as

$$z_t = f((x_{t+\tau})_{\tau\in \mathrm{J}}) \,,$$

and z_t is defined for all $t \in \mathrm{I}' = \{t : t + \tau \in \mathrm{I} \text{ for all } \tau \in \mathrm{J}\}$. The filter f shall be realized as an RBF network with $M = |\mathrm{J}|$ input dimensions. It can be implemented as a feed forward net with a single hidden layer of RBF units and a linear output layer (Haykin, 1999; Bishop, 1995). However we would rather use the technique of *Support Vector Regression (SVR)* Vapnik (1995) as it has a number of advantages over RBF feed forward networks. It offers adjustable model complexity depending on the training data, thus providing good generalization performance. The training of SVR is a quadratic, constrained optimization problem, which can be solved efficiently without being trapped into local minima.

In the linear case the formulation of the $\nu-$SVR, as it was introduced in Schölkopf et al. (2000), translated in our notation is:

minimize

$$L((w_\tau)_{\tau\in \mathrm{J}}, \xi^{(*)}, \varepsilon) := \frac{1}{2}\sum_{\tau\in \mathrm{J}} w_\tau^2 + c \cdot \left(\nu\varepsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)\right) \tag{4.17}$$

subject to

$$\left(w_0 + \sum_{\tau\in \mathrm{J}} w_\tau x_{t+\tau}\right) - y_t \leqslant \varepsilon + \xi_t \tag{4.18}$$

$$y_t - \left(w_0 + \sum_{\tau\in \mathrm{J}} w_\tau x_{t+\tau}\right) \leqslant \varepsilon + \xi_t^* \tag{4.19}$$

$$\xi_t^{(*)} \geqslant 0, \varepsilon \geqslant 0 \tag{4.20}$$

The constraints implement the ε-insensitive loss,

$$|y_t - f((x_{t+\tau})_{\tau\in \mathrm{J}})|_\varepsilon = \max\{0, |y_t - f((x_{t+\tau})_{\tau\in \mathrm{J}})| - \varepsilon\}, \tag{4.21}$$

as a distance measure, which is a basic feature of SVR and has been shown to yield robust estimation (cf. figure 4.2). The objective itself provides a solution of low complexity (small $\sum_{\tau\in \mathrm{J}} w_\tau^2$) and, at the same time, low errors balanced by c.

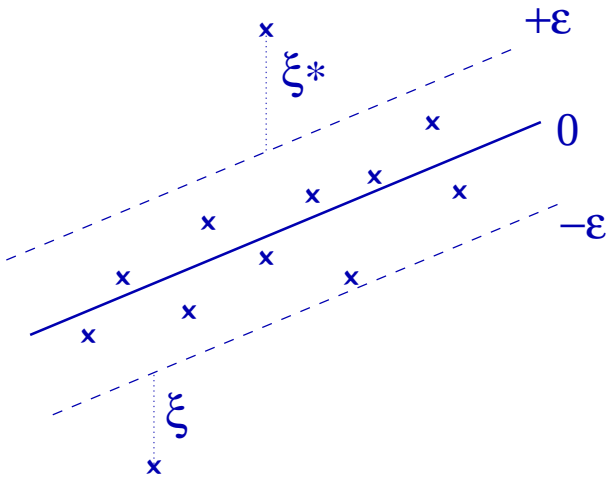


Figure 4.2: ε-sensitive loss in support vector regression. Points inside the ε-tube do not contribute. For outliers the slack variables $\xi^{(*)}$ are increased by the distance to the tube.

In contrast to $\varepsilon-$SVR, as it was introduced at first in Vapnik (1995), parameterization with the hyper parameter ν also allows optimization for the width ε of the insensitive region. Interacting with c, ν controls the complexity of the model. It provides an upper bound on the fraction of outliers (samples that do not fall into the epsilon tube) and a lower bound on the fraction of support vectors (SV). See Schölkopf et al. (2000) and Schölkopf and Smola (2002) for further details. As usual for SVM, the system is transformed into a nonlinear regressor by replacing the scalar product with a kernel, that fulfills Mercers condition (Mercer, 1909). With a Gaussian kernel (RBF kernel) the regression function is

$$z_t = \sum_{i=1}^{N} \alpha_i^{(*)} z_{i,t} + b \,, \tag{4.22}$$

where

$$z_{i,t} = k\left((w_{i,\tau})_{\tau\in \mathrm{J}}, (x_{t+\tau})_{\tau\in \mathrm{J}}\right) = \exp\left(-\frac{1}{\gamma}\sum_{\tau\in \mathrm{J}} (w_{i,\tau} - x_{t+\tau})^2\right) \tag{4.23}$$

is the Gaussian- or RBF-kernel. The resulting SVs w_i are a subset of the training examples, for which the constraints (4.18) and (4.19) hold with equality. They are assigned Lagrange multipliers $\alpha_i^{(*)} = (\alpha_i - \alpha_i^*) \neq 0$ to. In the analogy to an RBF network, the SVs are the centers of the basis functions while $\alpha_i^{(*)}$ are the weights of the output layer.

4.3.3 RBF Filtering

To evaluate an RBF network filter at location $t \in \mathrm{I}'$, all the basis functions have to be evaluated for the neighborhood $(x_{t+\tau})_{\tau\in \mathrm{J}}$. This calculation is computationally very expensive when computed in the straightforward way given by (4.23). According to equation (4.9), however, we can write the kernel as

$$z_{i,t} = \exp\left(-\frac{1}{\gamma}\left(\sum_{\tau\in \mathrm{J}} w_{i,\tau}^2 - 2z'_{i,t} + z''_{i,t}\right)\right) \,, \tag{4.24}$$

where

$$z'_{i,t} = \sum_{\tau\in \mathrm{J}} w_{i,\tau} x_{t+\tau} \qquad \text{and} \quad z''_{i,t} = \sum_{\tau\in \mathrm{J}} x_{t+\tau}^2 \,. \tag{4.25}$$

Now we are left with linear filtering operations only: the two cross correlations z' and z'', which can be efficiently computed in the frequency domain. There, the cross correlation of a

Table 4.1: Computation time examples for different filtering methods.

filtered according to equation (4.23)	6d 10h
FFT filtered, whole image	55m
FFT filtered, tiles of 256 × 256	24m

- image size 1686 × 1681 pixel
- 200 SV of 50 × 50 pixels size
- implementation in *MATLAB*
- *SUN F6800/750MHz*

signal with some filter becomes a multiplication of the signal's spectrum with the conjugate complex spectrum of the filter. This operation is so much faster that it offsets the additional computation costs of the Fourier transform. Note that in fact z'' is the cross correlation of x^2 with the filter $o_t = \delta(t \in \mathrm{J})$, which is 1 for $t \in \mathrm{J}$ and 0 everywhere else. We need to compute the following Fourier transforms:

$$\begin{aligned} X_k &= \mathcal{F}[x_t]\,, & X_k^{(2)} &= \mathcal{F}[x_t^2]\,, \\ W_{i,k} &= \mathcal{F}[w_{i,t}]\,, & O_k &= \mathcal{F}[o_t]\,. \end{aligned} \tag{4.26}$$

Now z_i is easily computed as

$$z_{i,t} = \mathcal{F}^{-1}\left[X_k^{(2)}\mathrm{conj}(O_k) - 2X_k\mathrm{conj}(W_{i,k})\right] + \sum_{\tau\in\mathrm{J}} w_{i,\tau}^2\,. \tag{4.27}$$

$W_{i,k}$ and O_k must be computed with the size of x. Therefore, w_i and o are zero filled to the required size and for all $\tau \notin \mathrm{I}$. It is necessary to take care of the placement of the origin $\tau = 0$ which depends on the implementation of the Fourier transform.

Also depending on the implementation of the DFT, the speed improvement is much higher when the size of x is even in terms of powers of 2 (or the next few small prime numbers, cf. Press et al. (1992) and the FFTW documentation, Frigo and Johnson (2003)). Thus, one should consider enlarging the image size by adding the appropriate number of zeros at the border. However, this can lead to large overhead regions, when the image size is not close to the next power of 2. For this reason a tiling scheme was used, which processes the image in smaller parts of even size, which can cover the entire image more closely. It is important to be aware of the distorted margins of the image or it's tiles, i.e. the pixels at $t \in \mathrm{I}\backslash\mathrm{I}'$, when filtering is done in the frequency domain. Because the cross correlation in the frequency domain is cyclic, points at the margin, for which the neighborhood J exceeds the image boundaries, have incorrect values in the filter's output. This is particularly important for the tiling scheme, which has to provide sufficient overlap for the tiles. For the same reason the tiles should not be too small, to avoid overhead from the overlapping margins. Table 4.1 summarizes the speed-up gain for the described filtering method. Most performance gain is obtained through the filtering in the frequency domain. However, splitting the image into tiles of appropriate size can improve speed even further.

4.3.4 Experiments

To test the performance of the method two images, one of wild type and one of mutant photoreceptor terminals, were used. The profiles of the terminals contain typically about 100 synaptic vesicles, the number of which can be altered by mutating the genes for membrane trafficking in the terminal. Detecting such numerical differences is a simple but tedious task best suited to a computational approach. The wild type images came from electron micrographs of the same animal under the same physiological conditions. For all images visual identification and hand written labelings of the vesicles were made. Image `ter01` (figure 4.3) was used for training. The validation error on `ter04` (figure 4.4, top) was considered for model selection. Then, the best model was tested on the mutant image `ter08` (figure 4.4, bottom).

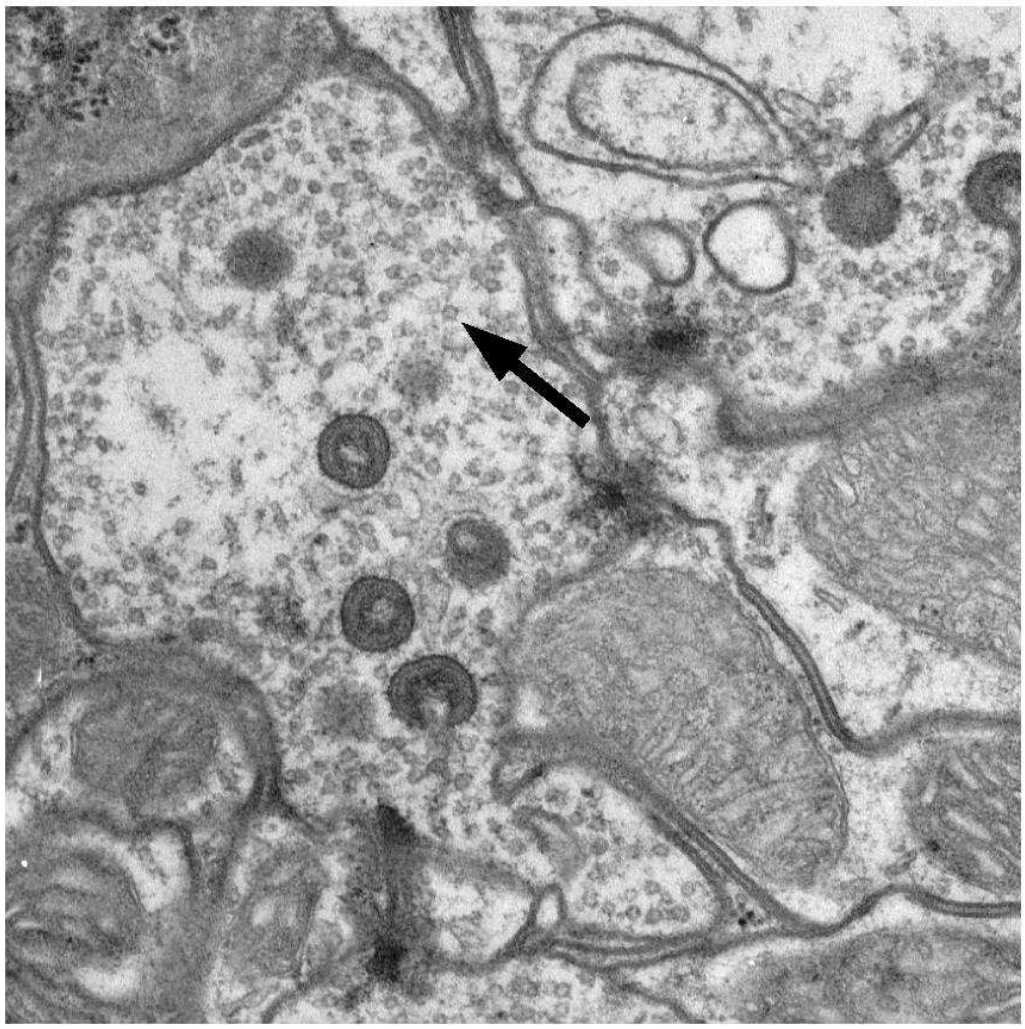


Figure 4.3: EM image of photoreceptor terminals of the wild type fruit fly, *Drosophila melanogaster*. The arrow points to an individual synaptic vesicle. This image (`ter01`) was used for training.

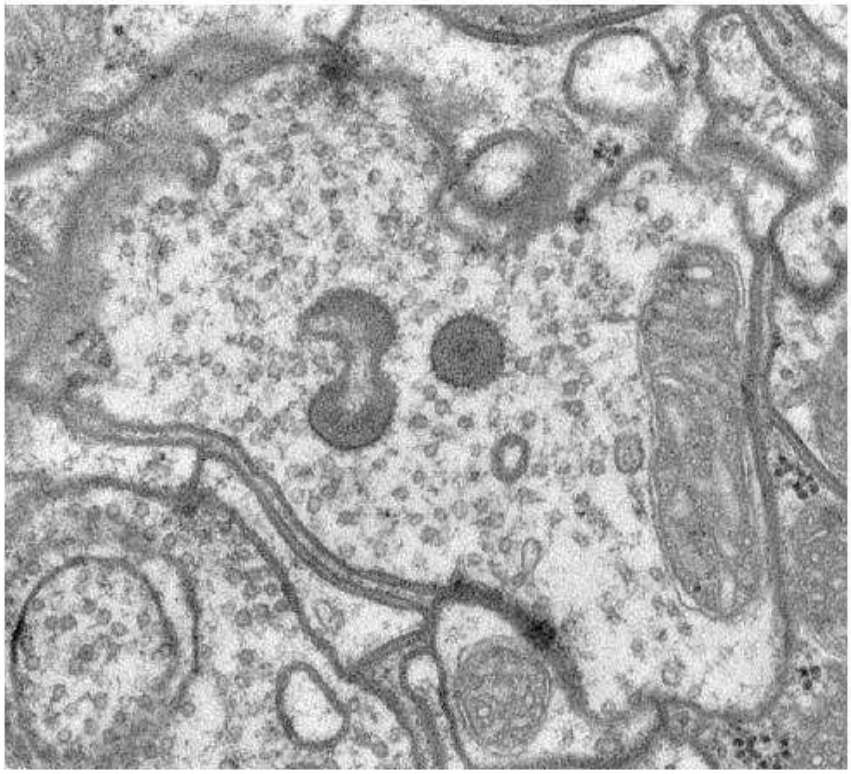


Figure 4.4: EM images of photoreceptor terminals of, top (`ter04`), the wild type and, bottom (`ter08`), genetic mutant fruit fly, *Drosophila melanogaster*. These images were used for model selection and validation.

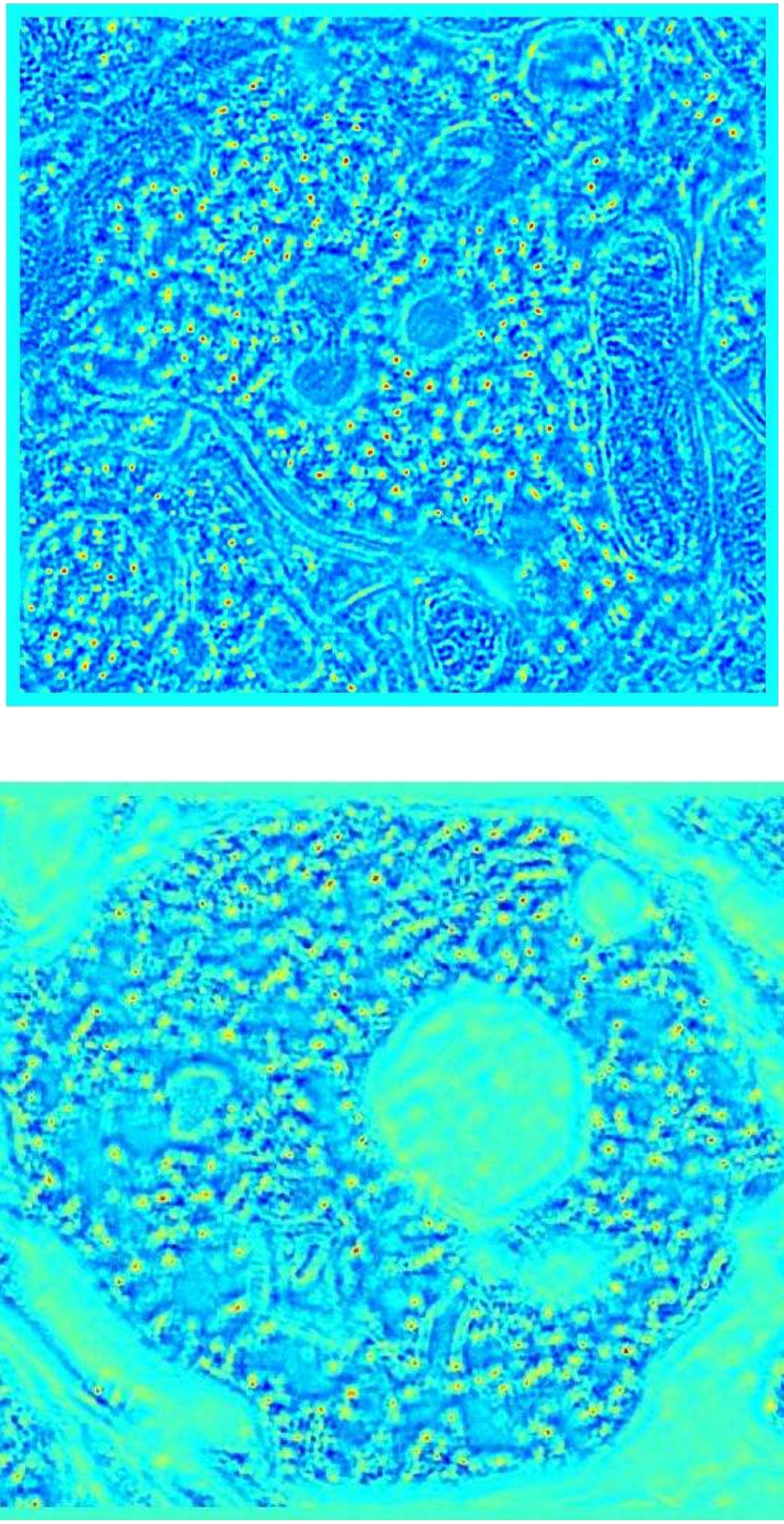


Figure 4.5: Output of the SVM filter applied to the images `ter04` and `ter08` (cf. figure 4.4). Red indicates large values. For parameters of the SVM see text.

4.3.5 Description of the procedure

`ter01` contains 286 hand-labeled vesicles at discrete positions. To generate a smooth target image y, circular Gauss blobs with $\sigma^2 = 40$ and a peak value of 1 were placed on every label. Now, training examples $(x_{t+\tau})_{\tau \in \mathrm{J}}$ where generated from `ter01` by taking square patches, centered around t. The patch size was set to $\mathrm{J} = [1 - \frac{T_f}{2}, \frac{T_f}{2}] \times [1 - \frac{T_f}{2}, \frac{T_f}{2}]$ with $T_f = 50$ pixels to cover an entire vesicle plus a little surrounding. The corresponding values y_t of the target image where used as targets for regression.

The most complete training set would clearly contain patches from all locations, which however would be computationally unfeasible. Instead patches from all hand-label positions and additionally 2000 patches from random positions were used. No patches exceeded the image boundaries. With these data the SVM was trained, using the `libsvm` implementation (Chang and Lin, 2003) which also contains, beside others, the ν-SVR.

Mainly three parameters had to be adjusted for training the ν-SVR: the width of the RBF kernel γ and the parameters ν and c. Because the training dataset is small compared with the input dimensionality, the validation error on `ter04` is subject to large variance. Therefore a model with not too much complexity can be expected to give the best generalization. It turned out that for the given conditions a kernel size of $\gamma = 20,000$ together with a small value $\nu = 0.1$ and $c = 0.01$ yield good validation results on `ter04`. The optimization returned 245 SVs, 185 of which where outliers w.r.t. the ε-sensitive loss. The kernel width was large compared to the average distance of the training examples in input space, which was $< 2,000$.

Because the computation time of the filter grows linearly with the number of SVs, one is strongly interested in a solution with only few of them. This requires small values of ν since it is a lower bound on the fraction of SVs. At the same time small ν values provide large ε and hence restrict the model complexity. The filter output on `ter04` and `ter08` for these parameters is illustrated in figure 4.5. Comparing with the figure 4.4, one can see the sharpest peaks for those vesicles that have the most distinct shape.

