論文 / 著書情報
Article / Book Information

題目(和文)	マルチウェイデータ解析のための特徴抽出および識別法
Title(English)	Feature Extraction and Classification Methods for Multi-way Data Analysis
著者(和文)	横田達也
Author(English)	Tatsuya Yokota
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第9466号, 授与年月日:2014年3月26日, 学位の種別:課程博士, 審査員:山下　幸彦,高田　潤一,阿部　直也,花岡　伸也,杉山　将,Andrzej Cichocki
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第9466号, Conferred date:2014/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Feature Extraction and Classification Methods for Multi-way Data Analysis

Tatsuya YOKOTA

Department of International Development Engineering
Graduate School of Science and Engineering
Tokyo Institute of Technology

Feburuary 2014

Abstract

The world is filled with data which have various types such as numeric data, non-numeric data, continuous valued data, discrete valued data, scalar data, vector data, matrix data, and higher-dimensional array data. Continuous valued and multi-dimensional array data is denoted by the "multi-way data" which can be processed by the computers. Thus, the multi-way data includes a scalar, a matrix, and a higher-dimensional array of continuous values. Techniques of multi-way data analysis play important roles in wide research fields such as audio processing, image processing, bioinformatics, chemometrics, and brain science.

In this thesis, after the techniques of feature extraction and classification methods for multi-way data analysis are summarized; new criteria and solving algorithms are proposed for them, and their advantages are demonstrated by experiments. Matrix/tensor decomposition has been widely used for visualization, clustering, blind source separation, and dimensionality reduction of multi-way data. We often impose some constraints such as orthogonality, sparsity, and nonnegativity based on the characteristics of target data. Techniques for the constraints such as singular value decomposition (SVD), penalized matrix decomposition (PMD), and nonnegative matrix factorization (NMF) have been well studied.

Function approximation in NMF is a technique to represent nonnegative feature vectors by linear combinations of basis functions. For example, if we choose smooth basis functions, then smooth feature vectors can be obtained. Zdunek proposed this technique as the GRBF-NMF method which uses the Gaussian radial basis functions for function approximation. However, its algorithm, in which the QP optimization and the active-set algorithm run alternately and iteratively, is so slow that it can not be applied to large-scale problems. Then, I propose a new fast algorithm for the function approximation by NMF, and choose more effective basis functions for function approximation. Furthermore, I extend this method to the nonnegative Tucker decomposition and the nonnegative canonical polyadic (CP) decomposition.

The common and individual feature extraction is a very important concept for data analysis because real world data always have some common and individual features. For examples all human faces have two eyes, two ears, a nose, and a mouse (common feature); but, their positions and shapes are different as individual personalities (individual feature). However, existing techniques for it are not efficient very well, and their techniques have been developed only for matrices. Then, I propose a tensor based common and individual feature extraction method in CP and Tucker model, and impose orthogonality,

sparsity, and nonnegativity constraints to both models.

Classification techniques are used for various objectives such as brain computer interface (BCI), fingerprinting identification, hand-written character recognition, and face recognition. In recent years, the support vector machine (SVM) has been the most popular classifier; however, SVM does not always provide the best performance. The criterion of SVM consists of the hinge-loss minimization and a regularization. In this research, I propose two novel weighted regularization methods for a variety of applications. Furthermore, I apply these regularization methods to the conventional SVM.

The Fisher discriminant analysis (FDA) is another famous method for classification. However, FDA does give an optimal projection only for Gaussian distributions with equal covariance matrices. In other words, FDA is not optimal in the case of heteroscedastic Gaussian distributions. In this research, I propose a novel criterion for FDA including a correction term based on the Bhattacharyya distance which is closely related to classification rate. Furthermore, the Chernoff distance based criterion and its kernelized version are proposed as its extensions.

The final proposition of this thesis is a new criterion of classifier based on the maximum a posteriori (MAP) estimation for a binary problem without estimating the posterior probability, called the quadratically constrained maximum a posteriori (QCMAP) estimation. The QCMAP consists of the maximization of the expectation of a cost function, which is derived from the maximum a posteriori probability, and a quadratic constraint. This criterion is highly general since its forms include least squares regressions (LSRs) and a SVM. I propose several efficient classifiers from the criterion by selecting various weight functions.

Finally, I show the experimental results of all proposed methods to demonstrate their advantages for the feature extraction and the pattern classification. In the experiments, the proposed smooth NMF/NTF algorithms using function approximation are applied to reconstruction of two and three dimensional array data, blind source separation, part-based representation to compare with other NMF methods (unconstrained, sparse and smooth NMF algorithms); the proposed common and individual feature extraction algorithms based on Tucker and CP decomposition models are applied to reconstruction and multi-way blind source separation; the proposed regularized SVM, Chernoff based FDA, and QCMAP classifiers are applied to several toy well-known benchmarks and real-world brain computer interface (BCI) problems to compare with other classification methods.

Acknowledgements

I would like to express my appreciation and gratitude to all those who supported me to complete my thesis. First, I thank to my advisers Prof. Yukihiko Yamashita and Prof. Andrzej Cichocki for giving me many advisement and supports. They have provided an environment conducive to leaning and research quality. Prof. Yamashita has introduced me to the field of interesting research on pattern recognition and machine learning from a mathematical point of view. Prof. Cichocki has introduced me to the field of interesting research on matrix/tensor based data analysis and common and individual feature extraction. Next, I also thank to Prof. Toru Wakahara and Dr. Hitoshi Sakano for giving me many advisement. Dr. Rafal Zdunek discussed with me in a research topic. Prof. Yoshikazu Washizawa supported me to create the environment of research. Ms. Wakako Honjo supported me for clerical work. I also thank to my colleagues of Yamashita Laboratory and Cichocki Laboratory for their encouragements and influence. Finally, I would like to acknowledge the support of my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The world is filled with data which have various types such as numeric data, non-numeric data, continuous valued data, discrete valued data, scalar data, vector data, matrix data, and higher-dimensional array data. Continuous valued and multi-dimensional array data are denoted by the "multi-way data" which can be processed by the computers. Thus, the multi-way data include a scalar, a matrix, and a higher-dimensional array of continuous values.

Techniques of multi-way data analysis play important roles in science and engineering such as chemometric data analysis, economic data analysis, psychological data analysis, audio processing, image processing, text mining, spectrometry analysis, bioinformatics, and brain signal analysis. In a more mathematical point of view, the multi-way data stand for N-dimensional arrays including a scalar ($N = 0$), a vector ($N = 1$), a matrix ($N = 2$), and a higher-dimensional array ($N \geq 3$). We call also an N-way array the "Nd-tensor". For example, an image data is a 2d-tensor and a video data is a 3d-tensor. These data are obtained from various measurement devices such as a camera for images/videos, a microphone for audio signals, a scanner for fingerprintings, and a MRI (magnetic resonance imaging) device for 3d-images of a brain. When we treat such multi-way data, it is undesirable to unfold a higher-dimensional array to a vector because it destroys the data structure.

One of purposes of multi-way data analysis is to visualize to show extracted key information to the analyst, and main research objective is to provide analysis methods and algorithms which are accurate, robust, and computationally efficient. Since such general analysis methods can be applied to various types of data, the social impacts of data analysis are very wide.

There are many applications of data analysis because there are many kinds of needs for individual subjects such as the brain signal analysis, image/video data analysis, and financial data analysis. In recent years, the multi-way data analysis is closely-watched and critically-needed for the "BIG DATA" analysis in the world. Firstly, I discuss the brain signal analysis. Improvements of brain computer interface (BCI) systems contribute to the nursing care and the rehabilitation. If we come to be able to diagnose Alzheimer disease in an early phase, we can suppress the symptoms to the minimum.

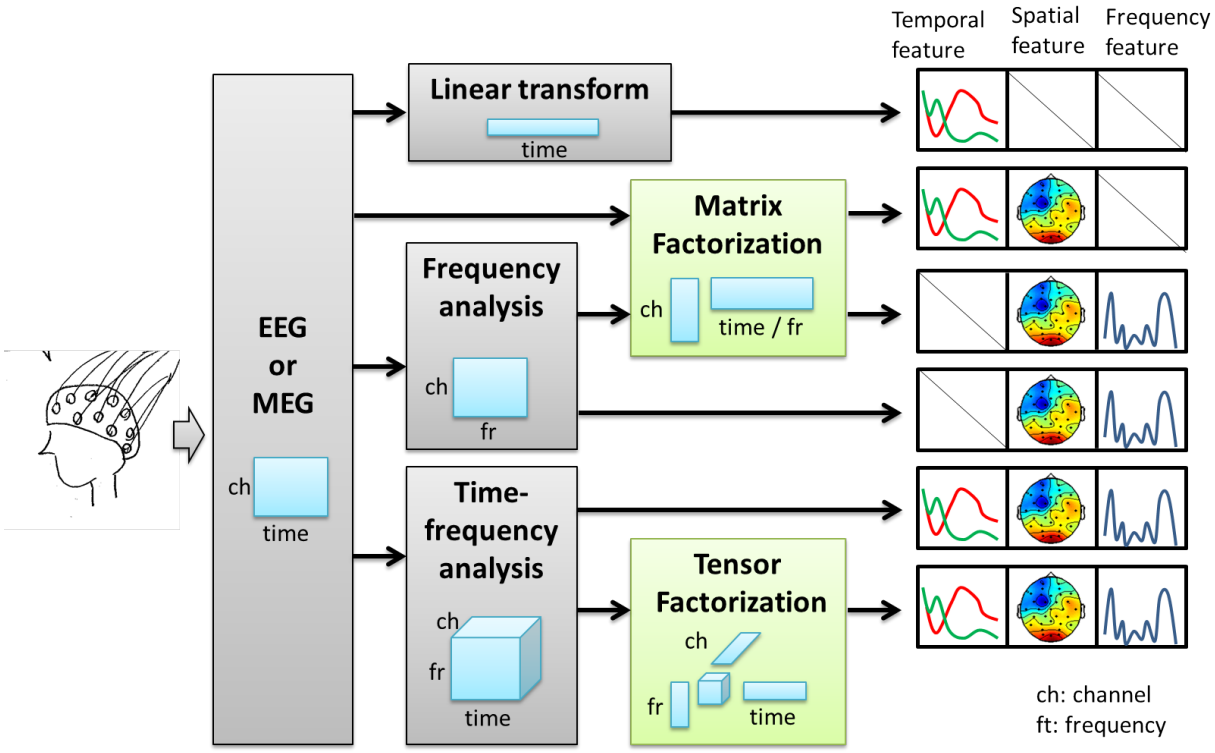


Figure 1.1: Flowchart for visualization

Improvements of brain fingerprinting systems contribute to the identification of the criminals. Secondly, I discuss the image analysis and recognition. Improvements of artificial intelligence for robots contribute to the industrial robots, the cleaning robots, the nursing-care robots, the rescue robots, and so on. Furthermore, improvements of artificial intelligence for security cameras contribute to the fire detection, the human tracking, the identification of criminals, and so on. Thirdly, I discuss the financial data analysis. If we can plan efficient economic strategies by data analysis, it contributes to the economic growth in the society. Precise prediction of stock prices will stabilize stock market. In other subjects, improvements of audio recognition contribute to the audio input systems of car navigation devices, mobile phones, computers, and so on; improvements of techniques of web data mining contribute to companies so that they can know customer's needs; improvements of weather prediction contribute to agriculture and preventing disaster; improvements of psychological data analysis contribute to curing mental disorders.

In this thesis, we focus on the tensor based feature extraction and classification methods.

1.1.1 Motivation for tensor based feature extraction

In this section, I introduce the motivation for tensor based feature extraction by focusing on time-series signals of EEG/MEG. Fig. 1.1 shows a flowchart for visualization of EEG/MEG signals. Temporal, spatial, and frequency features are often focused for this visualization. In this figure, several feature

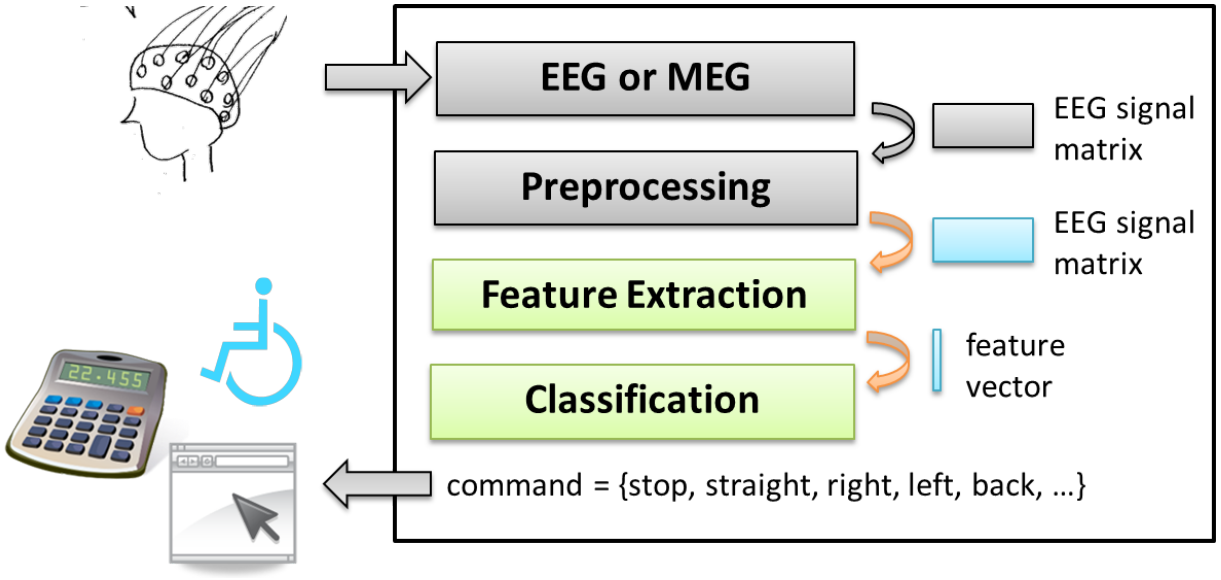


Figure 1.2: Flowchart for BCI

extraction methods are introduced. I explain frequency analysis, time-frequency analysis, and linear transform in Section 2.1, 2.2, 2.3 and 2.4. Matrix and tensor factorizations are explained in Chapter 3, and I discuss these techniques in this thesis, mainly.

From here, I mention about only EEG for simplicity; however almost all methods can be applied to also MEG. An EEG data is given as a $(ch \times time)$-matrix, where ch is the number of channels and $time$ is the number of sample points in time domain. An EEG data stands for a set of sampled voltages of the multiple electronodes on a head while a subject carries out some tasks for a fixed time interval. In order to find some important factor from the EEG data, we need a large number of such samples. Thus the EEG dataset is given as a $(ch \times time \times samples)$-tensor, where I call a multi dimensional (N-way) array as a 'tensor' (Nd-tensor). If we apply some time-frequency analysis technique to this tensor data, the 3d-tensor is transformed to a $(ch \times time \times fr \times samples)$-tensor (4d-tensor). The problem is how to process such a tensor data. One of the methods is to unfold a sample of tensor (e.g., $(ch \times time \times fr)$-tensor) to a vector (e.g., $[(ch) \cdot (time) \cdot (fr)]$-dimensional vector); however the unfolding destroys the data structure. Thus, tensor (including matrix) based feature extraction techniques are necessary for the brain signal processing, and also other multi-way data analysis such as video and fMRI image.

In general, matrix/tensor based feature extraction techniques have been widely studied to analyze various time-series signals, images, videos, physical spectra, and so on. The key-point of the matrix and tensor factorizations is to impose constraints such as orthogonality, sparsity, smoothness, and nonnegativity of factors. For example, EEGs are generally smooth, and frequency power spectrum, images, videos are always nonnegative. Therefore, the researches on algorithms and methods for matrix/tensor based feature extraction are very important. I introduce the details of matrix and tensor factorizations in Chapter 3.

1.1.2 Motivation for pattern classification

In this section, I introduce the motivation for pattern classification by focusing on EEG/MEG signal recognition as a subject. The pattern classification is a technique to assign some patterns into individual appropriate categories, automatically. For example, the EEG/MEG signal recognition is an important technique for BCI and the auto-diagnosis of the brain disease. BCI achieves to control a computer or a machine by using a brain signal without four limbs, for example typing text, inputting number, and controlling a wheel chair. These applications have possibilities of nursing care and welfare. Fig. 1.2 shows a flowchart for the EEG/MEG based BCI. Note that this flowchart is applied not only to BCI, but also to other objectives such as image recognition and text categorization. Preprocessing includes cutout, normalization, filtering, and so on. Feature extraction is to transform a EEG/MEG signal matrix (or other types of data) to a feature vector which includes important information for classification. Finally, classification is to assign a feature vector into a class which implies a command to control devices for BCI. In this thesis, we discuss the techniques of classification, mainly. Please note that the techniques of classification are not only for the brain signals, but also for other various types of patterns such as the handwritten characters, the fingerprintings, and the face images. Thus, the researches of classification are very important. I introduce the details of classification in Chapter 4.

1.1.3 Social impacts

The social impacts of multi-way data analysis are very wide and great. For example, if sufficient accuracy of pattern recognition which includes feature extraction and classification is achieved, the world would be significantly changed. Most of the works which persons have been done would be provided by machines, and persons have come to be dedicated to some more creative works. This technique has many possibilities such as the automated driving of car, the automated cleaning of building, the automated speech inputting, and the automated translation to different languages. Furthermore, if the auto-diagnosis of diseases becomes sufficiently accurate, people in regions where the number of medical doctors is not sufficient would be rescued by such systems.

On the other hand, there would be negative impacts caused by the world changes. Automation of works has a possibility to increase the jobless rate. For example, the automated driving system does away with the need to employ taxi drivers. There are risks that those systems are used for crimes such as the kidnapping, the smuggling, and the terror.

When the researchers publish their researches or new technologies, they should consider not only positive impacts but also negative impacts because their research might change the world significantly. Technology of nuclear energy provided new energy resources; however it threatened the world by the nuclear war. I hope that technologies complete the happiness of humankind.

1.2 Problems of existing methods and proposed methods

In this thesis, I discuss methods and algorithms for the matrix/tensor based feature extraction and the classification. The goal of this thesis is to contribute the progress of the feature extraction and the classification. In this section, I introduce existing methods of the matrix/tensor based feature extraction and classification, point out their problems, and propose new methods and algorithms to solve their problems, briefly.

1.2.1 FastGRBF

Function approximation by NMF is a technique to represent nonnegative feature vectors by linear combinations of basis functions. For example, if we choose smooth basis functions, then smooth feature vectors can be obtained. Zdunek proposed this technique as the GRBF-NMF method which uses the Gaussian radial basis functions for function approximation. However, its algorithm, in which the QP optimization and the active-set algorithm run alternately and iteratively, is so slow that it can not be applied to large-scale problems. Then, I propose a new fast algorithm for the function approximation by NMF, and choose more effective basis functions for function approximation. It is called the fastGRBF-NMF. Furthermore, I extend this method to the nonnegative Tucker decomposition (NTD) and the nonnegative canonical polyadic (CP) decomposition (NCPD), and they are called the fastGRBF-NTD and the fastGRBF-NCPD, respectively.

1.2.2 LCPTD/LTD

The common and individual feature analysis (CIFA) is a very important method for data analysis because real world data always have some common and individual features. For example all human faces have two eyes, two ears, a nose, and a mouse (common feature); but, their positions and shapes are different as individual personalities (individual feature). However, existing techniques for it are not very efficient, and their techniques have been developed only for matrix. Then, I propose a tensor based common and individual feature extraction method in CP and Tucker models, and impose orthogonality, sparsity, and nonnegativity constraints to both models. They are called the linked CP tensor decomposition (LCPTD) and the linked Tucker decomposition (LTD), respectively.

1.2.3 WRSVM

Support vector machine (SVM) has been the most popular classifier; however, SVM does not always provide the best performance. The criterion of SVM consists of the hinge-loss minimization and a regularization. In this research, I propose two novel weighted regularization methods for a variety of applications. Furthermore, I apply these regularization methods to the conventional SVM. It is called the SVM with weighted regularization (WRSVM)

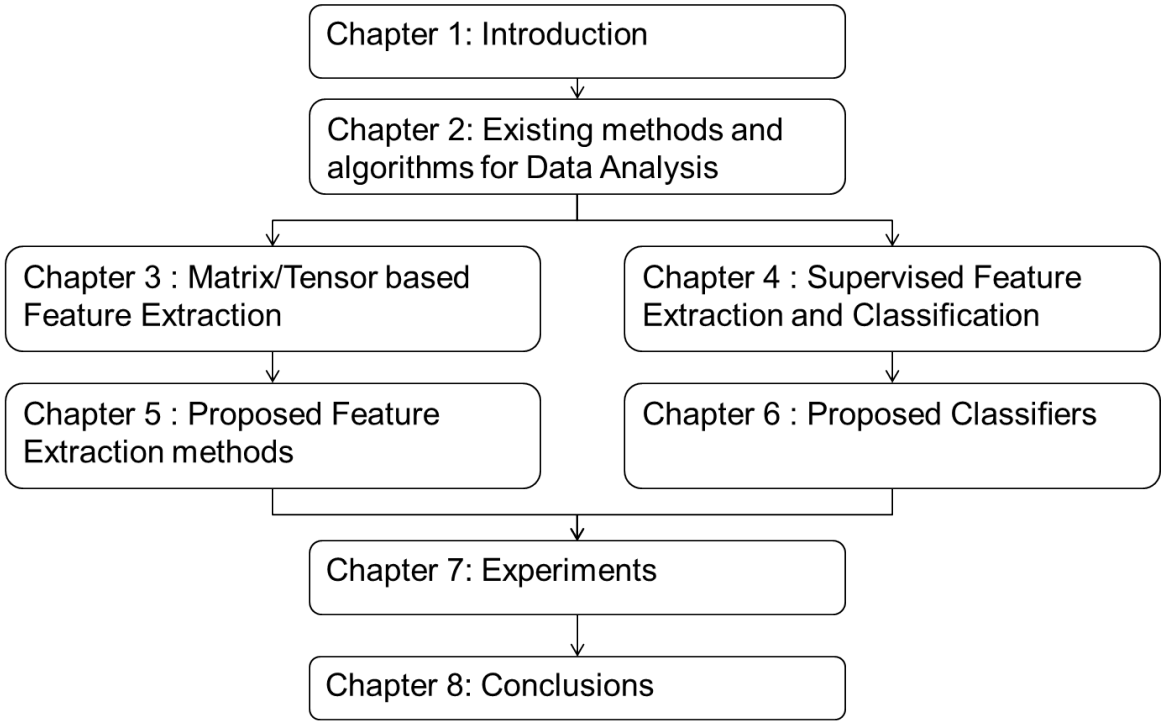


Figure 1.3: Flow of chapters in this thesis

1.2.4 CFDA/KCFDA

Fisher discriminant analysis (FDA) is another famous method for classification. However, FDA does give an optimal projection only for Gaussian distributions with equal covariance matrices. In other words, FDA is not optimal in the case of heteroscedastic Gaussian distributions. In this research, I propose a novel criterion for FDA including a correction term based on the Bhattacharyya distance which is closely related to classification rate. Furthermore, the Chernoff distance based criterion and its kernelized version are proposed as its extensions. They are called the Chernoff FDA (CFDA) and kernel CFDA (KCFDA), respectively.

1.2.5 QCMAP estimation

Final proposition of this thesis is a new criterion of classifier based on the maximum a posteriori (MAP) estimation for a binary problem without estimating the posterior probability, called the quadratically constrained maximum a posteriori (QCMAP) estimation. The QCMAP consists of the maximization of the expectation of a cost function, which is derived from the maximum a posteriori probability and a quadratic constraint. This criterion is highly general since its forms include least squares regressions (LSRs) and a SVM. I propose several efficient classifiers from the criterion by selecting various weight functions.

1.3 Organization

This thesis is organized as follows. The flowchart of chapters is depicted in Fig. 1.3.

In Chapter 2, I introduce several existing methods for data analysis: the discrete Fourier transform (DFT), the discrete wavelet transform (DWT), the principal component analysis (PCA), the sparse principal component analysis (SPCA), and the independent component analysis (ICA).

In Chapter 3, I introduce matrix and tensor decomposition techniques. The most basic matrix decomposition is the singular value decomposition (SVD). SVD provides a full-rank complete decomposition which consists of orthonormal left and right singular matrices and a diagonal rectangular matrix. Truncated SVD (tSVD) is a low-rank approximation model of SVD. tSVD provides the minimum mean squared error between an original data and a decomposition model. The SVD and the tSVD features are obtained based on the orthogonality constraint; however, the orthogonality is not always efficient. We have to use also the sparsity, the smoothness and the nonnegativity constraints as the situation demands. The penalized matrix factorization (PMD) is a technique to impose the sparsity or the smoothness constraint to the matrix decomposition model, and the nonnegative matrix factorization (NMF) is a technique to impose the nonnegativity constraint to the matrix decomposition model. Furthermore, the sparse NMF, the smooth NMF methods are introduced. The tensor decomposition is an extended technique of the matrix decomposition to analyze multi-dimensional array (tensor) data. There are two models in the tensor decomposition: CP model and Tucker model. The CP model is a direct extension of the matrix decomposition, and the Tucker model is a more general extension. I introduce methods that impose the orthogonality, the sparsity and the nonnegativity constraints to both models. Furthermore, I introduce the common and individual feature analysis (CIFA), and several solution algorithms. The common and individual feature extraction is a very important technique for data analysis since its model is well fitted into real problems.

In Chapter 5, I propose new methods for the matrix and tensor based feature extractions including new fast algorithms for smooth nonnegative matrix and tensor factorizations, and tensor extensions of the CIFA.

In Chapter 4, I introduce a supervised-future extraction and several basic classification methods: the common spatial pattern (CSP) filter, the kernel method, the least squares regression (LSR), the Fisher discriminant analysis (FDA), and the support vector machine (SVM).

In Chapter 6, I propose new classification methods such as the quadratically constrained maximum a posteriori estimation, SVM with weighted regularization, and the Chernoff FDA and its extension by the kernel method.

In Chapter 7, experiments are conducted to evaluate the proposed methods and their results are described. Finally, I conclude the research and provide future works in Chapter 8.

Table 1.1: Notations

$\mathbb{R}^N$	N-dimensional Euclidean space
$\langle \boldsymbol{f}, \boldsymbol{g} \rangle$	inner product of two vectors $\boldsymbol{f}$ and $\boldsymbol{g}$
$\|\|\boldsymbol{f}\|\|$	Euclidean norm of $\boldsymbol{f}$
$f(i)$	i-th component of $\boldsymbol{f}$
$\boldsymbol{I}_N$	$(N \times N)$ identity matrix
$\boldsymbol{A}^T$	transposition of a matrix $\boldsymbol{A}$
$\mathrm{vec}(\boldsymbol{A})$	vectorization of $\boldsymbol{A}$
$[\boldsymbol{A}]_{ij}, \boldsymbol{A}(i,j), a_{ij}$	(i,j) element of $\boldsymbol{A}$
$\mathrm{tr}[\boldsymbol{A}]$	trace of $\boldsymbol{A}$
$\mathrm{rank}(\boldsymbol{A})$	rank of $\boldsymbol{A}$

1.4 Notations

Table 1.1 describes a list of notations in this thesis. I denote individual variables by the following rules:

- Scalars are denoted by lowercase letters, e.g., $a \in \mathbb{R}$.
- Vectors are denoted by boldface lowercase letters, e.g., $\boldsymbol{a} \in \mathbb{R}^I$.
- Matrices are denoted by boldface capital letters, e.g., $\boldsymbol{A} \in \mathbb{R}^{I \times J}$.
- Higher order tensors are denoted by underlined and boldface capital letter, e.g., $\underline{\boldsymbol{A}}$.

Vectorization of a matrix is the transformation from the matrix to a vector. The vectorization of a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_J] \in \mathbb{R}^{I \times J}$ is given by

$$\mathrm{vec}(\boldsymbol{A}) := \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_J \end{pmatrix} \in \mathbb{R}^{IJ}. \tag{1.1}$$

Next, I introduce four product rules of matrices. The Hadamard product (element-wise multiplication) is defined as

$$\boldsymbol{A} \circledast \boldsymbol{B} := \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots \\ a_{21}b_{21} & a_{22}b_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}. \tag{1.2}$$

In similar way, the element wise division is defined as

$$\boldsymbol{A} \oslash \boldsymbol{B} := \begin{pmatrix} \frac{a_{11}}{b_{11}} & \frac{a_{12}}{b_{12}} & \cdots \\ \frac{a_{21}}{b_{21}} & \frac{a_{22}}{b_{22}} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}. \tag{1.3}$$

Simply, we have

$$(\boldsymbol{A} \circledast \boldsymbol{B})_{ij} = a_{ij} b_{ij}, \quad (\boldsymbol{A} \oslash \boldsymbol{B})_{ij} = \frac{a_{ij}}{b_{ij}}. \tag{1.4}$$

Let $\boldsymbol{A} \in \mathbb{R}^{I_a \times J_a}$ and $\boldsymbol{B} \in \mathbb{R}^{I_b \times J_b}$, the Kronecker product is defined by

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{I_a I_b \times J_a J_b}. \tag{1.5}$$

We have

$$(\boldsymbol{A} \otimes \boldsymbol{B})^T = \boldsymbol{A}^T \otimes \boldsymbol{B}^T, \tag{1.6}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{A} \otimes \boldsymbol{B})^T = (\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{A}^T \otimes \boldsymbol{B}^T) \tag{1.7}$$

$$= \boldsymbol{A}\boldsymbol{A}^T \otimes \boldsymbol{B}\boldsymbol{B}^T. \tag{1.8}$$

The Khatri-Rao product is defined by

$$\boldsymbol{A} \odot \boldsymbol{B} := \begin{pmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 & \cdots \end{pmatrix}, \tag{1.9}$$

$$= \begin{pmatrix} \mathrm{vec}(\boldsymbol{b}_1 \boldsymbol{a}_1^T) & \mathrm{vec}(\boldsymbol{b}_2 \boldsymbol{a}_2^T) & \cdots \end{pmatrix}. \tag{1.10}$$

We have

$$(\boldsymbol{A} \odot \boldsymbol{B})^T (\boldsymbol{A} \odot \boldsymbol{B}) = \begin{pmatrix} (\boldsymbol{a}_1^T \boldsymbol{a}_1)(\boldsymbol{b}_1^T \boldsymbol{b}_1) & (\boldsymbol{a}_1^T \boldsymbol{a}_2)(\boldsymbol{b}_1^T \boldsymbol{b}_2) & \cdots \\ (\boldsymbol{a}_2^T \boldsymbol{a}_1)(\boldsymbol{b}_2^T \boldsymbol{b}_1) & (\boldsymbol{a}_2^T \boldsymbol{a}_2)(\boldsymbol{b}_2^T \boldsymbol{b}_2) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{1.11}$$

$$= (\boldsymbol{A}^T \boldsymbol{A}) \circledast (\boldsymbol{B}^T \boldsymbol{B}). \tag{1.12}$$

Chapter 2

Existing methods and algorithms for Data Analysis

In this Chapter, I introduce existing methods and algorithms for multi-channel signal analysis including frequency analysis, time-frequency analysis, dimensionality reduction, and blind source separation.

2.1 Frequency Analysis

In this section, I introduce the Fourier transform as one of the techniques for frequency analysis. It is given by

$$F(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \mathrm{d}t. \tag{2.1}$$

In general, $F(\omega)$ becomes a complex function, and we often focus on $|F(\omega)|^2$ as a frequency power spectrum. Frequency is a very important factor for the brain signal analysis. Frequency bands of brain signals are called by various names. For example, a brain signal of which frequency components are in 8-13 Hz is called the alpha-wave. Furthermore, when we focus on the area of brain, the alpha-waves of the motor cortex, the temporal lobe, and the secondary somatosensory cortex are called the mu rhythm, the tau rhythm, and the sigma rhythm, respectively. Furthermore, the delta wave (0.1-4 Hz), the theta wave (4-7 Hz), the beta wave (12-30 Hz), and the gamma wave (25-100 Hz) are used in the brain signal analysis. In this way, it is very important to find out individual waves by frequency analysis.

2.2 Time-Frequency Analysis

Time-frequency analysis is a more important technique for the brain signal analysis than frequency analysis since we want to find out the temporal behavior for each frequency band. In this section, I introduce the wavelet transform (WT) as one of the techniques for time-frequency analysis. It is given

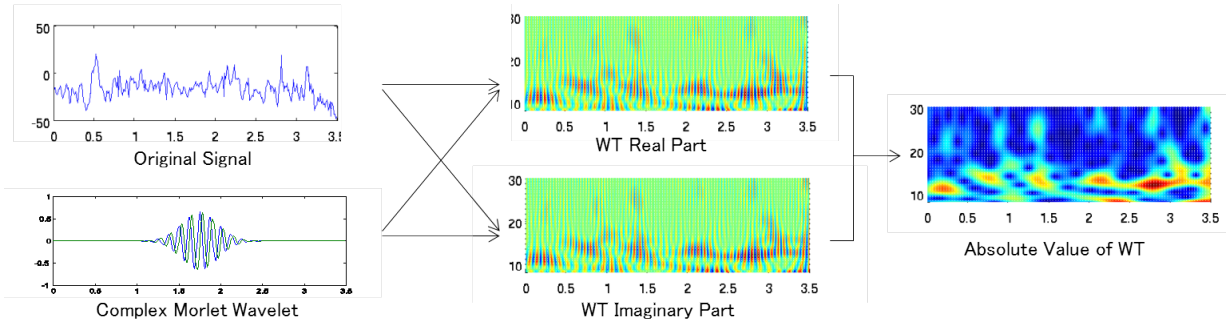


Figure 2.1: Wavelet transform

by

$$\boldsymbol{W}(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \overline{\psi\left(\frac{t-b}{a}\right)} \mathrm{d}t, \tag{2.2}$$

where $x(t)$ is a signal, $\psi(t)$ is a wavelet function. a and b stand for respectively variables of frequency domain and time domain. There are many kind of wavelets such as Haar wavelet, Meyer wavelet, Mexican Hat wavelet and Morlet wavelet. Fig. 2.1 shows an example of wavelet transform with the complex Morlet (CMOR) wavelet which is defined by

$$\psi_{f_b,f_c}(t) = \frac{1}{\sqrt{\pi f_b}} e^{i2\pi f_c t - (t^2/f_b)}, \tag{2.3}$$

where f_b and f_c are respectively a band width parameter and a frequency parameter. In this case, $\boldsymbol{W}$ becomes complex matrix and we often focus on $|\boldsymbol{W}|^2$ as a wavelet power spectrum matrix.

2.3 Dimensionality Reduction

When the dimension of data is very large, for example we analyze EEG signals which are recorded from 118 electronode channels to see an overall aspect of the brain without spatial information (e.g., P300), the dimensionality reduction techniques are quite useful. Furthermore, improvement of computational efficiency and noise reduction are also merits of the dimensionality reduction.

Its model is given by

$$\boldsymbol{b}_n = \boldsymbol{A}^T \boldsymbol{x}_n, \tag{2.4}$$

$$\boldsymbol{B} = \boldsymbol{A}^T \boldsymbol{X}, \tag{2.5}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in \mathbb{R}^{M \times N}$ is a set of original signals, $\boldsymbol{A} \in \mathbb{R}^{M \times R}$ is a transformation matrix, and $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_N] \in \mathbb{R}^{R \times N}$ is a set of transformed signals. Basically, we set R to be smaller than M because we want to reduce the dimensionality of the signals and extract only the important components. The goal of this method is to obtain the transformation matrix $\boldsymbol{A}$ which provides the meaningful feature $\boldsymbol{b}_n$.

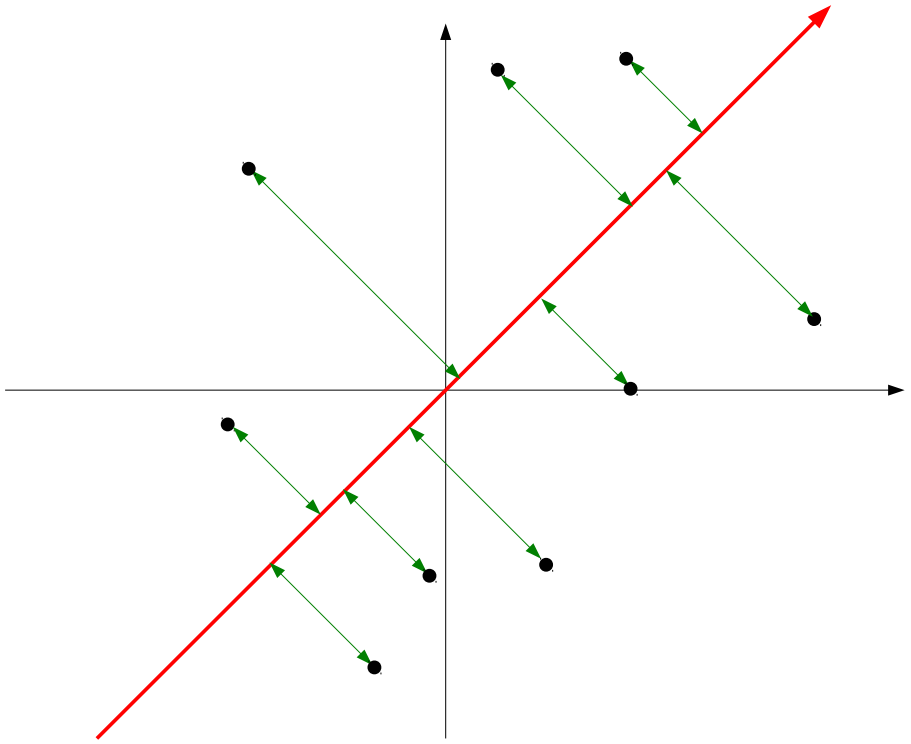


Figure 2.2: Concept of objective function of the PCA: black points stand for individual $\boldsymbol{x}_n$, red line stands for the transformed space, and green arrows stand for the difference between the original signals $\boldsymbol{x}_n$ and projected signals $\boldsymbol{b}_n$. The PCA minimize the sum of length of green arrows.

2.3.1 Principal Component Analysis

Principal Component Analysis (PCA) [52] is a very typical method for feature extraction and dimensionality reduction. It is defined by

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x}_n||^2 = ||\boldsymbol{X} - \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}||_F^2 \\ \text{subject to} \quad & \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_R, \end{aligned} \tag{2.6}$$

where $||\cdot||_F$ stands for the Frobenius norm and $\boldsymbol{I}_R \in \mathbb{R}^{R\times R}$ is an identity matrix. The objective function minimizes the Euclidean distance between the original signals $\boldsymbol{x}_n$ and projected signals of $\boldsymbol{b}_n$ (Fig. 2.2), and the constraint part imposes the orthogonality of the transformation matrix $\boldsymbol{A}$. This means that the PCA can be regarded as the minimization of the transformation loss.

On the other hand, the objective function can be transformed as

$$\begin{aligned}\sum_{n=1}^{N}||\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x}_n - \boldsymbol{x}_n||^2 &= ||\boldsymbol{A}\boldsymbol{A}^T X - X||_F^2 \\ &= \mathrm{tr}[(\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X} - \boldsymbol{X})(\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X} - \boldsymbol{X})^T] \\ &= \mathrm{tr}(\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{A}^T) - 2\mathrm{tr}(\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{X}^T) + \mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^T) \\ &= -\mathrm{tr}(\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}) + \mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^T), \end{aligned} \tag{2.7}$$

where $\mathrm{tr}(\cdot)$ describes 'trace'. Since $\mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^T)$ is a constant term, we can rewrite the PCA criterion as

$$\begin{aligned} &\text{maximize} \quad \mathrm{tr}(\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}), \\ &\text{subject to} \quad \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_R. \end{aligned} \tag{2.8}$$

Please note that this objective function is the same as $\mathrm{tr}(\boldsymbol{B}^T\boldsymbol{B})$. Thus, the PCA can be also regarded as the maximization of the auto-correlations of the transformed signals $\boldsymbol{b}_n$.

Solution of PCA

In this section, I provide a solution of the PCA criterion (2.8). First, its Lagrangian function is given by

$$L(\boldsymbol{A}, \boldsymbol{\Lambda}) = \mathrm{tr}(\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}) - \mathrm{tr}[(\boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I}_R)\boldsymbol{\Lambda}], \tag{2.9}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{R\times R}$ is a Lagrangian multiplier. Lagrange's conditions are given by

$$\frac{\partial L}{\partial \boldsymbol{A}} = 2\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A} - 2\boldsymbol{A}\boldsymbol{\Lambda} = 0, \tag{2.10}$$

$$\frac{\partial L}{\partial \boldsymbol{\Lambda}} = \boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I}_R = 0. \tag{2.11}$$

The solution must satisfy the conditions and maximizes the PCA objective function. Let $\lambda_1 \geq ... \geq \lambda_M$ and $\boldsymbol{\phi}_1$, ..., $\boldsymbol{\phi}_M$ be respectively the eigenvalues and eigenvectors of $(\boldsymbol{X}\boldsymbol{X}^T)$, then the solution of the PCA criterion is given by

$$\boldsymbol{A}_{\mathrm{PCA}} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_R]. \tag{2.12}$$

In other word, the solution is given as the eigenvectors of $(\boldsymbol{X}\boldsymbol{X}^T)$ corresponding to the largest R eigenvalues.

2.3.2 Sparse Principal Component Analysis

Sparseness is an important property to analyze large scale data by sparse representations. Furthermore, sparseness often provides robust analysis for noisy data. From this reasons, sparse extensions of data analysis methods have been studied very well such as the sparse nonnegative matrix factorization (NMF) [48, 49, 57], the sparse canonical correlation analysis (CCA) [79, 105], and the sparse partial

least squares (SPLS) [15]. The sparse principal component analysis (SPCA) [53, 121, 105] can be regarded as the most basic method among them.

Jolliffe et al. proposed the SCoTLASS (Simplified Component Technique-LASSO) method [53] by adding the LASSO (least absolute shrinkage and selection operator) constraint [95] to the PCA. The SCoTLASS criterion is given by

$$\begin{aligned} &\text{maximize} && \mathrm{tr}(\boldsymbol{a}_r^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{a}_r), \\ &\text{subject to} && \boldsymbol{a}_r^T \boldsymbol{a}_r = 1,\ \boldsymbol{a}_h^T \boldsymbol{a}_r = 0 \ (\text{for } r \geq 2 \text{ and } h < r), \\ & && ||\boldsymbol{a}_r||_1 \leq t, \end{aligned} \tag{2.13}$$

where $||\boldsymbol{a}||_1 := \sum_{m=1}^{M} |a_m|$ is a l1-norm, and $1 \leq t \leq \sqrt{M}$ is a tuning parameter. The smaller t, the solution becomes more sparse. For example, when $t = 1$, we must have one nonzero a_m, and when $t \geq \sqrt{M}$, we get the PCA. Please note that the problem (2.13) have many local optima, thus it needs numerical optimization to estimate parameters. Jolliffe et al. proposed to use the projected gradient approach [14, 46] as a solution method for the SCoTLASS.

However, Zou et al. [121] noted that the SCoTLASS fails to achieve sufficient sparsity and there is a possibility that orthogonality and sparsity constraints of the PC vectors are not fully compatible, thus they proposed an alternate method for the sparse PCA by extending it to the elastic net regression [120]. The SPCA criterion is given by

$$\begin{aligned} &\text{minimize} && ||\boldsymbol{X} - \widetilde{\boldsymbol{A}}\boldsymbol{A}^T \boldsymbol{X}||_F^2 + \lambda ||\boldsymbol{A}||_F^2 + \sum_{r=1}^{R} \lambda_1^{(r)} ||\boldsymbol{a}_r||_1, \\ &\text{subject to} && \widetilde{\boldsymbol{A}}^T \widetilde{\boldsymbol{A}} = \boldsymbol{I}_R, \end{aligned} \tag{2.14}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_R] \in \mathbb{R}^{M \times R}$ is a target sparse component, $\widetilde{\boldsymbol{A}} = [\widetilde{\boldsymbol{a}}_1, ..., \widetilde{\boldsymbol{a}}_R] \in \mathbb{R}^{M \times R}$ is regarded as a relaxing principal component, λ and $\lambda_1^{(r)}$ are nonnegative tuning parameters for penalty terms. Let consider $\widetilde{\boldsymbol{A}}_\perp \in \mathbb{R}^{M \times (M-R)}$ be an orthonomal matrix which satisfies that $[\widetilde{\boldsymbol{A}},\ \widetilde{\boldsymbol{A}}_\perp]$ is orthogonal, thus note that the first term of objective function (2.14) can be transformed as

$$\begin{aligned} ||\boldsymbol{X} - \widetilde{\boldsymbol{A}}\boldsymbol{A}^T \boldsymbol{X}||_F^2 &= \mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^T) - 2\mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^T \boldsymbol{A}\widetilde{\boldsymbol{A}}^T) + \mathrm{tr}(\widetilde{\boldsymbol{A}}\boldsymbol{A}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{A}\widetilde{\boldsymbol{A}}^T) \\ &= \mathrm{tr}(\widetilde{\boldsymbol{A}}_\perp^T \boldsymbol{X}\boldsymbol{X}^T \widetilde{\boldsymbol{A}}_\perp) + \mathrm{tr}(\widetilde{\boldsymbol{A}}^T \boldsymbol{X}\boldsymbol{X}^T \widetilde{\boldsymbol{A}}) - 2\mathrm{tr}(\widetilde{\boldsymbol{A}}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{A}) + \mathrm{tr}(\boldsymbol{A}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{A}) \\ &= \mathrm{tr}(\widetilde{\boldsymbol{A}}_\perp^T \boldsymbol{X}\boldsymbol{X}^T \widetilde{\boldsymbol{A}}_\perp) + ||\boldsymbol{X}^T \widetilde{\boldsymbol{A}} - \boldsymbol{X}^T \boldsymbol{A}||_F^2. \end{aligned} \tag{2.15}$$

Since the first term of (2.15) can be ignored for optimization, the first term of SPCA objective function can be replaced by the second term of (2.15).

Solution algorithm for SPCA

The problem (2.14) can be solved by an alternating algorithm of two sub-optimizations. The first optimization updates $\boldsymbol{A}$ with fixed $\widetilde{\boldsymbol{A}}$, and the second optimization updates $\widetilde{\boldsymbol{A}}$ with fixed $\boldsymbol{A}$. Both

Algorithm 1 SPCA algorithm

1: **Input:** $\boldsymbol{X}$, λ, $\lambda_1^{(r)}$ for $r = 1, ..., R$
2: **Initialize:** $\widetilde{\boldsymbol{A}}$ as the PCA result
3: **repeat**
4: Update $\boldsymbol{a}_r$ to solve the problem (2.16) for all r;
5: Update $\widetilde{\boldsymbol{A}}$ to solve the problem (2.19);
6: **until** convergence
7: Normalize $\boldsymbol{A}$ by $\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||$ for all r;
8: **Output:** $\boldsymbol{A}$, $\widetilde{\boldsymbol{A}}$

optimizations run alternately until the SPCA criterion is minimized. The first optimization problem is given by

$$\underset{\boldsymbol{a}_r}{\text{minimize}} \quad ||\boldsymbol{X}^T\widetilde{\boldsymbol{a}}_r - \boldsymbol{X}^T\boldsymbol{a}_r||^2 + \lambda||\boldsymbol{a}_r||^2 + \lambda_1^{(r)}||\boldsymbol{a}_r||_1. \tag{2.16}$$

Its problem is also called the elastic net problem [120], thus the solution is given by

$$\boldsymbol{a}_r(m) = \frac{\text{sgn}[\boldsymbol{v}_r(m)] \cdot \left[2\,|\boldsymbol{v}_r(m)| - \lambda_1^{(r)}\right]_+}{2(1+\lambda)}, \tag{2.17}$$

where $\boldsymbol{v}_r = \boldsymbol{X}\boldsymbol{X}^T\widetilde{\boldsymbol{a}}_r$, $[z]_+ := \max(z, \varepsilon)$ with a very small nonnegative parameter ε (e.g., $\varepsilon = 10^{-16}$), and $\text{sgn}(\cdot)$ function is defined by

$$\text{sgn}(z) = \begin{cases} 1 & z > 0 \\ 0 & z = 0 \\ -1 & z < 0 \end{cases}. \tag{2.18}$$

The second optimization problem is given by

$$\begin{aligned} \underset{\widetilde{\boldsymbol{A}}}{\text{minimize}} \quad & ||\boldsymbol{X} - \widetilde{\boldsymbol{A}}\boldsymbol{A}^T\boldsymbol{X}||_F^2, \\ \text{subject to} \quad & \widetilde{\boldsymbol{A}}^T\widetilde{\boldsymbol{A}} = \boldsymbol{I}_R. \end{aligned} \tag{2.19}$$

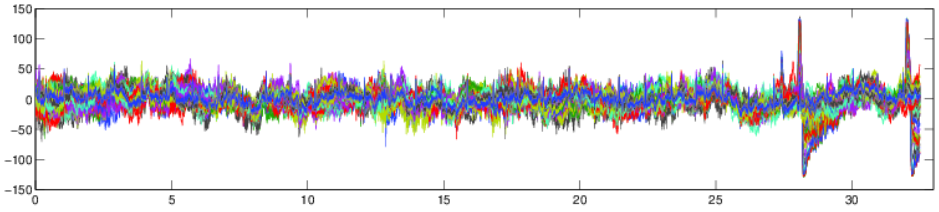
Its solution is given as

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{V}^T, \tag{2.20}$$

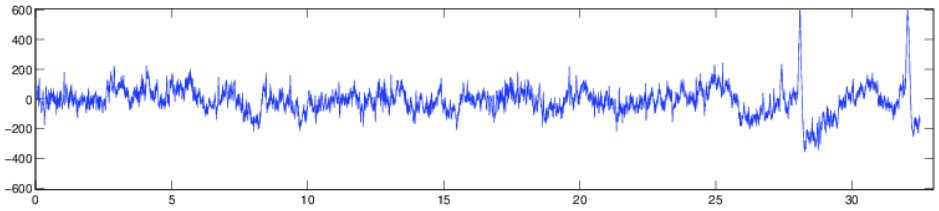
where $\boldsymbol{U} \in \mathbb{R}^{M \times R}$ and $\boldsymbol{V} \in \mathbb{R}^{R \times R}$ are respectively left and right factor matrices of singular value decomposition (SVD) of $\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. The details of SVD is described in Section 3.1.1. Finally, the SPCA algorithm can be summarized as Algorithm 1.

2.3.3 Example of Dimensionality Reduction

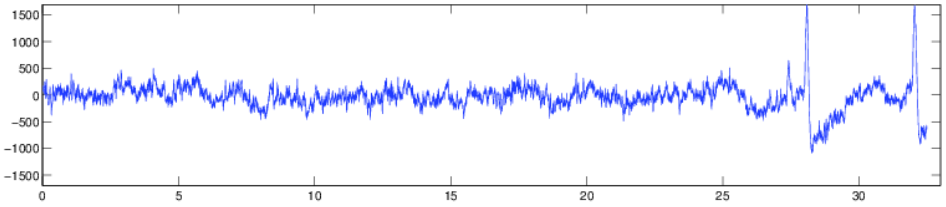
In this section, I introduce an example of application of dimensionality reduction. I used EEG signals from [8] of which sampling rate is 240 Hz and the number of channels is 64. This experiment was based



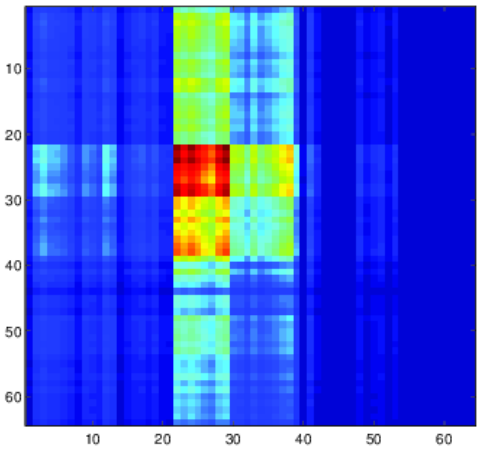
(a) 64 channels of EEG



(b) Embedded signal by the first PC component

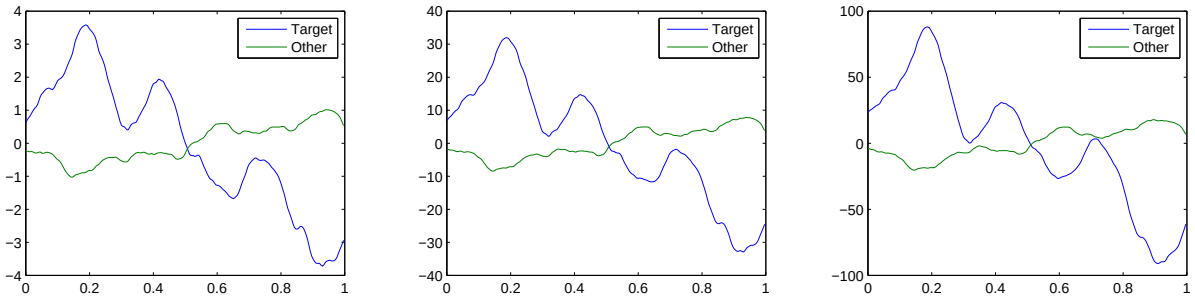


(c) Embedded signal by the first SPC component



(d) Correlation matrix between PC (vertical) and SPC (horizontal) components

Figure 2.3: Results of dimensionality reduction



(a) Average of 64 channels (b) PCA (c) SPCA

Figure 2.4: Results of ERP averages with target stimuli and other stimuli

on oddball paradigm, then I aimed to compare the event related potential (ERP) of target stimulus with ERP of other stimulus. Fig. 2.3 shows (a) the dataset of EEG signals, (b) reduced data by PCA, (c) reduced data by SPCA, and (d) correlation map between PC and SPC components. Their vertical and horizontal axis mean the signal amplitude and time [s], respectively. Seemingly, the results of PCA and SPCA are the almost same; however, we can see that only around 20-30 channels are important (correlated) from the correlation map. Fig. 2.4 shows ERPs of target and other stimuli compared by (a) average of 64 channels, (b) PCA, and (c) SPCA. Their vertical and horizontal axis mean the signal amplitude and time [s], respectively. In PCA and SPCA, the dimensionality was reduced from 64 into 1; however, results were very similar to original. Sigmal amplitudes of PCA/SPCA are significantly larger than original one because it is the first component including the largest eigenvalue for 64 eigenvalues. We can see that PCA and SPCA did not cut the important information.

2.4 Blind Source Separation

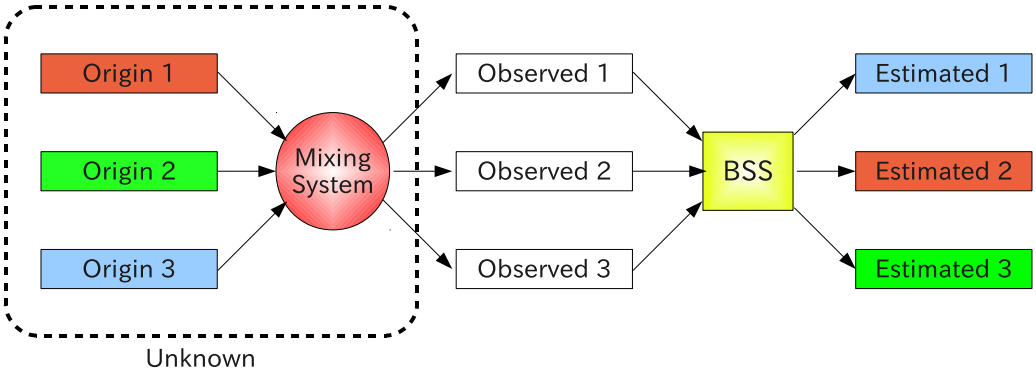
Blind Source Separation (BSS) is a method to estimate original signals from observed signals which consist of mixed original signals and noise. The cocktail party problem is a typical example of BSS. In case of the EEG signals, one electronode picks up signals from the surrounding numerous neurons, thus it can be considered that the regions covered by individual channels are overlapped each other (see Fig. 2.5).

Its model is given by

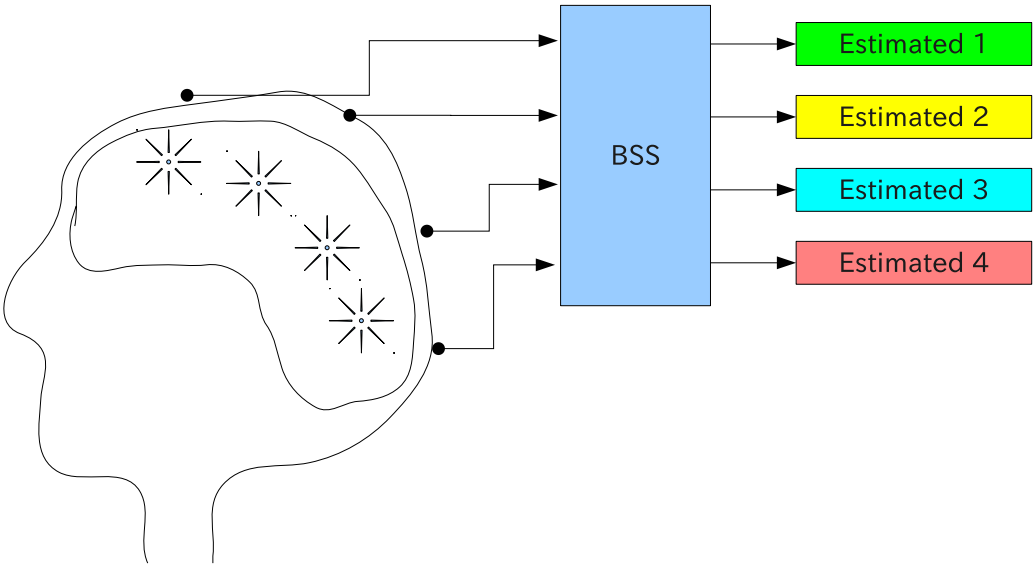
$$\boldsymbol{X} = \boldsymbol{B}\boldsymbol{S}, \tag{2.21}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_M]^T \in \mathbb{R}^{M\times N}$ and $\boldsymbol{S} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_R]^T \in \mathbb{R}^{R\times N}$ are respectively a observed signal matrix and a source signal matrix, and $\boldsymbol{B} \in \mathbb{R}^{M\times R}$ is a mixing matrix. When we regard $\boldsymbol{x}_m$ and $\boldsymbol{s}_r$ as functions $x_m(n)$ and $s_r(n)$ with respect to n, this model can be rewritten by

$$\boldsymbol{x}(n) = \boldsymbol{B}\boldsymbol{s}(n), \tag{2.22}$$



(a) General flow



(b) Example of brain

Figure 2.5: Blind Source Separation

where

$$\boldsymbol{x}(n) = \begin{pmatrix} x_1(n) \\ \vdots \\ x_M(n) \end{pmatrix}, \quad \boldsymbol{s}(n) = \begin{pmatrix} s_1(n) \\ \vdots \\ s_R(n) \end{pmatrix}. \tag{2.23}$$

The goal of BSS is to estimate $\boldsymbol{B}$ and $\boldsymbol{S}$ from $\boldsymbol{X}$. There are many methods for BSS such as the independent component analysis (ICA) [50], the sparse component analysis, and the nonnegative matrix factorization (NMF). In this section, I introduce ICA as the most typical approach for BSS, and the other methods are introduced in Chapter 3 since they are based on the matrix decomposition approach.

2.4.1 Independent Component Analysis

First, I introduce the hypothesis of ICA as follows:

Hypothesis of ICA

1. $\{s_r\}$ are statistically independent of each other,

$$p(s_1, s_2, \ldots, s_R) = p(s_1)p(s_2)\cdots p(s_R). \tag{2.24}$$

2. $\{s_r\}$ follow non-Gaussian distributions.
   If more than one of $\{s_r\}$ follow the Gaussian distribution, then ICA is impossible.

3. $\boldsymbol{B} \in \mathbb{R}^{M\times R}$ is a regular matrix (i.e., $M = R$).
   Therefore, we can rewrite the model as

$$\boldsymbol{s}(n) = \boldsymbol{A}^T\boldsymbol{x}(n), \tag{2.25}$$

   where $\boldsymbol{A}^T = \boldsymbol{B}^{-1}$. It is only necessary to estimate $\boldsymbol{A}$.

Before explaining the detail of ICA, it is helpful to show the overall procedures of ICA. ICA consists of two phases, mainly: the first phase is a whitening, and the second phase is a maximization of some non-Gaussianity measure. The reason why whitening is necessary is that the solution of $\boldsymbol{A}$ can be limited to an orthogonal matrix after whitening. The non-Gaussianity measure is very important for ICA because it expresses the statistical independence of s_r. Figure 2.6 shows an example of original independent signals and observed signals with $M = R = 2$, and its whitening and ICA results. The blue lines of Fig. 2.6(d) show the directions of new axes, thus original sources are almost completely estimated excluding the scale and the order of them.

Whitening

White signals are defined as any $\boldsymbol{z}$ which satisfy the following conditions

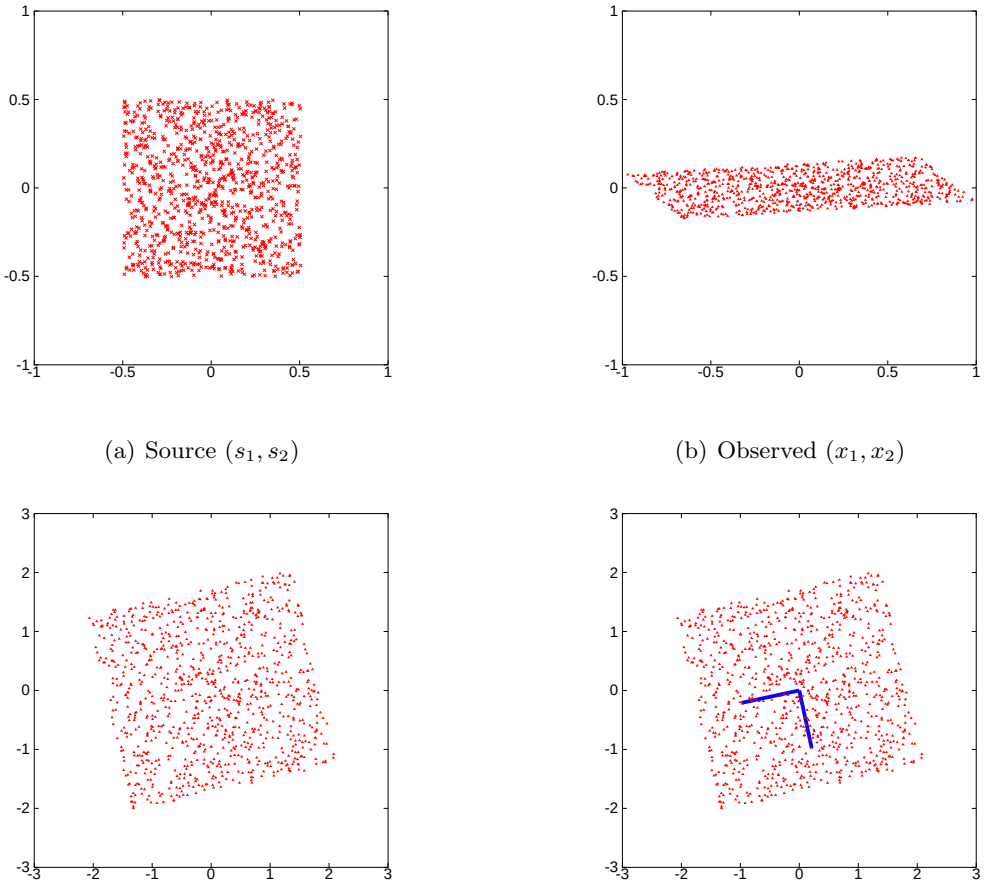
$$\mathrm{E}[\boldsymbol{z}] = \boldsymbol{0}, \quad \mathrm{E}[\boldsymbol{z}\boldsymbol{z}^T] = I. \tag{2.26}$$

First, we modify $\boldsymbol{X}$ as the zero average signal by

$$\boldsymbol{X}_0 = \boldsymbol{X}\left(\boldsymbol{I}_N - \frac{1}{N}\boldsymbol{1}_{N\times N}\right), \tag{2.27}$$

where $\boldsymbol{1}_{N\times N}$ is an $(N \times N)$-matrix of which all elements are 1. We denote the zero average signal as $\boldsymbol{x}_0(n)$. Second, the whitening result is given by $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{D}^{1/2}\boldsymbol{U}^T\boldsymbol{X}_0 \in \mathbb{R}^{R\times N}$ of eigenvalue decomposition of $[\frac{1}{N}\boldsymbol{X}_0\boldsymbol{X}_0^T] = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$. The whitening results are also denoted as $\boldsymbol{z}(n)$, and they are also given by

$$\boldsymbol{z}(n) = \boldsymbol{U}\boldsymbol{D}^{-1/2}\boldsymbol{U}^T\boldsymbol{x}_0(n). \tag{2.28}$$



(a) Source (s_1, s_2)

(b) Observed (x_1, x_2)

(c) Whitening result (z_1, z_2)

(d) ICA result

Figure 2.6: Individual phases of ICA

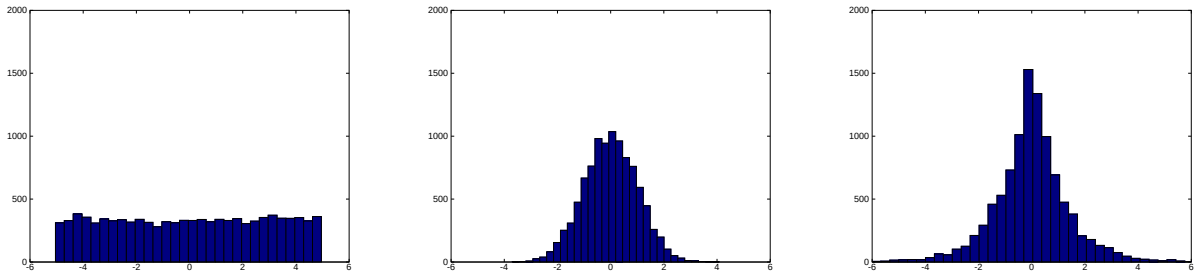
Thus, the model is givne by

$$\boldsymbol{s}(n) = \boldsymbol{W}^T \boldsymbol{z}(n) = \boldsymbol{W}^T \boldsymbol{U} \boldsymbol{D}^{-1/2} \boldsymbol{U}^T \boldsymbol{x}_0(n) = \boldsymbol{A}^T \boldsymbol{x}_0(n), \tag{2.29}$$

where $\boldsymbol{W} \in \mathbb{R}^{R \times R}$ is an orthogonal transform matrix. Therefore, it is only necessary to estimate $\boldsymbol{W}$ after whitening.

Non-Gaussianity

In order to separate the observed signals $\boldsymbol{x}(n)$ into statistically independent sources $\boldsymbol{s}(n)$, some measure of statistical independence is necessary. Non-Gaussianity can be regarded as a measure of statistical independence according to the central limit theorem (CLT). From the CLT, we can know that a distribution which consists of several independent and identically distributions approaches a Gaussian distribution. Thus, if we assume that $\{\boldsymbol{s}_r\}$ are identically distributed, then $\boldsymbol{x}_m = \sum_{r=1}^{R} b_{mr} s_r$ are closer to Gaussian distributions than $\{\boldsymbol{s}_r\}$. In other words, the non-Gaussianity of $\{\boldsymbol{s}_r\}$ are higher than $\{\boldsymbol{x}_m\}$.



(a) kurt$(y) = -1.23$ (b) kurt$(y) = -0.05$ (c) kurt$(y) = 2.99$

Figure 2.7: Kurtosis

Therefore, if some non-Gaussianity measure is given, we can estimate the parameter vector $\boldsymbol{a}_r$ by maximizing the non-Gaussianity of $\boldsymbol{a}_r^T \boldsymbol{x}(n)$. The criterion is given by

$$\underset{\boldsymbol{a}_r}{\text{maximize}} \quad \text{Non-Gaussianity}(\boldsymbol{a}_r^T \boldsymbol{x}(n)). \tag{2.30}$$

However, we can estimate only the first parameter vector from this criterion. In practice, we can limit the parameter vectors in orthogonal vectors after whitening, thus the criterion is given by

$$\begin{aligned} \underset{\boldsymbol{W}}{\text{maximize}} \quad & \sum_{r=1}^{R} \text{Non-Gaussianity}(\boldsymbol{w}_r^T \boldsymbol{z}(n)), \\ \text{subject to} \quad & \boldsymbol{W}^T \boldsymbol{W} = \boldsymbol{I}_R. \end{aligned} \tag{2.31}$$

There are several measures of non-Gaussianity such as Kurtosis and Neg-entropy. I introduce the kurtosis based ICA in Section 2.4.2 and the neg-entropy based ICA in Section 2.4.3.

2.4.2 Kurtosis based ICA

In this section, I introduce an algorithm for ICA based on Kurtosis. When the average of y is zero, the Kurtosis can be given by

$$\text{kurt}(y) = \text{E}[y^4] - 3(\text{E}[y^2])^2. \tag{2.32}$$

Furthermore, we assume that y is white (i.e. $\text{E}[y] = 0$, $\text{E}[y^2] = 1$), then it is given by

$$\text{kurt}(y) = \text{E}[y^4] - 3. \tag{2.33}$$

When y is distributed normally, kurt(y) becomes zero. On the other hand, y follows a non-Gaussian distribution, the kurtosis becomes some positive or negative value. Thus, the kurtosis can be used as a non-Gaussianity measure for ICA. When the kurt(y) is positive, its distribution is called the super-gaussian, and when the kurt(y) is negative, its distribution is called the sub-gaussian. Figure 2.7 show examples of histograms and their kurtoses.

The kurtosis based ICA criterion is given by

$$\begin{aligned} \text{maximize} \quad & \sum_{r=1}^{R} |\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z}(n))|, \\ \text{subject to} \quad & \boldsymbol{W}^T \boldsymbol{W} = \boldsymbol{I}_R. \end{aligned} \tag{2.34}$$

I denote $\boldsymbol{z}(n)$ as $\boldsymbol{z}$ for simplicity, and firstly focus to estimate one parameter vector $\boldsymbol{w}_r$ as

$$\text{maximize} \, |\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z})|, \text{ s.t. } \boldsymbol{w}_r^T \boldsymbol{w}_r = 1. \tag{2.35}$$

Differential of $|\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z})|$ is given by

$$\begin{aligned} \frac{\partial |\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z})|}{\partial \boldsymbol{w}_r} &= \frac{\partial}{\partial \boldsymbol{w}_r} \left| \text{E}\{(\boldsymbol{w}_r^T \boldsymbol{z})^4\} - 3\text{E}\{(\boldsymbol{w}_r^T \boldsymbol{z})^2\}^2 \right| \\ &= \frac{\partial}{\partial \boldsymbol{w}_r} \left| \text{E}\{(\boldsymbol{w}_r^T \boldsymbol{z})^4\} - 3\{||\boldsymbol{w}_r||^2\}^2 \right| \quad (\text{because } \text{E}(\boldsymbol{z}\boldsymbol{z}^T) = I) \\ &= 4\text{sign}[\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z})] \left[\text{E}\{\boldsymbol{z}(\boldsymbol{w}_r^T \boldsymbol{z})^3\} - 3\boldsymbol{w}_r ||\boldsymbol{w}_r||^2 \right] \end{aligned} \tag{2.36}$$

According to the gradient method, we can obtain following update rule:

$$\boldsymbol{w}_r \leftarrow \boldsymbol{w}_r + \Delta \boldsymbol{w}_r; \tag{2.37}$$

$$\boldsymbol{w}_r \leftarrow \frac{\boldsymbol{w}_r}{||\boldsymbol{w}_r||}; \tag{2.38}$$

where $\Delta \boldsymbol{w}_r \propto \text{sign}[\text{kurt}(\boldsymbol{w}_r^T \boldsymbol{z})] \left[\text{E}\{\boldsymbol{z}(\boldsymbol{w}_r^T \boldsymbol{z})^3\} - 3\boldsymbol{w}_r \right]$. Its algorithm converges when $\boldsymbol{w}_r \propto \Delta \boldsymbol{w}_r$, thus we can obtain another update rule as

$$\boldsymbol{w}_r \leftarrow \text{E}\{\boldsymbol{z}(\boldsymbol{w}_r^T \boldsymbol{z})^3\} - 3\boldsymbol{w}_r, \tag{2.39}$$

$$\boldsymbol{w}_r \leftarrow \frac{\boldsymbol{w}_r}{||\boldsymbol{w}_r||}. \tag{2.40}$$

It is well known as a fast fixed point algorithm for ICA.