After filtering the decision what point in $(z_t)_{t \in \mathrm{I}'}$ corresponds to a vesicle and what not has to be made. Although the regions of high amplitude form sharp peaks, they still have some spatial extension. Therefore the discrimination for the peak locations is done first, followed by an amplitude threshold. Peak locations are those locations t for which z_t is a local maximum in some neighborhood, which is determined roughly by the size of a vesicle, i.e. the peak locations constitute the set

$$\mathrm{Q}_d = \left\{ t : \; z_t = \max_{\{\tau \in \mathrm{I}' : \; |t-\tau| \leqslant d\}} z_\tau \right\} \; . \tag{4.28}$$

A threshold is applied to the candidates in Q_d to yield the set of locations which are considered as detected vesicles,

$$\mathrm{Q}_\theta = \{ t \in \mathrm{Q}_d : \; z_t > \theta \} \; . \tag{4.29}$$

The distance parameter was kept constant, $d = 15$, in all experiments. Only the threshold θ was varied.

4.3.6 Performance Evaluation

To evaluate the performance of the method, the set of detected vesicles Q_θ must be compared with set Q_{Exp}, which contains the locations detected by a human expert. Clearly, this is only meaningful when done on data which was not used to train the SVM. Note that the location of the same vesicle may vary slightly in Q_θ and Q_{Exp} due to fluctuations in the manual labeling, for example. So we need to find the set Q_{match}, containing pairs (t_1, t_2) with $t_1 \in \mathrm{Q}_\theta$, $t_2 \in \mathrm{Q}_{Exp}$ so that t_1 and t_2 are close to each other and describe the location of the same vesicle. This can be computed with a simple, greedy, but fast algorithm:

- compute the matrix $D_{ij} = \|t_i - t_j\|$ for all $t_i \in \mathrm{Q}_\theta$, $t_j \in \mathrm{Q}_{Exp}$

- while $D_{ij} = \min D \leqslant d_m$
  - put (t_i, t_j) into Q_{match}
  - fill i-th row and j-th column of D with $+\infty$

The resulting pairs of matching locations are closer than d_m, which should be set approximately to the radius of a vesicle. This algorithm does not generally find the global optimal solution, which would be a NP-complete problem, but for low point densities the error made by this algorithm is usually low. Now we can evaluate the fraction of correctly detected and the fraction of false positives,

$$f_c = \frac{|\mathrm{Q}_{match}|}{|\mathrm{Q}_{Exp}|}, \qquad f_{fp} = 1 - \frac{|\mathrm{Q}_{match}|}{|\mathrm{Q}_{\theta}|}, \tag{4.30}$$

where $|\mathrm{Q}|$ denotes the cardinality of the set Q. Depending on the threshold θ, $|\mathrm{Q}_\theta|$ may change and so does $|\mathrm{Q}_{match}|$. So we get different values for f_c and f_{fp}. These two rates are summarized in a diagram, which we may call, following Harvey, Jr. (1992), *Receiver Operating Characteristic (ROC)*. In comparison to Harvey, Jr. (1992), f_c represents the *hit rate* and f_{fp} represents the *false alarm rate*, cf. figure 4.6.

However, our ROC differs in some aspects. Since with changing threshold the cardinality of Q_θ changes, f_{fp} does not need to be a monotonic increasing function of θ, and hence our ROC does not need to be monotonic. Furthermore, f_c does not need to reach 1 for arbitrary low thresholds, as it is restricted by the set Q_d, which does not need to contain a match to all elements of Q_{Exp}.

If no a priori costs are assigned to f_c and f_{fp}, then a natural measure for quality is the area below the ROC, which would be close to 1 at best and 0 if no match would be contained in Q_d.

4.3.7 Results

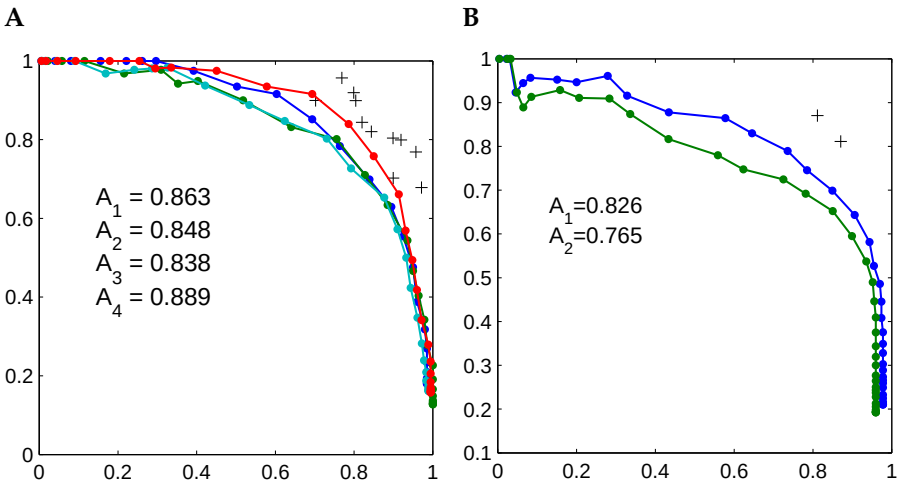


Figure 4.6: **A:** ROC of the validation with `ter04`, and **B:** with `ter08`. For various thresholds θ, f_c is plotted on the x-axis versus $1 - f_{fp}$ on the y-axis. The single crosses show the fraction of matching labels for every pair of hand labels of `ter04`. For detailed explanation, see text.

The ROC of the validation with `ter04` and `ter08` is shown in figure 4.6. The rates f_c and f_{fp} were computed for 50 different threshold values covering the interval $[\min_{t\in\mathrm{Q}_d} z_t, \max_{t\in\mathrm{Q}_d} z_t]$. For `ter04` there exist four and for `ter08` two human expert labelings. Therefore either four or two curves, respectively, can be plotted. To get an impression about the variance of the performance measure, one may consider the areas below the curves. Furthermore the multiple hand labelings allow to plot them against each other in the same figure (single crosses).

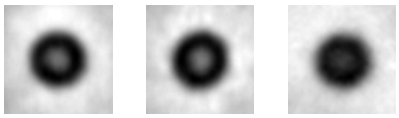


Figure 4.7: Mean vesicles obtained by averaging the hand labeled 50×50 patches. From left to right `ter01`, `ter04`, `ter08`

They indicate what performance is achievable at best. A curve passing these points can be considered to do the task as well on average as a human does. One can see that for the wild type image the curve gets close to that region. For the mutant the performance is slightly worse in terms of the area. This is certainly due to slight differences in the preparation of the electron micrographs and the fact that the image comes from a different animal, which was moreover a genetic mutant. For example, in figure 4.7 one can see that in `ter08` the vesicles look slightly different on average. They are a bit smaller, and the bright spot in the center is less prominent. Further, the background of `ter08` is slightly different in texture and contrast compared to the wild type images. Thus, a slightly worse performance on the mutant image could be expected. In cases where the training data do not cover all types of variation, it is difficult to generalize upon such variations. Thus, for practical applications one would clearly put as many different images as possible in the training set and, at the same time, try to keep the preparation and imaging conditions as constant as possible.

4.4 Sparseness regularization for second order kernel methods

The application presented in section 4.3 made use of Support Vector Regression in conjunction with RBF-kernels to supervisedly learn a nonlinear filter function as a basis function network (4.14) with quadratic basis functions (4.16). There, the number of the support vectors or, in other words, the number of basis functions linearly determines the costs of the filtering operations. Due to the nature of the SVM solution, which involves a quadratic loss function and box constraints, sparse solutions usually are favored. The parametrization as a ν-SVM particularly accounts for the sparseness of the solution.

However, for the large class of second order kernel methods, sparseness of the solutions is usually not provided by itself. Therfore, in this section we will introduce a new optimization procedure that allows to solve these problems and, at the same time, enforce sparseness of the solution.

4.4.1 Second order kernel methods

Kernel methods offer the possibility to turn linear supervised and unsupervised learning methods into nonlinear ones, retaining the particular computational simplicity of the linear method to some degree. This is achieved by the execution of the linear methods in some high dimensional feature space $\mathcal{F}$, which the data $\mathbf{x} \in \mathbb{R}^M$ have been nonlinearly mapped into beforehand. This mapping determines the degree of nonlinearity rsp. the complexity of the nonlinear method. Although in principle this mapping can be defined and carried out explicitly as some function $\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^M \to \mathcal{F}$, this direct mapping can become problematic if the dimension of $\mathcal{F}$ is large (or even infinite), which however is necessary for high complexity of the linear method in feature space.

Fortunately, many linear methods only make use of scalar products in $\mathbb{R}^M$, which in feature space allows to define only the scalar product

$$\mathbf{\Phi}^T(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}_j) := k(\mathbf{x}_i, \mathbf{x}_j) \tag{4.31}$$

and not the mapping itself. The function k is called *kernel* and implicitly defines the mapping $\mathbf{\Phi}$ and $\mathcal{F}$. The space $\mathcal{F}$ is populated in a subspace of dimension smaller or equal T, the

number of data points $\mathbf{x}$. Since only scalar products are involved, it is sufficient to consider only quantities from that subspace, all of which can be represented as linear combinations of the mapped data points. This procedure is often called the 'kernel trick', and the resulting nonlinear methods are called kernel methods.

Second order kernel methods are those, the underlying linear versions of which involve second order statistics of the data $\mathbf{x}$. This leads to learning methods that contain at most second order objective functions and constraints, which together can be solved in closed form as generalized eigenvalue problems (Schölkopf et al., 1998) or singular value decompositions (Bach and Jordan, 2002). The most prominent examples may be PCA and Kernel-PCA (Schölkopf et al., 1998). In a very similar way other linear second order learning methods have found their nonlinear counterparts. Methods of decorrelation based *Second Order Blind Source Separation* (Ziehe and Müller, 1998) have been transformed to non-linear BSS Harmeling et al. (2003). Similarly, kernelized versions of *Canonical Correlation Analysis* approach nonlinear Independent Component Analysis (Fyfe and Lai, 2000; Bach and Jordan, 2002). *Slow Feature Analysis*, which was originally proposed with an explicit nonlinear mapping (Wiskott and Sejnowski, 2002), can be transformed to Kernel-SFA in exactly the same way (Bray and Martinez, 2002). There may be further second order (kernel-) methods.

Due to the nature of the quadratic constraint, second order kernel methods usually don't yield sparse solutions meaning that any projections found in feature space are in principle linear combinations of all mapped data points rather than only a few *support vectors*. However, sparse solutions are highly desirable to keep the computational costs low, when new data shall be projected onto the resulting kernel principal components, kernel slow components etc..

In this section we will give up the quadratic objective function in order to allow for non-quadratic regularization terms which enforce sparse solutions. This has the consequence that the optimization problem cannot be solved in closed form by means of eigen- or singular value decomposition anymore. Therefore a gradient descent optimization procedure is introduced, which is restricted to the admissible hyper-ellipsoidal surface that is given by the quadratic constraint.

The reminder of this section is written in the context of kernel-PCA. However, in the derivations we will not make any assumptions that are specific to Kernel-PCA. Thus, the proposed algorithm is equally well suited to all of the above mentioned second order kernel methods. Generally the proposed *Hyper-Elliptical Conjugate Gradient Descent* method can be applied to optimize arbitrary nonlinear functions under positive definite quadratic constraints.

4.4.2 Sparseness regularization for kernel PCA

Kernel-PCA amounts to finding orthogonal projections in feature space that exhibit the largest variances of the projected data. The variance of the projection $\mathbf{w}_i$ is given by

$$\begin{aligned} d_i &:= \left\langle (\mathbf{w}_i^T \mathbf{\Phi}(\mathbf{x}))^2 \right\rangle - \left\langle \mathbf{w}_i^T \mathbf{\Phi}(\mathbf{x}) \right\rangle^2 \\ &= \mathbf{w}_i^T \left(\left\langle \mathbf{\Phi}(\mathbf{x}) \mathbf{\Phi}^T(\mathbf{x}) \right\rangle - \left\langle \mathbf{\Phi}(\mathbf{x}) \right\rangle \left\langle \mathbf{\Phi}^T(\mathbf{x}) \right\rangle \right) \mathbf{w}_i \,, \end{aligned}$$

where $\left\langle \cdot \right\rangle$ denotes the expectation over $\mathbf{x}$.

Finding a maximum clearly is only meaningful if the norm of the projection is constraint to a fixed value, say $\mathbf{w}_i^T \mathbf{w}_i = 1$. Different projections shall be orthogonal, thus $\mathbf{w}_i^T \mathbf{w}_j = 0$ for $i \neq j$. In a sequential approach finding the component $\mathbf{w}_i$, where other components $\mathbf{w}_j$ already have been found, amounts to the following constraint optimization problem:

$$\text{maximize} \quad d_i \qquad \text{s.t.} \quad \mathbf{w}_i^T \mathbf{w}_j = \delta_{ij} \,. \tag{4.32}$$

If the mapping $\mathbf{x} \to \mathbf{\Phi}(\mathbf{x})$ is given as such, the problem can be solved using ordinary PCA, i.e. by eigen-decomposition of the covariance matrix of $\mathbf{\Phi}(\mathbf{x})$.

However, a direct mapping into the feature space gives rise to difficulties if the dimension of $\mathbf{\Phi}(\mathbf{x})$ is too high for the necessary computations to be tractable. For problems that can be

expressed exclusively in terms of scalar products in feature space, these difficulties can be overcome with the so called 'kernel trick'. It constitutes the definition of the scalar product as a kernel function (4.31). At the same time the projection $\mathbf{w}_i$ is approximated by a linear combination of a finite number of data vectors $\hat{\mathbf{x}}_l$ mapped into the feature space,

$$\mathbf{w}_i = \sum_{l=1}^{N} \alpha_{il} \mathbf{\Phi}(\hat{\mathbf{x}}_l) \ .$$

The set $\{\hat{\mathbf{x}}_l, l = 1 \ldots N\}$ is called the initial[2] set of support vectors of $\mathbf{w}_i$. In terms of kernels a projection of a data point $\mathbf{x}$ onto $\mathbf{w}_i$ in feature space is given by

$$\mathbf{w}_i^T \mathbf{x} = \sum_{l=1}^{N} \alpha_{il} k(\hat{\mathbf{x}}_l, \mathbf{x}) \ . \tag{4.33}$$

Hence, the variance of the projection is given by

$$d_i \quad = \quad \sum_{l=1}^{N} \sum_{h=1}^{N} \alpha_{il} \alpha_{ih} \left(\left\langle k(\hat{\mathbf{x}}_l, \mathbf{x}) k(\hat{\mathbf{x}}_h, \mathbf{x}) \right\rangle - \left\langle k(\hat{\mathbf{x}}_l, \mathbf{x}) \right\rangle \left\langle k(\hat{\mathbf{x}}_h, \mathbf{x}) \right\rangle \right) \ , \tag{4.34}$$

where the constraints are given by

$$\sum_{l=1}^{N} \sum_{h=1}^{N} \alpha_{il} \alpha_{jh} k(\hat{\mathbf{x}}_l, \hat{\mathbf{x}}_h) = \boldsymbol{\alpha}_i^T \hat{\mathbf{K}} \boldsymbol{\alpha}_j = \delta_{ij} \ . \tag{4.35}$$

Maximizing (4.34) w.r.t. $\boldsymbol{\alpha}_i := (\alpha_{i1}, \ldots, \alpha_{iN})^T$ subject to the constraint (4.35) is a generalized eigenvalue problem, the dimensionality of which is determined by N rather than the dimensionality of $\mathcal{F}$.

Thus N should be at least small enough so that the eigen-decomposition can be actually computed. In practice, however, one is usually interested in much smaller N in order to achieve cheap computations of the projection (4.33). But then, N should be still large enough so that the span of the initial support vector set in $\mathcal{F}$ is a good approximation of the span of all data points in $\mathcal{F}$, and projections $\mathbf{w}_i$ can be composed with sufficient accuracy. This trade-off does not just regard their number, but, even more important, the choice of the initial support vectors.

Consider a particular problem where the expectations in (4.34) are estimates over T given training data vectors $\mathbf{x}_t$, $t = 1 \ldots T$. Then, there are $\binom{T}{N}$ possibilities to choose the support vectors from the training data[3]. So it may be impossible to find the optimal support vector set of given (small) size N.

This gives rise to the idea to start with a large set, that is tractable for computation, and is likely to contain a smaller subset that yields almost equally good projections. In the course of the adjustment of the α_{il}, we must assure that the majority of them eventually becomes zero leaving only those that correspond to a small and almost optimal set of support vectors. That means we must modify the constrained optimization problem (4.32) such that sparse solutions a favored.

Here it be proposed to achieve sparseness by adding a regularization term to the objective function that penalizes non-sparse solutions:

$$\text{max.} \quad d_i + \beta \left(\frac{\|\boldsymbol{\alpha}_i\|_1}{\|\boldsymbol{\alpha}_i\|_2} \right)^2 \qquad \text{s.t.} \quad \boldsymbol{\alpha}_i^T \hat{\mathbf{K}} \boldsymbol{\alpha}_j = \delta_{ij} \ . \tag{4.36}$$

The regularization term is the ratio of the L_1- and the L_2-norm of the vector $\boldsymbol{\alpha}_i$. It is always larger or equal to one with equality if, and only if, not more then one element of $\boldsymbol{\alpha}_i$ is

[2] It is called initial because in the result of a sparse optimization we can expect that many of the coefficients α_{il} become zero, and the remaining ones constitute the final set of support vectors.

[3] Note that, in contrast to Support Vector Machines, here the set of support vectors does not necessarily have to be a subset of the training data set.

different from zero. We apply the square of that ratio because also d_i is a quadratic function of $\boldsymbol{\alpha}_i$. Thus, the choice of a suitable regularization parameter β is independent of scalings of $\boldsymbol{\alpha}_i$.

In the application for Kernel-PCA one can achieve good results with the above regularization term, but other ones are applicable as well. For example, one could use the sample kurtosis of the elements of $\boldsymbol{\alpha}_i$ for regularization. Generally, in order to make the whole algorithm insensitive for small changes of β, it is usually a good idea to use a regularization term that scales with $\boldsymbol{\alpha}_i$ equally to d_i.

4.4.3 Hyper-ellipsoidal conjugate gradient

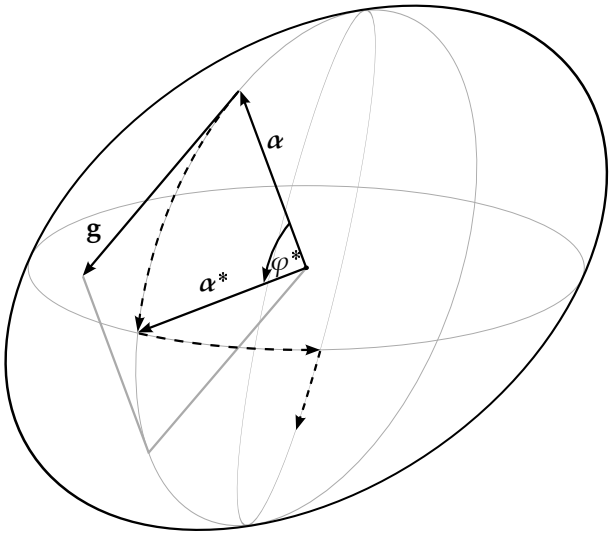


Figure 4.8: Gradient descent on a hyper-ellipsoidal surface. The vector to the current point, $\boldsymbol{\alpha}$, and the gradient projected into the tangent space at the current point, $\mathbf{g}$, define the search direction. Line search is done on the elliptical trajectory defined by these two vectors in terms of the angle φ, where the current point and the gradient vector are $\pi/2$ apart. At the minimum on the trajectory a new search direction is computed. The procedure is repeated until convergence.

The sparseness regularization in (4.36) comes at the expense that the structure of a generalized eigenvalue problem is lost. Thus, a solution in closed form does in general not exist. Therefore, in the following we derive a new gradient optimization procedure that takes the specific nature of the quadratic constraint into account: the *Hyper-Ellipsoidal Conjugate Gradient*.

We derive the algorithm for a general cost function $f(\boldsymbol{\alpha}_i)$, that may include any regularization terms and that shall be minimized with respect to the quadratic constraints

$$\boldsymbol{\alpha}_i^T \mathbf{C} \boldsymbol{\alpha}_j = \delta_{ij} \,, \tag{4.37}$$

where $\mathbf{C}$ is a positive definite matrix that establishes the constraints. Thus, with respect to $\mathbf{C}$, $\boldsymbol{\alpha}_i$ shall have unit length and shall be orthogonal to any previously found $\boldsymbol{\alpha}_j$. To be precise, (4.37) constitutes exactly one quadratic constraint for $i = j$ and none or a number of linear constraints for $i \neq j$ (w.o.l.o.g. $i > j$).