Next, I introduce the algorithm to estimate orthogonal parameter vectors $\boldsymbol{W}$. This algorithm consists of two steps: the first step is to update $\{\boldsymbol{w}_r\}$ individually, the second step is to project $\boldsymbol{W}$ onto the orthogonal matrix. There are several orthogonalization methods, and they can be separated into the sequential approach and the symmetrical approach. The Gram-Schmidt orthogonalization is a very typical method for the sequential approach. It is given by

$$\boldsymbol{w}_r \leftarrow \boldsymbol{w}_r - \sum_{k=1}^{r-1} (\boldsymbol{w}_r^T \boldsymbol{w}_k) \boldsymbol{w}_k, \tag{2.41}$$

for $r = 1, ..., R$. On the other hand, a symmetrical orthogonalization is given by

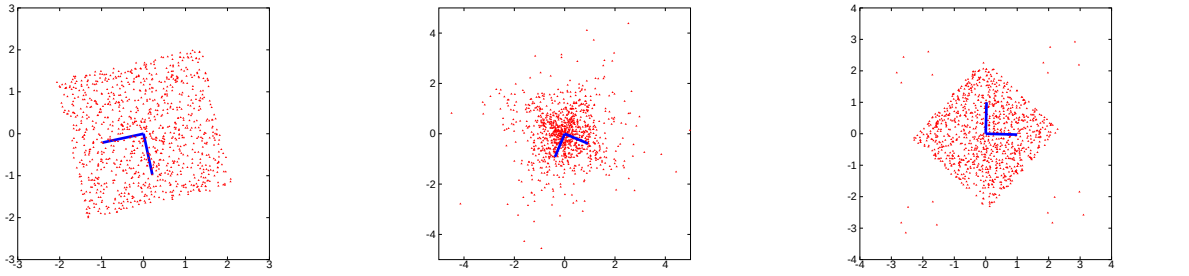
$$\boldsymbol{W} \leftarrow (\boldsymbol{W}\boldsymbol{W}^T)^{-1/2} \boldsymbol{W}. \tag{2.42}$$

When we want to treat individual vector fairly, the symmetrical approach is useful.

Finally, the kurtosis based ICA algorithm can be summarized as Algorithm 2.

Algorithm 2 Kurtosis based ICA algorithm

Input: $\boldsymbol{X} \in \mathbb{R}^{R \times N}$
Initialize: orthogonal matrix $\boldsymbol{W} \in \mathbb{R}^{R \times R}$
$\boldsymbol{X} \leftarrow \boldsymbol{X}(\boldsymbol{I}_N - \frac{1}{N}\boldsymbol{1}_{N \times N})$;
$\boldsymbol{Z} \leftarrow \boldsymbol{U}\boldsymbol{D}^{-1/2}\boldsymbol{U}^T\boldsymbol{X}$ with eigenvector $\boldsymbol{U}$ and eigenvalue $\boldsymbol{D}$ of $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T$;
repeat
    $\boldsymbol{w}_r \leftarrow \mathrm{E}\{\boldsymbol{z}(\boldsymbol{w}_r^T\boldsymbol{z})^3\} - 3\boldsymbol{w}_r$ for $r = 1, ..., R$;
    $\boldsymbol{w}_r \leftarrow \frac{\boldsymbol{w}_r}{||\boldsymbol{w}_r||}$ for $r = 1, ..., R$;
    Orthogonalize $\boldsymbol{W}$ by (2.41) or (2.42) ;
until convergence
$\boldsymbol{S} \leftarrow \boldsymbol{W}^T\boldsymbol{Z}$;
Output: $\boldsymbol{S}$



(a) subgaussian (b) supergaussian (c) subgaussian with outliers (20:1000)

Figure 2.8: Example of kurtosis based ICA

Example

Figure 2.8 shows the results of the kurtosis based ICA. Subgaussain distribution is generated by the uniform distribution, and supergaussian distribution is generated by the Laplacian distribution. In both distributions, the kurtosis based ICA worked well; however, it did not work with outliers. Actually, the kurtosis is a very sensitive measure since it is the fourth order function. The rate of outliers in Figure 2.8(c) is only 2% ; however it is enough to show the invalidity of the criterion.

2.4.3 Neg-entropy based ICA

The neg-entropy is often used for ICA since it is a robust measure of non-Gaussianity for outliers. Strictly, the approximation of neg-entropy is often used because it is more robust for outliers.

The neg-entropy is defined by

$$J(y) = H(y_{\mathrm{Gauss}}) - H(y), \tag{2.43}$$

where $H(y) = -\int p_y(\eta) \log p_y(\eta) \mathrm{d}\eta$ is a entropy and y_{Gauss} follows a Gaussian distribution with

$\mu = \mathrm{E}[y]$ and $\sigma^2 = \mathrm{E}[(y-\mu)^2]$. Thus, if y follows Gaussian distribution, then $J(y) = 0$, and if not, then $J(y) > 0$.

In this method, we approximate the neg-entropy by

$$J(y) \approx k_1(\mathrm{E}\{G_1(y)\})^2 + k_2(\mathrm{E}\{G_2(y)\} - \mathrm{E}\{G_2(\nu)\})^2, \tag{2.44}$$

where $G_1(\cdot)$ is a non-quadratically odd function, $G_2(\cdot)$ is a non-quadratically even function, k_1 and k_2 are positive constants, ν follows the standard Gaussian distribution, and we assume that y is white. This approximation is not strict, but it has the same properties of the neg-entropy. For example, this measure takes zero when y follows the Gaussian distribution, and positive when y does not follow the Gaussian distribution. Furthermore, this approximation can be robust measure by selecting appropriate functions for $G_1(\cdot)$ and $G_2(\cdot)$. The following functions are well known as a useful selection:

$$G_1(y) = \frac{1}{a_1} \log \cosh a_1 y, \tag{2.45}$$

$$G_2(y) = -\exp(-y^2/2), \tag{2.46}$$

where $1 \leq a_1 \leq 2$ (e.g., $a_1 = 1$ is often used).

If we use a single non-quadratic function, this approximation can be given by

$$J(y) \propto [\mathrm{E}\{G(y)\} - \mathrm{E}\{G(\nu)\}]^2. \tag{2.47}$$

This approximation has also the same properties of the neg-entropy, and it is simpler than the two functions version. Thus, we introduce the neg-entropy based ICA algorithm from this approximation. The objective function can be transformed as

$$\begin{aligned} J(y) &\propto [\mathrm{E}\{G(y)\} - \mathrm{E}\{G(\nu)\}]^2 \\ &\propto [\mathrm{E}\{G(y)\}]^2 - 2\mathrm{E}\{G(y)\}\mathrm{E}\{G(\nu)\} + [\mathrm{E}\{G(\nu)\}]^2. \end{aligned} \tag{2.48}$$

Furthermore, its differential function by $\boldsymbol{w}$ can be given by

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{w}} &\propto 2\mathrm{E}\{G(\boldsymbol{w}^T\boldsymbol{z})\}\mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} - 2\mathrm{E}\{G(\nu)\}\mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} \\ &\propto [\mathrm{E}\{G(\boldsymbol{w}^T\boldsymbol{z})\} - \mathrm{E}\{G(\nu)\}]\mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\}, \end{aligned} \tag{2.49}$$

where $g(\cdot)$ is the derivative of $G(\cdot)$. Thus, we can obtain the following updating rule based on a fixed point algorithm:

$$\boldsymbol{w} \leftarrow \mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\}, \tag{2.50}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w}/||\boldsymbol{w}||, \tag{2.51}$$

where we can ignore $[\mathrm{E}\{G(\boldsymbol{w}^T\boldsymbol{z})\} - \mathrm{E}\{G(\nu)\}]$ since $\boldsymbol{w}$ is normalized later. However, this algorithm do not converge well, because its approximation does not have a algebraic property that a true cumulant

Algorithm 3 Neg-entropy based ICA algorithm

1: **Input:** $\boldsymbol{X} \in \mathbb{R}^{R \times N}$
2: **Initialize:** orthogonal matrix $\boldsymbol{W} \in \mathbb{R}^{R \times R}$
3: $\boldsymbol{X} \leftarrow \boldsymbol{X}(\boldsymbol{I}_N - \frac{1}{N}\boldsymbol{1}_{N \times N})$;
4: $\boldsymbol{Z} \leftarrow \boldsymbol{U}\boldsymbol{D}^{-1/2}\boldsymbol{U}^T\boldsymbol{X}$ with eigenvector $\boldsymbol{U}$ and eigenvalue $\boldsymbol{D}$ of $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T]$;
5: **repeat**
6: $\quad \boldsymbol{w}_r \leftarrow \mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}_r^T\boldsymbol{z})\} - \mathrm{E}\{g'(\boldsymbol{w}_r^T\boldsymbol{z})\}\boldsymbol{w}_r$ for $r = 1, ..., R$;
7: $\quad \boldsymbol{w}_r \leftarrow \frac{\boldsymbol{w}_r}{||\boldsymbol{w}_r||}$ for $r = 1, ..., R$;
8: $\quad$ Orthogonalize $\boldsymbol{W}$ by (2.41) or (2.42) ;
9: **until** convergence
10: $\boldsymbol{S} \leftarrow \boldsymbol{W}^T\boldsymbol{Z}$;
11: **Output:** $\boldsymbol{S}$

has. Thus, we provide an improvement method for it by using a approximated newton's method. When we use (2.50) and (2.51), the convergence condition is given by

$$F(\boldsymbol{w}) = \mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \beta\boldsymbol{w} = \boldsymbol{0}, \tag{2.52}$$

where β is a parameter. We apply the approximated newton's method to the condition (2.52). Thus, the gradient of $F(\boldsymbol{w})$ is given by

$$\frac{\partial F}{\partial \boldsymbol{w}} = \mathrm{E}\{\boldsymbol{z}\boldsymbol{z}^T g'(\boldsymbol{w}^T\boldsymbol{z})\} + \beta\boldsymbol{I}_R, \tag{2.53}$$

where $g'(\cdot)$ is the derivative of $g(\cdot)$. Since $\boldsymbol{z}$ is white, we apply the following approximation,

$$\mathrm{E}\{\boldsymbol{z}\boldsymbol{z}^T g'(\boldsymbol{w}^T\boldsymbol{z})\} \approx \mathrm{E}\{\boldsymbol{z}\boldsymbol{z}^T\}\mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\} = \mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{I}_R. \tag{2.54}$$

Then, $\frac{\partial F}{\partial \boldsymbol{w}}$ becomes a diagonal matrix and its inverse matrix can be obtained easily. We can derive the following approximated newton's method

$$\begin{aligned}\boldsymbol{w} &\leftarrow \boldsymbol{w} - \frac{[\mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \beta\boldsymbol{w}]}{[\mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\} + \beta]} \propto \boldsymbol{w}[\mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\} + \beta] - [\mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \beta\boldsymbol{w}] \\ &= \boldsymbol{w}\mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\} - \mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\}\end{aligned} \tag{2.55}$$

Finally, the fast fixed point update rule for neg-entropy based ICA is given by

$$\boldsymbol{w} \leftarrow \mathrm{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} - \mathrm{E}\{g'(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{w}; \tag{2.56}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w}/||\boldsymbol{w}||; \tag{2.57}$$

and its algorithm can be summarized as Algorithm 3. $g(\cdot)$ and $g'(\cdot)$ in the Table 2.1 are often used. Please note that (g_3, g_3') is equivalent to Kurtosis based ICA.

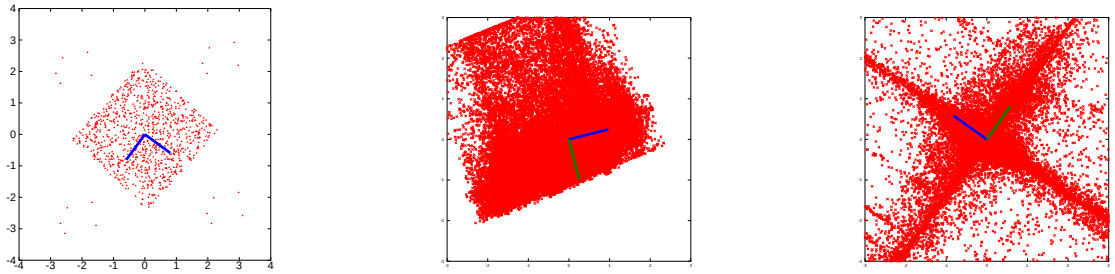
Table 2.1: Approximation functions for neg-entropy based ICA

label	$g(y)$	$g'(y)$
1	$g_1(y) = \tanh(a_1 y)$	$g_1'(y) = a_1(1 - \tanh^2(a_1 y))$
2	$g_2(y) = y \exp(-y^2/2)$	$g_2'(y) = (1 - y^2)\exp(-y^2/2)$
3	$g_3(y) = y^3$	$g_3'(y) = 3y^2$

Example

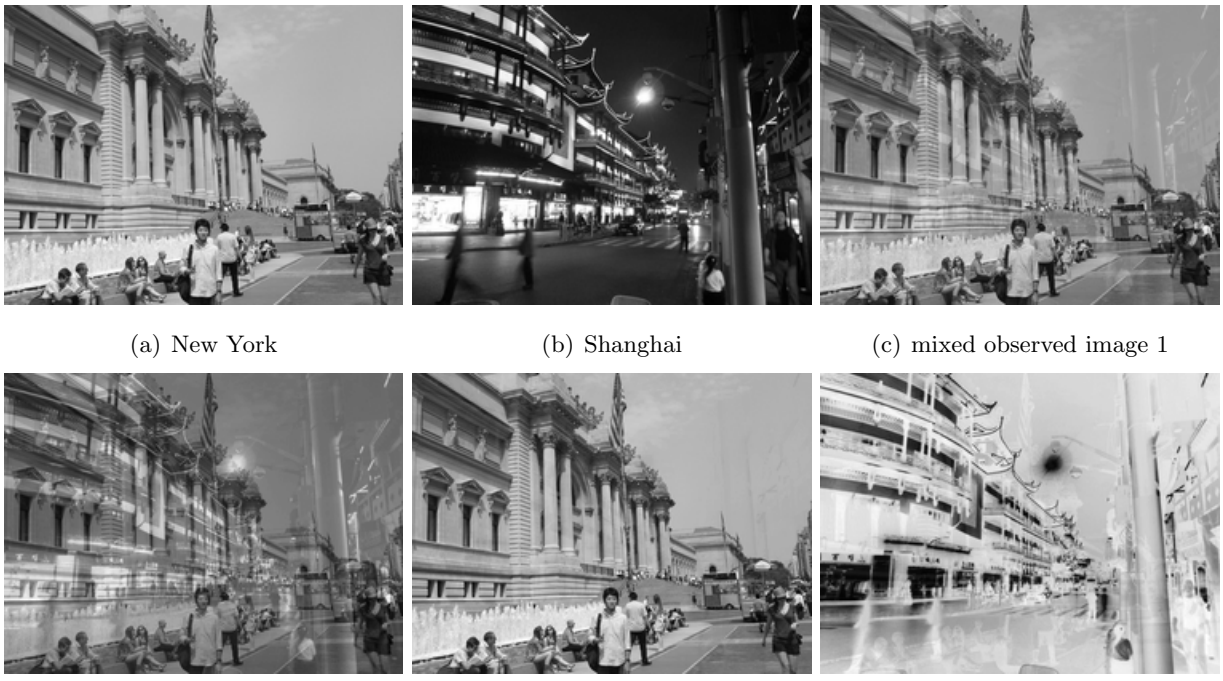
Figure 2.9 shows the results of the neg-entropy based ICA (using g_1). We can see that the neg-entropy based ICA is robust for outliers and works well for real data.

Fig. 2.10 shows an example of (a,b) original images, (c,d) mixed observed images, and (e,f) estimated sources. These images were separated clearly; however, black and white are reversed in the estimated signal 2. In this way, ICA can not estimate either the correct or not for the opposite sign.



(a) subgaussian with outliers (20:1000) (b) real data : subgaussian (c) real data : supergaussian

Figure 2.9: Neg-entropy based ICA (using g_1)



(a) New York (b) Shanghai (c) mixed observed image 1

(d) mixed observed image 2 (e) estimated source signal 1 (f) estimated source signal 2

Figure 2.10: Example of real image separation

Chapter 3

Matrix/Tensor based Feature Extraction

In this Chapter, I introduce matrix and tensor based feature extraction methods and point out their problems. Key-points of this chapter are decomposition models and imposed constraints. The decomposition models are separated into the matrix decomposition model and the tensor decomposition models. There are several constraints such as the orthogonality, the sparsity, the smoothness, and the nonnegativity. Properties of the method depend on the combination of the model and the constraint, and the models and the constraints are selected based on the situation demands.

3.1 Matrix decomposition model

In this section, I treat the feature extraction methods to analyze a matrix data. As an assumption, we have J observed signals $\{\boldsymbol{y}_j \in \mathbb{R}^I\}_{j=1}^J$ which can be given as a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_J] \in \mathbb{R}^{I \times J}$. The goal of feature extraction is to extract the latent important features included in the data. Matrix decomposition model is given by

$$\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X}, \tag{3.1}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_R] \in \mathbb{R}^{I \times R}$ and $\boldsymbol{X} = [x_{rj}] \in \mathbb{R}^{R \times J}$. This model can be also expressed as

$$\boldsymbol{y}_j \simeq \sum_{r=1}^{R} x_{rj} \boldsymbol{a}_r, \tag{3.2}$$

for $j = 1, ..., J$. Eq. (3.2) means that the matrix decomposition approximates $\boldsymbol{y}_j$ by a linear combination of bases $[\boldsymbol{a}_r]$. On the other hand, let $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_R] := \boldsymbol{X}^T \in \mathbb{R}^{J \times R}$, then the model can be given by

$$\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{B}^T = \sum_{r=1}^{R} \boldsymbol{a}_r \boldsymbol{b}_r^T. \tag{3.3}$$

Thus, the matrix decomposition approximates $\boldsymbol{Y}$ by a linear combination of one-rank matrices $[\boldsymbol{a}_r\boldsymbol{b}_r^T]$. The matrix decomposition model has a high-potential to apply to various real world problems such as clustering task, parts learning, denoising, and blind source separation.

When we try to extract the feature, it is important to make a criterion in concert with the target. For this reason, orthogonality, sparsity, smoothness, and nonnegativity are important factors to obtain such features. Actually, almost all methods consist of the minimization of some distance measure and some additional constraints. For example, a singular value decomposition (SVD) consists of the minimization of the Frobenius norm and the orthogonality constraint. In this way, we can say that the properties of methods are decided by the constraints.

3.1.1 Singular Value Decomposition

Singular value decomposition (SVD) is a basic matrix decomposition method. Its model is given by

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{3.4}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_I] \in \mathbb{R}^{I\times I}$, $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_J] \in \mathbb{R}^{J\times J}$ are unitary matrices, and $\boldsymbol{D} = [d_{ij}] \in \mathbb{R}^{I\times J}$ is a diagonal matrix that its diagonal elements are nonnegative values. Each $\boldsymbol{u}_r$, $\boldsymbol{v}_r$, and d_{rr} are called respectively a left-singular vector, a right-singular vector, and a singular value, thus they satisfy

$$\boldsymbol{u}_r^T\boldsymbol{Y} = \boldsymbol{v}_r^T d_{rr}, \tag{3.5}$$

$$d_{rr}\boldsymbol{u}_r = \boldsymbol{Y}\boldsymbol{v}_r, \tag{3.6}$$

for $r = 1, ..., \min(I, J)$. Please note that the SVD model (3.4) is defined as an equation since this model can completely provide the original data matrix $\boldsymbol{Y}$. The criterion of SVD is given by

$$\begin{aligned} &\text{minimize} \quad ||\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T||_F^2, \\ &\text{subject to} \quad \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_I,\ \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}_J, \end{aligned} \tag{3.7}$$

where $\boldsymbol{I}_I \in \mathbb{R}^{I\times I}$ and $\boldsymbol{I}_J \in \mathbb{R}^{J\times J}$ are identity matrices. The objective function can be transformed as

$$\begin{aligned} ||\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T||_F^2 &= \mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{Y}) - 2\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T) + \mathrm{tr}(\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T) \\ &= \mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{Y}) - 2\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T) + \mathrm{tr}(\boldsymbol{D}^T\boldsymbol{D}). \end{aligned} \tag{3.8}$$

Its Lagrange's condition with respect to $\boldsymbol{D}$ is given by

$$\frac{\partial}{\partial \boldsymbol{D}}||\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T||_F^2 = -2\boldsymbol{U}^T\boldsymbol{Y}\boldsymbol{V} + 2\boldsymbol{D} = \boldsymbol{0}. \tag{3.9}$$

Thus, the solution of $\boldsymbol{D}$ can be given by

$$\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{Y}\boldsymbol{V}. \tag{3.10}$$

We can see that eq. (3.10) satisfies the conditions (3.5) and (3.6). By using (3.8) and (3.10), the Lagrange's function can be given by

$$\begin{aligned} L(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2) = &- \operatorname{tr}(\boldsymbol{U}^T \boldsymbol{Y} \boldsymbol{V} \boldsymbol{V}^T \boldsymbol{Y}^T \boldsymbol{U}) \\ &+ \operatorname{tr}[(\boldsymbol{U}^T \boldsymbol{U} - \boldsymbol{I}_I)\boldsymbol{\Lambda}_1] + \operatorname{tr}[(\boldsymbol{V}^T \boldsymbol{V} - \boldsymbol{I}_J)\boldsymbol{\Lambda}_2], \end{aligned} \tag{3.11}$$

where $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ are Lagrange's coefficients. Its Lagrange's conditions with respect to $\boldsymbol{U}$ and $\boldsymbol{V}$ are given by

$$\frac{\partial L}{\partial \boldsymbol{U}} = -2(\boldsymbol{Y}\boldsymbol{Y}^T)\boldsymbol{U} + 2\boldsymbol{U}\boldsymbol{\Lambda}_1 = 0, \tag{3.12}$$

$$\frac{\partial L}{\partial \boldsymbol{V}} = -2(\boldsymbol{Y}^T\boldsymbol{Y})\boldsymbol{V} + 2\boldsymbol{V}\boldsymbol{\Lambda}_2 = 0. \tag{3.13}$$

Eq. (3.12) is an eigenvalue problem, thus the solution of $\boldsymbol{U}$ can be given as a set of eigenvectors of $(\boldsymbol{Y}\boldsymbol{Y}^T)$. Let $[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_I]$ and $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_I$ be respectively eigenvectors and eigenvalues of $(\boldsymbol{Y}\boldsymbol{Y}^T)$, then $\boldsymbol{U}$ is given by

$$\boldsymbol{U} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_I]. \tag{3.14}$$

In the same way, the solution of $\boldsymbol{V}$ can be given as a set of eigenvectors of $(\boldsymbol{Y}^T\boldsymbol{Y})$. Let $[\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, ..., \boldsymbol{\psi}_J]$ and $\eta_1 \geq \eta_2 \geq ... \geq \eta_J$ be respectively eigenvectors and their corresponding eigenvalues of $(\boldsymbol{Y}^T\boldsymbol{Y})$, then $\boldsymbol{V}$ is given by

$$\boldsymbol{V} = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, ..., \boldsymbol{\psi}_J]. \tag{3.15}$$

Let $R = \min(I, J)$, $D_{rr} = \sqrt{\lambda_r} = \sqrt{\eta_r}$ for $r = 1, ..., R$.

Truncated SVD

Truncated singular value decomposition (tSVD) is a method to obtain a low-rank approximation of original data matrix $\boldsymbol{Y}$. Its model is given by

$$\boldsymbol{Y} \simeq \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{3.16}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_R] \in \mathbb{R}^{I \times R}$, $\boldsymbol{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_R] \in \mathbb{R}^{J \times R}$ are orthonormal matrices, and $\boldsymbol{D} \in \mathbb{R}^{R \times R}$ is a diagonal matrix which its diagonal elements are singular values. In general, $R \leq \operatorname{rank}(\boldsymbol{Y}) \leq \min(I, J)$. The criterion of tSVD is given by

$$\text{minimize} \quad ||\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T||_F^2, \tag{3.17}$$

$$\text{subject to} \quad \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_R, \ \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_R, \tag{3.18}$$

where $\boldsymbol{I}_R \in \mathbb{R}^{R \times R}$ is an identity matrix. As its name suggests, the solution of tSVD is given by truncating the solution of SVD. Let $\boldsymbol{U}_0 = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_I]$, $\boldsymbol{V}_0 = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_J]$, and $\boldsymbol{D}_0 = [d_{ij}]$ be the

solution of SVD, then each factor matrices are given by

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_R], \tag{3.19}$$

$$\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_R], \tag{3.20}$$

$$\boldsymbol{D} = [\boldsymbol{D}_0]_{1:R,1:R}. \tag{3.21}$$

The tSVD gives an R-rank approximation of $\boldsymbol{Y}$ which minimizes the mean squared error.

Deflation based algorithm for tSVD

The above mentioned algorithm for tSVD is consequential and wasteful since it must obtain a full-rank decomposition. In this section, I introduce a deflation based algorithm to obtain the solution of tSVD, directly. First we consider one-rank approximation criterion as follow:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\boldsymbol{Y} - d\boldsymbol{u}\boldsymbol{v}^T||_F^2, \\ &\text{subject to} \quad ||\boldsymbol{u}||^2 = 1,\ ||\boldsymbol{v}||^2 = 1. \end{aligned} \tag{3.22}$$

The objective function of (3.22) can be transformed as

$$\frac{1}{2}||\boldsymbol{Y} - d\boldsymbol{u}\boldsymbol{v}^T||_F^2 = \frac{1}{2}\mathrm{tr}(\boldsymbol{Y}\boldsymbol{Y}^T) - d\boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v} + \frac{1}{2}d^2. \tag{3.23}$$

Focusing on this objective function with respect to only $\boldsymbol{u}$ and $\boldsymbol{v}$, a criterion (3.22) can be transformed to

$$\begin{aligned} &\text{maximize} \quad \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v}, \\ &\text{subject to} \quad ||\boldsymbol{u}||^2 = 1,\ ||\boldsymbol{v}||^2 = 1. \end{aligned} \tag{3.24}$$

We solve this criterion by updating $\boldsymbol{u}$ and $\boldsymbol{v}$, alternately and iteratively until convergence. Update rules for $\boldsymbol{u}$ and $\boldsymbol{v}$ are derived by the following criteria:

$$\boldsymbol{u} \leftarrow \underset{\boldsymbol{u}}{\mathrm{argmax}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t.}\ ||\boldsymbol{u}|| = 1, \tag{3.25}$$

$$\boldsymbol{v} \leftarrow \underset{\boldsymbol{v}}{\mathrm{argmax}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t.}\ ||\boldsymbol{v}|| = 1. \tag{3.26}$$

Thus, the update rule for $\boldsymbol{u}$ is given by

$$\boldsymbol{u} \leftarrow \boldsymbol{Y}\boldsymbol{v}, \tag{3.27}$$

$$\boldsymbol{u} \leftarrow \boldsymbol{u}/||\boldsymbol{u}||. \tag{3.28}$$

In a similar way, the update rule for $\boldsymbol{v}$ is given by

$$\boldsymbol{v} \leftarrow \boldsymbol{Y}^T\boldsymbol{u}, \tag{3.29}$$

$$\boldsymbol{v} \leftarrow \boldsymbol{v}/||\boldsymbol{v}||. \tag{3.30}$$

Algorithm 4 One-rank tSVD by power method

Input: $\boldsymbol{Y}$
Initialize: $\boldsymbol{v}$ by randomly
repeat
  $\boldsymbol{u} \leftarrow \boldsymbol{Y}\boldsymbol{v}$;
  $\boldsymbol{u} \leftarrow \boldsymbol{u}/||\boldsymbol{u}||$;
  $\boldsymbol{v} \leftarrow \boldsymbol{Y}^T\boldsymbol{u}$;
  $\boldsymbol{v} \leftarrow \boldsymbol{v}/||\boldsymbol{v}||$;
until convergence
$d \leftarrow \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v}$;
Output: d, $\boldsymbol{u}$, and $\boldsymbol{v}$

Algorithm 5 R-rank tSVD

Input: $\boldsymbol{Y}$, R
Initialize: $\boldsymbol{Z}_0 := \boldsymbol{Y}$
for $r = 1, ..., R$ **do**
  Obtain $(d_r, \boldsymbol{u}_r, \boldsymbol{v}_r)$ by Algorithm 4 inputting $\boldsymbol{Z}_{r-1}$;
  $\boldsymbol{Z}_r \leftarrow \boldsymbol{Z}_{r-1} - d_r\boldsymbol{u}_r\boldsymbol{v}_r^T$;
end for
Output: $[d_r, \boldsymbol{u}_r, \boldsymbol{v}_r]_{r=1}^R$

Its singular value is given by $d = \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v}$. Finally, the solution algorithm for one-rank approximation of tSVD can be summarized as Algorithm 4. This iterative algorithm is often called the power method.

Next we consider R-rank approximation of tSVD. We can construct the algorithm for R-rank version of tSVD by using one-rank approximation of tSVD. Let d_r, $\boldsymbol{u}_r$, and $\boldsymbol{v}_r$ be the r-th singular value and left- and right-singular vectors, and define the r-tSVD model $\widehat{\boldsymbol{Y}}_r = \sum_{k=1}^r d_k\boldsymbol{u}_k\boldsymbol{v}_k^T$ and the deflated matrix $\boldsymbol{Z}_r := \boldsymbol{Y} - \widehat{\boldsymbol{Y}}_r$, then we have

$$\begin{aligned}\mathrm{rank}(\boldsymbol{Y}) &= \mathrm{rank}(\widehat{\boldsymbol{Y}}_1) + \mathrm{rank}(\boldsymbol{Z}_1) = \mathrm{rank}(\widehat{\boldsymbol{Y}}_2) + \mathrm{rank}(\boldsymbol{Z}_2) = \cdots \\ &= \mathrm{rank}(\widehat{\boldsymbol{Y}}_r) + \mathrm{rank}(\boldsymbol{Z}_r) = \cdots = \mathrm{rank}(\widehat{\boldsymbol{Y}}_R) + \mathrm{rank}(\boldsymbol{Z}_R), \end{aligned} \tag{3.31}$$

since $\boldsymbol{Z}_r$ and $\widehat{\boldsymbol{Y}}_r$ are orthogonal (i.e., $\boldsymbol{Z}_r^T\widehat{\boldsymbol{Y}}_r = \boldsymbol{0}$ and $\boldsymbol{Z}_r\widehat{\boldsymbol{Y}}_r^T = \boldsymbol{0}$). Thus we can say that the r-th singular value and left- and right-singular vectors of $\boldsymbol{Y}$ can be given as the first singular value and left- and right-singular vectors of $\boldsymbol{Z}_{r-1}$. Therefore, the algorithm for R-rank approximation of tSVD can be summarized as Algorithm 5.

3.1.2 Penalized Matrix Decomposition

Penalized matrix decomposition [105] can be regarded as a penalized tSVD. It was proposed to add several constraints into the tSVD criterion and solve the criterion based on a deflation based algorithm.

Algorithm 6 One-rank PMD by power method

1: **Input:** $\boldsymbol{Y}$, P_1 c_1, P_2 and c_2
2: **Initialize:** $\boldsymbol{v}$ by randomly
3: **repeat**
4: $\quad \boldsymbol{u} \leftarrow \text{argmax}_{\boldsymbol{u}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t. } ||\boldsymbol{u}|| = 1,\ P_1(\boldsymbol{u}) \leq c_1;$
5: $\quad \boldsymbol{v} \leftarrow \text{argmax}_{\boldsymbol{v}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t. } ||\boldsymbol{v}|| = 1,\ P_2(\boldsymbol{v}) \leq c_2;$
6: **until** convergence
7: $d \leftarrow \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v}$;
8: **Output:** d, $\boldsymbol{u}$, and $\boldsymbol{v}$

First we consider one-rank approximation of PMD as follow:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\boldsymbol{Y} - d\boldsymbol{u}\boldsymbol{v}^T||_F^2, \\ &\text{subject to} \quad ||\boldsymbol{u}||^2 = 1,\ ||\boldsymbol{v}||^2 = 1,\ P_1(\boldsymbol{u}) \leq c_1,\ P_2(\boldsymbol{v}) \leq c_2, \end{aligned} \tag{3.32}$$

where $P_1(\boldsymbol{u})$ and $P_2(\boldsymbol{v})$ are convex penalty functions. In [105], two penalty functions are proposed, mainly: l1-norm penalty and fused lasso (FL) penalty. When we choose the l1-norm penalty, then PMD provides a sparse representation. When we choose the FL penalty, then PMD provides a sparse and smooth representation. Focusing on this objective function with respect to only $\boldsymbol{u}$ and $\boldsymbol{v}$, a criterion (3.32) can be transformed to

$$\begin{aligned} &\text{maximize} \quad \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v}, \\ &\text{subject to} \quad ||\boldsymbol{u}||^2 = 1,\ ||\boldsymbol{v}||^2 = 1,\ P_1(\boldsymbol{u}) \leq c_1,\ P_2(\boldsymbol{v}) \leq c_2. \end{aligned} \tag{3.33}$$

In a similar way of (3.25) and (3.26), the update rules for $\boldsymbol{u}$ and $\boldsymbol{v}$ are derived by the following criteria:

$$\boldsymbol{u} \leftarrow \underset{\boldsymbol{u}}{\text{argmax}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t. } ||\boldsymbol{u}|| = 1,\ P_1(\boldsymbol{u}) \leq c_1, \tag{3.34}$$

$$\boldsymbol{v} \leftarrow \underset{\boldsymbol{v}}{\text{argmax}}\ \boldsymbol{u}^T\boldsymbol{Y}\boldsymbol{v},\ \text{s.t. } ||\boldsymbol{v}|| = 1,\ P_2(\boldsymbol{v}) \leq c_2. \tag{3.35}$$

Therefore, the algorithms for one- and R-rank PMD are very similar to one- and R-rank tSVD, and they are summarized as Algorithms 6 and 7.

l1-norm penalty

In this section, we consider to choose the l1-norm penalty function as $P_{\text{l1}}(\boldsymbol{u}) := ||\boldsymbol{u}||_1 = \sum_{i=1}^{I} |u_i|$. Let $\boldsymbol{a} = \boldsymbol{Y}\boldsymbol{v}$, the problem (3.34) can be rewritten by

$$\boldsymbol{u} \leftarrow \underset{\boldsymbol{u}}{\text{argmax}}\ \boldsymbol{u}^T\boldsymbol{a},\ \text{s.t. } ||\boldsymbol{u}|| = 1,\ ||\boldsymbol{u}||_1 \leq c_1. \tag{3.36}$$

If $\left|\left|\frac{\boldsymbol{a}}{||\boldsymbol{a}||}\right|\right|_1 \leq c_1$, this solution is given by $\boldsymbol{u} = \frac{\boldsymbol{a}}{||\boldsymbol{a}||}$; otherwise this solution

$$\boldsymbol{u} = \frac{\boldsymbol{s}(\boldsymbol{a}, \Delta)}{||\boldsymbol{s}(\boldsymbol{a}, \Delta)||} \tag{3.37}$$

Algorithm 7 R-rank PMD

1: **Input:** $\boldsymbol{Y}$, R
2: **Initialize:** $\boldsymbol{Z}_0 := \boldsymbol{Y}$
3: **for** $r = 1, ..., R$ **do**
4: Obtain $(d_r, \boldsymbol{u}_r, \boldsymbol{v}_r)$ by Algorithm 6 inputting $\boldsymbol{Z}_{r-1}$;
5: $\boldsymbol{Z}_r \leftarrow \boldsymbol{Z}_{r-1} - d_r \boldsymbol{u}_r \boldsymbol{v}_r^T$;
6: **end for**
7: **Output:** $[d_r, \boldsymbol{u}_r, \boldsymbol{v}_r]_{r=1}^R$

is obtained by a line-search algorithm for a scalar parameter Δ so that it satisfy $||\boldsymbol{u}||_1 = c_1$, where $\boldsymbol{s}(\boldsymbol{a}, \Delta)$ is calculated by $s_i(\boldsymbol{a}, \Delta) = \text{sign}(a_i)[|a_i| - \Delta]_+$. The function $\boldsymbol{s}(\boldsymbol{a}, \Delta)$ is called the soft-thresholding operator. The PMD with l1-norm penalty can be regarded as a sparse representation of tSVD; however, sparse constraint and orthogonal constraint are incompatible, thus the PMD can not provide completely orthogonal factor matrices.

Fused lasso penalty

In this section, we consider to choose an FL penalty function as $P_{FL}(\boldsymbol{u}) := \sum_{i=1}^I |u_i| + \lambda \sum_{i=2}^I |u_i - u_{i-1}|$. In case of FL penalty, we solve a slightly modified Lagrange form, for simplicity, which is given by

$$\boldsymbol{u} \leftarrow \underset{\boldsymbol{u}}{\text{argmin}} \ \frac{1}{2}||\boldsymbol{u} - \boldsymbol{a}||_F^2 + \lambda_1 ||\boldsymbol{u}||_1 + \lambda_2 \sum_{i=2}^I |u_i - u_{i-1}|, \tag{3.38}$$

where λ_1 and λ_2 are trade-off parameters. This problem can be solved by the fused lasso regression [36, 96, 47]. The PMD with the FL penalty can be regarded as a sparse and smooth representation of tSVD. Although it can not provide completely orthogonal factor matrices, the smooth representations are focused in wide research area. In brain signal processing, temporal or spatial smoothness is considered as an important feature.

3.1.3 Nonnegative Matrix Factorization

Nonnegative matrix factorization (NMF) is a technique to decompose a nonnegative matrix into two nonnegative matrices. The standard NMF model is given by

$$\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X} \in \mathbb{R}_+^{I \times J}, \tag{3.39}$$

where $\boldsymbol{A} \in \mathbb{R}_+^{I \times R}$, $\boldsymbol{X} \in \mathbb{R}_+^{R \times J}$. $\boldsymbol{Y} = [\boldsymbol{y}_1, \ \boldsymbol{y}_2, \ \ldots, \ \boldsymbol{y}_J]$ is an observation matrix consisting of J observations. $\boldsymbol{A} = [\boldsymbol{a}_1, \ \boldsymbol{a}_2, \ \ldots, \ \boldsymbol{a}_R]$ is a feature matrix including feature vector bases, and R is the number of feature vectors. $\boldsymbol{X}$ is a mixing matrix to reproduce the observation matrix $\boldsymbol{Y}$ by $\boldsymbol{y}_j \simeq \sum_{r=1}^R \boldsymbol{a}_r x_{rj}$. The goal of NMF is to obtain $\boldsymbol{A}$ and $\boldsymbol{X}$ from the observation matrix $\boldsymbol{Y}$ with a given parameter R. For example, we let $R \leq J \ll I$ in BSS problems [18]. In this case, we want

to find R latent source signals from J mixed observations. The NMF gives $\boldsymbol{A}$ as an estimator of the latent source signals. In case of extracting parts of face images [62], I and J are respectively the number of pixels and images, then the NMF represents each face image by the linear combination of R nonnegative parts. In case of clustering tasks [106], $\boldsymbol{A}$ expresses the set of centroids of cluster, and $\boldsymbol{X}$ represents the weight parameters of clusters. In this way, the NMF is a high-potential technique and it can be applied in a wide area of a real world data analysis.

NMF has a long history from 1971 in chemometrics under the name of "self modeling curve resolution" [61]. In early works, it had been researched under the name of "positive matrix factorization" [78] in the middle of 1990s. It becomes more widely known as "nonnegative matrix factorization" after the publication on the journal of Nature in 1999 [62].

In this section I introduce several basic algorithms for NMF. The standard NMF criterion can be given by

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2, \\ &\text{subject to} \quad \boldsymbol{A}\geq 0,\ \boldsymbol{X}\geq 0. \end{aligned} \tag{3.40}$$

Its criterion is very simple; however, this problem is not convex and its solution is not unique. Thus, many algorithms have been studied and the researches of solution algorithm of NMF are still going now. In this section, I introduce several standard algorithms to solve the NMF problem.

ALS algorithm

In this section, I introduce the most standard algorithm for NMF which is called the alternating least squares (ALS) algorithm [26]. First we separate the NMF problem (3.40) into two sub-problems as follows:

$$\boldsymbol{A} \leftarrow \underset{\boldsymbol{A}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2 \text{ s.t. } \boldsymbol{A}\geq 0, \tag{3.41}$$

$$\boldsymbol{X} \leftarrow \underset{\boldsymbol{X}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2 \text{ s.t. } \boldsymbol{X}\geq 0. \tag{3.42}$$

Each problem is convex and have a unique solution. The procedure to solve the sub-problem for $\boldsymbol{A}$ consists of only two steps as follow:

$$\boldsymbol{A} \leftarrow \underset{\boldsymbol{A}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2; \tag{3.43}$$

$$\boldsymbol{A} \leftarrow [\boldsymbol{A}]_+; \tag{3.44}$$

This procedure does not provide strict solution of (3.41); however, it provides an approximated solution of (3.41) and performs fast calculation. In a similar way, we can solve the sub-problem for $\boldsymbol{X}$ as follow:

$$\boldsymbol{X} \leftarrow \underset{\boldsymbol{X}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2; \tag{3.45}$$

$$\boldsymbol{X} \leftarrow [\boldsymbol{X}]_+; \tag{3.46}$$

Algorithm 8 ALS-NMF algorithm

1: **Input:** $\boldsymbol{Y}$, R
2: **Initialize:** $\boldsymbol{X}$ randomly
3: **repeat**
4: $\quad \boldsymbol{A} \leftarrow [(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y}^T]^T$;
5: $\quad \boldsymbol{A} \leftarrow [\boldsymbol{A}]_+$;
6: $\quad \boldsymbol{X} \leftarrow (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$;
7: $\quad \boldsymbol{X} \leftarrow [\boldsymbol{X}]_+$;
8: **until** convergence
9: **Output:** $\boldsymbol{A}$, $\boldsymbol{X}$

The update procedures for $\boldsymbol{A}$ and $\boldsymbol{X}$ include standard least squares problems (3.43) and (3.45). Both problems can be solved by the same calculation method in principle. First the objective function can be transformed as

$$\begin{aligned} L :&= \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}||_F^2 \\ &= \frac{1}{2}||\boldsymbol{Y}^T - \boldsymbol{X}^T\boldsymbol{A}^T||_F^2 \\ &= \frac{1}{2}\mathrm{tr}[(\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X})^T(\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X})] \\ &= \frac{1}{2}\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{Y}) - \mathrm{tr}(\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Y}) + \frac{1}{2}\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}). \end{aligned} \tag{3.47}$$

Thus the stationary points of $\boldsymbol{X}$ can be founded by equating the gradient to 0:

$$\frac{\partial L}{\partial \boldsymbol{X}} = -\boldsymbol{A}^T\boldsymbol{Y} + \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X} = 0. \tag{3.48}$$

Thus, the solution is given by

$$\boldsymbol{X} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y}. \tag{3.49}$$

In a same way, the solution of $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = [(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y}^T]^T. \tag{3.50}$$

Finally, the ALS algorithm is summarized as Algorithm 8.

Multiplicative update rule

The ALS-NMF algorithm includes an inverse problem and thresholding for nonnegativity which are not stable. In practice, ALS algorithm do not converge, sometimes. The multiplicative-NMF algorithm [62] is proposed to update them by multiplying each element of $\boldsymbol{A}$ or $\boldsymbol{X}$ by some nonnegative value. This update rule is appropriate for NMF since it does not need any thresholding. The multiplicative-NMF algorithm is summarized as Algorithm 9.

Algorithm 9 Multiplicative-NMF algorithm

1: **Input:** $\boldsymbol{Y}$, R
2: **Initialize:** $\boldsymbol{X}$ randomly
3: **repeat**
4: $\quad \boldsymbol{A} \leftarrow \boldsymbol{A} \circledast (\boldsymbol{Y}\boldsymbol{X}^T) \oslash (\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^T)$;
5: $\quad \boldsymbol{X} \leftarrow \boldsymbol{X} \circledast (\boldsymbol{A}^T\boldsymbol{Y}) \oslash (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X})$;
6: **until** convergence
7: **Output:** $\boldsymbol{A}$, $\boldsymbol{X}$

HALS algorithm

Finally, I introduce the hierarchical alternating least squares (HALS) algorithm [24] for NMF. The HALS is related to ALS; however it does not include an inverse problem and is stable. The NMF model can be rewritten by

$$\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{B}^T = \sum_{r=1}^{R} \boldsymbol{a}_r \boldsymbol{b}_r^T, \tag{3.51}$$

where $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_R] \in \mathbb{R}_+^{J \times R}$ is equivalent to $\boldsymbol{X}^T$. In HALS algorithm, we consider a local-problem for $\boldsymbol{a}_r$ and $\boldsymbol{b}_r$ as follow:

$$\text{minimize} \, ||\boldsymbol{Y}_r - \boldsymbol{a}_r \boldsymbol{b}_r^T||_F^2, \text{ s.t. } \boldsymbol{a}_r \geq 0, \ \boldsymbol{b}_r \geq 0, \ ||\boldsymbol{a}_r|| = 1, \tag{3.52}$$

where $\boldsymbol{Y}_r := \boldsymbol{Y} - \sum_{k \neq r} \boldsymbol{a}_k \boldsymbol{b}_k^T$. We iterate to solve this local-problem for $r = 1, ..., R$ until convergence. Thus, we need update rules for $\boldsymbol{a}_r$ and $\boldsymbol{b}_r$. Since this local-problem (3.52) can be regarded as a nonnegativity constrained version of one-rank tSVD problem (3.22), then the update rules are given by

$$\boldsymbol{a}_r \leftarrow [\boldsymbol{Y}_r \boldsymbol{b}_r]_+; \tag{3.53}$$

$$\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||; \tag{3.54}$$

$$\boldsymbol{b}_r \leftarrow [\boldsymbol{Y}_r^T \boldsymbol{a}_r]_+; \tag{3.55}$$

Finally, the HALS-NMF algorithm is summarized as Algorithm 10. The HALS algorithm is similar to the deflation based algorithm for tSVD/PMD. The deflation based algorithm strictly solve one-rank approximation problem to repeat the updates inside the for-loop, one by one. On the other hand, the HALS algorithm has the repeat outside the for-loop. Thus, HALS can obtain more global solution than the deflation based algorithm. In fact, orthogonality is necessary to use deflation based algorithm properly. From this reason, the deflation based algorithm of PMD can not solve overall problem, but it solves just local-problem one by one using the matrix deflation technique. The HALS algorithm may have a possibility of improvement of PMD.

Algorithm 10 HALS-NMF algorithm

Input: $\boldsymbol{Y}$, R
Initialize: $[\boldsymbol{a}_r, \boldsymbol{b}_r]_{r=1}^{R}$ randomly
$\boldsymbol{E} \leftarrow \boldsymbol{Y} - \sum_{r=1}^{R} \boldsymbol{a}_r \boldsymbol{b}_r^T$;
repeat
  for $r = 1, ..., R$ **do**
    $\boldsymbol{Y}_r \leftarrow \boldsymbol{E} + \boldsymbol{a}_r \boldsymbol{b}_r^T$;
    $\boldsymbol{a}_r \leftarrow [\boldsymbol{Y}_r \boldsymbol{b}_r]_+$;
    $\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||$;
    $\boldsymbol{b}_r \leftarrow [\boldsymbol{Y}_r^T \boldsymbol{a}_r]_+$;
    $\boldsymbol{E} \leftarrow \boldsymbol{Y}_r - \boldsymbol{a}_r \boldsymbol{b}_r^T$;
  end for
until convergence
Output: $[\boldsymbol{a}_r, \boldsymbol{b}_r]_{r=1}^{R}$

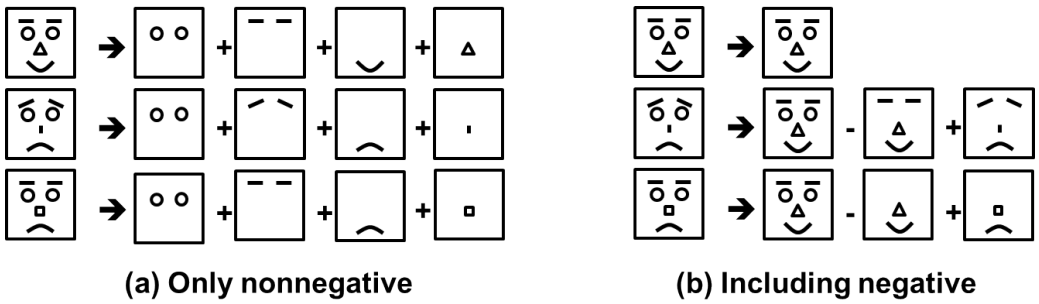


Figure 3.1: Comparison between only nonnegative factors and including negative factors

3.1.4 Why is NMF important and well-researched?

In this section, I discuss the importance of nonnegativity constraint. In the NMF, an observation is expressed by $\boldsymbol{y}_j \simeq \sum_{r=1}^{R} x_{rj} \boldsymbol{a}_r$, where $x_{rj} \geq 0$ and $\boldsymbol{a}_r \geq 0$. We can not cancel out any wasted components because any subtraction is not allowed in the NMF. As a result, the NMF provides local independent features $\{\boldsymbol{a}_r\}$. Figure 3.1 shows two examples of the facial-parts representation. If the subtraction is allowed, these faces would be separated into meaningless parts (Fig. 3.1 (b)). On the other hand, NMF can separate faces into the meaningful facial parts (Fig. 3.1 (a)).

Furthermore, nonnegativity constraint is important to get nonnegative component because almost all components of nonnegative data should be nonnegative. For example, image, video, probability and stock value do not include any negative values, so their parts souled be also nonnegative.

However, an optimization criterion of the NMF (3.40) is not convex and its solution is not unique. Thus, many algorithms have been studied and the researches of solution algorithm of NMF are still going now. The criterion is often based on the minimization of Frobenius norm; however, the uses of

many divergence measures have been proposed such as KL divergence [62], Csiszar's divergence [23], α- and β-divergences [25], Itakura-Saito divergence [34], and AB-divergence [20].

3.1.5 Combined methods of several constraints

In this section, I introduce several approaches to combine two or more constraints for matrix decomposition such as a combination of sparsity and nonnegativity.

Sparse NMF

A combination of sparsity and nonnegativity has been firstly proposed by Hoyer et al. [48] in 2002. After that, the algorithms for sparse NMF have been studied by many researchers [49, 94, 22, 55, 57, 40, 39]. In this section, I introduce two methods for the sparse NMF.

The first method is the regularized alternating least squares (RALS) algorithm [22]. The RALS algorithm prevents instability of the ALS algorithm to add some constraint terms for regularization and sparsity. These terms work to improve the uniqueness of NMF. The criterion of the RALS algorithm is given by

$$\begin{aligned} \text{minimize} \quad & D_F^{(\alpha)}(\boldsymbol{Y}||\boldsymbol{AX}) := \frac{1}{2}||\boldsymbol{W}^{-\frac{1}{2}}(\boldsymbol{Y}-\boldsymbol{AX})||_F^2 + \alpha_{A_s}||\boldsymbol{A}||_1 + \alpha_{X_s}||\boldsymbol{X}||_1 \\ & \qquad + \frac{\alpha_{A_r}}{2}||\boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{AL}_A||_F^2 + \frac{\alpha_{X_r}}{2}||\boldsymbol{L}_X\boldsymbol{X}||_F^2, \\ \text{subject to} \quad & \boldsymbol{A} \geq 0,\ \boldsymbol{X} \geq 0, \end{aligned} \tag{3.56}$$

where $\boldsymbol{W} \in \mathbb{R}^{I\times I}$ is a symmetric positive definite weighting matrix, α_{A_s} and α_{X_s} are trade-off parameters to tune sparsity level, α_{A_r} and α_{X_r} are trade-off parameters for regularization, and $\boldsymbol{L}_A$ and $\boldsymbol{L}_X$ are regularization matrices to enforce a certain application-dependent characteristics of the solution. The first term of the objective function can be transformed as

$$||\boldsymbol{W}^{-\frac{1}{2}}(\boldsymbol{Y}-\boldsymbol{AX})||_F^2 = \mathrm{tr}[(\boldsymbol{Y}-\boldsymbol{AX})^T\boldsymbol{W}^{-1}(\boldsymbol{Y}-\boldsymbol{AX})]. \tag{3.57}$$

If we set $\boldsymbol{W}$ to be covariance matrix of $\boldsymbol{Y}$, then it can be regarded as the Mahalanobis distance. In the RALS algorithm, problem (3.56) is solved by the following update rules:

$$\boldsymbol{A} \leftarrow (\boldsymbol{YX}^T - \alpha_{A_s}\boldsymbol{WS}_A + \alpha_{A_r}\boldsymbol{AL}_A\boldsymbol{L}_A^T)(\boldsymbol{XX}^T + \alpha_{A_r}\boldsymbol{L}_A\boldsymbol{L}_A^T)^\dagger, \tag{3.58}$$

$$\boldsymbol{X} \leftarrow (\boldsymbol{A}^T\boldsymbol{W}^{-1}\boldsymbol{A}^T + \alpha_{X_r}\boldsymbol{L}_X^T\boldsymbol{L}_X)^\dagger(\boldsymbol{A}^T\boldsymbol{W}^{-1}\boldsymbol{Y} - \alpha_{X_s}\boldsymbol{S}_X + \alpha_{X_r}\boldsymbol{L}_X^T\boldsymbol{L}_X\boldsymbol{X}), \tag{3.59}$$

where $\boldsymbol{S}_A := \mathrm{sign}(\boldsymbol{A})$ and $\boldsymbol{S}_X := \mathrm{sign}(\boldsymbol{X})$.

The second method is the Kim&Park's sparse NMF (KP-NMF) algorithm [55, 57]. Since the KP-NMF algorithm was proposed to apply to clustering task, the left-factor matrix is regularized as centroids and the right-factor matrix is constrained to be sparse for the weighting parameter vectors.

Then the criterion is given by

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2+\eta||\boldsymbol{A}||_F^2+\beta\sum_{j=1}^{J}||\boldsymbol{x}_j||_1^2 \\ \text{subject to} \quad & \boldsymbol{A}\geq 0,\ \boldsymbol{X}\geq 0. \end{aligned} \tag{3.60}$$

This problem can be separated into two sub-problems as follow:

$$\boldsymbol{X}\leftarrow\underset{\boldsymbol{X}}{\operatorname{argmin}}\left|\left|\begin{pmatrix}\boldsymbol{A}\\ \sqrt{\beta}\mathbf{1}_R^T\end{pmatrix}\boldsymbol{X}-\begin{pmatrix}\boldsymbol{Y}\\ \mathbf{0}_J^T\end{pmatrix}\right|\right|_F^2,\ \text{s.t. } \boldsymbol{X}\geq 0, \tag{3.61}$$

$$\boldsymbol{A}\leftarrow\underset{\boldsymbol{A}}{\operatorname{argmin}}\left|\left|\begin{pmatrix}\boldsymbol{X}^T\\ \sqrt{\eta}\boldsymbol{I}_R^T\end{pmatrix}\boldsymbol{A}^T-\begin{pmatrix}\boldsymbol{Y}^T\\ \mathbf{0}_{R\times I}^T\end{pmatrix}\right|\right|_F^2,\ \text{s.t. } \boldsymbol{A}\geq 0, \tag{3.62}$$

where $\mathbf{1}_R$ is the R-dimensional vector of which all elements are 1, $\mathbf{0}_J$ is the J-dimensional vector of which all elements are 0, and $\mathbf{0}_{R\times I}$ is an $(R\times J)$-matrix of which all elements are 0. Both sub-problems can be characterized as the nonnegative least squares (NLS) problem. Solution methods for NLS problem are studied by many researches [10, 100, 56, 114]. The KP-NMF algorithm employs the active-set algorithm from [56].

Smooth NMF

Temporal and spatial smoothness is focused in wide area of data analysis such as visual-audio data and brain signal processing [13, 116, 117, 32, 33]. Although real world data is often noisy, if we treat temporally or spatially smooth signals (e.g., natural image data, alpha brain wave, and financial data), smooth NMF [13, 116, 117, 32, 33, 115] is useful and effective. There are many methods for the smooth NMF and they can be separated into two approaches, mainly. The first one is to add some smooth constraint term into the NMF criterion. Chen et al. [13] proposed to add a temporal smoothness constraint and a spatial decorrelation constraint terms into the Frobenius norm and the Kullback-Leibler (KL) divergence based NMF for EEG analysis. Zdunek et al. [116, 117] proposed to add the Gibbs regularization term for smooth NMF. Drakakis et al. [32] proposed to add a sparsenss constraint for mixing matrix and a smoothness constraint for feature matrix into the Frobenius norm and the KL divergence based NMF for analysis of financial data. Essid et al. [33] applied the KL divergence based smooth NMF to audiovisual document structuring. In this section, I introduce two methods for the smooth NMF.

The first method is the Gibbs regularized NMF [116, 117]. The Gibbs regularization is proposed for robustness to Gaussian noise, and its cost function can be expressed as

$$\Psi:=||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2+2\alpha U(\boldsymbol{X}), \tag{3.63}$$

where $U(\boldsymbol{X})$ is a total energy function that measures the total roughness of $\boldsymbol{X}$, and α is a regularization parameter. The stationary points of Ψ can be derived from the gradients of Ψ with respect to $\boldsymbol{X}$ and

$\boldsymbol{A}$. Its conditions are given by

$$\nabla_X \Psi = 2\boldsymbol{A}^T(\boldsymbol{AX} - \boldsymbol{Y}) + 2\alpha\nabla_X U(\boldsymbol{X}) = \boldsymbol{0}, \tag{3.64}$$

$$\nabla_A \Psi = (\boldsymbol{AX} - \boldsymbol{Y})\boldsymbol{X}^T = \boldsymbol{0}. \tag{3.65}$$

From (3.64) and (3.65), we have

$$\frac{[\boldsymbol{A}^T\boldsymbol{Y} - \alpha\nabla_X U(\boldsymbol{X})]_{rj}}{[\boldsymbol{A}^T\boldsymbol{AX}]_{rj}} = 1, \ \frac{[\boldsymbol{YX}^T]_{ir}}{[\boldsymbol{AXX}^T]_{ir}} = 1. \tag{3.66}$$

Using multiplicative update rules, we obtain the Gibbs regularized multiplicative NMF algorithm as follows:

$$\boldsymbol{X} \leftarrow \boldsymbol{X} \circledast [\boldsymbol{A}^T\boldsymbol{Y} - \alpha\nabla_X U(\boldsymbol{X})]_+ \oslash [\boldsymbol{A}^T\boldsymbol{AX}], \tag{3.67}$$

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast [\boldsymbol{YX}^T] \oslash [\boldsymbol{AXX}^T], \tag{3.68}$$

$$\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||_1, \tag{3.69}$$

Another important point is how to decide the function $U(\boldsymbol{X})$. Many candidates can be considered in [45, 42, 38, 6]. In [117], the green function is employed. Thus, $U(\boldsymbol{X})$ is given by

$$U(\boldsymbol{X}) = \sum_{r=1}^{R}\sum_{j=1}^{J}\left[\sum_{l\in S_j} w_{jl}\sigma \log\left\{\cosh\left(\frac{x_{rj} - x_{rl}}{\sigma}\right)\right\}\right], \tag{3.70}$$

where w_{jl} is a weight parameter, and σ is a scaling parameter. Furthermore, $\nabla_X U(\boldsymbol{X})$ is given by

$$[\nabla_X U(\boldsymbol{X})]_{rj} = \sum_{l\in S_j} w_{jl}\tanh\left(\frac{x_{rj} - x_{rl}}{\sigma}\right). \tag{3.71}$$

The second method is the Essid's smooth NMF [33]. Let $\widehat{\boldsymbol{Y}} := \boldsymbol{AX}$, its criterion is given by

$$\text{minimize} \quad D_{KL}(\boldsymbol{Y}||\widehat{\boldsymbol{Y}}) + \beta S(\boldsymbol{X}), \ \text{s.t.} \ \boldsymbol{A} \geq 0, \ ||\boldsymbol{a}_r|| = 1, \ \boldsymbol{X} \geq 0, \tag{3.72}$$

where β is a trade-off parameter for smoothness, $D_{KL}(\boldsymbol{Y}||\widehat{\boldsymbol{Y}})$ is the KL-divergence defined as

$$D_{KL}(\boldsymbol{Y}||\widehat{\boldsymbol{Y}}) := \sum_{i=1}^{I}\sum_{j=1}^{J}\left(y_{ij}\log\frac{y_{ij}}{\hat{y}_{ij}} - x_{ij} + y_{ij}\right), \tag{3.73}$$

and the smoothness constraint function $S(\boldsymbol{X})$ is defined by

$$S(\boldsymbol{X}) := \frac{1}{2}\sum_{r=1}^{R}\sum_{j=2}^{J}(x_{rj} - x_{r(j-1)})^2. \tag{3.74}$$

We define $\boldsymbol{L} \in \mathbb{R}^{J\times(J-1)}$ as

$$l_{mn} := \begin{cases} -1 & \text{if } m = n \\ 1 & \text{if } m - n = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{3.75}$$

then $S(\boldsymbol{X})$ can be expressed by

$$S(\boldsymbol{X}) = \frac{1}{2}\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{X}). \tag{3.76}$$

If $J = 5$, $\boldsymbol{L}$ can be given by

$$\boldsymbol{L} := \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.77}$$

Update rule for $\boldsymbol{X}$ can be given by

$$\boldsymbol{X} \leftarrow \frac{1}{2}\left[\sqrt{\boldsymbol{X}_b \circledast \boldsymbol{X}_b + 4\boldsymbol{X}_a} - \boldsymbol{X}_b\right]\boldsymbol{D}_\beta^{-1}, \tag{3.78}$$

where

$$\boldsymbol{X}_a := \boldsymbol{\Psi}\boldsymbol{D}_\beta, \tag{3.79}$$

$$\boldsymbol{X}_b := \boldsymbol{1}_{R\times J} + \boldsymbol{X}\boldsymbol{D}_\beta^{(2)}, \tag{3.80}$$

$$\boldsymbol{\Psi} := \boldsymbol{X} \circledast \boldsymbol{A}^T(\boldsymbol{Y} \oslash \widehat{\boldsymbol{Y}}), \tag{3.81}$$

$$\boldsymbol{D}_\beta := \mathrm{diag}(\boldsymbol{\beta}, 2\beta, \cdots, 2\beta, \beta) \in \mathbb{R}^{J\times J}, \tag{3.82}$$

$$[\boldsymbol{D}_\beta^{(2)}]_{mn} := \begin{cases} \beta & \text{if } |m-n| = 1 \\ 0 & \text{otherwise} \end{cases} \in \mathbb{R}^{J\times J}. \tag{3.83}$$

If $J = 5$, $\boldsymbol{D}_\beta^{(2)}$ is given by

$$\boldsymbol{D}_\beta^{(2)} = \begin{pmatrix} 0 & \beta & 0 & 0 & 0 \\ \beta & 0 & \beta & 0 & 0 \\ 0 & \beta & 0 & \beta & 0 \\ 0 & 0 & \beta & 0 & \beta \\ 0 & 0 & 0 & \beta & 0 \end{pmatrix} \in \mathbb{R}^{J\times J}. \tag{3.84}$$

Update rule for $\boldsymbol{A}$ can be given by

$$\boldsymbol{A} \leftarrow \frac{1}{2\beta}\left[\sqrt{\boldsymbol{A}_b \circledast \boldsymbol{A}_b + 4\boldsymbol{A}_a} - \boldsymbol{A}_b\right]\boldsymbol{D}_s^{-1}, \tag{3.85}$$

$$\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r/||\boldsymbol{a}_r||, \tag{3.86}$$

where

$$\boldsymbol{A}_a := \beta \boldsymbol{\Phi} \boldsymbol{D}_s, \tag{3.87}$$

$$\boldsymbol{A}_b := \beta(\mathbf{1}_{I\times I} - \boldsymbol{I}_I)\boldsymbol{A}\boldsymbol{D}_s + \mathbf{1}_{I\times R}\boldsymbol{D}_x, \tag{3.88}$$

$$\boldsymbol{\Phi} := \boldsymbol{A} \circledast \{(\boldsymbol{Y} \oslash \widehat{\boldsymbol{Y}})\boldsymbol{X}^T\}, \tag{3.89}$$

$$[\boldsymbol{D}_s]_{mn} := \begin{cases} [\boldsymbol{X}^T\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{X}]_{mm} & \text{if } m = n \\ 0 & \text{otherwise} \end{cases} \in \mathbb{R}^{R\times R}, \tag{3.90}$$

$$[\boldsymbol{D}_x]_{mn} := \begin{cases} \sum_{j=1}^{J} x_{mj} & \text{if } m = n \\ 0 & \text{otherwise} \end{cases} \in \mathbb{R}^{R\times R}. \tag{3.91}$$

In summary, a standard approach of smooth NMF can be given by a combination of some divergence measure between $\boldsymbol{Y}$ and $\boldsymbol{AX}$, and some smooth constraint. In Section 3.1.6, I introduce a different approach for smooth NMF.

3.1.6 Function approximation based model

In this section, I review the smooth NMF using function approximation proposed in [115]. Please note that we impose a smoothness constraint to $\boldsymbol{A}$. This method is to approximate a feature vector $\boldsymbol{a}_r$ by the following model:

$$\boldsymbol{a}_r = \sum_{n=1}^{N} \boldsymbol{\phi}_n w_{nr}, \tag{3.92}$$

where $\{\boldsymbol{w}_{nr}\}$ are real-valued coefficients, and $\{\boldsymbol{\phi}_n\}$ are smooth basis functions (e.g., the Gaussian function). Let $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N] \in \mathbb{R}_+^{I\times N}$ and $\boldsymbol{W} = [w_{nr}] \in \mathbb{R}^{N\times R}$, then we have the following model for NMF:

$$\boldsymbol{Y} \simeq \boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}, \text{ s.t. } \boldsymbol{\Phi}\boldsymbol{W} \geq 0, \text{ and } \boldsymbol{X} \geq 0. \tag{3.93}$$

In this model, the feature matrix $\boldsymbol{A}$ is approximated by $\boldsymbol{\Phi}\boldsymbol{W}$. Instead the degree of freedom of estimation is decreased by function approximation $\boldsymbol{\Phi}\boldsymbol{W}$, we can obtain a smooth estimator. When the observed data $\boldsymbol{Y}$ includes some noise, this model can reduce affections of the noise. For optimization, we estimate two parameter matrices $\boldsymbol{W}$ and $\boldsymbol{X}$ since $\boldsymbol{\Phi}$ is fixed.

How to Choose $\boldsymbol{\Phi}$

In [115] it was proposed to use the Gaussian radial basis functions (GRBF) for $\boldsymbol{\Phi}$ with a standard deviation σ as

$$\boldsymbol{\Phi}(i, n) = \exp\left[-\frac{(i - n\Delta t)^2}{2\sigma^2}\right], \tag{3.94}$$

where Δt is an interval which satisfies $N = \lfloor (I-1)/\Delta t \rfloor + 1$ (see Fig. 3.2). When σ is large, the degree of freedom of representation will decrease but it is expected that the NMF becomes robust for

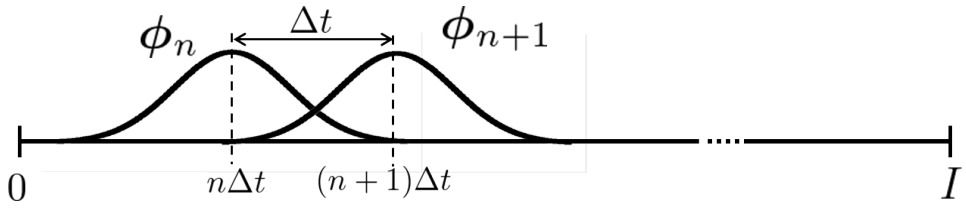


Figure 3.2: Basis function $\boldsymbol{\phi}_n$

noisy data at the same time. On the other hand, when σ is small, $\boldsymbol{\phi}_n$ will reach orthogonal bases and the degree of freedom of representation will increase but it is expected that the NMF becomes weak for noisy data. Thus, σ can be regarded as a trade-off parameter.

Algorithm

To estimate $\boldsymbol{W}$ and $\boldsymbol{X}$, the most popular criterion minimizes the Frobenius norm as

$$\underset{\boldsymbol{W},\,\boldsymbol{X}}{\text{minimize}}\ \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}||_F^2,\ \text{s.t.}\ \boldsymbol{\Phi}\boldsymbol{W}\geq 0,\ \boldsymbol{X}\geq 0. \tag{3.95}$$

Since this is not convex, we separate this criterion into the following two sub-problems and solve them, alternately and iteratively:

$$\underset{\boldsymbol{X}}{\text{minimize}}\ \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}||_F^2,\ \text{s.t.}\ \boldsymbol{X}\geq 0, \tag{3.96}$$

$$\underset{\boldsymbol{W}}{\text{minimize}}\ \frac{1}{2}||\boldsymbol{Y}-\boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}||_F^2,\ \text{s.t.}\ \boldsymbol{\Phi}\boldsymbol{W}\geq 0. \tag{3.97}$$

To solve (3.96), we can apply basic NMF approach such as the right-side update rule of ALS [26]. The regularized fast combinatorial nonnegative least squares (FC-NNLS) algorithm [100] is used. This is based on the active-set algorithm. To solve (3.97), we transform the objective function to the following vectorized form (defined by (3.104)):

$$\begin{aligned}&\frac{1}{2}||\boldsymbol{Y}-\boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}||_F^2=\frac{1}{2}||\bar{\boldsymbol{y}}-(\boldsymbol{X}^T\otimes\boldsymbol{\Phi})\bar{\boldsymbol{w}}||^2\\&=\frac{1}{2}\bar{\boldsymbol{y}}^T\bar{\boldsymbol{y}}-\bar{\boldsymbol{y}}^T(\boldsymbol{X}^T\otimes\boldsymbol{\Phi})\bar{\boldsymbol{w}}+\frac{1}{2}\bar{\boldsymbol{w}}^T(\boldsymbol{X}\boldsymbol{X}^T\otimes\boldsymbol{\Phi}^T\boldsymbol{\Phi})\bar{\boldsymbol{w}},\end{aligned} \tag{3.98}$$

where $\bar{\boldsymbol{w}}=\text{vec}(\boldsymbol{W})\in\mathbb{R}^{RN}$ and $\bar{\boldsymbol{y}}=\text{vec}(\boldsymbol{Y})\in\mathbb{R}^{IJ}$ are vectorized forms of the matrices $\boldsymbol{W}$ and $\boldsymbol{Y}$, respectively, and $\otimes$ stands for the Kronecker product. Finally, we solve the following quadratic programming (QP) problem:

$$\underset{\bar{\boldsymbol{w}}}{\text{minimize}}\ \frac{1}{2}\bar{\boldsymbol{w}}^T\boldsymbol{Q}\bar{\boldsymbol{w}}+\boldsymbol{c}^T\bar{\boldsymbol{w}},\ \text{s.t.}\ (\boldsymbol{I}_R\otimes\boldsymbol{\Phi})\bar{\boldsymbol{w}}\geq 0, \tag{3.99}$$

where $\boldsymbol{Q}=\boldsymbol{X}\boldsymbol{X}^T\otimes\boldsymbol{\Phi}^T\boldsymbol{\Phi}\in\mathbb{R}^{RN\times RN}$, $\boldsymbol{c}=-(\boldsymbol{X}\otimes\boldsymbol{\Phi}^T)\bar{\boldsymbol{y}}\in\mathbb{R}^{RN}$, and $\boldsymbol{I}_R\in\mathbb{R}^{R\times R}$ is the identity matrix. This method is called the GRBF-NMF.

Computational Issue

According to [115], the GRBF-NMF is robust in respect to noisy data. This optimization algorithm consists of the active set algorithm for $\boldsymbol{X}$ and the QP optimization for $\boldsymbol{W}$. Each algorithm is excellent as a strict single optimization method; however, it is not so effective to use as a step of alternating optimization algorithm. The reason is that the dimension of parameter space of the QP optimization would be large (i.e., it is RN) and the active-set method has to solve iteratively the least squares problem for evaluation of the optimality of current active-set. Thus, its computational cost becomes very high. The use of strict optimization methods is not indispensable for a step of iterative algorithm. A low cost step is better in many cases even though it does not give a strict solution but an approximated solution. Based on this philosophy, a new fast algorithm for the GRBF-NMF is proposed in Chapter 5.

3.2 Tensor algebra

We call a multi-dimensional array of numerical values as a tensor. An Nd-tensor (i.e., N-way array) has N indexes, for example,

- $N = 0 \rightarrow$ Scalar, e.g., x
- $N = 1 \rightarrow$ Vector, e.g., x_i
- $N = 2 \rightarrow$ Matrix, e.g., x_{ij}
- $N = 3 \rightarrow$ like a "Cube", e.g., x_{ijk}
- $\cdots$

Decompositions of higher-order tensors (i.e., N-way arrays with $N \geq 3$) have many applications [59] in psychometrics, chemometrics, signal processing, numerical linear algebra, computer vision, numerical analysis, data mining, neuroscience, graph analysis, and elsewhere. Higher-order (i.e., $N \geq 3$) tensors are denoted by underlined and boldface capital letters, e.g.,

$$\underline{\boldsymbol{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, \tag{3.100}$$

$$\underline{\boldsymbol{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}. \tag{3.101}$$

We denote these element-wise value by

$$y_{i_1 i_2 \cdots i_N}, \tag{3.102}$$

$$g_{r_1 r_2 \cdots r_N}. \tag{3.103}$$

Fig. 3.3 shows visualization of various orders of tensor data.

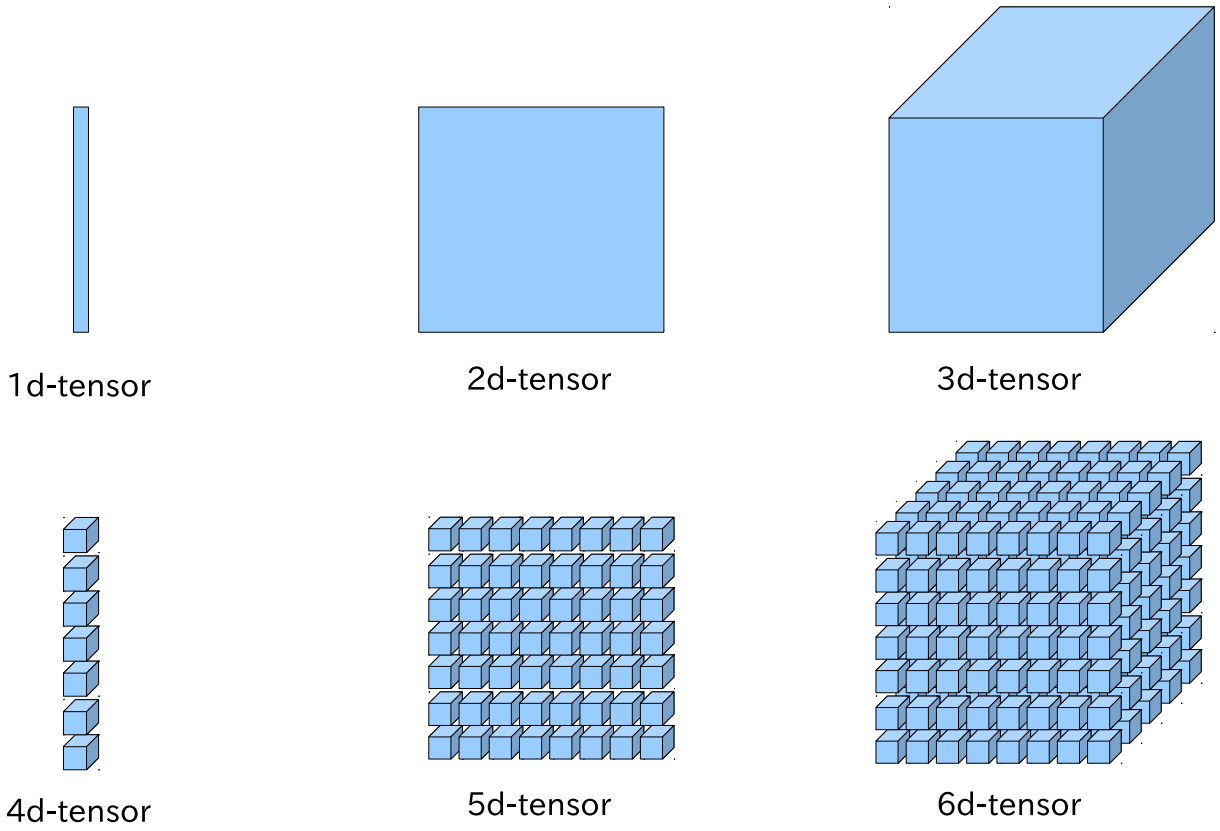


Figure 3.3: Tensors for various orders

Vectorization is the transformation from a tensor to a vector. To begin with, I introduce a vectorization of matrix. Vectorization of matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_J] \in \mathbb{R}^{I \times J}$ is expressed by

$$\mathrm{vec}(\boldsymbol{Y}) := \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_J \end{pmatrix} \in \mathbb{R}^{IJ}. \tag{3.104}$$

Fig. 3.4 shows a visualization of each-mode vectorization of a 3d-tensor. In regularly, the vectorization means the upper first-mode vectorization in Fig. 3.4. By using a notation of slice of tensor $[\underline{\boldsymbol{Z}}_{::k}] \in \mathbb{R}^{I \times J}$ which is k-th slice of tensor $\underline{\boldsymbol{Z}} \in \mathbb{R}^{I \times J \times K}$, the vectorization of tensor $\underline{\boldsymbol{Z}}$ is difined by

$$\mathrm{vec}(\underline{\boldsymbol{Z}}) := \mathrm{vec}([\underline{\boldsymbol{Z}}_{::1}, \underline{\boldsymbol{Z}}_{::2}, ..., \underline{\boldsymbol{Z}}_{::K}]) \in \mathbb{R}^{IJK}. \tag{3.105}$$

We have

$$||\boldsymbol{Y}||_F^2 = \sum_{i,j} y_{ij}^2 = \mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{Y}) = \mathrm{vec}(\boldsymbol{Y})^T\mathrm{vec}(\boldsymbol{Y}) = ||\mathrm{vec}(\boldsymbol{Y})||^2, \tag{3.106}$$

$$||\underline{\boldsymbol{Z}}||_F^2 = \sum_{i,j,k} z_{ijk}^2 = \mathrm{vec}(\underline{\boldsymbol{Z}})^T\mathrm{vec}(\underline{\boldsymbol{Z}}) = ||\mathrm{vec}(\underline{\boldsymbol{Z}})||^2. \tag{3.107}$$

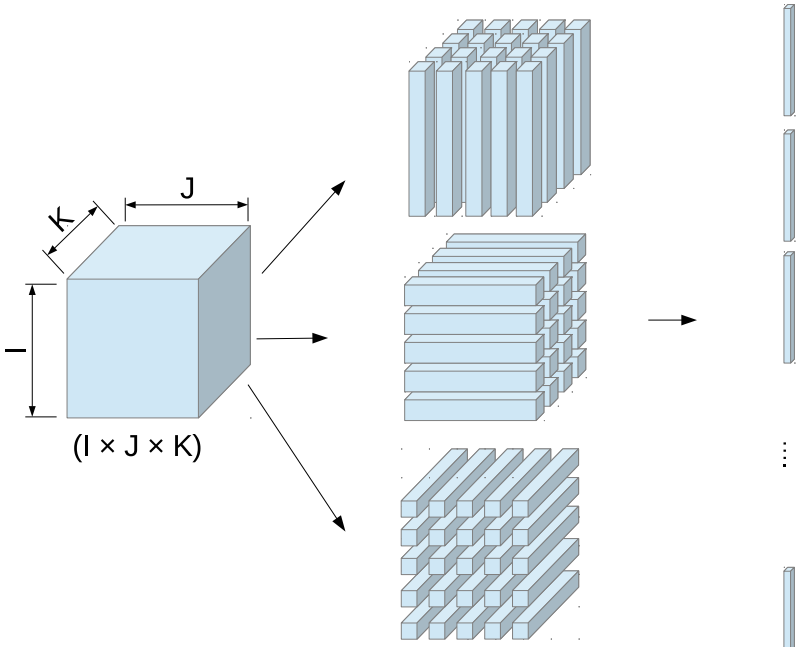


Figure 3.4: Vectorization of 3d-tensor

Matricization is transformation of a tensor into a matrix. The first-, second- and third-mode matricization of tensor $\underline{\boldsymbol{Y}} \in \mathbb{R}^{I\times J\times K}$ can be expressed by $\boldsymbol{Y}_{(1)} \in \mathbb{R}^{I\times JK}$, $\boldsymbol{Y}_{(2)} \in \mathbb{R}^{J\times IK}$, and $\boldsymbol{Y}_{(3)} \in \mathbb{R}^{K\times IJ}$, respectively. Fig. 3.5 shows a visualization of each-mode matricization of a 3d-tensor. Notation of a slice of tensor is often used, e.g., $[\underline{\boldsymbol{Y}}_{::k}] \in \mathbb{R}^{I\times J}$ is k-th slice of tensor $\underline{\boldsymbol{Y}}$. Using this notation, each-mode matricization can be defined as

$$\boldsymbol{Y}_{(1)} := [\mathrm{vec}(\underline{\boldsymbol{Y}}_{1::}),\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{2::}),\ ...,\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{I::})]^T \in \mathbb{R}^{I\times JK}, \tag{3.108}$$

$$\boldsymbol{Y}_{(2)} := [\mathrm{vec}(\underline{\boldsymbol{Y}}_{:1:}),\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{:2:}),\ ...,\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{:J:})]^T \in \mathbb{R}^{J\times KI}, \tag{3.109}$$

$$\boldsymbol{Y}_{(3)} := [\mathrm{vec}(\underline{\boldsymbol{Y}}_{::1}),\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{::2}),\ ...,\ \mathrm{vec}(\underline{\boldsymbol{Y}}_{::K})]^T \in \mathbb{R}^{K\times IJ}. \tag{3.110}$$

In this way, three-order tensor has three ways of matricization. We have

$$||\underline{\boldsymbol{Y}}||_F^2 = \mathrm{tr}(\boldsymbol{Y}_{(1)}\boldsymbol{Y}_{(1)}^T) = ||\boldsymbol{Y}_{(1)}||_F^2 \tag{3.111}$$

$$= \mathrm{tr}(\boldsymbol{Y}_{(2)}\boldsymbol{Y}_{(2)}^T) = ||\boldsymbol{Y}_{(2)}||_F^2 \tag{3.112}$$

$$= \mathrm{tr}(\boldsymbol{Y}_{(3)}\boldsymbol{Y}_{(3)}^T) = ||\boldsymbol{Y}_{(3)}||_F^2. \tag{3.113}$$

n-mode product of a tensor $\underline{\boldsymbol{Z}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and a matrix $\boldsymbol{V} \in \mathbb{R}^{R\times I_n}$ is defined by

$$[\underline{\boldsymbol{Z}} \times_n \boldsymbol{V}]_{i_1 i_2\cdots i_{n-1} r i_{n+1}\cdots i_N} := \sum_{i_n=1}^{I_n} (v_{r i_n})(z_{i_1 i_2\cdots i_{n-1} i_n i_{n+1}\cdots i_N}). \tag{3.114}$$

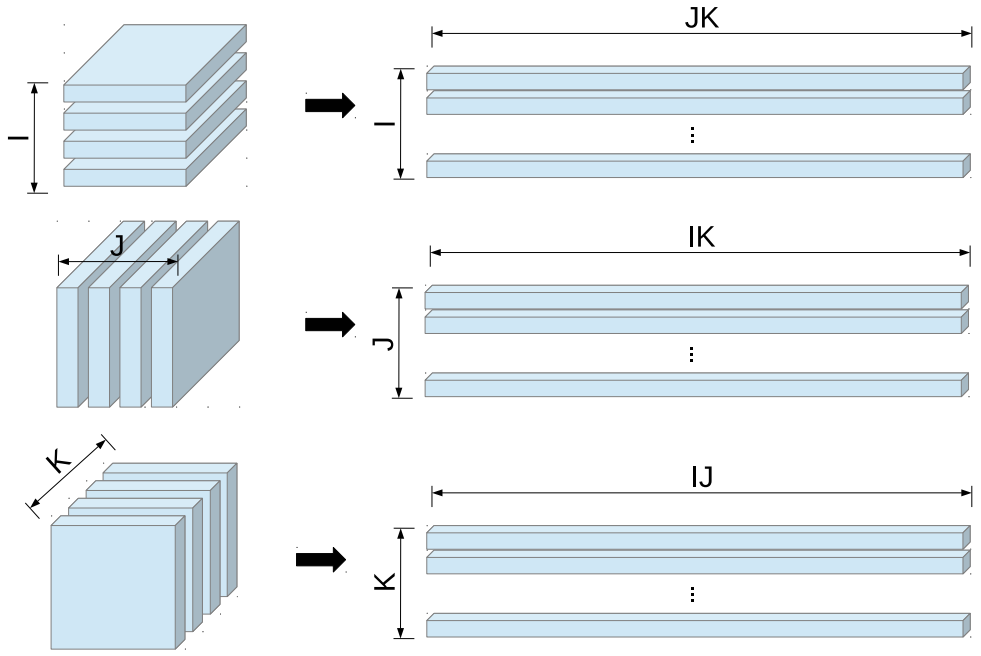


Figure 3.5: Matricization of each mode

For example, when we consider a 3d-tensor $\underline{\boldsymbol{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, and the first-mode product of $\underline{\boldsymbol{G}}$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{I \times R_1}$ is denoted by $\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \in \mathbb{R}^{I \times R_2 \times R_3}$. Its element-wise value is given by

$$[\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}]_{ir_2r_3} = \sum_{r_1=1}^{R_1} a_{ir_1} g_{r_1r_2r_3}. \tag{3.115}$$

Fig. 3.6 shows a visualization of first-mode tensor product. In a matricized form, we have

$$[\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}]_{(1)} = \boldsymbol{A}\boldsymbol{G}_{(1)}, \tag{3.116}$$

and

$$||\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}||_F^2 = \mathrm{tr}(\boldsymbol{A}\boldsymbol{G}_{(1)}\boldsymbol{G}_{(1)}^T\boldsymbol{A}^T). \tag{3.117}$$

All mode product of a tensor $\underline{\boldsymbol{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and a set of matrices $\{\boldsymbol{A} \in \mathbb{R}^{I \times R_1}, \boldsymbol{B} \in \mathbb{R}^{J \times R_2}, \boldsymbol{C} \in \mathbb{R}^{K \times R_3}\}$ is denoted by $\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \in \mathbb{R}^{I \times J \times K}$. We have

$$(\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C})_{(1)} = \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T), \tag{3.118}$$

$$(\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C})_{(2)} = \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C}^T \otimes \boldsymbol{A}^T), \tag{3.119}$$

$$(\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C})_{(3)} = \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B}^T \otimes \boldsymbol{A}^T). \tag{3.120}$$

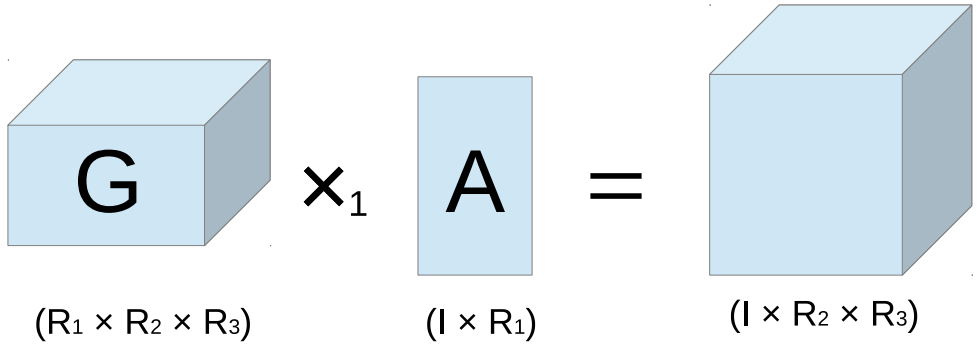


Figure 3.6: First mode tensor product

Figs. 3.7 and 3.8 show visualizations of all mode product in tensor and matricized forms. Moreover we have

$$||\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}||_F^2 = \mathrm{tr}(\boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C}^T\boldsymbol{C} \otimes \boldsymbol{B}^T\boldsymbol{B})\boldsymbol{G}_{(1)}^T\boldsymbol{A}^T), \tag{3.121}$$

$$= \mathrm{tr}(\boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C}^T\boldsymbol{C} \otimes \boldsymbol{A}^T\boldsymbol{A})\boldsymbol{G}_{(2)}^T\boldsymbol{B}^T), \tag{3.122}$$

$$= \mathrm{tr}(\boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B}^T\boldsymbol{B} \otimes \boldsymbol{A}^T\boldsymbol{A})\boldsymbol{G}_{(3)}^T\boldsymbol{C}^T). \tag{3.123}$$

If each matrices are orthonormal (i.e., $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_{R_1}$, $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_{R_2}$, $\boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}_{R_3}$), thus we have

$$||\underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}||_F^2 = ||\underline{\boldsymbol{G}}||_F^2. \tag{3.124}$$

3.3 CP decomposition

In this section, I introduce a practical tensor decomposition technique. First, we define a one-rank tensor that it can be written as the outer product of N vectors, i.e.,

$$\underline{\boldsymbol{U}} = \boldsymbol{u}^{(1)} \circ \boldsymbol{u}^{(2)} \circ \cdots \circ \boldsymbol{u}^{(N)}. \tag{3.125}$$

The CP (canonical polyadic) decomposition factorizes a tensor into a sum of one-rank tensors. The CP model with $N = 3$ (i.e., three way tensor decomposition) is given by

$$\underline{\boldsymbol{Y}} \simeq \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!], \tag{3.126}$$

where $\boldsymbol{A} := [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_R]$, $\boldsymbol{B} := [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_R]$, and $\boldsymbol{C} := [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_R]$. Fig. 3.9 shows a visualization of the CP model. The CP model is also called PARAFAC (parallel factor analysis) [43] or CANDECOMP (canonical decomposition) [11]. The CP model can be considered as the low rank approximation of tensor $\underline{\boldsymbol{Y}}$, when R is smaller than the rank of tensor $\underline{\boldsymbol{Y}}$. Thus, the CP decomposition can be considered as an extension of tSVD. We can denote the matricized forms of CP model as

$$\boldsymbol{Y}_{(1)} \simeq \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T, \tag{3.127}$$

$$\boldsymbol{Y}_{(2)} \simeq \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^T, \tag{3.128}$$

$$\boldsymbol{Y}_{(3)} \simeq \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^T. \tag{3.129}$$

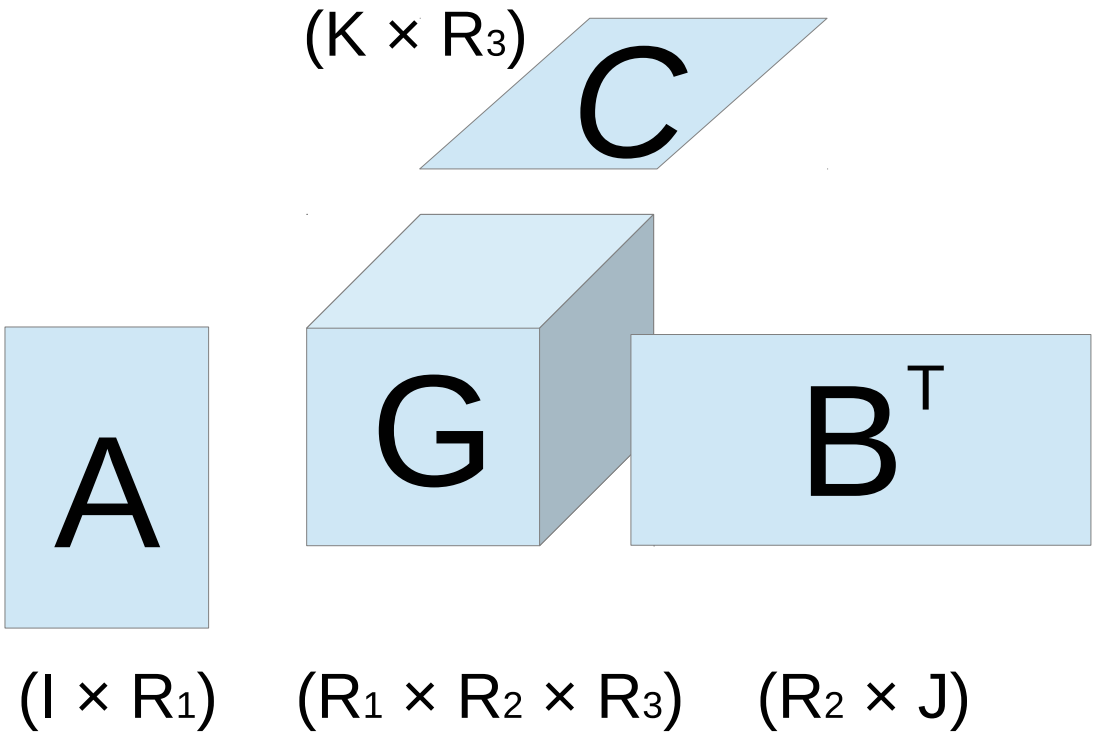


Figure 3.7: All mode product

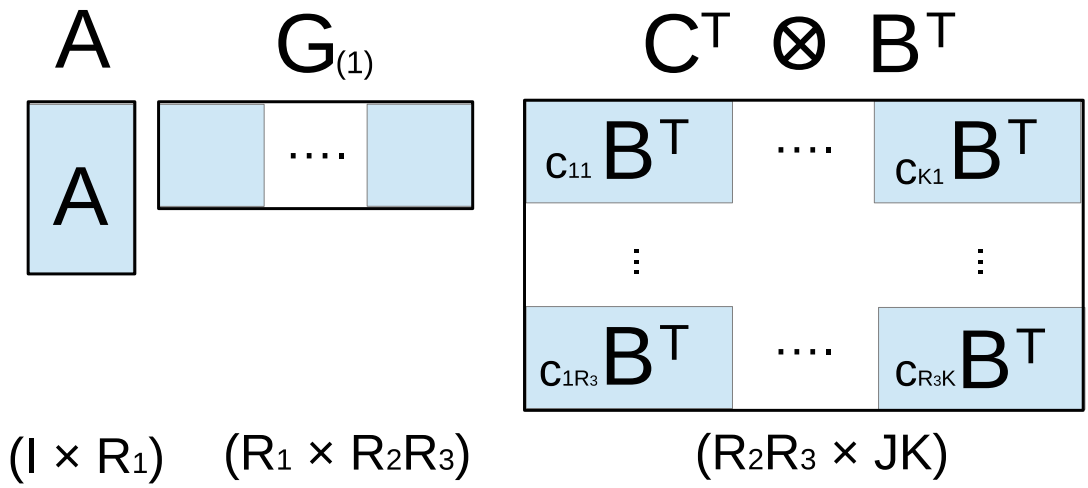


Figure 3.8: Matricization of all mode product

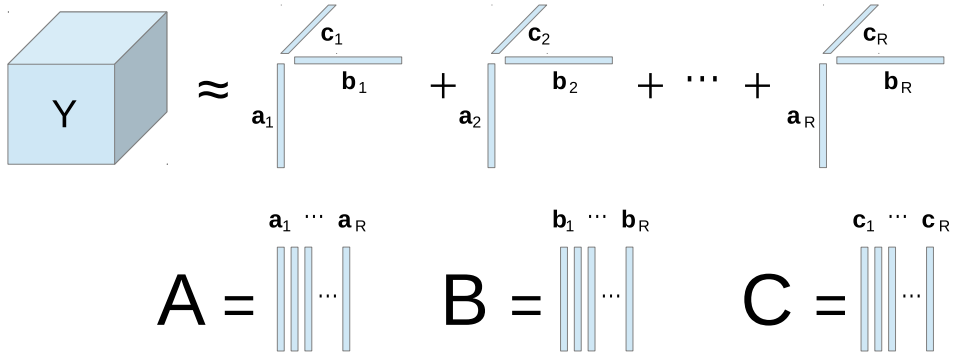


Figure 3.9: The CP decomposition

Criterion for the CP decomposition is given by

$$\text{minimize} \, ||\underline{\boldsymbol{Y}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]||_F^2. \tag{3.130}$$

When $N \geq 3$, the CP decomposition with $R = \text{rank}(\underline{\boldsymbol{Y}})$ has an unique solution without any constraints unlike the SVD. But, when $R < \text{rank}(\underline{\boldsymbol{Y}})$, the CP decomposition can be considered as low-rank approximation of $\underline{\boldsymbol{Y}}$. It is considered that the best R-rank approximation does not exist, uniquely.

3.3.1 ALS algorithm

The ALS algorithm for CP decomposition is originally proposed in [43, 11]. In this method, we impose the normalization condition to factor matrices and add scalar parameters $\boldsymbol{g}_r$, thus the criterion is given by

$$\begin{aligned} \text{minimize} \quad & \left|\left| \underline{\boldsymbol{Y}} - \sum_{r=1}^{R} g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r \right|\right|_F^2, \\ \text{subject to} \quad & ||\boldsymbol{a}_r|| = ||\boldsymbol{b}_r|| = ||\boldsymbol{c}_r|| = 1. \end{aligned} \tag{3.131}$$

g_r can be regarded as a singular value and $\boldsymbol{a}_r$, $\boldsymbol{b}_r$, and $\boldsymbol{c}_r$ can be regarded as singular vectors. Let $\boldsymbol{D}_g := \text{diag}(g_1, ..., g_R)$, the cost function of (3.131) can be transformed as

$$\begin{aligned} L :&= ||\boldsymbol{Y}_{(1)} - \boldsymbol{A}\boldsymbol{D}_g(\boldsymbol{C} \odot \boldsymbol{B})^T||_F^2 \\ &= \text{tr}(\boldsymbol{Y}_{(1)}\boldsymbol{Y}_{(1)}^T) - 2\text{tr}(\boldsymbol{A}\boldsymbol{D}_g(\boldsymbol{C} \odot \boldsymbol{B})^T\boldsymbol{Y}_{(1)}^T) + \text{tr}(\boldsymbol{A}\boldsymbol{D}_g(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{B}^T\boldsymbol{B})\boldsymbol{D}_g\boldsymbol{A}^T). \end{aligned} \tag{3.132}$$

The stationary point of L with respect to $\boldsymbol{A}$ is given by

$$\frac{\partial L}{\partial \boldsymbol{A}} = 2\boldsymbol{A}\boldsymbol{D}_g(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{B}^T\boldsymbol{B})\boldsymbol{D}_g - 2\boldsymbol{Y}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{D}_g = \boldsymbol{0}. \tag{3.133}$$

From (3.133), the update rules for $\boldsymbol{A}$ can be given by

$$\boldsymbol{A} \leftarrow \boldsymbol{Y}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{B}^T\boldsymbol{B})^\dagger; \tag{3.134}$$

$$\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||; \tag{3.135}$$

for $r = 1, ..., R$, where † stands for the Moore-Penrose pseudo-inverse. Finally, ALS-CP algorithm is summarized as Algorithm 11.

Algorithm 11 ALS-CP algorithm

Input: $\underline{\boldsymbol{Y}}$, R
Initialize: $\boldsymbol{B}$, $\boldsymbol{C}$ randomly
repeat
  $\boldsymbol{A} \leftarrow \boldsymbol{Y}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{B}^T\boldsymbol{B})^{\dagger}$;
  $\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r/||\boldsymbol{a}_r||$ for $r = 1, ..., R$;
  $\boldsymbol{B} \leftarrow \boldsymbol{Y}_{(2)}(\boldsymbol{C} \odot \boldsymbol{A})(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{A}^T\boldsymbol{A})^{\dagger}$;
  $\boldsymbol{b}_r \leftarrow \boldsymbol{b}_r/||\boldsymbol{b}_r||$ for $r = 1, ..., R$;
  $\boldsymbol{C} \leftarrow \boldsymbol{Y}_{(3)}(\boldsymbol{B} \odot \boldsymbol{A})(\boldsymbol{B}^T\boldsymbol{B} \circledast \boldsymbol{A}^T\boldsymbol{A})^{\dagger}$;
  $\boldsymbol{c}_r \leftarrow \boldsymbol{c}_r/||\boldsymbol{c}_r||$ for $r = 1, ..., R$;
until convergence
$g_r \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{a}_r^T \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T$ for $r = 1, ..., R$;
Output: g_r, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$

3.3.2 Deflation based algorithm

In this section, I introduce a greedy deflation based algorithm for the CP decomposition. This is a straight-forward extension of the deflation based algorithm for tSVD. First, we consider the following one-rank approximation problem:

$$\begin{aligned} &\text{minimize} \quad ||\underline{\boldsymbol{Y}} - g\boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}||_F^2, \\ &\text{subject to} \quad ||\boldsymbol{a}|| = ||\boldsymbol{b}|| = ||\boldsymbol{c}|| = 1. \end{aligned} \tag{3.136}$$

The cost function L can be transformed as

$$\begin{aligned} L :&= ||\boldsymbol{Y}_{(1)} - g\boldsymbol{a}(\boldsymbol{c} \otimes \boldsymbol{b})^T||_F^2 \\ &= \mathrm{tr}(\boldsymbol{Y}_{(1)}\boldsymbol{Y}_{(1)}^T) - 2\mathrm{tr}(\boldsymbol{a}g(\boldsymbol{c} \otimes \boldsymbol{b})^T\boldsymbol{Y}_{(1)}^T) + g^2||\boldsymbol{a}||^2. \end{aligned} \tag{3.137}$$

The stationary point of L with respect to $\boldsymbol{a}_r$ is given by

$$\frac{\partial L}{\partial \boldsymbol{a}} = 2g^2\boldsymbol{a} - 2g\boldsymbol{Y}_{(1)}(\boldsymbol{c} \otimes \boldsymbol{b}) = \boldsymbol{0}. \tag{3.138}$$

From (3.138), the update rules for $\boldsymbol{a}$ can be given by

$$\boldsymbol{a} \leftarrow \underline{\boldsymbol{Y}} \times_2 \boldsymbol{b}^T \times_3 \boldsymbol{c}^T; \tag{3.139}$$

$$\boldsymbol{a} \leftarrow \boldsymbol{a}/||\boldsymbol{a}||; \tag{3.140}$$

Finally, the power method for one-rank approximation and the deflation based algorithm for CP decomposition are summarized as Algorithms 12 and 13, respectively.

3.3.3 HALS algorithm

HALS algorithm for CP decomposition is also a straight-forward extension of the HALS-NMF algorithm. In fact, the HALS algorithm provides a better approximation than the greedy deflation based

Algorithm 12 one-rank-CP decomposition by power method

Input: $\underline{\boldsymbol{Y}}$
Initialize: $\boldsymbol{b}$, $\boldsymbol{c}$ randomly
repeat
  $\boldsymbol{a} \leftarrow \underline{\boldsymbol{Y}} \times_2 \boldsymbol{b}^T \times_3 \boldsymbol{c}^T$;
  $\boldsymbol{a} \leftarrow \boldsymbol{a}/||\boldsymbol{a}||$;
  $\boldsymbol{b} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{a}^T \times_3 \boldsymbol{c}^T$;
  $\boldsymbol{b} \leftarrow \boldsymbol{b}/||\boldsymbol{b}||$;
  $\boldsymbol{c} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{a}^T \times_2 \boldsymbol{b}^T$;
  $\boldsymbol{c} \leftarrow \boldsymbol{c}/||\boldsymbol{c}||$;
until convergence
$g \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{a}^T \times_2 \boldsymbol{b}^T \times_3 \boldsymbol{c}^T$;
Output: g, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$

Algorithm 13 Deflation-based-CP algorithm

Input: $\underline{\boldsymbol{Y}}$, R
for $r = 1, ..., R$ **do**
  Obtain $(g_r, \boldsymbol{a}_r, \boldsymbol{b}_r, \boldsymbol{c}_r)$ by the power method (Algorithm 12) inputting $\underline{\boldsymbol{Y}}$;
  $\underline{\boldsymbol{Y}} \leftarrow \underline{\boldsymbol{Y}} - g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$;
end for
Output: $[g_r, \boldsymbol{a}_r, \boldsymbol{b}_r, \boldsymbol{c}_r]_{r=1}^R$

algorithm. Its criterion is given by

$$\begin{aligned} &\text{minimize} \quad ||\underline{\boldsymbol{Y}}_r - g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r||_F^2, \\ &\text{subject to} \quad ||\boldsymbol{a}_r|| = ||\boldsymbol{b}_r|| = ||\boldsymbol{c}_r|| = 1, \end{aligned} \tag{3.141}$$

where $\underline{\boldsymbol{Y}}_r := \underline{\boldsymbol{Y}} - \sum_{k \neq r} g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$. Problems of (3.136) and (3.141) are basically equivalent. Thus the same update rules are given. Finally, we can get Algorithm 14.

3.3.4 Sparse/Nonnegative CP decomposition

Sparse CP decomposition can be available by imposing some sparsity constraint (e.g., l1-norm) into the CP criterion. In this case, techniques for the sparse PCA and the penalized matrix decomposition (PMD) are useful. For example, update rules (3.139) and (3.37) can be combined as

$$\boldsymbol{a}_r \leftarrow \frac{\boldsymbol{s}(\underline{\boldsymbol{Y}}_r \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T, \Delta)}{||\boldsymbol{s}(\underline{\boldsymbol{Y}}_r \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T, \Delta)||} \tag{3.142}$$

to obtain a sparse solution for the CP decomposition. The details are studied by [2].

In a similar way, a nonnegative version of the CP decomposition can be also available by imposing nonnegativity constraint into the CP criterion. In this case, techniques for the NMF are useful. Thus,

Algorithm 14 HALS-CP algorithm

Input: $\underline{\boldsymbol{Y}}$, R
Initialize: g_r, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ randomly
$\underline{\boldsymbol{E}} \leftarrow \underline{\boldsymbol{Y}} - \sum_{r=1}^{R} g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$;
repeat
  $\underline{\boldsymbol{Y}}_r \leftarrow \underline{\boldsymbol{E}} + g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$;
  $\boldsymbol{a}_r \leftarrow \underline{\boldsymbol{Y}}_r \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T$;
  $\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||$;
  $\boldsymbol{b}_r \leftarrow \underline{\boldsymbol{Y}}_r \times_1 \boldsymbol{a}_r^T \times_3 \boldsymbol{c}_r^T$;
  $\boldsymbol{b}_r \leftarrow \boldsymbol{b}_r / ||\boldsymbol{b}_r||$;
  $\boldsymbol{c}_r \leftarrow \underline{\boldsymbol{Y}}_r \times_1 \boldsymbol{a}_r^T \times_2 \boldsymbol{b}_r^T$;
  $\boldsymbol{c}_r \leftarrow \boldsymbol{c}_r / ||\boldsymbol{c}_r||$;
  $g_r \leftarrow \underline{\boldsymbol{Y}}_r \times_1 \boldsymbol{a}_r^T \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T$;
until convergence
Output: g, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$

the update rule for the nonnegative CP decomposition can be given by

$$\boldsymbol{a}_r \leftarrow [\underline{\boldsymbol{Y}}_r \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T]_+. \tag{3.143}$$

Note that the greedy deflation based algorithm is not applicable for the nonnegative CP since the orthogonality of nonnegative factor matrices is too low. The HALS algorithm can be only applied to the nonnegative CP decomposition, directly. In the ALS algorithm for the nonnegative CP decomposition, the multiplicative update rule is stable. Since the matricized form of the CP criterion can be characterized as the standard NMF criterion, then the update rule can be given by

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast [\boldsymbol{Y}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})] \oslash [\boldsymbol{A}(\boldsymbol{C}^T\boldsymbol{C} \circledast \boldsymbol{B}^T\boldsymbol{B})]; \tag{3.144}$$

3.4 Tucker decomposition

The Tucker decomposition factorizes a tensor into some of factor matrices and a core tensor. The Tucker model with $N = 3$ (i.e., Tucker3 model [98]) is given by

$$\underline{\boldsymbol{Y}} \simeq \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \boldsymbol{a}_{r_1} \circ \boldsymbol{b}_{r_2} \circ \boldsymbol{c}_{r_3}, \tag{3.145}$$

$$= \underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}, \tag{3.146}$$

$$= [\![\underline{\boldsymbol{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]. \tag{3.147}$$

Fig. 3.10 shows a visualization of Tucker3 model. The Tucker model can be considered as an generalized model of the CP decomposition since if a core tensor is diagonal tensor, then it is equivalent to the

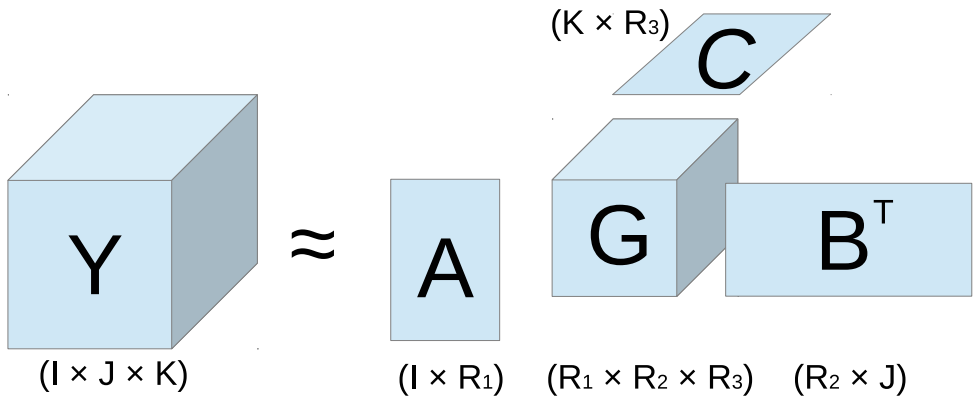


Figure 3.10: The Tucker3 decomposition

CP model (i.e., the CP model is a special case of the Tucker model). We can denote the matricized forms of the Tucker model as

$$\boldsymbol{Y}_{(1)} \simeq \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T, \tag{3.148}$$

$$\boldsymbol{Y}_{(2)} \simeq \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^T, \tag{3.149}$$

$$\boldsymbol{Y}_{(3)} \simeq \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^T. \tag{3.150}$$

Objective for the Tucker decomposition is given by

$$\text{minimize} \, ||\underline{\boldsymbol{Y}} - [\![\underline{\boldsymbol{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]||_F^2. \tag{3.151}$$

However, this solution is not unique and meaningless because the degree of freedom of the Tucker model is too high. In order to have a more meaningful decomposition, we need to impose appropriate constraints on factor matrices, e.g., orthogonality, sparsity, and nonnegativity.

3.4.1 HOOI algorithm for orthogonal Tucker decomposition

Degree of freedom of Tucker decomposition is too high to extract significant features since its solution is not unique. As the most basic approach, use of orthogonality constraint for each factor matrix has been introduced, thus the criterion for the three mode orthogonal Tucker decomposition can be given by

$$\begin{aligned} &\text{minimize} \quad ||\underline{\boldsymbol{Y}} - \underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}||_F^2, \\ &\text{subject to} \quad \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_{R_1}, \ \boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_{R_2}, \ \boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}_{R_3}, \ \boldsymbol{G}_{(n)}\boldsymbol{G}_{(n)}^T = \boldsymbol{\Lambda}_{R_n}, \end{aligned} \tag{3.152}$$

for $n = 1, 2, 3$. Using vectorized form, the cost function can be transformed as

$$\begin{aligned} L :=& ||\text{vec}(\underline{\boldsymbol{Y}}) - (\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\text{vec}(\underline{\boldsymbol{G}})|| \\ =& ||\text{vec}(\underline{\boldsymbol{Y}})||_F^2 - 2\text{vec}(\underline{\boldsymbol{Y}})^T(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\text{vec}(\underline{\boldsymbol{G}}) + ||\text{vec}(\underline{\boldsymbol{G}})||_F^2. \end{aligned} \tag{3.153}$$

Thus the stationary point with respect to $\mathrm{vec}(\underline{\boldsymbol{G}})$ can be given by

$$\frac{\partial L}{\partial \mathrm{vec}(\underline{\boldsymbol{G}})} = -2(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})^T \mathrm{vec}(\underline{\boldsymbol{Y}}) + 2\mathrm{vec}(\underline{\boldsymbol{G}}) = 0. \tag{3.154}$$

Finally, we can get the solution of $\underline{\boldsymbol{G}}$ as

$$\underline{\boldsymbol{G}} = \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T. \tag{3.155}$$

By using (3.155), the cost function can be transformed as

$$\begin{aligned} L &= ||\underline{\boldsymbol{Y}}||_F^2 - 2\mathrm{vec}(\underline{\boldsymbol{Y}})^T(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\mathrm{vec}(\underline{\boldsymbol{G}}) + ||\underline{\boldsymbol{G}}||_F^2 \\ &= ||\underline{\boldsymbol{Y}}||_F^2 - 2||\underline{\boldsymbol{G}}||_F^2 + ||\underline{\boldsymbol{G}}||_F^2 \\ &= ||\underline{\boldsymbol{Y}}||_F^2 - ||\underline{\boldsymbol{G}}||_F^2 \\ &= ||\underline{\boldsymbol{Y}}||_F^2 - ||\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T||_F^2, \end{aligned} \tag{3.156}$$

thus the minimization of cost function is equivalent to the maximization of $||\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T||_F^2$. This term can be transformed to

$$\begin{aligned} &\mathrm{tr}[\boldsymbol{A}^T \boldsymbol{Y}_{(1)}(\boldsymbol{C}\boldsymbol{C}^T \otimes \boldsymbol{B}\boldsymbol{B}^T)\boldsymbol{Y}_{(1)}^T \boldsymbol{A}] \\ =&\mathrm{tr}[\boldsymbol{B}^T \boldsymbol{Y}_{(2)}(\boldsymbol{C}\boldsymbol{C}^T \otimes \boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{Y}_{(2)}^T \boldsymbol{B}] \\ =&\mathrm{tr}[\boldsymbol{C}^T \boldsymbol{Y}_{(3)}(\boldsymbol{B}\boldsymbol{B}^T \otimes \boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{Y}_{(3)}^T \boldsymbol{C}]. \end{aligned} \tag{3.157}$$

Thus, the criterion with respect to each factor matrix is equivalence to the PCA criterion. For example, solution of $\boldsymbol{A}$ can be given as R_1 eigenvectors of $\boldsymbol{Y}_{(1)}(\boldsymbol{C}\boldsymbol{C}^T \otimes \boldsymbol{B}\boldsymbol{B}^T)\boldsymbol{Y}_{(1)}^T$ or R_1 left-singular vectors of $\boldsymbol{Y}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})$.

The first algorithm for the Tucker decomposition is proposed as "Tucker's method I" in [99]. Today, this method is better-known as the "higher-order SVD" (HOSVD) from [28]. The HOSVD does not minimize the cost function, however it can be used as an initialization algorithm (see Algorithm 15). Thus, the first algorithm in a real sense could be proposed as "three-mode PCA" by using ALS approach in [60]. After that its N-way extension is proposed in [54]. Finally, its algorithm is well-known as the higher order orthogonal iteration (HOOI) algorithm from [29]. The HOOI algorithm can be summarized as Algorithm 15.

3.4.2 Sparse Tucker decomposition

Sparse Tucker decomposition is a method to impose some sparsity constraint into each factor matrix of the Tucker decomposition. As above mentioned, the criterion for each factor matrix can be characterized as PCA criterion. Thus, we can apply the techniques of the sparse PCA or the penalized matrix decomposition to the method. For example, let $\underline{\boldsymbol{V}} := \underline{\boldsymbol{Y}} \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$, the SPCA like criterion for $\boldsymbol{A}$ of the Tucker decomposition can be given by

$$\begin{aligned} &\text{minimize} \quad ||\boldsymbol{V}_{(1)} - \widetilde{\boldsymbol{A}}\boldsymbol{A}^T\boldsymbol{V}_{(1)}|| + \lambda||\boldsymbol{A}||_F^2 + \sum_{r_1=1}^{R_1} ||\boldsymbol{a}_{r_1}||_1, \\ &\text{subject to} \quad \widetilde{\boldsymbol{A}}^T\widetilde{\boldsymbol{A}} = \boldsymbol{I}_{R_1}. \end{aligned} \tag{3.158}$$

Algorithm 15 HOOI algorithm for orthogonal Tucker decomposition

Input: $\underline{\boldsymbol{Y}}$, R_1, R_2, R_3
[Initialization by HOSVD algorithm]
$\boldsymbol{A} \leftarrow R_1$ left-singular vectors of $\boldsymbol{Y}_{(1)}$;
$\boldsymbol{B} \leftarrow R_2$ left-singular vectors of $\boldsymbol{Y}_{(2)}$;
$\boldsymbol{C} \leftarrow R_3$ left-singular vectors of $\boldsymbol{Y}_{(3)}$;
$\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$;
[Optimization step]
repeat
  $\boldsymbol{A} \leftarrow R_1$ left-singular vectors of $\boldsymbol{Y}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})$;
  $\boldsymbol{B} \leftarrow R_2$ left-singular vectors of $\boldsymbol{Y}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})$;
  $\boldsymbol{C} \leftarrow R_3$ left-singular vectors of $\boldsymbol{Y}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})$;
  $\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$;
until convergence
Output: $\underline{\boldsymbol{G}}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$

Furthermore, the left factor matrix of PMD of $\boldsymbol{V}_{(1)}$ can be also used to update $\boldsymbol{A}$. Note that the update rule (3.155) can not be used for the sparse tucker decomposition since it has no orthogonality. In this case, we must use

$$\underline{\boldsymbol{G}} = \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^{\dagger} \times_2 \boldsymbol{B}^{\dagger} \times_3 \boldsymbol{C}^{\dagger}. \tag{3.159}$$

3.4.3 Nonnegative Tucker decomposition

In this section, we consider to impose the nonnegativity constraint to all factors of Tucker decomposition. In a similar way to CP decomposition, its update rules can be given by applying the multiplicative update rule for NMF to the matricized form of Tucker decomposition. Thus, let $\underline{\boldsymbol{V}} := \underline{\boldsymbol{Y}} \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$, the update rule for $\boldsymbol{A}$ is given by

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast [\boldsymbol{Y}_{(1)} \boldsymbol{V}_{(1)}^T] \oslash [\boldsymbol{A} \boldsymbol{V}_{(1)} \boldsymbol{V}_{(1)}^T]; \tag{3.160}$$

The update rule for the core tensor $\underline{\boldsymbol{G}}$ can be given by the vectorized form of Tucker decomposition, and expressed as

$$\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{G}} \circledast (\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T) \oslash (\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}\boldsymbol{A}^T \times_2 \boldsymbol{B}\boldsymbol{B}^T \times_3 \boldsymbol{C}\boldsymbol{C}^T); \tag{3.161}$$

Finally, the algorithm for the nonnegative Tucker decomposition (NTD) [58] can be summarized as Algorithm 16. The algorithms for NTD are well studied in [83, 85, 84].

Algorithm 16 ALS-NTD algorithm

Input: $\underline{\boldsymbol{Y}}$, R_1, R_2, R_3
[Initialization step]
$\boldsymbol{A} \leftarrow$ left-factor matrix of NMF of $\boldsymbol{Y}_{(1)}$;
$\boldsymbol{B} \leftarrow$ left-factor matrix of NMF of $\boldsymbol{Y}_{(2)}$;
$\boldsymbol{C} \leftarrow$ left-factor matrix of NMF of $\boldsymbol{Y}_{(3)}$;
$\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{G}} \circledast (\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T) \oslash (\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T\boldsymbol{A} \times_2 \boldsymbol{B}^T\boldsymbol{B} \times_3 \boldsymbol{C}^T\boldsymbol{C})$;
[Optimization step]
repeat
  $\underline{\boldsymbol{V}} \leftarrow \underline{\boldsymbol{Y}} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$;
  $\boldsymbol{A} \leftarrow \boldsymbol{A} \circledast [\boldsymbol{Y}_{(1)}\boldsymbol{V}_{(1)}^T] \oslash [\boldsymbol{A}\boldsymbol{V}_{(1)}\boldsymbol{V}_{(1)}^T]$;
  $\underline{\boldsymbol{V}} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A} \times_3 \boldsymbol{C}$;
  $\boldsymbol{B} \leftarrow \boldsymbol{B} \circledast [\boldsymbol{Y}_{(2)}\boldsymbol{V}_{(2)}^T] \oslash [\boldsymbol{B}\boldsymbol{V}_{(2)}\boldsymbol{V}_{(2)}^T]$;
  $\underline{\boldsymbol{V}} \leftarrow \underline{\boldsymbol{Y}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B}$;
  $\boldsymbol{C} \leftarrow \boldsymbol{C} \circledast [\boldsymbol{Y}_{(3)}\boldsymbol{V}_{(3)}^T] \oslash [\boldsymbol{C}\boldsymbol{V}_{(3)}\boldsymbol{V}_{(3)}^T]$;
  $\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{G}} \circledast (\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T) \oslash (\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}\boldsymbol{A}^T \times_2 \boldsymbol{B}\boldsymbol{B}^T \times_3 \boldsymbol{C}\boldsymbol{C}^T)$;
until convergence
Output: $\underline{\boldsymbol{G}}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$

3.5 Common and Individual Feature Extraction

In this section, I introduce common and individual feature extraction methods. When a group of data is given from multiple sources in a same situation, data have common features for all sources and individual features depending on individual sources. For example, when we record voices of the same sentence "this is a pen" from several persons. The audio data have a common feature as a content "this is a pen" and individual features depend on a personal differences which might be including intonations, accents, vocal sounds and so on. In this case, common feature is important for speech recognition, and individual feature is important for speaker recognition. Thus, it is very important to analyze common and individual features for group data. However, this research is quite primitive now.

Algorithm 17 JIVE algorithm

1: **Input:** $\{\boldsymbol{Y}_k, R_k\}_{k=1}^K, R$
2: **Initialize:** $\boldsymbol{J} = [\boldsymbol{J}_1, \boldsymbol{J}_2, ..., \boldsymbol{J}_K]$
3: **repeat**
4: $\quad \boldsymbol{H}_k \leftarrow \text{tSVD}(\boldsymbol{Y}_k - \boldsymbol{J}_k,\ R_k)$ for all k;
5: $\quad \boldsymbol{J} \leftarrow \text{tSVD}([\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_K] - [\boldsymbol{H}_1, \boldsymbol{H}_2, ..., \boldsymbol{H}_K],\ R)$;
6: **until** convergence
7: **Output:** $\boldsymbol{J}$, $\{\boldsymbol{H}_k\}_{k=1}^K$

3.5.1 JIVE

The joint and individual variation explained (JIVE) method [66] can be regarded as the most basic model for common and individual feature analysis. JIVE model is given by

$$\begin{aligned}
\boldsymbol{Y}_k &\simeq \boldsymbol{J}_k + \boldsymbol{H}_k, && (3.162)\\
&= \boldsymbol{A}_C \boldsymbol{B}_C^{(k)T} + \boldsymbol{A}_I^{(k)} \boldsymbol{B}_I^{(k)T}, && (3.163)\\
&= [\boldsymbol{A}_C, \boldsymbol{A}_I^{(k)}][\boldsymbol{B}_C^{(k)}, \boldsymbol{B}_I^{(k)}]^T, && (3.164)\\
&= \boldsymbol{A}^{(k)} \boldsymbol{B}^{(k)T}, && (3.165)
\end{aligned}$$

where $\{\boldsymbol{Y}_k\}_{k=1}^K$ is a group of given data, $\boldsymbol{J}_k$ and $\boldsymbol{H}_k$ are joint and individual structure, respectively. Both structure matrices can be decomposed by

$$\begin{aligned}
\boldsymbol{J}_k &= \boldsymbol{A}_C \boldsymbol{B}_C^{(k)T}, && (3.166)\\
\boldsymbol{H}_k &= \boldsymbol{A}_I^{(k)} \boldsymbol{B}_I^{(k)T}. && (3.167)
\end{aligned}$$

We put $\boldsymbol{J} = [\boldsymbol{J}_1, \boldsymbol{J}_2, ..., \boldsymbol{J}_K]$, then $\boldsymbol{J}$ can be factorized by

$$\boldsymbol{J} = \boldsymbol{A}_C [\boldsymbol{B}_C^{(1)T}, \boldsymbol{B}_C^{(2)T}, ..., \boldsymbol{B}_C^{(K)T}]. \tag{3.168}$$

In the JIVE method, we consider an R-rank decomposition of $\boldsymbol{J}$ and R_k-rank decompositions of $\boldsymbol{H}_k$. The cost function of this decomposition is defined by

$$L = \sum_{k=1}^{K} ||\boldsymbol{Y}_k - \boldsymbol{J}_k - \boldsymbol{H}_k||_F^2. \tag{3.169}$$

In this method, we update iteratively and alternately $\boldsymbol{J}$ and $\{\boldsymbol{H}_k\}_{k=1}^K$ to minimize L until convergence. Thus, when $\boldsymbol{J}$ is given, we find $\{\boldsymbol{A}_I^{(k)}, \boldsymbol{B}_I^{(k)}\}_{k=1}^K$ to minimize $\sum_{k=1}^K ||\boldsymbol{Y}_k - \boldsymbol{J}_k - \boldsymbol{A}_I^{(k)} \boldsymbol{B}_I^{(k)T}||$. This solution can be given by R_k-rank tSVD of $(\boldsymbol{Y}_k - \boldsymbol{J}_k)$ for each $\boldsymbol{H}_k$. When $\{\boldsymbol{H}_k\}_{k=1}^K$ is given, we find an optimal $\boldsymbol{J}$ to minimize the cost function. We put $\boldsymbol{Y} = [\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_K]$ and $\boldsymbol{H} = [\boldsymbol{H}_1, \boldsymbol{H}_2, ..., \boldsymbol{H}_K]$, then the solution of $\boldsymbol{J}$ can be given by the R-rank tSVD of $(\boldsymbol{Y} - \boldsymbol{H})$. Finally, the JIVE algorithm can be summarized as Algorithm 17.

3.5.2 Group NMF

The group NMF [63] is a technique of common and individual feature extraction for a group of nonnegative data. Its framework includes a nonnegative extension and a more generalized model of the JIVE method; however, it had been proposed before JIVE. The decomposition model of the group NMF is given by

$$\boldsymbol{Y}_k \simeq \boldsymbol{A}^{(k)}\boldsymbol{X}^{(k)} = [\boldsymbol{A}_C^{(k)}\boldsymbol{A}_I^{(k)}]\boldsymbol{X}^{(k)}, \tag{3.170}$$

where the basis matrix $\boldsymbol{A}^{(k)} \in \mathbb{R}_+^{I\times R}$ is composed of two types of bases, where $\boldsymbol{A}_C^{(k)} \in \mathbb{R}_+^{I\times R_C}$ consists of common bases for all sources, and $\boldsymbol{A}_I^{(k)} \in \mathbb{R}_+^{I\times R_I}$ $(R = R_C + R_I)$ consists of bases which are depend on individual characteristics. When we impose a constraint $\boldsymbol{A}_C^{(1)} = \boldsymbol{A}_C^{(2)} = \cdots = \boldsymbol{A}_C^{(K)} =: \boldsymbol{A}_C$, this model is equivalent to the model of JIVE. This model is called the "FFX-NMF". Furthermore, we consider a group of data has no individual factor (i.e., $R_I = 0$). In this case, its decomposition model can be given by

$$[\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_K] \simeq \boldsymbol{A}_C[\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ..., \boldsymbol{X}^{(K)}]. \tag{3.171}$$

This model is called the "One-NMF".

Algorithm for One-NMF

One-NMF model is equivalent to the standard NMF model. We put $\boldsymbol{Y} = [\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_K]$ and $\boldsymbol{X} = [\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ..., \boldsymbol{X}^{(K)}]$, then it becomes $\boldsymbol{Y} \simeq \boldsymbol{A}_C\boldsymbol{X}$ which can be solved by standard NMF algorithms such as multiplicative algorithm, ALS algorithm, and HALS algorithm. In addition, we can apply sparse NMF and smooth NMF algorithms to this model, directly.

Algorithm for FFX-NMF

Introducing $\boldsymbol{X}^{(k)} = [\boldsymbol{X}_C^{(k)}, \boldsymbol{X}_I^{(k)}]$, the objective function of FFX-NMF is given by

$$J_{FFX} = \sum_{k=1}^{K} ||\boldsymbol{Y}_k - \boldsymbol{A}_C\boldsymbol{X}_C^{(k)} - \boldsymbol{A}_I^{(k)}\boldsymbol{X}_I^{(k)}||_F^2 + \gamma\left\{K||\boldsymbol{A}_C||_F^2 + \sum_{k=1}^{K}||\boldsymbol{A}_I^{(k)}||_F^2\right\}. \tag{3.172}$$

We find optimal $\boldsymbol{A}_C$, $\boldsymbol{A}_I^{(k)}$, and $\boldsymbol{X}^{(k)}$ to minimize J_{FFX}. Individual update rules are given by

$$\boldsymbol{A}_C \leftarrow \boldsymbol{A}_C \circledast \left(\sum_k \boldsymbol{Y}_k\boldsymbol{X}_C^{(k)T}\right) \oslash \left(\sum_k \boldsymbol{A}^{(k)}\boldsymbol{X}^{(k)}\boldsymbol{X}_C^{(k)T} + \gamma K\boldsymbol{A}_C\right), \tag{3.173}$$

$$\boldsymbol{A}_I^{(k)} \leftarrow \boldsymbol{A}_I^{(k)} \circledast \left(\boldsymbol{Y}_k\boldsymbol{X}_I^{(k)T}\right) \oslash \left(\boldsymbol{A}^{(k)}\boldsymbol{X}^{(k)}\boldsymbol{X}_I^{(k)T} + \gamma\boldsymbol{A}_I^{(k)}\right), \tag{3.174}$$

$$\boldsymbol{X}^{(k)} \leftarrow \boldsymbol{X}^{(k)} \circledast \left(\boldsymbol{A}^{(k)T}\boldsymbol{Y}_k\right) \oslash \left(\boldsymbol{A}^{(k)T}\boldsymbol{A}^{(k)}\boldsymbol{X}^{(k)}\right). \tag{3.175}$$

The solution can be obtained by the iteration to update individual parameter matrices until convergence.

Algorithm for GNMF

One-NMF and FFX-NMF are special cases of group NMF (GNMF). In this section, I introduce an algorithm for GNMF. We consider the following objective function:

$$\begin{aligned} J_{GNMF} =& \lambda \sum_{k=1}^{K} ||\boldsymbol{Y}_k - \boldsymbol{A}^{(k)}\boldsymbol{X}^{(k)}||_F^2 + \gamma \sum_{k=1}^{K} ||\boldsymbol{A}^{(k)}|| \\ &+ \frac{\alpha}{2} \sum_{i \neq j} ||\boldsymbol{A}_C^{(i)} - \boldsymbol{A}_C^{(j)}||_F^2 - \frac{\beta}{2} \sum_{i \neq j} ||\boldsymbol{A}_I^{(i)} - \boldsymbol{A}_I^{(j)}||_F^2, \end{aligned} \tag{3.176}$$

where λ, γ, α, and β are adjusting parameters. When $\alpha = \beta = 0$ and $\boldsymbol{A}_C = \boldsymbol{A}_C^{(k)}$ for all k, then its objective function is equivalent to J_{FFX}. The key point here is the third and the fourth terms in the objective function. The third term minimizes the squared error between individual $\boldsymbol{A}_C^{(k)}$, and the fourth term maximizes the squared difference between individual $\boldsymbol{A}_I^{(k)}$. We find optimal $\boldsymbol{A}_C^{(k)}$, $\boldsymbol{A}_I^{(k)}$, and $\boldsymbol{X}^{(k)}$ to minimize J_{GNMF}. Update rule of $\boldsymbol{X}^{(k)}$ is the same as (3.175). Update rule of $\boldsymbol{A}_C^{(k)}$ and $\boldsymbol{A}_I^{(k)}$ are given by

$$\boldsymbol{A}_C^{(k)} \leftarrow \boldsymbol{A}_C^{(k)} \circledast \boldsymbol{H}_C, \tag{3.177}$$

$$\boldsymbol{A}_I^{(k)} \leftarrow \boldsymbol{A}_I^{(k)} \circledast \boldsymbol{H}_I, \tag{3.178}$$

where

$$\boldsymbol{H}_C = \left(\boldsymbol{Y}_k \boldsymbol{X}_C^{(k)T} + \frac{\alpha}{\lambda} \sum_{j \neq k} \boldsymbol{A}_C^{(j)} \right) \oslash \left(\boldsymbol{A}^{(k)} \boldsymbol{X}^{(k)} \boldsymbol{X}_C^{(k)T} + \frac{\alpha}{\lambda}(L-1)\boldsymbol{A}_C^{(k)} + \frac{\gamma}{\lambda}\boldsymbol{A}_C^{(k)} \right), \tag{3.179}$$

$$\boldsymbol{H}_I = \left(\boldsymbol{Y}_k \boldsymbol{X}_I^{(k)T} + \frac{\beta}{\lambda}(L-1)\boldsymbol{A}_I^{(k)} \right) \oslash \left(\boldsymbol{A}^{(k)} \boldsymbol{X}^{(k)} \boldsymbol{X}_I^{(k)T} + \frac{\beta}{\lambda} \sum_{j \neq k} \boldsymbol{A}_I^{(j)} + \frac{\gamma}{\lambda}\boldsymbol{A}_I^{(k)} \right). \tag{3.180}$$

The solution can be obtained by the iteration to update individual parameter matrices until convergence.

Chapter 4

Supervised Feature Extraction and Classification methods for Pattern Recognition

In this Chapter, I introduce supervised signal processing methods for pattern recognition and point out their problems. Pattern recognition is a technique to recognize some patterns, automatically. For example, pattern recognition includes the face recognition, the handwritten character recognition, the fingerprinting identification, and the brain signal recognition.

4.1 Pattern Recognition

Flow of pattern recognition consists of pre-processing, feature extraction, and classification. The pre-processing includes cutout, normalization, filtering, and so on, and the feature extraction is to transform various types of data (tensor) to vector data. As an example, I introduce several supervised feature extraction techniques for the EEG based brain signal recognition in Sections 4.2. The classification is to assign an unknown pattern x into a category y. Fig 4.1 depicts the basic concept of pattern classification. Almost any data is allowed as a pattern x such as a discrete vector, a matrix of continuous values and a non-numeric data. The category y is a discrete value. I introduce several typical techniques of classification in Section 4.3. In this thesis, I focus on the classification methods, mainly.

4.1.1 Heuristic Approach vs. Statistical Approach

There are two major approaches to find the rules for recognition, heuristic approach and statistical approach. In the heuristic approach, we find rules from a human subjective idea. For example, we know that apple and lemon can be classified by color, thus we implement a program so that the computer output 'apple' and 'lemon' if it is red and yellow, respectively. When computers did not

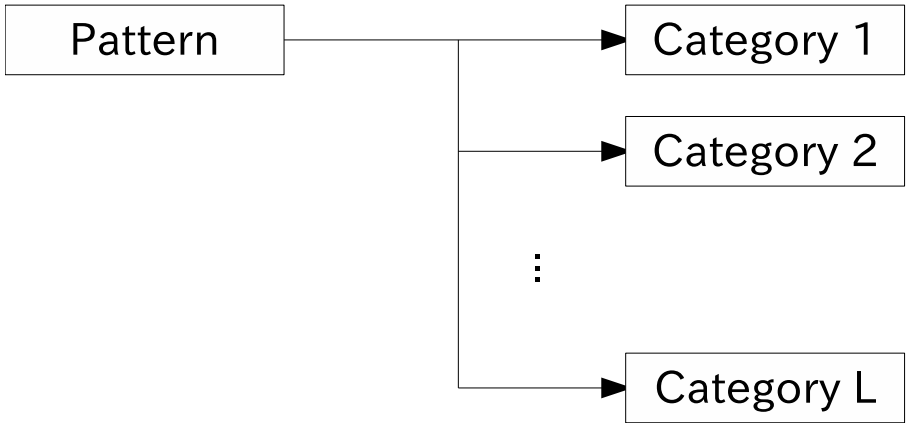


Figure 4.1: Basic concept of pattern classification

have a high computational capacity, the heuristic approach has been studied as a mainstream.

On the other hand, statistical approach finds the rules from observed pattern samples. In this case, pattern samples are very important since the classifier strongly depends on them. When we have enough number of samples, its classifier shows the excellent performance. With progress of computers, the statistical approach comes to be studied actively, and it became a mainstream. Statistical pattern recognition methods are based on supervised learning.

4.1.2 Supervised Learning

In this section, I explain the basic concept of supervised learning. We consider three components in supervised learning as follows [103].

- The generator of the data.
- The target operator (supervisor).
- The learning machine (classifier).

Fig 4.2 depicts the relation of three components. The generator outputs a random pattern vector $\boldsymbol{x}$ that is independent and identically distributed (i.i.d.). The supervisor inputs $\boldsymbol{x}$ and outputs its category y. But the supervisor can not realized in a computer. The learning machine has two functions. The first function inputs the training samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ that are given from the generator and the supervisor. And a classifier is constructed to imitate or identify the supervisor by using the training samples. The second function inputs an unknown pattern $\boldsymbol{x}$ and outputs its estimated category $\hat{y}$.

In the EEG based BCI system, we do not consider directly an EEG data as $\boldsymbol{x}$ since the data size of an EEG matrix $\boldsymbol{E}$ is very large which consists of many channels and many sampling points at every time. We consider the feature vector of an EEG matrix as $\boldsymbol{x}$. Thus, we first need a feature extraction method to transform EEG signal matrices to feature vectors. Fig 4.3 shows a standard feature extraction scheme for motor imagery based BCI from an EEG signal $\boldsymbol{E}$ to a feature vector $\boldsymbol{x}$.

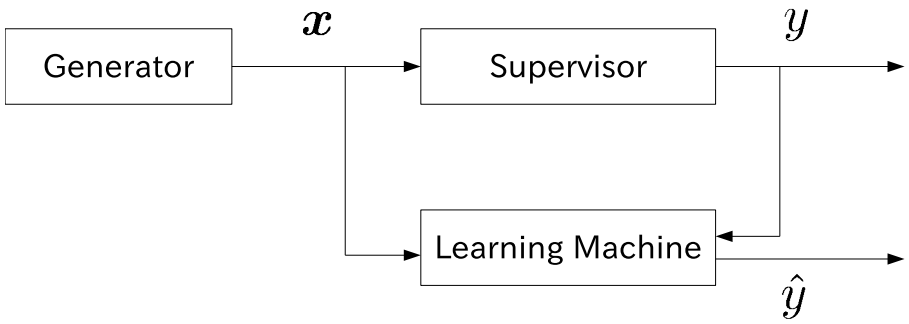


Figure 4.2: Basic concept of supervised learning

After the feature extraction, we shift to a classification procedure. Fig 4.4 shows a classification scheme. In order to classify a feature vector $\boldsymbol{x}$ to a command y, a classification rule (i.e., classifier) is necessary. A classifier has to learn the classification rule from a training set $[\boldsymbol{X}_{train},\ \boldsymbol{y}_{train}]$. Thus, a BCI system can be constructed.

4.2 Supervised Feature Extraction methods

In this section, I introduce a typical spatial filter and its extension methods for the EEG motor imagery classification. The proper spatial filter provides signals that are easily classified. The goal of this research is to design spatial filters that lead to better recognition rates.

EEG signals are formalized as

$$\{\boldsymbol{E}_n\}_{n=1}^{N} \in \mathbb{R}^{ch \times time}, \tag{4.1}$$

where ch, $time$, and N are the number of channels, the number of sampling points of time, and the number of trials, respectively. To apply the EEG signals to classification, we have to transform the each EEG signal to a feature vector. Thus, some transformation from the EEG signal to the feature vector

$$\boldsymbol{E}_n \in \mathbb{R}^{ch \times time} \mapsto \boldsymbol{x}_n \in \mathbb{R}^{d}, \tag{4.2}$$

is necessary. Thus, we obtain a set of feature vectors $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$.

The problem is how to extract important features for the best classification. The key-points here are

- EEG signals are very noisy (noise reduction),
- There exists the best frequency band for classification (frequency band selection),
- There exists important channels and un-important channels for classification (channel selection or spatial filters).

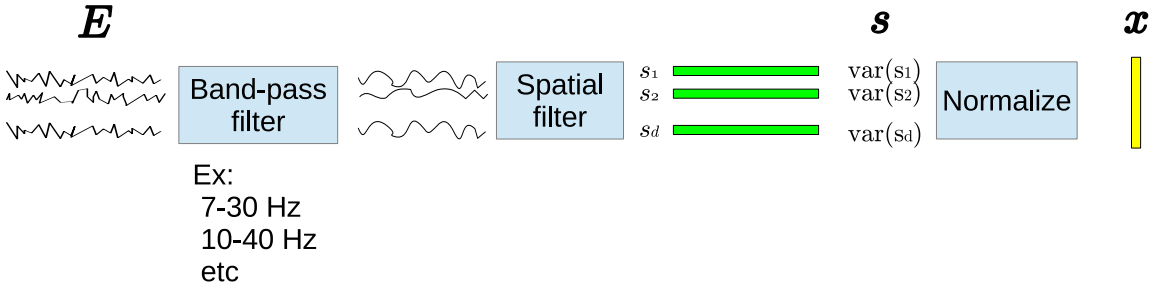


Figure 4.3: Standard feature extraction scheme

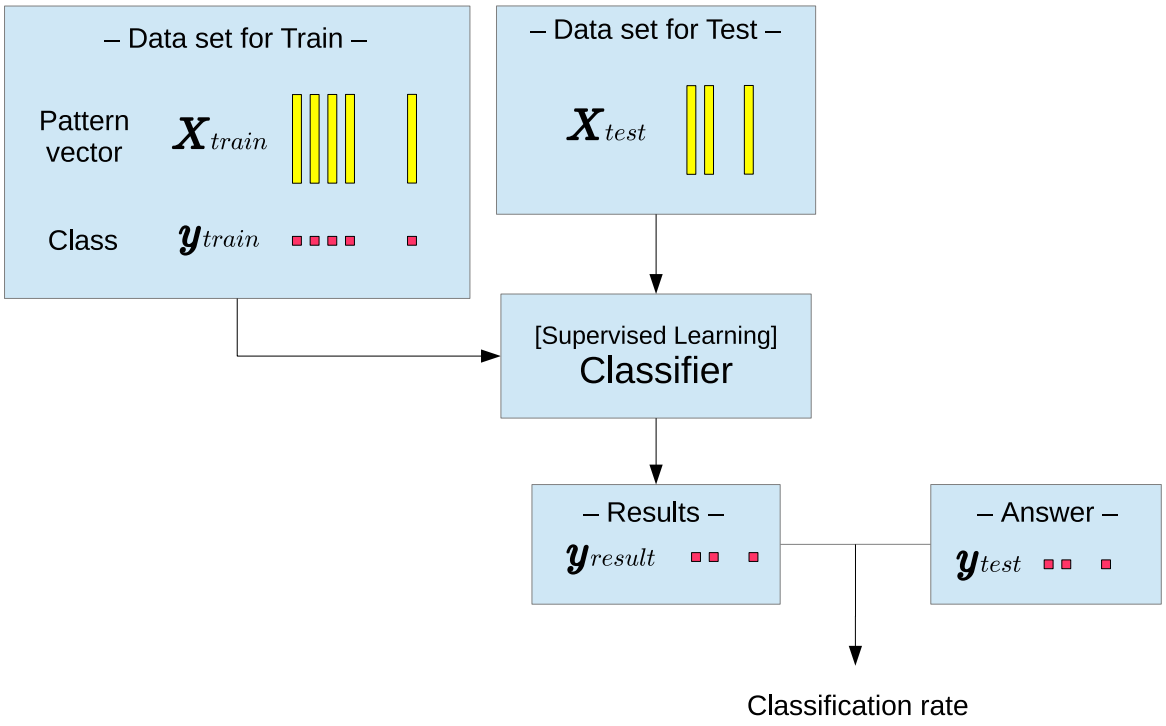


Figure 4.4: Classification scheme

4.2.1 CSP filter

The "Common Spatial Pattern" (CSP) filter [75] is a typical spatial filter for motor imagery classification by using many channels of EEG signals. We denote the CSP filter by

$$\boldsymbol{S} = \boldsymbol{W}^T \boldsymbol{E} \quad \text{or} \quad \boldsymbol{s}(t) = \boldsymbol{W}^T \boldsymbol{e}(t), \tag{4.3}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times ch}$ is the spatial filter matrix, $\boldsymbol{S} \in \mathbb{R}^{d \times time}$ is the filtered signal matrix.