To start the gradient descent, we need an admissible starting point that fulfills the constraints. We start with a random $\boldsymbol{\alpha}_i$ and first make it orthogonal w.r.t. $\mathbf{C}$ to all previously found $\boldsymbol{\alpha}_{j<i}$

$$\boldsymbol{\alpha}_i \quad \leftarrow \quad \boldsymbol{\alpha}_i - \mathbf{M}\mathbf{M}^\dagger \boldsymbol{\alpha}_i, \quad \mathbf{M} = \mathbf{C} \cdot (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{i-1}) \,, \tag{4.38}$$

where $\mathbf{M}^\dagger$ denotes the pseudo inverse of $\mathbf{M}$. Then, the length of $\boldsymbol{\alpha}_i$ is scaled to match $\boldsymbol{\alpha}_i^T \mathbf{C} \boldsymbol{\alpha}_i = 1$,

$$\boldsymbol{\alpha}_i \quad \leftarrow \quad (\boldsymbol{\alpha}_i^T \mathbf{C} \boldsymbol{\alpha}_i)^{-\frac{1}{2}} \boldsymbol{\alpha}_i \,. \tag{4.39}$$

We now have an admissible starting point $\boldsymbol{\alpha}_i$ and compute the negative gradient

$$\mathbf{g} = -\frac{\partial f(\boldsymbol{\alpha}_i)}{\partial \boldsymbol{\alpha}_i} \tag{4.40}$$

at the starting point. The gradient needs to be projected into the tangent space orthogonal to $\boldsymbol{\alpha}_i$ and to all previously found $\boldsymbol{\alpha}_j$,

$$\mathbf{g} \quad \leftarrow \quad \mathbf{g} - \mathbf{M}\mathbf{M}^\dagger \mathbf{g}, \quad \mathbf{M} = \mathbf{C} \cdot (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_i) \,, \tag{4.41}$$

and it's length needs to be normalized,

$$\mathbf{g} \quad \leftarrow \quad (\mathbf{g}^T \mathbf{C} \mathbf{g})^{-\frac{1}{2}} \mathbf{g} \,. \tag{4.42}$$

Now, we seek a local minimum on the elliptical trajectory that is defined by $\boldsymbol{\alpha}_i$ and $\mathbf{g}$ (cf. Figure 4.8). The local minimum is given by

$$\boldsymbol{\alpha}_i \quad \leftarrow \quad \cos(\varphi^*) \boldsymbol{\alpha}_i + \sin(\varphi^*) \mathbf{g} \,. \tag{4.43}$$

Clearly, because $\boldsymbol{\alpha}_i$ and $\mathbf{g}$ are of unit length, are mutually orthogonal, and are orthogonal w.r.t. $\mathbf{C}$ to all $\boldsymbol{\alpha}_{j<i}$, also the new $\boldsymbol{\alpha}_i$ holds the constraints. The angle φ^* is found with *Golden Section Search* (cf. Press et al. (1992, §10.1)) on the interval $[0, 2\pi)$. If the cost function is symmetric, $f(\boldsymbol{\alpha}_i) = f(-\boldsymbol{\alpha}_i)$, then the search interval can be reduced to $[0, \pi)$. The new $\boldsymbol{\alpha}_i$ is used to continue the iterations at (4.38)[4].

These iterations are continued until a suitable combination of the following stopping criteria is fulfilled:

(i) After the Golden Section Search $\varphi*$ falls below some preset threshold ε_φ.

(ii) The value of the cost function before and after (4.43) decreases by an amount smaller than ε_f.

(iii) The norm of the gradient before scaling, $(\mathbf{g}^T \mathbf{C} \mathbf{g})^{\frac{1}{2}}$, falls below some threshold ε_g.

(iv) A preset maximum number of iterations has been exceeded.

ε_g can be seen as a protection against numerical instabilities. It should be set to the smallest value that still allows to perform step (4.42) with sufficient accuracy. Thus, criterion (iii) should terminate the iterations in any case. For (i) and (ii), depending on the particular problem, the user may choose whether both criteria or either one terminates the optimization procedure.

Once the iterations for $\boldsymbol{\alpha}_i$ are done, one may proceed to iterate for $\boldsymbol{\alpha}_{i+1}$ starting at (4.38). As many components $\boldsymbol{\alpha}_i$ as the rank of $\mathbf{C}$ can be found at most. More than that cannot be pairwise orthogonal, (4.37). Since in the present context $\mathbf{C}$ is the result of the embedding into a high dimensional feature space, it is usually of much higher rank than the number of desired components.

Conjugate gradient

In many situations the performance of the above described optimization procedure can be considerably improved by the application of *Conjugate Gradient Descent* (cf. Press et al. (1992, §10.6)). There, the line search is along the direction of a modified gradient $\mathbf{h}$, which takes the former search directions into account. It can be assumed that this works equally

[4] Actually one could begin the iteration loop with the computation of a new gradient, (4.40). However, in order to avoid accumulated errors one should perform the orthogonalization operation (4.38) in every iteration.

well in a local neighborhood of a quadratic hyper-surface as in the Euclidean space, as long as the neighborhood is small enough to be well approximated by it's tangent space.

It turns out that the Fletcher-Reeves update formula is also applicable in the hyper-elliptical space. In every iteration step the new search direction is given by

$$\mathbf{h}_{new} = \mathbf{g}_{new} + \gamma \mathbf{h}^*_{old} \qquad \text{with} \quad \gamma = \frac{\mathbf{g}^T_{new}\mathbf{C}\,\mathbf{g}_{new}}{\mathbf{g}^T_{old}\mathbf{C}\,\mathbf{g}_{old}}\,. \tag{4.44}$$

$\mathbf{h}_{new}$ and $\mathbf{g}_{new}$ are the search direction and the gradient at the current position, respectively. The necessary scaling (4.42) prior to the line search procedure (4.43) is performed on the actual search direction $\mathbf{h}_{new}$. The gradient $\mathbf{g}_{new}$ is only projected according to (4.41) to hold the orthogonality conditions, but not length normalized. $\mathbf{h}_{old}$ and $\mathbf{g}_{old}$ are the corresponding quantities at the starting point of the last line search. Note that $\mathbf{g}_{new}$ and $\mathbf{h}_{old}$ come from different tangent spaces so that, in contrast to the original Fletcher-Reeves formula, they cannot be simply added in (4.44). However, a transport of $\mathbf{h}_{old}$ from one space to the other is well defined by

$$\mathbf{h}^*_{old} = \cos(\varphi^*)\mathbf{h}_{old} - \sin(\varphi^*)\boldsymbol{\alpha}_{i,old}$$

meaning that $\mathbf{h}^*_{old}$ is the direction at $\boldsymbol{\alpha}_{i,new}$ that we came from, when we left $\boldsymbol{\alpha}_{i,old}$ in direction $\mathbf{h}_{old}$. In this respect (4.44) is equivalent to Fletcher-Reeves conjugate gradient in Euclidean spaces. Note that such transformation is not possible for the often better performing Pollack-Ribiere update formula, which involves the computation of the value

$$\gamma = \frac{(\mathbf{g}_{new} - \mathbf{g}_{old})^T\mathbf{C}\,\mathbf{g}_{new}}{\mathbf{g}^T_{old}\mathbf{C}\,\mathbf{g}_{old}}$$

because $\mathbf{g}_{old}$ and $\mathbf{g}_{new}$ reside in different tangent spaces and there is no well defined transformation from one space to the other.

The Conjugate Gradient update formula (4.44) does not violate the constraints: if $\mathbf{h}_{old}$ was orthogonal to all $\boldsymbol{\alpha}_{j<i}$ and $\boldsymbol{\alpha}_{i,old}$, then $\mathbf{h}^*_{old}$ and, hence, $\mathbf{h}_{new}$ are orthogonal to all $\boldsymbol{\alpha}_{j<i}$ and $\boldsymbol{\alpha}_{i,new}$. Thus, the projection (4.41) only needs to be performed on $\mathbf{g}_{new}$, not on $\mathbf{h}_{new}$. However, $\mathbf{h}_{new}$ must be length normalized before the line search step.

Reduction

If the cost function involves a regularization term that gives priority to sparse solutions, the optimization will lead to solutions $\boldsymbol{\alpha}_i$ with many elements close to zero. Thus, in order to save computation costs, the idea to discard these elements already during optimization suggests itself. Computation costs can be saved in the objective function and the regularization term, as well as in the constraint. In the constraint (4.37) and in the computation of the scalings (4.39) and (4.42), for every discarded element of $\boldsymbol{\alpha}_i$ the corresponding rows and columns of $\mathbf{C}$ can be discarded. Similarly for the computation of $\mathbf{M}$ in (4.38) and (4.41), except that here the discarded rows of $\mathbf{C}$ are determined by $\boldsymbol{\alpha}_i$, and the discarded columns are those that correspond to elements that are discarded in all $\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_{i-1}$ rsp. $\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_i$.

A quadratic cost function is reduced in the same way – by discarding rows and columns of the matrix in the second order term and by discarding the corresponding elements of the linear term (if there is any). The reduction of the costs for the computation of the L_1- and L_2-norms in the regularization term (4.36) is obvious.

It remains to define when and which elements of $\boldsymbol{\alpha}_i$ are discarded. A good place to link reduction into the iteration loop is right before (4.38) because after reduction the constraints may become slightly violated. For the experiments shown below, the most simple heuristic was used: discard elements that are closer to zero than some positive value ε_α. Problems with this heuristic could arise for elements that are slowly changing their sign and get caught when they cross zero. However, in various experiments these effects showed not to be severe, and always good results could be achieved. It appears to be the case that due to the line minimization a slow zero crossing hardly happens. Interestingly, even without any regularization term, sparse solutions can become favored by reduction alone, in particular when ε_α is comparably large.

4.4.4 Experiments

This is a little example that is meant to qualitatively demonstrate how sparse optimization for Kernel PCA works. Two-dimensional input data $\mathbf{x}$ where generated by a mixture model with three Gaussian components,

$$\mu_1 = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix},$$
$$\mu_2 = \begin{pmatrix} -3 \\ 2 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\mu_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Each component generated 150 data points shown as small dots in the panels of figures 4.9 and 4.11. Figure 4.9 shows the six first kernel principal components which result from the solution of the generalized eigenvalue problem

$$\frac{1}{T}\hat{\mathbf{K}}\hat{\mathbf{K}}^T\boldsymbol{\alpha}_i = d_i\hat{\mathbf{K}}\boldsymbol{\alpha}_i\,, \tag{4.45}$$

where $\hat{\mathbf{K}}$ is the Gram matrix of the data for a spherical Gaussian kernel with unit width, $\hat{K}_{ij} = \exp\left(-\frac{\|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2}{2}\right)$. For simplicity there was searched for projections with large second moments rather than large variance. Otherwise the columns of $\hat{\mathbf{K}}$ would have had to be normalized by it's column mean on the left side of (4.45).

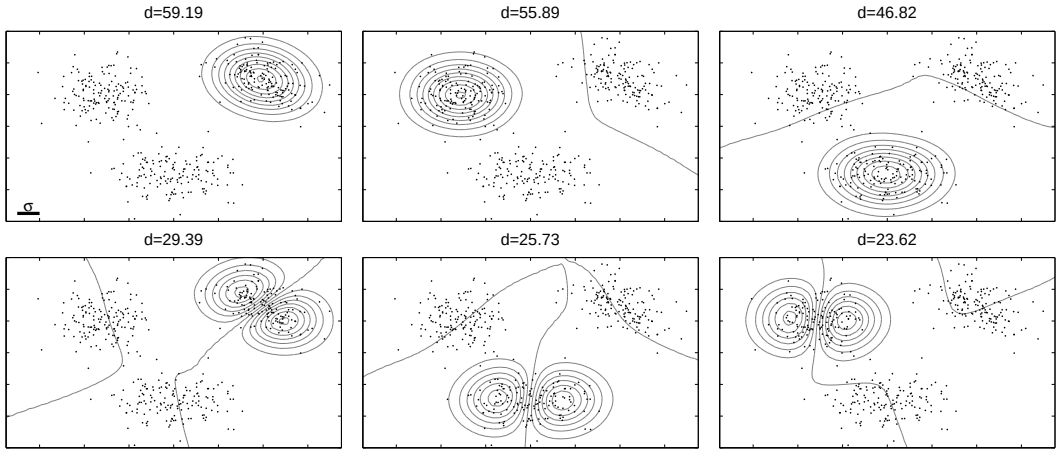


Figure 4.9: The first six principal components achieved through non-sparse Kernel PCA using a Gaussian kernel with width σ. The two-dimensional input data (small dots) are distributed over 3 Gaussian clusters, containing 150 data points each. All of them are support vectors. Gray lines are iso lines of the projection function (4.33). Values d are the empirical second moments of the projections.

In this small example all data points were used as support vectors, thus $N = T = 450$. The projections $\mathbf{w}_i$ of the largest second moments are linear combinations of nearly all $\hat{\mathbf{x}}_i$. Figure 4.10 shows the absolute values of the coefficients α_{il}, sorted in ascending order individually for all six principal components. One can see that most of them assume small or intermediate values. Only few of them get really close to zero. This is not further surprising, and is shown here just to illustrate that, in contrast to e.g. Support Vector Machines, sparse solutions usually do not occur by itself in second order kernel methods.

Next, sparse Kernel PCA was performed on the same data. The constrained optimization problem (4.36) was solved sequentially for the first six components $\boldsymbol{\alpha}_i$, $i = 1 \ldots 6$. The

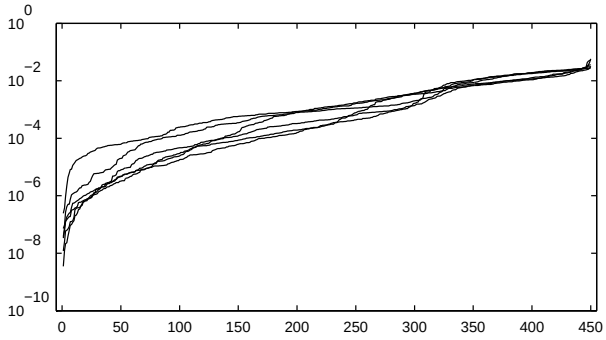


Figure 4.10: Absolute values of the elements of $\boldsymbol{\alpha}_i$ sorted in ascending order for the kernel principal components shown in Figure 4.9. Every line belongs to a different $\boldsymbol{\alpha}_i$, $i = 1 \ldots 6$. Because sparseness was not enforced, the majority of elements assumes intermediate values in the range $|\alpha_{il}| \approx 10^{-6} \ldots 10^{-2}$.

regularization term was weighted by $\beta = 1.2$. As well as in the previous experiment, second moments,

$$d_i = \frac{1}{T} \boldsymbol{\alpha}_i^T \hat{\mathbf{K}} \hat{\mathbf{K}}^T \boldsymbol{\alpha}_i \,,$$

were maximized starting with all data points as initial support vectors. Figure 4.11 shows the resulting principal component projections. They are nearly identical to those shown in Figure 4.9, but are linear combinations of not more than 4–7 support vectors. The achieved second moments are slightly below the values of the unregularized Kernel PCA solution.

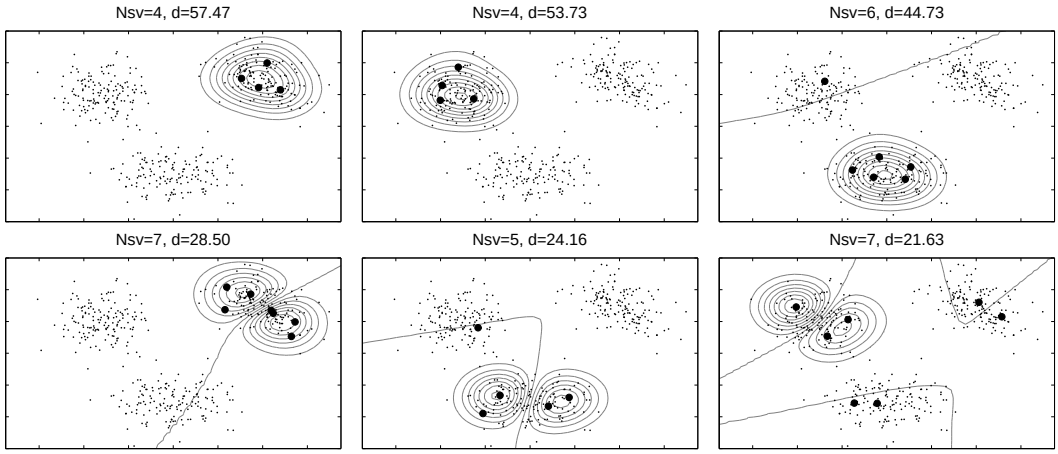


Figure 4.11: The same setup as of Figure 4.9, results for Kernel PCA with sparseness regularization (4.36). With just 4–7 support vectors (large black dots), projections very similar to those of unregularized Kernel PCA are achieved. The empirical second moments of the projections are only little smaller.

In order to quantitatively investigate how the enforced sparseness trades against accuracy experiments on the MNIST dataset of hand written digits (LeCun and Cortes) were done. This dataset represents a nice 'real data problem', that had been subject to kernel based learning methods in numerous publications. However, the focus of attention was not the performance of any particular clustering or classification task. To stay in the context of this section, the objective for the experiments was the somewhat 'abstract' performance measure 'achievable output variance' and how this behaves with the sparseness of the solution.

The MNIST dataset consists of a training dataset of $T = 60,000$ examples of hand written

digits '0'...'9', each of which is 28×28 pixels in size assuming values from 0 (background) to 255 (foreground). Further, there is a test set with 10,000 examples of the same size coming from different writers. The data were used without any preprocessing or normalization.

The new method was compared with the naive approach of initially selecting sufficiently small subsets and the approach proposed by Tipping (2001). Methods that deviate from the quadratic constraint (see e.g. Smola et al. (1999)) were disregarded because comparisons in terms of variance or second moments are not really meaningful with different constraints.

The approach of selecting really small initial support vector sets randomly from the data would yield good performance if just the number of support vectors would matter. If, however, the proper choice of the support vector set plays a more important role, one would expect this method to show large variations in performance reflecting the luck in choosing the initial support vector set.

The Sparse Kernel PCA approach of Tipping is based on an approximation of the covariance matrix of the data projected into feature space,

$$\mathbf{C} = \sigma^2 \mathbf{I} + \sum_{t=1}^{T} w_t \mathbf{\Phi}(\mathbf{x}_t) \mathbf{\Phi}^T(\mathbf{x}_t) \; . \tag{4.46}$$

It is assumed that the vectors in feature space are subject to a $\mathcal{N}(0; \mathbf{C})$ normal distribution. A sparse solution with coefficients w_t is found by maximizing the likelihood of this model w.r.t. w_t and fixed positive σ^2. An expectation-maximization learning rule for w_t can be derived, which can be expressed exclusively in terms of scalar products in feature space, hence, in terms of kernels. Here, just the update rule in one of the two forms presented in Tipping (2001) be shown (for a detailed derivation see there):

$$w_t \leftarrow \frac{\sum_{t=1}^{T} \mu_{ti}^2}{T(1 - \Sigma_{tt}/w_t)} \; , \tag{4.47}$$

where

$$\mathbf{\Sigma} = (\mathbf{W}^{-1} + \sigma^{-2} \hat{\mathbf{K}})^{-1}$$

and

$$\mu_{ti} = \sigma^{-2} \sum_{j=1}^{T} \Sigma_{tj} k(\mathbf{x}_t, \mathbf{x}_j) \; .$$

With the coefficient w_t summarized in the diagonal $T \times T$ matrix $\mathbf{W}$, the sparse kernel principal components $\boldsymbol{\alpha}_i$ are found by the solution of

$$\mathbf{W}^{\frac{1}{2}} \hat{\mathbf{K}} \mathbf{W}^{\frac{1}{2}} \boldsymbol{\alpha}_i' = \lambda_i' \boldsymbol{\alpha}_i' \tag{4.48}$$

and

$$\boldsymbol{\alpha}_i = \frac{1}{\sqrt{\lambda_i'}} \mathbf{W}^{\frac{1}{2}} \boldsymbol{\alpha}_i' \; .$$

The results of the experiments with the MNIST dataset are shown in Figure 4.12 . All performance measures where computed on the test set, while the kernel principal components where estimated on the training set.

Starting point was a subset $\{\hat{\mathbf{x}}_l, \, l = 1 \ldots N\}$ of $N = 500$ elements of the training set, which was used as the initial support vector set. The 500 elements were randomly chosen and remained the same in all trials. The objective was to maximize the output second moments of 10 nonlinear projections (4.33) under the constraint that these projections have unit length in feature space and are pairwise orthogonal, (4.35).