The criterion of CSP is given by

$$\text{maximize} \quad \text{tr}\boldsymbol{W}^T \boldsymbol{\Sigma}_1 \boldsymbol{W}, \tag{4.4}$$

$$\text{subject to} \quad \boldsymbol{W}^T (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2) \boldsymbol{W} = \boldsymbol{I}, \tag{4.5}$$

where

$$\boldsymbol{\Sigma}_1 = \underset{\boldsymbol{E}_n \in \{\text{class } 1\}}{\text{Exp}} \frac{\boldsymbol{E}_n \boldsymbol{E}_n^T}{\text{tr} \boldsymbol{E}_n \boldsymbol{E}_n^T}, \tag{4.6}$$

$$\boldsymbol{\Sigma}_2 = \underset{\boldsymbol{E}_n \in \{\text{class } 2\}}{\text{Exp}} \frac{\boldsymbol{E}_n \boldsymbol{E}_n^T}{\text{tr} \boldsymbol{E}_n \boldsymbol{E}_n^T}. \tag{4.7}$$

This problem can be solved by the generalized eigenvalue problem. However, we can also solve it by solving two standard eigenvalue problems.

First we decompose $\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2$ as

$$\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 = \boldsymbol{U} \boldsymbol{D} \boldsymbol{U}^T, \tag{4.8}$$

where $\boldsymbol{U}$ is a set of eigenvectors, and $\boldsymbol{D}$ is a diagonal matrix of eigenvalues.

Next, let $\boldsymbol{P} := \sqrt{\boldsymbol{D}^{-1}} \boldsymbol{U}^T$, and calculate

$$\widehat{\boldsymbol{\Sigma}}_1 = \boldsymbol{P} \boldsymbol{\Sigma}_1 \boldsymbol{P}^T, \tag{4.9}$$

$$\widehat{\boldsymbol{\Sigma}}_2 = \boldsymbol{P} \boldsymbol{\Sigma}_2 \boldsymbol{P}^T. \tag{4.10}$$

Please note that we have $\widehat{\boldsymbol{\Sigma}}_1 + \widehat{\boldsymbol{\Sigma}}_2 = \boldsymbol{I}$, here. Thus, any orthonormal matrices $\boldsymbol{V}$ satisfy $\boldsymbol{V}^T (\widehat{\boldsymbol{\Sigma}}_1 + \widehat{\boldsymbol{\Sigma}}_2) \boldsymbol{V} = \boldsymbol{I}$.

Finally, we decompose $\widehat{\boldsymbol{\Sigma}}_1$ as

$$\widehat{\boldsymbol{\Sigma}}_1 = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^T, \tag{4.11}$$

where $\boldsymbol{V}$ is a set of eigenvectors, and $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues.

A set of CSP filters is obtained as

$$\boldsymbol{W} = \boldsymbol{P}^T \boldsymbol{V}. \tag{4.12}$$

We have

$$\boldsymbol{W}^T \boldsymbol{\Sigma}_1 \boldsymbol{W} = \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{ch} \end{pmatrix}, \tag{4.13}$$

$$\boldsymbol{W}^T \boldsymbol{\Sigma}_2 \boldsymbol{W} = \boldsymbol{I} - \boldsymbol{\Lambda} = \begin{pmatrix} 1 - \lambda_1 & & \\ & \ddots & \\ & & 1 - \lambda_{ch} \end{pmatrix}, \tag{4.14}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{ch}$. Therefore, first CSP filter $\boldsymbol{w}_1$ provides maximum variance for class 1, and last CSP filter $\boldsymbol{w}_{ch}$ provides the maximum variance for class 2.

We select first and last m filters and construct a matrix $\boldsymbol{W}$ as

$$\boldsymbol{W}_{csp} = \begin{pmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_m & \boldsymbol{w}_{ch-m+1} & \cdots & \boldsymbol{w}_{ch} \end{pmatrix} \in \mathbb{R}^{2m \times ch}, \tag{4.15}$$

and the filtered signal matrix $\boldsymbol{s}(t)$ is given by

$$\boldsymbol{s}(t) = \boldsymbol{W}_{csp}^T \boldsymbol{e}(t) = \begin{pmatrix} s_1(t) & \cdots & s_d(t) \end{pmatrix}^T, \tag{4.16}$$

where $d = 2m$.

Finally, the feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$ is calculated by

$$x_i = \log \left(\frac{\mathrm{var}[s_i(t)]}{\sum_{i=1}^{d} \mathrm{var}[s_i(t)]} \right). \tag{4.17}$$

Example

As an example, I show feature vectors for the motor imagery EEG dataset [8]. The motor imagery EEG is closely related to primary motor cortex in brain (see Fig 4.5). Fig. 4.6 shows topographs of CSP weight at right hand and foot imagery. We can see that CSP filter provides optimal weightings so that left area is weighted for right hand imagery and central area is weighted for foot imagery. Right hand imagery pattern and foot imagery pattern of EEG signals can be easily separated by using CSP filter unlike PCA (see Fig. 4.7).

4.2.2 CSSP filter

The Common Spatio-Spectral Pattern (CSSP) filter is an extension of the CSP filter [64]. The CSSP can be regarded as a CSP method with the time delay embedding.

The algorithm is similar to that of the standard CSP. In CSP, we consider the following transformation

$$\boldsymbol{S} = \boldsymbol{W}^T \boldsymbol{E} \quad \text{or} \quad \boldsymbol{s}(t) = \boldsymbol{W}^T \boldsymbol{e}(t). \tag{4.18}$$

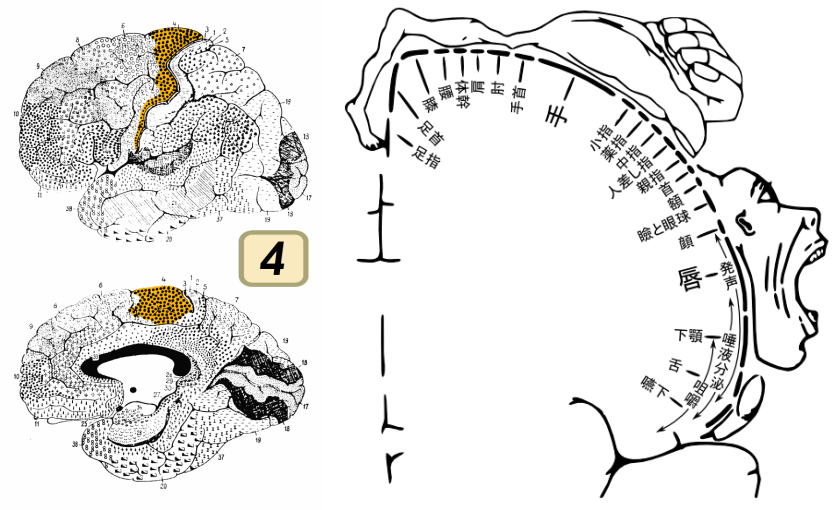
However, the CSSP's transform is given by

$$\boldsymbol{S} = \boldsymbol{W}^T \boldsymbol{E} + \boldsymbol{W}_\tau^T \boldsymbol{E}_\tau = \widehat{\boldsymbol{W}}^T \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{E}_\tau \end{pmatrix}, \tag{4.19}$$

$$\text{or} \quad \boldsymbol{s}(t) = \boldsymbol{W}^T \boldsymbol{e}(t) + \boldsymbol{W}_\tau^T \boldsymbol{e}(t+\tau) = \widehat{\boldsymbol{W}}^T \begin{pmatrix} \boldsymbol{e}(t) \\ \boldsymbol{e}(t+\tau) \end{pmatrix}, \tag{4.20}$$

where $\boldsymbol{E}_\tau$ is a τ-time delayed signal matrix of $\boldsymbol{E}$, and $\widehat{\boldsymbol{W}}^T = [\boldsymbol{W}^T,\ \boldsymbol{W}_\tau^T]$ is a CSSP matrix. It can be also regarded that the number of channels increases to double. The deference between CSP and CSSP is only this point, and then we can apply this method easily in a same way as the CSP algorithm. However, τ is a hyper-parameter.

The key-point here is this method can be interpreted into a spatial and a spectral filters. Therefore let $\widehat{\boldsymbol{w}}$ denote the i-th column of the CSSP matrix $\widehat{\boldsymbol{W}}$, then the projected signal $s(t)$ can be expressed

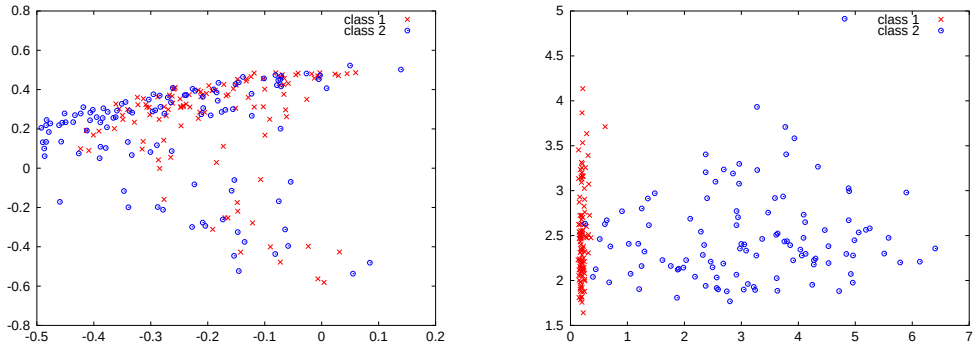


(a) Brodmann's brain area (b) Penfield's motor homunculus

Figure 4.5: Primary motor cortex

(a) Right Hand (al) (b) Foot (al)

Figure 4.6: Topographies of CSP filter: BCI competition III(IVa)



(a) PCA feature (b) CSP feature

Figure 4.7: 2D-plot of CSP feature: classes 1 and 2 stand for right hand and foot imageries, respectively. Horizontal and vertical axes in CSP feature space are first and last components, respectively.

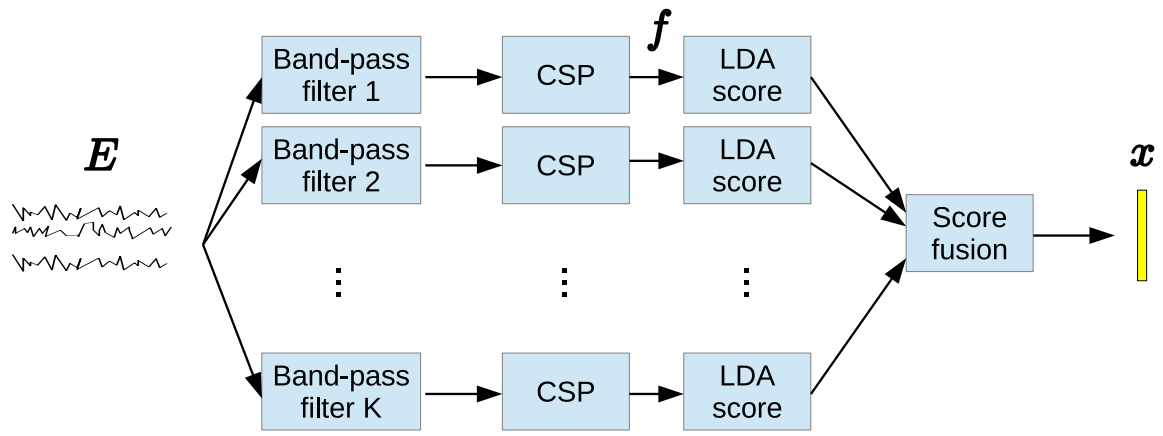


Figure 4.8: System Flowchart of SBCSP: for example frequency bands of individual filters are 4-8Hz, 8-12Hz, ... , 36-40Hz.

as

$$s(t) = \boldsymbol{w}^T \boldsymbol{e}(t) + \boldsymbol{w}_\tau^T \boldsymbol{e}(t+\tau) \tag{4.21}$$

$$= \sum_{j=1}^{ch} w_j e_j(t) + (w_\tau)_j e_j(t+\tau) \tag{4.22}$$

$$= \sum_{j=1}^{ch} \gamma_j \left(\frac{w_j}{\gamma_j} e_j(t) + \frac{(w_\tau)_j}{\gamma_j} e_j(t+\tau) \right), \tag{4.23}$$

where

- γ_j can be regarded as a pure spatial filter,
- $\left[\frac{w_j}{\gamma_j},\ 0, \ldots, 0, \frac{(w_\tau)_j}{\gamma_j} \right]$ can be regarded as a finite impulse response (FIR) filter for each channel.

Since the CSSP filter includes a temporal filter, it can be considered that the CSSP becomes robust for noise or outliers.

4.2.3 SBCSP filter

Here, I introduce an alternative method based on Sub-band CSP (SBCSP) and score fusion [77]. Fig. 4.8 shows a flowchart of feature extraction by using the SBCSP filter. How to decide a frequency band of band-pass filter is a difficult problem of CSP/CSSP filter; however, the SBSCP can automatically decide the frequency band of the band-pass filter. First, we make candidates of frequency bands, for example 4-8 Hz, 8-12 Hz, 12-16 Hz, ..., 36-40 Hz. Next we use all candidates of frequency bands for band-pass filter, and obtain CSP features from each filtered signals. Thus, we have individual CSP feature data matrices $\{\boldsymbol{F}^{(1)}, \boldsymbol{F}^{(2)}, \ldots, \boldsymbol{F}^{(K)}\}$. Next, the process of LDA score is given by follow:

- Calculate the LDA projection vector $\widehat{\boldsymbol{w}}^{(k)}$ for each data matrices $\boldsymbol{F}^{(k)}$.

- LDA score vector is given by

$$\boldsymbol{s}_n = \begin{pmatrix} \widehat{\boldsymbol{w}}^{(1)T} \boldsymbol{f}_n^{(1)} \\ \widehat{\boldsymbol{w}}^{(2)T} \boldsymbol{f}_n^{(2)} \\ \vdots \\ \widehat{\boldsymbol{w}}^{(K)T} \boldsymbol{f}_n^{(K)} \end{pmatrix}. \tag{4.24}$$

There are two approaches of score fusion method.

- Recursive Band Elimination (RBE)
- Meta-Classifier (MC)

In this way, we do not need to select frequency bands in the SBCSP by hand, thus it can be regarded as an automatically selection of the frequency band.

Recursive Band Elimination

In RBE, first we set a RBE-order that is an integer number $r \in [1, K]$. The first data set is $\boldsymbol{X} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_N] \in \mathbb{R}^{K \times N}$. In this method, we remove $(K - r)$ rows from $\boldsymbol{X}$ by support vector machine (SVM) feature selection (SVM is explained in Section 4.3.5). The algorithm is as follow

1. Train the parameter $\boldsymbol{w}$ in linear model $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ by SVM
2. Remove the row of $\boldsymbol{X}$ with the smallest w_k^2
3. If number of rows of $\boldsymbol{X}$ becomes r, this algorithm is finished, else back to 1.

The survival data $\boldsymbol{X}$ can be used as a training data matrix.

Meta-Classifier

In MC, we use a Bayesian classifier for score fusion. We assume that each band score $s_k \in \{\text{class 1}\}$ and $s_k \in \{\text{class 2}\}$ are distributed normally. And we can estimate individual parameters of normal distributions (i.e., $\{\mu_1^{(k)}, \sigma_1^{(k)}\}$ and $\{\mu_2^{(k)}, \sigma_2^{(k)}\}$). We calculate the training data matrix by

$$x_{kn} = \log \left\{ \frac{p(s_{kn} | \mu_1^{(k)}, \sigma_1^{(k)})}{p(s_{kn} | \mu_2^{(k)}, \sigma_2^{(k)})} \right\}, \tag{4.25}$$

where x_{kn} is the (k, n)-element of $\boldsymbol{X}$.

4.3 Classification methods

In this section, I introduce practical classification methods.

4.3.1 Generative Model vs Discriminant Model

For classification, it is very important to decide the model of classifier. Models of classifier can be divided into either generative model or discriminant model.

Generative model is based on probability such that we can classify any patterns $\boldsymbol{x}$ into the category $\hat{y}$ based on a posterior probability $P(y|\boldsymbol{x})$ as

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y|\boldsymbol{x}). \tag{4.26}$$

It is called maximum a *posteriori* (MAP) estimation. In terms of recognition rate, the MAP estimation is the best estimator. For these reasons, it is very famous and extensively studied. A key point of MAP estimation is how to estimate the posterior probabilities accurately.

The discriminant model is based on some discriminant functions $D_y(\boldsymbol{x})$ as

$$D_y : \mathbb{R}^d \to \mathbb{R} \tag{4.27}$$

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, D_y(\boldsymbol{x}). \tag{4.28}$$

If we assume that $D_y(\boldsymbol{x}) := P(y|\boldsymbol{x})$, we can see that the discriminant model includes the generative model. However, their approaches are different. Generally, we define some non-negative-valued function $R(\boldsymbol{D}(\boldsymbol{x}), y)$ named error function, where $\boldsymbol{D}(\boldsymbol{x})$ is a vector-valued function of which element is $D_y(\boldsymbol{x})$. The discriminant functions are estimated by minimizing the expectation of error function as follows:

$$\text{minimize} \sum_y \int_{\mathcal{D}} P(\boldsymbol{x}, y) R(\boldsymbol{D}(\boldsymbol{x}), y) \mathrm{d}\boldsymbol{x}. \tag{4.29}$$

When we have training samples, it can be approximated by replacing ensemble mean by sample mean as

$$\sum_y \int_{\mathcal{D}} P(\boldsymbol{x}, y) R(\boldsymbol{D}(\boldsymbol{x}), y) \mathrm{d}\boldsymbol{x} \approx \frac{1}{N} \sum_{n=1}^{N} R(\boldsymbol{D}(\boldsymbol{x}_n), y_n). \tag{4.30}$$

Then we can obtain the discriminant functions by minimizing this formula if the definition of error function and the model of discriminant functions are given.

Generative models

In this section, I introduce the generative model based classification methods. In general, its approach based on Bayes theorem:

$$P(y|\boldsymbol{x}) = \frac{P(y)P(\boldsymbol{x}|y)}{P(\boldsymbol{x})}. \tag{4.31}$$

If we estimate each probabilities of right side, then any $\boldsymbol{x}$ can be classified by the MAP rule. Since $P(y|\boldsymbol{x}) \propto P(y)P(\boldsymbol{x}|y)$, we can assign any $\boldsymbol{x}$ into $\hat{y}$ by $\operatorname{argmax}_y P(y) f_y(\boldsymbol{x}|\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$. $P(y)$ can be estimated as the appearance ratio of individual y. Thus, the key-point here is to estimate $P(\boldsymbol{x}|y)$ by

some generative model function. There are three approaches to estimate the generative model: the parametric model, the non-parametric model, and the semi-parametric model.

The parametric model is generally given by $f_y(\boldsymbol{x}; \Theta_y)$, where Θ_y is a set of parameters. For example, when we assume f_y follows the Gaussian distribution, then it is given by

$$f_y(\boldsymbol{x}|\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\boldsymbol{\Sigma}_y|}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_y^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_y)\right], \tag{4.32}$$

where $\Theta_y = \{\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y\}$. In this case, we only need to estimate a mean parameter vector $\boldsymbol{\mu}_y$ and a covariance parameter matrix $\boldsymbol{\Sigma}_y$ for each y. Maximum likelihood estimation is often used to estimate these parameters.

The non-parametric model does not have any parameter to estimate; but it has some hyper-parameters. Kernel density estimation is often used for non-parametric estimation.

The semi-parametric model can be considered as an intermediate model between parametric and non-parametric models. Gaussian mixture model is often used, and it is often obtained by EM-algorithm.

Discriminant models

There are several discriminant models. Most basic model is a linear model which is given by

$$D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \theta, \tag{4.33}$$

where $\boldsymbol{w}$ and θ are the model parameters, and $\langle \cdot, \cdot \rangle$ stands for the inner-product. It can be expanded by using basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ as

$$D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle + \theta. \tag{4.34}$$

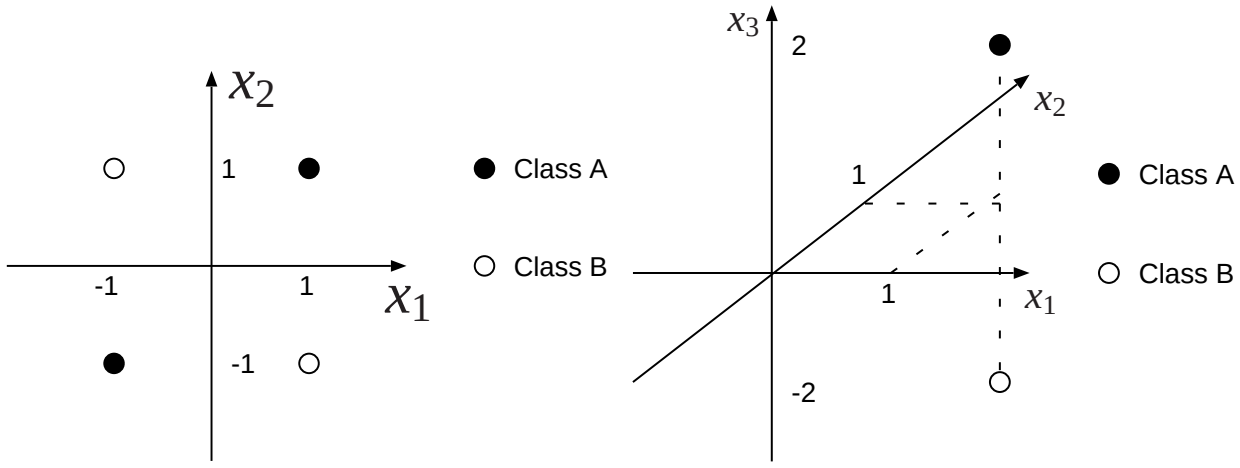
Note that, the dimensions of parameter vectors $\boldsymbol{w}$ in Eq.(4.33) and in Eq.(4.34) are different. The linear model is often used for theoretical study of classifier. For example, perceptron learning [88], optimal hyperplane classifier (OHC) [103] and Fisher discriminant analysis (FDA) [72] are using the linear model.

Next I introduce the quadratic model as

$$D(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x} \rangle + \langle \boldsymbol{b}, \boldsymbol{x} \rangle + \theta, \tag{4.35}$$

where $\boldsymbol{A}$, $\boldsymbol{b}$ and θ are parameters. Its typical example is given by the class-featuring information compression (CLAFIC) method [104].

In general, if the order of the model or the number of parameters is higher, we can construct a more complex classifier. However, if it is too high, the classifier over-fits to the training samples and its generalization capability is decreased. Model selection and regularization techniques are very important for classification to prevent the over-fitting. Especially, the cross validation method is very famous and useful for model selection. I explain its details in section 4.3.3.



(a) original space (b) feature space

Figure 4.9: Example of training samples

4.3.2 Kernel Method

There are many cases such that training samples can not be separated linearly. In such cases, it is very efficient to map pattern vectors to a higher dimensional space. For example, we consider a mapping $\boldsymbol{\phi}$ as follows:

$$\begin{aligned} \boldsymbol{\phi} \quad : \quad & \mathbb{R}^2 \to \mathbb{R}^3, \\ & (t_1,\ t_2)^T \mapsto (t_1^2,\ t_2^2,\ 2t_1t_2)^T, \end{aligned} \tag{4.36}$$

where $(\cdot)^T$ denotes the transposition. This mapping is based on correlations of each element. We assume that there are four training samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^4$ with categories $y \in \{A, B\}$ as follow:

$$\boldsymbol{x}_1 = (1,1)^T, \qquad y_1 = A, \tag{4.37}$$
$$\boldsymbol{x}_2 = (-1,-1)^T, \qquad y_2 = A, \tag{4.38}$$
$$\boldsymbol{x}_3 = (1,-1)^T, \qquad y_3 = B, \tag{4.39}$$
$$\boldsymbol{x}_4 = (-1,1)^T, \qquad y_4 = B. \tag{4.40}$$

Fig 4.9(a) depicts the training samples in a 2-dimensional original Euclid space. In this case, we can not separate the training samples into 2 categories linearly. However, if we consider the mapping (4.36), it can be separated linearly. Feature vectors as mapped samples are given as

$$\boldsymbol{\phi}(\boldsymbol{x}_1) = (1,1,2)^T, \qquad y_1 = A, \tag{4.41}$$
$$\boldsymbol{\phi}(\boldsymbol{x}_2) = (1,1,2)^T, \qquad y_2 = A, \tag{4.42}$$
$$\boldsymbol{\phi}(\boldsymbol{x}_3) = (1,1,-2)^T, \qquad y_3 = B, \tag{4.43}$$
$$\boldsymbol{\phi}(\boldsymbol{x}_4) = (1,1,-2)^T, \qquad y_4 = B. \tag{4.44}$$

Fig 4.9(b) depicts the training samples in the 3-dimensional feature space. In this space, we can separate the training samples linearly.

It often happens that the performance of classifier is improved by mapping data into the high dimensional feature space. In this way, classification capability can be improved by using a high-dimensional feature mapping $\boldsymbol{\phi}$; however, $\boldsymbol{\phi}(\boldsymbol{x})$ has a very large dimension and its computational cost is also very high. Fortunately, we can solve this problem by using the kernel method.

In the kernel method, instead of calculating the high-dimensional feature vector, we calculate the inner-product between vectors in feature space by using a kernel function:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y}) \rangle \,. \tag{4.45}$$

In such case, we have to show that a mapping $\boldsymbol{\phi}$ that satisfies Eq.(4.45) exists. If the kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ satisfies Mercer's theorem, the existence of $\boldsymbol{\phi}$ is guaranteed. Mercer's theorem is described as follows:

Theorem 1 (Mercer [103])**.** *Let $\mathcal{X}$ be a pattern space, $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, and $\mathcal{L}_2(\mathcal{X})$ be a set of all quadratically integrable functions, to guarantee that a continuous symmetric function $k(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{L}_2(\mathcal{X})$ has an expansion*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{\infty} a_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y}) \tag{4.46}$$

with positive coefficients $a_i > 0$ (i.e., $k(\boldsymbol{x}, \boldsymbol{y})$ describes an inner product in some feature space), it is necessary and sufficient that the condition

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \geq 0 \tag{4.47}$$

be valid for all $f \in \mathcal{L}_2(\mathcal{X})$□

Now, we define the kernel discriminant function by using sample patterns $\{\boldsymbol{x}_n\}_{n=1}^N$ as

$$D(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}). \tag{4.48}$$

This discriminant function model is called the kernel model. We can handle classification problems in a high dimensional feature space without the calculation of mapping by using the kernel model. This technique is called "kernel trick". For examples, the following kernel functions satisfy Mercer's theorem.

$$\begin{aligned} &k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle && : \text{Linear kernel} \\ &k(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle + 1)^d && : \text{Polynomial kernel} \\ &k(\boldsymbol{x}, \boldsymbol{y}) = \exp\Big(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2\Big) && : \text{Gaussian kernel} \end{aligned} \tag{4.49}$$

Since $k(\boldsymbol{x}_n, \boldsymbol{x}) = \langle \boldsymbol{\phi}(\boldsymbol{x}_n), \boldsymbol{\phi}(\boldsymbol{x}) \rangle$, the discriminant function is given by

$$D(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{\phi}(\boldsymbol{x}_n), \boldsymbol{\phi}(\boldsymbol{x}) \rangle \tag{4.50}$$

$$= \left\langle \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n), \boldsymbol{\phi}(\boldsymbol{x}) \right\rangle. \tag{4.51}$$

Then, this classifier in the feature space is expressed as

$$D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle, \tag{4.52}$$

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n). \tag{4.53}$$

From Eq.(4.53), the parameter vector $\boldsymbol{w}$ is confined in a subspace $[\boldsymbol{\phi}(\boldsymbol{x}_n)]_{n=1}^N$ that is spanned by feature vectors $\{\boldsymbol{\phi}(\boldsymbol{x}_n)\}_{n=1}^N$. It is an basic property of kernel models that $||\boldsymbol{w}||^2$ is expressed as

$$||\boldsymbol{w}||^2 = \left\langle \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n), \sum_{m=1}^{N} \alpha_m \boldsymbol{\phi}(\boldsymbol{x}_m) \right\rangle \tag{4.54}$$

$$= \sum_{n=1}^{N} \alpha_n \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{4.55}$$

$$= \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}, \tag{4.56}$$

where

$$\boldsymbol{\alpha} := \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_N \end{pmatrix}^T, \tag{4.57}$$

$$\boldsymbol{K} := \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}. \tag{4.58}$$

4.3.3 Least Square Regression

In this section, I explain the least square regression (LSR). LSR is not a classification method but a regression method for general purposes, however we use LSR as a classifier here. And we also assume that the discriminant function is given by

$$D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle. \tag{4.59}$$

Furthermore, the training samples are given by $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, and we consider a binary classification problem as $y \in \{+1, -1\}$.

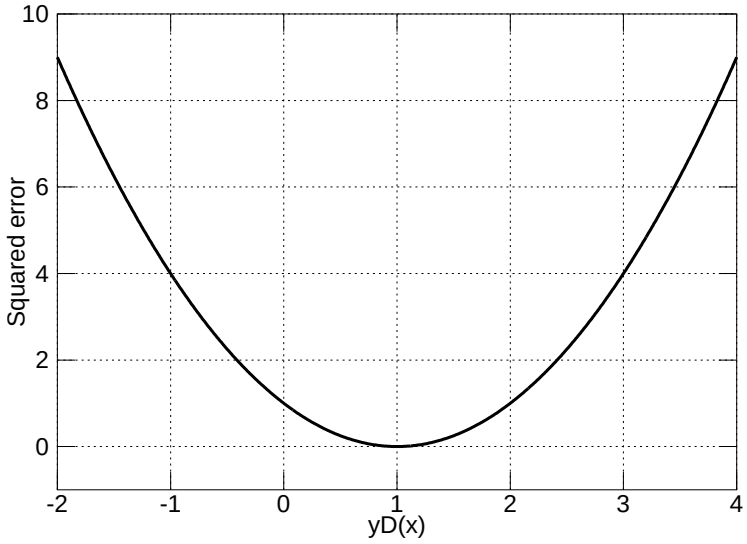


Figure 4.10: Squared error

The basic idea of LSR is to minimize the expectation of the squared error $(y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle)^2$. Since $y^2 = 1$, the error function is given by

$$\begin{aligned} R(D(\boldsymbol{x}), y) &= (y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle)^2 && (4.60) \\ &= y^2 - 2y \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \langle \boldsymbol{w}, \boldsymbol{x} \rangle^2 \\ &= 1 - 2y \langle \boldsymbol{w}, \boldsymbol{x} \rangle + y^2 \langle \boldsymbol{w}, \boldsymbol{x} \rangle^2 \\ &= (1 - y \langle \boldsymbol{w}, \boldsymbol{x} \rangle)^2. && (4.61) \end{aligned}$$

Fig 4.10 depicts the squared error function based on Eq.(4.61). Note that the squared error is a convex function. The expectation of the squared error is given by

$$E[(y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle)] = \sum_y \int_{\mathcal{D}} P(\boldsymbol{x}, y)(y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle)^2 \mathrm{d}\boldsymbol{x}. \tag{4.62}$$

It can be approximated by replacing ensemble mean by sample mean as

$$E[(y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle)] \approx \frac{1}{N} \sum_{n=1}^{N} (y_n - \langle \boldsymbol{w}, \boldsymbol{x}_n \rangle)^2 =: J. \tag{4.63}$$

It can be transformed as

$$\begin{aligned} J &= \frac{1}{N} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) && (4.64) \\ &= \frac{1}{N} (\boldsymbol{y}^T \boldsymbol{y} - 2\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w}), && (4.65) \end{aligned}$$

where

$$\boldsymbol{y} := \begin{pmatrix} y_1 & y_2 & \cdots & y_N \end{pmatrix}^T, \tag{4.66}$$

$$\boldsymbol{X} := \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_N \end{pmatrix}^T. \tag{4.67}$$

The goal of LSR is to calculate the optimal $\boldsymbol{w}$ that minimizes Eq.(4.65). Then, its solution $\widehat{\boldsymbol{w}}_{\text{LSR}}$ can be calculated analytically as follow:

$$\frac{\partial J}{\partial \boldsymbol{w}} = \frac{1}{N}(-2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}) = \boldsymbol{0}, \tag{4.68}$$

$$\widehat{\boldsymbol{w}}_{\text{LSR}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{4.69}$$

We can classify an unlearned pattern $\boldsymbol{x}$ based on the following rule:

$$\hat{y} = \begin{cases} +1 & \text{if } \langle \widehat{\boldsymbol{w}}_{\text{LSR}}, \boldsymbol{x} \rangle > 0 \\ -1 & \text{if } \langle \widehat{\boldsymbol{w}}_{\text{LSR}}, \boldsymbol{x} \rangle < 0 \end{cases}. \tag{4.70}$$

Kernelized LSR

In this section, I explain LSR with the kernel model. We can adapt the kernel model to LSR easily by replacing $\boldsymbol{X}\boldsymbol{w}$ by $\boldsymbol{K}\boldsymbol{\alpha}$. Then, J is given by

$$J = \frac{1}{N}(\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha})^T(\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}) \tag{4.71}$$

$$= \frac{1}{N}(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{y} + \boldsymbol{\alpha}^T\boldsymbol{K}^2\boldsymbol{\alpha}). \tag{4.72}$$

Its solution $\widehat{\boldsymbol{\alpha}}_{\text{LSR}}$ can be calculated analytically as follow:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{1}{N}(-2\boldsymbol{K}\boldsymbol{y} + 2\boldsymbol{K}^2\boldsymbol{\alpha}) = \boldsymbol{0}, \tag{4.73}$$

$$\widehat{\boldsymbol{\alpha}}_{\text{LSR}} = \boldsymbol{K}^{-1}\boldsymbol{y}. \tag{4.74}$$

Regularization

Fig 4.11 is depicts the discriminant functions trained by LSR and kernelized LSR. In LSR, model function is too simple to fit to training samples. On the other hand, function with kernel model is too free, and fits to training samples too precisely. Its problem is called "over-fitting". Regularization can prevent the over-fitting.

Adding $\lambda||\boldsymbol{w}||^2$ to J is often used for regularization, where $\lambda > 0$ is called the regularization parameter. Since $||\boldsymbol{w}||^2 = \boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}$ and $\frac{1}{N}$ can be abbreviated, J is given by

$$J = (\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha})^T(\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}) + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} \tag{4.75}$$

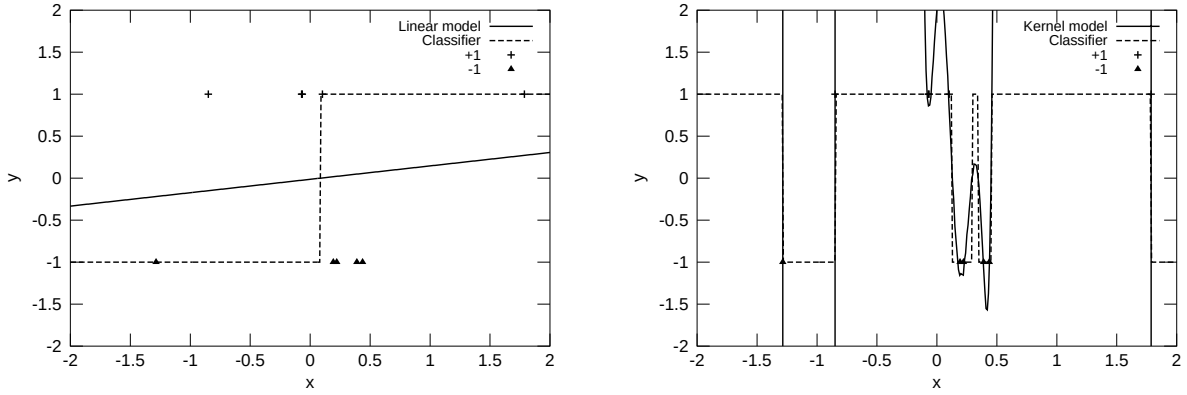
$$= (\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{y} + \boldsymbol{\alpha}^T\boldsymbol{K}(\boldsymbol{K} + \lambda\boldsymbol{I})\boldsymbol{\alpha}), \tag{4.76}$$

where $\boldsymbol{I}$ is the identity matrix. Then its solution is given by

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{1}{N}(-2\boldsymbol{K}\boldsymbol{y} + 2\boldsymbol{K}(\boldsymbol{K} + \lambda\boldsymbol{I})\boldsymbol{w}) = \boldsymbol{0}, \tag{4.77}$$

$$\widehat{\boldsymbol{\alpha}}_{\text{LSR}} = (\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}. \tag{4.78}$$

Fig 4.12 depicts the discriminant function trained by LSR with this regularization method. We can see that the over-fitting is prevented.



(a) LSR (b) Kernelized LSR

Figure 4.11: Examples of LSR and kernelized LSR

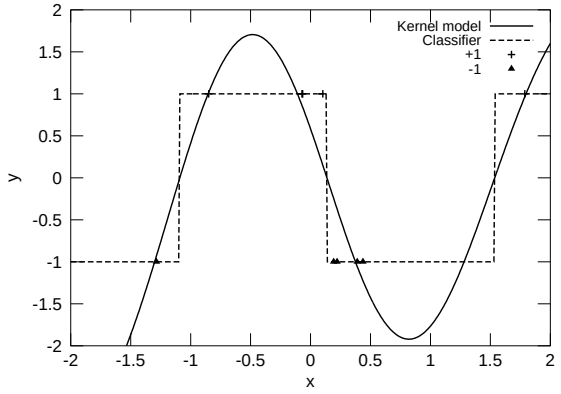


Figure 4.12: Regularization

Model Selection

Many kernel functions have kernel parameters such as d of the polynomial kernel and γ of the Gaussian kernel. Such a parameter is called the model parameter too. If the model parameter is chosen improperly, it degrades the classifiers. The task of the model is to estimate the optimal model parameter such that the best classifier is obtained. Here, I introduce the cross validation method [7] that is very famous and effective.

For the cross validation, we separate training samples into a training set and a test set, and to select the optimal parameter based on its accuracy. For example, we divide samples into 4 parts and there are 4 combinations to separate them into 3 parts of training set and 1 part of test set. Fig 4.13 depicts the 4-fold cross validation method. This method is generalized as the k-fold cross validation. Especially, when we assume that the number of samples is N, a method that separates them into $(N-1)$ samples in training set and 1 samples in test set is called the leave-one-out cross validation.

These methods are useful to select not only the model parameter, but also the regularization pa-

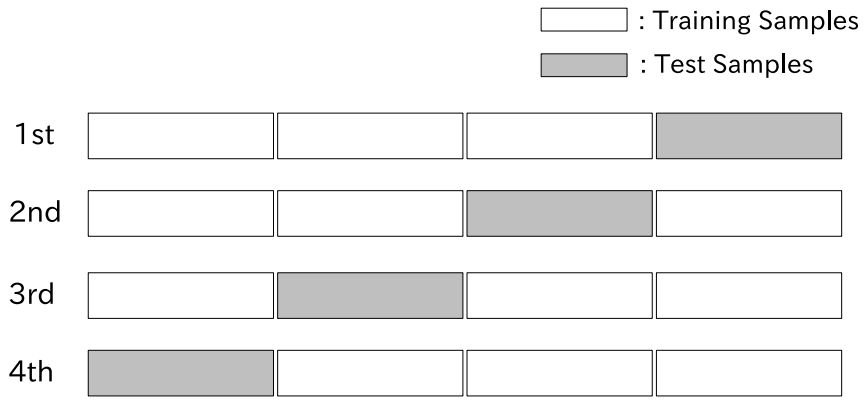


Figure 4.13: 4-fold cross validation

rameter. But the computational complexity becomes much higher.

4.3.4 Fisher Linear Discriminant Analysis

I introduce the Fisher's linear discriminant analysis (FDA). The FDA is a very famous binary classification method. It is based on mean vectors and covariance matrices of patterns for individual classes. We consider the linear model as $z = \boldsymbol{w}^T \boldsymbol{x}$. The FDA gives an optimal projection $\boldsymbol{w}$ so that the distribution of z is easily discriminated. Fig. 4.14 shows the concept of FDA, and there are two projection vectors $\boldsymbol{w}$ and $\boldsymbol{w}^\star$. We can see that $\boldsymbol{w}^\star$ is clearly easier to discriminate patterns than $\boldsymbol{w}$. This concept can be provided by the following FDA criterion

$$\text{maximize}\, J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1 + s_2}, \tag{4.79}$$

where

- m_1 and m_2 denote averages for $z_n \in \{$ class 1 $\}$ and $z_n \in \{$ class 2 $\}$, respectively,
- s_1 and s_2 denote variances for $z_n \in \{$ class 1 $\}$ and $z_n \in \{$ class 2 $\}$, respectively.

We have

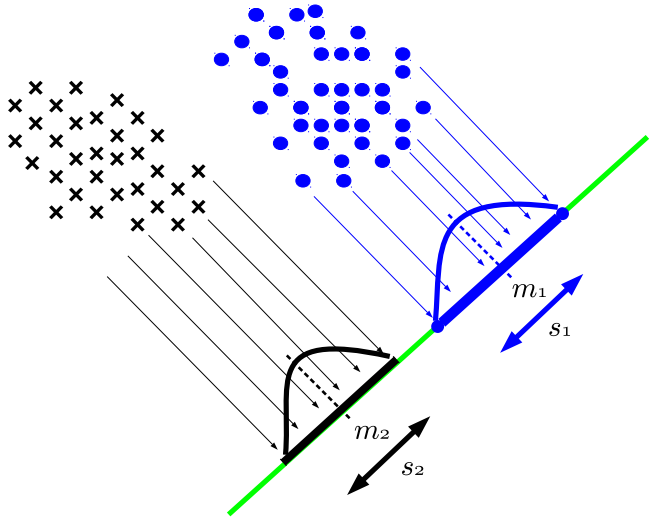
$$\begin{aligned}(m_1 - m_2)^2 &= \left(\boldsymbol{w}^T \boldsymbol{\mu}_1 - \boldsymbol{w}^T \boldsymbol{\mu}_2\right) \left(\boldsymbol{w}^T \boldsymbol{\mu}_1 - \boldsymbol{w}^T \boldsymbol{\mu}_2\right)^T \\ &= \boldsymbol{w}^T \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right) \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w},\end{aligned} \tag{4.80}$$

and

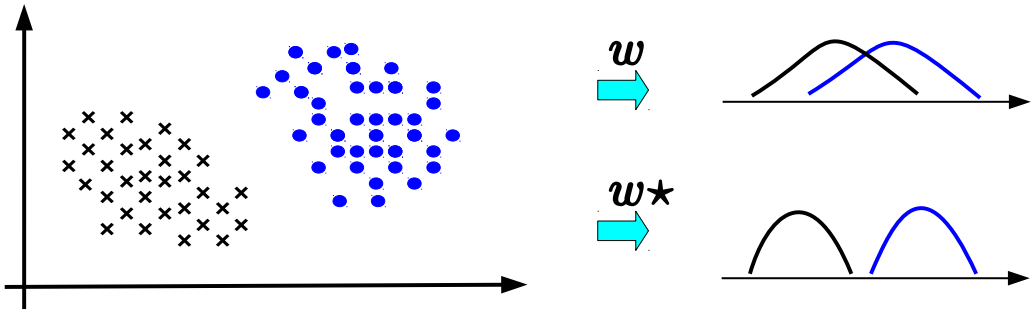
$$\begin{aligned}s_1 + s_2 &= \boldsymbol{w}^T \boldsymbol{\Sigma}_1 \boldsymbol{w} + \boldsymbol{w}^T \boldsymbol{\Sigma}_2 \boldsymbol{w} \\ &= \boldsymbol{w}^T \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right) \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w},\end{aligned} \tag{4.81}$$

where

- $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ denote mean vectors for $\boldsymbol{x}_n \in \{$ class 1 $\}$ and $\boldsymbol{x}_n \in \{$class 2 $\}$, respectively,



(a) Linear projection



(b) Examples of projection results

Figure 4.14: Fisher's Linear Discriminant Analysis

- $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ denote covariance matrices for $\boldsymbol{x}_n \in \{$ class 1 $\}$ and $\boldsymbol{x}_n \in \{$ class 2 $\}$, respectively.

The cost function $J(\boldsymbol{w})$ in original space can be written as

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}, \tag{4.82}$$

and then this solution is given by

$$\widehat{\boldsymbol{w}} \propto \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{4.83}$$

Finally, we choose an optimal threshold z_0, then we can classify any $\boldsymbol{x}$ by

$$\widehat{\boldsymbol{w}}^T \boldsymbol{x} \geq z_0 \rightarrow \boldsymbol{x} \in \{\text{class 1}\}, \tag{4.84}$$

$$\widehat{\boldsymbol{w}}^T \boldsymbol{x} < z_0 \rightarrow \boldsymbol{x} \in \{\text{class 2}\}. \tag{4.85}$$

For example, $z_0 = (m_1 + m_2)/2$ could be usable.

Problem of FDA

In this section, I explain that FDA does not provide the best optimal hyper-plane in heteroscedastic Gaussian case. We denote the probability density functions of y which belongs to class 1 and 2 by

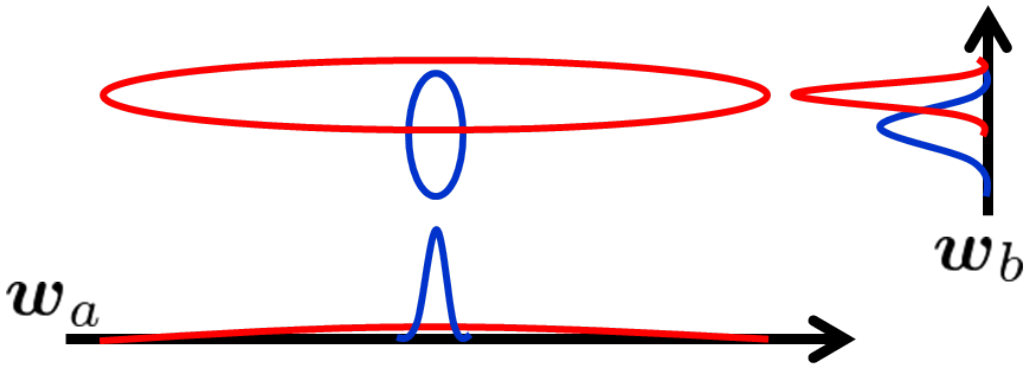


Figure 4.15: Example of which FDA does not work well

$f_1(y)$ and $f_2(y)$, and the prior probabilities of class 1 and 2 by p_1 and p_2, respectively. We assume $f_1(y)$ and $f_2(y)$ follows the Gaussian distributions. Then, the Bayes decision rule for the minimum error is given by

$$\boldsymbol{x} \text{ is assigned to } \begin{cases} \Omega_1 & \text{if } p_1 f_1(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) > p_2 f_2(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) \\ \Omega_2 & \text{if } p_1 f_1(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) < p_2 f_2(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) \end{cases} . \tag{4.86}$$

Fig. 4.15 shows a simple example of which FDA does not work well. Red and blue ellipses show individual Gaussian distributions in this figure. We consider two projection vectors $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$. The FDA evaluation function of $\boldsymbol{w}_a$ becomes $J(\boldsymbol{w}_a) = 0$ which is the lowest value; then, FDA has to select $\boldsymbol{w}_b$, preferentially. However, the classification rate by $\boldsymbol{w}_a$ is better than $\boldsymbol{w}_b$ under the classification rule (4.86). In this way, the improvement of evaluation function is necessary for the heteroscedastic Gaussian case.

4.3.5 Support Vector Machine

Support vector machine (SVM) is a typical kernel classifier and it is an extension of the optimal hyper plane classifier (OHC) by the kernel method. [102, 101] OHC and SVM are trained by quadratic programming. In this section, I explain the OHC at first. After that, I explain the SVM as an extension of OHC.

Optimal Hyperplane classifier

We assume a binary classification problem as $y \in \{+1, -1\}$. The discriminant function of OHC is given by

$$D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \theta. \tag{4.87}$$

Fig. 4.16 depicts an example of OHC in 2-dimensional space. The boundary of two categories is given by a $(d-1)$-dimensional hyperplane described as $D(\boldsymbol{x}) = 0$, where d is the dimension of $\boldsymbol{x}$. In this

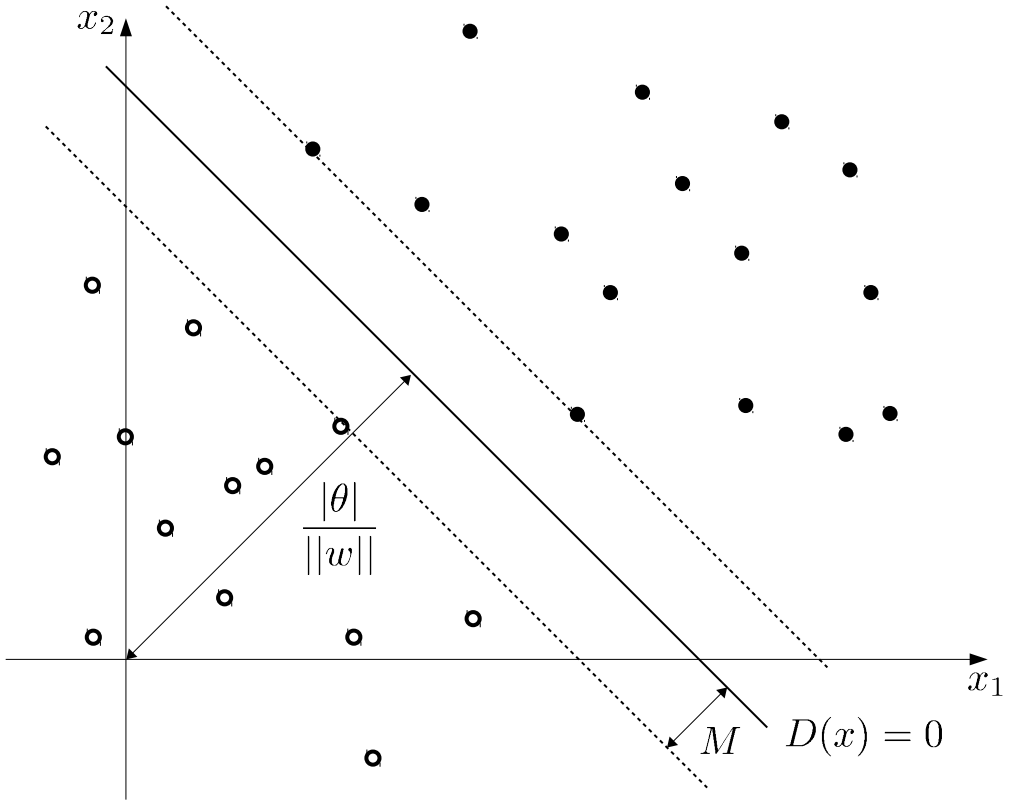


Figure 4.16: Basic concept of optimal hyperplane classifier

figure, M stands for "margin" that is the minimum distance between the boundary and its nearest samples. Basic concept of OHC is to maximize "margin" that is described by "M" in Fig. 4.16. The margin is given by

$$M = \min_{n=1,\ldots,N} \frac{y_n D(\boldsymbol{x}_n)}{\|\boldsymbol{w}\|}. \tag{4.88}$$

Furthermore, we normalize the numerator of Eq. (4.88) as

$$\min_{n=1,\ldots,N} y_n D(\boldsymbol{x}_n) = 1, \tag{4.89}$$

then the margin is given by

$$M = \frac{1}{||\boldsymbol{w}||}. \tag{4.90}$$

From Eqs.(4.89) and (4.90), OHC is trained by the following quadratic programming problem,

$$\text{minimize} \quad \frac{1}{2}||\boldsymbol{w}||^2 \tag{4.91}$$

$$\text{subject to} \quad y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) \geq 1, \quad n = 1, \ldots, N. \tag{4.92}$$

We can solve it directly, however it can be simplified by transforming to a dual problem [68]. We define Lagrange's function L as

$$L(\boldsymbol{w}, \theta, \boldsymbol{\gamma}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=1}^{N} \gamma_n (y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) - 1), \tag{4.93}$$

where $\gamma_n \geq 0$ are Lagrange coefficients. Lagrange primal and dual problems are expressed as

$$P \quad : \quad \min_{\boldsymbol{w},\theta} \sup_{\boldsymbol{\gamma} \geq 0} L(\boldsymbol{w}, \theta, \boldsymbol{\gamma}), \tag{4.94}$$

$$D \quad : \quad \max_{\boldsymbol{\gamma} \geq 0} \inf_{\boldsymbol{w},\theta} L(\boldsymbol{w}, \theta, \boldsymbol{\gamma}), \tag{4.95}$$

respectively. The two problems are equivalent. From the Karuch-Kuhn-Tucker (KKT) conditions we have

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{\gamma} = \boldsymbol{0}, \quad \frac{\partial L}{\partial \theta} = \boldsymbol{\gamma}^T \boldsymbol{y} = 0, \tag{4.96}$$

$$\gamma_n (y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) - 1) = 0, \tag{4.97}$$

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) - 1 \geq 0, \quad \gamma_n \geq 0 \quad n = 1, \ldots, N, \tag{4.98}$$

where

$$Y := \mathrm{diag}\,(\boldsymbol{y})\,. \tag{4.99}$$

Then, the dual problem is given by

$$\text{minimize} \quad -\frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{Y} \boldsymbol{X} \boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{\gamma} + \boldsymbol{1}^T \boldsymbol{\gamma} \tag{4.100}$$

$$\text{subject to} \quad \boldsymbol{y}^T \boldsymbol{\gamma} = 0, \quad \gamma_n \geq 0, \quad n = 1, \ldots, N, \tag{4.101}$$

where $\boldsymbol{1}$ is an N-dimensional vector of which all elements are one. Its solution $\widehat{\boldsymbol{\gamma}}$ is almost always sparse. Then, let S be the set of indexes of non-zero $\widehat{\gamma}_n$ as

$$S = \{n | \gamma_n \neq 0, \quad n = 1, \ldots, N\}. \tag{4.102}$$

The solution of the original problem is described as

$$\widehat{\boldsymbol{w}} = \sum_{n=1}^{N} y_n \widehat{\gamma}_n \boldsymbol{x}_n = \sum_{s \in S} y_s \widehat{\gamma}_s \boldsymbol{x}_s, \tag{4.103}$$

$$\widehat{\theta} = \frac{1}{|S|} \sum_{s \in S} (\widehat{\boldsymbol{w}}^T \boldsymbol{x}_s - y_s). \tag{4.104}$$

Eqs. (4.103) and (4.104) imply the optimal parameters depend only on the support vectors.

This method is also called the hard margin method. In this method, training samples that can not be separated linearly are not allowed. It is a serious issue, so that the soft margin method was proposed.

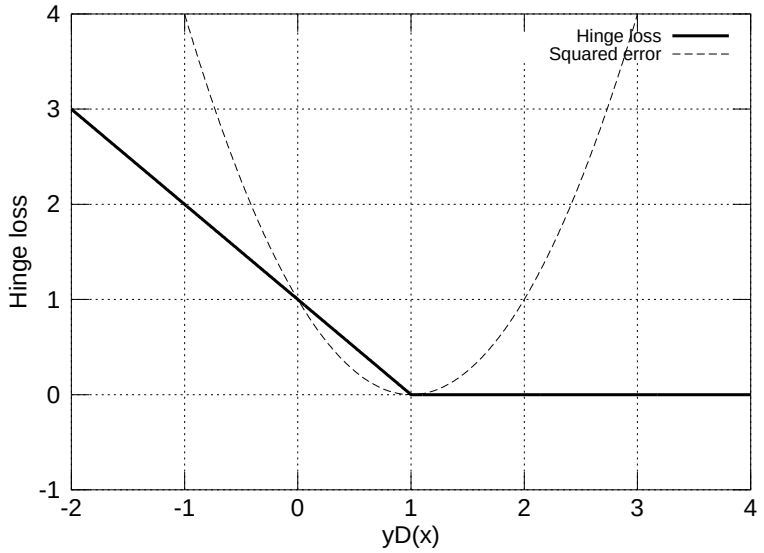


Figure 4.17: Hinge loss

Soft margin

For the soft margin, we introduce slack variables $\xi_n \geq 0$ to relent the constraints Eq. (4.92) as follow:

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) \geq 1 - \xi_n \quad n = 1, \ldots, N. \tag{4.105}$$

The variable ξ_n expresses how much condition (4.92) is not satisfied. ξ_n are called the hinge loss and it should be minimized as well as $||\boldsymbol{w}||$. ξ_n is also given by

$$\xi_n = \max(1 - y_n D(\boldsymbol{x}_n), 0). \tag{4.106}$$

Fig. 4.17 depicts the comparison between the hinge loss function and the squared loss function. The hinge loss is a piecewise linear function, and note that the hinge loss is a convex function. Since the hinge loss is evaluated linearly, then it is more robust for outlier than the squared error.

The OHC with soft margin is trained by the following quadratic programming:

$$\text{minimize} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} \xi_i \tag{4.107}$$

$$\text{subject to} \quad y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + \theta) \geq 1 - \xi_n \quad n = 1, \ldots, N, \tag{4.108}$$

where λ is a constant parameter to adjust weights between the hinge loss term and the margin because both terms have to be minimized. The margin term $||\boldsymbol{w}||^2$ and λ are regarded as a regularization term and a regularization parameter.

Its dual problem is given by

$$\text{minimize} \quad -\frac{1}{2\lambda}\boldsymbol{\gamma}^T \boldsymbol{Y}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{\gamma} + \mathbf{1}^T\boldsymbol{\gamma} \tag{4.109}$$

$$\text{subject to} \quad 0 \leq \gamma_n \leq 1, \quad \boldsymbol{y}^T\boldsymbol{\gamma} = 0, \quad n = 1, \ldots, N, \tag{4.110}$$

where $\boldsymbol{\gamma}$ is a dual parameter vector. Then, the solution is calculated by

$$\widehat{\boldsymbol{w}} = \frac{1}{\lambda}\sum_{s\in S} y_s \widehat{\gamma}_s \boldsymbol{x}_s, \tag{4.111}$$

$$\widehat{\theta} = \frac{1}{|S|}\sum_{s\in S}(\widehat{\boldsymbol{w}}^T \boldsymbol{x}_s - y_s). \tag{4.112}$$

Support Vector Machine

Support Vector Machine (SVM) is a kernelized version of OHC. That is the OHC defined in high dimensional feature space.

Since $D(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + \theta$ and $||\boldsymbol{w}|| = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$, SVM is trained by the following quadratic programming:

$$\text{minimize} \quad \frac{\lambda}{2}\boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha} + \sum_{i=1}^{n}\xi_i \tag{4.113}$$

$$\text{subject to} \quad y_n\left(\sum_{m=1}^{N}\alpha_m k(\boldsymbol{x}_m, \boldsymbol{x}_n) + \theta\right) \geq 1 - \xi_n \quad n = 1, \ldots, N. \tag{4.114}$$

And its dual problem is given by

$$\text{minimize} \quad -\frac{1}{2\lambda}\boldsymbol{\gamma}^T \boldsymbol{K}\boldsymbol{\gamma}^T + \mathbf{1}^T\boldsymbol{\gamma} \tag{4.115}$$

$$\text{subject to} \quad 0 \leq \gamma_n \leq 1, \quad \boldsymbol{y}^T\boldsymbol{\gamma} = 0, \quad n = 1, \ldots, N, \tag{4.116}$$

where $\boldsymbol{\gamma}$ is a dual parameter vector. Its solution is calculated by

$$\widehat{\boldsymbol{\alpha}} = \frac{1}{\lambda}\boldsymbol{Y}\widehat{\boldsymbol{\gamma}}, \tag{4.117}$$

$$\widehat{\theta} = \frac{1}{|S|}\sum_{s\in S}\left(\sum_{n=1}^{N}\alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}_s) - y_s\right). \tag{4.118}$$

Chapter 5

Proposed Feature Extraction methods

In this Chapter, I propose several new methods for matrix and tensor based feature extractions. This chapter consists of three parts, mainly. The first part is a proposition of new methods and algorithms for smooth nonnegative matrix and tensor factorizations. I propose a new fast algorithm for the GRBF-NMF, several extension methods, and its tensor versions with the CP and the Tucker models. The second part is a proposition of new model and algorithms for the common and individual feature analysis (CIFA) for the multi-block CP decomposition. I call this method as the "linked CP tensor decomposition" (LCPTD). Furthermore, I propose the sparse LCPTD and the nonnegative LCPTD. The third part is a proposition of new model and algorithms for the common and individual feature analysis (CIFA) for the multi-block Tucker decomposition. I call this method as the "linked Tucker decomposition" (LTD). Furthermore, I propose the orthogonal LTD, the sparse LTD, and the nonnegative LTD.

5.1 Improvements of NMF with function approximation

In this section, I propose an improved algorithm for function approximation based smooth NMF (GRBF-NMF) introduced in Section 3.1.6. Furthermore, I propose some extended methods and tensor versions.

5.1.1 Fast algorithm for GRBF-NMF

First, I modify the GRBF-NMF optimization problem (3.95) as

$$\begin{aligned} &\underset{\boldsymbol{W},\boldsymbol{X}}{\text{minimize}} \ ||\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}||_F^2, \\ &\text{s.t. } \boldsymbol{W} \geq 0, \ \boldsymbol{X} \geq 0, \ ||\boldsymbol{x}_r||^2 = 1 \text{ for } r = 1, ..., R, \end{aligned} \tag{5.1}$$

where $\boldsymbol{x}_r^T$ is the r-th row vector of $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_R]^T$. In this problem, the constraint $\boldsymbol{\Phi}\boldsymbol{W} \geq 0$ is replaced by $\boldsymbol{W} \geq 0$. Since $\boldsymbol{\Phi} \geq 0$, $\boldsymbol{W} \geq 0$ is a sufficient condition for $\boldsymbol{\Phi}\boldsymbol{W} \geq 0$. Under this constraint, the flexibility of model will be decreased; however it is not necessarily bad. In many regression models,

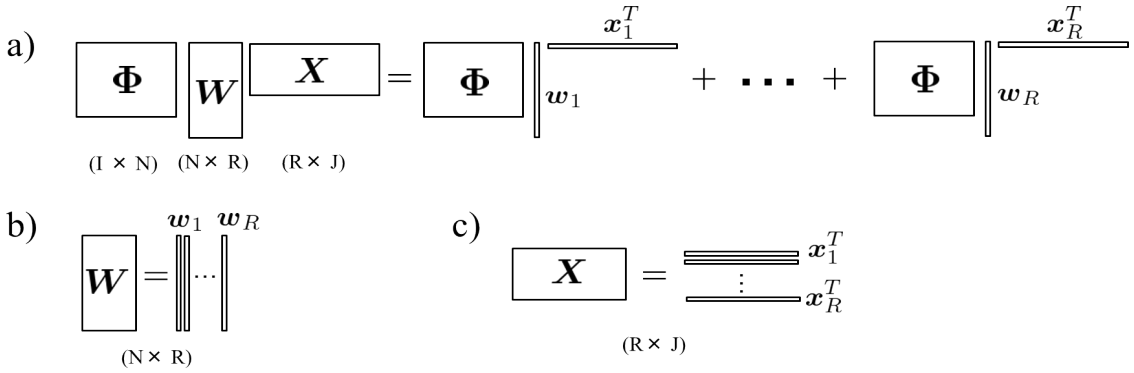


Figure 5.1: Decomposition of GRBF-NMF model

a highly flexible model without any regularization has a problem of over-fitting. When the data is noisy, such a highly flexible model often represents not only the main feature but also noise and outliers. In order to prevent the over-fitting, one of the solutions is to use a simpler model, and the proposed model is simpler than the original model since I do not allow negative value for $\boldsymbol{W}$. Thus, the proposed model can be considered as more robust for noise and outliers than the original model. In addition, I impose a constraint $||\boldsymbol{x}_r||^2 = 1$ which does not change the flexibility. This constraint becomes a key-point for the new fast algorithm.

In order to obtain the solution of (5.1), I consider to separate (5.1) into sub-problems based on the HALS method [24]. Since the GRBF-NMF model can be decomposed as $\boldsymbol{\Phi W X} = \boldsymbol{\Phi w}_1\boldsymbol{x}_1^T + \boldsymbol{\Phi w}_2\boldsymbol{x}_2^T + \cdots + \boldsymbol{\Phi w}_R\boldsymbol{x}_R^T$ (See Fig 5.1), the problem (5.1) can be separated as the following sub-problems:

$$\underset{\boldsymbol{x}_r}{\text{minimize}} \ ||\boldsymbol{Y}_r - \boldsymbol{\Phi w}_r\boldsymbol{x}_r^T||_F^2, \ \text{s.t.} \ \boldsymbol{x}_r \geq 0, \ ||\boldsymbol{x}_r||^2 = 1. \tag{5.2}$$

$$\underset{\boldsymbol{w}_r}{\text{minimize}} \ ||\boldsymbol{Y}_r - \boldsymbol{\Phi w}_r\boldsymbol{x}_r^T||_F^2, \ \text{s.t.} \ \boldsymbol{w}_r \geq 0, \tag{5.3}$$

where $\boldsymbol{Y}_r := \boldsymbol{Y} - \sum_{k\neq r} \boldsymbol{\Phi w}_k\boldsymbol{x}_k^T$.

First I consider the problem (5.2). Since $\boldsymbol{\Phi w}_r$ is currently fixed and it is equivalent to a simple least squares problem with a nonnegativity constraint. Thus, its problem has been already studied in many papers such as [19, 24, 26]. Then, the update rule is given by

$$\boldsymbol{x}_r \leftarrow [\boldsymbol{Y}_r^T\boldsymbol{\Phi w}_r]_+, \tag{5.4}$$

$$\boldsymbol{x}_r \leftarrow \boldsymbol{x}_r/||\boldsymbol{x}_r||, \tag{5.5}$$

where $[x]_+ := \max(x, \varepsilon)$, and ε is a very small positive value (e.g., $\varepsilon = 10^{-16}$).

Next, I consider the problem (5.3). The objective function can be transformed as

$$\begin{aligned} &||\boldsymbol{Y}_r - \boldsymbol{\Phi w}_r\boldsymbol{x}_r^T||_F^2 \\ &= \text{tr}(\boldsymbol{Y}_r^T\boldsymbol{Y}_r) - 2\text{tr}(\boldsymbol{Y}_r^T\boldsymbol{\Phi w}_r\boldsymbol{x}_r^T) + \text{tr}(\boldsymbol{w}_r^T\boldsymbol{\Phi}^T\boldsymbol{\Phi w}_r) \end{aligned} \tag{5.6}$$

since $\boldsymbol{x}_r^T\boldsymbol{x}_r = 1$. By computing the gradient of the objective function, the condition for a solution is

given by

$$\frac{\partial}{\partial \boldsymbol{w}_r}||\boldsymbol{Y}_r - \boldsymbol{\Phi}\boldsymbol{w}_r\boldsymbol{x}_r^T||_F^2 = 2\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}_r - 2\boldsymbol{\Phi}^T\boldsymbol{Y}_r\boldsymbol{x}_r = 0. \tag{5.7}$$

This direct solution will be $\boldsymbol{w}_r = [(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{Y}_r\boldsymbol{x}_r]_+$; however, the inverse problem is unstable and high-cost. Thus, I do not employ this update rule, but propose a more effective iterative update rule. Note that the following problem is equivalent to (5.3) via the same differential since $||\boldsymbol{x}_r|| = 1$:

$$\underset{\boldsymbol{w}_r}{\text{minimize}}\ ||\boldsymbol{Y}_r\boldsymbol{x}_r - \boldsymbol{\Phi}\boldsymbol{w}_r||^2,\ \text{s.t.}\ \boldsymbol{w}_r \geq 0. \tag{5.8}$$

This problem can be assigned to NNLS (nonnegative least squares). Finally, I employ the following multiplicative update rule:

$$\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast (\boldsymbol{\Phi}^T\boldsymbol{Y}_r\boldsymbol{x}_r) \oslash (\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}_r)]_+, \tag{5.9}$$

where $\circledast$ and $\oslash$ are respectively element-wise multiplication and element-wise division.

The algorithm is summarized in Algorithm 18. I call this method the fastGRBF-NMF. The computational cost for each iteration of the fastGRBF-NMF is very low because it consists of only four operations and a thresholding without any complex optimization procedure. If we assume that $N = I$ and $I > J > R$, the computational complexity of one update for $\boldsymbol{W}$ in the fastGRBF-NMF algorithm is $\mathcal{O}(I^2R)$. On the other hand, if we assume the QP optimization in the GRBF-NMF algorithm includes the Cholesky decomposition, its computational complexity is $\mathcal{O}(I^3R^3)$ because the number of parameters are IR.

The computational efficiency is often considered as a more important factor than the strictness of solution, especially for a large-scale problem. Since the NMF solution is not unique, the most important viewpoint is a balance between computational efficiency and a meaningful solution. In Section 7.1, we can confirm that the proposed method provides not only the improvement of computational efficiency but also meaningful results.

5.1.2 Extensive Bases for the fastGRBF-NMF

In this section, I discuss how to choose $\boldsymbol{\Phi}$ more efficiently. I propose to use various bases $\boldsymbol{\phi}$ to make $\boldsymbol{\Phi}$, for example multiple bases with various σ can be mixed. Thus, I propose to construct $\boldsymbol{\Phi}$ by

$$\boldsymbol{\Phi} = [\boldsymbol{\Phi}_{\sigma_1}, \boldsymbol{\Phi}_{\sigma_2}, ..., \boldsymbol{\Phi}_{\sigma_U}] \in \mathbb{R}^{I\times N}, \tag{5.10}$$

where U is the number of σ_u and $\boldsymbol{\Phi}_{\sigma_u}$ stands for the basis matrix with a standard deviation σ_u. For example, if we set

$$\sigma_1 = \sigma,\ \sigma_2 = 2\sigma,\ \sigma_3 = 4\sigma,\ ...,\ \sigma_U = 2^{U-1}\sigma, \tag{5.11}$$

$$\Delta t_1 = \delta t,\ \Delta t_2 = 2\delta t,\ \Delta t_3 = 4\delta t,\ ...,\ \Delta t_U = 2^{U-1}\delta t, \tag{5.12}$$

then $N < 2I/\delta t$ holds for any U. Let us put the horizontal size of $\boldsymbol{\Phi}_{\sigma_1}$ as N_0, then the horizontal size of $\boldsymbol{\Phi}$ is roughly bounded as $N = N_0 + \frac{1}{2}N_0 + \frac{1}{4}N_0 + \cdots + \frac{1}{2^{U-1}N_0} < 2N_0$ for any U. Thus, the

Algorithm 18 FastGRBF-HALS algorithm

Input: $\boldsymbol{Y}$, R, and $\boldsymbol{\Phi}$
Initialize: $\boldsymbol{W}$ and $\boldsymbol{X}$
$\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{W}\boldsymbol{X}$;
repeat
  for $r = 1, \ldots, R$ **do**
    $\boldsymbol{Y}_r \leftarrow \boldsymbol{E} + (\boldsymbol{\Phi}\boldsymbol{w}_r)\boldsymbol{x}_r^T$;
    $\boldsymbol{x}_r \leftarrow [\boldsymbol{Y}_r^T(\boldsymbol{\Phi}\boldsymbol{w}_r)]_+$;
    $\boldsymbol{x}_r \leftarrow \boldsymbol{x}_r / ||\boldsymbol{x}_r||$;
    $\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast \{\boldsymbol{\Phi}^T(\boldsymbol{Y}_r\boldsymbol{x}_r)\} \oslash \{\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{w}_r)\}]_+$;
    $\boldsymbol{E} \leftarrow \boldsymbol{Y}_r - (\boldsymbol{\Phi}\boldsymbol{w}_r)\boldsymbol{x}_r^T$
  end for
until $||\boldsymbol{E}||_F^2$ converges
Output: $\boldsymbol{W}$ and $\boldsymbol{X}$

increase of complexity (which is double at the maximum) is not a large problem since the proposed algorithm is well improved to be faster by the HALS based method. Furthermore, I propose to add an additional DC component, then $\boldsymbol{\Phi}$ is given by

$$\boldsymbol{\Phi} = [\boldsymbol{\Phi}_{\sigma_1}, \boldsymbol{\Phi}_{\sigma_2}, ..., \boldsymbol{\Phi}_{\sigma_U}, \mathbf{1}]. \tag{5.13}$$

Finally, we have $N = \sum_{u=1}^{U}[\lfloor (I-1)/(2^{u-1}\delta t)\rfloor + 1] + 1$. By this extension we can express the data with various resolutions.