The kernel was a spherical Gaussian with $\sigma = 700$. The quantity to compare is the summed second moments of the outputs computed on the test set with $T = 10,000$ examples,

$$E_{test} = \frac{1}{T} \sum_{i=1}^{10} \boldsymbol{\alpha}_i \mathbf{K} \mathbf{K}^T \boldsymbol{\alpha}_i^T \; , \tag{4.49}$$

where **K** is the kernel matrix between all support vectors and all data points,

$$K_{lt} := k(\hat{\mathbf{x}}_l, \mathbf{x}_t)\ ,\ l = 1 \ldots N,\ t = 1 \ldots T\ .$$

Figure 4.10.A shows this value in dependence of the number of support vectors for the three considered approaches to Sparse Kernel PCA. Black dots mark the results of sparseness regularization (4.36) solved with Hyper-Elliptical Conjugate Gradient descent. The unregularized cost function was

$$d_i = \frac{1}{T}\boldsymbol{\alpha}_i \mathbf{K}\mathbf{K}^T \boldsymbol{\alpha}_i^T \tag{4.50}$$

with **K** computed on the training set. In every trial the regularization term was weighted with random β which was uniformly distributed (on the log-scale) in the range from 10^{-10} to 10^{-3}. The different regularization strength lead to different numbers of support vectors, which are the x-axes of the plot. Because, in this approach each of the 10 components can have different (and differently many) support vectors, their average number provides the x-coordinate. The y-axes show the corresponding value of E_{test}, (4.49).

Plus signs indicate the results for random subsets of 20 to 500 elements of the initial N support vectors, which were used to compute Kernel-PCA without enforcing sparseness any further. The average performance is considerably worse compared to the other two methods in particular for small subsets. At the same time it is subject to large variations. Thus, the proper choice of the support vectors appears to be crucial.

Gray dots indicate the results of Sparse Kernel PCA according to Tipping (2001). Here, the value of σ^2 in (4.46) controls the sparseness of the solution. Random σ in the range $0.03 \ldots 0.06$ were used. One can see that already with all 500 support vectors there is a degradation of performance. This is due to the fact that this algorithm actually has to start with all T data points as initial support vectors because the eigen-system is defined only by the initial support vectors, (matrix $\hat{\mathbf{K}}$ in (4.48)). One can see this also from (4.46) when **C** is approximated by all initial support vectors, but should be by all data points. However, the maximum number of data points is limited to small values because of the matrix inversion of order up to T in the update rule (4.47). In contrast, in the other two approaches with the objective defined as (4.50), the matrix product $\frac{1}{T}\mathbf{K}\mathbf{K}^T$ contains the summation over all data points defining the eigensystem regardless of the number of initial support vectors (rows of **K**).

Figures 4.12.B and 4.12.C show the dependence between β rsp. σ^2 (y-axis) and the resulting number of support vectors (x-axis). The plotted dots correspond to those of panel A. They show a quite deterministic relation, even though in the conjugate gradient approach all $\boldsymbol{\alpha}_i$ were initialized randomly in every trial. Interestingly, for values $\sigma^2 < 2{\cdot}10^{-3}$ no reduction of the support vector set happens, where for values above a 'phase transition' occurs. This may make it difficult to adjust σ^2 to values that yield moderate sparseness.

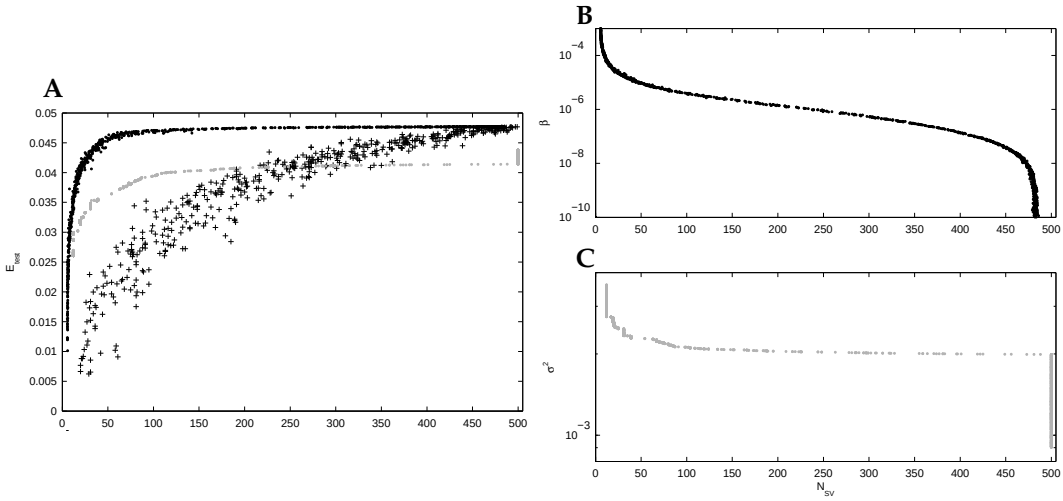


Figure 4.12: **A**: Comparison of the performance measure (4.49) in dependence of the reduced number of support vectors, *black dots:* for sparseness regularization (4.36), *gray dots:* for the maximum likelihood approach (4.46), and *plus signs:* for the selection of random subsets. **B** and **C**: relation between the parameters β rsp. σ^2 and the resulting sparseness of the kernel principal components. For explanation see text.

Chapter 5

Slow Feature Analysis

5.1 Projection pursuit with SFA

There is evidence that spatial and temporal sparseness is one of the working principles of the early stages in the visual system of higher vertebrates. Measurements have shown that cells in area V1 of the visual cortex of primates do a linear processing of a local receptive field of the visual input (Kandel et al., 1991). In simulations with natural image data it was found that these linear filters maximize the sparseness of its outputs (Olshausen and Field, 1996). Very similar results were achieved with ICA (Bell and Sejnowski, 1997; Hyvarinen et al., 2001). The sparseness principle was also the basis for the experiments with multi-channel blind deconvolution on video data (see section 3.4.5).

However, when the sparseness principle shall be adopted to higher levels of visual processing, in technical applications problems arise. Usually maximum likelihood estimators, in particular those involving higher order statistics are not robust to outlying data. However, with increasing complexity of the architectures, in particular for highly non-linear mappings of the inputs, the achievable sparseness must logically increase more and more. Hence, any empirical estimates eventually become fully dominated by outliers. This, can be helped – at least to some degree – with robust estimates (Moors, 1988; Baloch et al., 2005), and of course, with much more training data, which, however, may give rise to other problems.

In parallel to the sparseness principle, temporal coherence was proposed as working principle in the early visual system (Stone, 1996; Wiskott and Sejnowski, 2002). It constitutes that, when processing temporal inputs, the output representations vary as little as possible over time. It could be shown that this leads to the emergence of the same simple and complex cell structures from natural image inputs, as with the sparse coding approach (Hurri and Hyvärinen, 2003). However, because it is based only on second order statistics, the temporal coherence principle allows for much more robust estimators. It can be formalized as follows (Wiskott and Sejnowski, 2002):

> *Given an N-dimensional input signal* $(\mathbf{x}_t)_{t\in\mathrm{I}}$ *find an instantaneous function* $\mathbf{g}$ *generating the M-dimensional output signal* $(\mathbf{y}_t)_{t\in\mathrm{I}}$ *with* $y_{j,t} = g_j(\mathbf{x}_t)$ *such that for each* $j = \{1,\ldots,M\}$
>
> $$\Delta_j := \left\langle \dot{y}_{j,t}^2 \right\rangle \quad \textit{is minimal} \tag{5.1}$$
>
> *under the constraints*
>
> $$\left\langle y_{j,t} \right\rangle = 0 \qquad \textit{(zero mean)}\,, \tag{5.2}$$
>
> $$\left\langle y_{j,t}^2 \right\rangle = 1 \qquad \textit{(unit variance)}\,, \tag{5.3}$$
>
> $$\forall j' < j:\ \left\langle y_{j',t}\, y_{j,t} \right\rangle = 0 \qquad \textit{(decorrelation)}\,. \tag{5.4}$$

This unsupervised learning problem is known as *Slow Feature Analysis (SFA)*. It comprises the search for projections g_j so that the outputs are temporally as smooth as possible with

respect to the average squared local slope $\dot{y}_{j,t}$, which has to be defined reasonably. In the case of time discrete signals with a one-dimensional index set $\mathrm{I} = \mathbb{Z}$ one usually uses the discrete derivative

$$\dot{y}_{j,t} := y_{j,t+1} - y_{j,t} \,.$$

The constraints provide non-trivial solutions. With the unit variance constraint constant outputs are avoided, and the decorrelation constraint assures that in fact different output signals emerge.

5.1.1 Linear SFA

The SFA learning problem can be formalized in matrix notation as follows:

$$\text{Minimize} \quad \mathrm{tr}\left\langle \dot{\mathbf{y}}_t \dot{\mathbf{y}}_t^T \right\rangle \qquad \text{subject to} \quad \left\langle \mathbf{y}_t \mathbf{y}_t^T - \left\langle \mathbf{y}_t \right\rangle \left\langle \mathbf{y}_t^T \right\rangle \right\rangle = \mathbf{I} \,. \tag{5.5}$$

To see how it can be solved, consider at first the linear case, where $\mathbf{g}$ is an affine transformation

$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t) = \mathbf{W}\mathbf{x}_t + \mathbf{b} \,, \tag{5.6}$$

where $\mathbf{W}$ is a $M \times N$ matrix. The offset vector $\mathbf{b}$ is consumed by the zero mean constraint, and, hence, can always be neglected. The linear projection leads to a generalized eigenvalue problem as the solution of (5.5). It is convenient to introduce the non-singular N-dimensional coordinate transformation

$$\mathbf{P} := \left(\left\langle \mathbf{x}_t \mathbf{x}_t^T - \left\langle \mathbf{x}_t \right\rangle \left\langle \mathbf{x}_t^T \right\rangle \right\rangle \right)^{\frac{1}{2}} \,, \quad \mathbf{W}' := \mathbf{W}\mathbf{P} \,.$$

With respect to $\mathbf{W}'$ the optimization problem then becomes

$$\text{Minimize} \quad \mathrm{tr}\left(\mathbf{W}' \mathbf{P}^{-1} \left\langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^T \right\rangle \mathbf{P}^{-T} \mathbf{W}'^T \right) \qquad \text{subject to} \quad \mathbf{W}'\mathbf{W}' = \mathbf{I} \,. \tag{5.7}$$

Its solution is the matrix $\mathbf{W}'$, the row vectors of which are the eigenvectors of $\mathbf{P}^{-1} \left\langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^T \right\rangle \mathbf{P}^{-T}$ that correspond to the M smallest eigenvalues. Back transformation yields the optimal $\mathbf{W}$

$$\mathbf{W} = \mathbf{W}' \mathbf{P}^{-1} \,.$$

5.1.2 Non-linear SFA by expansion

A nice property of the linear SFA is the particular simplicity of its solution. The whole learning problem is in terms of second order statistics of the inputs. However, for non-linear projections $\mathbf{g}$, this structure is in general lost.

This problem can be solved with the execution of the linear SFA in some feature space, the data have been non-linearly projected into, beforehand,

$$\mathbf{x}_t \to \mathbf{\Phi}(\mathbf{x}_t) \,.$$

Then the expanded vector is the input for linear SFA, and all second order statistics and discrete derivatives are with respect to $\mathbf{\Phi}(\mathbf{x}_t)$. Usually the feature space is of much higher dimension than the input space. Wiskott and Sejnowski (2002) suggest to do the expansion explicitly, using all monomials of the elements of $\mathbf{x}_t$ up to a certain order d. For example, $d = 2$ results in quadratic SFA and the expanded input vector $\mathbf{\Phi}(\mathbf{x}_t)$ contains all elements of $\mathbf{x}_t$ together with all products of any two elements of $\mathbf{x}_t$,

$$\mathbf{\Phi}(\mathbf{x}_t) = (x_{1,t}, \ldots, x_{N,t}, x_{1,t}^2, \ldots, x_{1,t}x_{N,t}, x_{2,t}^2, \ldots, x_{N,t}^2)^T \,.$$

This expansion soon becomes rather large – an order d expansion of a N-dimensional vector has the length $\sum_{k=1}^{d} \binom{N+k-1}{k}$. Thus, small expansions of order $d = 2$ are often used. To achieve nevertheless a high degree of non-linearity, multiple quadratic SFA can be performed, i.e. the outputs $\mathbf{y}_t$ are expanded and $\mathbf{\Phi}(\mathbf{y}_t)$ is input for the adjacent SFA.

5.1.3 Kernel-SFA

The fact that the linear SFA is fully based on second order statistics allows to introduce non-linear projection functions $\mathbf{g}$ also by an implicit mapping into a high dimensional feature space. This is completely equivalent to the extension of Principal Component Analysis to kernel-PCA (Schölkopf et al., 1998), and to other second order kernel methods. Thus, only a brief derivation of the kernel SFA learning problem shall be given here. For more details see Bray and Martinez (2002), (cf. also section 4.4).

As with all kernel methods, the mapping to feature space is not explicitly defined, but only by means of scalar products of feature vectors,

$$\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{z}) := k(\mathbf{x}, \mathbf{z}) \,,$$

where $k: \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is a suitable kernel function. Given a finite dataset $(\mathbf{x}_t)_{t\in \mathrm{I}}$ of length $T = |\mathrm{I}|$ and a set of $T' = |\mathrm{I}'|$ support vectors $(\mathbf{z}_t)_{t=\mathrm{I}'}$, all possible scalar products can be summarized in the kernel matrix $\mathbf{K}$,

$$K_{ij} = k(\mathbf{x}_{t_i}, \mathbf{z}_{t_j}) \,, \quad t_i \in \mathrm{I},\ t_j \in \mathrm{I}' \,.$$

However, the second order quantities in the linear SFA problem formulation would contain outer products of the feature vectors, rather than scalar products. Therefore, the row vectors $\mathbf{w}_i^T$ of $\mathbf{W}$, which are quantities in the feature space in the kernel-SFA approach, are represented by linear combinations of the support vectors mapped to feature space,

$$\mathbf{w}_i^T = \sum_{j=1}^{T'} V_{ij} \mathbf{\Phi}(\mathbf{z}_{t_j})^T \,, \quad t_j \in \mathrm{I}' \,.$$

This eventually allows a problem formulation in feature space which is exclusively in terms of scalar products:

$$\text{Minimize} \quad \mathrm{tr}\left(\mathbf{V}\dot{\mathbf{K}}^T\dot{\mathbf{K}}\mathbf{V}^T\right) \qquad \text{s.t.} \quad \mathbf{V}\left(\frac{1}{T-1}\mathbf{K}^T\mathbf{K} - \frac{1}{T^2}(\mathbf{K}^T\bar{\mathbf{1}})(\bar{\mathbf{1}}^T\mathbf{K})\right)\mathbf{V}^T = \mathbf{I} \,, \tag{5.8}$$

where $\dot{K}_{ij} := k(\mathbf{x}_{t_i+1}, \mathbf{z}_{t_j}) - k(\mathbf{x}_{t_i}, \mathbf{z}_{t_j})$, $\mathbf{V}$ is the $M \times T'$ matrix with the elements V_{ij}, and $\bar{\mathbf{1}}$ is a vector containing T times the element 1. This linear constrained optimization problem in feature space is of dimensionality T', the number of support vectors.

Depending on T and T', care must be taken that the both matrices of the resulting generalized eigenvalue problem are not singular, otherwise numerical instabilities could arise. Usually the support vector set is much smaller than the dataset, $T' \ll T$, and not necessarily a subset of the dataset. With

$$\mathbf{k}(\mathbf{x}_t)^T := (k(\mathbf{x}_t, \mathbf{z}_{t_1}), \ldots, k(\mathbf{x}_t, \mathbf{z}_{t_{T'}})) \tag{5.9}$$

being a row vector of the kernel matrix $\mathbf{K}$, (5.8) can also be written as

$$\text{Min.} \quad \mathrm{tr}\left(\mathbf{V}\left\langle\dot{\mathbf{k}}(\mathbf{x}_t)\dot{\mathbf{k}}(\mathbf{x}_t)^T\right\rangle\mathbf{V}^T\right) \qquad \text{s.t.} \quad \mathbf{V}\left(\left\langle\mathbf{k}(\mathbf{x}_t)\mathbf{k}(\mathbf{x}_t)^T\right\rangle - \left\langle\mathbf{k}(\mathbf{x}_t)\right\rangle\left\langle\mathbf{k}(\mathbf{x}_t)^T\right\rangle\right)\mathbf{V}^T = \mathbf{I} \,,$$

where the expectation is over all $t \in \mathrm{I}$. In this respect kernel-SFA is the same as SFA with explicit expansion, were the expansion $\mathbf{\Phi}(\mathbf{x}_t) = \mathbf{k}(\mathbf{x}_t)$ is a non-linear function $\mathbb{R}^N \to \mathbb{R}^{T'}$ parameterized with $(\mathbf{z}_t)_{t\in\mathrm{I}'}$. Note that these two different ways of interpretation are common to all second order kernel methods. Often one may find it easier to give an interpretation of the feature space mapping, when the expansion is defined in terms of kernel functions, with a support vector set, that is a subset of the data set.

5.2 Empirical slowness

For a signal $(\mathbf{y}_t)_{t\in\mathrm{I}}$, that is the result of a slow feature analysis, the overall slowness of $\mathbf{y}$ is the value

$$S(\mathbf{y}) := \sum_i \left\langle (y_{i,t} - y_{i,t+\delta t})^2 \right\rangle = \mathrm{tr}\left\langle \dot{\mathbf{y}}_t \dot{\mathbf{y}}_t^T \right\rangle = \left\langle \dot{\mathbf{y}}_t^T \dot{\mathbf{y}}_t \right\rangle \tag{5.10}$$

given that

$$\left\langle \mathbf{y}_t \right\rangle = 0 \quad \text{and} \quad \left\langle \mathbf{y}_t \mathbf{y}_t^T \right\rangle = \mathbf{I} \,. \tag{5.11}$$

Equation (5.10) is generally unsuited for the computation of empirical slowness values, just by calculating empirical expectation values, because the conditions (5.11) may be not fulfilled and results may be distorted. Thus, the estimates for (5.11) must be included in the slowness value.

$$\begin{aligned} S(\mathbf{y}) &= \operatorname{tr}\left(\left\langle \mathbf{y}_t \mathbf{y}_t^T - \left\langle \mathbf{y}_t \right\rangle \left\langle \mathbf{y}_t^T \right\rangle \right\rangle^{-\frac{1}{2}} \left\langle \dot{\mathbf{y}}_t \dot{\mathbf{y}}_t^T \right\rangle \left\langle \mathbf{y}_t \mathbf{y}_t^T - \left\langle \mathbf{y}_t \right\rangle \left\langle \mathbf{y}_t^T \right\rangle \right\rangle^{-\frac{1}{2}} \right) \\ &= \operatorname{tr}\left(\left\langle \mathbf{y}_t \mathbf{y}_t^T - \left\langle \mathbf{y}_t \right\rangle \left\langle \mathbf{y}_t^T \right\rangle \right\rangle^{-1} \left\langle \dot{\mathbf{y}}_t \dot{\mathbf{y}}_t^T \right\rangle \right) \end{aligned} \tag{5.12}$$

Apparently, this slowness value is invariant to non-singular affine transformations, hence

$$S(\mathbf{y}_t) = S(\mathbf{A}\mathbf{y}_t + \mathbf{b}) \,. \tag{5.13}$$

When the empirical slowness of a finitely long signal $(\mathbf{y}_t)_{t \in \mathrm{I}}$ shall be computed, expectations are replaced by empirical expectations (we assume that for every $t \in \mathrm{I}$ also a value $\mathbf{y}_{t+\delta t}$ is available)

$$\hat{S}(\mathbf{y}_t) = \frac{T-1}{T} \operatorname{tr}\left(\left(\sum_{t \in \mathrm{I}} \mathbf{y}_t \mathbf{y}_t^T - \frac{1}{T} \sum_{t \in \mathrm{I}} \mathbf{y}_t \sum_{t \in \mathrm{I}} \mathbf{y}_t^T \right)^{-1} \sum_{t \in \mathrm{I}} \dot{\mathbf{y}}_t \dot{\mathbf{y}}_t^T \right) , \tag{5.14}$$

where $T = |\mathrm{I}|$.

The question arises, whether this estimator is biased and how its variance behaves with respect to the number samples $|\mathrm{I}|$. Therefore, in figure 5.1 empirical slowness values are displayed as functions of the number of samples in I.

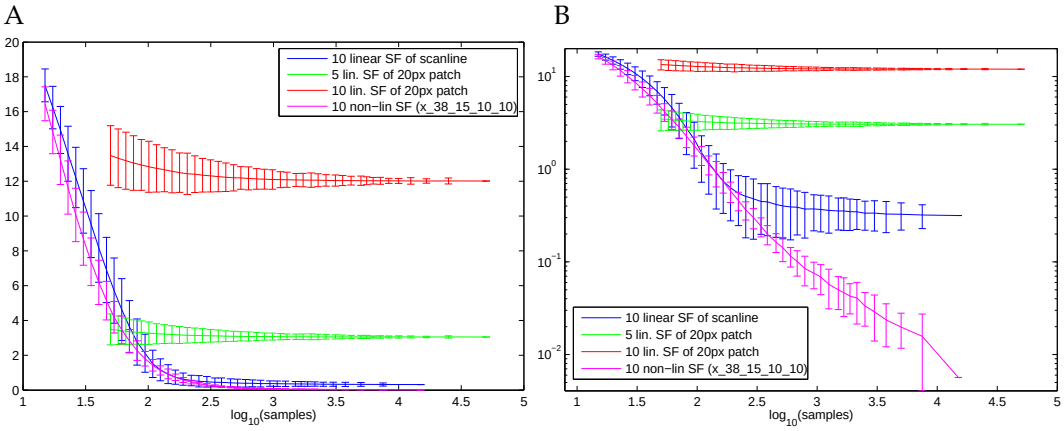


Figure 5.1: Empirical slowness values of 4 different slow signals as functions of the number of samples. **A**: linear y-axis, **B**: logarithmic y-axis. Error bars indicate the standard deviation intervals. For explanation see text.

The experiments were done as follows: A slow signal was split into non-overlapping blocks of equal size. For every block the empirical slowness was computed according to equation (5.14). The average slowness values, together with their standard deviation intervals, are plotted as functions of the number of samples per block. The experiment was done with 4 slow signals, and up to 1,000-fold splits. The first signal (blue curve) contained the 10 slowest linear projections of a horizontal scan line taken form a natural video. The second (green curve) and third (red curve) contained the 5, rsp. 10 slowest linear features, that were derived from 20 pixels long, randomly positioned patches of that video scan line. See section 5.4 for a detailed description of the video scan line experiments. The fourth slow signal (magenta curve) contained the 10 slowest non-linear features, that were learned

from principal components projections of natural video data. See section 5.3 for a detailed description of this experiment.

The curves show that the slowness value is biased to larger values, when the number of samples is very small. However, one can also see that the bias becomes small for reasonable sample sizes, in the scan line experiments for about 10^3 and more samples. This value is an important to know in order to design experiments that involve cross validation for model selection. There, the test sets should be large enough to achieve unbiased test values.

5.3 SFA on video data

Video data were recorded with a *BumbleBee* two-lens stereo vision camera system from *Point Grey Research Inc.*[1]. The camera was mounted on top of a cart, which was pushed in the corridor of an office building. The video was recorded at 16 Hz frame rate with 15,000 frames in total. Figure 5.5 shows some example frames of the stereo video.

From this video 38 principal components were extracted using generalized Hebbian learning (Sanger, 1989). Figure 5.2 shows these principal components and the mean of all frames in the video. One can see that, because of the scenery in the video, vertical structures dominate the principal components. Larger principal components contain lower frequencies. Interestingly, there is not much stereo information contained – the left and right halfs are almost equal in all principal components.

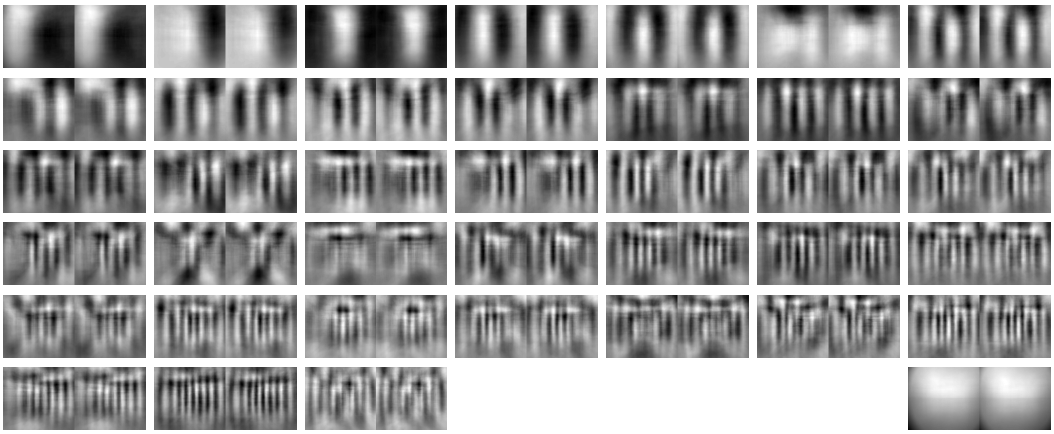


Figure 5.2: The first 38 largest principal components of the video frames ordered by decreasing eigenvalues from left/top to right/bottom. The rightmost image in the bottom row is the mean of all frames in the video.

Then, every video frame was projected onto the principal components, leading to a time series of a 38 dimensional vector, which was used as input for non-linear slow feature analysis in three steps. The setup was

$$38\,\text{PC's} \;\rightarrow\; 15\,\text{SC's} \;\rightarrow\; 15\,\text{SC's} \;\rightarrow\; 10\,\text{SC's}\,, \tag{5.15}$$

meaning that from the 38 dimensional input the 15 slowest components were computed. These were used to compute the 15 slowest components out of them, and so on, leading to ten output components. Clearly, such an architecture is only meaningful, when every SFA step involves non-linearities. In the present experiment non-linearity was achieved by explicit quadratic expansion of the inputs of every SFA processing step.

This procedure results in 10 signals of unit variance, which are pairwise uncorrelated and slowly changing. However, besides that these signals exhibit another interesting property, which can be seen in figure 5.3: The slow signals are also sparse. However, the sparse directions are not aligned to the coordinate axes, and, hence, the slow signals are

[1] `http://www.ptgrey.com/products/bumblebee/index.asp`

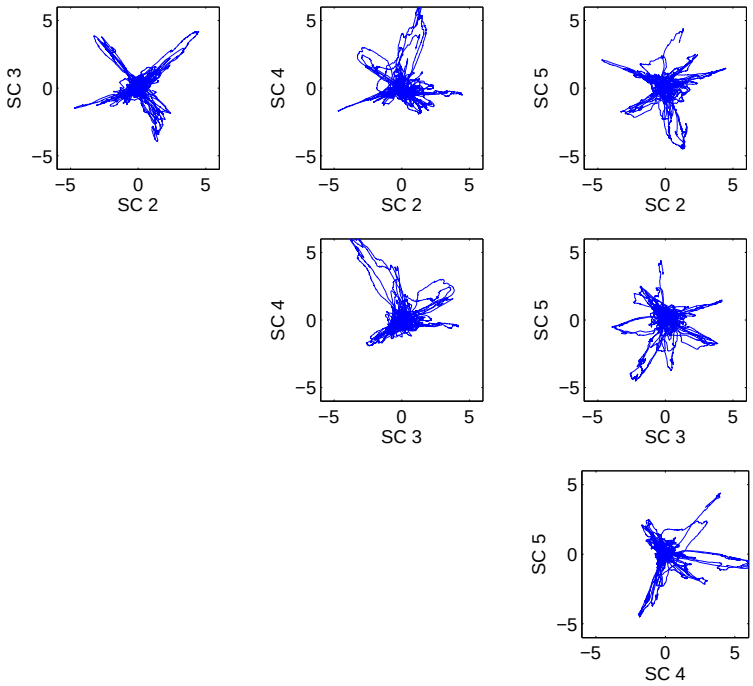


Figure 5.3: Scatter plot of the slow signals 2-5 resulting from the setup (5.15). For every pair the trajectory of one signal is plotted against the other. The slow signals have directions of strong sparseness.

not independent. This gives rise to the idea to apply some linear ICA algorithm to the slow signals as post processing. This is well justified, because the overall slowness value does not change under non-singular affine transformations, (5.13). Figure 5.4 shows the time course of the signals that are achieved through the application of FastICA to the slow signals. The overall slowness of these signals did not change, but they are much sparser now. Interestingly, they exhibit strong unipolarity, which could be interpreted as the degree of "presence of features" detected by the non-linear SFA. Not only in this experiment it could be observed that these features tend to follow each other. For example, note the peaks near 3,000 and 12,000 of signals 5 and 7, which come very close, but do not overlap.

In figure 5.5 there are displayed some frames taken from the video at positions where some of the slow signals assume particularly large values. One can see that they all detect a certain scene, but allow for variations of the viewing perspective to some degree. Thus, the non-linear SFA was able to unsupervisedly learn invariances of features contained in the presented data.

So far, this experiment was performed without taking care about the generalization performance of the non-linear SFA architecture (5.15). On the training set, which was the whole 15,000 frames of the video, an empirical slowness value of 0.0079 was achieved for all 10 output signals together. However, the architecture with best empirical slowness on 15 fold cross-validation test sets was

$$30\,\text{PC's} \;\rightarrow\; 6\,\text{SC's} \;\rightarrow\; 10\,\text{SC's}\;, \tag{5.16}$$

which had a average slowness value on the test sets of $S_{\text{test}} = 0.0282$. The training slowness value was with $S_{\text{train}} = 0.0161$ considerably below that. However, the intention was to illustrate at least qualitatively, what the detected features are when the slowness value is the best achievable test value. Therefore, an architecture was searched, that yields a training slowness value of about 0.0282. This architecture was found with

$$12\,\text{PC's} \;\rightarrow 10\,\text{SC's} \;\rightarrow\; 10\,\text{SC's}\;, \tag{5.17}$$

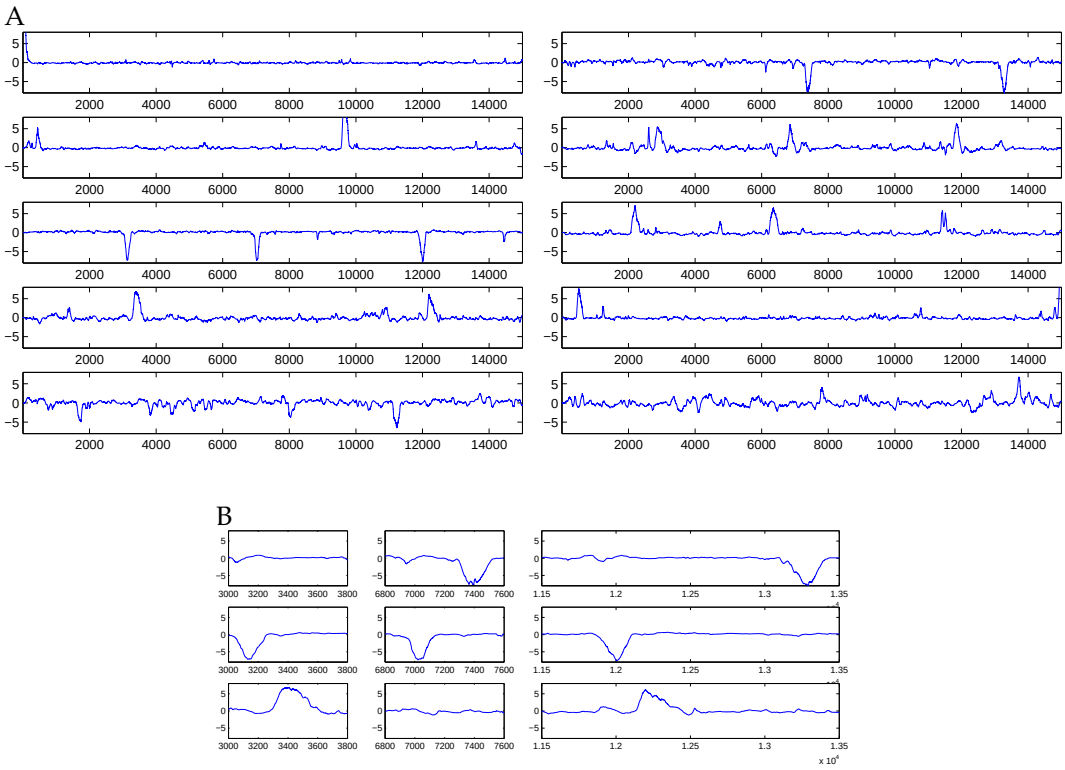


Figure 5.4: **A**: Slow signals of the setup (5.15) after the application of FastICA. The signals are at the same time sparse and slow. **B**: Closeup of signals 2, 5, and 7. A clip of the video with these 3 signals coded in color can be found on the accompanying CD (`video1.avi`).

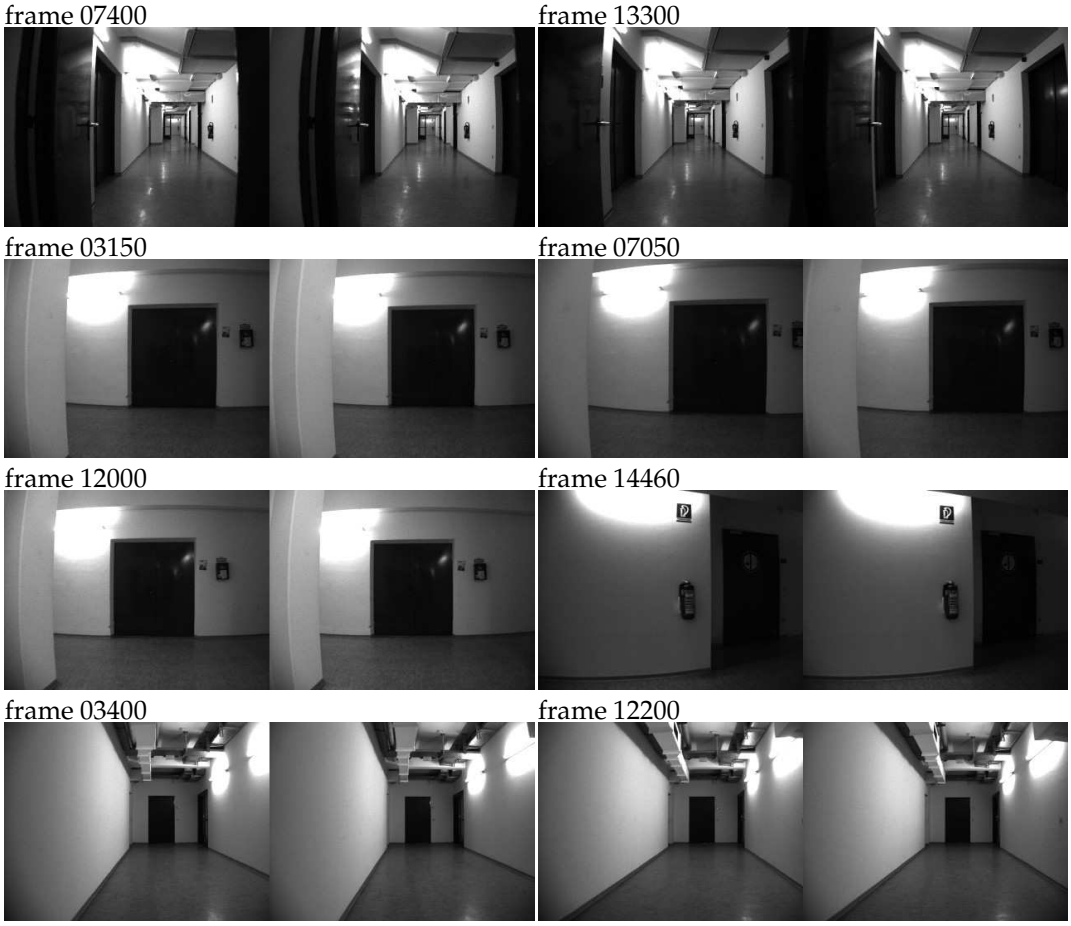


Figure 5.5: Few frames of the stereo video for the illustration of the non-linear SFA results.

and a training value of $S_{\text{train}} = 0.0288$. These slow signals – although they actually constituted the training set – can be considered to be as slow as the slowest possible ones on a test set of architecture (5.16). Figure 5.6 shows the 5th slow signal of the architecture (5.15) together with the most similar one obtained with the less complex architecture (5.17). Besides stronger overall variations, its most apparent that the peak near 14460 strongly increased. Thus, the better generalizing architecture would summarize different, although similar, scenes in one feature (cf. figure 5.5). For the other slow signals the situation is similar.

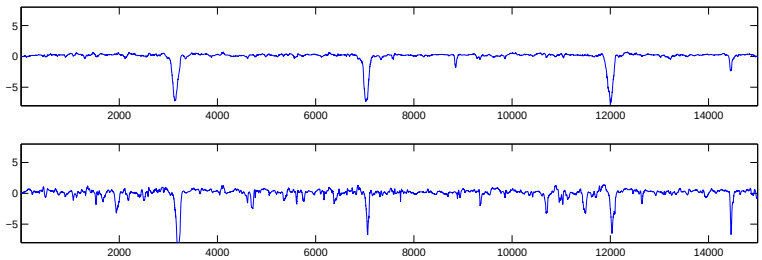


Figure 5.6: Comparison of the 5th slow signal of architecture (5.15) (top) and the most similar one of the architecture (5.17) (bottom).

Thus, one can conclude that non-linear SFA is able to learn specific features, together with their naturally occurring variations, unsupervisedly from video data.

5.4 Hierarchical SFA

5.4.1 Hierarchical architecture

Consider a vector valued stochastic signal

$$(\mathbf{x_t})_{\mathbf{t}\in \mathrm{I}} \ ,$$

which is defined over an index set $\mathrm{I} \subseteq \mathbb{Z}^{d\geqslant 2}$ that is at least two dimensional. Thus, the index $\mathbf{t}$ can be split into a temporal index t, that is used to define the slowness of a signal, and a spatial index r that summarizes the remaining dimensions of $\mathbf{t}$. Without loss of generality be

$$\mathbf{t} = (t, r)^T \quad \text{and} \quad \mathrm{I} = \mathrm{I}_t \times \mathrm{I}_r \ .$$

Note that the terms "temporal" and "spatial" here are arbitrarily defined, and, hence, are detached from the interpretation of a particular dataset (cf. section 1.4.1). Thus, both, t and r, can be in principle scalar or vector valued.

In the following superscripts do not denote powers but the level, or layer, in the hierarchy of the hierarchical SFA. Starting with the signal $(\mathbf{x_t})_{\mathbf{t}\in \mathrm{I}}$ we define

$$\mathbf{y}^0_{tr} := \mathbf{x}_{tr} = \mathbf{x_t} \ .$$

Now, in every layer i of the hierarchy a non-linear multi-channel filter function $\boldsymbol{f}^i$, which is applied only to the spatial dimensions, is learned,

$$\mathbf{y}^i_{tr} = \boldsymbol{f}^i \left(\left(\mathbf{y}^{i-1}_{t(r'+r)} \right)_{r'\in \mathrm{J}^i} ; \mathbf{w}^i \right) \ . \tag{5.18}$$

The function is parameterized with the parameter vector $\mathbf{w}^i$. From the perspective of the temporal dimension this is an instantaneous, stationary function. The set J^i is the receptive field of $\boldsymbol{f}^i$. If the index set in the previous layer is open with respect to the convolution with J^i, it may decrease from layer to layer,

$$\mathrm{I}^i_r = \{ r \mid \quad r + r' \in \mathrm{I}^{i-1} \text{ for all } \quad r' \in \mathrm{J}^i \} \ ,$$

where $\mathrm{I}^0_r = \mathrm{I}_r$. Thus, the hierarchy must end before I^i_r becomes empty. A further property of the hierarchical architecture is that every layer has its own error functions, but no errors are propagated back to the previous layers. For the SFA, in every layer the parameters $\mathbf{w}^i$ must be adjusted so that the slowness value of the signals in the same layer is minimized,

$$S^i(\mathbf{w}^i) = \left(\left\langle \mathbf{y}^i_{tr} (\mathbf{y}^i_{tr})^T \right\rangle - \left\langle \mathbf{y}^i_{tr} \right\rangle \left\langle \mathbf{y}^i_{tr} \right\rangle^T \right)^{-1} \left\langle \dot{\mathbf{y}}^i_{tr} (\dot{\mathbf{y}}^i_{tr})^T \right\rangle \ , \tag{5.19}$$

where

$$\dot{\mathbf{y}}^i_{tr} := \mathbf{y}^i_{(t+\delta t)r} - \mathbf{y}^i_{tr} \ .$$

Depending on the form of $\mathbf{i}$ it may be computationally more convenient separate the normalization term from the cost function, and rather solve a constrained optimization problem:

$$\text{Minimize} \quad \left\langle \dot{\mathbf{y}}^i_{tr} (\dot{\mathbf{y}}^i_{tr})^T \right\rangle \qquad \text{s.t.} \quad \left\langle \mathbf{y}^i_{tr} (\mathbf{y}^i_{tr})^T \right\rangle - \left\langle \mathbf{y}^i_{tr} \right\rangle \left\langle \mathbf{y}^i_{tr} \right\rangle^T = \mathbf{I} \ . \tag{5.20}$$

With every layer the overall receptive field, i.e. the receptive field of $\mathbf{y}^i_{tr}$ with respect to the input signal increases. For a K-layer architecture it is

$$\mathrm{J}_r = \mathrm{J}^K_r \star \mathrm{J}^{K-1}_r \star \ldots \star \mathrm{J}^1_r \ ,$$

where

$$\mathrm{A} \star \mathrm{B} := \{ a + b \mid a \in \mathrm{A}, b \in \mathrm{B} \} \ .$$

This large receptive field size can allow for pretty complex projections performed by the whole network. At the same time, the receptive field in every layer can be comparably small, which is necessary for good generalization performance. If the whole network is taken as one large non-linear filter function of $(\mathbf{x}_{tr})_{r\in \mathrm{I}_r}$, then the hierarchical approach is equivalent to massive weight sharing in every level of the hierarchy. One can expect this to work best if $(\mathbf{x}_{tr})_{r\in \mathrm{I}_r}$ is truly stationary over I_r.