2D-basis

I propose another kind of the Gaussian basis function for smooth image signals. I assume that the observed signal $\boldsymbol{y}_j \in \mathbb{R}^I$ is an unfolded vector of an $(I_1 \times I_2)$-matrix $\boldsymbol{Y}_j \in \mathbb{R}^{I_1 \times I_2}$, where $I = I_1 I_2$. In this case, if we use $\boldsymbol{\phi}_n$ in (3.94), it represents only vertical smoothness. However, natural images usually have vertical and horizontal smoothnesses. Thus I consider the following 2D-basis:

$$\boldsymbol{\Phi}_n^{(2)}(i_1, i_2) := \exp\left[\frac{1}{2\sigma^2}\left|\left|\begin{pmatrix} i_1 \\ i_2 \end{pmatrix} - \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}\right|\right|^2\right] \in \mathbb{R}^{I_1 \times I_2}, \tag{5.14}$$

where

$$n_1 = \{(n\Delta t - 1) \bmod I_1\} + 1, \tag{5.15}$$

$$n_2 = \lfloor (n\Delta t - 1)/I_1 \rfloor + 1. \tag{5.16}$$

Thus, we have $n\Delta t = (n_2 - 1)I_1 + n_1$, $1 \le n_1 \le I_1$, and $1 \le n_2 \le I_2$. In practice, I unfold the matrix $\boldsymbol{\Phi}_n^{(2)}$ into a vector as $\boldsymbol{\phi}_n^{(2)} = \mathrm{vec}(\boldsymbol{\Phi}_n^{(2)}) \in \mathbb{R}^I$, and construct the basis matrix $\boldsymbol{\Phi}^{(2)}$ as

$$\boldsymbol{\Phi}^{(2)} = [\boldsymbol{\phi}_1^{(2)}, \boldsymbol{\phi}_2^{(2)}, ..., \boldsymbol{\phi}_N^{(2)}] \in \mathbb{R}^{I \times N}. \tag{5.17}$$

Its multi-σ version can be also applied in the same way as

$$\boldsymbol{\Phi} = [\boldsymbol{\Phi}^{(2)}_{\sigma_1}, \boldsymbol{\Phi}^{(2)}_{\sigma_2}, ..., \boldsymbol{\Phi}^{(2)}_{\sigma_U}, \mathbf{1}]. \tag{5.18}$$

5.1.3 Reduction of the size of Basis Matrix

The function approximation method occupies a lot of memory when the size of basis matrix $\boldsymbol{\Phi}$ is very large. It will be $8I^2$ bytes in double-precision floating-point with $U = 1$ and $\delta t = 1$. For example, when we factorize several (256×256)-images by NMF, each image must be often unfolded to a 65536-dimensional vector. Then the size of basis matrix $\boldsymbol{\Phi}$ becomes (65536×65536) which needs about 34.36 Gbytes. This is a critical issue of the GRBF and the fastGRBF methods. In this section, I propose a method to reduce the size of basis matrix $\boldsymbol{\Phi}$. We fold a observed signal $\boldsymbol{y}_j$ to matrix $\boldsymbol{Y}^{(j)}$, for example a 1000-dimensional-vector is folded to a (100×10)-matrix. Next, we replace the observed matrix by $\boldsymbol{Y}' = [\boldsymbol{Y}^{(1)}, ..., \boldsymbol{Y}^{(J)}]$. In the previous example we fold the 1000-dimensional-vectors to (100×10)-matrices, then we can reduce the matrix size of $\boldsymbol{\Phi}$ by almost 99 % of the conventional method.

For image data, it is effective to split an image into several blocks and vectorize these blocks. I assume the image $\boldsymbol{Y}_j$ is given by

$$\boldsymbol{Y}_j = \begin{pmatrix} \boldsymbol{D}^{(j)}_{11} & \cdots & \boldsymbol{D}^{(j)}_{1q} \\ \vdots & \ddots & \vdots \\ \boldsymbol{D}^{(j)}_{p1} & \cdots & \boldsymbol{D}^{(j)}_{pq} \end{pmatrix} \in \mathbb{R}^{I_1 \times I_2}. \tag{5.19}$$

Then its folded data is given by $\boldsymbol{Y}^{(j)} = [\mathrm{vec}(\boldsymbol{D}^{(j)}_{11}),\ \mathrm{vec}(\boldsymbol{D}^{(j)}_{12}),\ ...,\ \mathrm{vec}(\boldsymbol{D}^{(j)}_{pq})] \in \mathbb{R}^{I_1 I_2/(pq) \times pq}$. In this case, 2D-basis can be applicable and useful. I call this method the fastGRBF-block-NMF.

Please note that it is related to the block transform by 2D discrete cosine transform (2D-DCT) [1] for image compression because it often splits an image into (8×8)-blocks and factorizes each block by 64 2D-cosine bases. If we apply the 2D-cosine bases to $\boldsymbol{\Phi}$, it becomes similar to the block transform. However, it is different from the block transform because we estimate constrained (e.g., low-rank approximation) and the most correlated R bases modeled with the 2D-cosine bases by a function approximation.

A problem is how to fold the large-scale vector when the dimension is a prime number. In this case, there are two approaches: to reduce the dimension or to expand the dimension. Although the reduction approach has a possibility to lose some information, its computational cost will be decreased. On the other hand, the expand approach does not lose any information; however, computational cost and wasted information will be added. I propose a repeat type and a symmetry type as the methods of the expansion approach. Let us consider to expand $\boldsymbol{y} \in \mathbb{R}^N$ to $\boldsymbol{z} \in \mathbb{R}^{N+M}$. When $\boldsymbol{y}$ has a cyclic feature, it might be proper to use the repeat type which is given as the expansion from $\boldsymbol{y} = [y_1, y_2, ..., y_N]^T$ to $\boldsymbol{z} = [y_1, y_2, ..., y_N, y_1, y_2, ..., y_M]^T$. The symmetry type is given as the expansion from $\boldsymbol{y} = [y_1, y_2, ..., y_N]^T$ to $\boldsymbol{z} = [y_1, y_2, ..., y_N, y_N, y_{N-1}, ..., y_{N-M+1}]^T$, and it could be well worked to prevent the influence of additional information for any $\boldsymbol{y}$.

5.1.4 Tensor Extension

Nonnegative tensor decompositions found applications in positron emission tomography (PET), electroencephalography (EEG), spectroscopy, chemometrics and environmental science [26, 9, 82]. There are two models in tensor decompositions, mainly.

The first model is Tucker model [98]. The three-mode version of the Tucker (Tucker-3) model can be expressed as

$$\begin{aligned}\underline{\boldsymbol{Y}} &\simeq \underline{\boldsymbol{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \\ &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1,r_2,r_3} \boldsymbol{a}_{r_1} \circ \boldsymbol{b}_{r_2} \circ \boldsymbol{c}_{r_3},\end{aligned} \tag{5.20}$$

where $\underline{\boldsymbol{Y}} \in \mathbb{R}_+^{I \times J \times K}$ is a observed tensor signal, $\underline{\boldsymbol{G}} \in \mathbb{R}_+^{R_1 \times R_2 \times R_3}$ is a core tensor, $\boldsymbol{A} \in \mathbb{R}_+^{I \times R_1}$, $\boldsymbol{B} \in \mathbb{R}_+^{J \times R_2}$, $\boldsymbol{C} \in \mathbb{R}_+^{K \times R_3}$ are factor matrices, $\times_k$ is the k-th way tensor and matrix product, and $\circ$ means outer product. Its nonnegativity constrained decomposition is called the nonnegative Tucker decomposition (NTD). The Tucker model can be rewritten by the following matricized and vectorized forms as

$$\boldsymbol{Y}_{(1)} \simeq \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T), \tag{5.21}$$

$$\boldsymbol{Y}_{(2)} \simeq \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C}^T \otimes \boldsymbol{A}^T), \tag{5.22}$$

$$\boldsymbol{Y}_{(3)} \simeq \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B}^T \otimes \boldsymbol{A}^T), \tag{5.23}$$

$$\bar{\boldsymbol{y}} \simeq (\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\bar{\boldsymbol{g}}, \tag{5.24}$$

where $\boldsymbol{Y}_{(1)} \in \mathbb{R}_+^{I \times JK}$ and $\boldsymbol{G}_{(1)} \in \mathbb{R}_+^{R_1 \times R_2 R_3}$ are the first-way matricized form of tensor $\underline{\boldsymbol{Y}}$ and $\underline{\boldsymbol{G}}$, respectively, others are their second-way and third-way versions. If we put $\boldsymbol{\Xi} := \boldsymbol{G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T)$, it can be regarded as NMF $\boldsymbol{Y}_{(1)} \simeq \boldsymbol{A}\boldsymbol{\Xi}$. Thus, $\boldsymbol{A}$ can be updated by NMF based update rules. In similar way, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\underline{\boldsymbol{G}}$ can be also updated by NMF based update rules (e.g., by applying ALS).

The second model is the Canonical Polyadic (CP) model which is also called PARAFAC [43] or CANDECOMP [11]. The CP model is a special case of the Tucker model that $R_1 = R_2 = R_3 = R$ and $\underline{\boldsymbol{G}} = \underline{\boldsymbol{\Lambda}} \in \mathbb{R}_+^{R \times R \times R}$ is a diagonal tensor with entries g_r on the main diagonal as

$$\underline{\boldsymbol{Y}} \simeq \sum_{r=1}^{R} g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r. \tag{5.25}$$

The CP model can be regarded as a straightforward tensor extension of NMF since NMF can be rewritten by $\boldsymbol{Y} \simeq \sum_{r=1}^{R} g_r \boldsymbol{a}_r \circ \boldsymbol{b}_r$. Thus, the CP model gives an R-rank approximation of the observed tensor signal.

In this section, I consider how to extend the fastGRBF-NMF into tensor decompositions. For example, if we assume $\boldsymbol{a}_r$, $\boldsymbol{b}_r$, and/or $\boldsymbol{c}_r$ are smooth, then these are approximated by $\boldsymbol{\Phi}_a \boldsymbol{w}_r$, $\boldsymbol{\Phi}_b \boldsymbol{v}_r$, and/or $\boldsymbol{\Phi}_c \boldsymbol{h}_r$, respectively. The update rule for $\boldsymbol{w}_r$ can be applied to $\boldsymbol{v}_r$ and $\boldsymbol{h}_r$, hence I provide an update rule for only $\boldsymbol{w}_r$ in Sections 5.1.4 and 5.1.5.

Tucker model

In this section, I extend the fastGRBF-NMF into the nonnegative Tucker decomposition (NTD) to obtain $\boldsymbol{A}$ by the function approximation of $\boldsymbol{\Phi W}$. The criterion is given by

$$\begin{aligned} &\underset{\underline{\boldsymbol{G}},\boldsymbol{W},\boldsymbol{B},\boldsymbol{C}}{\text{minimize}} \ ||\underline{\boldsymbol{Y}} - \underline{\boldsymbol{G}} \times_1 \boldsymbol{\Phi W} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}||_F^2, \\ &\text{s.t. } \underline{\boldsymbol{G}} \geq 0, \ \boldsymbol{W} \geq 0, \ \boldsymbol{B} \geq 0, \ \boldsymbol{C} \geq 0. \end{aligned} \tag{5.26}$$

I call this method the fastGRBF-NTD. The key point of this problem is how to update $\boldsymbol{W}$. This is because the other factors $(\boldsymbol{B}, \boldsymbol{C}, \underline{\boldsymbol{G}})$ can be updated by the ALS (alternating least square) based algorithm for the nonnegative Tucker decomposition [84, 85, 83, 58]. Focusing on only $\boldsymbol{W}$ and using a matricized form, the problem can be rewritten as

$$\underset{\boldsymbol{W}}{\text{minimize}} \, ||\boldsymbol{Y}_{(1)} - \boldsymbol{\Phi W G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T)||_F^2, \text{ s.t. } \boldsymbol{W} \geq 0. \tag{5.27}$$

Putting $\boldsymbol{Z}_r := \boldsymbol{Y}_{(1)} - \sum_{k \neq r} \boldsymbol{\Phi} \boldsymbol{w}_k \boldsymbol{\xi}_k^T$ and $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_{R_1}]^T := \boldsymbol{G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T) \in \mathbb{R}^{R_1 \times JK}$, I consider the problem for $\boldsymbol{w}_r$ as

$$\underset{\boldsymbol{w}_r}{\text{minimize}} \ ||\boldsymbol{Z}_r - \boldsymbol{\Phi} \boldsymbol{w}_r \boldsymbol{\xi}_r^T||_F^2, \text{ s.t. } \boldsymbol{w}_r \geq 0. \tag{5.28}$$

(5.28) is essentially equivalent to (5.3). Since the condition of $||\boldsymbol{\xi}_r|| = 1$ must be satisfied, then an update rule for $\boldsymbol{w}_r$ is given by

$$\boldsymbol{\xi}_r \leftarrow [\boldsymbol{Z}_r^T \boldsymbol{\Phi} \boldsymbol{w}_r]_+; \tag{5.29}$$

$$\boldsymbol{\xi}_r \leftarrow \boldsymbol{\xi}_r / ||\boldsymbol{\xi}_r||; \tag{5.30}$$

$$\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast (\boldsymbol{\Phi}^T \boldsymbol{Z}_r \boldsymbol{\xi}_r) \oslash (\boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w}_r)]_+; \tag{5.31}$$

Note that the update rules (5.29) and (5.30) are important steps to update $\boldsymbol{w}_r$ accurately.

The algorithm for fastGRBF-NTD can be summarized as Algorithm 19. The 5-13th lines are update procedures for $\boldsymbol{W}$. If we apply the function approximation to factor matrices $\boldsymbol{B}$ and/or $\boldsymbol{C}$, the 14th and/or 15th lines must be modified in a similar way to $\boldsymbol{W}$.

5.1.5 CP model

In this section, I extend the fastGRBF-NMF into the nonnegative CP decomposition (NCPD) to obtain $\boldsymbol{A}$ by function approximation $\boldsymbol{\Phi W}$. The criterion is given by

$$\begin{aligned} \underset{g_r,\boldsymbol{w}_r,\boldsymbol{b}_r,\boldsymbol{c}_r}{\text{minimize}} \ & ||\underline{\boldsymbol{Y}} - \sum_{r=1}^{R} g_r(\boldsymbol{\Phi} \boldsymbol{w}_r) \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r||_F^2, \\ \text{s.t.} \ & g_r \geq 0, \ \boldsymbol{w}_r \geq 0, \ \boldsymbol{b}_r \geq 0, \ \boldsymbol{c}_r \geq 0, \\ & ||\boldsymbol{\Phi} \boldsymbol{w}_r|| = ||\boldsymbol{b}_r|| = ||\boldsymbol{c}_r|| = 1, \end{aligned} \tag{5.32}$$

Algorithm 19 FastGRBF-NTD algorithm

Input: $\underline{\boldsymbol{Y}}$, R_1, R_2, R_3, and $\boldsymbol{\Phi}$
Initialize: $\boldsymbol{W}$ by randomly, and $\boldsymbol{B}$, $\boldsymbol{C}$, $\underline{\boldsymbol{G}}$ by some initialization method for NTD
$\underline{\boldsymbol{E}} \leftarrow \underline{\boldsymbol{Y}} - \underline{\boldsymbol{G}} \times_1 \boldsymbol{\Phi}\boldsymbol{W} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$;
repeat
  $\boldsymbol{\Xi} \leftarrow \boldsymbol{G}_{(1)}(\boldsymbol{C}^T \otimes \boldsymbol{B}^T)$;
  for $r = 1, \ldots, R_1$ **do**
    $\boldsymbol{Z}_r \leftarrow \boldsymbol{E}_{(1)} + (\boldsymbol{\Phi}\boldsymbol{w}_r)\boldsymbol{\xi}_r^T$;
    $\boldsymbol{\xi}_r \leftarrow [\boldsymbol{Z}_r^T(\boldsymbol{\Phi}\boldsymbol{w}_r)]_+$;
    $\boldsymbol{\xi}_r \leftarrow \boldsymbol{\xi}_r / ||\boldsymbol{\xi}_r||$;
    $\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast \{\boldsymbol{\Phi}^T(\boldsymbol{Z}_r\boldsymbol{\xi}_r)\} \oslash \{\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{w}_r)\}]_+$;
    $\boldsymbol{E}_{(1)} \leftarrow \boldsymbol{Z}_r - (\boldsymbol{\Phi}\boldsymbol{w}_r)\boldsymbol{\xi}_r^T$;
  end for
  $\boldsymbol{A} \leftarrow \boldsymbol{\Phi}\boldsymbol{W}$;
  $\boldsymbol{B} \leftarrow \boldsymbol{B} \circledast (\boldsymbol{Y}_{(2)}(\boldsymbol{A} \otimes \boldsymbol{C})\boldsymbol{G}_{(2)}^T) \oslash (\boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{A}^T\boldsymbol{A} \otimes \boldsymbol{C}^T\boldsymbol{C})\boldsymbol{G}_{(2)}^T)$;
  $\boldsymbol{C} \leftarrow \boldsymbol{C} \circledast (\boldsymbol{Y}_{(3)}(\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{G}_{(3)}^T) \oslash (\boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{A}^T\boldsymbol{A} \otimes \boldsymbol{B}^T\boldsymbol{B})\boldsymbol{G}_{(3)}^T)$;
  $\underline{\boldsymbol{G}} \leftarrow \underline{\boldsymbol{G}} \circledast (\underline{\boldsymbol{Y}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T) \oslash (\underline{\boldsymbol{G}} \times_1 \boldsymbol{A}\boldsymbol{A}^T \times_2 \boldsymbol{B}\boldsymbol{B}^T \times_3 \boldsymbol{C}\boldsymbol{C}^T)$;
  $\underline{\boldsymbol{E}} \leftarrow \underline{\boldsymbol{Y}} - \underline{\boldsymbol{G}} \times_1 \boldsymbol{\Phi}\boldsymbol{W} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$;
until $||\boldsymbol{E}||_F^2$ converges
Output: $\boldsymbol{W}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\underline{\boldsymbol{G}}$

for $r = 1,\ 2,\ ...,\ R$. I call this method the fastGRBF-NCPD. We apply the HALS based algorithm to the problem (5.32). Putting $\underline{\boldsymbol{Y}}_r := \underline{\boldsymbol{Y}} - \sum_{k \neq r} g_r(\boldsymbol{\Phi}\boldsymbol{w}_r) \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$ and using the first-way matricized form, the sub-problem for $\boldsymbol{w}_r$ can be rewritten by

$$\begin{aligned} \underset{\boldsymbol{w}_r}{\text{minimize}}\ & ||\boldsymbol{Y}_{r(1)} - g_r(\boldsymbol{\Phi}\boldsymbol{w}_r)(\boldsymbol{c}_r^T \otimes \boldsymbol{b}_r^T)||_F^2, \\ \text{s.t.}\ & ||\boldsymbol{\Phi}\boldsymbol{w}_r|| = 1, \boldsymbol{w}_r \geq 0. \end{aligned} \tag{5.33}$$

In the similar relation between (5.3) and (5.8), problem (5.33) can be transformed to

$$\begin{aligned} \underset{\boldsymbol{w}_r}{\text{minimize}}\ & ||\boldsymbol{Y}_{r(1)}(\boldsymbol{c}_r \otimes \boldsymbol{b}_r) - g_r\boldsymbol{\Phi}\boldsymbol{w}_r||_F^2, \\ \text{s.t.}\ & ||\boldsymbol{\Phi}\boldsymbol{w}_r|| = 1, \boldsymbol{w}_r \geq 0, \end{aligned} \tag{5.34}$$

where we have $(\boldsymbol{c}_r^T \otimes \boldsymbol{b}_r^T)(\boldsymbol{c}_r \otimes \boldsymbol{b}_r) = (\boldsymbol{c}_r^T\boldsymbol{c}_r \otimes \boldsymbol{b}_r^T\boldsymbol{b}_r) = 1$. Since there is a constraint $||\boldsymbol{\Phi}\boldsymbol{w}_r|| = 1$ in (5.34), its solution is not dependent on g_r. Thus, the update rule for $\boldsymbol{w}_r$ is given by

$$\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast \{\boldsymbol{\Phi}^T\boldsymbol{Y}_{r(1)}(\boldsymbol{c}_r \otimes \boldsymbol{b}_r)\} \oslash \{\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}_r\}]_+, \tag{5.35}$$

$$\boldsymbol{w}_r \leftarrow \boldsymbol{w}_r / ||\boldsymbol{\Phi}\boldsymbol{w}_r||. \tag{5.36}$$

Algorithm 20 FastGRBF-NCPD algorithm

1: **Input:** $\underline{\boldsymbol{Y}}$, R, and $\boldsymbol{\Phi}$
2: **Initialize:** $\boldsymbol{W}$ by randomly, and $\boldsymbol{B}$, $\boldsymbol{C}$, $\underline{\boldsymbol{G}}$ by some initialization method for nonnegative CP decomposition
3: $\underline{\boldsymbol{E}} \leftarrow \underline{\boldsymbol{Y}} - \underline{\boldsymbol{G}} \times_1 \boldsymbol{\Phi}\boldsymbol{W} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$;
4: **repeat**
5: **for** $r = 1, \ldots, R$ **do**
6: $\underline{\boldsymbol{Y}}_r \leftarrow \underline{\boldsymbol{E}} + g_r(\boldsymbol{\Phi}\boldsymbol{w}_r) \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$;
7: $\boldsymbol{w}_r \leftarrow [\boldsymbol{w}_r \circledast \{\boldsymbol{\Phi}^T(\underline{\boldsymbol{Y}}_r \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T)\} \oslash \{\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}_r\}]_+$;
8: $\boldsymbol{w}_r \leftarrow \boldsymbol{w}_r / ||\boldsymbol{\Phi}\boldsymbol{w}_r||$;
9: $\boldsymbol{b}_r \leftarrow [\underline{\boldsymbol{Y}}_r \times_1 (\boldsymbol{\Phi}\boldsymbol{w}_r)^T \times_3 \boldsymbol{c}_r^T]_+$;
10: $\boldsymbol{b}_r \leftarrow \boldsymbol{b}_r / ||\boldsymbol{b}_r||$;
11: $\boldsymbol{c}_r \leftarrow [\underline{\boldsymbol{Y}}_r \times_1 (\boldsymbol{\Phi}\boldsymbol{w}_r)^T \times_2 \boldsymbol{b}_r^T]_+$;
12: $\boldsymbol{c}_r \leftarrow \boldsymbol{c}_r / ||\boldsymbol{c}_r||$;
13: $g_r \leftarrow [\underline{\boldsymbol{Y}}_r \times_1 (\boldsymbol{\Phi}\boldsymbol{w}_r)^T \times_2 \boldsymbol{b}_r^T \times_3 \boldsymbol{c}_r^T]_+$;
14: $\underline{\boldsymbol{E}} \leftarrow \underline{\boldsymbol{Y}}_r - g_r(\boldsymbol{\Phi}\boldsymbol{w}_r) \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$;
15: **end for**
16: **until** $||\boldsymbol{E}||_F^2$ converge
17: **Output:** $\boldsymbol{W}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\underline{\boldsymbol{G}}$

The fastGRBF-NCPD method can be summarized as Algorithm 20. The 7-8th lines are update procedures for $\boldsymbol{w}_r$. If we apply the function approximation to also $\boldsymbol{b}_r$ and/or $\boldsymbol{c}_r$, the 9-10th and/or 11-12th lines must be modified in a similar way to $\boldsymbol{w}_r$.

5.2 Linked CP Tensor Decomposition

In Section 3.5, I introduced the matrix based common and individual feature extraction methods; however, their methods can not be applied to multiway-array data. In this section, I extend this common and individual feature extraction model to the CP tensor decomposition model, called the linked CP tensor decomposition (LCPTD) [107]. Furthermore, I impose the sparsity and nonnegativity constraints to this model.

LCPTD model is given by

$$\underline{\boldsymbol{Z}}^{(s)} \approx \widehat{\underline{\boldsymbol{Z}}}^{(s)} = [\![\underline{\boldsymbol{G}}^{(s)}; \boldsymbol{U}^{(1,s)}, \ldots, \boldsymbol{U}^{(N,s)}]\!] = \sum_{j=1}^{J} g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \ldots \circ \boldsymbol{u}_j^{(N,s)}, \tag{5.37}$$

where each factor matrix $\boldsymbol{U}^{(n,s)} = [\boldsymbol{U}_C^{(n)}, \boldsymbol{U}_I^{(n,s)}] \in \mathbb{R}^{I_n \times J}$ is composed of two set of bases: $\boldsymbol{U}_C^{(n)} \in \mathbb{R}^{I_n \times L_n}$ (with $0 \leq L_n \leq J$), which is a common factor matrix for all blocks and corresponds to the same or maximally correlated components, and $\boldsymbol{U}_I^{(n,s)} \in \mathbb{R}^{I_n \times J - L_n}$, which corresponds to different

individual characteristics.

The LCPTD can be considered as a generalized model of the simultaneous decomposition. When we let $L_n = J$, its decomposition is equivalent to the simultaneous common factor decomposition [82]. On the other hand, when $L_n = 0$, its decomposition of each subject is equivalent to the standard tensor decomposition. Then the LTD is an intermediate decomposition between the simultaneous and the ordinary tensor decompositions.

5.2.1 LCPTD-HALS algorithm

In this section, I introduce a new HALS algorithm for LCPTD. Optimization criterion for LCPTD is given by

$$\text{minimize} \quad \sum_{s=1}^{S} \left|\left| \underline{\boldsymbol{Z}}^{(s)} - \sum_{j=1}^{J} g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \ldots \circ \boldsymbol{u}_j^{(N,s)} \right|\right|_F^2, \tag{5.38}$$

$$\text{subject to} \quad \boldsymbol{u}_j^{(n,1)} = \cdots = \boldsymbol{u}_j^{(n,S)} \text{ for } j \leq L_n,\ ||\boldsymbol{u}_j^{(n,s)}|| = 1, \tag{5.39}$$

for all n, s, and j. Furthermore, I add $||\boldsymbol{u}_j^{(n,s)}||_1 < v$ or $u_{ji}^{(n,s)} \geq 0,\ g_j^{(s)} \geq 0\ \ \forall i, j, n, s$ into (5.39) if we want to obtain sparse or nonnegative components.

The hierarchical ALS (HALS) algorithm was first proposed for the NMF and NTF in [24]. The HALS algorithm was applied to the CP model and it achieved a high performance in [19]. In this algorithm, I consider J local-problems and solve them sequentially and iteratively instead of solving (5.38) and (5.39), directly. Let $\underline{\boldsymbol{Y}}_j^{(s)} := \underline{\boldsymbol{Z}}^{(s)} - \sum_{i \neq j} g_i^{(s)} \boldsymbol{u}_i^{(1,s)} \circ \ldots \circ \boldsymbol{u}_i^{(N,s)}$, the j-th local problem is given by

$$\text{minimize} \quad \sum_{s=1}^{S} ||\underline{\boldsymbol{Y}}_j^{(s)} - g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \ldots \circ \boldsymbol{u}_j^{(N,s)}||_F^2, \tag{5.40}$$

$$\text{subject to} \quad \boldsymbol{u}_j^{(n,1)} = \cdots = \boldsymbol{u}_j^{(n,S)} \text{ if } j \leq L_n,\ ||\boldsymbol{u}_j^{(n,s)}|| = 1, \tag{5.41}$$

for all n and s. The LTD-HALS algorithm can be summarized as Algorithm 21; note that it does not require matrix inversion and is solved by only simple calculations.

If we want to obtain sparse components, I implement the following updates after (5.42) in Algorithm 21:

$$\boldsymbol{u}_j^{(n,s)} \leftarrow \text{sign}(\boldsymbol{u}_j^{(n,s)}) \circledast [\text{abs}(\boldsymbol{u}_j^{(n,s)}) - \xi_n \mathbf{1}]_+ \text{ for all } s; \tag{5.44}$$

where ξ_n is a positive parameter deciding on their sparsity.

If we want to obtain nonnegative components, I implement the following updates after (5.42) and (5.43) in Algorithm 21:

$$\boldsymbol{u}_j^{(n,s)} \leftarrow [\boldsymbol{u}_j^{(n,s)}]_+ \text{ for all } s, \tag{5.45}$$

$$g_j^{(s)} \leftarrow [g_j^{(s)}]_+ \text{ for all } s. \tag{5.46}$$

Algorithm 21 LTD-HALS algorithm

Input: $\{\underline{\boldsymbol{Z}}^{(s)}\}_{s=1}^{S}$, J, and $\{L_n\}_{n=1}^{N}$
Initialize: $\{\boldsymbol{g}^{(s)},\ \{\boldsymbol{U}^{(n,s)}\}_{n=1}^{N}\}_{s=1}^{S}$.
$\underline{\boldsymbol{E}}^{(s)} = \underline{\boldsymbol{Z}}^{(s)} - \Sigma_{j=1}^{J} g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \cdots \circ \boldsymbol{u}_j^{(N,s)}$ for all s;
repeat
  for $j = 1, \ldots, J$ **do**
    $\underline{\boldsymbol{Y}}_j^{(s)} = \underline{\boldsymbol{E}}^{(s)} + g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \cdots \circ \boldsymbol{u}_j^{(N,s)}$ for all s;
    for $n = 1, \ldots, N$ **do**
      Updating $\boldsymbol{u}_j^{(n,s)}$:

$$\boldsymbol{u}_j^{(n,s)} \leftarrow g_j^{(s)} \underline{\boldsymbol{Y}}_j^{(s)} \times_1 \boldsymbol{u}_j^{(1,s)} \cdots \times_{n-1} \boldsymbol{u}_j^{(n-1,s)} \times_{n+1} \boldsymbol{u}_j^{(n+1,s)} \cdots \times_N \boldsymbol{u}_j^{(N,s)} \text{ for all } s; \tag{5.42}$$

      if $j \leq L_n$, $\boldsymbol{t} \leftarrow \sum_{s=1}^{S} \boldsymbol{u}_j^{(n,s)}$; $\boldsymbol{u}_j^{(n,s)} \leftarrow \boldsymbol{t}$ for all s; **end if**
      Normalize $\boldsymbol{u}_j^{(n,s)} \leftarrow \boldsymbol{u}_j^{(n,s)} / ||\boldsymbol{u}_j^{(n,s)}||$ for all s;
    end for
    Update $g_j^{(s)}$:

$$g_j^{(s)} \leftarrow \underline{\boldsymbol{Y}}_j^{(s)} \times_1 \boldsymbol{u}_j^{(1,s)} \cdots \times_N \boldsymbol{u}_j^{(N,s)} \text{ for all } s; \tag{5.43}$$

    $\underline{\boldsymbol{E}}^{(s)} = \underline{\boldsymbol{Y}}_j^{(s)} - g_j^{(s)} \boldsymbol{u}_j^{(1,s)} \circ \cdots \circ \boldsymbol{u}_j^{(N,s)}$ for all s;
  end for
until $\sum_{s=1}^{S} ||\underline{\boldsymbol{E}}^{(s)}||_F^2$ converge
Output: $\{\boldsymbol{g}^{(s)},\ \{\boldsymbol{U}^{(n,s)}\}_{n=1}^{N}\}_{s=1}^{S}$

5.3 Linked Tucker Decomposition

In this section, I extend the LCPTD model to the Tucker model, called the linked Tucker decomposition (LTD). Since Tucker model is very general and does not have a unique solution, non-constraint is not suitable. I consider orthogonality, sparsity, and nonnegativity constraints for LTD model.

The LTD model is given by

$$\begin{aligned} \underline{\boldsymbol{Z}}^{(s)} \approx \widehat{\underline{\boldsymbol{Z}}}^{(s)} &= \underline{\boldsymbol{G}}^{(s)} \times \{\boldsymbol{U}^{(s)}\} \\ &= \underline{\boldsymbol{G}}^{(s)} \times_1 \boldsymbol{U}^{(1,s)} \cdots \times_N \boldsymbol{U}^{(N,s)}, \end{aligned} \tag{5.47}$$

where each factor matrix $\boldsymbol{U}^{(n,s)} = [\boldsymbol{U}_C^{(n)}, \boldsymbol{U}_I^{(n,s)}] \in \mathbb{R}^{I_n \times J_n}$ is composed of two set of bases: $\boldsymbol{U}_C^{(n)} \in \mathbb{R}^{I_n \times L_n}$ (with $0 \leq L_n \leq J_n$), which is common factor matrices for all blocks and corresponds to the same or maximally correlated components, and $\boldsymbol{U}_I^{(n,s)} \in \mathbb{R}^{I_n \times R_n}$ (i.e. $L_n + R_n = J_n$), which corresponds to different individual characteristics. Fig. 5.2 depicts a three-modes case of LTD model. This model was first briefly introduced as an idea for the linked multiway blind source separation in

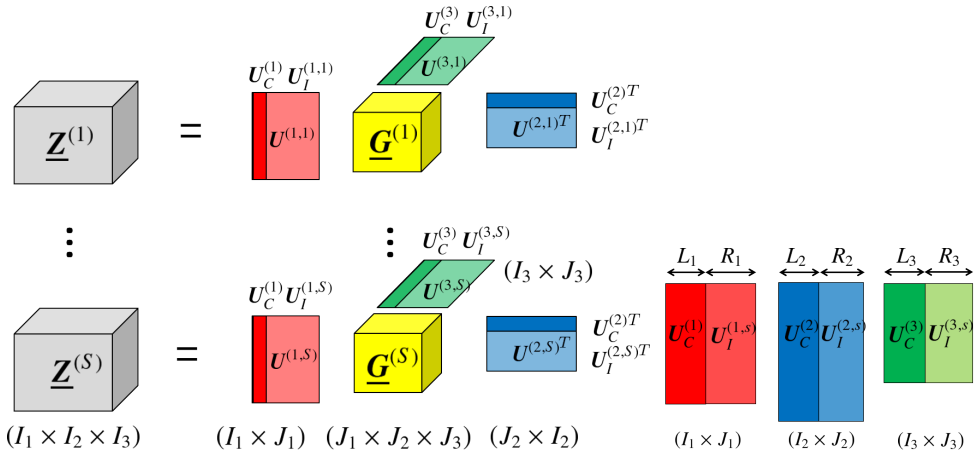


Figure 5.2: Linked Tucker Decomposition (LTD): in special case, we obtain STD for $L_n = J_n$ and ITD for $L_n = 0$.

[17], but its algorithms and applications have not been proposed yet. In this paper, I propose a new approach and its application.

The LTD can be considered as a generalized model of the STD. When we put $L_n = J_n$, its decomposition is equivalent to the simultaneous common factor decomposition (i.e. the STD model) [82]. On the other hand, when $L_n = 0$, its decomposition of each subject is equivalent to the standard tensor decomposition (i.e. the ITD model). Then the LTD is an intermediate decomposition between the simultaneous and the individual tensor/Tucker decompositions.

5.3.1 Orthogonal LTD method

In this section, I propose a novel method called the "linked multi-way principal component analysis" (LMPCA) and associated algorithm called the LTD-ALS algorithm. The LMPCA can be considered as a prototype of all linked group tensor analysis methods, just as PCA is a prototype of single block data analysis methods. We could derive many extensions for examples the linked multiway sparse component analysis (LMSCA), the linked nonnegative Tucker decomposition (LNTD), the linked multiway independent component analysis (LMICA) and so on. The ALS algorithm was first proposed for the three-way PCA in [60]. The LTD-ALS can be derived in a similar way. Optimization criterion for the LMPCA is given by

$$\begin{aligned} \text{minimize} \quad & C(\Theta) := \sum_{s=1}^{S} ||\underline{\boldsymbol{Z}}^{(s)} - \underline{\boldsymbol{G}}^{(s)} \times \{\boldsymbol{U}^{(s)}\}||_F^2, \\ \text{subject to} \quad & \boldsymbol{U}^{(n,s)T}\boldsymbol{U}^{(n,s)} = \boldsymbol{I}_{J_n} \text{ for all } n \text{ and } s, \end{aligned} \tag{5.48}$$

where $C(\Theta)$ is the cost function and Θ represents a set of parameters (i.e., $\{\{\boldsymbol{U}^{(n,s)}\}_{n=1}^{N}, \underline{\boldsymbol{G}}^{(s)}\}_{s=1}^{S}$). Substituting one of the KKT conditions $\underline{\boldsymbol{G}}^{(s)} = \underline{\boldsymbol{Z}}^{(s)} \times \{\boldsymbol{U}^{(s)T}\}$ into (5.48) and transforming this, then

Algorithm 22 Local algorithm to update n-mode factor matrices for orthogonal LTD

Input: $\{\boldsymbol{Y}_n^{(s)}, \boldsymbol{U}_I^{(n,s)}\}_{s=1}^S$, $\boldsymbol{U}_C^{(n)}$, L_n, and R_n
repeat
  $\boldsymbol{Y}_n \leftarrow [\boldsymbol{Y}_n^{(1)} - \boldsymbol{U}_I^{(n,1)}\boldsymbol{U}_I^{(n,1)T}\boldsymbol{Y}_n^{(1)}, \boldsymbol{Y}_n^{(2)} - \boldsymbol{U}_I^{(n,2)}\boldsymbol{U}_I^{(n,2)T}\boldsymbol{Y}_n^{(2)},$
  $\quad \ldots, \boldsymbol{Y}_n^{(S)} - \boldsymbol{U}_I^{(n,S)}\boldsymbol{U}_I^{(n,S)T}\boldsymbol{Y}_n^{(S)}]$;
  $\boldsymbol{U}_C^{(n)} \leftarrow$ left singular matrix of tSVD$(\boldsymbol{Y}_n, L_n)$;
  for $s = 1, \ldots, S$ **do**
    $\boldsymbol{X} \leftarrow \boldsymbol{Y}_n^{(s)} - \boldsymbol{U}_C^{(n)}\boldsymbol{U}_C^{(n)T}\boldsymbol{Y}_n^{(s)}$;
    $\boldsymbol{U}_I^{(n,s)} \leftarrow$ left singular matrix of tSVD$(\boldsymbol{X}, R_n)$;
  end for
until $C(\Theta)$ converge
Output: $U_C^{(n)}, \{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^S$

we can obtain a simplified optimization problem with respect to n-mode factor matrices as

$$\begin{aligned} \text{minimize} \quad & \sum_{s=1}^{S} ||\boldsymbol{Y}_n^{(s)} - \boldsymbol{U}^{(n,s)}\boldsymbol{U}^{(n,s)T}\boldsymbol{Y}_n^{(s)}||_F^2, \\ \text{subject to} \quad & \boldsymbol{U}^{(n,s)T}\boldsymbol{U}^{(n,s)} = \boldsymbol{I}_{J_n} \text{ for all } s, \end{aligned} \tag{5.49}$$

where $\boldsymbol{Y}_n^{(s)} := [\underline{\boldsymbol{Z}}^{(s)} \times_{-n} \{\boldsymbol{U}^{(s)T}\}]_{(n)} \in \mathbb{R}^{I_n \times \prod_{i \neq n} J_i}$. Problem (5.49) can be considered as the PCA problem with respect to $\boldsymbol{U}^{(n,s)}$. Please note that the solutions of PCA and the left factor matrix of truncated SVD (tSVD) are the same. Thus I consider to factorize $\boldsymbol{Y}_n^{(s)}$ by the common and the individual left factor matrices in a similar way to JIVE algorithm. To update $\boldsymbol{U}_C^{(n)}$, I run the tSVD as

$$\begin{aligned} [\boldsymbol{Y}_n^{(1)} - \boldsymbol{U}_I^{(n,1)}\boldsymbol{U}_I^{(n,1)T}\boldsymbol{Y}_n^{(1)}, \boldsymbol{Y}_n^{(2)} - \boldsymbol{U}_I^{(n,2)}\boldsymbol{U}_I^{(n,2)T}\boldsymbol{Y}_n^{(2)}, \\ \ldots, \boldsymbol{Y}_n^{(S)} - \boldsymbol{U}_I^{(n,S)}\boldsymbol{U}_I^{(n,S)T}\boldsymbol{Y}_n^{(S)}] \simeq \boldsymbol{U}_C\boldsymbol{D}_C\boldsymbol{V}_C^T. \end{aligned} \tag{5.50}$$

To update $\boldsymbol{U}_I^{(n,s)}$, I run the tSVD one by one, individually, which is given by

$$[\boldsymbol{Y}_n^{(s)} - \boldsymbol{U}_C\boldsymbol{U}_C^T\boldsymbol{Y}_n^{(s)}] \simeq \boldsymbol{U}_I^{(n,s)}\boldsymbol{D}_I^{(s)}\boldsymbol{V}_I^{(s)T}. \tag{5.51}$$

In order to obtain optimal parameters, I iterate the common and the individual steps alternately until convergence. Thus, the algorithm for local problem (5.49) can be summarized as Algorithm 22.

In the orthogonal LTD algorithm, I solve the local problems (5.49), sequentially and iteratively. To solve (5.49) I consider two steps: the first and the second steps are for common and individual factors, respectively. The orthogonal LTD algorithm can be summarized as Algorithm 23.

5.3.2 Sparse LTD

In this section, I consider to impose the sparse constraint to each factor matrices $\boldsymbol{U}^{(n,s)}$. Sparsity is a very important property for some special sparse data analysis such as text-mining data. The one of

Algorithm 23 Orthogonal LTD algorithm

Input: $\{\underline{\boldsymbol{Z}}^{(s)}\}_{s=1}^{S}$ and $\{L_n, R_n\}_{n=1}^{N}$
Initialize: orthonormal matrices $\{\boldsymbol{U}_C^{(n)},\ \{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^{S}\}_{n=1}^{N}$
repeat
  for $n = 1, \ldots, N$ **do**
    $\boldsymbol{Y}_n^{(s)} \leftarrow [\underline{\boldsymbol{Z}}^{(s)} \times_{-n} \{\boldsymbol{U}^{(s)T}\}]_{(n)}$ for all s;
    Update $\boldsymbol{U}_C^{(n)}$ and $\{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^{S}$ by Algorithm 22;
  end for
  $\underline{\boldsymbol{G}}^{(s)} = \underline{\boldsymbol{Z}}^{(s)} \times \{\boldsymbol{U}^{(s)T}\}$ for all s;
until $C(\Theta)$ converge
Output: $\{\underline{\boldsymbol{G}}^{(s)},\ \{\boldsymbol{U}^{(n,s)}\}_{n=1}^{N}\}_{s=1}^{S}$

most popular approaches to give a sparse representation is the l1-norm regularization. For example, there are the sparse PCA [121] and the penalized matrix decomposition (PMD) [105] as the methods of sparse analysis. Since the PMD can be regarded as a sparse extension of tSVD, we can apply the PMD method to LTD. The local problem with respect to n-mode factor matrices can be given by

$$\begin{aligned} \text{minimize} \quad & \sum_{s=1}^{S} ||\boldsymbol{Y}_n^{(s)} - \boldsymbol{U}^{(n,s)}\boldsymbol{U}^{(n,s)\dagger}\boldsymbol{Y}_n^{(s)}||_F^2, \\ \text{subject to} \quad & ||\boldsymbol{u}_{j_n}^{(n,s)}||_1 \leq c_n \text{ for all } s, \end{aligned} \tag{5.52}$$

where † describes the Moore-Penrose's pseudo inverse (i.e., $\boldsymbol{U}^{(n,s)\dagger} = (\boldsymbol{U}^{(n,s)T}\boldsymbol{U}^{(n,s)})^{-1}\boldsymbol{U}^{(n,s)T}$), $||\cdot||_1$ is the l1-norm, and c_n is a trade-off parameter for l1-norm regularization. I set this parameter in $1 < c_n < \sqrt{J_n}$. This solution can be obtained by a similar way to Algorithm 22. The differences are that the transposition ^{T} becomes the pseudo inverse † in the matrix deflation formula, and tSVD becomes PMD for update. Thus the local and global algorithm for sparse LTD can be summarized as Algorithms 24 and 25.

5.3.3 Nonnegative LTD

In this section, I impose the nonnegative constraint to the LTD method. I call this method the nonnegative linked Tucker decomposition (NLTD). In order to give an algorithm for NLTD, techniques of nonnegative Tucker decomposition [58, 83, 84, 85] and group NMF [63] can be used. In the same manner as that the algorithm for NMF is quite different from the algorithm for tSVD, the algorithm for NLTD is also different from the OLTD algorithm. Optimization criterion is given by

$$\begin{aligned} \text{minimize} \quad & C(\Theta) := \sum_{s=1}^{S} ||\underline{\boldsymbol{Z}}^{(s)} - \underline{\boldsymbol{G}}^{(s)} \times \{\boldsymbol{U}^{(s)}\}||_F^2, \\ \text{subject to} \quad & \boldsymbol{U}^{(n,s)} \geq 0,\ \underline{\boldsymbol{G}}^{(s)} \geq 0 \text{ for all } n \text{ and } s. \end{aligned} \tag{5.53}$$

Algorithm 24 Local algorithm to update n-mode factor matrices for sparse LTD

Input: $\{\boldsymbol{Y}_n^{(s)}, \boldsymbol{U}_I^{(n,s)}\}_{s=1}^S$, $\boldsymbol{U}_C^{(n)}$, L_n, R_n, and c_n
repeat
  $\boldsymbol{Y}_n \leftarrow [\boldsymbol{Y}_n^{(1)} - \boldsymbol{U}_I^{(n,1)}\boldsymbol{U}_I^{(n,1)\dagger}\boldsymbol{Y}_n^{(1)}, \boldsymbol{Y}_n^{(2)} - \boldsymbol{U}_I^{(n,2)}\boldsymbol{U}_I^{(n,2)\dagger}\boldsymbol{Y}_n^{(2)},$
  $\quad \ldots, \boldsymbol{Y}_n^{(S)} - \boldsymbol{U}_I^{(n,S)}\boldsymbol{U}_I^{(n,S)\dagger}\boldsymbol{Y}_n^{(S)}]$;
  $\boldsymbol{U}_C^{(n)} \leftarrow$ left factor matrix of PMD$(\boldsymbol{Y}_n, L_n, c_n)$;
  for $s = 1, \ldots, S$ **do**
    $\boldsymbol{X} \leftarrow \boldsymbol{Y}_n^{(s)} - \boldsymbol{U}_C^{(n)}\boldsymbol{U}_C^{(n)\dagger}\boldsymbol{Y}_n^{(s)}$;
    $\boldsymbol{U}_I^{(n,s)} \leftarrow$ left factor matrix of PMD$(\boldsymbol{X}, R_n, c_n)$;
  end for
until $C(\Theta)$ converge
Output: $U_C^{(n)}, \{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^S$

Algorithm 25 Sparse LTD algorithm

Input: $\{\underline{\boldsymbol{Z}}^{(s)}\}_{s=1}^S$ and $\{L_n, R_n, c_n\}_{n=1}^N$
Initialize: orthonormal matrices $\{\boldsymbol{U}_C^{(n)}, \{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^S\}_{n=1}^N$
repeat
  for $n = 1, \ldots, N$ **do**
    $\boldsymbol{Y}_n^{(s)} \leftarrow [\underline{\boldsymbol{Z}}^{(s)} \times_{-n} \{\boldsymbol{U}^{(s)T}\}]_{(n)}$ for all s;
    Update $\boldsymbol{U}_C^{(n)}$ and $\{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^S$ by Algorithm 24;
  end for
  $\underline{\boldsymbol{G}}^{(s)} = \underline{\boldsymbol{Z}}^{(s)} \times \{\boldsymbol{U}^{(s)\dagger}\}$ for all s;
until $C(\Theta)$ converge
Output: $\{\underline{\boldsymbol{G}}^{(s)}, \{\boldsymbol{U}^{(n,s)}\}_{n=1}^N\}_{s=1}^S$

Please note that nonnegative constraints are imposed to not only factor matrices $\boldsymbol{U}^{(n,s)}$ but also core tensors $\underline{\boldsymbol{G}}^{(s)}$. Thus, we can not obtain $\underline{\boldsymbol{G}}^{(s)}$, analytically, unlike OLTD and SLTD. From this reason, the algorithm for NLTD becomes quite complex. Let us use the following notations

$$\boldsymbol{U}_{(-n)}^{(s)} := \boldsymbol{U}^{(N,s)} \otimes \cdots \otimes \boldsymbol{U}^{(n+1,s)} \otimes \boldsymbol{U}^{(n-1,s)} \otimes \cdots \otimes \boldsymbol{U}^{(1,s)}, \tag{5.54}$$

$$\boldsymbol{Y}_{(n)}^{(s)} := \boldsymbol{G}_{(n)}^{(s)}\boldsymbol{U}_{(-n)}^{(s)T}, \tag{5.55}$$

$$\boldsymbol{Y}_{C(n)}^{(s)} := \text{first } L_n \text{ row vectors of } \boldsymbol{Y}_{(n)}^{(s)}, \tag{5.56}$$

$$\boldsymbol{Y}_{I(n)}^{(s)} := \text{last } R_n \text{ row vectors of } \boldsymbol{Y}_{(n)}^{(s)}, \tag{5.57}$$

where $\otimes$ describes Kronecker product, then the cost function can be rewritten as

$$\begin{aligned} C(\Theta) &= \sum_{s=1}^{S} ||\boldsymbol{Z}_{(n)}^{(s)} - \boldsymbol{U}^{(n,s)} \boldsymbol{G}_{(n)}^{(s)} \boldsymbol{U}_{(-n)}^{(s)T}||_F^2 \\ &= \sum_{s=1}^{S} ||\boldsymbol{Z}_{(n)}^{(s)} - \boldsymbol{U}^{(n,s)} \boldsymbol{Y}_{(n)}^{(s)}||_F^2 \\ &= \sum_{s=1}^{S} ||\boldsymbol{Z}_{(n)}^{(s)} - \boldsymbol{U}_C^{(n)} \boldsymbol{Y}_{C(n)}^{(s)} - \boldsymbol{U}_I^{(n,s)} \boldsymbol{Y}_{I(n)}^{(s)}||_F^2. \end{aligned} \tag{5.58}$$

Therefore, the update rules of common and individual factor matrices are given by

$$\boldsymbol{U}_C^{(n)} \leftarrow \boldsymbol{U}_C^{(n)} \circledast \left[\sum_{s=1}^{S} \boldsymbol{Z}_{(n)}^{(s)} \boldsymbol{Y}_{C(n)}^{(s)T} \right] \oslash \left[\sum_{s=1}^{S} \boldsymbol{U}^{(n,s)} \boldsymbol{Y}_{(n)}^{(s)} \boldsymbol{Y}_{C(n)}^{(s)T} \right], \tag{5.59}$$

$$\boldsymbol{U}_I^{(n,s)} \leftarrow \boldsymbol{U}_I^{(n,s)} \circledast \left[\boldsymbol{Z}_{(n)}^{(s)} \boldsymbol{Y}_{I(n)}^{(s)T} \right] \oslash \left[\boldsymbol{U}^{(n,s)} \boldsymbol{Y}_{(n)}^{(s)} \boldsymbol{Y}_{I(n)}^{(s)T} \right] \tag{5.60}$$

for all s, where $\circledast$ and $\oslash$ are element-wise multiplication and division, respectively. Finally, a technique in NTD [58] can be applied to the update rule of core tensors $\underline{\boldsymbol{G}}^{(s)}$ as follow:

$$\underline{\boldsymbol{G}}^{(s)} \leftarrow \underline{\boldsymbol{G}}^{(s)} \circledast \underline{\boldsymbol{Z}}^{(s)} \times \{\boldsymbol{U}^{(s)T}\} \oslash \underline{\boldsymbol{G}}^{(s)} \times \{\boldsymbol{U}^{(s)T} \boldsymbol{U}^{(s)}\} \tag{5.61}$$

for all s. Furthermore, we have to be careful for the rotations of updates. In this algorithm, I separate the updates of all common factor matrices and the updates of all individual factor matrices, completely. In the common step, I update all common factor matrices and individual core tensors while individual factor matrices are fixed. In the individual step, I update all individual factor matrices and individual core tensors while common factor matrices are fixed. I iterate both steps until convergence.

Thus, the algorithms for nonnegative LTD can be summarized as Algorithm 26. In this algorithm, I iterate K times to update by each step. When K is very small, convergence becomes very slow; however, when K is too large, calculation time of one cycle becomes very long. For example, I set $K = 50$ or 100.

Algorithm 26 Nonnegative LTD algorithm

Input: $\{\underline{\boldsymbol{Z}}^{(s)}\}_{s=1}^{S}$, $\{L_n, R_n\}_{n=1}^{N}$, and K
Initialize: nonnegative matrices $\{\boldsymbol{U}_C^{(n)},\ \{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^{S}\}_{n=1}^{N}$ and core tensors $\{\underline{\boldsymbol{G}}^{(s)}\}_{s=1}^{S}$
repeat
  % common step
  for $k = 1, \ldots, K$ **do**
    for $n = 1, \ldots, N$ **do**
      Update $\boldsymbol{U}_C^{(n)}$ by calculation (5.59) ;
    end for
    for $s = 1, \ldots, S$ **do**
      Update $\underline{\boldsymbol{G}}^{(s)}$ by calculation (5.61);
    end for
  end for
  % individual step
  for $s = 1, \ldots, S$ **do**
    for $k = 1, \ldots, K$ **do**
      for $n = 1, \ldots, N$ **do**
        Update $\{\boldsymbol{U}_I^{(n,s)}\}_{s=1}^{S}$ by calculation (5.60) ;
      end for
      Update $\underline{\boldsymbol{G}}^{(s)}$ by calculation (5.61);
    end for
  end for
until $C(\Theta)$ converge
Output: $\{\underline{\boldsymbol{G}}^{(s)},\ \{\boldsymbol{U}^{(n,s)}\}_{n=1}^{N}\}_{s=1}^{S}$

Chapter 6

Proposed Classifiers

In this Chapter, I propose new three classification methods: the QCMAP estimation in Section 6.1, SVM with weighted regularization in Section 6.2, and the Chernoff FDA in Section 6.3.

6.1 QCMAP

6.1.1 QCMAP estimation

Let $Q(\boldsymbol{x})$ be a weight function that satisfies

$$\int_{\mathcal{D}} Q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1, \quad Q(\boldsymbol{z}) > 0 \text{ for all } \boldsymbol{z} \in \mathcal{D}, \tag{6.1}$$

where $\mathcal{D}$ is the data domain (e.g., $\mathbb{R}^d$).

The *quadratically constrained MAP* (QCMAP) classifier [113, 109, 111] is then defined as follows:

$$\text{maximize} \sum_{y\in\{+1,-1\}} \int_{\mathcal{D}} P(\boldsymbol{x}, y) \min(yW'(\boldsymbol{x}), 1)\mathrm{d}\boldsymbol{x}, \tag{6.2}$$

$$\text{subject to} \quad \int_{\mathcal{D}} Q(\boldsymbol{x})|W'(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x} \leq 1. \tag{6.3}$$

Then the solution of the QCMAP problem is given by

$$W'(\boldsymbol{x}) \begin{cases} \geq 1 & \text{if } P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x}) = 0 \\ = 1 & \text{if } P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x}) \neq 0 \\ = \zeta(x) & \text{if } P(+1|\boldsymbol{x}) = P(-1|\boldsymbol{x}) \\ = -1 & \text{if } P(-1|\boldsymbol{x}) > P(+1|\boldsymbol{x}) \neq 0 \\ \leq -1 & \text{if } P(-1|\boldsymbol{x}) > P(+1|\boldsymbol{x}) = 0 \end{cases}, \tag{6.4}$$

where $\zeta(x)$ is an arbitrary number ($-1 \leq \zeta(x) \leq 1$). In addition, if the measure of $\{x|P(+1|x) = P(-1|x)\}$ is zero (a condition that is satisfied in almost all classification problems), then $W'(\boldsymbol{x})$ is given by

$$W'(\boldsymbol{x}) = \begin{cases} 1 & \text{if } P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x}) \\ -1 & \text{if } P(+1|\boldsymbol{x}) < P(-1|\boldsymbol{x}) \end{cases}. \tag{6.5}$$

In both cases, $W'(\boldsymbol{x})$ yields the same results as the MAP classifier.

Furthermore, we can interpret that the objective and constraint functions (6.2) and (6.3) play fitting and regularization roles, respectively. Classification criteria typically have a tradeoff parameter between fitting and regularization functions; however our criterion does not need such a tradeoff parameter. This lack of tradeoff is an interesting and important point in using our classifier.

6.1.2 Theoretical Interpretation of QCMAP estimation

As previously stated, the QCMAP estimation yields the same results as the MAP classifier. In this section, I discuss a theoretical interpretation of the QCMAP classifier.

Consider the following problem:

$$\text{maximize} \sum_{y\in\{+1,-1\}} \int_{\mathcal{D}} P(\boldsymbol{x}, y) y W'(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{6.6}$$

$$\text{subject to} \quad \int_{\mathcal{D}} P(\boldsymbol{x}) |W'(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \leq 1. \tag{6.7}$$

Here, the $\min(\cdot)$ function is removed from the QCMAP estimation and $P(\boldsymbol{x})$ is substituted for $Q(\boldsymbol{x})$. This can be transformed to

$$\max \int_{\mathcal{D}} P(\boldsymbol{x}) \left[W'(\boldsymbol{x}) \left\{ P(+1|\boldsymbol{x}) - P(-1|\boldsymbol{x}) \right\} \right] \mathrm{d}\boldsymbol{x}, \tag{6.8}$$

$$\text{s.t.} \int_{\mathcal{D}} P(\boldsymbol{x}) |W'(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \leq 1. \tag{6.9}$$

From the Cauchy-Schwarz inequality, the solution is given by

$$W'(\boldsymbol{x}) \propto P(+1|\boldsymbol{x}) - P(-1|\boldsymbol{x}). \tag{6.10}$$

This is a direct estimation of a difference of posterior probabilities up to a constant factor, so $W'(\boldsymbol{x})$ yields the same result as the MAP classifier.

If $Q(\boldsymbol{x}) \neq P(\boldsymbol{x})$, the problem can be written as

$$\max \int_{\mathcal{D}} Q(\boldsymbol{x}) \left[W'(\boldsymbol{x}) \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \left\{ P(+1|\boldsymbol{x}) - P(-1|\boldsymbol{x}) \right\} \right] \mathrm{d}\boldsymbol{x}, \tag{6.11}$$

$$\text{s.t.} \int_{\mathcal{D}} Q(\boldsymbol{x}) |W'(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \leq 1. \tag{6.12}$$

In this case, the solution is given by

$$W'(\boldsymbol{x}) \propto \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \left\{ P(+1|\boldsymbol{x}) - P(-1|\boldsymbol{x}) \right\}, \tag{6.13}$$

and $W'(\boldsymbol{x})$ too yields the same result as the MAP classifier. I call this the direct difference estimation of the posterior probabilities (DDEP).

Standard MAP estimation provides $P(+1|\boldsymbol{x})$ and $P(-1|\boldsymbol{x})$, individually. After estimation U compare these values, but their difference is sufficient for classification. A direct estimation of the difference (i.e., DDEP) may improve the accuracy of estimation as compared with the difference between individual estimations (the standard MAP estimation).

If $yW'(\boldsymbol{x}) > 1$, then it could be clearly discriminative. Because the important region is that where $yW'(\boldsymbol{x})$ is small, it is not smart to include large values of $yW'(\boldsymbol{x})$ in the evaluation, as large values of $yW'(\boldsymbol{x})$ affect the evaluation of the important region. In the QCMAP estimation, I clip the larger parts of $yW'(\boldsymbol{x})$ using the cost function $\min(yW'(\boldsymbol{x}), 1)$. Therefore we can consider the QCMAP estimation as a clipped DDEP.

6.1.3 Derivation of Mathematical Programming Problem

I define a discriminant model $D(\boldsymbol{x})$ as an approximation of $W'(\boldsymbol{x})$ that can be expressed as a linear combination of the components of the basis functions $\phi_i(\boldsymbol{x})$ and their parameters w_i:

$$W'(\boldsymbol{x}) \approx D(\boldsymbol{x}) := \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle , \tag{6.14}$$

where $\boldsymbol{w}$ and $\boldsymbol{\phi}(\boldsymbol{x})$ are a parameter vector and a basis function vector, respectively. Also assume that $\{\phi_i(\boldsymbol{x})\}_{i=1}^{M}$ is a set of linear independent functions.

Now, I describe QCMAP learning from training samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. Since the QCMAP method includes integration with $P(\boldsymbol{x}, y)$ and $Q(\boldsymbol{x})$, finding the optimal value of estimates is nontrivial. The objective part of QCMAP (6.2) can be considered as the maximization of the expected value of a cost function $\min(yW'(\boldsymbol{x}), 1)$ with respect to $(\boldsymbol{x}, y)$. Since expectation can be approximated by a sample mean, the method can be simplified through an approximation by replacing the expectation with the sample mean and by substituting Eq. (6.14) into the QCMAP estimation such that

$$\begin{aligned} \sum_{y \in \{+1,-1\}} \int_{\mathcal{D}} P(\boldsymbol{x}, y) \min(yD(\boldsymbol{x}), 1) \mathrm{d}\boldsymbol{x} &= \mathrm{E}_{\boldsymbol{x},y}[\min(y \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle , 1)] \\ &\simeq \frac{1}{N} \sum_{n=1}^{N} \min \left(y_n \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n) \rangle , 1 \right). \end{aligned} \tag{6.15}$$

The left part of Eq. (6.3) can be written as

$$\begin{aligned} \int_{\mathcal{D}} Q(\boldsymbol{x}) |D(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} &= \int_{\mathcal{D}} Q(\boldsymbol{x}) \left| \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}) \right|^2 \mathrm{d}\boldsymbol{x} \\ &= \sum_{i=1}^{M} \sum_{j=1}^{M} w_i w_j \int_{\mathcal{D}} Q(\boldsymbol{x}) \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}, \end{aligned} \tag{6.16}$$

where the (i, j) element of matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H}(i, j) := \int_{\mathcal{D}} Q(\boldsymbol{x}) \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{6.17}$$

We must calculate the integral in (6.17) to obtain $\boldsymbol{H}$. One way to do this is to apply a numerical integration method such as the Monte Carlo method [71], but this approach is computationally expensive. In Sec. 6.1.9 I provide an alternative approach that allows analytical calculation in a special

case. Finally, the QCMAP estimation criterion can be summarized as

$$\text{maximize} \quad \sum_{n=1}^{N} \min\left(y_n \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n)\right\rangle, 1\right), \tag{6.18}$$

$$\text{subject to} \quad \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w} \leq 1. \tag{6.19}$$

Here, the objective function is a convex piecewise linear function because of the presence of the min function. If $\boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w} = 0$, we have $D(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ in $\mathcal{D}$ because $Q(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathcal{D}$ and we have $\boldsymbol{w} = 0$ or $\boldsymbol{\phi}(\boldsymbol{x})$ are linearly dependent. Therefore $\boldsymbol{H}$ is a positive definite matrix and the constraint function is convex. QCMAP problems are convex and can be solved (trained) by the primal-dual interior point method [122] described in Section 6.1.7.

6.1.4 Relation to Hinge Loss Minimization

Hinge loss is defined as

$$J(\boldsymbol{w}) = \sum_{n=1}^{N} \max(1 - y_n \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n)\right\rangle, 0). \tag{6.20}$$

In fact, maximization of the QCMAP cost function and minimization of the hinge loss function are equivalent, since we have

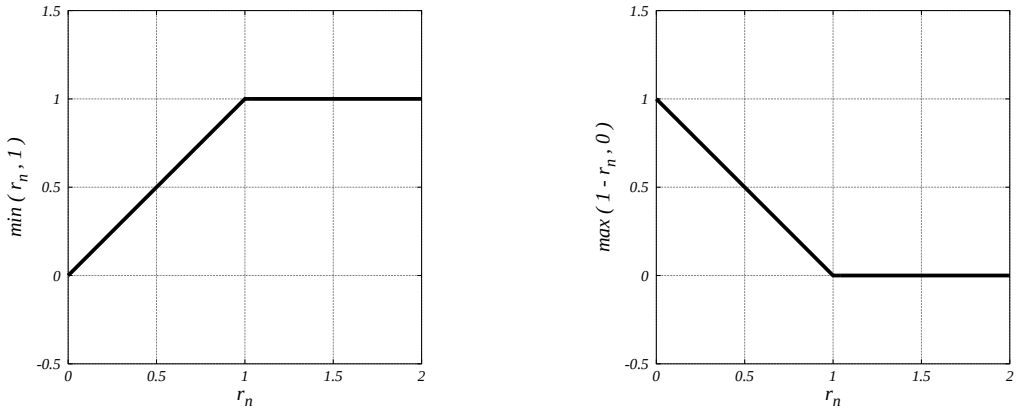
$$\max(1 - r_n, 0) = 1 - \min(r_n, 1), \tag{6.21}$$

where $r_n := y_n \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n)\right\rangle$. Fig. 6.1 show s a single term of the QCMAP cost function and the hinge loss function. The vertical axis expresses the evaluation value and horizontal axis expresses the value of r_n. The QCMAP estimation maximizes the evaluation value, so r_n seeks to be at least 1. On the other hand, the hinge loss minimization expects the evaluation value to be as small as possible, so again r_n seeks to be at least 1. The results of both approaches are therefore equal.

6.1.5 Relation to the Tikhonov Regularization

The constraint (6.19) on the QCMAP problem can be considered as a special case of the generalized Tikhonov regularization (TR) [97]. As described above, the matrix $\boldsymbol{H}$ is a positive definite matrix, and so can be decomposed as $\boldsymbol{H} = \boldsymbol{\Gamma}^T \boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma}$ is called the Tikhonov matrix. Moreover, the TR constraint can be written as $||\boldsymbol{\Gamma}\boldsymbol{w}||^2 \leq \sigma$, where σ is a regularization (tradeoff) parameter. The key point of TR is choosing the Tikhonov matrix $\boldsymbol{\Gamma}$ and σ, but this selection is nontrivial due to the large number of degrees of freedom.

On the other hand, the key point of the QCMAP approach is to select the weight function $Q(\boldsymbol{x})$. Unfortunately, unlike $\boldsymbol{\Gamma}$ we cannot arbitrarily select $Q(\boldsymbol{x})$. For some $\boldsymbol{\Gamma}$, there may not exist a $Q(\boldsymbol{x})$ that satisfies condition (6.1). However, we can select $Q(\boldsymbol{x})$ instead of $\boldsymbol{\Gamma}$ in the TR method because the existence of a $\boldsymbol{\Gamma}$ is ensured for any $Q(\boldsymbol{x})$ satisfying Eq. (6.1).



(a) QCMAP cost function (b) Hinge loss function

Figure 6.1: The QCMAP cost function and the hinge loss function

We can therefore say that the quadratic constraint of QCMAP is a special case of the TR method. The contribution of our method is that if $\boldsymbol{H}$ is obtained by a $Q(\boldsymbol{x})$ that satisfies condition (6.1), and if we let $\sigma = 1$, then it is sufficient to provide a MAP-based classification.

6.1.6 Uniqueness of QCMAP Solutions

I next examine whether a QCMAP problem has a unique solution. In fact, we cannot strictly say that QCMAP problems always have a unique solution. However, the QCMAP problems have unique solutions in many cases.

Fig. 6.2 illustrates two potential solutions to a QCMAP problem. If the hinge loss function cannot be optimized to zero because $\boldsymbol{H}$ is a positive definite matrix then the QCMAP problem has a unique solution (Fig. 6.2(a)). On the other hand, if the hinge loss function can be easily optimized to zero, then the solution to the QCMAP problem permits some range within $\{\boldsymbol{w}|\ J(\boldsymbol{w}) = 0\ \cap\ \boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w} \leq 1\}$ (Fig. 6.2(b)). An empirical risk (including hinge loss) of zero implies over-fitting and that the degree of freedom of the model is too high, or that the problem is overly simplistic. In such cases, we might have to reduce the degree of freedom of the model to have a unique solution.

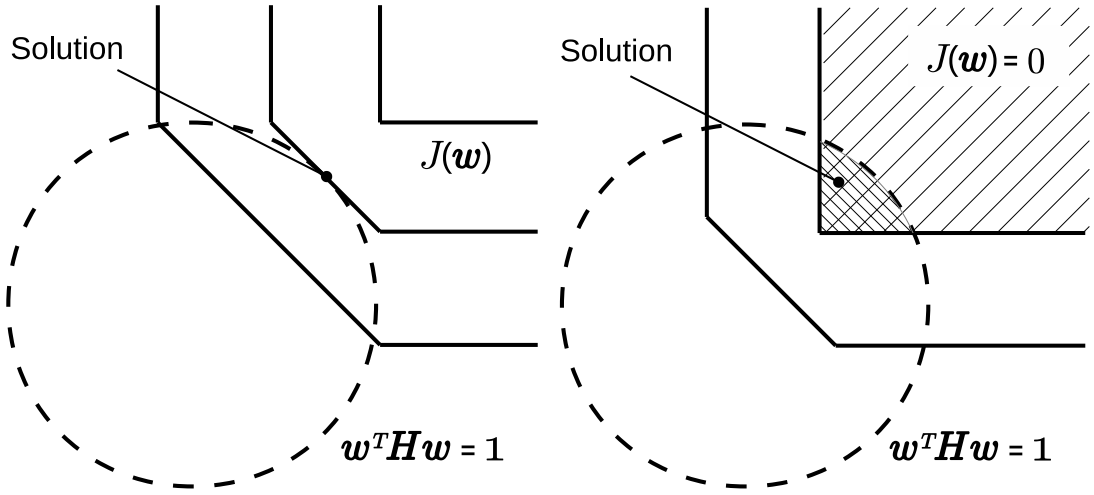
I propose using an approximation function as the hinge loss function in Sec. 6.1.7. In this case, the empirical risk cannot be optimized to zero, and a unique solution is always given.

6.1.7 Solution Method for QCMAP and Approximation of Cost Function

To solve the QCMAP problem, I use the primal-dual interior point algorithm [122]. Let $\boldsymbol{w} \in \mathbb{R}^M$ be a primal parameter vector and $f(\boldsymbol{w})$ be a convex function, and consider the following problem:

$$\text{minimize} \quad f(\boldsymbol{w}), \tag{6.22}$$

$$\text{subject to} \quad \boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w} \leq 1, \tag{6.23}$$



(a) (b)

Figure 6.2: Two solution areas for the QCMAP problem: solid and dotted lines depict the contour lines of $J(\boldsymbol{w})$ and $\boldsymbol{w}^T H \boldsymbol{w} = 1$, respectively

where $\boldsymbol{H}$ is an (M, M)-positive semidefinite matrix. A solution to this problem requires that the function $f(\boldsymbol{w})$ is differentiable, yet (6.18) and (6.20) are not differentiable. Therefore, I define $f(\boldsymbol{w})$ as an approximation function of (6.20) given by

$$J(\boldsymbol{w}) \approx f(\boldsymbol{w}) := \sum_{n=1}^{N} h(r_n|t), \tag{6.24}$$

where

$$h(r_n|t) := \frac{1}{t}\log(1 + e^{t(1-r_n)}), \tag{6.25}$$

and t is a parameter [87]. As $t \to \infty$, the objective function $f(\boldsymbol{w})$ approaches the hinge loss function. If $r_n \geq 1$ (i.e., $t(1 - r_n)$ is zero or negative), then $\lim_{t\to\infty} h(r_n|t) = 0$. If $r_n < 1$ (i.e., $t(1 - r_n)$ is positive) and t is very large, then $e^{(t(1-r_n))} \gg 1$ and $\log(1 + e^{(t(1-r_n))}) \approx \log(e^{(t(1-r_n))}) = t(1 - r_n)$. Therefore, we can derive

$$\begin{aligned} \lim_{t\to\infty} h(r_n|t) &= \lim_{t\to\infty} \frac{1}{t}\log(1 + e^{t(1-r_n)}) \\ &= \begin{cases} 0 & r_n \geq 1 \\ 1 - r_n & r_n < 1 \end{cases}. \end{aligned} \tag{6.26}$$

This is equivalent to the hinge loss, i.e., $\max(1 - r_n, 0)$.