5.4.2 Experiment: video slice

In the following experiment, hierarchical SFA was applied to natural video data. This experiment should be seen as a first step towards a complete artificial visual system, that is able to extract and recognize relevant features and landmarks from visual scenes. These would serve as the optimal inputs to subsequent processing units, which, for example, could perform steering tasks of a mobile robot. The intension of this experiment was to investigate in how far the hierarchical SFA approach constitutes a promising principle for the unsupervised learning of visual processing units. Therefore, the setup was rather simplified in certain aspects and in comparison to a possible real world application.

The data for this experiment were the video used for the experiment in section 5.3, and another video recorded with the same camera and parameters, but on a different floor of the same building. The later one was used for training, while the former one was used for testing purposes. Both videos can be found on the accompanying CD (training video: `floor_+2.avi`, test video: `floor_-1.avi`). The before mentioned simplifications are mainly that the stereo information, i.e. the right half image, was discarded in every frame, and that only one horizontal scan line of every frame was used for the input data. The scan line was the 120th, which is in the center of the frame. Figure 5.7 shows the first 1,000 scan lines of both videos, as images. Such arrays will be called *video slice* in the following. The whole training and test video slices had a length of 15,000 frames/scan lines.

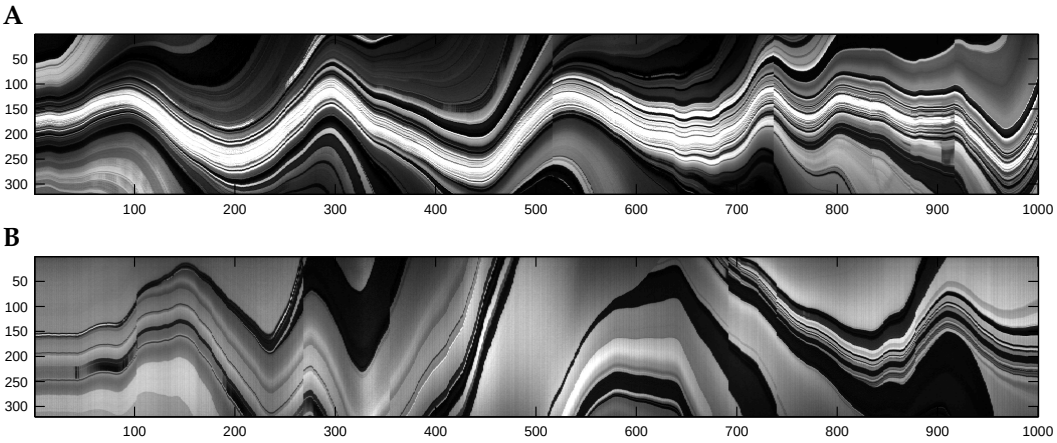


Figure 5.7: Video slice composed from the 120th scan line of the first 1,000 frames of the training video (**A**) and the test video (**B**). Every column is one scan line, the x-axis is the time $t \in [1,\ 10^3]$.

Kernel-SFA with RBF kernels was performed in the first layer, to generate 10 slow signals. Parameter exploration was done for input receptive field sizes of $|\mathrm{J}_r^1|$ =10, 20, 30, and 40 samples. That means the input data were clips or patches of the length $|\mathrm{J}_r^1|$ randomly taken from all scan lines of the video slice. Figure 5.8 shows the results of the exploration of the first layer parameters. These are the number of support vectors (**A**: $N_{SV}^1 = 200$, **B**: $N_{SV}^1 = 300$, **C**: $N_{SV}^1 = 500$) and the kernel width σ^1 (x-axis in the figures). For all trials that had the same values N_{SV}^1 and σ^1, one and the same support vector set was used. The support vector sets were drawn randomly iid. from the whole video slice. The graphs show the training (dashed lines) and the test slowness values (solid lines) with 10-fold cross validation. Therefore the whole slice was partitioned into 10 blocks of equal length, one of which was used for test, the other 9 for training. From the corresponding partitions a total number of $20,000$ patches for training, and $5,000$ patches for test were drawn randomly, iid. The error bars in the figure mark the single standard deviation interval of the training and test slowness values over the 10 cross validation iterations. One can see that

(i) For every input dimension there is an optimal value of the kernel parameter σ.

(ii) The test slowness is always above the training values.

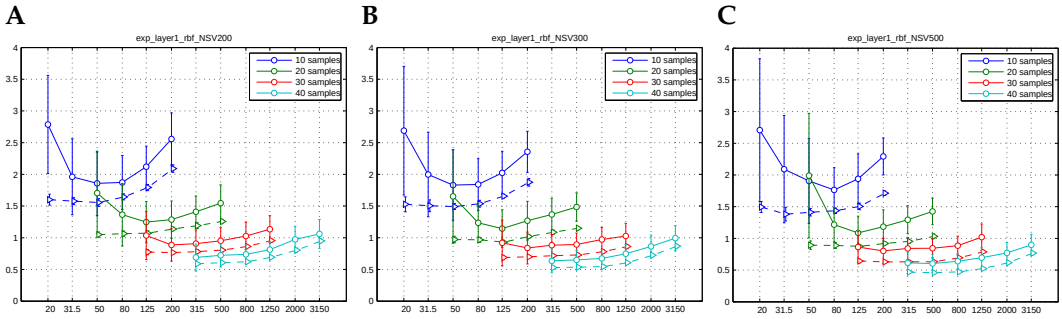


Figure 5.8: Training (dashed lines) and test (solid lines) slowness values of the first layer outputs, for receptive field sizes $|J_r^1| = 10$, 20, 30 and 40, plotted as functions of the first layer kernel parameter σ^1. **A:** $N_{SV}^1 = 200$, **B:** $N_{SV}^1 = 300$, and **C:** $N_{SV}^1 = 500$ support vectors.

(iii) The influence of the number of support is small in the considered range from 200 to 500.

For the second layer various architectures based on RBF kernels with 10 output slow features were tested. The cross validation procedure was the same as in the first layer, except that the inputs now were the outputs of the first layer. Thus, while in the first layer patches were taken from a video scan line, in the second layer patches were taken from a 10 dimensional "vector valued scan line". The inputs and the support vectors for the second layer came from the best performing first layer architectures with $N_{SV}^1 = 300$, which were $\sigma^1 = 80$ for $|J_r^1| = 10$, $\sigma^1 = 125$ for $|J_r^1| = 20$, and $\sigma^1 = 200$ for $|J_r^1| = 30$ (cf. figure 5.8). That means that after the best first layer architectures were detected using cross validation, these architectures were trained once more with 20,000 patches from the unpartioned slice, the whole slice was filtered with this re-trained architecture, and any input samples and support vectors for the next layer were taken from these outputs. All trials, that had the same layer 1 architecture, and that had the same layer 2 receptive field size, used the same set of support vectors, which were randomly iid. taken from the layer 1 outputs. Because the inputs to layer 2 are already 10 dimensional, smaller receptive field sizes of $|J_r^2| = 5$, 10, 20 and 30 samples were tested here. The second layer results are displayed in figure 5.9.

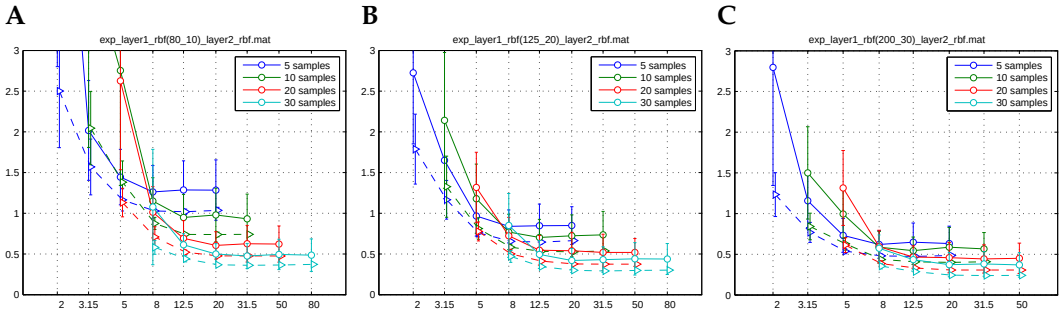


Figure 5.9: Training and test slowness values for the second layer. Inputs were the outputs of the first layer architectures **A:** $|J_r^1| = 10$, $\sigma^1 = 80$, **B:** $|J_r^1| = 20$, $\sigma^1 = 125$, and **C:** $|J_r^1| = 30$, $\sigma^1 = 200$. Dashed lines mark the training and solid lines the test slowness values for receptive field sizes $|J_r^2| = 5$, 10, 20 and 30 as functions of the second layer kernel parameter σ^2.

If one sorts the results shown in figures 5.8 and 5.9 by the overall receptive field size one can evaluate the different architectures. It is in general not meaningful to compare architectures with different overall receptive field sizes. Table 5.1 lists the corresponding optimal test slowness values. One can see that for the same (or one element less) overall receptive field sizes, the two layer approach always leads to better performance in the scan line SFA experiment.

$\lvert J_r \rvert$	$\lvert J_r^1 \rvert$	$\lvert J_r^2 \rvert$	S
19	10	10	0.95
20	20	–	1.14
29	10	20	0.60
29	20	10	0.70
30	30	–	0.84
39	10	30	0.47
39	20	20	0.52
39	30	10	0.54
40	40	–	0.65
49	20	30	0.42
49	30	20	0.44

Table 5.1: Performance comparison of the optimal one- and two-layer architectures for different overall receptive field sizes $|J_r|$. At the same (or slightly smaller) receptive field sizes, the two-layer approach always shows better performance, in terms of the average test slowness values.

The experiment was carried on with the two-layer architecture

$$|J_r^1| = 20,\ \sigma^1 = 125,\ N_{SV}^1 = 300 \quad \rightarrow \quad |J_r^2| = 30,\ \sigma^2 = 200,\ N_{SV}^2 = 300\,. \tag{5.21}$$

Both layers were re-trained on the whole slice, using the same support vector sets, that were also used in the cross-validation experiment. An affine transformation determined by FastICA was applied as post processing, so that the sparseness of the individual outputs was maximized. Figure 5.10 shows parts of the resulting 10 slow signals as individual slices. The scan lines are now by $|J_r| - 1 = 48$ samples shorter. One can observe the same properties, that had been found in the experiment of section 5.3: the slow features are sparse, unipolar and smooth in the temporal direction (x-axis). Also the slow features seem to adjoin each other, like it was observed in the other experiment.

The remainder of this experiment is about to show that the learned slow features are really useful for visual processing tasks. Because the present video material is not labeled in any way, also the next step had to be learned unsupervisedly. For a steering architecture, that is based on visual inputs only, it is very important to be able to detect certain fixed points in the presented visual scene and to focus on them. Interestingly, this task can be solved to some degree already with the presented simplistic approach of slow features learned from single scan lines.

A simple linear combination, $\mathbf{a} \in \mathbb{R}^{10}$ of the second layer slow features, $\mathbf{y}_{tr}^2$, was searched, so that a spatial minimum of $\mathbf{a}^T\mathbf{y}_{tr}^2$ would correspond to the location of some landmark in the visual scene. Parameterized with $\mathbf{a}$ the minimum is

$$q_t(\mathbf{a}) := \arg\min_r \left(\mathbf{a}^T\mathbf{y}_{tr}^2\right)\ .$$

In the following figures the range of definition of $\left(\mathbf{a}^T\mathbf{y}_{tr}^2\right)_{r\in I_r^2}$, is mapped to the interval $I_r^2 := [-1, 1]$ for convenience. Because the video data were not labeled, the optimal $\mathbf{a}$ had to be learned unsupervisedly. The objective was again motivated by SFA: the optimal $\mathbf{a}$ should be the one for which the signal $(q_t(\mathbf{a}))_{t\in I_t}$ was as slow as possible,

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sqrt{\left\langle (q_t - q_{t+1})^2 \right\rangle}\,.$$

However, no differentiable (w.r.t. $\mathbf{a}$) objective function could be derived therefrom. Therefore, $\mathbf{a}$ was determined by simulated annealing. The optimization problem turned out to be a fairly "good-natured" one, which certainly is also due to the comparable low dimensionality of $\mathbf{a}$. Thus, a close to optimal $\mathbf{a}$ could be robustly determined as

$$\mathbf{a} = (-3.91,\ -4.75,\ -3.91,\ 0.91,\ -1.01,\ 1.43,\ 1.77,\ -4.35,\ -2.02,\ -0.93)^T\,,$$

with an objective function value $\sqrt{\left\langle (q_t - q_{t+1})^2 \right\rangle} = 0.1187$. Figure 5.11.A shows a piece of the input video, $(x_{tr})_{t\in[8400,9400],\, r\in I_r^0}$, and figure 5.11.B shows $\left(\mathbf{a}^T\mathbf{y}_{tr}^2\right)_{t\in[8400,9400],\, r\in[-1,1]}$, both

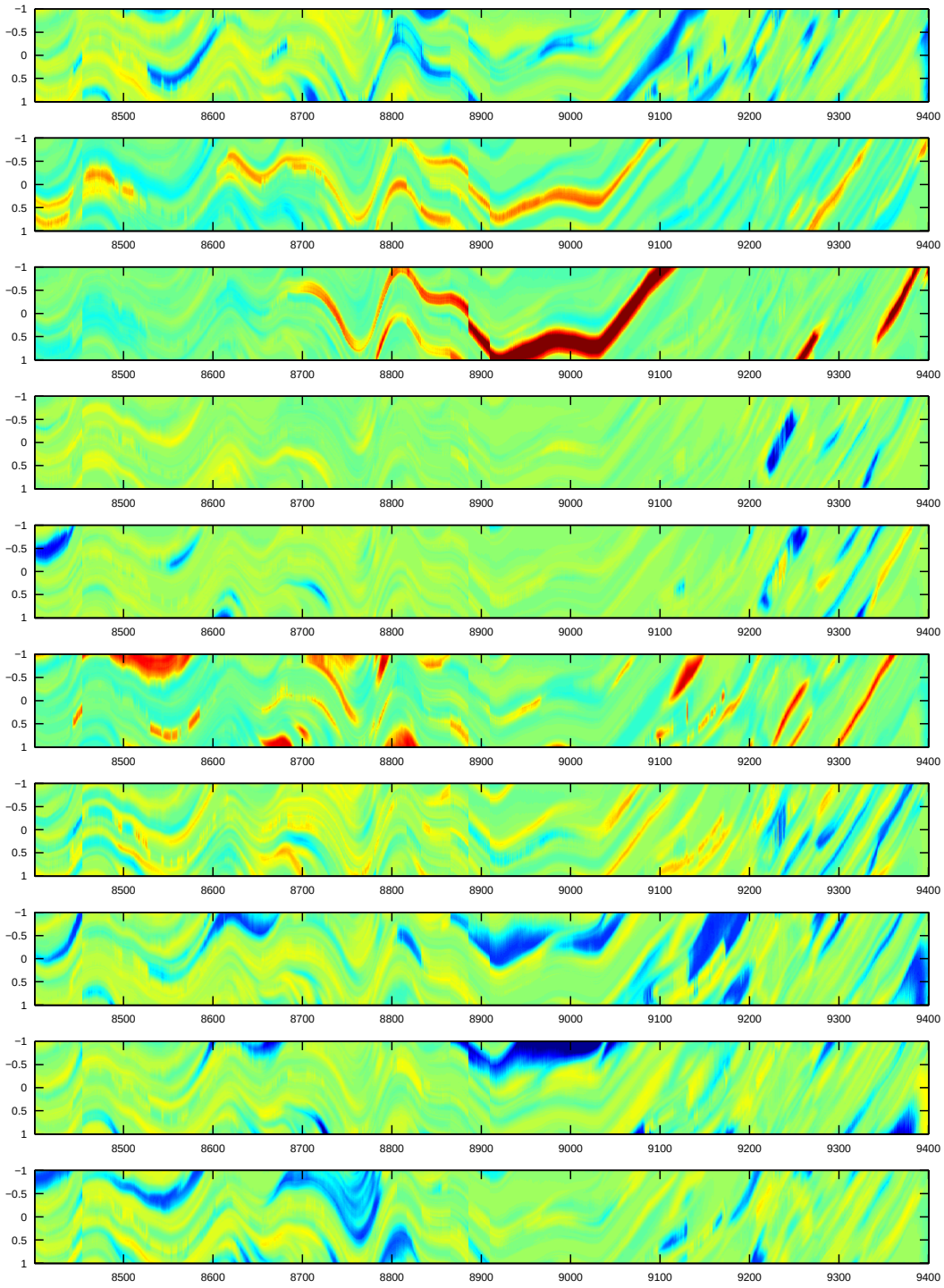


Figure 5.10: A piece of 1,000 scan lines of the training video slice, filtered with the two-layer SFA architecture (5.21). Every one of the 10 outputs is displayed as a separate slice. Color ranges from -6 (dark blue) over 0 (green) to +6 (deep red).

with $(q_t)_{t\in[8400,9400]}$ plotted on top of it (blue dots). One can see that the course of q_t is piecewise smooth and aligned to the course of the input scan lines. In particular, there where the course is smooth, q_t seems to fixate upon an imaginary point in the real world environment. This is illustrated in figure 5.11.C. The three frames show the same visual scene from different viewing angles. The horizontal line is the scan line used for the analysis – the remainder of the frame is only for illustration purposes. The curve on the upper half is the course of $(\mathbf{a}^T\mathbf{y}_{tr}^2)_{r\in[-1,1]}$, stretched so that its minimum attaches the middle, and its maximum the top of the frame. The vertical line goes through the minimum of the curve, and, hence, marks q_t. One can see that it sticks to some imaginary point, that is fixed with respect to the real world environment. This behavior can be observed throughout the whole video (cf. the video `res_floor_+2.avi` on the accompanying CD): q_t fixates upon some point until this point leaves the visual scene, or another point becomes more attracting. Often $(\mathbf{a}^T\mathbf{y}_{tr}^2)_{r\in[-1,1]}$ has two or more almost equally good minima, in which cases one could extract the other local minima as well.

In order to evaluate the generalization performance of the slow feature filters, as well as the projecting vector $\mathbf{a}$, they were applied to the test video, which was recorded with the same setup but in a different environment. No learning or parameter adjustments were done with respect to this video. Figure 5.12 shows the equivalent to figure 5.11 on the test video. As expected the value of $\sqrt{\left\langle (q_t - q_{t+1})^2 \right\rangle} = 0.1943$ was worse than for the training video. This is clear, because the environment and lightning conditions in the test video had been quite different, and had not been part of the training. However, also this yields reasonable results which one can see from the example frames in figure 5.12.C and the whole video (file `res_floor_-1.avi` on the CD).

5.4.3 Discussion of the hierarchical SFA experiment

At the beginning of the section it was mentioned that the presented experiment should be considered as a first step towards the unsupervised learning of a complete hierarchical SFA based visual system. Because this is still work in progress, some consideration about possible real world applications are appropriate here.

The detection of landmarks, i.e. points that are fixed with respect to the true environment, is of particular interest for steering tasks, that are based on visual inputs. If one could reliably identify one and the same landmark point in two (or more) frames, and if at the same time the motion of the camera from one frame to the other was known, then the real world position of the landmark point relative to the camera could be inferred. Vice versa, if the real world distance of the landmark to the camera was known, then the egomotion of the camera could be inferred from the position changes of the landmark from one frame to the other. It can be assumed that the detection of several landmarks at once can help to increase the accuracy of such methods.

The learned slow features, however, may not only be useful for the detection of landmarks. In case that labeled video data were available, one could think of supervised learning of interesting outputs q_t or q_{tr}, may be with more complex post processing models than the linear projection $\mathbf{a}$. For a steering application one would use target data $(z_{tr})_{r\in\mathrm{I}_r}$ in every frame, that define how expensive it is to go towards a direction r. It is important, however, that the target data is a slow signal, otherwise one would not expect the hierarchical SFA to be the appropriate method for the first processing steps.

Of course, there is actually no good reason to discard most of every frame and use only the information of a single scan line. It was done here in this way just in order to keep the setup of this first experiment as simple as possible. Going towards real applications one would use two-dimensional patches of the whole frame, rather than patches from a scan line. However, the choice to use a horizontal scan line in the presented experiment, and not a vertical one, was not arbitrary. Because the camera motion during recording the video was of three degrees of freedom in the horizontal plane (forward- and sidewards motion, rotation around the vertical axis), horizontal lines are much more informative than vertical ones. By the same reasons one can expect that horizontally elongated patches are superior

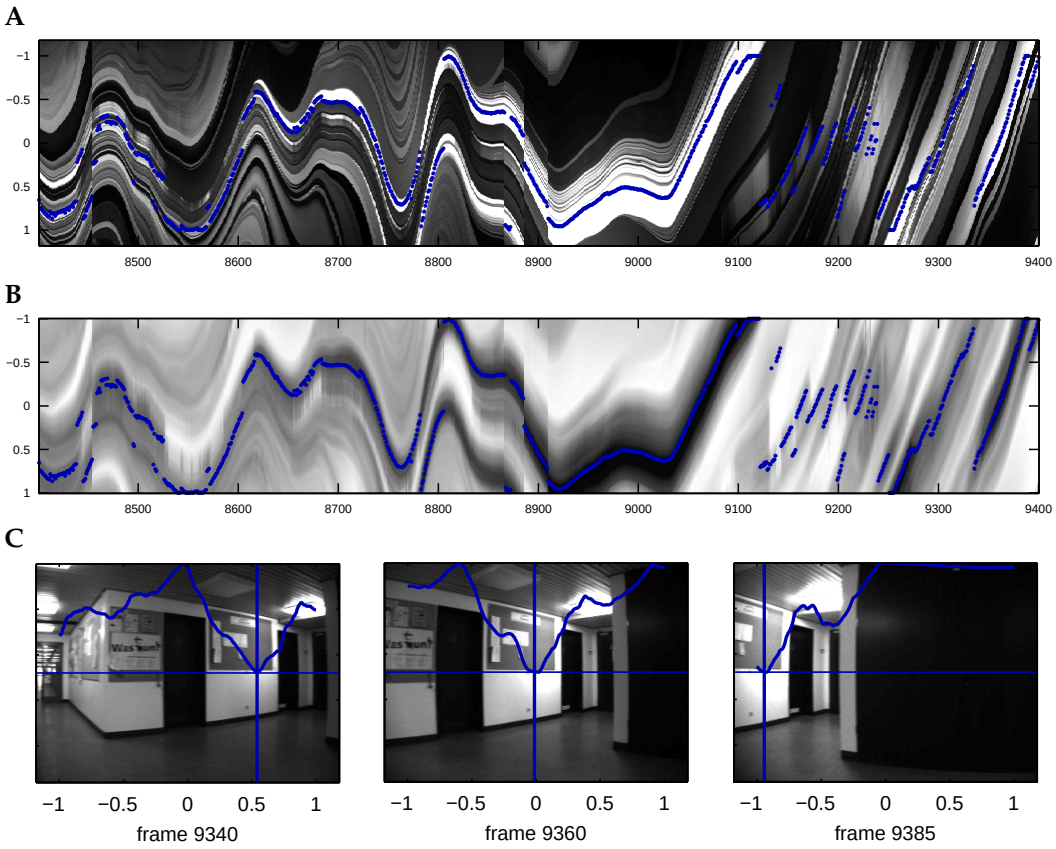


Figure 5.11: **A:** A clip of 1000 scan lines of the training video slice. **B:** The results of the analysis for the same clip, $(\mathbf{a}^T\mathbf{y}_{tr}^2)_{t\in[8400,9400],r\in[-1,1]}$. The blue dots in **A** and **B** mark the value of q_t. **C:** The processed scan line (horizontal line), the value q_t (position of the vertical line), and the course of $(\mathbf{a}^T\mathbf{y}_{tr}^2)_{r\in[-1,1]}$ plotted for three example frames of the training video. The fixation of q_t upon an imaginary point in the real environment is invariant to the viewing perspective.

to squared or vertically elongated ones, when two-dimensional receptive fields are used.

Clearly, with two-dimensional patches the computational costs for the filtering operations increase accordingly. This has the strongest impact on the parameter exploration phase, but also is important when a learned architecture is applied to new input videos. If the non-linearities in the individual SFA layers are achieved through kernel SFA, then most of the computational costs can be saved with a sparse solution of only few support vectors. Because kernel SFA is a second order optimization procedure, the Hyper-Elliptical Conjugate Gradient Descent algorithm (cf. section 4.4) can be used to find sparse solutions.

Improvements of the accuracy of algorithms that base on hierarchical SFA can be expected with more training data. This does not just mean to purely increase the size of the datasets, but also to include as much of the variability as possible, that can occur in new data. In the presented example the training video was taken for an instance of finite length of a stochastic process $(x_{tr})_{(t,r)^T\in\mathrm{I}}$. During learning any expectations were replaced by empirical expectations over t and r. In doing so one assumes stationarity and ergodicity (cf. section 1.2) of the stochastic process. Applying any learned filters to a new video is of course only meaningful, if it can be considered as another instance of the same stochastic process. Then, however, one may find that test errors are much larger than training errors, which is an indicator for the ergodicity no to be provided as assumed. The only chance to improve matters in such situations is to compute empirical expectations not only over the spatial and temporal dimensions, but also over as many instances of the stochastic process, i.e. different videos, as possible. Besides the non-ergodicity effects, more training data

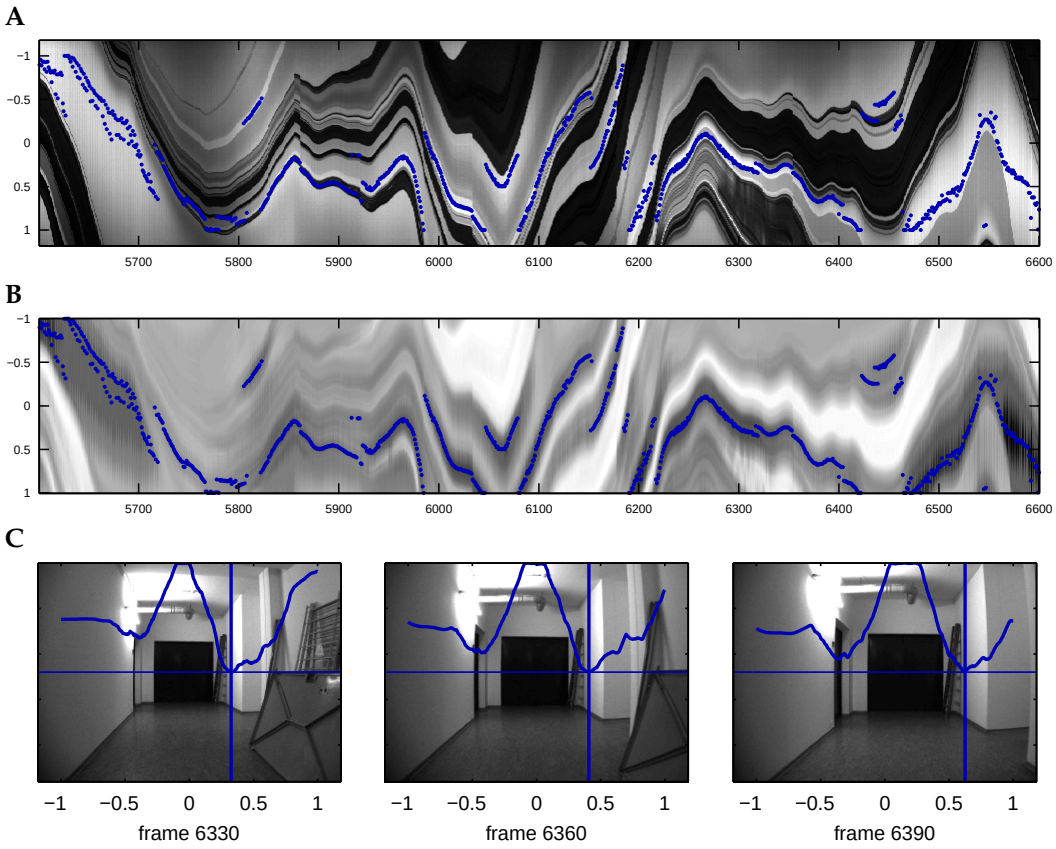


Figure 5.12: The same illustrations as in figure 5.12, but the slow feature filters and the projection vector **a** are applied here to an unseen test video. The results are comparable, which proves the good generalization abilities of the method.

allow for increasingly complex architectures. This, mainly involves increasing the number of layers, increasing the receptive field sizes and using more complex architectures in the individual layers (e.g. smaller kernels).

Thus, while in the present work the hierarchical SFA approach was shown to be promising, for unsupervised learning of relevant features to build an artificial visual system, for the future work it remains to further explore the relations between the amount of training data and the model complexity, under consideration of the required computational costs.

Appendix A

Independence of mixture densities

Consider a family of random vectors $(\mathbf{x}_n)_{n=1\ldots N}$ the elements of which are independent random variables. It holds for all $n \in [1, N]$

$$p(\mathbf{x}_n) = \prod_{i=1}^{M} p(x_{n,i}) \,. \tag{A.1}$$

We call these *independent random vectors* because their elements are independent random variables. The question is under what conditions a mixture of these independent random vectors is independent as well. For the mixture random vector $\mathbf{x}$ holds

$$p(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n p(\mathbf{x}_n) \,. \tag{A.2}$$

The elements of $\mathbf{x}$ are not necessarily independent. An example to show this is provided in figure A.1.

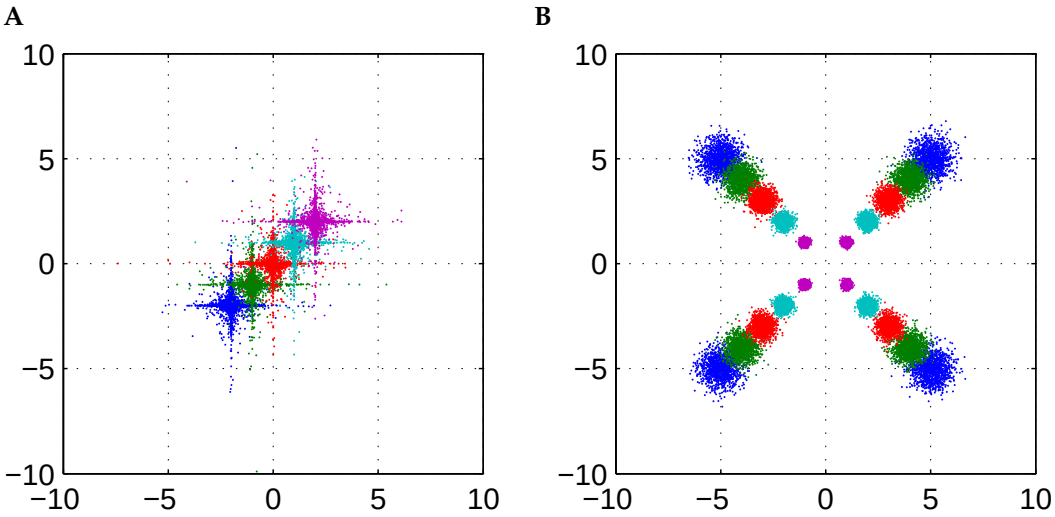


Figure A.1: Two examples for mixture densities, which are composed of five individually independent random vectors. **A:** The mixture components differ in their means and introduce second order dependencies. **B:** The mixture components differ in their variances and introduce higher order dependencies.

Independence of the components x_i of the random mixture vector $\mathbf{x}$ is equivalent to the relation

$$p(\mathbf{x}) = \prod_{i} p(x_i) = \prod_{i=1}^{M} \sum_{n=1}^{N} \alpha_n p(x_{n,i}) \,. \tag{A.3}$$

However, the distribution of the mixture is given by

$$p(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n p(\mathbf{x}_n) = \sum_{n=1}^{N} \alpha_n \prod_{i=1}^{M} p(x_{n,i}) \; . \tag{A.4}$$

Apparently, these two equations are equivalent if for all mixture components $n_1, n_2 \in [1, N]$ holds

$$p(\mathbf{x}_{n_1}) = p(\mathbf{x}_{n_2}) \; . \tag{A.5}$$

Thus, a sufficient (but somewhat trivial) condition for the independence of the mixture is that all its components are independent and identically distributed.

However, this condition can be considerably relaxed for independence up to limited order. Independence of a random vector is equivalent to vanishing cross-cumulants of this vector's elements. So if one can provide conditions under which the cumulants of the mixture are proportional to the sum of the components cumulants, these conditions are sufficient for the independence of $\mathbf{x}$.

Cumulants are the coefficients of the Taylor expansion of the *cumulant generating function* (cf. McCullagh (1987) for a foundation on multi-variate cumulants and moments),

$$K_{\mathbf{x}}(\boldsymbol{\xi}) = \log M_{\mathbf{x}}(\boldsymbol{\xi}^T \mathbf{x}) = \log \left\langle \exp(\boldsymbol{\xi}^T \mathbf{x}) \right\rangle \; , \tag{A.6}$$

where $\left\langle \cdot \right\rangle$ denotes the expectation, and $M_{\mathbf{x}}$ is called the *moment generating function*. Because the expectation occurs inside the logarithm cumulants of mixtures are not additive, but moments are:

$$M_{\mathbf{x}}(\boldsymbol{\xi}^T \mathbf{x}) = \sum_{n=1}^{N} \alpha_n M_{\mathbf{x}_n}(\boldsymbol{\xi}^T \mathbf{x}_n) \; . \tag{A.7}$$

Thus, for the moments, the coefficients of the Taylor expansion of (A.7), holds

$$\begin{aligned}
\mu^i &= \left\langle x_i \right\rangle = \sum_{n=1}^{N} \alpha_n \left\langle x_{n,i} \right\rangle = \sum_{n=1}^{N} \alpha_n \mu_n^i \; , \\
\mu^{ij} &= \left\langle x_i x_j \right\rangle = \sum_{n=1}^{N} \alpha_n \left\langle x_{n,i} x_{n,j} \right\rangle = \sum_{n=1}^{N} \alpha_n \mu_n^{ij} \; , \\
\mu^{ijk} &= \left\langle x_i x_j x_k \right\rangle = \sum_{n=1}^{N} \alpha_n \left\langle x_{n,i} x_{n,j} x_{n,k} \right\rangle = \sum_{n=1}^{N} \alpha_n \mu_n^{ijk} \; , \\
&\dots
\end{aligned}$$

Cumulants can be expressed in terms of moments. Be $\Upsilon = \{\upsilon_1, \dots, \upsilon_\nu\}$ a partition of p indices into ν non-empty blocks. Υ_p is the partition into p blocks and $\kappa(\Upsilon_p) = \kappa^{i_1,\dots,i_p}$ is the cumulant of order p. In this notation the relation between cumulants and moments is

$$\kappa(\Upsilon_p) = \sum_{\Upsilon} (-1)^{\nu-1} (\nu - 1)! \, \mu(\upsilon_1) \dots \mu(\upsilon_\nu) \; , \tag{A.8}$$

where the summation goes over all possible partitions Υ of p indices. In index notation the first four cumulants in terms of moments are

$$\begin{aligned}
\kappa^i &= \mu^i \\
\kappa^{i,j} &= \mu^{ij} - \mu^i \mu^j \\
\kappa^{i,j,k} &= \mu^{ijk} - \mu^i \mu^{jk}[3] + 2\mu^i \mu^j \mu^k \\
\kappa^{i,j,k,l} &= \mu^{ijkl} - \mu^i \mu^{jkl}[4] - \mu^{ij} \mu^{kl}[3] + 2\mu^i \mu^j \mu^{kl}[6] - 6\mu^i \mu^j \mu^k \mu^l \; .
\end{aligned}$$

The numbers in brackets are a common short notation for terms of equal structure, e.g. $\mu^i \mu^{jk}[3] = \mu^i \mu^{jk} + \mu^j \mu^{ik} + \mu^k \mu^{ij}$.

The cumulants in terms of the mixture components are given by

$$\kappa(\Upsilon_p) = \sum_{\Upsilon}(-1)^{\nu-1}(\nu-1)! \left(\sum_{n=1}^{N} \alpha_n \mu_n(v_1)\right) \dots \left(\sum_{n=1}^{N} \alpha_n \mu_n(v_\nu)\right) . \tag{A.9}$$

Obviously, if we can provide that in every partition Υ for all but at most one block (which be the first one, without loss of generality) the moments of the mixture components are equal,

$$\mu_n(v_i) = \mu(v_i) \quad \text{for all} \quad n \in [1, N] \, , \; i \in [2, \nu] \, , \tag{A.10}$$

then the summation over the mixture components can be taken outside the summation over all partitions and it holds,

$$\kappa(\Upsilon_p) = \sum_{\Upsilon}(-1)^{\nu-1}(\nu-1)! \sum_{n=1}^{N} \alpha_n \mu_n(v_1) \dots \mu(v_\nu) = \sum_{n=1}^{N} \alpha_n \kappa_n(\Upsilon_p) \, . \tag{A.11}$$

Equation (A.10) constitutes a sufficient condition for independence of $\mathbf{x}$ up to cumulants of order p. It is obviously fulfilled if all moments μ_n up to order $\lceil p/2 \rceil$ are identical in all mixture components, because at most one block of size larger or equal $p/2$ can be contained in any partition of p indices.

Independence up to a certain order p is sufficient for many applications. For example, a number of ICA algorithms make use of the fourth order cumulants alone (Cardoso and Souloumiac, 1996; Hyvärinen and Oja, 1997). Thus, for fourth order independence of a mixture it is sufficient that the mixture components are independent (at least up to fourth order) and equal in mean and variance.

Appendix B

Reconstruction error

The reconstruction error *re* is a measure for the success of a source separation. It compares the estimated demixing matrix $\mathbf{W}$ with the inverse of the original mixing matrix $\mathbf{A}$ with respect to the indeterminacies: scalings and permutations. It is always nonnegative and equals zero if, and only if $\mathbf{Q} = \mathbf{WA}$ is a nonsingular permutation matrix. This is the case when for every row of $\mathbf{Q}$ exactly one element is different from zero and the rows of $\mathbf{Q}$ are orthogonal, i.e. $\mathbf{QQ}^T$ is a diagonal matrix. The reconstruction error is the sum of measures for both aspects

$$\begin{aligned} re &:= 2\sum_{i=1}^{N}\log\sum_{j=1}^{N}\mathbf{Q}_{ij}^2 - \sum_{i=1}^{N}\log\sum_{j=1}^{N}\mathbf{Q}_{ij}^4 \quad + \quad \sum_{i=1}^{N}\log\sum_{j=1}^{N}\mathbf{Q}_{ij}^2 - \log\,\det\mathbf{QQ}^T \\ &= 3\sum_{i=1}^{N}\log\sum_{j=1}^{N}\mathbf{Q}_{ij}^2 - \sum_{i=1}^{N}\log\sum_{j=1}^{N}\mathbf{Q}_{ij}^4 - \log\,\det\mathbf{QQ}^T\,. \end{aligned} \tag{B.1}$$

Appendix C

Theorems

Theorem C.1 *For any $N \times 1$ vector $\mathbf{v}$ and any symmetric, positive definite $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$ the quantity*

$$q(\mathbf{A}) = \frac{\mathbf{v}^T\mathbf{A}\mathbf{B}^{-1}\mathbf{A}\mathbf{v}}{(\mathbf{v}^T\mathbf{A}\mathbf{v})^2} \tag{C.1}$$

has a global (but not unique) minimum in the space of $\mathbf{A}$ at $\mathbf{A} = \mathbf{B}$.

Proof: q is invariant under multiplications of non-zero scalars c with $\mathbf{A}$,

$$q(\mathbf{A}) = q(c\mathbf{A}) \,,$$

and so is the minimum. Without loss of generality be $\mathbf{A}' = c\mathbf{A}$ such that

$$\mathbf{v}^T\mathbf{A}'\mathbf{v} = \mathbf{v}^T\mathbf{B}\mathbf{v} \,. \tag{C.2}$$

Because $\mathbf{A}$ and $\mathbf{B}$ are positive definite, there always exist a non-negative scalar

$$c = \frac{\mathbf{v}^T\mathbf{A}\mathbf{v}}{\mathbf{v}^T\mathbf{B}\mathbf{v}} \,,$$

for which the constraint is fulfilled. Under the constraint (C.2) the denominator of equation (C.1) is constant, and it is sufficient to minimize the numerator, using Lagrange multipliers.

$$L = \mathbf{v}^T\mathbf{A}'\mathbf{B}^{-1}\mathbf{A}'\mathbf{v} + \lambda(\mathbf{v}^T\mathbf{A}'\mathbf{v} - \mathbf{v}^T\mathbf{B}\mathbf{v}) \tag{C.3}$$

At the stationary point the gradient of L w.r.t. $\mathbf{A}'$ and λ vanishes.

$$\frac{\partial L}{\partial \mathbf{A}} = 2\mathbf{v}\mathbf{v}^T\mathbf{A}'\mathbf{B}^{-1} + \lambda\mathbf{v}\mathbf{v}^T, \qquad \text{and} \quad \frac{\partial L}{\partial \lambda} = \mathbf{v}^T\mathbf{A}'\mathbf{v} - \mathbf{v}^T\mathbf{B}\mathbf{v} \tag{C.4}$$

This is clearly the case for $\mathbf{A}' = \mathbf{B}$ and $\lambda = -2$. Because the unconstrained optimization problem is convex for positive definite $\mathbf{A}', \mathbf{B}$, and the constraint is linear, $\mathbf{A} = c^{-1}\mathbf{B}$ is a global minimum, as well as $\mathbf{A} = \mathbf{B}$.

Box

Theorem C.2 *For any vector $\mathbf{x} \neq 0$ the function*

$$f(p) = \left(\sum_n |x_n|^p\right)^{\frac{1}{p}} \tag{C.5}$$

is monotonically decreasing for all $p > 0$. If $\mathbf{x}$ contains more than one element different from zero, it is strictly monotonically decreasing. Otherwise it is constant.