Smaller values of t yield smoother approximations of the hinge loss. For example, Fig. 6.3 shows $h(r_n|t)$ for $t = 1, 2$, and 3. Smoothness of the objective function is an important factor for smooth convergence in optimization methods. However, smoothness and sharpness are tradeoffs: sharpness yields a better approximation of the hinge loss but complicates the optimization procedure.

In this experiment, I fixed $t = 2$ (see Sec. 7.4.4) because empirical investigation showed that this value led to good, smooth optimizations. Methods of selecting the value of t remains an open problem.

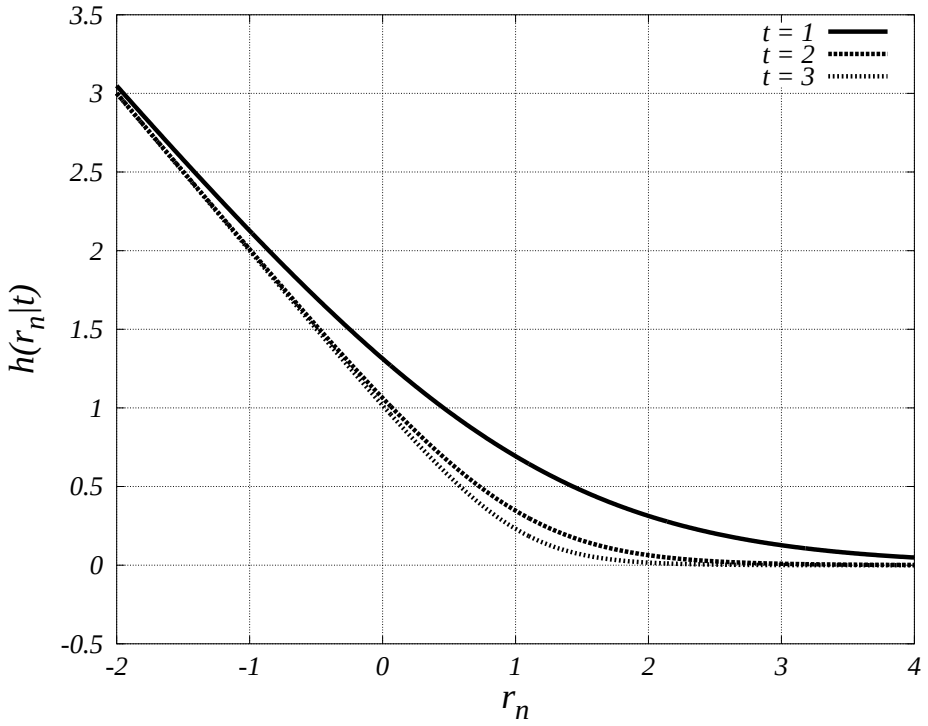


Figure 6.3: Approximation Function $h(r_n|t)$ for changing t

Primal-Dual Interior Point Algorithm for QCMAP

First, I consider the Lagrange function of this problem:

$$L(\boldsymbol{w}, z) = f(\boldsymbol{w}) - z(1 - \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}), \tag{6.27}$$

where z is a dual parameter. From the Karush-Kuhn-Tucker (KKT) conditions, we have

$$\frac{\partial f}{\partial \boldsymbol{w}} + 2z\boldsymbol{H}\boldsymbol{w} = \boldsymbol{0}, \tag{6.28}$$

$$z(1 - \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}) = 0, \tag{6.29}$$

$$z \geq 0, \quad 1 - \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w} \geq 0. \tag{6.30}$$

A pair $(\boldsymbol{w}, z)$ that satisfies condition (6.30) is called a "primal-dual interior point." Furthermore, if $(\boldsymbol{w}, z)$ satisfies Eqs. (6.28) and (6.29), then the pair is the optimal solution of the primal dual problem of QCMAP.

Thus, I define the residual function as

$$\boldsymbol{r}(\boldsymbol{w}, z) := \begin{pmatrix} \frac{\partial f}{\partial \boldsymbol{w}} + 2z\boldsymbol{H}\boldsymbol{w} \\ z(1 - \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}) \end{pmatrix}, \tag{6.31}$$

and the goal of the solution algorithm is to optimize $(\boldsymbol{w}, z)$ such that

$$||\boldsymbol{r}(\boldsymbol{w}, z)|| \to 0, \tag{6.32}$$

Algorithm 27 Primal-dual interior point (PDIP) algorithm

Input: training samples, $\boldsymbol{H}$, T, $I_{\max}$ and $\alpha_{\min}$;
Initialize: $\boldsymbol{w}_0 = \boldsymbol{0}$, $\boldsymbol{z}_0 = 1$, $\alpha = 1$ and $k = 0$;
while $\frac{||\boldsymbol{r}(\boldsymbol{w}_k, z_k)||}{M+1} > T$ & $k < I_{\max}$ & $\alpha > \alpha_{\min}$ **do**
  Update $(\boldsymbol{w}_{k+1}, z_{k+1})$ by (6.36), (6.37), and Algorithm 28;
  $k = k + 1$;
end while
Output: $\boldsymbol{w}_k$;

while satisfying Eq. (6.30). In the actual algorithm, I use a tolerance T (e.g., $T = 10^{-5}$) as a threshold for convergence of the residual function. Specifically, the convergence condition is given by

$$\frac{||\boldsymbol{r}(\boldsymbol{w}, z)||}{M+1} < T. \tag{6.33}$$

The basic concept of the primal-dual interior point algorithm is as follows: First, I set the values of an initial pair $(\boldsymbol{w}_0, z_0)$ such that the pair is an interior point, for example, $\boldsymbol{w}_0 = \boldsymbol{0}$ and $z_0 = 1$. Next, I iteratively update $(\boldsymbol{w}_k, z_k)$ through

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha\Delta\boldsymbol{w}, \tag{6.34}$$

$$z_{k+1} = z_k - \alpha\Delta z. \tag{6.35}$$

Here, $(\Delta\boldsymbol{w}, \Delta z)$ is the direction for the update and α is the step size. Once the residual function $\boldsymbol{r}(\boldsymbol{w}_k, z_k)$ converges, the algorithm is completed. An outline of the primal-dual interior point (PDIP) algorithm is described in Algorithm 27.

In this regard, I set the maximum number of iterations $I_{\max}$ and the minimum step size $\alpha_{\min}$ (e.g., $I_{\max} = 500$ and $\alpha_{\min} = 10^{-6}$).

The key point in executing this algorithm is correctly determining $(\Delta\boldsymbol{w}, \Delta z)$ and α. Many methods and algorithms have been proposed and developed for doing so. In this paper, I introduce an interior path-following method to decide $(\Delta\boldsymbol{w}, \Delta z)$ and a back-tracking search algorithm to decide α.

Path-Following Method

The interior path-following method provides a $(\Delta\boldsymbol{w}, \Delta z)$ that approaches the optimal solution following the analytic center path [41]. In this method, $(\Delta\boldsymbol{w}, \Delta z)$ is given by

$$\Delta\boldsymbol{w} = \left(A + \frac{z_k}{c}\boldsymbol{b}\boldsymbol{b}^T\right)^{-1}\left(\boldsymbol{g} + \frac{\mu}{c}\boldsymbol{b}\right), \tag{6.36}$$

$$\Delta z = z_k - \frac{\mu}{c} + \frac{z_k}{c}\boldsymbol{b}^T\Delta\boldsymbol{w}_k, \tag{6.37}$$

Algorithm 28 Back tracking search algorithm

Input: $0 < \alpha_{\min} < \alpha_{\max}$, $0 < \beta < 1$, $(\boldsymbol{w}, z)$ and $(\Delta\boldsymbol{w}, \Delta z)$;
Initialize: $\alpha = \alpha_{\max}$;
$(\boldsymbol{w}', z') = (\boldsymbol{w}, z) - \alpha(\Delta\boldsymbol{w}, \Delta z)$;
repeat
  $\alpha \leftarrow \beta\alpha$;
  $(\boldsymbol{w}', z') = (\boldsymbol{w}, z) - \alpha(\Delta\boldsymbol{w}, \Delta z)$;
until $(\boldsymbol{w}', z')$ satisfies conditions (6.30) and (6.42), or $\alpha < \alpha_{\min}$
Output: α, $(\boldsymbol{w}', z')$;

where

$$A = \nabla^2 f(\boldsymbol{w}_k) + 2z_k\boldsymbol{H}, \quad \boldsymbol{b} = 2\boldsymbol{H}\boldsymbol{w}_k, \tag{6.38}$$

$$c = 1 - \boldsymbol{w}_k^T\boldsymbol{H}\boldsymbol{w}_k, \quad \boldsymbol{g} = \frac{\partial}{\partial\boldsymbol{w}} f(\boldsymbol{w}_k), \tag{6.39}$$

$$\mu = \gamma z_k(1 - \boldsymbol{w}_k^T\boldsymbol{H}\boldsymbol{w}_k), \tag{6.40}$$

$$\gamma = \min\left(\sqrt{\frac{||\boldsymbol{r}(\boldsymbol{w}_k, z_k)||}{M+1}}, 0.5\right), \tag{6.41}$$

and μ is a barrier parameter to prevent $(\boldsymbol{w}, z)$ from falling outside the region of interior points.

Backtracking Search Algorithm

In the backtracking search algorithm, I iteratively decrease the value of $\alpha_{\min} \leq \alpha \leq \alpha_{\max}$ until it satisfies condition (6.30) and

$$||\boldsymbol{r}_\mu(\boldsymbol{w}', z')|| \leq (1 - \alpha(1 - \eta))||\boldsymbol{r}_\mu(\boldsymbol{w}, z)||, \tag{6.42}$$

where

$$\boldsymbol{r}_\mu(\boldsymbol{w}, z) := \begin{pmatrix} \frac{\partial f}{\partial\boldsymbol{w}} + 2z\boldsymbol{H}\boldsymbol{w} \\ z(1 - \boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w}) - \mu \end{pmatrix}, \tag{6.43}$$

$$\eta = \min\left(\frac{||\boldsymbol{r}_\mu(\boldsymbol{w}, z)||}{M+1}, 0.25\right). \tag{6.44}$$

Algorithm 28 gives pseudocode for this search method, where values for $\alpha_{\min}$, $\alpha_{\max}$, and β are selected a priori, for example, $\alpha_{\min} = 10^{-6}$, $\alpha_{\max} = 0.995$, $\beta = 0.75$.

6.1.8 Equivalence of QCMAP to LSR and SVM

In this section, I briefly discuss the equivalence of the QCMAP estimation with LSR and the linear SVM. The QCMAP estimation has a weight function $Q(\boldsymbol{x})$, and an LSR or the linear SVM can be derived from the QCMAP estimation by selecting appropriate functions for $Q(\boldsymbol{x})$.

Equivalence to LSR

I first present the equivalence to LSR. The solution of an LSR problem, $\widehat{\boldsymbol{w}}_{\mathrm{LSR}}$, is given by

$$\widehat{\boldsymbol{w}}_{\mathrm{LSR}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}, \tag{6.45}$$

where

$$\boldsymbol{\Phi} := \begin{pmatrix}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_N)\end{pmatrix}^T, \tag{6.46}$$

$$\boldsymbol{y} := \begin{pmatrix}y_1 & y_2 & \cdots & y_N\end{pmatrix}^T. \tag{6.47}$$

If we choose $Q_A(\boldsymbol{x}) = P(\boldsymbol{x})$ and the expectation is approximated by the sample mean, then the constraint matrix is given by

$$\begin{aligned}\boldsymbol{H}_A(i,j) &= \int_{\mathcal{D}} P(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})^T \mathrm{d}\boldsymbol{x} = \mathrm{E}_{\boldsymbol{x}}[\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})^T] \\ &\simeq \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^T = \frac{1}{N}\boldsymbol{\Phi}^T\boldsymbol{\Phi}.\end{aligned} \tag{6.48}$$

The constraint is then written in the form

$$\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} \leq N. \tag{6.49}$$

Therefore, the QCMAP problem with $Q_A(\boldsymbol{x})$ is given by

$$\text{maximize} \quad \sum_{n=1}^{N}\min\left(y_n\left\langle\boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n)\right\rangle, 1\right), \tag{6.50}$$

$$\text{subject to} \quad \boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} \leq N. \tag{6.51}$$

Theorem 2. *The solution of a QCMAP problem with $Q_A(\boldsymbol{x})$ is equivalent to $\widehat{\boldsymbol{w}}_{\mathrm{LSR}}$.*

Proof of Theorem 2

I prove that $\widehat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$ is the solution of the following optimization problem:

$$\text{maximize} \quad \sum_{n=1}^{N}\min\left(y_n\left\langle\boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n)\right\rangle, 1\right), \tag{6.52}$$

$$\text{subject to} \quad \boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} \leq N, \tag{6.53}$$

where

$$\boldsymbol{\Phi} := \begin{pmatrix}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_N)\end{pmatrix}^T, \tag{6.54}$$

$$\boldsymbol{y} := \begin{pmatrix}y_1 & y_2 & \cdots & y_N\end{pmatrix}^T. \tag{6.55}$$

Proof. First, this objective can be rewritten using slack variables ξ_n:

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{N} \xi_n, && (6.56)\\ \text{subject to} \quad & y_n \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_n) \rangle \geq 1 - \xi_n, && (6.57)\\ & \xi_n \geq 0, \quad n = 1, \ldots, N, && (6.58)\\ & \boldsymbol{w}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w} \leq 1. && (6.59) \end{aligned}$$

Next I derive its dual problem in terms of a Lagrange function with KKT conditions.

The corresponding Lagrangian of this problem is then

$$\begin{aligned} L(\boldsymbol{\xi}, \boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \eta) &= \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \beta_n \xi_n \\ &- \sum_{n=1}^{N} \gamma_n (\xi_n - 1 + y_n \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n)) - \frac{\eta}{2}(N - \boldsymbol{w}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w}), \end{aligned} \tag{6.60}$$

where β_n, γ_n, and η are Lagrange coefficients. From the KKT conditions we have

$$\begin{aligned} & \frac{\partial L}{\partial \xi_n} = 1 - \beta_n - \gamma_n = 0, && (6.61)\\ & \frac{\partial L}{\partial w_i} = \sum_{n=1}^{N} \phi_i(\boldsymbol{x}_n)(\eta \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n \gamma_n) = 0, && (6.62)\\ & \beta_n \xi_n = 0, \ \gamma_n(\xi_n - 1 + y_n \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n)) = 0, && (6.63)\\ & \frac{\eta}{2}(N - \boldsymbol{w}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w}) = 0, && (6.64)\\ & \beta_n \geq 0, \ \gamma_n \geq 0, \ \eta \geq 0, \ n = 1, \ \ldots, \ N. && (6.65) \end{aligned}$$

Thus, the dual problem can be expressed as

$$\begin{aligned} \text{maximize} F(\boldsymbol{\gamma}, \eta) &= -\frac{1}{2\eta} \sum_{n=1}^{N} \gamma_n^2 + \sum_{n=1}^{N} \gamma_n - \frac{\eta}{2} N, \\ \text{subject to} \quad & 0 \leq \gamma_n \leq 1, \ \eta \geq 0, \ n = 1, \ldots, N. \end{aligned}$$

The solution of this problem is given by

$$\frac{\partial F}{\partial \gamma_n} = -\frac{1}{\eta} \gamma_n + 1 = 0 \quad \Rightarrow \quad \gamma_n = \eta. \tag{6.66}$$

The optimal parameter vector $\widehat{\boldsymbol{w}}$ from Eq. (6.62) is then

$$\begin{aligned} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)(\eta \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n \eta) &= \mathbf{0} \ (\text{since } \gamma_n = \eta) \\ \eta \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n) \boldsymbol{\phi}(\boldsymbol{x}_n)^T \boldsymbol{w} &= \eta \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n) y_n \\ \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{w} &= \boldsymbol{\Phi}^T \boldsymbol{y} \\ \Rightarrow \quad \widehat{\boldsymbol{w}} &= (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}. \end{aligned} \tag{6.67}$$

□

Equivalence to SVM

Here, I show the relation between the QCMAP estimation and the linear SVM. To describe the key points first, the QCMAPs with uniform weight function are equivalent to a special case of SVM in a linear discriminant model $D(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$. Similarly, if we use some orthonormal basis functions and a uniform weight function for the QCMAP, it is equivalent to a special case of SVM. Unfortunately, the QCMAP is not equivalent to SVM in general, non-linear models. For example, we cannot claim equivalence to SVM when we use a Gaussian kernel model. However, we can show that the QCMAP is similar to the SVM when a weighting function reaches a uniform function.

Theorem 3. *Assume that* $||\boldsymbol{x}|| = 1$ *and the data domain* $\mathcal{D}$ *is equal to the unit* $(d-1)$*-dimensional hyperspherical surface* S^{d-1} *(i.e.,* $S^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}|| = 1\}$*). Choose* $Q_B(\boldsymbol{x}) = 1/S_{d-1}$*, where* S_{d-1} *is the surface area of* S^{d-1}*. Then condition* (6.1) *and*

$$\int_{\mathcal{D}} Q_B(\boldsymbol{x}) |\langle \boldsymbol{w}, \boldsymbol{x} \rangle|^2 \mathrm{d}\boldsymbol{x} = \frac{1}{d} ||\boldsymbol{w}||^2 \tag{6.68}$$

are satisfied.

Proof of Theorem 3

I prove that if we assume $\boldsymbol{x} \in S^{d-1}$, $Q_B(\boldsymbol{x}) := \frac{1}{S_{d-1}}$, where S^{d-1} presents the unit $(d-1)$-dimensional hyper-spherical surface (i.e., $S^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}|| = 1\}$), and S_{d-1} is the surface area of S^{d-1}, then it is satisfied that

$$\int_{S^{d-1}} Q_B(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1, \quad Q_B(\boldsymbol{z}) > 0 \text{ for all } \boldsymbol{z} \in \mathcal{D}, \tag{6.69}$$

$$\int_{S^{d-1}} Q_B(\boldsymbol{x}) |\langle \boldsymbol{w}, \boldsymbol{x} \rangle|^2 \mathrm{d}\boldsymbol{x} = \frac{1}{d} ||\boldsymbol{w}||^2. \tag{6.70}$$

Proof. First, we can derive

$$\int_{S^{d-1}} Q_B(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{1}{S_{d-1}} \int_{S^{d-1}} \mathrm{d}V_{d-1}(\boldsymbol{x}) = 1, \tag{6.71}$$

where $\mathrm{d}V_{d-1}(\boldsymbol{x})$ is the volume element of S^{d-1} and is given by

$$\begin{aligned} dV_{M-1} =& \sin^{M-2}(\psi_1) \sin^{M-3}(\psi_2) \\ & \cdots \sin(\psi_{M-2}) \mathrm{d}\psi_1 \mathrm{d}\psi_2 \ldots \mathrm{d}\psi_{M-1}. \end{aligned} \tag{6.72}$$

Then, $Q_B(\boldsymbol{x})$ satisfies eq. (6.69).

Furthermore, we have

$$\int_{S^{d-1}} \frac{1}{S_{d-1}} |\langle \boldsymbol{w}, \boldsymbol{x} \rangle|^2 \mathrm{d}\boldsymbol{x} = \frac{||\boldsymbol{w}||^2}{S_{d-1}} \int_{S^{d-1}} \cos^2\theta \mathrm{d}V_{d-1}(\boldsymbol{x}), \tag{6.73}$$

where θ is the angle between $\boldsymbol{w}$ and $\boldsymbol{x}$. We can assume $\psi_1 = \theta$ because the integral is isometrically defined. Thus, Eq. (6.73) is rewritten as

$$\begin{aligned}
&\frac{||\boldsymbol{w}||^2}{S_{d-1}} \int_0^{\pi} \cdots \int_0^{\pi} \int_0^{2\pi} \cos^2\theta \mathrm{d}V_{d-1} \\
&= \frac{||\boldsymbol{w}||^2}{S_{d-1}} \int_0^{\pi} \cdots \int_0^{\pi} \int_0^{2\pi} \cos^2\theta \sin^{d-2}(\theta) \mathrm{d}\theta \mathrm{d}V_{d-2} \\
&= ||\boldsymbol{w}||^2 \frac{S_{d-2}}{S_{d-1}} \int_0^{\pi} \cos^2\theta \sin^{d-2}(\theta) \mathrm{d}\theta = \frac{1}{d}||\boldsymbol{w}||^2. \qquad (6.74)
\end{aligned}$$

□

From Theorem 3, the constraint of QCMAP with $Q_B(\boldsymbol{x})$ is given by $||\boldsymbol{w}||^2 \leq d$, and the constraint matrix becomes $\boldsymbol{H}_B = \frac{1}{d}\boldsymbol{I}_d$, where $\boldsymbol{I}_d \in \mathbb{R}^{d\times d}$ is an identity matrix. In this case, the criterion of QCMAP estimation with $Q_B(\boldsymbol{x})$ is given by

$$\text{minimize} \quad J(\boldsymbol{w}), \qquad (6.75)$$

$$\text{subject to} \quad ||\boldsymbol{w}||^2 \leq d. \qquad (6.76)$$

On the other hand, the criterion of SVM estimation is given by

$$\text{minimize} \quad J(\boldsymbol{w}) + \frac{\lambda}{2}||\boldsymbol{w}||^2, \qquad (6.77)$$

where λ is a regularization parameter. The differences between the two criteria are only whether $||\boldsymbol{w}||^2$ is in the objective or the constraint part of the criterion, and whether a regularization parameter exists. Because the QCMAP estimation does not have a regularization parameter, we are unable to state that the two criteria are strictly equivalent. However, the resulting effects of the two criteria are equivalent: they both minimize $J(\boldsymbol{x})$ and regularize $||\boldsymbol{w}||^2$.

Here, I prove that QCMAP and SVM estimators are equivalent under the condition $||\boldsymbol{w}||^2 = d$. First, we have that the solution of QCMAP is always included in an area where $||\boldsymbol{w}||^2 = d$ is satisfied. Next, I assume that $||\boldsymbol{w}||^2 = d$ is satisfied in the SVM estimation. Since $\frac{\lambda}{2}||\boldsymbol{w}||^2$ is now constant, it can be disregarded. In this case, the criterion of SVM estimation can be expressed by minimization of $J(\boldsymbol{w})$, s.t. $||\boldsymbol{w}||^2 = d$. Therefore, the QCMAP estimation with $Q_B(\boldsymbol{x})$ is realized by an SVM when its solution satisfies $||\boldsymbol{w}||^2 = d$. If we choose λ such that $||\boldsymbol{w}||^2 = d$ in SVM, then SVM and QCMAP estimators are equivalent. Specifically, an SVM with $||\boldsymbol{w}|| = d$ can be regarded as a MAP-based classifier.

Next, I consider a set of orthonormal basis functions $\{\phi_i(\boldsymbol{x})\}_{i=1}^M$ as

$$\int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \qquad (6.78)$$

If we assume that Ω is the volume of $\mathcal{D}$, which is bounded, and uniform weight function $Q(\boldsymbol{x}) = \frac{1}{\Omega}$ exists, then we have

$$\frac{1}{\Omega}\sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \frac{1}{\Omega}||\boldsymbol{w}||^2. \qquad (6.79)$$

Similar to the previous description, this is equivalent to a special case of SVM.

Next, I consider a Gaussian kernel model as

$$D(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}), \tag{6.80}$$

where $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma||\boldsymbol{x}-\boldsymbol{y}||^2)$ and α_n are parameters. In the kernel method, the criterion of SVM is given by

$$\text{minimize } J(\boldsymbol{\alpha}) + \frac{\lambda}{2}\boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha}, \tag{6.81}$$

where $\boldsymbol{K}(i,j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Assume that a uniform weight function $Q(\boldsymbol{x}) = \frac{1}{\Omega}$ can be defined, then the constraint matrix is given by

$$H(i,j) = \frac{1}{\Omega}\int_{\mathcal{D}} k(\boldsymbol{x}_i, \boldsymbol{x})k(\boldsymbol{x}_j, \boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{6.82}$$

$$= \frac{1}{\Omega}\left(\frac{\pi}{2\gamma}\right)^{\frac{d}{2}} \sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \tag{6.83}$$

In SVM with a Gaussian kernel, the regularization term depends on $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. On the other hand, the constraint of QCMAP depends on $\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}$. The two approaches are different, but they are similar. This is indicating that a QCMAP with Gaussian kernel model and the uniform weight function is similar to the kernel SVM.

6.1.9 Gaussian QCM Classifier

I next introduce the Gaussian QCM (GQCM) classifier. The key points are that we choose a pdf with a normal distribution for $Q(\boldsymbol{x})$, then fit a Gaussian kernel model to $D(\boldsymbol{x})$. Many applications using the Gaussian kernel model are studied and developed for various research topics [76, 81]. As a result, we can analytically calculate the integral to obtain $\boldsymbol{H}$ in (6.3).

Construction of GQCM Classifiers

In this section I consider a concrete definition of the discriminant function, given by

$$D(\boldsymbol{x}) := \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + \beta, \tag{6.84}$$

where $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma||\boldsymbol{x}-\boldsymbol{y}||^2)$ and, α_n and β are parameters. We can regard this definition (6.84) as the inner product between a parameter vector and a basis function vector, redefining it as

$$\boldsymbol{w} := \begin{pmatrix}\alpha_1 & \cdots & \alpha_N & \beta\end{pmatrix}^T, \tag{6.85}$$

$$\boldsymbol{\phi}(\boldsymbol{x}) := \begin{pmatrix}k(\boldsymbol{x}_1, \boldsymbol{x}) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}) & 1\end{pmatrix}^T. \tag{6.86}$$

Therefore, we have

$$\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + \beta. \tag{6.87}$$

Moreover, if we use the following weight function

$$\begin{aligned} Q(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) :=& N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ =& \frac{1}{(\sqrt{2\pi})^d \sqrt{|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \end{aligned} \tag{6.88}$$

then the constraint matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H}(i,j) := \int_{\mathcal{D}} \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathrm{d}\boldsymbol{x}. \tag{6.89}$$

QCMAP estimation using Eq. (6.89) thus provides the GQCM classifiers. To calculate $\boldsymbol{H}$, we must compute the integral given in Eq. (6.89). However, if Gaussian kernels are employed as basis functions and $Q(\boldsymbol{x})$ is still a normal distribution in the original space, we can analytically calculate $\boldsymbol{H}$.

Analytical Calculation of Constraint Matrix

Since the basis functions are either Gaussian kernels or 1, $\boldsymbol{H}$ is given by

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{U} & \boldsymbol{u} \\ \boldsymbol{u}^T & u \end{pmatrix}, \tag{6.90}$$

where, $\boldsymbol{U}$, $\boldsymbol{u}$, and u are an $(N \times N)$-matrix, an N-dimensional vector, and a value, respectively:

$$\boldsymbol{U}(i,j) = \int_{\mathcal{D}} N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) k(\boldsymbol{x}_i, \boldsymbol{x}) k(\boldsymbol{x}_j, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{6.91}$$

$$\boldsymbol{u}(i) = \int_{\mathcal{D}} N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) k(\boldsymbol{x}_i, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{6.92}$$

$$u = \int_{\mathcal{D}} N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathrm{d}\boldsymbol{x} = 1. \tag{6.93}$$

By using the general equation of a Gaussian integral [92, 16],

$$\int e^{-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}} \mathrm{d}\boldsymbol{x} = \sqrt{\frac{(2\pi)^d}{|\boldsymbol{A}|}} e^{\frac{1}{2}\boldsymbol{b}^T \boldsymbol{A}^{-1} \boldsymbol{b}}, \tag{6.94}$$

we can analytically calculate Eqs. (6.91) and (6.92) as follows. From Eq. (6.91)

$$\boldsymbol{U}(i,j) = \frac{1}{\sqrt{|4\gamma \boldsymbol{\Sigma} + \boldsymbol{I}_d|}} \exp\left(\frac{1}{2} \boldsymbol{b}_{ij}^T \boldsymbol{A}^{-1} \boldsymbol{b}_{ij} + c_{ij} \right), \tag{6.95}$$

where

$$\boldsymbol{A} = 4\gamma \boldsymbol{I}_d + \boldsymbol{\Sigma}^{-1}, \tag{6.96}$$

$$\boldsymbol{b}_{ij} = 2\gamma(\boldsymbol{x}_i + \boldsymbol{x}_j) + \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}, \tag{6.97}$$

$$c_{ij} = -\gamma(||\boldsymbol{x}_i||^2 + ||\boldsymbol{x}_j||^2) - \frac{1}{2} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}. \tag{6.98}$$

The N-dimensional vector $\boldsymbol{u}$ is given by

$$\boldsymbol{u}(i) = \frac{1}{\sqrt{|2\gamma\boldsymbol{\Sigma} + \boldsymbol{I}_d|}} \exp\left(\frac{1}{2}\boldsymbol{b}_i'^T \boldsymbol{A}'^{-1}\boldsymbol{b}_i' + c_i'\right), \tag{6.99}$$

where

$$\boldsymbol{A}' = 2\gamma\boldsymbol{I}_d + \boldsymbol{\Sigma}^{-1}, \tag{6.100}$$

$$\boldsymbol{b}_i' = 2\gamma\boldsymbol{x}_i + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \tag{6.101}$$

$$c_i' = -\gamma||\boldsymbol{x}_i||^2 - \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}. \tag{6.102}$$

6.1.10 Extended GQCM Classifiers

The weight functions used thus far are a pdf of $\boldsymbol{x}$, a uniform distribution, and a normal distribution. However, we do not know what constitutes a good weight function. To ascertain the ideal weight function, experiments involving various new weight functions are necessary.

Mixture of Gaussians Model

First, I propose to use the mixture of Gaussians model for $Q(\boldsymbol{x})$ given by

$$Q(\boldsymbol{x}|\rho_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \rho_L, \boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L) := \sum_{l=1}^{L} \rho_l N(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l), \tag{6.103}$$

where ρ_l are weight parameters that satisfy

$$0 \leq \rho_l \leq 1, \quad \sum_{l=1}^{L} \rho_l = 1. \tag{6.104}$$

From the equality in (6.89), $\boldsymbol{H}$ can be calculated analytically, and

$$\boldsymbol{H}(i,j) = \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) \sum_{l=1}^{L} \rho_l N(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\mathrm{d}\boldsymbol{x} \tag{6.105}$$

$$= \sum_{l=1}^{L} \rho_l \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) N(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\mathrm{d}\boldsymbol{x} \tag{6.106}$$

$$= \sum_{l=1}^{L} \rho_l \boldsymbol{H}_l(i,j), \tag{6.107}$$

where

$$\boldsymbol{H}_l(i,j) = \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) N(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\mathrm{d}\boldsymbol{x}. \tag{6.108}$$

QCMAP estimation with (6.107) therefore provides a novel classifier, which I term the "mixture of Gaussians QCMAP" (MOGQCM) classifier. The problem in using this method is then how to select values for parameters L, ρ_l, $\boldsymbol{\mu}_l$, and $\boldsymbol{\Sigma}_l$.

I propose to choose parameters L, ρ_l, $\boldsymbol{\mu}_l$, and $\boldsymbol{\Sigma}_l$ based on the strategy of the kernel density estimation. Using the kernel density function as the weight function gives

$$Q(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} N(\boldsymbol{x}|\boldsymbol{x}_n, s\boldsymbol{I}_d). \tag{6.109}$$

In other words, I choose $L := N$, $\rho_l := \frac{1}{N}$, $\boldsymbol{\mu}_l = \boldsymbol{x}_l$, and $\boldsymbol{\Sigma}_l = s\boldsymbol{I}_d$, for $l = 1, \ldots, N$.

This classifier is similar to an LSR for very small s, since we can prove that if $s \to 0$, then

$$\boldsymbol{H}_{\mathrm{kdw}}(i,j) \to \frac{1}{N}\sum_{n=1}^{N} \phi_i(\boldsymbol{x}_n)\phi_j(\boldsymbol{x}_n). \tag{6.110}$$

Conversely, this classifier is similar to an SVM for very large s, because $Q(\boldsymbol{x})$ then approximates a uniform distribution.

I call this the "KDEQCM" classifier.

Difference of Gaussians Function

In general, a mixture of Gaussian distributions must satisfy condition (6.104). However, we can break condition (6.104) when condition (6.1) would still hold. As a result, I define the difference of Gaussians (DOG) function as

$$Q(\boldsymbol{x}) := \left(1 + \frac{\kappa}{\nu^d - 1}\right) N(\boldsymbol{x}|\boldsymbol{\mu}, \nu^2\boldsymbol{\Sigma}) - \frac{\kappa}{\nu^d - 1} N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6.111}$$

where $0 < \kappa < 1$ and $\nu > 1$. In general, the DOG function is often used as a window function for an edge extraction filter [27]; however, here I use the DOG function in a very different way (as a weight function).

If we assume that $\boldsymbol{\Sigma}$ is a diagonal matrix, then this weight function always satisfies (6.1). Fig. 6.4 shows examples of such weight functions, and we see that if ν is increased, the weight function has a smoother and wider distribution. It is effective to set the value of κ to be near 1 (e.g., $\kappa = 0.9$), and if κ is small, the function becomes similar to a Gaussian distribution. The associated constraint matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H}(i,j) = \left(1 + \frac{\kappa}{\nu^d - 1}\right) \boldsymbol{H}_1(i,j) - \frac{\kappa}{\nu^d - 1}\boldsymbol{H}_2(i,j), \tag{6.112}$$

where,

$$\boldsymbol{H}_1(i,j) = \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) N(\boldsymbol{x}|\boldsymbol{\mu}, \nu^2\boldsymbol{\Sigma})\mathrm{d}\boldsymbol{x}, \tag{6.113}$$

$$\boldsymbol{H}_2(i,j) = \int_{\mathcal{D}} \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\mathrm{d}\boldsymbol{x}. \tag{6.114}$$

Hence, MAP estimation with (6.112) also provides a novel classifier, which I term the "difference of Gaussians QCMAP" (DOGQCM) classifier.

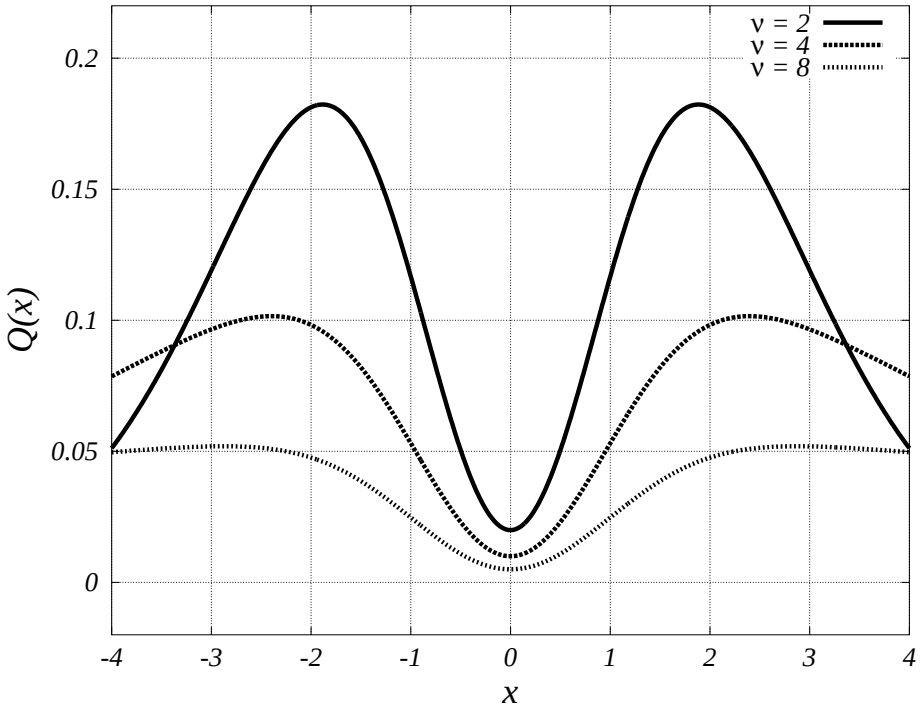


Figure 6.4: Difference of Gaussian function: Here, $\kappa = 0.9$ is fixed and ν varies.

Regularization is not so important in the central area of distribution if the samples are normally distributed because we have a sufficient number of samples. Conversely, regularization is very important in the peripheral area of the distribution because we have only a small number of samples. Therefore, the DOG weight function is effective in such a case.

Weight Function Combinations

I have introduced and proposed a number of weight functions, and calculated their constraint matrices. Thus far we have treated these weight functions individually; however, they can also be used in combination. For example, if we consider two constraints $\boldsymbol{w}^T\boldsymbol{H}_1\boldsymbol{w} \leq 1$ and $\boldsymbol{w}^T\boldsymbol{H}_2\boldsymbol{w} \leq 1$, we can make a new constraint by combining $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$:

$$\boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w}^T = p\boldsymbol{w}^T\boldsymbol{H}_1\boldsymbol{w} + (1-p)\boldsymbol{w}^T\boldsymbol{H}_2\boldsymbol{w} \leq 1,\ 0 \leq p \leq 1. \tag{6.115}$$

In general, when we consider L constraints:

$$\boldsymbol{w}^T\boldsymbol{H}_l\boldsymbol{w} \leq 1,\ l = 1, \ldots, L, \tag{6.116}$$

I unite the constraints as

$$\sum_{l=1}^{L} p_l \boldsymbol{w}^T\boldsymbol{H}_l\boldsymbol{w} \leq 1 \text{ and } \sum_{l=1}^{L} p_l = 1. \tag{6.117}$$

I therefore propose a combination DOGQCM + KDEQCM with the strategies combined equally (i.e., $p = 0.5$).

6.2 Weighted regularization SVM (WRSVM)

In this section, I propose a novel regularization criterion for robust classifiers [110]. The criterion can produce many types of regularization terms by selecting an appropriate weighting function. L2 regularization terms, which are used for support vector machines (SVMs), can be realized by this criterion when the norm of patterns is normalized. In this regard, I propose two novel regularization terms based on the new criterion for a variety of applications. Furthermore, I propose new classifiers by applying these regularization terms to conventional SVMs.

Let a weighting function $Q(\boldsymbol{x})$ satisfy

$$Q(\boldsymbol{z}) > 0 \tag{6.118}$$

for all $\boldsymbol{z} \in \mathcal{D}$, where $\mathcal{D}$ is our data domain. The new regularization criterion is given by

$$R := \int_{\mathcal{D}} Q(\boldsymbol{x}) |\langle \boldsymbol{w}, \boldsymbol{x} \rangle|^2 \mathrm{d}\boldsymbol{x}. \tag{6.119}$$

This regularization term can be rewritten as

$$\int_{\mathcal{D}} Q(\boldsymbol{x}) |\langle \boldsymbol{w}, \boldsymbol{x} \rangle|^2 \mathrm{d}\boldsymbol{x} = \boldsymbol{w}^T H \boldsymbol{w}. \tag{6.120}$$

where $H(i,j) := \int_{\mathcal{D}} Q(\boldsymbol{x}) x_i x_j \mathrm{d}\boldsymbol{x}$ and $H(i,j)$ denotes element (i,j) of the regularization matrix H. Note that H becomes a positive definite matrix from condition (6.118).

Combining our regularization approach with the hinge loss function, I propose a classification criterion whereby

$$\text{minimize} \quad \boldsymbol{w}^T H \boldsymbol{w} + c \sum_{n=1}^{N} \xi_n, \tag{6.121}$$

$$\text{subject to} \quad y_n \left(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b \right) \geq 1 - \xi_n, \tag{6.122}$$

$$\xi_n \geq 0, \quad n = 1, \ldots, N, \tag{6.123}$$

where ξ_n are slack variables. The proposed criterion can produce not only a basic SVM, but also various new classifiers. I demonstrate this relation in the following sections 6.2.1 and 6.2.2. In this regard, I refer to the proposed classifier as "Weighted Regularization SVM" (WR-SVM).

6.2.1 Basic Support Vector Machines

In this section, I demonstrate that our regularization criterion includes the basic regularization term $||\boldsymbol{w}||^2$. In other words, WR-SVM includes basic SVM.

Let us assume that $||\boldsymbol{x}|| = 1$, and $\{x_i, x_j\}(i \neq j)$ are orthogonal. The following assumption holds in the Gaussian kernel model:

$$||\phi(\boldsymbol{x})||^2 = k(\boldsymbol{x}, \boldsymbol{x}) = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}||^2) = 1, \tag{6.124}$$

where $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma||\boldsymbol{x}-\boldsymbol{y}||^2)$ is the Gaussian kernel function. We let $Q(\boldsymbol{x})$ be a constant function: $Q_1(\boldsymbol{x}) := \frac{1}{S}$, where S is the volume of $\mathcal{D}$. Then, the constraint matrix is given by

$$H(i,j) = \frac{1}{S}\int_{\mathcal{D}} x_i x_j \mathrm{d}\boldsymbol{x} \propto \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}. \tag{6.125}$$

We can see that the regularization matrix is defined as $H_1 := I_M$, as well as that it is equivalent to $||\boldsymbol{w}||^2$. Thus, we can regard our regularization method as an extension of the basic regularization term. Also, we can infer that the weighted regularization becomes basic if $Q(\boldsymbol{x})$ is constant (i.e., no weight).

6.2.2 Novel Weighted Regularization

Next, I search for an appropriate weighting function. There are two approaches to this purpose. One is to make $Q(\boldsymbol{x})$ large in a mixed area of categories. Therefore, I define $Q_2(\boldsymbol{x})$ as a normal distribution:

$$Q_2(\boldsymbol{x}) := N(\boldsymbol{x}|\boldsymbol{\mu}, s\Sigma) = \frac{1}{(\sqrt{2\pi})^d\sqrt{|s\Sigma|}} \exp\left(-\frac{1}{2s}(\boldsymbol{x} - \boldsymbol{\mu})\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{6.126}$$

where

$$\boldsymbol{\mu} := \frac{\bar{\boldsymbol{x}}_{+1} + \bar{\boldsymbol{x}}_{-1}}{2}, \quad \Sigma(i,j) := \begin{cases} \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{\mu}(i) - \boldsymbol{x}_n(i))^2 & i = j \\ 0 & i \neq j \end{cases}, \tag{6.127}$$

and $\bar{\boldsymbol{x}}_{+1}$ and $\bar{\boldsymbol{x}}_{-1}$ denote the mean vectors of patterns labeled by $+1$ and -1, respectively. The classifier becomes robust if patterns of different categories are mixed in the central area of the pattern distribution. Furthermore, if we let the parameter s is sufficiently large, this function is similar to a uniform function. Hence, its classifier becomes to the standard SVM.

The other approach is to make $Q(\boldsymbol{x})$ small in dense areas and large in sparse areas. Then, this classifier becomes robust for outliers. Thus, I define a weighting function as the difference between two types of normal distributions:

$$Q_3(\boldsymbol{x}) := \left(1 + \frac{\kappa}{\nu^d - 1}\right) N(\boldsymbol{x}|\boldsymbol{\mu}, \nu^2\Sigma) - \frac{\kappa}{\nu^d - 1} N(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma), \tag{6.128}$$

where $0 < \kappa < 1$ and $\nu > 1$. If we assume that Σ is a diagonal matrix, then this weighting function always satisfies Eq. (6.118). If ν increases, the weighting function becomes smoother and wider. Essentially, κ should be near 1 (e.g., $\kappa = 0.9$), and if κ is small, $Q_3(\boldsymbol{x})$ is similar to $Q_2(\boldsymbol{x})$.

The calculation of these regularization matrices includes integration; however, if we use the Gaussian kernel as a basis function, then we can calculate H analytically since $Q(\boldsymbol{x})$ consists of the Gaussian functions. I present the details of this approach in Section 6.2.3.

6.2.3 Analytical Calculation of Regularization Matrices

I define the regularization matrices H_2 and H_3 as

$$H_t(i,j) = \int_{\mathcal{D}} Q_t(\boldsymbol{x}) k(\boldsymbol{x}_i, \boldsymbol{x}) k(\boldsymbol{x}_j, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \quad t = 2, 3. \tag{6.129}$$

Note that it is only necessary to integrate the product of the normal distribution and two Gaussian kernel functions analytically. Then, I consider only the following integration:

$$U(i,j) = \int_{\mathcal{D}} N(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) k(\boldsymbol{x}_i, \boldsymbol{x}) k(\boldsymbol{x}_j, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{6.130}$$

Using the general formula for a Gaussian integral;

$$\int e^{-\frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}\boldsymbol{x}} \mathrm{d}\boldsymbol{x} = \sqrt{\frac{(2\pi)^M}{|A|}} e^{\frac{1}{2}\boldsymbol{b}^T A^{-1} \boldsymbol{b}}, \tag{6.131}$$

we can calculate Eqs. (6.130) analytically as follows.

$$U(i,j) = \frac{1}{\sqrt{|4\gamma\Sigma + I_N|}} \exp\left(\frac{1}{2}\boldsymbol{b}_{ij}^T A^{-1} \boldsymbol{b}_{ij} + C_{ij}\right), \tag{6.132}$$

$$A = 4\gamma I_N + \Sigma^{-1}, \tag{6.133}$$

$$\boldsymbol{b}_{ij} = 2\gamma(\boldsymbol{x}_i + \boldsymbol{x}_j) + \Sigma^{-1}\boldsymbol{\mu}, \tag{6.134}$$

$$C_{ij} = -\gamma(||\boldsymbol{x}_i||^2 + ||\boldsymbol{x}_j||^2) - \frac{1}{2}\boldsymbol{\mu}^T \Sigma^{-1} \boldsymbol{\mu}. \tag{6.135}$$

H_2 and H_3 can also be calculated in a similar manner.

In practice, the regularization matrix H_t is normalized by $(NH_t)/\mathrm{tr}(H_t)$ so that the constant factor of $\boldsymbol{U}(i,j)$ is independent of multiplication factor.

6.2.4 Novel Classifiers

In this section, I propose novel classifiers by using weighted regularization terms.

I assume that the discriminant function is given by

$$D(\boldsymbol{x}|\boldsymbol{\alpha}, b) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b. \tag{6.136}$$

Then, the training problem is given by

$$\text{minimize} \frac{1}{2}\boldsymbol{\alpha}^T H_t \boldsymbol{\alpha} + c\sum_{n=1}^{N} \xi_n, \tag{6.137}$$

$$\text{subject to} \quad y_n\left(\sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_n) + b\right) \geq 1 - \xi_n, \tag{6.138}$$

$$\xi_n \geq 0, \quad n = 1, \ldots, N. \tag{6.139}$$

I solve this problem by two steps. First, I solve its dual problem:

$$\text{maximize} -\frac{1}{2}\boldsymbol{\beta}^T Y K H_t^{-1} K Y \boldsymbol{\beta} + \sum_{n=1}^{N} \beta_n, \tag{6.140}$$

$$\text{subject to} \quad 0 \leq \beta_n \leq c, \quad \sum_{n=1}^{N} \beta_n y_n = 0, \quad n = 1, \ldots, N, \tag{6.141}$$

where $Y := \text{diag}(\boldsymbol{y})$, $\boldsymbol{\beta}$ is a dual parameter vector, and its solution $\hat{\boldsymbol{\beta}}$ can be obtained by quadratic programming [73]. For the problem, a number of quadratic programming solvers have been developed thus far, such as LOQO [5]. Second, the estimated parameters $\hat{\boldsymbol{\alpha}}$ and $\hat{b}$ are given by

$$\hat{\boldsymbol{\alpha}} = H_t^{-1} K Y \hat{\boldsymbol{\beta}}, \quad \hat{b} = \frac{\mathbf{1}^T \boldsymbol{y} - \boldsymbol{k}^T \hat{\boldsymbol{\alpha}}}{N}. \tag{6.142}$$

Substituting H_2 or H_3 into Eq.(6.140), we can construct two novel classifiers. I denote these classifiers as "SVM_G" and "SVM_D", respectively (based on the initials of Gaussian and Difference).

6.3 Chernoff FDA

Fisher discriminant analysis (FDA) is a popular supervised feature extractor. FDA has many applications in machine learning and pattern recognition [7, 37]. However, FDA does not give an optimal projection when two covariance matrices do not have the same shape even if we assume each category follows a Gaussian distribution (i.e., heteroscedastic case).

In order to handle a heteroscedastic case, it is a key-point how to evaluate a distance between two heteroscedastic Gaussian distributions, and various methods have been studied [30, 93, 118, 67, 89, 70, 69]. Decell et al. [30] proposed to evaluate the distance of distributions by the Kullback-Leibler (KL) divergence, and Tao et al. [93] generalized the KL divergence based FDA by using Bregman divergence and generalized average. Bhattacharyya and Chernoff distances have been also used to evaluate the distance of distributions for a heteroscedastic case in many studies [118, 67, 89, 70]. Zhang et al. [118] proposed to maximize Fisher and Bhattacharyya criteria for binary- and multi-class problems via a Fukunaga-Koontz transform. Loog et al. [67] proposed to use an approximated criterion for the Chernoff distance and Rueda et al. [89] developed a gradient-based algorithm to maximize the Chernoff distance. The Matusita separability measure [70] and a sufficient statistic [69] also have been used.

On the other hand, there are several approaches to extend the objects of the FDA to non-Gaussian distributions. These approaches include non-linear methods using a quadratic polynomial model [91] and a kernel trick [3, 72, 12], Gaussian mixture model based methods [44, 119], and an extended FDA based on auto-correlation matrices [90].

In this section, I propose a new algorithm to maximize the Bhattacharyya distance as an extension of FDA [112, 108]. The Bhattacharyya distance provides a new criterion which consists of the conventional FDA criterion and an additional term. This additional term can be regarded as a correction term for the heteroscedastic Gaussian case since the Bhattacharyya distance is closely related

to misclassification rate via the Hellinger distance. Moreover, the Bhattacharyya distance is a special case of the Chernoff distance. Hence, I propose to use the Chernoff distance as an extension of the Bhattacharyya distance. I call these methods by "Bhattacharyya FDA" (BFDA) and "Chernoff FDA" (CFDA). BFDA is applicable to heteroscedastic Gauss distributions, and CFDA is robust against unbalanced training sample size. Furthermore, I propose their kernelized versions and conduct experiments to evaluate their abilities as binary classifiers. These kernelized classifiers work well even when given samples follow non-Gaussian distributions.

Our approach is related to the prior works [67, 89] because they also tried to maximize the Chernoff distance. However, the former did not provide the strict solution for maximizing the Chernoff distance. Although the latter proposed a strict solution algorithm for maximizing the Chernoff distance, the gradient-based algorithm was not stable. Furthermore, both works did not discuss a hyper-parameter selection for the Chernoff distance carefully just by substituting a prior probability for the hyper-parameter. The contributions and advantages of our research are to propose a new stable algorithm for maximizing the Chernoff distance, to propose several methods of hyper-parameter estimation, and to provide their kernelized versions as applied to the non-Gaussian case.

6.3.1 Bhattacharyya and Chernoff Distances

In this section, I show the theoretical relationships between misclassification rate and the Bhattacharyya distance, when the class distributions are heteroscedastic Gaussian.

First, I describe the definition of misclassification rate. Suppose we classify $\boldsymbol{x}$ according to the rule of (4.86) and the prior probabilities of class 1 and 2 are the same (i.e., $p_1 = p_2 = 0.5$), the misclassification rate is defined by

$$\frac{1}{2}\int_{y|f_1(y)<f_2(y)} f_1(y)\mathrm{d}y + \frac{1}{2}\int_{y|f_1(y)>f_2(y)} f_2(y)\mathrm{d}y \tag{6.143}$$

The minimization of (6.143) is closely related to the maximization of the Hellinger distance [65] that is a measure of distance between two probability density functions:

$$\begin{aligned} h(f_1, f_2) :&= \frac{1}{2}\int_{-\infty}^{\infty}\left(\sqrt{f_1(y)} - \sqrt{f_2(y)}\right)^2 \mathrm{d}y \\ &= 1 - \int_{-\infty}^{\infty}\sqrt{f_1(y)f_2(y)}\mathrm{d}y. \end{aligned} \tag{6.144}$$

Figure 6.5 illustrates the conceptual sketch of relationship between misclassification rate and the Hellinger distance. We have

$$\left(\sqrt{f_1(y)} - \sqrt{f_2(y)}\right)^2 \simeq \begin{cases} f_1(y) & f_1(y) \gg f_2(y) \\ f_2(y) & f_2(y) \ll f_2(y) \end{cases}. \tag{6.145}$$

Although the maximization of the Hellinger distance and the minimization of misclassification rate are not equivalent, we can know from (6.145) that the former can be used as an approximation of the latter.

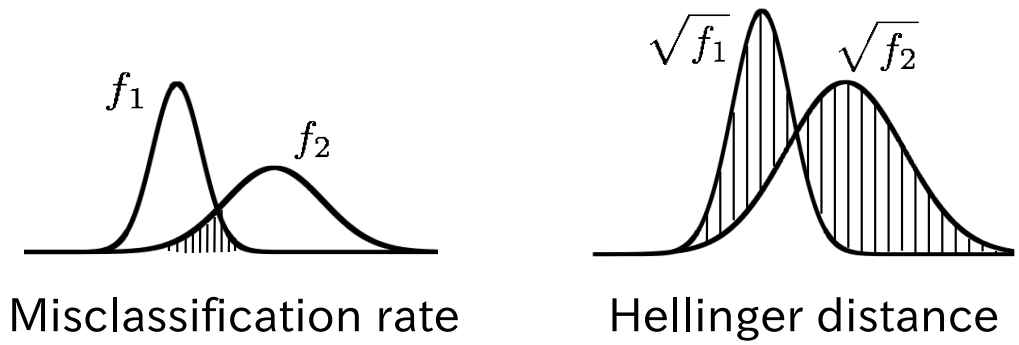


Figure 6.5: Misclassification rate and Hellinger distance

The second term of (6.144) is called the Bhattacharyya coefficient. The Bhattacharyya distance is also defined by

$$B_h(f_1, f_2) := -\log \int_{-\infty}^{\infty} \sqrt{f_1(y) f_2(y)} \mathrm{d}y. \tag{6.146}$$

The maximization of the Bhattacharyya distance and that of the Hellinger distance are clearly equivalent. Therefore, it is effective to use the Bhattacharyya distance as a criterion of supervised feature extraction for classification.

In addition, I introduce the Chernoff distance by

$$C_h(f_1, f_2) := -\log \int_{\infty}^{\infty} f_1(y)^{\rho_1} f_2(y)^{\rho_2} \mathrm{d}y, \tag{6.147}$$

where ρ_1 and ρ_2 are positive weighting parameters, and $\rho_1 + \rho_2 = 1$. The Chernoff distance is an extension of the Bhattacharyya distance since the Chernoff distance with $\rho_1 = \rho_2 = 0.5$ equals to the Bhattacharyya distance. It should be noted that when $\rho_1 < \rho_2$ the weight of $f_1(y)$ is larger than that of $f_2(y)$; the smaller value of ρ yields the higher weight. This fact can be reconfirmed by experiments. Loog et al. suggested that ρ_1 and ρ_2 are estimated by prior probabilities of class 2 and class 1, respectively (i.e. $\rho_1 \leftarrow p_2$ and $\rho_2 \leftarrow p_1$) [67].

Next, I define the probability density function $f_c(y)$ for category c by

$$f_c(y) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left[-\frac{(y - m_c)^2}{2\sigma_c^2}\right] \quad (c = 1, 2). \tag{6.148}$$

Substituting (6.148) into (6.146), I can obtain the following explicit form:

$$\begin{aligned} B_h(f_1, f_2) &= \frac{(m_1 - m_2)^2}{4(\sigma_1^2 + \sigma_2^2)} - \frac{1}{2} \log \frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2} \\ &= \frac{\langle \boldsymbol{w}, \boldsymbol{S}_B \boldsymbol{w} \rangle}{4 \langle \boldsymbol{w}, \boldsymbol{S}_W \boldsymbol{w} \rangle} - \frac{1}{2} \log \frac{2\sqrt{\langle \boldsymbol{w}, \boldsymbol{\Sigma}_1 \boldsymbol{w} \rangle} \sqrt{\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2 \boldsymbol{w} \rangle}}{\langle \boldsymbol{w}, \boldsymbol{S}_W \boldsymbol{w} \rangle}. \end{aligned} \tag{6.149}$$

The first term of (6.149) is equivalent to the FDA criterion. If $\Sigma_1 = \beta\Sigma_2$ for any positive β, the second

term of (6.149) is given by

$$\log \frac{2\sqrt{\langle \boldsymbol{w}, \beta\boldsymbol{\Sigma}_2\boldsymbol{w}\rangle}\sqrt{\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle}}{\langle \boldsymbol{w}, (\beta\boldsymbol{\Sigma}_2 + \boldsymbol{\Sigma}_2)\boldsymbol{w}\rangle} = \log \frac{2\sqrt{\beta}\,\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle}{(1+\beta)\,\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle} = \log \frac{2\sqrt{\beta}}{(1+\beta)}. \tag{6.150}$$

Because this is constant, the criterion does not depend on the second term in such a case. Since most of classification problems will violate the assumption, I must say that the FDA criterion is not enough for general-purpose classification.

Substituting (6.148) into (6.147), the Chernoff version is obtained as

$$\begin{aligned} C_h(f_1, f_2) &= \frac{\rho_1\rho_2(m_1 - m_2)^2}{2(\rho_2\sigma_1^2 + \rho_1\sigma_2^2)} - \frac{1}{2}\log \frac{\sigma_1^{2\rho_2}\sigma_2^{2\rho_1}}{\rho_2\sigma_1^2 + \rho_1\sigma_2^2} \\ &= \frac{\rho_1\rho_2\,\langle \boldsymbol{w}, \boldsymbol{S}_B\boldsymbol{w}\rangle}{2\,\langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle} - \frac{1}{2}\log \frac{\langle \boldsymbol{w}, \boldsymbol{\Sigma}_1\boldsymbol{w}\rangle^{\rho_2}\,\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle^{\rho_1}}{\langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle}, \end{aligned} \tag{6.151}$$

where $\boldsymbol{S}_C := \rho_2\Sigma_1 + \rho_1\Sigma_2$. Note that (6.151) is equal to (6.149) at $\rho_1 = \rho_2 = 0.5$.

6.3.2 Chernoff FDA

According to the discussion in Section 6.3.1, I propose to maximize (6.149) and (6.151) as new criteria. The second term will play an important role in many actual classification problems as a correction term of FDA. Since the Chernoff distance includes the Bhattacharyya distance, I discuss only CFDA. As an important property, (6.151) is invariant with respect to the scale of $\boldsymbol{w}$. Then, I can introduce a constraint that $\langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle = 1$, and a new criterion for CFDA is given by

$$\begin{aligned} \max_{\boldsymbol{w}}\ C_h(f_1, f_2) = \rho_1\rho_2\,\langle \boldsymbol{w}, \boldsymbol{S}_B\boldsymbol{w}\rangle &- \log\langle \boldsymbol{w}, \boldsymbol{\Sigma}_1\boldsymbol{w}\rangle^{\rho_2} \\ &- \log\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle^{\rho_1}, \\ \text{s.t.}\ \ \langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle &= 1. \end{aligned} \tag{6.152}$$

When $\rho_1 = \rho_2 = 0.5$, (6.152) is the criterion of BFDA.

Optimization Algorithm

In this section, I explain an optimization algorithm for the CFDA criterion. From the CFDA criterion (6.152), its Lagrangian is given by

$$\begin{aligned} L(\boldsymbol{w}) =& \rho_1\rho_2\,\langle \boldsymbol{w}, \boldsymbol{S}_B\boldsymbol{w}\rangle - \rho_2\log\langle \boldsymbol{w}, \boldsymbol{\Sigma}_1\boldsymbol{w}\rangle \\ &- \rho_1\log\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle - \lambda(\langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle - 1), \end{aligned} \tag{6.153}$$

where λ is a Lagrangian multiplier. Then, Lagrange's conditions are given by

$$\left[\boldsymbol{S}_B - \frac{\boldsymbol{\Sigma}_1}{\rho_1\,\langle \boldsymbol{w}, \boldsymbol{\Sigma}_1\boldsymbol{w}\rangle} - \frac{\boldsymbol{\Sigma}_2}{\rho_2\,\langle \boldsymbol{w}, \boldsymbol{\Sigma}_2\boldsymbol{w}\rangle}\right]\boldsymbol{w} = \lambda\boldsymbol{S}_C\boldsymbol{w}, \tag{6.154}$$

$$\langle \boldsymbol{w}, \boldsymbol{S}_C\boldsymbol{w}\rangle = 1. \tag{6.155}$$

Since it is difficult to solve (6.154) directly, I propose an updating and iterating approach. For the k-th iteration, I solve

$$\left[\boldsymbol{S}_B - \frac{\boldsymbol{\Sigma}_1}{\rho_1 \langle \boldsymbol{w}_k, \boldsymbol{\Sigma}_1 \boldsymbol{w}_k \rangle} - \frac{\boldsymbol{\Sigma}_2}{\rho_2 \langle \boldsymbol{w}_k, \boldsymbol{\Sigma}_2 \boldsymbol{w}_k \rangle}\right] \boldsymbol{v}_i = \lambda_i \boldsymbol{S}_C \boldsymbol{v}_i \tag{6.156}$$

as a generalized eigenvalue problem. Then, I obtain a set of eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^d$. Next I normalize individual eigenvectors by

$$\boldsymbol{v}_i \leftarrow \frac{\boldsymbol{v}_i}{\sqrt{\langle \boldsymbol{v}_i, \boldsymbol{S}_C \boldsymbol{v}_i \rangle}}, \text{ for } i = 1, \ldots, d. \tag{6.157}$$

Finally, I update

$$\boldsymbol{w}_{k+1} \leftarrow \underset{\boldsymbol{v}_i}{\operatorname{argmax}} C_h(f_1, f_2) \text{ with } \boldsymbol{v}_i, \tag{6.158}$$

$$k \leftarrow k + 1. \tag{6.159}$$

In this way, $\boldsymbol{w}_{k+1}$ always satisfies equations (6.156) and (6.155), and maximizes the Chernoff distance. When $||\boldsymbol{w}_{k+1} - \boldsymbol{w}_k|| < \varepsilon$, $\boldsymbol{w}_{k+1}$ satisfies both Lagrange's conditions (6.154) and (6.155), the algorithm is finished and output $\boldsymbol{w}_{k+1}$ as a solution of CFDA criterion, where ε is a very small positive tolerance parameter (e.g., $\varepsilon = 10^{-8}$). I summarize this algorithm in Algorithm 29.

Sometimes, numerical calculation of the generalized eigenvalue problem provides a strange solution so that the solution includes a complex number. In this case, it is effective to add some regularization term. For example, $\boldsymbol{\Sigma}_c \leftarrow \boldsymbol{\Sigma}_c + \eta \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix, and η is a regularization parameter.

Algorithm 29 CFDA-algorithm

Input: $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_2$, $\boldsymbol{S}_B$, $\boldsymbol{S}_C$, ρ_1, ρ_2 and tolerance $\varepsilon > 0$.

Initialize: $k = 0$, and $\boldsymbol{w}_1$ so that it satisfies (6.155).

repeat

$k \leftarrow k + 1$,

Obtain the set of eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^d$ by (6.156),

Normalize $\{\boldsymbol{v}_i\}_{i=1}^d$ by (6.157),

Update $\boldsymbol{w}_{k+1} \leftarrow \operatorname{argmax}_{\boldsymbol{v}_i} C_h(f_1, f_2)$ with $\boldsymbol{v}_i$

until $||\boldsymbol{w}_k - \boldsymbol{w}_{k+1}||^2 < \varepsilon$.

Output: $\boldsymbol{w}_k$.

Hyper-parameter Estimation

In this section, I discuss how to estimate ρ_1 and ρ_2 of CFDA. There are two parameters, although I need to estimate only one parameter ρ_1 since ρ_2 is decided as $1 - \rho_1$. I propose two methods to estimate these hyper-parameters.

In the first method, I estimate the parameters based on prior probabilities. Thus, $\rho_1 \leftarrow p_2$ and $\rho_2 \leftarrow p_1$. I denote this method by 'CFDA-p'. From the view point of weighting parameter, it is expected as good estimators.

In the second method, I estimate the parameters based on the Chernoff distance by

$$(\rho_1^*, \rho_2^*) = \underset{\rho_1,\rho_2}{\operatorname{argmax}} C_h(f_1, f_2), \text{ s.t. } \rho_1 + \rho_2 = 1. \tag{6.160}$$

However, this optimization problem is not convex. Therefore, I use a simple one dimensional search. First I consider a set of candidates of parameters. The number of candidates can be selected to few or much. Note that too few or much number of candidates are not so effective in this method. I propose to use $P = \{0.1,\ 0.2,\ \ldots,\ 0.8,\ 0.9\}$ as the set of candidates. Thus, I estimate the weighting parameters from these candidates by

$$(\rho_1^*, \rho_2^*) = \underset{\rho_1,\rho_2 \in P}{\operatorname{argmax}} C_h(f_1, f_2), \text{ s.t. } \rho_1 + \rho_2 = 1. \tag{6.161}$$

This solution can be obtained by nine iterations of CFDA-algorithm. I denote this method by 'CFDA-m'. I summarize the CFDA-m-algorithm in Algorithm 30.

Algorithm 30 CFDA-m-algorithm

Input: $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_2$, $\boldsymbol{S}_B$, $\boldsymbol{S}_C$, and tolerance $\varepsilon > 0$.

for $k = 1, 2, \cdots, 9$ **do**

$\rho_1 \leftarrow \frac{k}{10}$,

$\rho_2 \leftarrow 1 - \rho_1$,

$\boldsymbol{w}_k$ is obtained by CFDA-algorithm with (ρ_1, ρ_2),

$J_k \leftarrow C_h(f_1, f_2)$ with $(\boldsymbol{w}_k, \rho_1, \rho_2)$.

end for

$k^* \leftarrow \operatorname{argmax}_k J_k$.

Output: $\boldsymbol{w}_{k^*}$.

6.3.3 Kernel Extension of CFDA

In this section, I kernelize the CFDA criterion. Kernel method is to map from an original space to a high dimensional feature space by $\boldsymbol{\phi}(\boldsymbol{x})$, and perform some linear analysis method in the high dimensional feature space. In this regard, I do not calculate $\boldsymbol{\phi}(\boldsymbol{x})$ directly, but I need to calculate only an inner product $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y}) \rangle$. In this paper, I use the Gaussian RBF kernel function as $k(\boldsymbol{x}, \boldsymbol{y}) := \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{y}||^2)$. In a similar way of the kernel FDA [72, 3], CFDA can be also extended to a kernel classifier.

In this method, $\boldsymbol{w}$ is given by

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = \boldsymbol{\Phi}\boldsymbol{\alpha}, \tag{6.162}$$

where $\boldsymbol{\Phi} := [\boldsymbol{\phi}(\boldsymbol{x}_1) \cdots \boldsymbol{\phi}(\boldsymbol{x}_N)]$, $\boldsymbol{\alpha} := [\alpha_1, \cdots, \alpha_N]^T$, and N is the number of samples. Then, I define

$$\boldsymbol{k}(\boldsymbol{x}) := \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}) \\ \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}) \end{pmatrix} = \boldsymbol{\Phi}^T \boldsymbol{\phi}(\boldsymbol{x}), \tag{6.163}$$

$$\boldsymbol{k}_c := \frac{1}{N_c} \sum_{\boldsymbol{x}_n \in \Omega_c} \boldsymbol{k}(\boldsymbol{x}_n), \tag{6.164}$$

$$\boldsymbol{R}_c := \frac{1}{N_c} \sum_{\boldsymbol{x} \in \Omega_c} (\boldsymbol{k}(\boldsymbol{x}_n) - \boldsymbol{k}_c)(\boldsymbol{k}(\boldsymbol{x}_n) - \boldsymbol{k}_c)^T, \tag{6.165}$$

where N_c is the number of samples belonging to class c. Projected one-dimensional vector is given by

$$y = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{k}(\boldsymbol{x}). \tag{6.166}$$

Average and variance in the projected space are given by

$$m_c = \boldsymbol{\alpha}^T \boldsymbol{k}_c, \tag{6.167}$$

$$\sigma_c^2 = \boldsymbol{\alpha}^T \boldsymbol{R}_c \boldsymbol{\alpha}. \tag{6.168}$$

Finally, the criterion of the 'kernel CFDA' (KCFDA) is given by

$$\begin{aligned} &\max\ \rho_1\rho_2 \left\langle \boldsymbol{\alpha}, \boldsymbol{K}_B \boldsymbol{\alpha} \right\rangle - \rho_2 \log \left\langle \boldsymbol{\alpha}, \boldsymbol{R}_1 \boldsymbol{\alpha} \right\rangle - \rho_1 \log \left\langle \boldsymbol{\alpha}, \boldsymbol{R}_2 \boldsymbol{\alpha} \right\rangle, \\ &\text{s.t.}\ \ \left\langle \boldsymbol{w}, \boldsymbol{K}_C \boldsymbol{w} \right\rangle = 1, \end{aligned} \tag{6.169}$$

where

$$\boldsymbol{K}_B = (\boldsymbol{k}_1 - \boldsymbol{k}_2)(\boldsymbol{k}_1 - \boldsymbol{k}_2)^T, \tag{6.170}$$

$$\boldsymbol{K}_C = \rho_2 \boldsymbol{R}_1 + \rho_1 \boldsymbol{R}_2. \tag{6.171}$$

When $\rho_1 = \rho_2 = 0.5$, I call it the 'kernel BFDA' (KBFDA). In this regard, I do not calculate the feature mapping $\boldsymbol{\phi}(\boldsymbol{x})$ directly, but I need to calculate only the kernel function.

The numerical calculation of KCFDA is given by just replacing $\boldsymbol{x}$ by $\boldsymbol{k}(\boldsymbol{x})$, and applying the CFDA algorithm to obtain $\boldsymbol{\alpha}$. I also propose KCFDA-p and KCFDA-m by applying the hyper-parameter estimation methods.

Chapter 7

Experiments

In this Chapter, I show the experimental results of new feature extraction and classification methods. To evaluate proposed feature extraction methods, I conduct experiments for the low-rank approximation, the face reconstruction, the blind source separation (BSS), the denoising, the parts-based representation, and the compression in Sections 7.1, 7.2, and 7.3. To evaluate proposed classification methods, I conduct experiments of classification with UCI datasets and BCI dataset in Section 7.4.

7.1 fastGRBF based methods

7.1.1 Low-rank approximation

The Yale Face Database

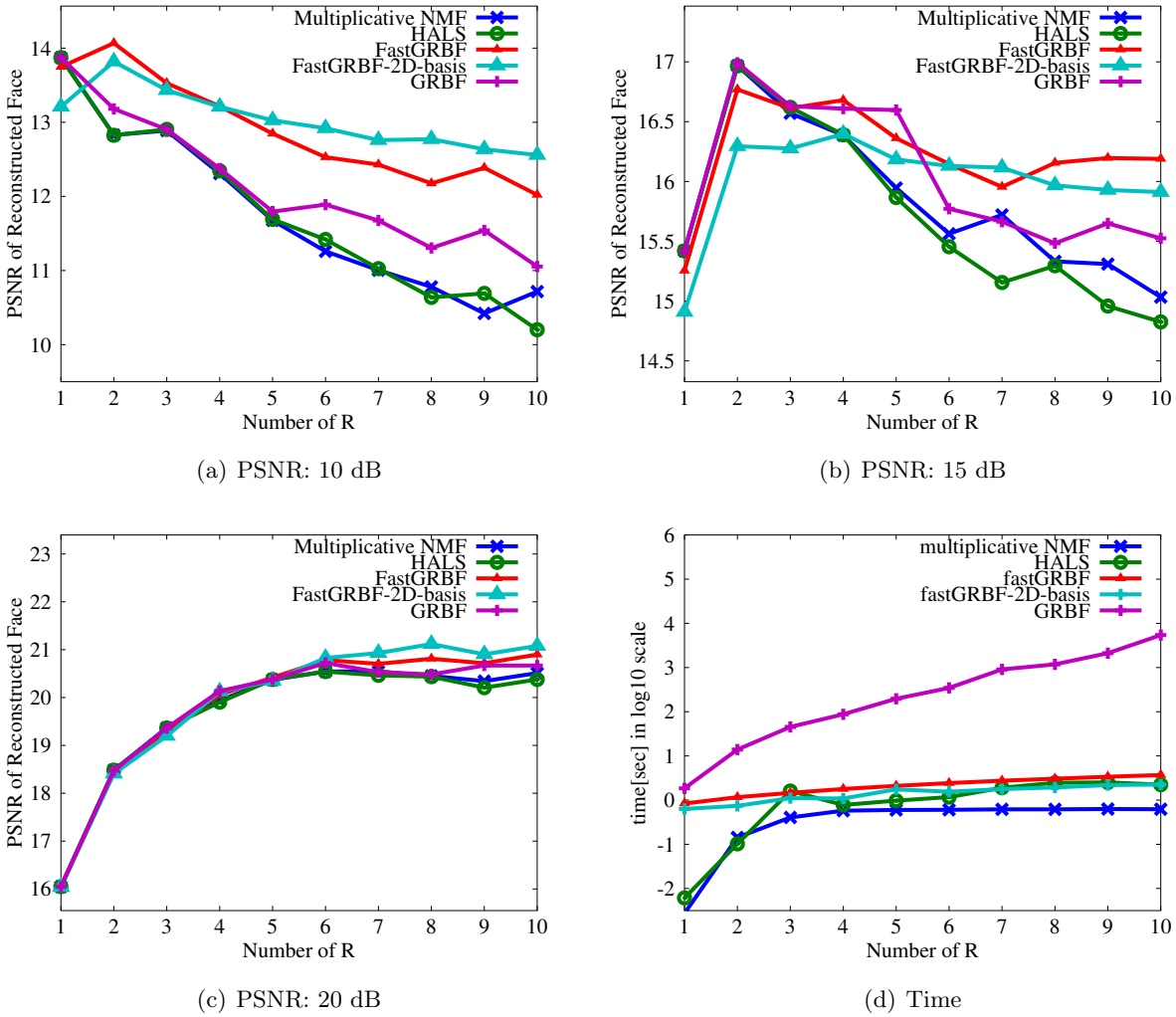
In this experiment, I examine the robustness for the salt-and-pepper noise and computational times of our algorithms. First, I use 10 face images of a subject in the Yale Face Database [4] for denoising experiment. Its images are down-sampled and unfolded to 99 dimensional vectors. Then, an image is converted to a $(99 \times 10)-$matrix. The reason why one image is so small is that the original GRBF algorithm is slow even if I try such a small matrix factorization, and I conduct this experiment to compare the proposed fastGRBF-NMF with the original GRBF. Salt-and-pepper noise was added to the dataset so that PSNRs of datasets are 10, 15, and 20 dB, where PSNR (Peak Signal to Noise Ratio) of $\boldsymbol{Y}$, $\boldsymbol{Y}' \in \mathbb{R}^{I \times J}$ is defined by

$$\mathrm{PSNR}(\boldsymbol{Y}, \boldsymbol{Y}') := 10 \log_{10} \left[\frac{(255^2) IJ}{||\boldsymbol{Y}' - \boldsymbol{Y}||_F^2} \right]. \tag{7.1}$$

For tensors $\underline{\boldsymbol{Y}}$, $\underline{\boldsymbol{Y}}' \in \mathbb{R}^{I \times J \times K}$, it is also defined by

$$\mathrm{PSNR}(\underline{\boldsymbol{Y}}, \underline{\boldsymbol{Y}}') := 10 \log_{10} \left[\frac{(255^2) IJK}{||\underline{\boldsymbol{Y}}' - \underline{\boldsymbol{Y}}||_F^2} \right]. \tag{7.2}$$

I tried to run the HALS-NMF [24], the multiplicative NMF [62], the GRBF-NMF, fastGRBF-NMF, the fastGRBF-NMF-2Dbasis for all PSNRs. Table 7.1 shows the parameter settings for the GRBF



(a) PSNR: 10 dB

(b) PSNR: 15 dB

(c) PSNR: 20 dB

(d) Time

Figure 7.1: PSNR of NMF results on 10 faces for various ranks

and the fastGRBF methods. The degree of freedom of the function approximation in the GRBF is higher than in the fastGRBF since the constraints $\boldsymbol{\Phi W} \geq 0$ for the GRBF is weaker than $\boldsymbol{W} \geq 0$ for the fastGRBF. Therefore, I set the standard deviation parameters of the GRBF to a little larger value than those of the fastGRBF.

Figure 7.1 shows PSNRs and log 10-scale computational times for various rank decompositions. The HALS and the multiplicative algorithms decreased the PSNRs for large R; however, the GRBF and the fastGRBF algorithms kept PSNRs for large R. The fastGRBF performed almost always the best PSNRs in case of 10 dB. The constraint of $\boldsymbol{W} \geq 0$ would work well for noisy data used in the fastGRBF based methods. In Figure 7.1(d), the computational time of the GRBF exponentially increased for large R. On the other hand, the fastGRBF spent a much less to obtain its solution. Table 7.2 describes the average computational times of one decomposition by four algorithms. The fastGRBF is about 300 times faster than the GRBF.

Next, I try to compare the values of $\boldsymbol{W}$ and $\boldsymbol{A}$ between by the proposed fastGRBF-NMF and by

Table 7.1: Parameter setting

PSNR	GRBF	FastGRBF
10	$\Delta t = 1$, $\sigma = 1.2$	$U = 4$, $\delta t = 1$, $\sigma = 1.0$
15	$\Delta t = 1$, $\sigma = 0.8$	$U = 4$, $\delta t = 1$, $\sigma = 0.7$
20	$\Delta t = 1$, $\sigma = 0.5$	$U = 4$, $\delta t = 1$, $\sigma = 0.5$

Table 7.2: Computational times for various algorithms

Algorithms	Multiplicative	HALS	GRBF	FastGRBF
Ave. of times [s]	0.499	0.759	720.0	2.23

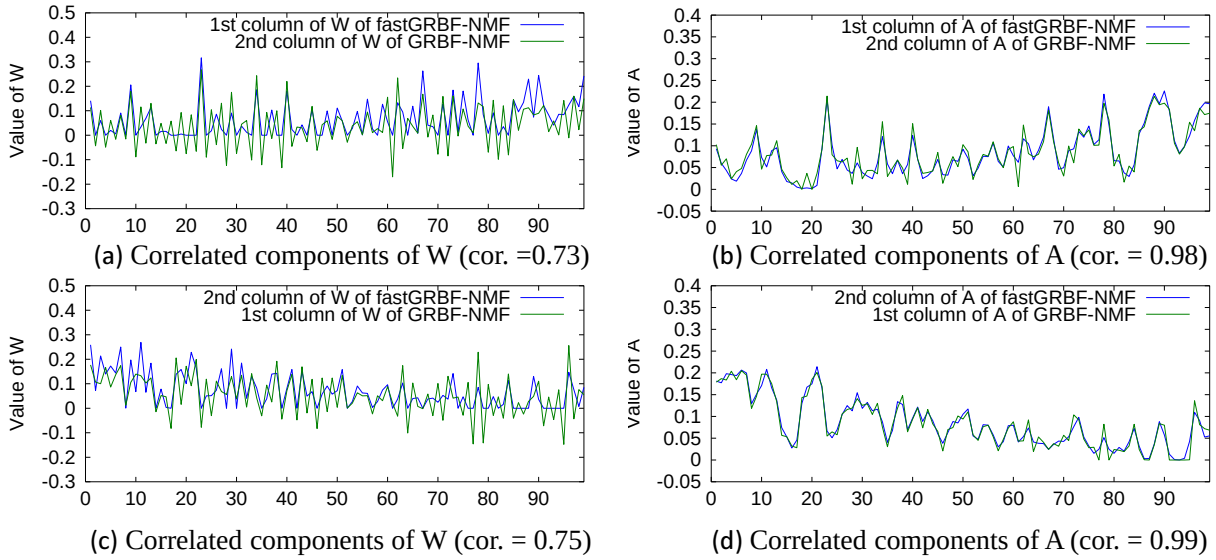
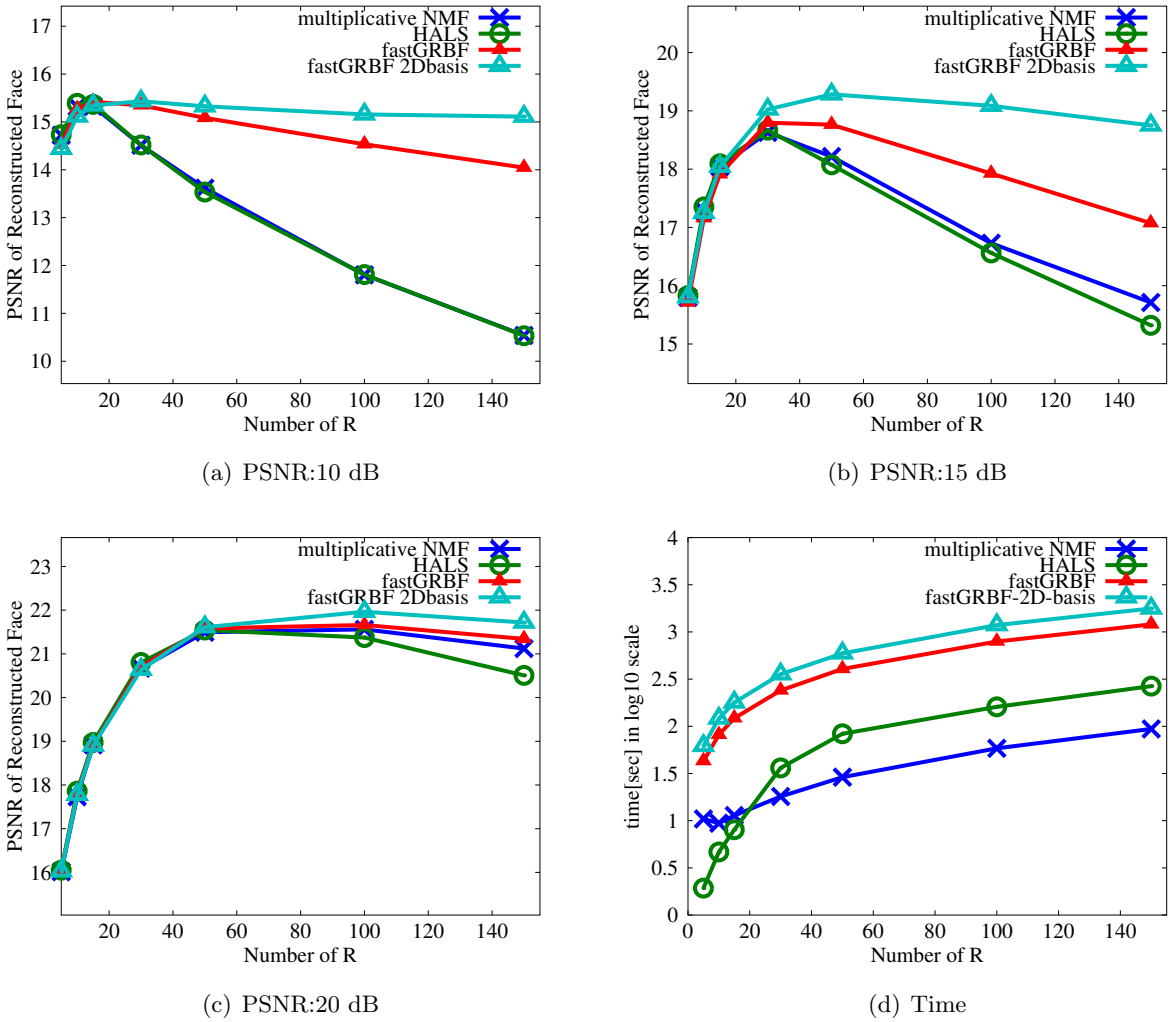


Figure 7.2: Values of $\boldsymbol{W}$ and $\boldsymbol{A}$ obtained by the fastGRBF-NMF and the GRBF-NMF

the original GRBF-NMF. I set the parameters as follows: $U = 1$, $\delta t = 1$, $\sigma = 0.8$, and $R = 2$ for the fastGRBF-NMF, $\Delta t = 1$, $\sigma = 0.8$, and $R = 2$ for the GRBF-NMF. Fig. 7.2 shows the values of $\boldsymbol{W}$ and $\boldsymbol{A}$ obtained by the fastGRBF-NMF and the GRBF-NMF. In spite of the difference between the constraints $\boldsymbol{W} \geq 0$ and $\boldsymbol{\Phi W} \geq 0$, the results of $\boldsymbol{A}$ are highly correlated. Furthermore, the fastGRBF-NMF provides smoother components than the original GRBF-NMF. Many spikes in the result by the GRBF-NMF are smoothed by the fastGRBF-NMF method. Thus, it can be considered that the modification from $\boldsymbol{\Phi W} \geq 0$ to $\boldsymbol{W} \geq 0$ gives smoother features.

Next, I used also the Yale Faces Database including 165 face images of 15 subjects for experiments on NMF and NTF. One face image can be expressed as an 858 dimensional vector. The data matrix is an $(858 \times 165)-$matrix $\boldsymbol{Y}$. The data tensor is an $(858 \times 11 \times 15)$-tensor $\underline{\boldsymbol{Y}}$. The salt-and-pepper noise was added to the dataset so that PSNRs of datasets are 10, 15, and 20 dB. I run NMF and NTF with the HALS, the Multiplicative, the fastGRBF, the fastGRBF-2Dbasis, the ALS-NTD [58], the HALS-NCPD [24], the fastGRBF-NTD, the fastGRBF-NCPD, the fastGRBF-NTD-2Dbasis, and the



(a) PSNR:10 dB

(b) PSNR:15 dB

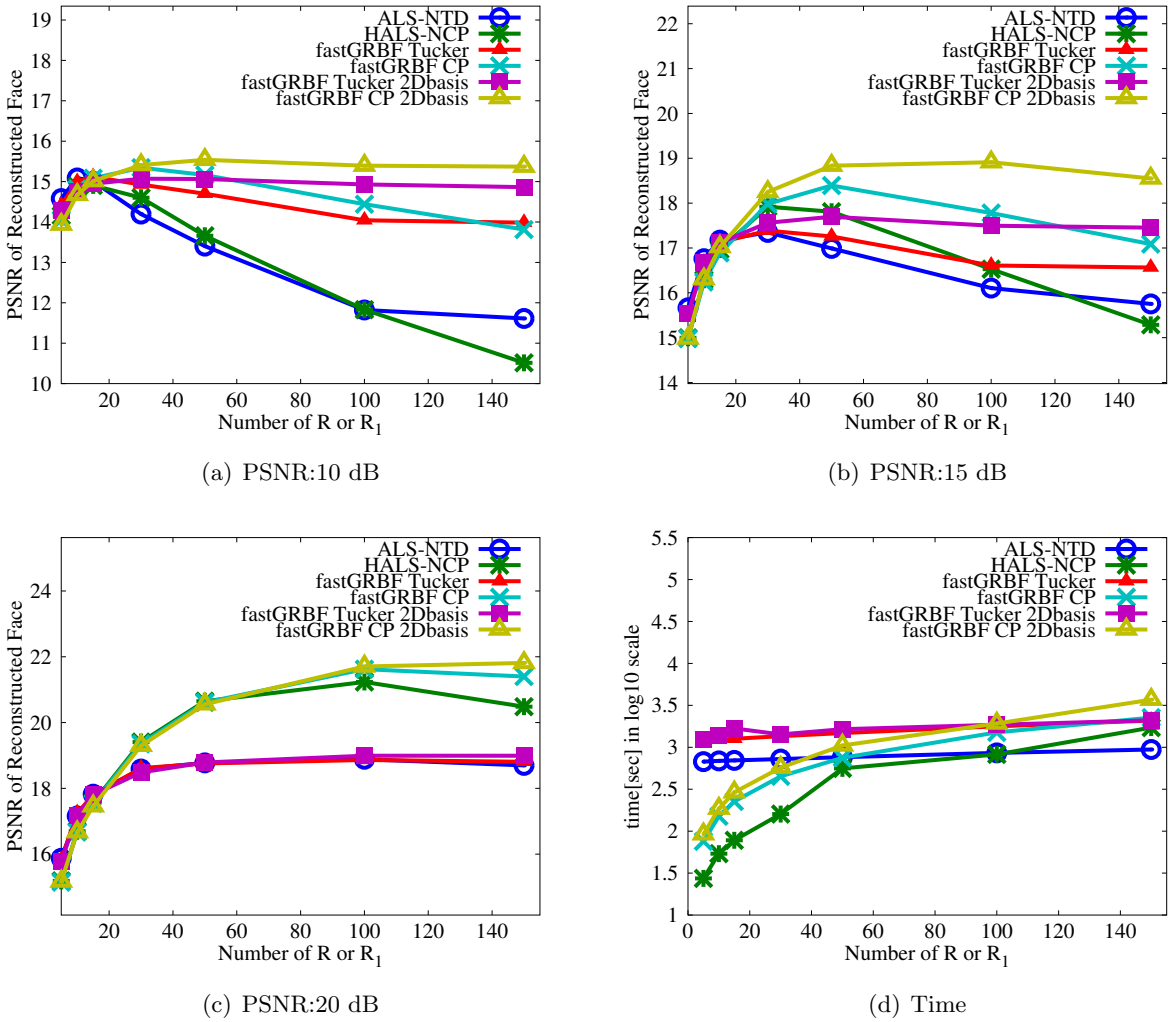
(c) PSNR:20 dB

(d) Time

Figure 7.3: PSNR of NMF results on 165 faces for various ranks

fastGRBF-NCPD-2Dbasis algorithms for all PSNRs. The rank of NMF and CP models was changed from 5 to 150. R_1 was changed from 5 to 150, and $R_2 = R_3 = 10$ were fixed in Tucker model. I set $U = 4$, $\delta t = 1$, $\sigma = 2.0$ for PSNR=10 dB, $\sigma = 1.0$ for PSNR=15 dB, and $\sigma = 0.5$ for PSNR=20 dB in all fastGRBF based methods.

Figures 7.3 and 7.4 show the PSNRs and the log 10-scale computational times for various rank decompositions. The fastGRBF-NMF, the fastGRBF-NTD, and the fastGRBF-NCPD were almost better than traditional algorithms. Furthermore, the fastGRBF-NMF-2Dbasis, the fastGRBF-NTD-2Dbasis, and the fastGRBF-NCPD-2Dbasis were almost better than the fastGRBF-NMF, the fastGRBF-NTD, and the fastGRBF-NCPD algorithms. The computational times of the proposed methods were not shorter than those of the traditional algorithms since the sizes of parameter spaces for the proposed methods are larger. However, they are promising since we can obtain better performances.



(a) PSNR:10 dB

(b) PSNR:15 dB

(c) PSNR:20 dB

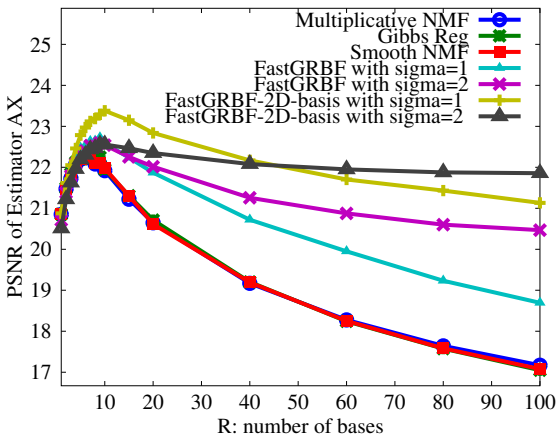
(d) Time

Figure 7.4: PSNR of NTF results on 165 (11×15) faces for various ranks

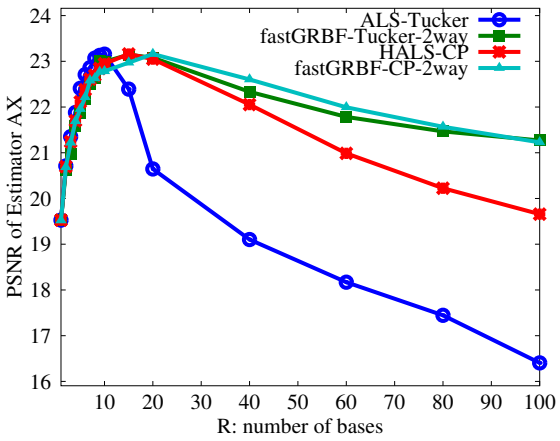
CBCL faces

In this experiment, I examined the robustness of the proposed methods for Gaussian noise and applied to the two-way function approximation for the tensor decomposition using 100 images of the CBCL face dataset[2]. A face image is a 361-dimensional vector or a (19×19)-matrix, and its data matrix is a (361×100)-matrix $\boldsymbol{Y}$ or a $(19 \times 19 \times 100)$-tensor $\underline{\boldsymbol{Y}}$. The Gaussian noise (PSNR=16.2 dB, Fig. 7.5(c)) is added to all images in the dataset. I try to run the NMF with multiplicative updates, the Gibbs Regularized NMF [116, 117], the Essid's Smooth NMF [33], the fastGRBF and the fastGRBF-2Dbasis with $\sigma = 1.0$ and $\sigma = 2.0$ for decompositions of various ranks. To tensor data, I applied the function approximation by two ways as $\boldsymbol{A} = \boldsymbol{\Phi}_a \boldsymbol{W}$ and $\boldsymbol{B} = \boldsymbol{\Phi}_b \boldsymbol{V}$. I denote the methods by fastGRBF-NTD-2way and fastGRBF-NCPD-2way, respectively. The size of the core tensor is changed from

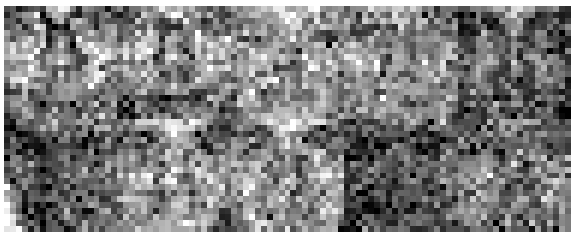
[2]CBCL Face Database #1, MIT Center For Biological and Computation Learning, `http://www.ai.mit.edu/projects/cbcl`



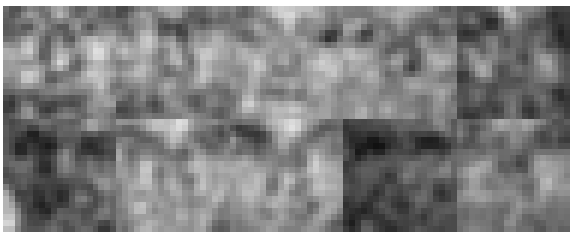
(a) PSNRs of NMF



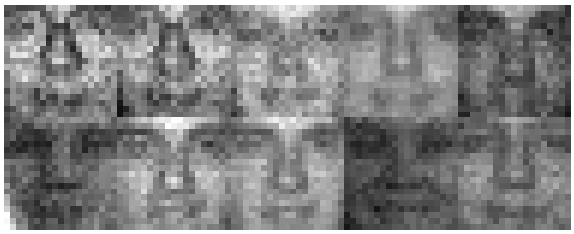
(b) PSNRs of NTF



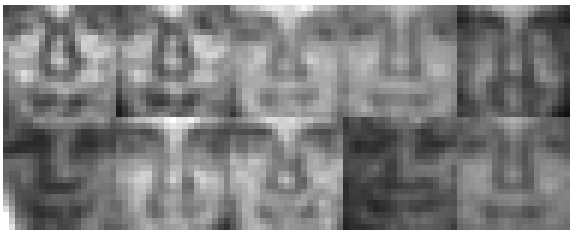
(c) CBCL faces : 60 of 100 images (16.2 dB)



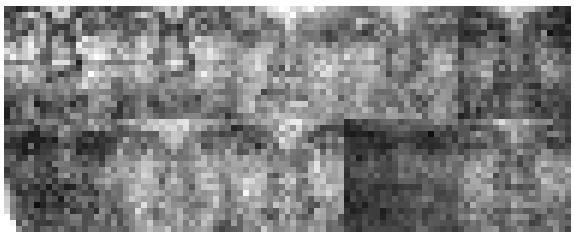
(d) 3x3 moving average filter (22.8 dB)



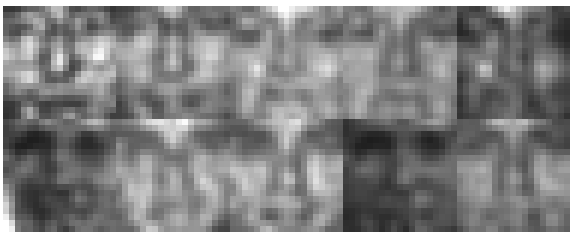
(e) Multiplicative NMF (21.9 dB)



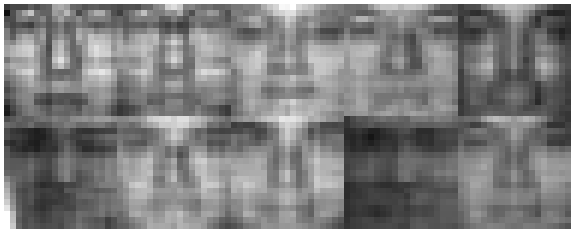
(f) fastGRBF-2Dbasis (23.4 dB)



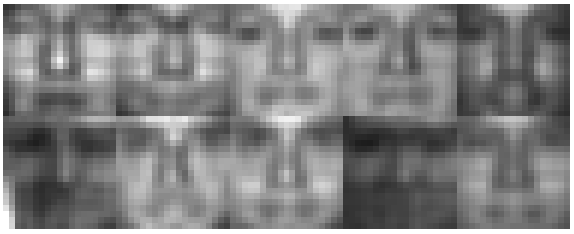
(g) ALS-NTD (20.7 dB)



(h) fastGRBF-NTD-2way (23.2 dB)



(i) HALS-NCPD (23.2 dB)



(j) fastGRBF-NCPD-2way (23.3 dB)

Figure 7.5: Denoising and face reconstruction experiments on CBCL faces database: noise level (PSNR) is 16.2 dB.

$R_1 = R_2 = R_3 = R = 1$ to $R_1 = R_2 = R_3 = R = 100$. I set $U = 4$, $\delta t = 1$ for the fastGRBF-based methods.

Figure 7.5 shows the results of this experiment. We can see that the proposed method provided better results than the traditional and state-of-the-art methods. The fastGRBFs with $\sigma = 2.0$ kept higher PSNRs than with $\sigma = 1.0$ for large R. The two-way function approximation worked well in NTF. Figure 7.5 (d) shows the result by the moving average filter with 3x3 window for reference. Figures 7.5 (e)-(j) show the reconstructed results by the individual methods with $R = 10$ for NMF, and with $R = 20$ for NTF. We can see that the fastGRBF based methods provided smoother images than the traditional methods because the proposed methods are constructed by smooth bases.

In practical cases, we can not measure PSNR between a reconstructed image and an original image since the original image is unknown. Thus, it is difficult to estimate optimal R. However, we can expect that the fastGRBF-based approach gives better results than traditional approaches in a very wide range of R if we assume the images are smooth. This fact is remarkable for practical use in signal processing.

3D-Tensor Data Reconstruction

In this experiment, I applied the fastGRBF-NTD/NCPD-3way to the reconstruction of a noisy 3D-tensor data (Fig. 7.6(c)). The size of the tensor is $(50 \times 50 \times 50)$, and its original data (Fig. 7.6(b)) is generated by a mixture of several 3D-Gaussian functions. Figure 7.6(a) shows the results of SIR (signal to inference ratio) of each estimator for various R. The fastGRBF-NCPD and the fastGRBF-NTD kept high SIRs for large R and Figures 7.6(d)-(g) show the robustness of the two methods for noisy data.

7.1.2 Blind source separation

In this experiment, I applied the fastGRBF to the BSS problem by using artificial and real-world datasets. The generative model is given by

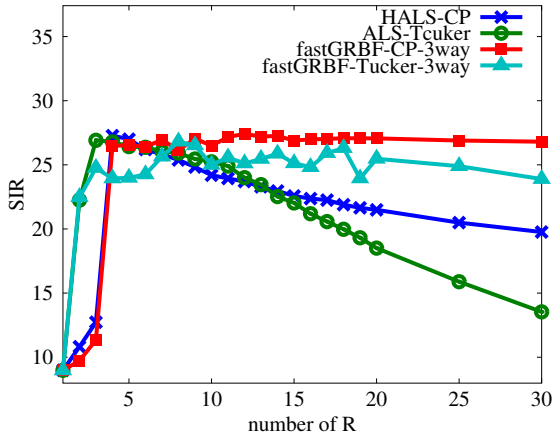
$$\boldsymbol{Y} = [\boldsymbol{S}\boldsymbol{X}_0 + \boldsymbol{E}_0]_+, \tag{7.3}$$

where $\boldsymbol{S} \in \mathbb{R}_+^{I \times R}$ is an original source signal matrix, $\boldsymbol{X}_0 \in \mathbb{R}_+^{R \times J}$ is a mixing matrix, $\boldsymbol{E}_0 \in \mathbb{R}^{I \times J}$ is a Gaussian noise matrix, and $\boldsymbol{Y} \in \mathbb{R}_+^{I \times J}$ is an observed signal matrix. The SNR (Signal to Noise Ratio) is defined by

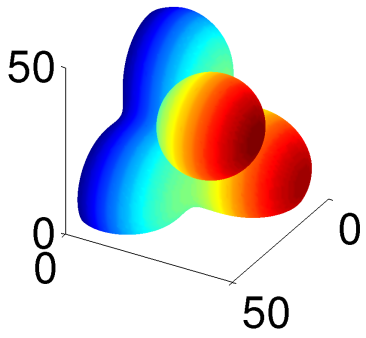
$$\text{SNR} := 10 \log_{10} \left[\frac{||\boldsymbol{S}\boldsymbol{X}_0||_F^2}{||\boldsymbol{S}\boldsymbol{X}_0 - \boldsymbol{Y}||_F^2} \right]. \tag{7.4}$$

I use the SIR (Signal to Inference Ratio) measure to evaluate the results $\boldsymbol{A}\boldsymbol{X}$ obtained by NMF. The SIR is defined by

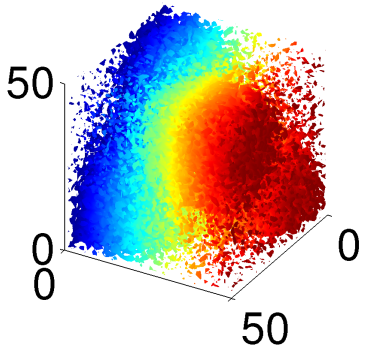
$$\text{SIR} := 10 \log_{10} \left[\frac{||\boldsymbol{S}\boldsymbol{X}_0||_F^2}{||\boldsymbol{S}\boldsymbol{X}_0 - \boldsymbol{A}\boldsymbol{X}||_F^2} \right]. \tag{7.5}$$



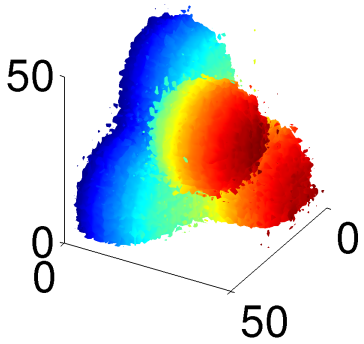
(a) SIR graph



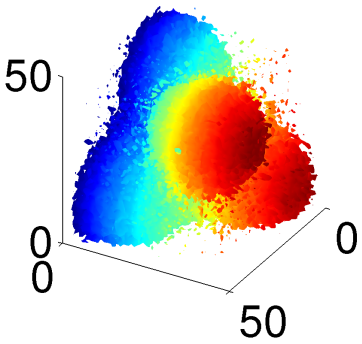
(b) Original



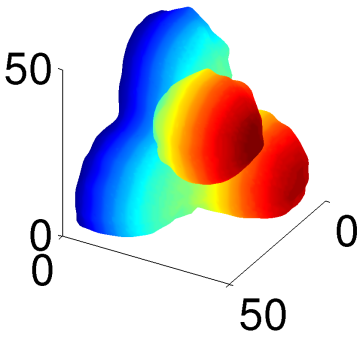
(c) Noisy (7.21 dB)



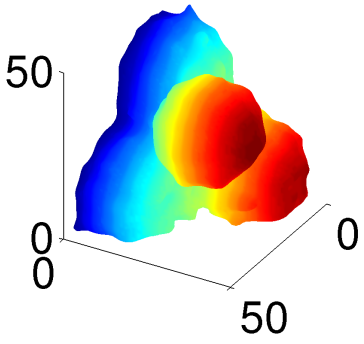
(d) HALS-NCPD (19.8 dB)



(e) ALS-NTD (13.5 dB)



(f) fastGRBF-NCPD-3way (26.8 dB)



(g) fastGRBF-NTD-3way (23.9 dB)

Figure 7.6: Experimental result shown by iso-surfaces at $R = 30$

By comparing the SIR with the SNR, I can obtain information with respect to the reduction of the noise level. Furthermore, I evaluated the estimated source $\boldsymbol{A} = \boldsymbol{\Phi W}$ by SIR_M which is calculated by Algorithm 31. First, the normalization of each signal is necessary because the NMF problem does not have a unique solution. Next, the SIR combination matrix is calculated by $\boldsymbol{M}(r_1, r_2) = \mathrm{SIR}(\boldsymbol{s}_{r_1}, \boldsymbol{a}_{r_2})$. For example, assume that $\boldsymbol{M}$ is given as

$$\boldsymbol{M} = \begin{pmatrix} 3.02 & 0.74 & 4.13 \\ 5.37 & 2.38 & 3.30 \\ 3.51 & 3.54 & 5.58 \end{pmatrix}. \tag{7.6}$$

The maximum element of this matrix is $\boldsymbol{M}(3,3) = 5.58$. Then I set $-\infty$ on the third row and the third column of $\boldsymbol{M}$:

$$\boldsymbol{M} = \begin{pmatrix} 3.02 & 0.74 & -\infty \\ 5.37 & 2.38 & -\infty \\ -\infty & -\infty & -\infty \end{pmatrix}. \tag{7.7}$$

In similar way, the next maximum element is $\boldsymbol{M}(2,1) = 5.37$. Then I set $-\infty$ on the second row and the first column of $\boldsymbol{M}$:

$$\boldsymbol{M} = \begin{pmatrix} -\infty & 0.74 & -\infty \\ -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty \end{pmatrix}. \tag{7.8}$$

Finally, SIR_M is given by $(5.58 + 5.37 + 0.74)/3 = 3.90$. Actually the maximum combination is given by $(5.37 + 4.13 + 3.54)/3 = 4.35$, however we need to calculate $R!$ combinations to obtain this solution. SIR_M gives a useful approximated solution of the combinatorial maximization problem of the mean of SIR for $\boldsymbol{A}$.

First, I use the 'X_5smooth' benchmark from NMFLAB [21]. Figs. 7.7(a) and 7.7(b) show the original sources $\boldsymbol{S} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_R] \in \mathbb{R}_+^{I \times R}$ and the observed signals $\boldsymbol{Y}$ where $R = 5$, $I = 200$, and $J = 20$. Each element of mixing matrix $\boldsymbol{X}_0$ is generated by the absolute value of the Gaussian distribution, randomly. I try to run the traditional and state-of-the-art NMF methods of the multiplicative NMF [62], the Kim & Park's Sparse NMF [57], the Regularized ALS Sparse NMF [22], the Gibbs Regularized NMF [116, 117], the Essid's Smooth NMF [33], and the fastGRBF-NMF to separate blind source signals for various noise levels. The parameters of the fastGRBF were used as $U = 4$, $\delta t = 1$, and $\sigma = 1.0$. Figs. 7.7(c) and 7.7(d) show the result of Fig. 7.7(b) by the multiplicative NMF and the fastGRBF, respectively. Figs. 7.7(e) and 7.7(f) show the result of SIR and SIR_M of all the methods. We can see that the fastGRBF is the robustest to noise in both measures, significantly.

Next, I applied the fastGRBF-NMF to BSS of real world data, far-infrared (FIR) spectra from the database of California Institute of Technology[1] in the same conditions, excluding $\sigma = 4.0$. Fig. 7.8(a) shows FIR spectra of 'Phosgenite', 'Rhodocrosite', and 'Cancrinite'. Fig. 7.8(b) shows observed mixed

[1] `http://minerals.gps.caltech.edu/FILES/Infrared_Far/Index.html`

Algorithm 31 Calculation algorithm of SIR_M

Input: $\boldsymbol{S}$, $\boldsymbol{A} \in \mathbb{R}^{I \times R}$
Initialize: $v = 0$
$\boldsymbol{s}_r \leftarrow \boldsymbol{s}_r / ||\boldsymbol{s}_r||$ for $r = 1, ..., R$;
$\boldsymbol{a}_r \leftarrow \boldsymbol{a}_r / ||\boldsymbol{a}_r||$ for $r = 1, ..., R$;
Calculate $\boldsymbol{M}(r_1, r_2) = 10 \log_{10} \left[\frac{||\boldsymbol{s}_{r_1}||^2}{||\boldsymbol{s}_{r_1} - \boldsymbol{a}_{r_2}||^2} \right]$ for all r_1, r_2;
for $r = 1, ..., R$ **do**
  $[r_1^*, r_2^*] \leftarrow \mathrm{argmax}_{r_1, r_2} \boldsymbol{M}(r_1, r_2)$;
  $v \leftarrow v + \boldsymbol{M}(r_1^*, r_2^*)/R$;
  $\boldsymbol{M}(r_1^*, :) \leftarrow -\infty$;
  $\boldsymbol{M}(:, r_2^*) \leftarrow -\infty$;
end for
Output: v

signals with additive Gaussian noise. Fig. 7.8(c) and Fig. 7.8(d) show the results of estimated sources obtained by the multiplicative NMF and the fastGRBF, respectively. The fastGRBF gave smooth representations of signals, and SIR and SIR_M were better than those by the multiplicative NMF. Fig. 7.8(e) and Fig. 7.8(f) show SIR and SIR_M for various noise levels with various state-of-the-art methods. We can see that the fastGRBF almost always achieved the best performance in comparison with all the methods and for all noise levels.

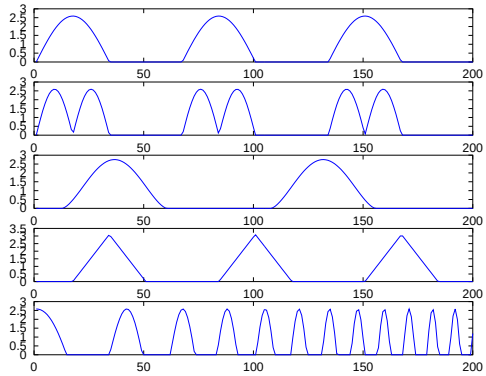
7.1.3 Block-based part representation

Block-wise Image Factorization

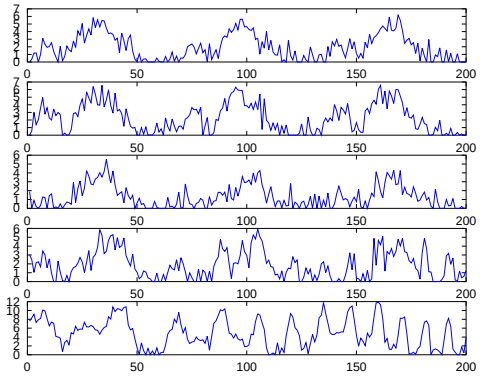
In this section, I apply the fastGRBF-block-NMF to the decomposition of several noise-free (64×64)-image data. I try to run the fastGRBF methods with the parameters settings shown in Table 7.3. Please note that the parameter in the first line implies a block for an image so that the fastGRBF-NMF and the fastGRBF-block-NMF are equivalent. Other parameters are given as $U = 1$, $\delta t = 1$, and $\sigma = 1$. Fig. 7.9 shows the results of this experiment. We can see that the methods with smaller block size gave high PSNRs in a short time, however they did not give a smooth representation. On the other hand, the methods with a larger block size gave smooth representations, however they are very slow. Thus, the block size can be regarded as a trade-off parameter. Furthermore, the block distortion occurs in this method along with the DCT.

Large-scale image analysis

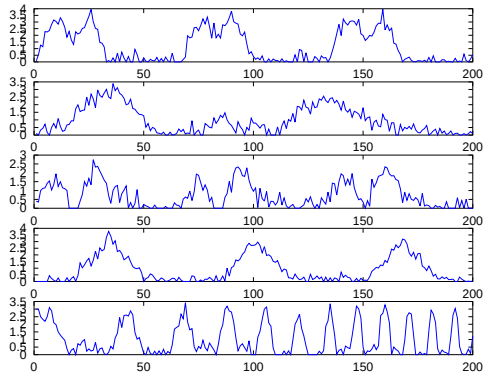
Next I apply the fastGRBF-block-NMF to large-scale image analysis: denoising and local-parts analysis. A noisy (3456×4608)-image of which noise level is 10 dB is used. Image can be transformed as (1024×15552)-nonnegative matrix data by the unfolding of individual (32×32)-blocks. The



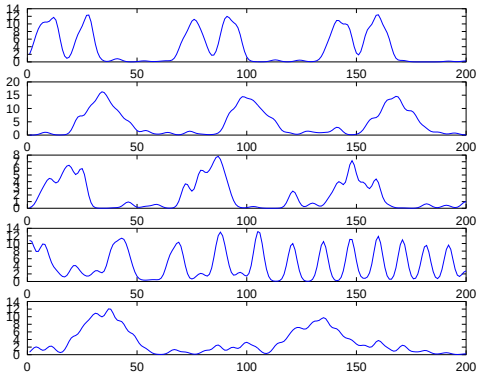
(a) Source signals $\boldsymbol{S}$



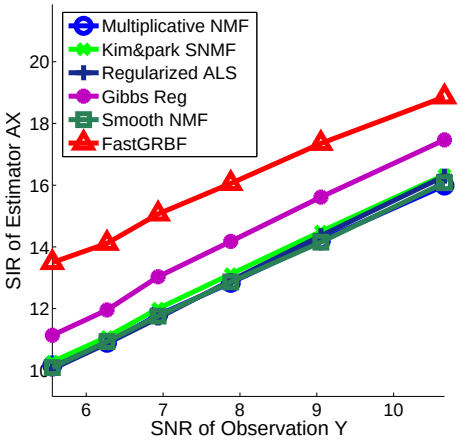
(b) First 5 of 20 observed signals (SNR=12.4 dB)



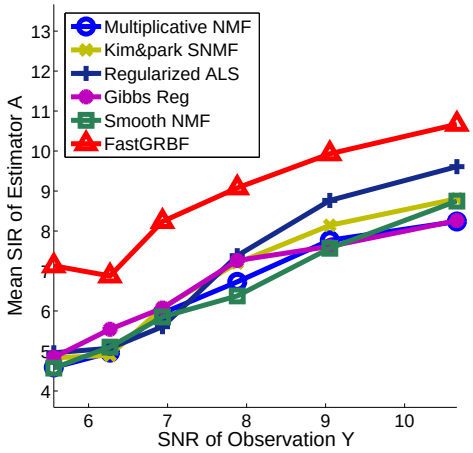
(c) Estimated signals $\boldsymbol{A}$ (Multiplicative NMF, SIR= 17.6 dB, SIR_M = 9.3 dB



(d) Estimated signals $\boldsymbol{A}$ (FastGRBF, SIR= 19.6 dB, SIR_M=13.3 dB)

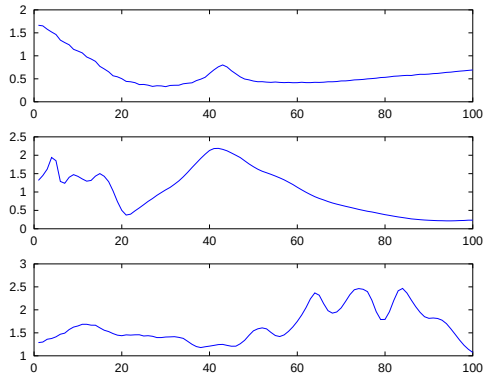


(e) Evaluation with SIR of $\boldsymbol{AX}$

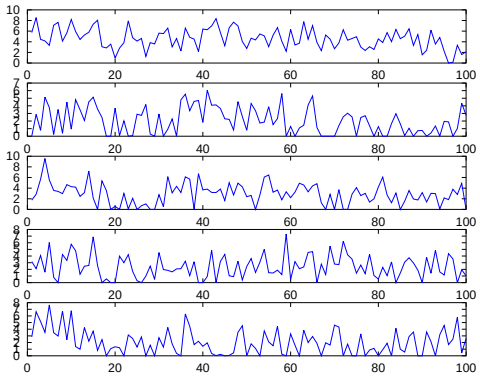


(f) Evaluation with SIR_M of $\boldsymbol{A}$

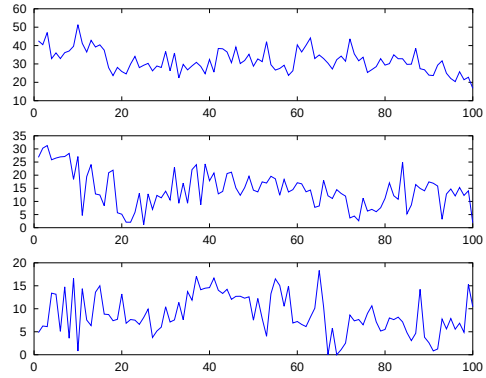
Figure 7.7: Visualizations and experimental results on NMFLAB: 'X_5smooth'



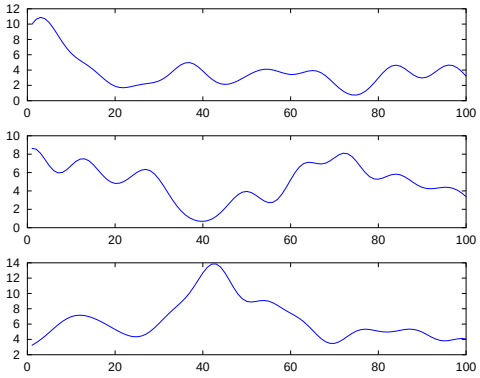
(a) Source signals $\boldsymbol{S}$: Phosgenite (top), Rhodocrosite (middle), Cancrinite (bottom)



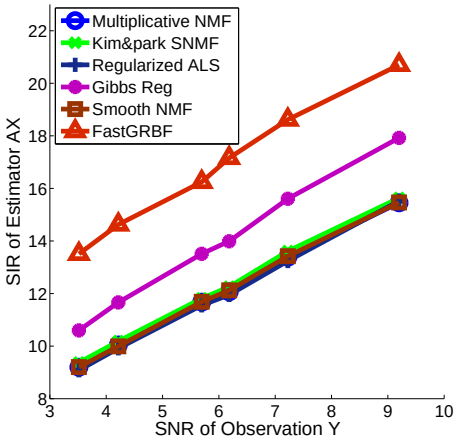
(b) First 5 of 20 observed signals (SNR=4.4 dB)



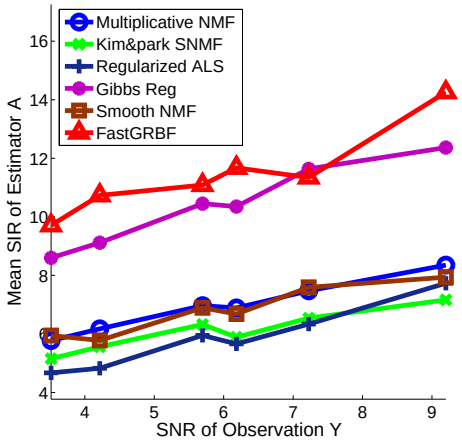
(c) Estimated signals $\boldsymbol{A}$ (Gibbs reg. NMF, SIR= 11.5 dB, SIR_M = 8.7 dB



(d) Estimated signals $\boldsymbol{A}$ (FastGRBF, SIR= 15.1 dB, SIR_M=10.7 dB)



(e) Evaluation with SIR of $\boldsymbol{AX}$

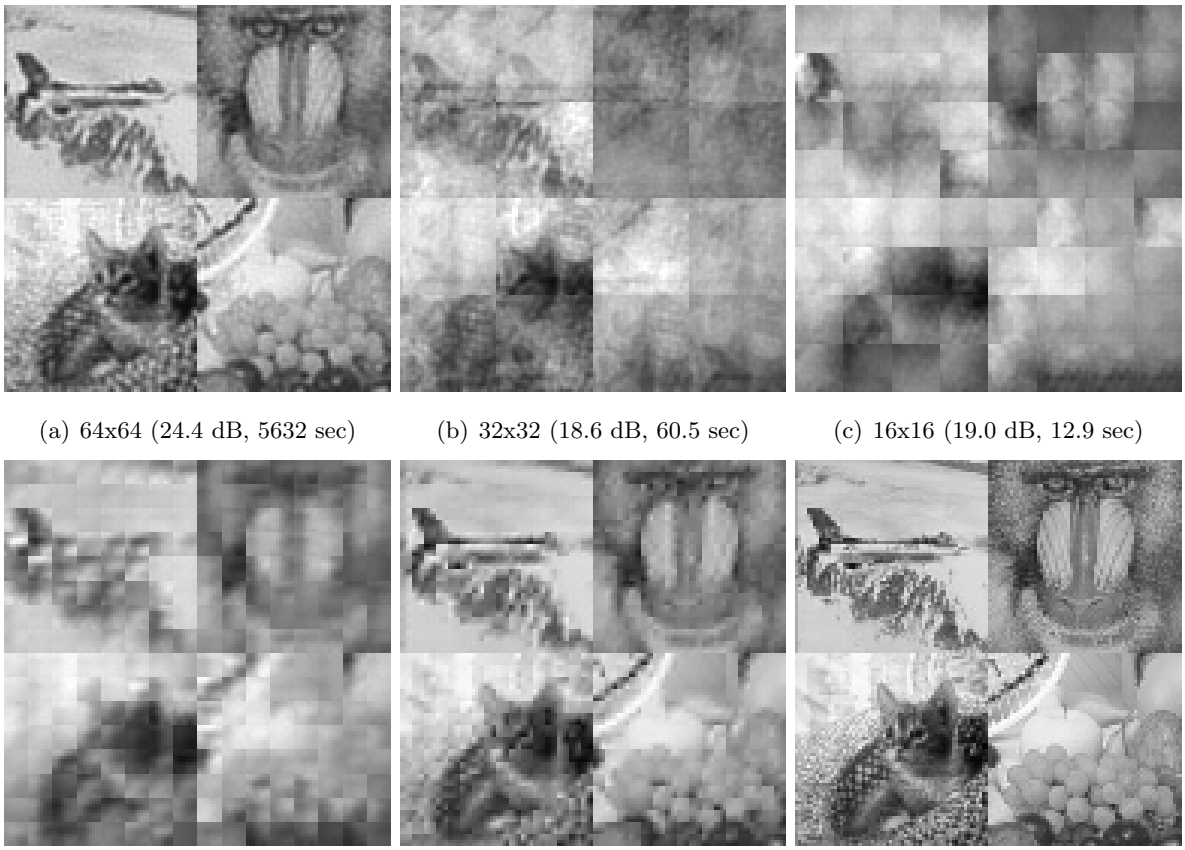


(f) Evaluation with SIR_M of $\boldsymbol{A}$

Figure 7.8: Visualizations and experimental results on far-infrared spectra

Table 7.3: Individual parameters of fastGRBF-block-NMF

block size	# of blocks	# of I	# of R	# of J	# of param.
64×64	1×1	4096	4	4	16640
32×32	2×2	1024	4	16	4160
16×16	4×4	256	4	64	1280
8×8	8×8	64	4	256	1280
4×4	16×16	16	4	1024	4160
2×2	32×32	4	4	4096	16400



(a) 64x64 (24.4 dB, 5632 sec) (b) 32x32 (18.6 dB, 60.5 sec) (c) 16x16 (19.0 dB, 12.9 sec)

(d) 8x8 (20.3 dB, 11.5 sec) (e) 4x4 (22.4 dB, 11.7 sec) (f) 2x2 (28.4 dB, 3.1 sec)

Figure 7.9: Results of fastGRBF-block-NMF

fastGRBF-NMF-2Dbasis is applied to this data, and finally its reconstructed matrix is transformed to a (3456×4608)-image.

Fig. 7.10 shows the reconstruction result. I tried to compare it with results by the moving average filter, the low-pass (butter worth) filter, and the wavelet shrinkage denoising [31]. Individual top and bottom images are zoomed (801×1001)- and (201×401)-images, respectively. Wavelet shrinkage denoising seems to reduce the many noise visually; however, image is blurred and small noise is still remained. In the proposed method, the block distortion can be recognized, clearly. According to the value of PSNR, the low-pass filter is the best method and the stripe pattern of the roof was not destroyed. In this way, the denoising result of the proposed method is not the best; nontheless, the proposed method provided the second best PSNR, and it can be improved by a deblocking filter.

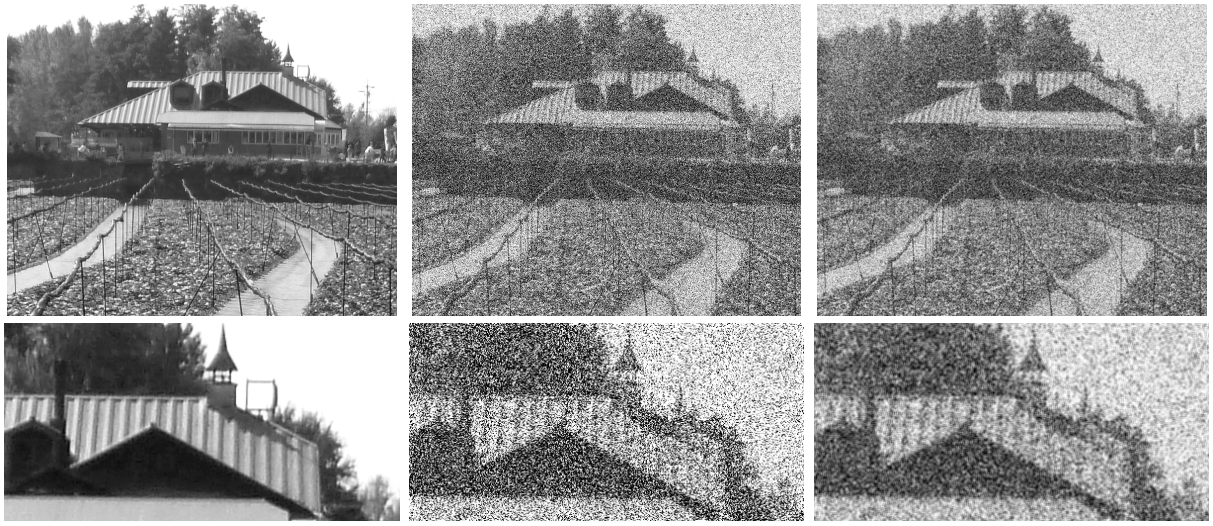
Next I apply the fastGRBF-block-NMF to the local parts analysis of a noisy large-scale image which is the same as the above one. The (1024×15552)-matrix data is generated by the same unfolding method as above one. I extract $R = 20$ parts by the fastGRBF-block-NMF method. Fig. 7.11 shows the result of local parts (reverse image). I tried to compare it with the standard multiplicative NMF [62], the non-smooth NMF (nsNMF) [80], the Chen's smooth NMF [13] and the Gibbs regularized smooth NMF [116, 117]. Only (a) is learned from the original noise free image, and the others (b-f) are learned only from the noisy image (10 dB). We can see that almost all parts includes a lot of noise excluding the nsNMF and the fastGRBF-NMF of which parts are clear. SIRs between the result of the multiplicative NMF from the noise free image and the results of the other methods are (b) 14.36 dB, (c) 16.65 dB, (d) 13.68 dB, (e) 14.42 dB, and (f) 16.99 dB, then the fastGRBF provides the best result among these methods. In fact, a little noise is still remained in the parts of nsNMF. As is well-known, the sparseness is efficiently worked for parts-based representation. The fastGRBF does not imposed any sparse constraint but smoothness by the function approximation, it provide the best result. Thus, the fastGRBF-NMF is promising to combine the smoothness with the sparseness for parts-based representation.

7.2 LCPTD

7.2.1 Low rank approximation

In this part, the LCPTD was applied to face reconstruction problems and the performances were compared with other models. The Yale face database consists of 165 gray-scale images of 15 individuals. There are 11 images per subject with different facial expressions or configurations. In this experiment, I used 15 full-face images; I took an image from each subject. The size of an image is 215×171 pixels, then we considered that $I_1 = 215$, $I_2 = 171$, and $S = 15$. Furthermore, I prepared salt-and-pepper noised data sets: SNR $\in \{5,\ 10,\ 20\}$ [dB].

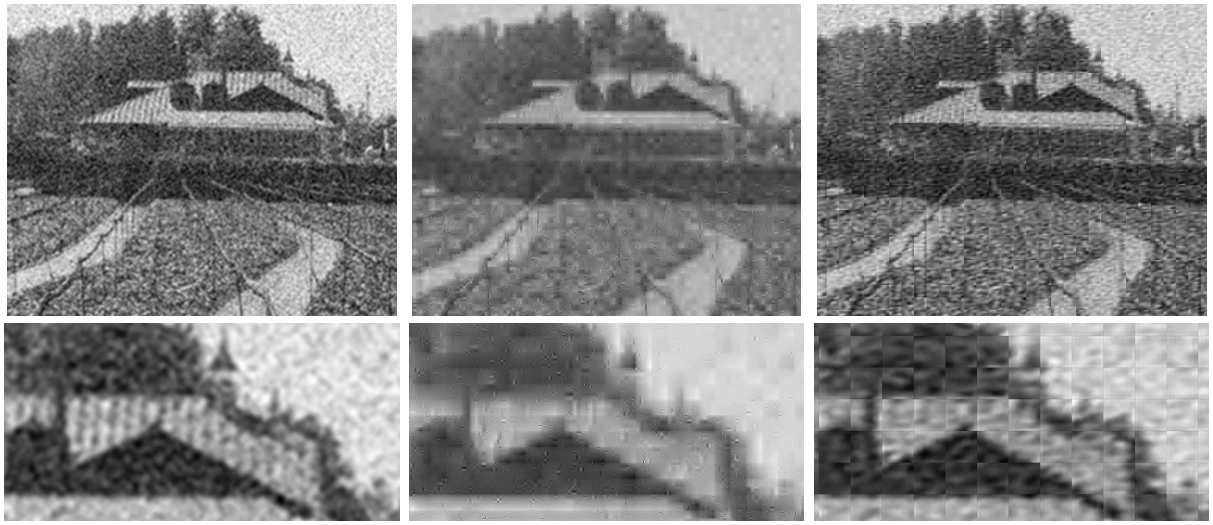
I applied our new LCPTD model, the sparse LCPTD, and the nonnegative LCPTD with various numbers of common components for the noise free and the noisy data sets; the number of bases was fixed as $J = 40$, and the number of common components was changed as $L_n \in \{0,\ 5,\ 10,\ \ldots,\ 40\}$. I



(a) Original

(b) Noisy (10 dB)

(c) 3x3 moving average filter (16.80 dB)



(d) Low-pass filter (18.53dB)

(e) Wavelet Shrinkage Denoising (17.78 dB)

(f) fastGRBF 32x32 block (18.21 dB)

Figure 7.10: Denoising result

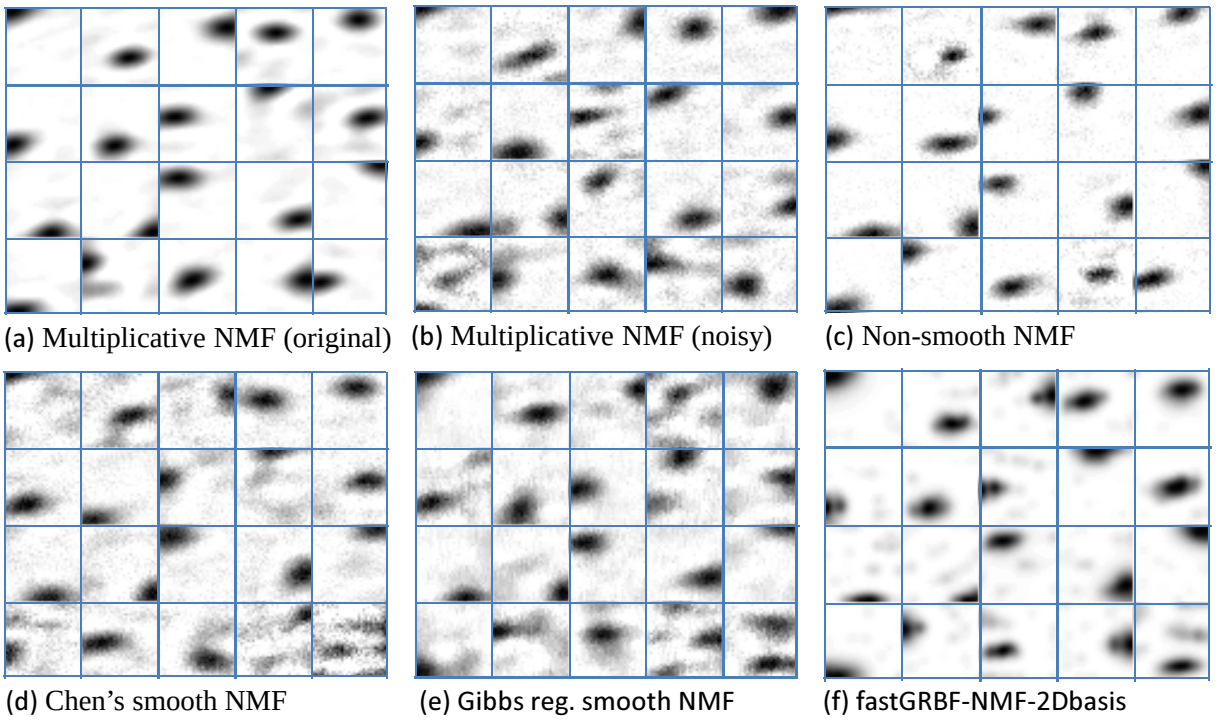


Figure 7.11: Parts-based representation

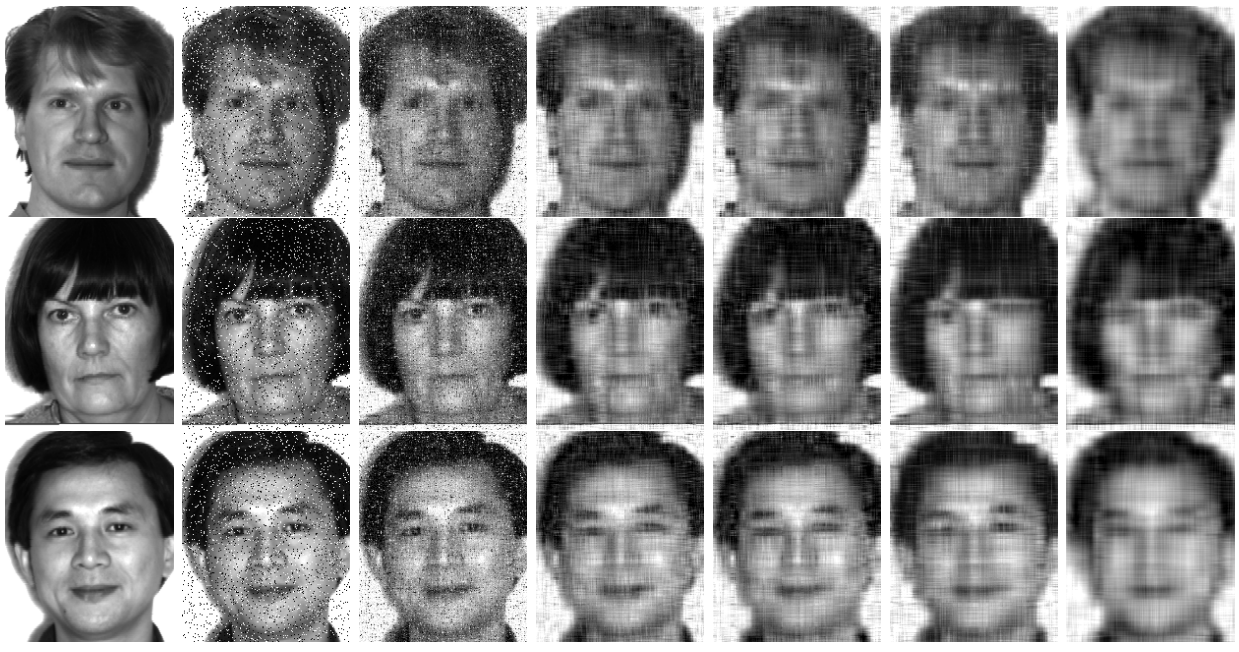
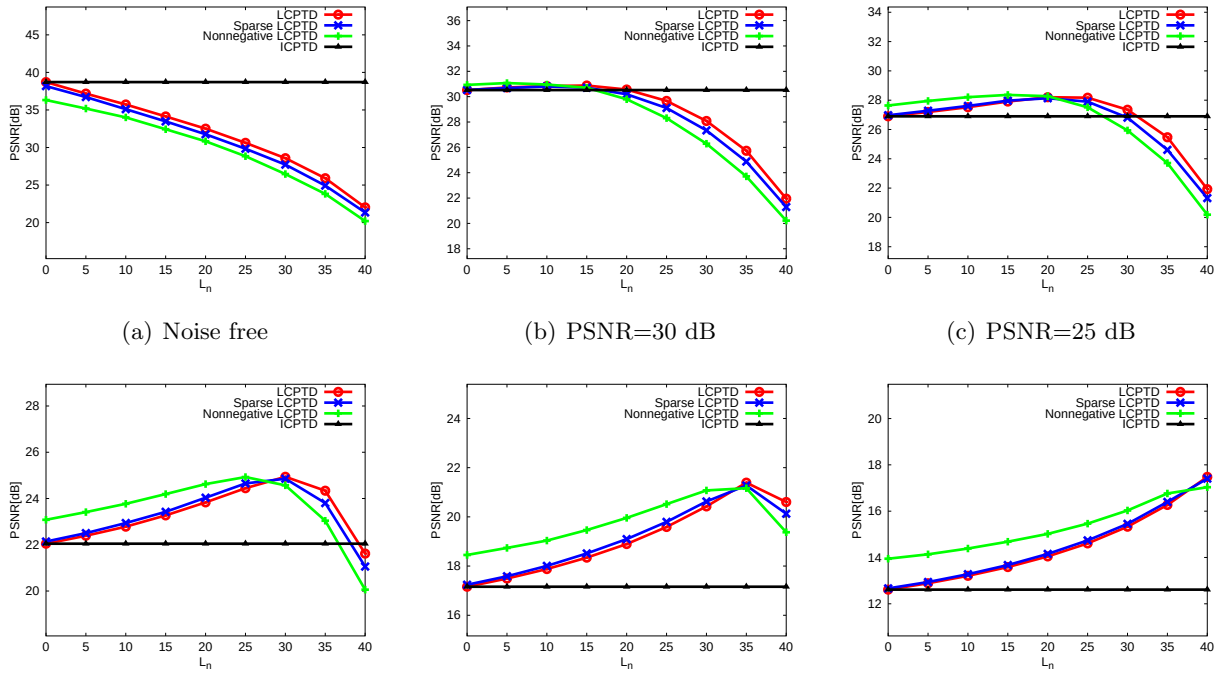


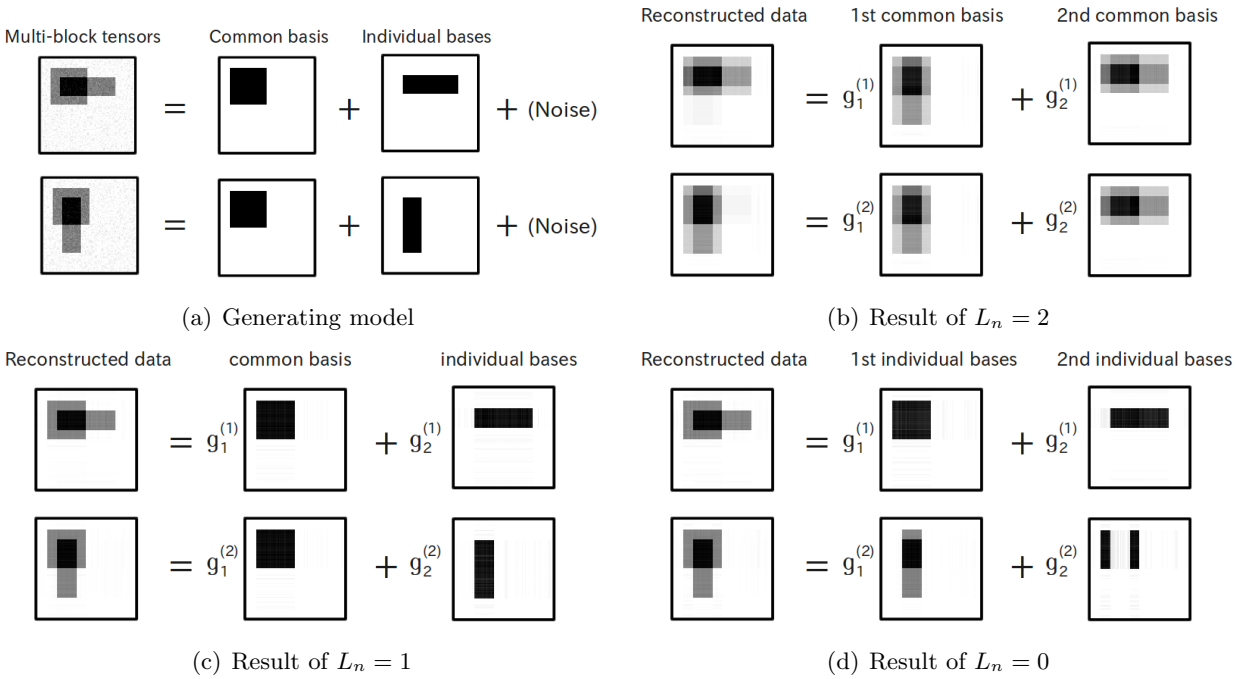
Figure 7.12: Face images corrupted by additive noise and the reconstructed images (PSNR= 15 dB, $J = 40$): 1st column: original images, 2nd column: noisy images, 3rd column: ICPTD model, 4th column: LCPTD ($L_n = 35$), 5th column: LCPTD with sparse constraint ($L_n = 35$), 6th column: LCPTD with nonnegative constraint ($L_n = 35$), 7th column: SCPTD model.



(a) Noise free (b) PSNR=30 dB (c) PSNR=25 dB

(d) PSNR=20 dB (e) PSNR=15 dB (f) PSNR=10 dB

Figure 7.13: PSNRs for various noisy data sets



(a) Generating model (b) Result of $L_n = 2$

(c) Result of $L_n = 1$ (d) Result of $L_n = 0$

Figure 7.14: Linked Multi-block Tensor Factorization

computed the PSNRs between the original faces and the reconstructed faces.

Fig. 7.12 depicts the results of face reconstruction. We can see that the ICPTD model couldn't reduce noise well, and the SCPTD model was robust with respect to noise but reconstructed faces were too fuzzy (distorted). However, the LCPTD based methods gave the appropriate and intermediate reconstructions comparing to the two models.

Fig. 7.13 depicts the graphs of PSNRs with various number of common components. We can see that if the noise level is larger, the maximum points of PSNR is obtained by a larger R. In noise free data set Fig. 7.13 (a), the maximum PSNR was obtained at $L_n = 0$ for all methods; so the ICPTD model is the best in this case. On the other hand, the maximum PSNRs were obtained by the LCPTD based methods in Fig. 7.13 (b,c,d,e). It is also interesting that the nonnegative LCPTD kept a high PSNR for smaller number of common components in comparison with the other methods in high noise level (see Fig. 7.13 (d,e,f)). Thus, we can say that the nonnegative constraint is useful for this problem.

In general, because real data often includes some noise, the proposed method could be very useful and practical for the real tensor data analysis. The higher noise level requires a large number of common components, but the multitude of common components is often only harmful for fitting. Then, we have to select the best parameter of L_n and its selection method is an open problem. We may be able to select L_n depending on PSNR if it is known as prior information.

7.2.2 Linked Blind Source Separation

In this part, I applied the LCPTD to the blind source separation of a simple data for the linked multi-block tensor factorization. I generated two block data tensors consisting of a one common basis factor and two individual basis factors with noise (see Fig. 7.14(a)). And I decompose them by our LCPTD model with nonnegative constraints for various number of common bases $L_n \in \{2, 1, 0\}$ for $n = 1, 2$. Figs. 7.14 (b,c,d) depict the results of this experiment. It is obvious that the result of $L_n = 2$ couldn't represent the original data tensors since its degree of freedom of model is not sufficient. On the other hand, the result of $L_n = 0$ could represent the original data tensors, but each basis is not matched, completely. The result of $L_n = 1$ could represent not only the original data tensors, but also each basis; besides, the additive noise were reduced.