Proof: Consider the derivative of f w.r.t. p

$$\frac{\mathrm{d}}{\mathrm{d}p}\left(\sum_n |x_n|^p\right)^{\frac{1}{p}} = \left(\sum_n |x_n|^p\right)^{\frac{1}{p}} \left(-\frac{1}{p^2}\log\left(\sum_n |x_n|^p\right) + \frac{\frac{1}{p}\sum_n |x_n|^p \log|x_n|}{(\sum_n |x_n|^p)}\right) , \tag{C.6}$$

where the sum is over all $\{n : x_n \neq 0\}$. It is less or equal zero if, and only if

$$\begin{aligned} &\frac{1}{p}\log\left(\sum_n |x_n|^p\right) \geqslant \frac{\sum_n |x_n|^p \log|x_n|}{(\sum_n |x_n|^p)} \\ \Longleftrightarrow &\sum_n |x_n|^p \log\left(\sum_m |x_m|^p\right) \geqslant \sum_n |x_n|^p \log(|x_n|^p) , \end{aligned} \tag{C.7}$$

which holds, because $\sum_m |x_m|^p \geqslant |x_n|^p$ for all n. The equivalence in equation (C.7) is also valid for the equality alone. It is achieved if, and only if, the set $\{n : x_n \neq 0\}$ contains not more than one element. Otherwise equation (C.7) holds strictly.

□

Appendix D

Solution of the Lagrange equation (3.6)

The derivative of the Lagrangian, equation (3.6),

$$L = \sum_{\tau=-T_l}^{T_l} l_\tau^2 + \alpha \sum_{t=-T_f}^{T_f} f_t^2 \quad + \lambda(l_0 - 1)\,,$$

w.r.t. f and λ yields the gradient of L.

$$\frac{\partial l_\tau}{\partial f_t} = \xi_{t+\tau}$$

$$\frac{\partial L}{\partial f_t} = 2 \sum_{\tau=-T_l}^{T_l} l_\tau \xi_{t+\tau} + 2\alpha f_t + \lambda \xi_t \tag{D.1}$$

$$\frac{\partial L}{\partial \lambda} = l_0 - 1\,. \tag{D.2}$$

For the Hessian we need the second derivatives

$$\frac{\partial^2 L}{\partial f_{t_1} \partial f_{t_2}} = 2 \textstyle\sum_{\tau=-T_l}^{T_l} \xi_{t_1+\tau} \xi_{t_2+\tau} + \delta_{t_1-t_2}\, 2\alpha \tag{D.3}$$

$$= 2(\xi \star \xi)_{t_1-t_2} + \delta_{t_1-t_2}\, 2\alpha \tag{D.4}$$

$$\frac{\partial^2 L}{\partial f_t \partial \lambda} = \xi_t\,, \tag{D.5}$$

To summarize the derivatives in matrix or vector notation we introduce a time vector $\mathbf{t} = (T_f, \ldots, -T_f)^T$ with . Now we can write the Hessian as

$$\mathbf{H}' = \begin{pmatrix} \mathbf{H} & \mathbf{h} \\ \mathbf{h}^T & 0 \end{pmatrix}\,, \tag{D.6}$$

where

$$H_{ij} = \frac{\partial^2 L}{\partial f_{t_i} \partial f_{t_j}} \quad \text{and} \quad h_i = \frac{\partial^2 L}{\partial f_{t_i} \partial \lambda} \quad \text{and} \quad t_i = T_f - i + 1\,.$$

For the solution of the optimization problem, we need to find a stationary point $(\bar{f}, \lambda)$ of the gradient. The Lagrangian contains no term of higher order than two. Thus, the Taylor expansion of the gradient has only linear terms:

$$0 = \begin{pmatrix} \frac{\partial L}{\partial \bar{f}} \\ \frac{\partial L}{\partial \lambda} \end{pmatrix}_{\bar{f}^*, \lambda^*} = \begin{pmatrix} \frac{\partial L}{\partial \bar{f}} \\ \frac{\partial L}{\partial \lambda} \end{pmatrix}_{\bar{f}=0, \lambda=0} + \mathbf{H}' \cdot \begin{pmatrix} \bar{f}^* \\ \lambda^* \end{pmatrix}\,. \tag{D.7}$$

The stationary point can be easily found with

$$\begin{pmatrix} \bar{f}^* \\ \lambda^* \end{pmatrix} = -\mathbf{H'}^{-1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix}. \tag{D.8}$$

Because of the partitioned structure of $\mathbf{H'}$, its inverse can be written as (Lüthkepol, 1996, 3.5.3.(1))

$$\mathbf{H'}^{-1} = \begin{pmatrix} \mathbf{H}^{-1} - \mathbf{H}^{-1}\mathbf{h}(\mathbf{h}^T\mathbf{H}^{-1}\mathbf{h})^{-1}\mathbf{h}^T\mathbf{H}^{-1} & \mathbf{H}^{-1}\mathbf{h}(\mathbf{h}^T\mathbf{H}^{-1}\mathbf{h})^{-1} \\ (\mathbf{h}^T\mathbf{H}^{-1}\mathbf{h})^{-1}\mathbf{h}^T\mathbf{H}^{-1} & -(\mathbf{h}^T\mathbf{H}^{-1}\mathbf{h})^{-1} \end{pmatrix}. \tag{D.9}$$

The optimal filter $\bar{f}^*$ and the Lagrange multiplier λ^* at the stationary point are contained in the last column of $\mathbf{H'}^{-1}$. With $\mathbf{h} = \bar{\xi}$,

$$\bar{f}^* = \frac{\mathbf{H}^{-1}\bar{\xi}}{\bar{\xi}^T\mathbf{H}^{-1}\bar{\xi}} \qquad \text{and} \quad \lambda^* = \frac{-1}{\bar{\xi}^T\mathbf{H}^{-1}\bar{\xi}} \tag{D.10}$$

holds.

Appendix E

Files on the CD

floor_+2.avi

This file is a video recorded with the *BumbleBee* two lens camera in the floor of an office building. The camera was mounted on a cart while it was pushed over the floor. The video consists of 15,000 frames at 16 fps and has a resolution of 640 × 240 pixels, gray scale.

floor_-1.avi

This video was recorded with the same setup as `floor_+2.avi`, but in the basement floor of the same building.

The videos `floor_+2.avi` and `floor_-1.avi` were used as training and test videos in the experiments presented in this thesis.

filt_8_11_12.avi

This file is the left half of the video `floor+2.avi` with the responses of the multi-channel filters of figure 3.15.A, f^8 (red), f^{11} (green), and f^{12} (blue), that exceed a threshold value of ± 10 (the responses had unit variance).

filt_3_4_5_7_14.avi

This file is the left half of the video `floor+2.avi` with the responses of the multi-channel filters of figure 3.15.A, f^3 (red), f^4 (green), f^5 (blue), f^7 (cyan), and f^{14} (magenta), that exceed a threshold value of ± 10 (the responses had unit variance).

video1.avi

This file contains three clips of the video used for the experiment in section 5.3. You see those parts, that are shown in the closeup, figure 5.4.B . The signals are color coded, the first one red, the second one green, and the third one blue. If no signal is active, the video is black-and-white. If a signal gets active the white value changes to the corresponding color.

res_floor+2.avi, res_floor-1.avi

These videos are the complete videos the example frames of figure 5.11.C and 5.12.C were taken from.

Bibliography

FastICA Matlab toolbox, v2.5. http://www.cis.hut.fi/projects/ica/fastica/. online.

S. Amari. Neural learning in structured parameter spaces - "Nature"ural Riemannian gradient. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, 1996.

S. Amari. Natural gradient works efficiently in learnig. *Neural Computation*, 10:251–276, 1998.

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, 1995.

C. Andrieu, E. Barat, and A. Doucet. Baysian deconvolution of noisy filtered point processes. *IEEE Transactions on Signal Processing*, 49(1), January 2001.

H. Attias and C. E. Schreiner. Blind source separation and deconvolution: The dynamic component analysis algorithm. *Neural Comput.*, 10:1373–1424, 1998.

F. R. Bach and M. I. Jordan. Kernel indepedendent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

S. H. Baloch, H. Krim, and M. G. Genton. Robust independent component analysis. In *IEEE Workshop Statistical SIgnal Processing*, 2005. http://www4.ncsu.edu/šhbaloch/publications/RICA_SSP05.pdf.

A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Res.*, 37:3327–3338, 1997.

A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2): 434–444, 1997.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

G. G. Blasdel and G. Salama. Voltage-sensitive dyes reveal a modular organization in monkey striate cortex. *Nature*, 321:579–585, 1986.

H. Brauer. Wahrscheinlichkeitstheorie, 2002.

A. Bray and D. Martinez. Kernel-based extraction of slow features: Complex cells learn disparity and translation invariance from natural images. In *Advances in Neural Information Processing Systems*, volume 15, pages 253–260, 2002.

J.-F. Cardoso. Blind signal separation: statistical principles. In R.-W. Liu and L. Tong, editors, *Proceedings of the IEEE, special issue on blind identification and estimation*. 1998.

J.-F. Cardoso, S. Bose, and B. Friedlander. On optimal source separation based on second and fourth order cumulants. In *Proc. IEEE Workshop on SSAP, Corfou, Greece*, 1996.

J.-F. Cardoso and A. Souloumiac. Blind beam forming for non gaussian signals. *IEE Proceedings-F*, 140(6):362–370, dec. 1993.

J.-F. Cardoso and A. Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 17(1):161–164, 1996.

J. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young"decomposition. *Psychometrika*, 35:283–319, 1970.

C.-C. Chang and C.-J. Lin. *LIBSVM – A Library for Support Vector Machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm/, April 2003.

S. Choi and A. Cichocki. Blind separation of nonstationary and temporally correlated sources from noisy mixtures. In *Proceedings of the IEEE International Workshop on Neural Networks for Signal Processing*, pages 405–414, 2000.

P. Comon and B. Mourrain. Decomposition of quantics in sums of powers of linear forms. *Signal Processing*, 53(2):96–107, Sept 1996.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

R. O. Duda, E. Hart, and G. D. Stork. *Pattern Classification*. Wiley, 2nd edition, 2001.

R. Fabian-Fine, P. Verstreken, P. Hiesinger, J. Horne, R. Kostyleva, H. Bellen, and I. Meinertzhagen. Endophilin acts after synaptic vesicle fission in drosophila photoreceptor terminals. *J. Neurosci.*, 2003. (in press).

M. S. Fee, P. P. Mitra, and D. Kleinfeld. Automatic sorting of multiple unit neural signals in the presence of anisotropic non-gaussian variability. *Journal of Neuroscience Method*, (69): 175–188, 1996.

M. Frigo and S. Johnson. *FFTW User Manual*. `http://www.fftw.org/fftw3.pdf`, 2003. online.

J. Fröhlich, S. Novikov, and D. Ruelle, editors. *Dynamical Systems, Ergodic Theory and Applications*, chapter I.1. Springer-Verlag, Berlin Heidelber New York, 1999.

C. Fyfe and P. Lai. ICA using kernel canonical correlation analysis. In *In Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, 2000.

C. Gray, P. Maldonado, M. Wilson, and B. McNaughton. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *Journal of Neuroscience Methods*, 63(1–2):43–54, December 1995.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15(5):1089–1124, 2003.

R. Harshman. Foundations of the parafac procedure: Model and conditions for an "explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

L. O. Harvey, Jr. The critical operating characteristic and the evaluation of expert judgment. *Organizational Behavior and Human Decision Processes*, 53(2):229–251, 1992.

S. Haykin. *Neural Networks*. Prentice Hall, Inc., 1999.

J. Hurri and A. Hyvärinen. Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, 15(3):663–691, 2003.

A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

A. Hyvärinen and E. Oja. A fast fixed point algorithm for independent component analysis. *Neural Comput.*, 9(7):1483–1492, 1997.

M. Joho and K. Rahbar. Joint diagonalization of correlation matrices by using newton methods with application to blind signal separation. In *Sensor Array and Multichannel Signal Processing Workshop Proceedings*, pages 403–407, Aug. 2002.

I. T. Jolliffe. *Principal Component Analysis*. SpringerVerlag, New York, 1986.

E. Kandel, J. Schwarz, and T. Jessel, editors. *Principles of Neural Science*. Appleton & Lange, third edition edition, 1991.

J. Kingman and S. Taylor. *Introduction to measure and probability*. Cambridge University Press, Cambridge, England, 1966.

L. D. Lathauwer, B. D. Moor, and J. Vandewalle. Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition. *SIAM Journal on Matrix Analysis and Applications*, 26(2):295–327, 2004.

Y. LeCun and C. Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/mnist/`.

T.-W. Lee, M. Lewicki, M. Girolami, and T. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, April 1999.

M. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):53–78, Nov 1998.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

H. Lüthkepol. *Handook of Matrices*. John Wiley & Sons Ltd., 1996.

D. Malonek and A. Grinvald. Interactions between electrical activity and cortical microcirculation revealed by imaging spectroscopy: implications for functional brain mapping. *Science*, 272:551–554, 1996.

K. Matsuoka, M. Ohya, and M. Kawamoto. A neural net for blind separation of nonstationary signals. *Neural Networks*, 8(3):411–419, 1995.

P. McCullagh. *Tensor Methods in Statistics*, chapter 1,2,4. Chapman and Hall, 1987.

B. L. McNaughton, J. O'Keefe, and C. A. Barnes. The stereotrode: a new technique for simultaneous isolation of several single units in the central nervous system from multiple unit records. *Journal of Neuroscience Methods*, 8:391–397, 1983.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London* **A**, 209:415–446, 1909.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Phys. Rev. Lett.*, 72:3634–3637, 1994.

J. Moors. A quantile alternative for kurtosis. *The Statistician*, 37(1):25–32, 1988.

D. Nguyen, L. Frank, and E. Brown. An application of reversible-jump markov chain monte carlo to spike classification of multi-unit extracellular recordings. *Network: Computation in Neural Systems*, (14):61–82, 2003.

B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

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.

J. S. Pezaris, M. Sahani, and R. A. Andersen. Tetrodes for monkeys. In *CNS*95*, July 12-16 1995.

D. T. Pham. Joint approximate diagonalization of positive definite hermitian matrices. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1136–1152, 2001.

D. T. Pham and P. Garat. Blind separation of mixtures of independent sources through a quasi maximum likelihood approach. *IEEE Trans. Signal Processing*, 45(7):1712–1725, 1997.

G. Pipa, E. Rodriguez, E. Staedtler, S. Gruen, and M. Munk. Neuronal dynamics in monkey prefrontal cortex during visual short-term memory: temporal patterns of spike synchronization. In *Society for Neuroscience Abstracts*, Washington, DC, 2003. Society for Neuroscience. Online.

C. Pouzat, M. Delescluse, P. Viot, and J. Diebolt. Improved spike-sorting by modelling firing statistics and burst-dependent spike amplitude attenuation: A markov chain monte carlo approach. *Journal of Neurophysiology*, (91):2910–2928, 2004.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd. edition, 1992.

S. P. Rebrik, B. D. Wright, A. A. Edmondi, and K. D. Miller. Cross channel correlations in tetrode recordings: implications for spike-sorting. In *Proceedings of Computation and Neural Systems Meeting*, Big Sky Montana, 1999.

M. L. Reece and J. O'Keefe. The tetrode: A new technique for multi-unit extracellular recording. *Society of Neuroscience Abstracts*, 15:1250, 1989.

W. M. Roberts and D. K. Hartline. Separation of multi-unit nerve impulse trains by a multi-channel linear filter algorithm. *Brain Research*, 94:141–149, 1975.

M. Sahani. *Latent Variable Models for Neural Data Analysis*. PhD thesis, California Institute of Technology, Pasadena, California, 1999.

S.Amari and H.Nagaoka. *Methods of Information Geometry*. Translations of Mathematical Monographs. American Mathmatical Society, January 2001.

T. Sanger. Optimal unsupervised learning in a singlelayer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, May 2000. URL `citeseer.nj.nec.com/276248.html`.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, 2002.

M. Scholz, R. Vollgraf, T. Bartels, A. Maye, I. Meinertzhagen, and K. Obermayer. Computer-based segmentation and structural analysis of electron micrographs from drosophila photoreceptor mutants. In *Soc. Neurosci. Abstr.*, number 29, Washington DC, 2003. online.

H. Schöner, M. Stetter, I. Schießl, J. Mayhew, J. Lund, N. McLoughlin, , and K. Obermayer. Application of blind separation of sources to optical recording of brain activity. In T. L. S.A. Solla and K.-R. Mller, editors, *Advances in Neural Information Processing Systems NIPS 12*, pages 949–955. MIT Press, 2000.

H. Schöner, M. Stetter, I. Schießl, J. Mayhew, J. Lund, N. McLoughlin, and K. Obermayer. Application of blind separation of sources to optical recording of brain activity. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems NIPS 12*, pages 949–955. MIT Press, 2000. In press.

S. Shoham, M. Fellows, and R. A. Normann. Robust, automatic spike sorting using mixtures of multivariate *t*-distributions. *Journal of Neuroscience Methods*, (127):111–122, 2003.

A. Smola, O. Mangasarian, and B. Schölkopf. Sparse kernel feature analysis. Technical report 99-03, University of Wisconsin, Data Mining Institute, Madison, 1999.

J. Stone. Learning visual parameters using spatiotemporal smoothness constraints. *Neural Computation*, 8(7):1463–1492, 1996.

R. Stowers and T. Schwarz. A genetic method for generating drosophila eyes composed exclusively of mitotic clones of a single genotype. *Genetics*, (152):1631–1639, 1999.

S. Takahashi, Y. Anzai, and Y. Sakurai. Automatic sorting for multi-neural activity recorded with tetrodes in the presense of overlapping spikes. *Journal of Neurophysiology*, (89): 2245–2258, 2003.

Thomas RECORDING Gmbh. *Heptode (7-cores Multifiber Electrode) – Datasheet*. Giessen, GERMANY, 2002. http://www.thomasrecording.com/pdf/HeptodeDataSheet.pdf.

M. Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems*, number 13. MIT Press, 2001.

A. van der Veen. Joint diagonalization via subspace fitting techniques. In *Proc. IEEE ICASSP*, Salt Lake City (UT), May 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. 1995.

R. Vollgraf, M. Munk, and K. Obermayer. Spike sorting of multi-electrode recordings based on amplitude ratios. In K. Krieglstein and H. Zimmermann, editors, *30th Göttinger Neurobiology Report*, 2005a. CD-ROM.

R. Vollgraf, M. Munk, and K. Obermayer. Spike sorting of multi-site electrode recordings. *Network – Computation in Neural Systems*, 16:85–113, 2005b.

R. Vollgraf and K. Obermayer. Multi dimensional ICA to separate correlated sources. In *Advances in Neural Information Processing Systems 14*, pages 993–1000, Cambrdige, Massachusetts, 2002. MIT Press.

R. Vollgraf and K. Obermayer. Improved optimal linear filters for the discrimination of multi-channel waveform templates for spike-sorting applications. *IEEE Signal Processing Letters*, 13:121–124, 2006a.

R. Vollgraf and K. Obermayer. Quadratic optimization for simultaneous matrix diagonalization. *IEEE Transactions on Signal Processing*, 2006b. in press.

R. Vollgraf and K. Obermayer. Sparse optimization for second order kernel methods. In *IJCNN 2006 Conference Proceedings*, 2006c. in press.

R. Vollgraf, I. Schießl, and K. Obermayer. Blind source separation of single components from linear mixtures. In *Proc. Int. Conf. on Artificial Neural Networks - ICANN 01*, pages 509–514, 2001.

R. Vollgraf, M. Scholz, I. Meinertzhagen, and K. Obermayer. Nonlinear filtering of electron micrographs by means of support vector regression. In *Advances in Neural Information Processing Systems*, volume 16, pages 717–724, Cambridge, Massachusetts, 2004. MIT Press.

R. Vollgraf, M. Stetter, and K. Obermayer. Convolutive decorrelation procedures for blind source separation. In P. Pajunen and J. Karhunen, editors, *Proceedings of the 2. International Workshop on ICA, Helsinki*, pages 515–520, 2000.

P. Walters. *An Introduction to Ergodic Theory*. Springer Verlag New York Inc., 1982.

M. Wax and J. Sheinvald. A least-squares approach to joint diagonalization. *IEEE Signal Processing Letters*, 4(2), feb. 1997.

L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Transactions on Signal Processing*, 50(7):1545–1553, July 2002.

M. Zibulevski and B. A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 12(3):863–882, April 2001.

A. Ziehe, P. Laskov, K.-R. Mueller, and G. Nolte. A linear least-squares algorithm for joint diagonalization. pages 469–474, Nara, Japan, Apr. 2003.

A. Ziehe, P. Laskov, G. Nolte, and K.-R. Müller. A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation. *Journal of Machine Learning Research*, 5:777–800, 2004.

A. Ziehe and K.-R. Müller. TDSEP – an efficient algorithm for blind source separation using time structure. In *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 675–680, Berlin, 1998. Springer-Verlag.

A. Ziehe, G. Nolte, G. Curio, and K.-R. Müller. Ofi: Optimal filtering algorithms for source separation. In P. Pajunen and J. Karhunen, editors, *Proceedings of the 2. International Workshop on ICA, Helsinki*, pages 127–132, 2000.