We can see that the LCPTD model can be very useful assuming that some components are common in generating model. The blind source separation can separate into two original sources from two observed signals. It is very interesting that the LCPTD can separate three bases (i.e., a common and two individual bases) from only two observed tensors. We should note that the selection of L_n could be a very important deciding factor to obtain proper decomposition for the LCPTD method from this experiment.

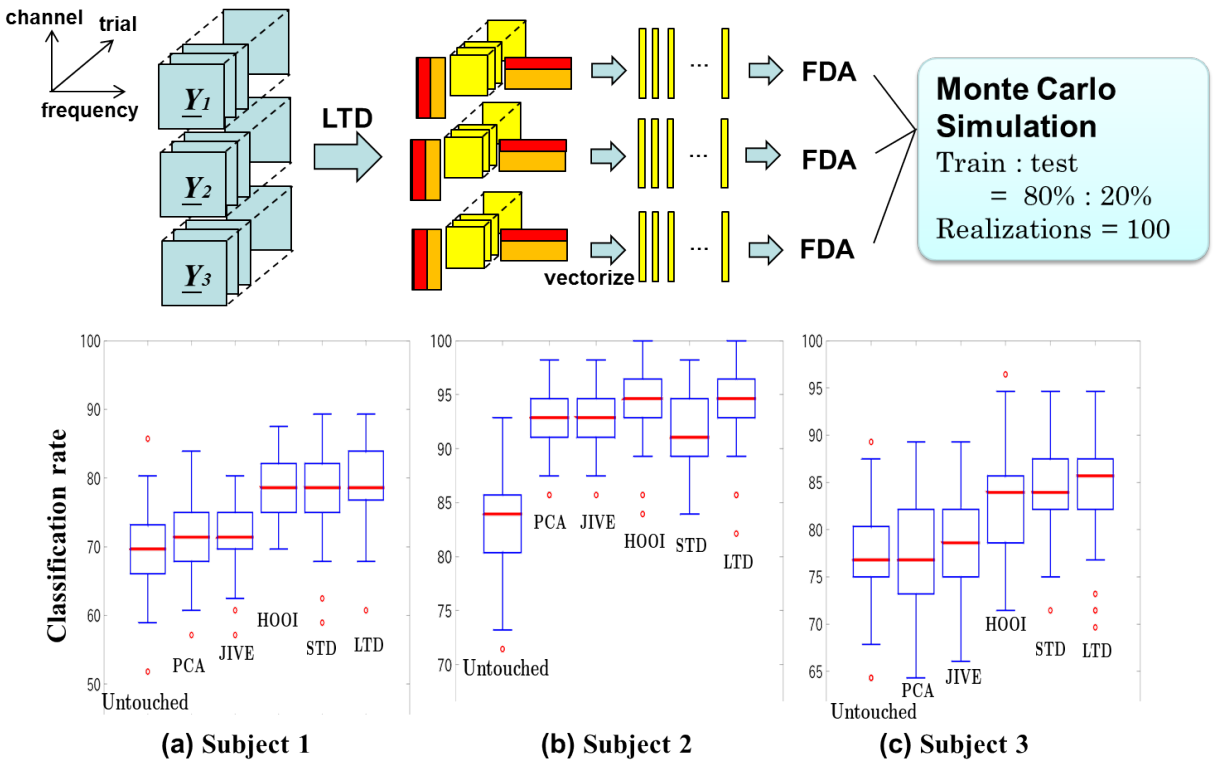


Figure 7.15: Experiments for LTD

7.3 LTD

7.3.1 Feature extraction for classification

In this experiment, I applied the LTD to the feature extraction for the motor imagery BCI. Figure 7.15 shows a flow of this experiment. At first, I used three frequency power tensors which were taken from three persons. Each tensor is constructed by 21 channels, 25 frequency bands, and 280 trials. I decomposed these three tensors by the orthogonal LTD with $L_1 = L_2 = R_1 = R_2 = 4$ without the third-mode factorization. As the result of LTD, we can obtain three core tensors. After that I vectorize each trial matrix, and try to classify by the FDA. In this regard, I iterate to calculate classification rates for 100 times. In each iteration, I randomly separate the dataset into two parts for train (80%) and test (20%), respectively. I compared the LTD with the untouched data, the PCA, the JIVE, the HOOI, and the STD. Figure 7.15 (a,b,c) show the results of statistics of classification rates by individual feature extraction methods. We can see that the LTD outperform the other feature extraction methods.

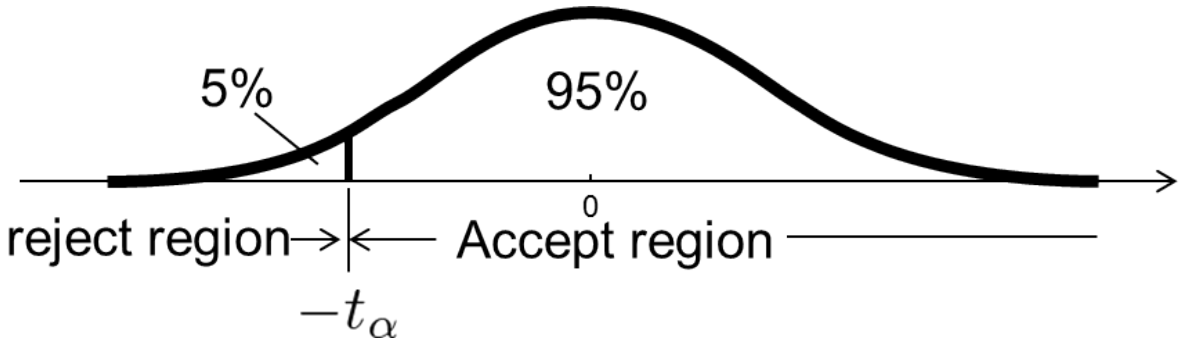


Figure 7.16: t-distribution and reject/accept region with significant level 5%

7.4 Classification

7.4.1 One-tailed t-test

Before showing the experimental results, I introduce the one-tailed t-test that a method to compare a result of proposed method with a result of existing method. The one-tailed t-test plays important roles in statistical data analysis. The classifiers are evaluated by misclassification rates. The lower misclassification rate, the better. The population means of the misclassification rates with some dataset by the proposed method and the existing method are denoted as μ_1 and μ_2, respectively. We test that μ_1 is significantly smaller than μ_2. Thus, the null hypothesis $\mathcal{H}_0$ is given by $\mu_1 = \mu_2$, and the alternative hypothesis $\mathcal{H}_1$ is given by $\mu_1 < \mu_2$. If the null hypothesis is rejected and the alternative hypothesis is accepted with some significant level $\alpha = \alpha_0$, then we can say that μ_1 is significantly smaller than μ_2 with significant level α_0.

In order to test $\mathcal{H}_0$ and $\mathcal{H}_1$, samples of misclassification rates by individual methods, that are denoted by $\{x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(N_1)}\}$ and $\{x_2^{(1)}, x_2^{(2)}, \ldots, x_2^{(N_2)}\}$, are necessary, where N_1 and N_2 are numbers of samples. Sample means $\{\bar{x}_1, \bar{x}_2\}$ and sample variances $\{s_1^2, s_2^2\}$ are given by

$$\bar{x}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} x_i^{(n)}, \tag{7.9}$$

$$s_i^2 = \frac{1}{N_i - 1} \sum_{n=1}^{N_i} (x_i - \bar{x}_i)^2, \tag{7.10}$$

for $i = 1, 2$. In the one-tailed t-test, we consider the value of t which is given by

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s\sqrt{1/N_1 + 1/N_2}}, \tag{7.11}$$

where

$$s^2 = \frac{(N_1 - 1)s_1^2 + (N_2 - 1)s_2^2}{N_1 + N_2 - 2}, \tag{7.12}$$

and assume that t follows the t-distribution with the degree of freedom is $N_1 + N_2 - 2$. Fig. 7.16 shows the image of t-distribution and its reject/accept region with significant level 5%. In this figure, $-t_\alpha$

is a boundary between reject and accept regions. Assuming that $\mathcal{H}_0$ is true, then t is given by

$$t_0 = \frac{(\bar{x}_1 - \bar{x}_2)}{s\sqrt{1/N_1 + 1/N_2}}. \tag{7.13}$$

Under this assumption, if $t_0 < -t_\alpha$, then $\mathcal{H}_0$ is rejected, and if $t_0 \geq -t_\alpha$, then $\mathcal{H}_0$ is accepted. Therefore, we can test that the proposed classifier is significantly better than the existing classifier by the one-tailed t-test with some significant level.

7.4.2 WRSVM

UCI database

In this experiment, I used thirteen UCI data sets for binary problems to compare the two novel classifiers $\mathrm{SVM_G}$ and $\mathrm{SVM_D}$ with SVM and L1-norm regularized SVM (L1-SVM).

These data sets are summarized in Table 7.4, which lists the data set name, the respective numbers of training samples, test samples, realizations, and dimensions.

Several hyper parameters must be optimized, namely, the kernel parameter γ, the adjusting parameter c, and the weighting parameters λ of $Q_2(\boldsymbol{x})$ and ν of $Q_3(\boldsymbol{x})$, but $\rho = 0.9$ is fixed. These parameters are optimized on the first five realizations of each data set. The best values of each parameter are obtained by using each realization. Finally, the median of the five values is selected. After that, the classifiers are trained and tested for all of the remaining realizations (i.e., 95 or 15 realizations) by using the same parameters.

Table 7.5 contains the results of this experiment. The values in the classifier name column show "average $\pm$ standard deviation" of the error rates for all of the remaining realizations, and the minimum values among all classifiers are marked with bold font. The values in the columns for λ and ν show the value selected through model selection for each data set. The signs in columns L2 and L1 show the results of a significance test (t-test with $\alpha = 5\%$) for the differences between the $\mathrm{SVM_G}$/$\mathrm{SVM_D}$ and the SVM/L1-SVM, respectively. "+" indicates that the error obtained with the novel classifier is significantly smaller, while "−" indicates that this error is significantly larger. The penultimate line for "Mean %", is computed by using the average values for all data sets as follows. First, I normalize the error rates by taking

$$\left\{ \frac{(\text{particular value})}{(\text{minimum value})} - 1 \right\} \times 100[\%] \tag{7.14}$$

for each data set. Next, the "average" values are computed for each classifier. This evaluation method is taken from [86]. The last line shows the average of the p-value between "particular" and "minimum" (i.e., the minimum p-value is 50 %).

$\mathrm{SVM_G}$ provides the best results for two data sets. Compared to the SVM, the $\mathrm{SVM_G}$ is significantly better for four data sets and significantly worse for five data sets. Compared to the L1-SVM, the $\mathrm{SVM_G}$ is significantly better for five data sets and significantly worse for three data sets.

Table 7.4: UCI Data sets

No.	Name	# of training samples	# of test samples	# of realizations	Dimensions
1	Banana	400	4900	100	2
2	Breast Cancer	200	77	100	9
3	Diabetes	468	300	100	8
4	Flare-Solar	666	400	100	9
5	German	700	300	100	20
6	Heart	170	100	100	13
7	Image	1300	1010	20	18
8	Ringnorm	400	7000	100	20
9	Splice	1000	2175	20	60
10	Thyroid	140	75	100	5
11	Titanic	150	2051	100	3
12	Twonorm	400	700	100	20
13	Waveform	400	4600	100	21

Table 7.5: Experimental results

	λ	SVM_G	L2	L1	ν	SVM_D	L2	L1	SVM	L1-SVM
Banana	1	10.6 ± 0.4	+		8	**10.4 ± 0.4**	+		11.5 ± 0.7	10.5 ± 0.4
B.Cancer	.01	26.3 ± 5.2			4	26.0 ± 4.4			26.0 ± 4.7	**25.4 ± 4.5**
Diabetes	100	24.1 ± 2.0	−	−	8	24.0 ± 2.0	−	−	23.5 ± 1.7	**23.4 ± 1.7**
F.Solar	10	36.7 ± 5.2	−	−	2	**32.4 ± 1.8**		+	**32.4 ± 1.8**	32.9 ± 2.7
German	10	24.0 ± 2.4			16	24.7 ± 2.3			**23.6 ± 2.1**	24.0 ± 2.3
Heart	10	**15.3 ± 3.2**			2	15.6 ± 3.2			16.0 ± 3.3	15.4 ± 3.4
Image	100	4.3 ± 0.9	−		16	4.1 ± 0.8	−	+	**3.0 ± 0.6**	4.8 ± 1.3
Ringnorm	100	**1.5 ± 0.1**	+	+	8	**1.5 ± 0.1**	+	+	1.7 ± 0.1	1.6 ± 0.1
Splice	10	11.3 ± 0.7	−	+	8	11.0 ± 0.5		+	**10.9 ± 0.7**	12.4 ± 0.9
Thyroid	1	7.3 ± 2.9	−	−	2	**4.1 ± 2.0**	+	+	4.8 ± 2.2	5.4 ± 2.4
Titanic	.01	22.4 ± 1.0		+	2	22.5 ± 0.5		+	**22.4 ± 1.0**	23.0 ± 2.1
Twonorm	10	2.4 ± 0.1	+	+	4	**2.3 ± 0.1**	+	+	3.0 ± 0.2	2.7 ± 0.2
Waveform	1	9.7 ± 0.4	+	+	2	**9.6 ± 0.5**	+	+	9.9 ± 0.4	10.1 ± 0.5
Mean %		12.0				**4.2**			6.1	10.7
P-value %		87.2				**71.5**			79.2	87.5

Furthermore, the SVM_D provides the best results for six data sets. Compared to the SVM, the SVM_D is significantly better for five data sets and significantly worse for two data sets. Compared to the L1-SVM, the SVM_D is significantly better for eight data sets and significantly worse for one data sets.

According to the results for both "mean" and "p-value", the SVM_D classifier is the best among the four classifiers considered.

7.4.3 CFDA

Toy Simulation I

In this simulation, I show the classification error with respect to the difference of two covariance matrices. I assume that there are two normal distributions in the original feature space characterized by

$$\boldsymbol{\mu}_1 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \ \boldsymbol{\Sigma}_1 = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix},$$
$$\boldsymbol{\mu}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ \boldsymbol{\Sigma}_2 = \begin{pmatrix} 1 & 0 \\ 0 & q^2 \end{pmatrix}.$$

The two prior probabilities are equal (i.e., $p_1 = p_2 = 0.5$), and I let the weighting parameters as $\rho_1 = \rho_2 = 0.5$. I evaluate classification errors of the FDA, the Loog's approximated Chernoff FDA (aCFDA) [67], the BFDA, and the CFDA-m by changing the value of q. Figure 7.17 shows the result of this simulation. When $q = 1$, all criteria gave the same result since the two covariance matrices have the same shape. The BFDA and the CFDA-m gave almost the same results. When the value of q increase, the difference of errors between the FDA and the BFDA/CFDA-m increase. Note that classification errors of the BFDA and the CFDA-m were dramatically decrease from $q = 8$. The BFDA and the CFDA-m are very effective when covariance matrices are very different. The aCFDA error was larger than the FDA from $q = 5$ to $q = 7$ although it almost reached to the BFDA/CFDA-m from $q = 8$.

Toy Simulation II

In this simulation, I show the classification error with respect to the difference of prior probabilities of two classes. I assume that there are two classes of samples in a two dimensional space. The number of samples of class 1 is fixed as $N_1 = 100$, and that of class 2 is changed as $N_2 \in \{100,\ 150,\ 200,\ 300\}$. I assumed the samples of each class are distributed normally, and generated samples by Matlab/Octave function 'normrnd'. Mean vectors and covariance matrices are calculated from samples.

Fig. 7.18 shows the result of the classification errors with respect to the difference of prior probabilities. The FDA and the aCFDA gave the same result at $N_2/N_1 = 1$. The aCFDA and the CFDA-p gave similar results. When N_2/N_1 increases, the aCFDA and the CFDA-p almost reached the CFDA-m. Note that the CFDA-m always gave the best results.

Fig. 7.19 shows the samples and the projected space of the FDA, the aCFDA, the CFDA-p, and the CFDA-m at $N_2 = 150$. Class 1 and 2 are denoted by '$\times$' and '$\circ$', respectively. The obtained projected one-dimensional spaces were different among the methods. They can be ranked in the order of the CFDA-m, the CFDA-p, the aCFDA, and the FDA.

Fig. 7.20 shows two distributions in the projected one-dimensional space of individual methods at $N_2 = 150$. They are approximations by normal distribution with the sample mean and the sample

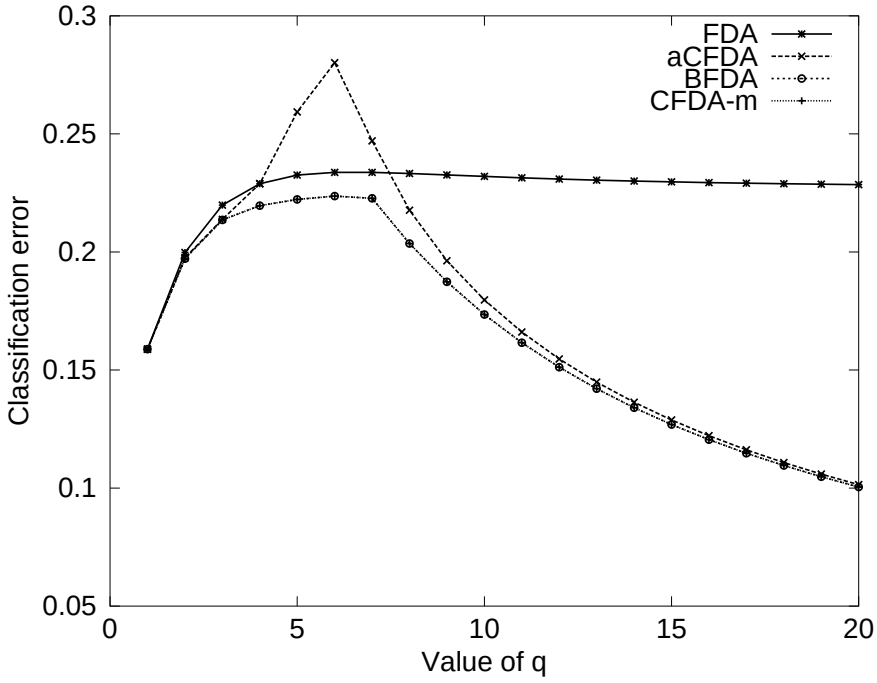


Figure 7.17: Error rates with difference of covariance matrices

variance. Results of the kernel based methods are calculated by (6.167) and (6.168) with $\gamma = 10^{-1}$. Note that one distribution is very narrow and the another distribution is wide in the Chernoff based methods. These can be ranked in the order of the CFDA-m, the CFDA-p, and the aCFDA. The KCFDA-p worked well in the feature space. Note that the KCFDA-m gave an impressive result that the variance of one distribution extremely approached to zero.

UCI database

In this section, I show the results for actual classification datasets. I used nine datasets from the UCI machine learning repository [35]. Table 7.6 describes their properties. Since most of datasets are not for binary classification, I separated them into two classes or focused on two classes. Two datasets of "MNIST" and "Poker hand" have too many instances to apply the kernel method. Therefore, I trained a kernel classifier by using randomly selected 500 instances for training of their datasets. I calculate the 5-fold cross validation (5-CV) errors for all datasets and all methods. In kernel methods, I selected the best kernel parameter γ by hand tuning from candidates of $\{10^{-10},\ 10^{-9},\ \ldots,\ 10^{2},\ 10^{3}\}$.

Table 7.7 describes the averages of maximized values of the Chernoff distance obtained by the CFDA-algorithm and the Rueda's gradient-based algorithm [89]. The Rueda's algorithm can not maximize the Chernoff distance, sometimes. On the other hand, our algorithm provided larger or equal value of the Chernoff distance to the Rueda's algorithm for all datasets.

Table 7.8 describes the number of iterations for main updates in these methods. The values show

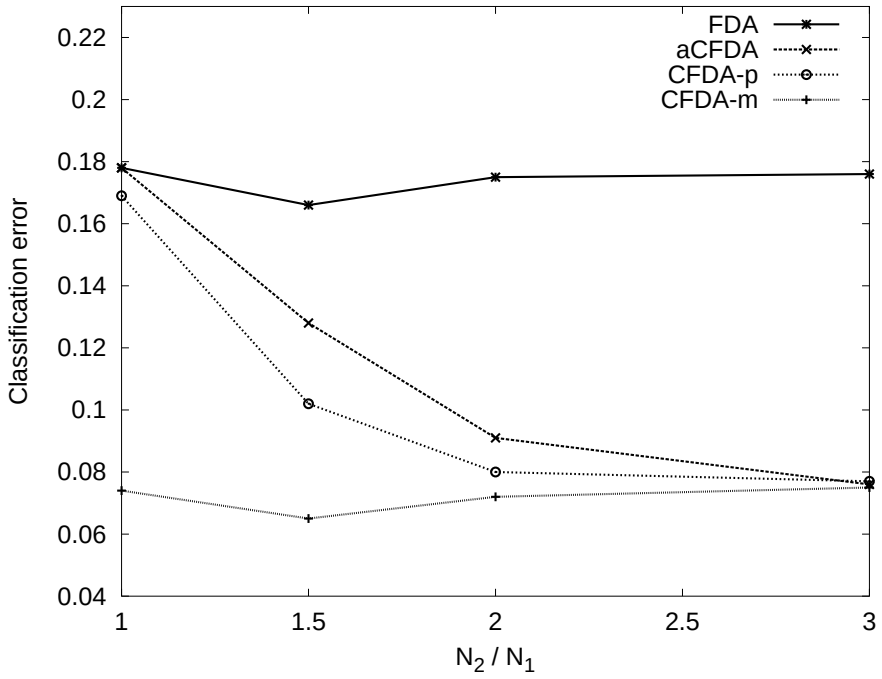


Figure 7.18: Error rates with difference of prior probabilities

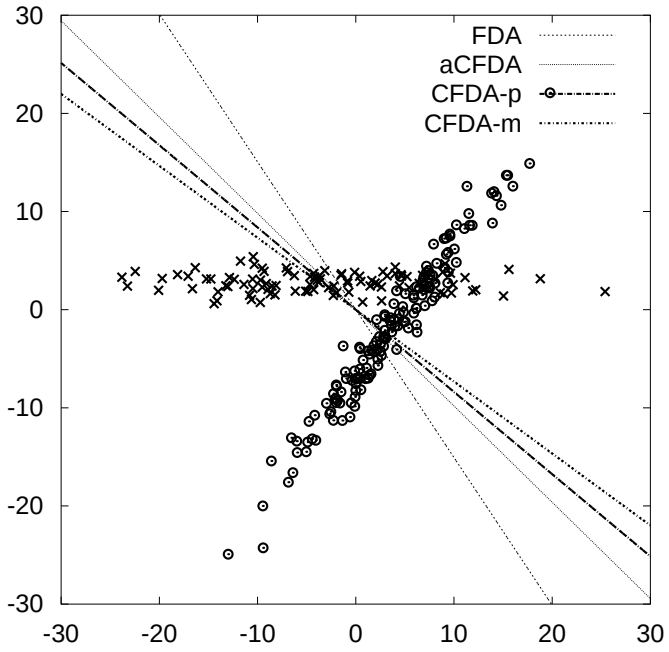


Figure 7.19: Samples and projected space ($N_2/N_1 = 1.5$)

Table 7.6: Properties of UCI datasets

label	name	attributes	instance	class	binalize	p_1	p_2
1	Breast cancer wisconsin	9	569	2	1 vs 2	0.65	0.35
2	Pima indians diabetes	8	768	2	1 vs 2	0.35	0.65
3	SPECTF heart	44	267	2	1 vs 2	0.79	0.21
4	Thyroid gland	5	215	2	1 vs 2	0.30	0.70
5	Iris plants	4	150	3	1,3 vs 2	0.67	0.33
6	Wine	13	178	3	1,2 vs 3	0.73	0.27
7	Glass identification	9	163	7	1,2 vs 3,4	0.90	0.10
8	MNIST digit-recognition	256	1000	10	4 vs 9	0.50	0.50
9	Poker hand	10	25010	10	1 vs 2	0.46	0.54

"average ± standard deviation" of number of iterations. Almost all results of the CFDA-algorithm converged by several iterations. The numbers of iterations of the CFDA-m-algorithm are about nine times of those of the CFDA-algorithm, respectively. These results show that the convergence of the CFDA-algorithm does not depend on the number of attributes and instances. However, we can see more iterations are necessary for the kernel extensions. The Rueda's gradient-based algorithm [89] needed several iterations to converge for almost all data set, however it could not maximize the Chernoff distance for several datasets signed by '*'.

Table 7.9 describes the results of misclassification rates for nine UCI datasets. I described "average ± standard deviation" of 20 realizations of classification errors, which are calculated by using two-third instances for training and one-third instances for test, and the average of estimated values of ρ_1 in the CFDA-p, the CFDA-m, the KCFDA-p, and the KCFDA-m. The Rueda's method is not included since it provides the same result of the CFDA-p when it reached to the maxima of Chernoff distance. The minimum errors are signed by '$\circ$' and significantly better results than existing methods are described by bold font. They were calculated by 5 % significant test. In strictly, it is the one-tailed t-test.

Proposed methods provided minimum error results for five datasets. In linear methods, the proposed methods provided significant improvements for "SPECTF", "Iris", and "Poker hand". In kernel methods, the proposed methods provided significant improvements for "SPECTF", "Glass", and "Poker hand". On the other hand, the linear Bhattacharyya/Chernoff FDA provided clearly wrong results for "Thyroid" and "Glass". These results might be derived from over-fittings or non-Gaussianity. The linear aCFDA provided a significant result for "Thyroid" because of proper fit to heteroscedastic Gaussians. For "Glass", extending a linear discriminant to a non-linear discriminant provided dramatically improvements, and this dataset is made from a complex combination of heteroscedastic and non-Gaussian distributions.

The difference among the BFDA, the CFDA-p and the CFDA-m is only in values of the hyper-parameters ρ_1 and ρ_2. The estimation of the best value of ρ_1 is still a open problem for the Chernoff approaches.

Table 7.7: Maximized values of Chernoff distance by each algorithm

label	1	2	3	4	5	6	7	8	9
CFDA-p	5.71	0.437	4.50	3.09	1.02	7.33	0.851	15.0	0.0342
Gradient	5.71	0.436	1.37	3.09	1.00	0.905	0.340	15.0	0.00424

Table 7.8: Number of iterations for main updates in each method

label	Gradient	BFDA	CFDA-p	CFDA-m	KBFDA	KCFDA-p	KCFDA-m
1	4.0 ± 0.0	2.4 ± 0.5	2.4 ± 0.6	23.3 ± 2.1	5.5 ± 2.0	6.7 ± 2.5	58.4 ± 17.8
2	2.6 ± 0.7	2.8 ± 0.9	3.1 ± 1.0	27.6 ± 2.9	11.8 ± 4.4	12.9 ± 3.8	89.5 ± 25.8
3	* 111.8 ± 102.6	4.4 ± 0.6	4.5 ± 0.5	41.2 ± 4.9	6.2 ± 0.6	6.5 ± 0.6	56.5 ± 2.8
4	17.7 ± 2.7	6.0 ± 0.6	5.8 ± 0.5	52.2 ± 4.0	5.7 ± 2.5	6.5 ± 2.5	57.7 ± 18.4
5	21.1 ± 5.3	5.9 ± 0.8	6.7 ± 1.5	54.7 ± 8.3	9.1 ± 3.8	10.7 ± 3.7	87.2 ± 35.4
6	* 656.5 ± 450.4	2.9 ± 0.4	2.9 ± 0.5	28.0 ± 1.5	4.4 ± 0.6	8.0 ± 1.3	46.0 ± 2.9
7	* 20.5 ± 16.0	5.1 ± 0.6	5.1 ± 0.6	47.0 ± 5.6	5.5 ± 0.9	5.2 ± 0.7	55.5 ± 4.2
8	2.0 ± 0.0	5.9 ± 0.9	6.2 ± 0.9	58.2 ± 5.4	4.0 ± 0.5	4.0 ± 0.5	47.6 ± 4.8
9	* 2.4 ± 1.3	3.5 ± 0.5	3.5 ± 0.5	31.6 ± 3.0	7.0 ± 1.7	6.9 ± 1.2	60.6 ± 11.2

(*: could not reach to the maxima of Chernoff distance)

7.4.4 QCMAP

UCI database: Optimization

In this section, I discuss the computational cost of the QCMAP estimation in comparison with other methods. I implemented an optimizer based on the primal-dual interior point (PDIP) algorithm for the QCMAP problem in the programming language C. The PDIP algorithm was explained in Section 6.1.7.

I used the PR_LOQO optimizer for the SVM problem for comparison. The PR_LOQO is an optimizer for convex quadratic programming, created by A. Smola in 1997 and implemented in the programming language C. It is based on the PDIP algorithm including a path-following method.

In both implementations, individual calculation procedures (e.g., matrix and vector multiplication, and Cholesky decomposition) are almost equivalent. Therefore, the two optimizers operate under nearly equal conditions.

For reference, I also tested the IBM ILOG CPLEX Optimizer [51] for the SVM problem.

I tested their optimizers for each problem using the UCI data sets [35] (see Table 7.4), and measured computational times [sec] for each realization when running on a computer with an Intel Core i7 3.33 GHz CPU and 12 GBytes of RAM. Table 7.10 gives average times ± standard deviations for all realizations.

Table 7.9: Classification errors for UCI dataset

label	FDA	aCFDA	BFDA	CFDA-p	ρ_1 ave.	CFDA-m	ρ_1 ave.
1	2.6 ± 0.9	2.4 ± 0.8	2.6 ± 0.9	2.4 ± 0.9	0.35	2.4 ± 0.9	0.32
2	23.7 ± 1.9	23.5 ± 2.0	23.6 ± 2.1	23.5 ± 1.8	0.65	23.5 ± 2.1	0.50
3	31.7 ± 3.9	34.8 ± 3.7	∘ **19.4 ± 3.5**	∘**19.4 ± 3.5**	0.21	∘**19.4 ± 3.5**	0.90
4	13.1 ± 4.1	**8.4 ± 3.0**	18.2 ± 4.3	18.2 ± 4.3	0.69	18.2 ± 4.3	0.80
5	26.6 ± 5.0	27.4 ± 4.6	**6.6 ± 2.7**	**6.6 ± 2.7**	0.33	**6.6 ± 2.7**	0.80
6	1.0 ± 1.4	1.9 ± 1.9	1.0 ± 1.4	1.9 ± 1.9	0.28	∘ 0.9 ± 1.2	0.70
7	**17.2 ± 5.6**	21.3 ± 7.3	27.0 ± 14.9	27.0 ± 14.9	0.11	27.0 ± 14.9	0.83
8	3.8 ± 1.2	3.8 ± 1.2	3.8 ± 1.2	3.8 ± 1.2	0.50	3.9 ± 1.4	0.55
9	50.4 ± 0.9	50.3 ± 1.0	∘**41.1 ± 0.4**	∘**41.1 ± 0.4**	0.54	∘**41.1 ± 0.4**	0.50

label	KFDA	Kernel aCFDA	KBFDA	KCFDA-p	ρ_1 ave.	KCFDA-m	ρ_1 ave.
1	∘ 2.2 ± 0.7	∘ 2.2 ± 0.7	∘ 2.2 ± 0.8	∘ 2.2 ± 0.7	0.35	∘ 2.2 ± 0.8	0.51
2	23.4 ± 1.6	∘ 22.9 ± 1.8	23.5 ± 1.8	23.4 ± 2.0	0.65	23.7 ± 1.9	0.52
3	25.7 ± 4.2	28.5 ± 3.4	∘**19.4 ± 3.1**	∘**19.4 ± 3.1**	0.21	∘**19.4 ± 3.1**	0.90
4	6.2 ± 2.4	∘ 5.3 ± 2.0	6.2 ± 2.5	5.5 ± 2.0	0.69	6.5 ± 2.3	0.73
5	∘ 2.8 ± 2.2	3.1 ± 2.6	3.1 ± 2.4	3.3 ± 2.5	0.33	3.4 ± 2.4	0.66
6	3.7 ± 2.6	3.6 ± 3.1	6.9 ± 3.6	3.7 ± 3.3	0.28	6.6 ± 3.6	0.58
7	18.3 ± 3.7	26.8 ± 5.2	**9.5 ± 3.0**	20.2 ± 5.8	0.11	∘**9.4 ± 3.3**	0.89
8	0.7 ± 0.4	0.7 ± 0.3	0.7 ± 0.4	∘ 0.6 ± 0.4	0.50	0.8 ± 0.4	0.51
9	44.6 ± 1.0	44.7 ± 1.2	**41.9 ± 0.8**	**42.2 ± 0.9**	0.54	**41.8 ± 1.0**	0.56

UCI database: Classification

I tested four classifiers provided by individual and combined novel weight function strategies. These classifiers and their parameter settings are listed in Table 7.11, where $\bar{\boldsymbol{x}}$ and $\hat{\boldsymbol{\Sigma}}$ are given by

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n, \tag{7.15}$$

$$\hat{\boldsymbol{\Sigma}} = \begin{cases} \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{\mu}(i) - \boldsymbol{x}_n(i))^2 & i = j \\ 0 & i \neq j \end{cases}. \tag{7.16}$$

I also have to select the parameter s in the KDEQCM method, and if we want to make the constraints strong, then s must be small. However, if s is too small in KDEQCM, then the classifier is almost the same as LSR and its results are meaningless. Therefore, I selected $s = 0.01$ in KDEQCM. Nevertheless, we cannot guarantee these parameter settings are optimal.

Conversely, the kernel parameter γ is optimized as follows:

- Repeat the five-fold cross validation five times using different realizations, and record their optimal parameters: $\{\gamma^{(1)}, \ldots, \gamma^{(5)}\}$.
- Select a median value from their values:

$$\gamma^* = \text{Median}\left[\gamma^{(1)}, \ldots, \gamma^{(5)}\right]. \tag{7.17}$$

Table 7.10: Computational time of optimizers

Optimizer	PR_LOQO in C for SVM	CPLEX in C for SVM	PDIP in C for QCMAP
Banana	5.93 ± 0.05	0.26 ± 0.10	5.09 ± 1.22
Breast Cancer	0.76 ± 0.01	0.08 ± 0.01	1.13 ± 0.30
Diabetes	9.24 ± 0.06	0.37 ± 0.03	18.60 ± 3.33
Flare-Solar	27.29 ± 0.16	0.09 ± 0.00	93.13 ± 45.12
German	30.45 ± 0.15	0.89 ± 0.02	211.20 ± 39.16
Heart	0.53 ± 0.03	0.04 ± 0.00	0.65 ± 0.29
Image	207.28 ± 1.17	5.86 ± 0.19	346.48 ± 63.91
Ringnorm	6.44 ± 0.03	0.36 ± 0.01	5.67 ± 1.11
Splice	86.51 ± 0.36	2.36 ± 0.04	318.77 ± 87.80
Thyroid	0.33 ± 0.02	0.03 ± 0.00	0.22 ± 0.06
Titanic	0.35 ± 0.01	0.02 ± 0.00	0.30 ± 0.20
Twonorm	5.55 ± 0.03	0.35 ± 0.02	6.83 ± 3.14
Waveform	6.08 ± 0.04	0.30 ± 0.07	7.15 ± 1.44

This cross-validation procedure is taken from [86, 74] and following that study. I selected the kernel parameters from among:

$$\{10^{-6.0},\ 10^{-5.75},\ 10^{-5.50}, \ldots,\ 10^{0.50},\ 10^{0.75},\ 10^{1.00}\}.$$

For QCMAP estimation, we have only one hyper-parameter for model selection. This is an important property of the QCMAP method since computation time of cross-validation increases exponentially with the number of hyper-parameters.

The classifiers were trained by using the PDIP algorithm described in Section 6.1.7. In this regard, I set the approximation parameter $t = 2$. Although this value does not result in highly accurate approximations of $\min(r_n, 1)$, it maintains smooth progression of the algorithm.

The k-nearest neighbor (kNN) method, the regularized AdaBoost (AB_R) method and SVM were used for comparison with the proposed classifiers. In the kNN method, I select the optimal value of k from $\{1,\ 3,\ 5,\ \ldots,\ 61\}$. Results of AB_R and SVM are referenced from [86, 74]. The RBF Gaussian kernel was used in SVM. Regularization parameters of the AB_R and the tradeoff (regularization) parameter and kernel parameter of SVM were selected in the same way (the five-fold cross-validation method). Table 7.12 shows the results of this experiment.

The values listed in this table show the mean value ± standard deviation of the error rates for all realizations, and the minimum error rate for each data set is denoted by a bold font. Symbols '•', '∘', and '∗' indicate that the error of QCMAP is significantly smaller than the errors of kNN, AB_R, and SVM, respectively, by one tailed t-test with significance level of 5%.

Table 7.11: Proposed Classifiers tested in the experiment

Number	Classifier(s)	Parameter Setting (First Classifier)	Parameter Setting (Second Classifier)	Mixing Rate
1	GQCM	$\boldsymbol{\mu} = \bar{\boldsymbol{x}}$, $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$		
2	DOGQCM	$\boldsymbol{\mu} = \bar{\boldsymbol{x}}$, $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$, $\nu = 4$, $\kappa = 0.9$		
3	KDEQCM	$\boldsymbol{\mu}_n = \boldsymbol{x}_n$, $s = 0.01$, $n = 1, \ldots, N$		
4	DOGQCM + KDEQCM	$\boldsymbol{\mu} = \bar{\boldsymbol{x}}$, $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$, $\nu = 4$, $\kappa = 0.9$	$\boldsymbol{\mu}_n = \boldsymbol{x}_n$, $s = 0.01$, $n = 1, \ldots, N$	$p = 0.5$

Table 7.12: Experimental results of UCI data sets

	kNN	AB_R	SVM	GQCM		DOGQCM		KDEQCM		DOG+KDE	
Banana	11.3 ± 0.6	10.9 ± 0.4	11.5 ± 0.7	•○∗	**10.5 ± 0.4**	•○∗	**10.5 ± 0.4**	•○∗	**10.5 ± 0.5**	•○∗	**10.5 ± 0.4**
Breast Cancer	**25.3 ± 4.0**	26.5 ± 4.5	26.0 ± 4.5		26.0 ± 4.4	○	25.4 ± 4.3		26.0 ± 4.5		25.5 ± 4.5
Diabetes	25.4 ± 1.8	23.8 ± 1.8	23.5 ± 1.7	•○	**23.2 ± 2.0**	•○	**23.2 ± 1.7**	•○	**23.2 ± 2.0**	•○	**23.2 ± 1.8**
Flare-Solar	34.8 ± 1.9	34.2 ± 2.2	**32.4 ± 1.8**	•○	33.3 ± 1.8	•○	33.3 ± 1.7	•○	33.3 ± 1.8	•○	33.3 ± 1.7
German	25.2 ± 2.3	24.3 ± 2.1	**23.6 ± 2.1**	•○	23.7 ± 2.2	•○	23.7 ± 2.2	•○	23.7 ± 2.2	•	23.8 ± 2.3
Heart	15.7 ± 3.4	16.5 ± 3.5	16.0 ± 3.3	○	**15.6 ± 3.2**	○	**15.6 ± 3.3**		15.7 ± 3.4		15.7 ± 3.4
Image	3.4 ± 0.5	2.7 ± 0.6	3.0 ± 0.6		3.6 ± 0.6	•	2.7 ± 0.6		3.1 ± 0.7	•∗	**2.6 ± 0.5**
Ringnorm	35.0 ± 1.4	**1.6 ± 0.1**	1.7 ± 0.1	•	1.9 ± 0.1	•	1.8 ± 0.1	•	4.6 ± 0.8	•	2.0 ± 0.4
Splice	26.1 ± 2.2	**9.5 ± 0.7**	10.9 ± 0.7	•	10.8 ± 0.6	•	10.7 ± 0.7	•	10.9 ± 0.7	•	10.8 ± 0.8
Thyroid	4.4 ± 2.2	4.6 ± 2.2	4.8 ± 2.2	∗	4.1 ± 2.7	•○∗	3.8 ± 2.1		6.9 ± 3.0	•○∗	**3.7 ± 2.1**
Titanic	22.8 ± 1.1	22.6 ± 1.2	22.4 ± 1.0	•○	22.2 ± 1.0	•○	22.2 ± 1.0	•○∗	**21.9 ± 1.0**	•○	22.3 ± 1.1
Twonorm	2.6 ± 0.2	2.7 ± 0.2	3.0 ± 0.2	•○∗	2.4 ± 0.1	•○∗	**2.3 ± 0.1**	•○∗	2.4 ± 0.1	•○∗	**2.3 ± 0.1**
Waveform	10.7 ± 1.0	**9.8 ± 0.8**	9.9 ± 0.4	•	10.2 ± 0.6	•	**9.8 ± 0.5**	•	10.1 ± 0.5	•	9.9 ± 0.5

•: Superior to kNN solution by t-test with significance level 5%.
○: Superior to AB_R solution by t-test with significance level 5%.
∗: Superior to SVM solution by t-test with significance level 5%.

Weight function selection

In this section, I try to select a good weight function for a data set by using the results of previous experiments. From the results, I found that similar weight functions performed in similar accuracy, DOGQCM was superior to GQCM for all data sets, and DOG+KDE was partially superior to DOGQCM. Therefore, the number of candidates of weighting functions can be reduced. In this experiment, I employed DOGQCM and DOG+KDE as the candidates, and propose a method to select either of the weight functions. DOG+KDE is suitable for an easy data set and DOGQCM is suitable for a difficult data set because DOG+KDE would be more similar to LSR than DOGQCM. Fig. 7.21 shows the 2d-plot of NN and LDA error rates for 13 UCI data sets. NN error rates are calculated as the rate of samples of which the nearest neighbor belongs to the other class. LDA error rates are calculated by using the same samples for training and testing. From the previous results, DOG+KDE was almost always better for 'Image', 'Thyroid', and 'Twonorm', and DOG+KDE was almost always better for the other data sets. Then I decided the boundary line between the two weight functions by LDA. When another data set is given, we can select either of the weight functions by calculating NN and LDA error rates and separating it with the boundary line. Since the computational costs of NN and LDA are much smaller than kernelized methods, we can select a weight function effectively. I call it the weight function selection QCMAP (WS-QCM).

Application to Brain Signal

I conducted an experiment by using the challenging brain computer interface (BCI) data set [8]. This is for motor imagery classification to assign EEG signals into either 'foot' imagery or 'right hand' imagery. It consists of 280 EEG signals for 5 subjects. I extracted features from EEG signals into six dimensional feature vectors by the common spacial pattern (CSP) filter [75]. And I calculated generalization errors by 5-fold cross validation (CV) for each subject. Table 7.13 describes the result of this experiment. I compared the proposed method to SVM with two types of hyper parameter selection approaches. SVM has two hyper parameters: a kernel parameter γ and a tradeoff parameter λ. I selected an optimal combination from grid-like candidates by the 5-fold CV. The number of candidates of γ and λ are 29 and 13, then the number of all combinations is 377 in this method. The first approach is to optimize two parameters from all 377 combinations of grid-like candidates by the 5-fold CV. I denoted this method as "SVM-1" in the table. The other approach consists of two phases. In the first phase, I obtain a combination of values from rough grid-like candidates by the 5-fold CV. In the second phase, I optimize two parameters from fine grid-like candidates around the combination by the 5-fold CV. I denoted this method as "SVM-2" in the table. From the result, SVM-2 (two phase CV) was better than SVM-1 (all search CV) for three subjects. The fact implies that the CV based hyperparameter selection is very difficult.

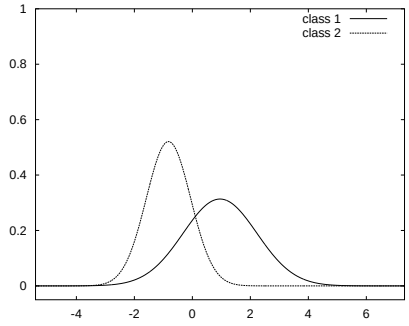
Furthermore, I applied the SVM_G, the SVM_D, and the KCFDA-m to this experiment. Hyperparameters of the SVM_G and the SVM_D were optimized by the two phase CV. Kernel paramter γ of the KCFDA-m was optimized by the 5-fold CV.

QCMAP approach has one hyper parameter as a kernel parameter γ and two candidates of weight functions. I first selected either of the weight functions by the boundary line in the NN and LDA error rate space. After that, I optimized its kernel parameter γ by the 5-fold CV. I denoted this method as "WS-QCM" in the table. The column of "DOG:D&K" shows numbers of times that DOGQCM and DOG+KDE were selected, respectively. The columns of "only DOG" and "only D&K" show the error rates of individual weight functions. From the results, weight function selection worked well for three out of five subjects. The data of Subjects 2 and 4 are easy for classification, and DOG+KDE provided better performances. The data of Subjects 3 is difficult, and DOGQCM provided a better performance. The data of Subjects 1 and 5 are intermediate, and we could not select the proper weight function. However, the differences between error rates of DOGQCM and DOG+KDE are small for the two subjects. The experimental result shows the proposed selection method can be applied to practical problems. Another notable result is that QCMAP approaches provided better performances than SVM for almost all subjects. Furthermore, I have to recall that one hyper parameter is more easily optimized than two hyper parameters, and we can use QCMAP more easily than SVM.

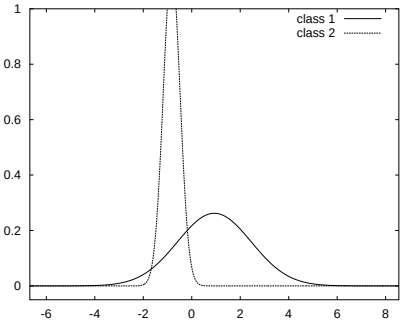
Table 7.13: Experimental results of a BCI data set by the weight function selection

	SVM-1(all search)	SVM-2(two phase)	SVM_G	SVM_D	KCFDA-m
Subject 1	28.2 ± 5.8	28.6 ± 5.9	23.6 ± 6.1	24.3 ± 7.4	23.9 ± 4.5
Subject 2	2.1 ± 1.5	2.9 ± 2.0	2.1 ± 1.5	1.4 ± 1.5	1.8 ± 1.3
Subject 3	37.1 ±10.8	35.7 ± 7.7	36.4 ± 11.1	38.6 ± 6.0	47.9 ± 8.8
Subject 4	8.6 ± 2.0	8.2 ± 3.0	8.2 ± 3.0	8.6 ± 2.9	**6.4 ± 3.0**
Subject 5	15.0 ± 4.8	13.6 ± 3.9	15.7 ± 5.0	15.7 ± 2.0	13.2 ± 6.3

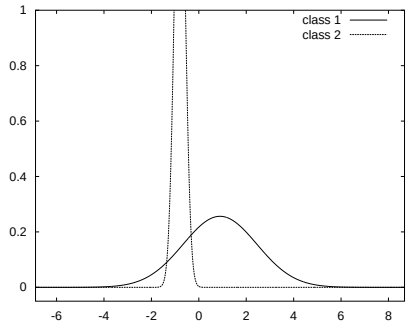
WS-QCM	DOG:D&K	only DOG	only D&K
23.9 ± 4.8	5 : 0	23.9 ± 4.8	**23.2 ± 4.6**
1.1 ± 1.6	0 : 5	1.4 ± 1.5	**1.1 ± 1.6**
33.6 ± 6.5	5 : 0	**33.6 ± 6.5**	35.7 ± 6.1
7.9 ± 2.0	0 : 5	9.3 ± 2.9	7.9 ± 2.0
13.6 ± 5.0	1 : 4	**12.9 ± 5.8**	13.6 ± 5.0



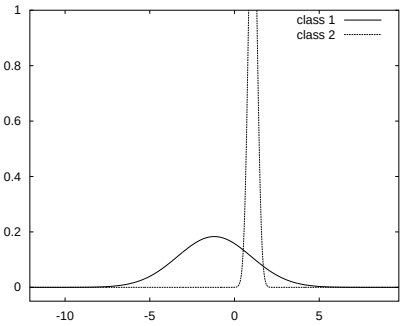
(a) FDA



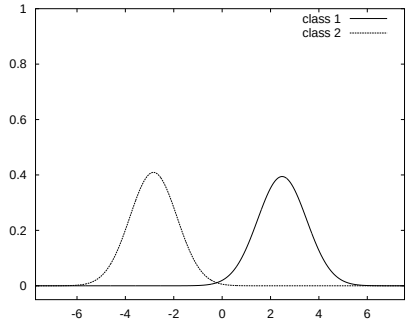
(b) aCFDA



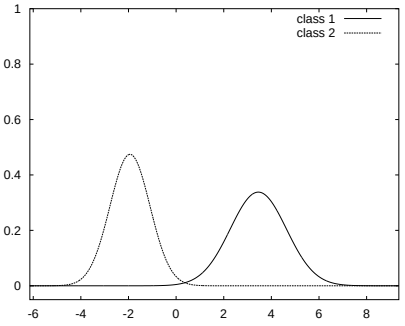
(c) CFDA-p



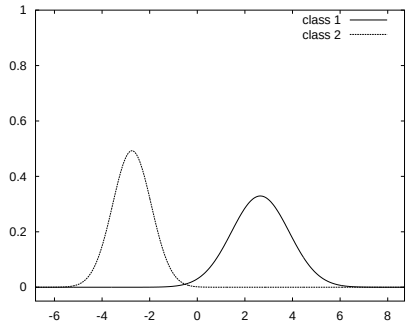
(d) CFDA-m



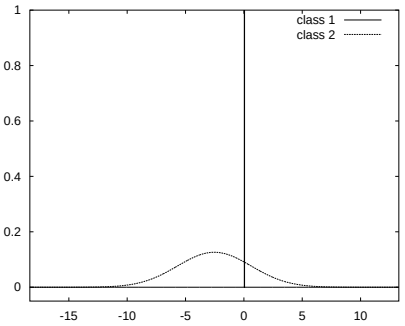
(e) KFDA



(f) Kernel aCFDA



(g) KCFDA-p



(h) KCFDA-m

Figure 7.20: Distributions in projected spaces ($N_2/N_1 = 1.5$)

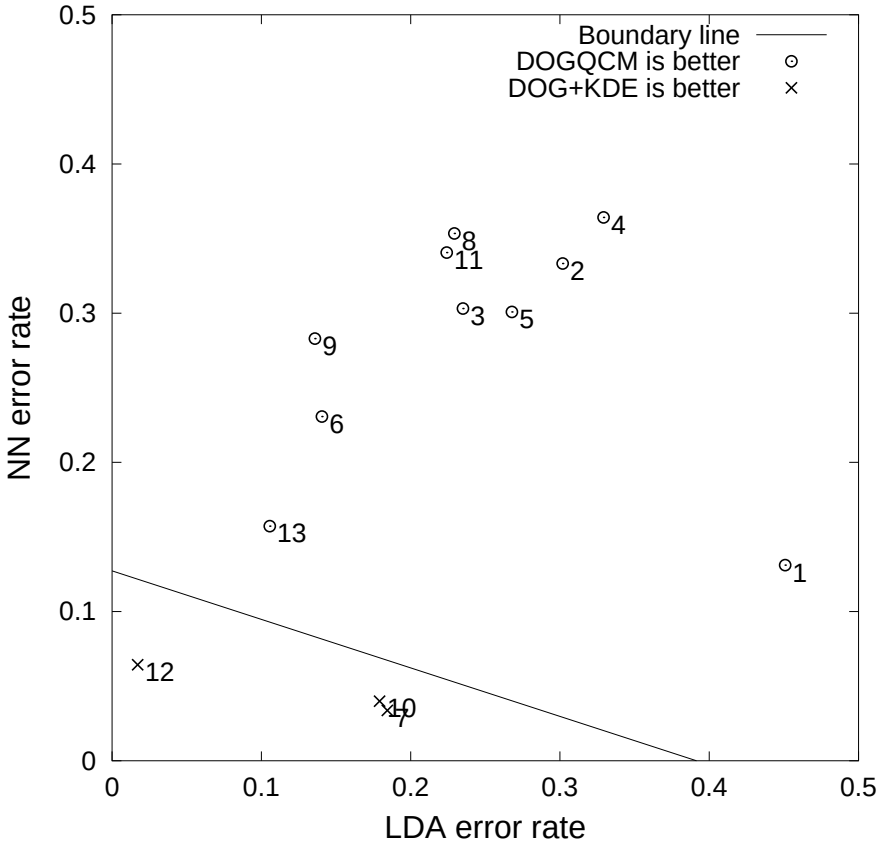


Figure 7.21: Score plot of UCI data sets

Chapter 8

Conclusions

8.1 Summary

In this thesis, I have explained the matrix and tensor based feature extraction and the classification methods, and proposed several new methods for them.

In Chapter 2, I have introduced several existing methods for data analysis: the discrete Fourier transform (DFT), the discrete wavelet transform (DWT), the principal component analysis (PCA), the sparse principal component analysis (SPCA), and the independent component analysis (ICA).

In Chapter 3, I have introduced matrix and tensor decomposition techniques. The most basic matrix decomposition is the singular value decomposition (SVD). The SVD provides a full-rank complete decomposition which consists of orthonormal left and right singular matrices and a diagonal rectangular matrix. The truncated SVD (tSVD) is a low-rank approximation model of SVD. The tSVD provides the minimum mean squared error between an original data and a decomposition model. The SVD and the tSVD features are obtained based on orthogonality constraint; however, orthogonality is not always efficient. I have to use also sparsity, smoothness and nonnegativity constraints as the situation demands. The penalized matrix factorization (PMD) is a technique to impose a sparsity or a smoothness constraint to the matrix decomposition model, and the nonnegative matrix factorization (NMF) is a technique to impose a nonnegativity constraint to the matrix decomposition model. Furthermore, the sparse NMF, the smooth NMF methods have been introduced.

The tensor decomposition is an extension technique of the matrix decomposition to analyze multi-dimensional array (tensor) data. There are two models in the tensor decomposition: CP model and Tucker model. The CP model is a direct extension of the matrix decomposition, and the Tucker model is more general extension. I have introduced to impose the orthogonality, the sparsity and the nonnegativity constraints to both models.

Furthermore, I have introduced the common and individual feature extraction, and several solution algorithms. The common and individual feature extraction is a very important technique for data analysis; however, this is technically complex.

In Chapter 4, I have introduced a supervised-future extraction and several basic classification meth-

ods: the common spatial pattern (CSP) filter, the kernel method, the least squares regression (LSR), the Fisher discriminant analysis (FDA), and the support vector machine (SVM).

In Chapter 5, I have proposed the following new algorithms for matrix and tensor based feature extraction:

- Fast algorithm and extension methods for Smooth NMF with function approximation
  - HALS based fast improvement algorithm
  - Extensive basis function
  - Reduction method of the size of basis matrix
  - CP tensor extension
  - Tucker tensor extension
- Linked (common and individual) CP tensor decomposition (LCPTD)
  - Nonconstrained LCPTD
  - Sparse LCPTD
  - Nonnegative LCPTD
- linked (common and individual) Tucker decomposition (LTD)
  - Orthogonal LTD
  - Sparse LTD
  - Nonnegative LTD

In Chapter 6, I have proposed the following new classification methods:

- QCMAP classifiers
  - QCMAP classifier with the Gaussian weight function (GQCM)
  - QCMAP classifier with the difference of the Gaussian weight functions (DoGQCM)
  - QCMAP classifier with the kernel density estimated weight function (KDEQCM)
  - QCMAP classifier with the DoG and the KDE weight functions (Dog+KDE-QCM)
- Support vector machine with weighted regularizations (WRSVM)
  - WRSVM with the Gaussian weight function (SVM_G)
  - WRSVM with the difference of the Gaussian weight functions (SVM_D)
- Chernoff Fisher discriminant analysis (CFDA) classifiers and their kernel extensions
  - Bhattacharyya FDA (BFDA)

- Chernoff FDA with prior based parameter (CFDA-p)
- Chernoff FDA with error minimization (CFDA-m)
- their kernel extensions (KBFDA, KCFDA-p, KCFDA-m)

In Chapter 7, I have shown the experimental results of new feature extraction and classification methods. To evaluate proposed feature extraction methods, I conducted experiments for the low-rank approximation, the face reconstruction, the blind source separation (BSS), the denoising, the parts-based representation, and the compression. I showed the advantages of a new fast algorithm for smooth NMF/NTF and new linked tensor decomposition models. To evaluate proposed classification methods, I conducted experiments of classification with UCI datasets and BCI dataset. I showed the advantages of the proposed classifiers.

8.2 Open problems

In this section, I mention about open problems for individual methods.

8.2.1 Feature Extraction

In almost all cases, matrix/tensor decomposition is used as a low-rank approximation; however, the best rank is usually unknown. For example, it is difficult to know the number of original sources in BSS problem. It should be careful to decide the amount of reduction of the information of data in dimensionality reduction. Another problem is how to apply our new methods to big data. In order to process the big data, it is necessary to reduce computation time and online algorithm is also important. Furthermore, researches to apply the proposed method to various types of brain data is necessary.

8.2.2 Classification

Hyper parameter selection methods can be considered as an open problem in classification methods. In recent research, most of efficient classifiers are based on kernel method such as SVM. Furthermore, most of robust approaches are based on regularization techniques. Therefore, most of recent classifiers have at least two hyper parameters: a kernel parameter and a regularization parameter. For two hyper parameters, their parameter domain is a 2-dimensional hyper plane. In this case, there are too many candidates of parameter combinations to select. WRSVM has this problem. CFDA based classifiers have only the kernel parameter; however, they still have a possibility of over-fitting. Furthermore, I have proposed only one-dimensional feature model, then the research for multi-dimensional feature model is necessary. The QCMAP classifiers have only a kernel parameter and it is robust for outliers. How to select kernel parameter is an open problem. It can be considered to apply the multiple kernel learning (MKL) methods to the QCMAP estimation. This will provide parameter-less efficient classifiers. Reduction of computational time (especially, learning time) is sometimes considered less serious; however, it is also very important from the view point of online learning. Difficult and complex

criteria and optimization algorithms are not efficient for online learning. It is necessary to apply the proposed methods to various types of brain data (especially multi-categorical data).

Bibliography

[1] T. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *Computers, IEEE Transactions on*, 100(1):90–93, 1974.

[2] G.I. Allen. Regularized tensor factorizations and higher-order principal components analysis. *arXiv preprint arXiv:1202.2476*, 2012.

[3] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Comput.*, 12(10):2385–2404, October 2000.

[4] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 711–720, 1997.

[5] H.Y. Benson and R.J. Vanderbei. Solving problems with semidefinite and related constraints using interior-point methods for nonlinear programming, 2002.

[6] J. Besag. Towards bayesian image analysis*. *Journal of Applied statistics*, 20(5-6):107–119, 1993.

[7] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] B. Blankertz, K.R. Muller, D. Krusienski, G. Schalk, J.R. Wolpaw, A. Schlogl, G. Pfurtscheller, J.R. Millan, M. Schröder, and N. Birbaumer. The BCI competition III: Validating alternative approaches to actual BCI problems. *IEEE Trans. Neural Systems and Rihabilitation Engineering*, 14:153–159, 2006.

[9] R. Bro. Multi-way analysis in the food industry - models, algorithms, and applications. Technical report, MRI, EPG and EMA, ” Proc ICSLP 2000, 1998.

[10] R. Bro and S.D. Jong. A fast non-negativity-constrained least squares algorithm. *Journal of chemometrics*, 11(5):393–401, 1997.

[11] J. Carroll and J.J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of “ eckart-young ” decomposition. *Psychometrika*, 35:283–319, 1970.

[12] Y.H. Chao, H.M. Wang, and R.C. Chang. GMM-based Bhattacharyya kernel fisher discriminant analysis for speaker recognition. In *Proc. ICASSP*, volume 3, pages 3–6, 2005.

[13] Z. Chen, A. Cichocki, and T.M. Rutkowski. Constrained non-negative matrix factorization method for EEG analysis in early detection of alzheimer disease. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages 893–896. IEEE, 2006.

[14] M.T. Chu and N.T. Trendafilov. The orthogonally constrained regression revisited. *Journal of Computational and Graphical Statistics*, 10(4):pp. 746–771, 2001.

[15] H. Chun and S. Kele∘s. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(1):3–25, 2010.

[16] R.V. Churchill. *Complex Variables and Applications*. McGraw-Hill, New York, 1960.

[17] A. Cichocki. Tensor decompositions: New concepts for brain data analysis? *Journal of Control, Measurement, and System Integration (SICE)*, 7:507–517, 2011.

[18] A. Cichocki, S. Amari, et al. *Adaptive blind signal and image processing*. John Wiley Chichester, 2002.

[19] A. Cichocki and A.H. Phan Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.

[20] A. Cichocki, S. Cruces, and S. Amari. Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization. *Entropy*, 13(1):134–170, 2011.

[21] A. Cichocki and R. Zdunek. NMFLAB-NTFLAB for signal and image processing. *Technical Report, Laboratory for Advanced Brain Processing, BSI, RIKEN*, 2006.

[22] A. Cichocki and R. Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorization. In *Advances in Neural Networks–ISNN 2007*, pages 793–802. Springer, 2007.

[23] A. Cichocki, R. Zdunek, and S. Amari. Csiszar 's divergences for non-negative matrix factorization: Family of new algorithms. In *Independent Component Analysis and Blind Signal Separation*, pages 32–39. Springer, 2006.

[24] A. Cichocki, R. Zdunek, and S. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3d tensor factorization. In *Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.

[25] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Non-negative tensor factorization using alpha and beta divergences. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 3, pages III–1393. IEEE, 2007.

[26] A. Cichocki, R. Zdunek, A.H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley Publishing, 2009.

[27] J.L. Crowley and A.C. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(2):156 –170, march 1984.

[28] L.D. Lathauwer, B.D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[29] L.D. Lathauwer, B.D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r_1, r_2,..., r_n) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[30] H.P. Decell Jr. and S.M. Mayekar. Feature combinations and the divergence criterion. *Computers & Mathematics with Applications*, 3(1):71 – 76, 1977.

[31] D.L. Donoho and I.M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.

[32] K. Drakakis, S. Rickard, R.D. Frein, and A. Cichocki. Analysis of financial data using non-negative matrix factorization, 2008.

[33] S. Essid and C. Fevotte. Smooth nonnegative matrix factorization for unsupervised audiovisual document structuring. *Multimedia, IEEE Transactions on*, 15(2):415–425, 2013.

[34] C. Fevotte, N. Bertin, and J.L. Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[35] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[36] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[37] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. John Wiley & Sons, second edition, 1990.

[38] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on pattern analysis and machine intelligence*, 14(3):367–383, 1992.

[39] N. Gillis. Sparse and unique nonnegative matrix factorization through data preprocessing. *arXiv preprint arXiv:1204.2436*, 2012.

[40] N. Gillis and F. Glineur. Using under approximations for sparse nonnegative matrix factorization. *Pattern Recognition*, 43(4):1676–1687, 2010.

[41] C.C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992.

[42] P.J. Green. Bayesian reconstructions from emission tomography data using a modified em algorithm. *Medical Imaging, IEEE Transactions on*, 9(1):84–93, 1990.

[43] R.A. 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.

[44] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):pp. 155–176, 1996.

[45] T. Hebert and R. Leahy. A generalized EM algorithm for 3-D bayesian reconstruction from poisson data using gibbs priors. *Medical Imaging, IEEE Transactions on*, 8(2):194–202, 1989.

[46] U. Helmke and J.B. Moore. *Optimization and dynamical systems*. Springer, 1994.

[47] H. Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010.

[48] P.O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565. IEEE, 2002.

[49] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[50] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, 2001.

[51] IBM ILOG CPLEX Optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`.

[52] I.T. Jolliffe. *Principal component analysis*, volume 487. Springer-Verlag New York, 1986.

[53] I.T. Jolliffe, N.T. Trendafilov, and M. Uddin. A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12(3):531–547, 2003.

[54] A. Kapteyn, H. Neudecker, and T. Wansbeek. An approach to n-mode components analysis. *Psychometrika*, 51(2):269–275, 1986.

[55] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[56] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[57] J. Kim and H. Park. Sparse nonnegative matrix factorization for clustering. *Technical report, Georgia Institute of Technology*, 2008.

[58] Y.D. Kim and S. Choi. Nonnegative Tucker decomposition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[59] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM REVIEW*, 51(3):455–500, 2009.

[60] P.M. Kroonenberg and J.D. Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[61] W.H. Lawton and E.A. Sylvestre. Self modeling curve resolution. *Technometrics*, 13(3):617–633, 1971.

[62] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):789, 1999.

[63] H. Lee and S. Choi. Group nonnegative matrix factorization for eeg classification. *Journal of Machine Learning Research - Proceedings Track*, 5:320–327, 2009.

[64] S. Lemm, B. Blankertz, G. Curio, and K.R. Muller. Spatio-spectral filters for improving the classification of single trial EEG. *Biomedical Engineering, IEEE Transactions on*, 52(9):1541 –1548, sept. 2005.

[65] F. Liese and K.J. Miescke. *Statistical Decesion Theory: Estimation, Testing, and Selection*. Statistics. Springer, 2008.

[66] E.F. Lock, K.A. Hoadley, J.S. Marron, and A.B. Nobel. Joint and individual variation explained (JIVE) for integrated analysis of multiple datatypes. *The Annals of Applied Statistics*, 7(1):523–542, 2013.

[67] M. Loog and R.P.W. Duin. Linear dimensionality reduction via a heteroscedastic extension of LDA: The chernoff criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:732–739, 2004.

[68] D.G. Luenberger. *Optimization by vector space methods*. John Wiley and sons, 1969.

[69] M.S. Mahanta, A.S. Aghaei, K.N. Plataniotis, and S. Pasupathy. Heteroscedastic linear feature extraction based on sufficiency conditions. *Pattern Recognition*, 45(2):821 – 830, 2012.

[70] M.S. Mahanta and K.N. Plataniotis. A heteroscedastic extension of LDA based on multi-class matusita affinity. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1921–1924, 2012.

[71] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[72] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.R. Muller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41 –48, August 1999.

[73] V. Moraru. An algorithm for solving quadratic programming problems. *Computer Science Journal of Moldova*, 5(2):223–235, 1997.

[74] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*, 12(2):181–201, 2001.

[75] J. Muller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Clinical Neurophysiology*, 110(5):787 – 798, 1999.

[76] C.H. Nguyen and H. Mamitsuka. Latent feature kernels for link prediction on sparse graphs. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(11):1793–1804, 2012.

[77] Q. Novi, C. Guan, T.H. Dat, and P. Xue. Sub-band common spatial pattern (SBCSP) for brain-computer interface. In *Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on*, pages 204 –207, may 2007.

[78] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[79] E. Parkhomenko. *Sparse canonical correlation analysis*. PhD thesis, University of Toronto, 2008.

[80] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R.D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsnmf). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):403–415, 2006.

[81] X. Peng and Y. Wang. Geometric algorithms to large margin classifier based on affine hulls. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(2):236–246, 2012.

[82] A.H. Phan and A. Cichocki. Tensor decompositions for feature extraction and classification of high dimensional datasets. *IEICE, NOLTA*, 1(1):37–68, 2010.

[83] A.H. Phan and A. Cichocki. Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*, 74(11):1956 – 1969, 2011.

[84] A.H. Phan and A. Cichocki. Fast and efficient algorithms for nonnegative tucker decomposition. In *Advances in Neural Networks - ISNN 2008*, volume 5264 of *Lecture Notes in Computer Science*, pages 772–782. Springer Berlin Heidelberg, 2008.

[85] A.H. Phan and A. Cichocki. Local learning rules for nonnegative tucker decomposition. In *Neural Information Processing*, volume 5863 of *Lecture Notes in Computer Science*, pages 538–545. Springer Berlin Heidelberg, 2009.

[86] G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for adaboost. *Tech. Rep. NC-TR-1998-021, Royal Holloway College, University of London, UK*, 42(3):287–320, 1998.

[87] J.D.M. Rennie. Maximum-margin logistic regression. `http://people.csail.mit.edu/jrennie/writing`, February 2005.

[88] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, Nov. 1958.

[89] L. Rueda and M. Herrera. Linear dimensionality reduction by maximizing the chernoff distance in the transformed space. *Pattern Recognition*, 41(10):3138–3152, 2008.

[90] H. Sakano, T. Ohashi, A. Kimura, H. Sawada, and K. Ishiguro. Extended Fisher criterion based on auto-correlation matrix information. In *SSPR/SPR'12*, pages 409–416, 2012.

[91] A. Sierra. High-order Fisher's discriminant analysis. *Pattern Recognition*, 2002.

[92] M. Swanson. *Path Integrals and Quantum Processes*. Academic Press, CA, 1992.

[93] D. Tao, X. Li, X. Wu, and S.J. Maybank. General averaged divergence analysis. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 302–311, 2007.

[94] F.J. Theis, K. Stadlthanner, and T. Tanaka. First results on uniqueness of sparse non-negative matrix factorization. In *Proceedings of the 13th European Signal Processing Conference (EUSIPCO '05)*, 2005.

[95] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[96] R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1):18–29, 2008.

[97] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. Winston & Sons, 1977.

[98] L.R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. In C. W. Harris, editor, *Problems in measuring change.*, pages 122–137. University of Wisconsin Press, Madison WI, 1963.

[99] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[100] M.H.V. Benthem and M.R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of chemometrics*, 18(10):441–450, 2004.

[101] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

[102] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

[103] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[104] S. Watanabe and N. Pakvasa. Subspace method of pattern recognition. In *Proceedings of 1st International Joint Conference of Pattern Recognition*, pages 25–32, 1973.

[105] D.M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[106] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[107] T. Yokota, A. Cichocki, and Y. Yamashita. Linked PARAFAC/CP tensor decomposition and its fast implementation for multi-block tensor analysis. In *Neural Information Processing*, volume 7665, pages 84–91. Springer, 2012.

[108] T. Yokota, T. Wakahara, and Y. Yamashita. Heteroscedastic Gaussian based correction term for Fisher discriminant analysis and its kernel extension. In *Proceedings of IJCNN2013*, pages 662–669. IEEE, 2013.

[109] T. Yokota and Y. Yamashita. Quadratically constrained maximum a posteriori estimation for binary classifier. In *Machine Learning and Data Mining in Pattern Recognition*, volume 6871, pages 1–15. Springer, 2011.

[110] T. Yokota and Y. Yamashita. Support vector machines with weighted regularization. In *Neural Information Processing*, volume 7063, pages 471–480. Springer, 2011.

[111] T. Yokota and Y. Yamashita. A quadratically constrained MAP classifier using the mixture of Gaussians models as a weight function. *IEEE Transactions on Neural Networks and Learning Systems*, 24(7):1127–1140, 2013.

[112] T. Yokota, T. Wakahara, H. Sakano, and Y. Yamashita. Correction term of Fisher discriminant analysis based on normal distribution. *IEICE Tech. Rep, IBISML2012-185*, 112(495):31–36, 2012.

[113] T. Yokota and Y. Yamashita. A new quadratically constrained criterion based on maximum a posteriori estimation for binary classifier. *IEICE Tech. Rep, IBISML2010-85*, 110(265):187–194, 2010.

[114] R. Zdunek. Regularized active set least squares algorithm for nonnegative matrix factorization in application to raman spectra separation. In *Advances in Computational Intelligence*, pages 492–499. Springer, 2011.

[115] R. Zdunek. Approximation of feature vectors in nonnegative matrix factorization with gaussian radial basis functions. In *Proceedings of the 19th international conference on Neural Information Processing - Volume Part I*, ICONIP'12, pages 616–623, Berlin, Heidelberg, 2012. Springer-Verlag.

[116] R. Zdunek and A. Cichocki. Gibbs regularized nonnegative matrix factorization for blind separation of locally smooth signals. In *15th IEEE International Workshop on Nonlinear Dynamics of Electronic Systems (NDES 2007), Tokushima, Japan*, pages 317–320, 2007.

[117] R. Zdunek and A. Cichocki. Blind image separation using nonnegative matrix factorization with gibbs smoothing. In *Neural information processing*, pages 519–528. Springer, 2008.

[118] S. Zhang and T. Sim. Discriminant subspace analysis: a Fukunaga-Koontz approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1732–1745, 2007.

[119] M. Zhu and A.M. Martinez. Subclass discriminant analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1274 –1286, aug. 2006.

[120] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[121] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

[122] M. Kojima, S. Mizuno, T. Tsutiya, and H. Yabe. *Interior point methods.* Asakura Shoten, 2001. (in Japanese)

[123] S. Akaho. *Multivariate kernel analysis -new approach for nonlinear data analysis-.* Iwanami Shoten, 2008. (in Japanese)