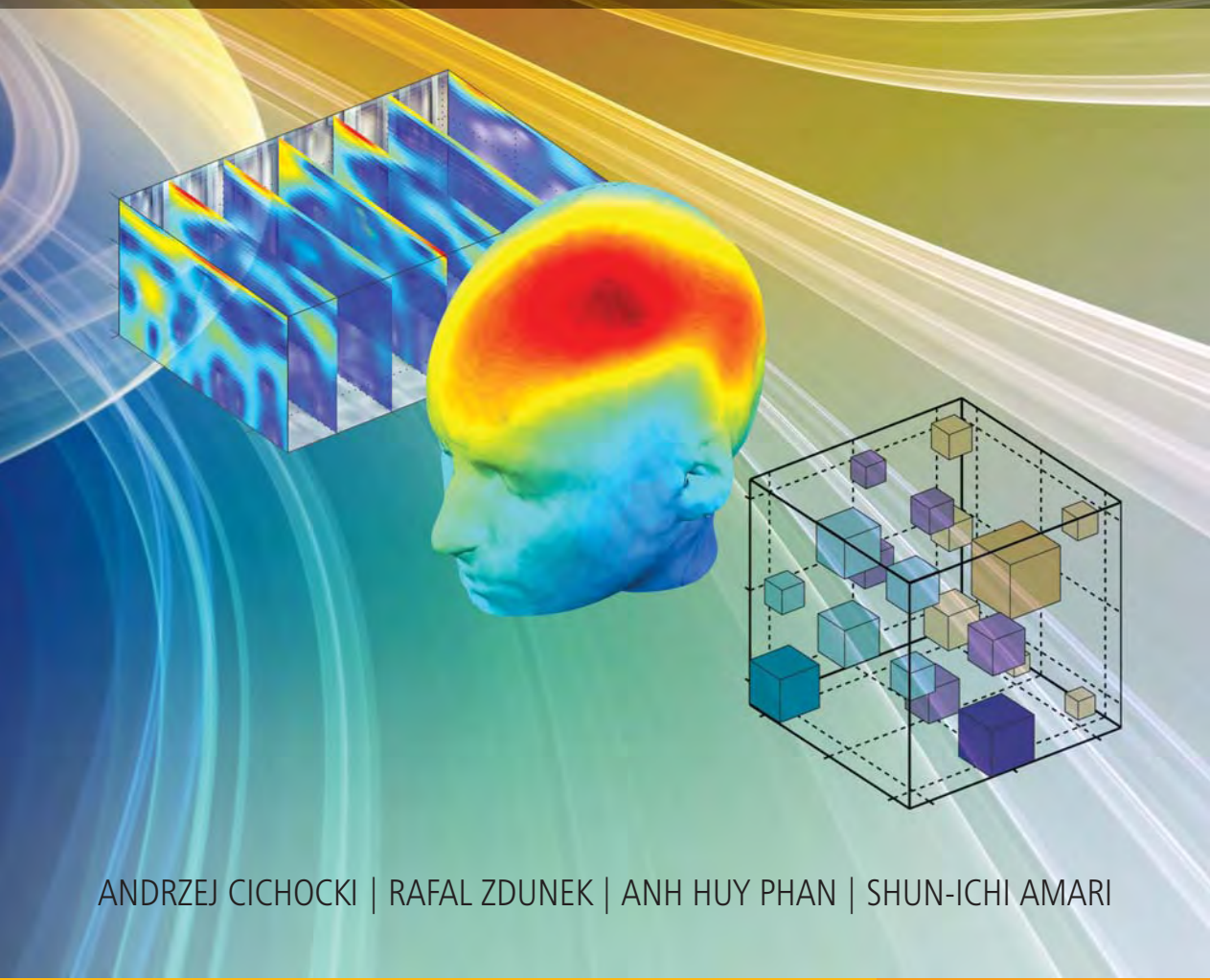
Nonnegative Matrix and Tensor Factorizations

Applications to Exploratory Multi-way Data Analysis and Blind Source Separation



ANDRZEJ CICHOCKI | RAFAL ZDUNEK | ANH HUY PHAN | SHUN-ICHI AMARI

WILEY

Nonnegative Matrix and Tensor Factorizations

Applications to Exploratory Multiway Data Analysis and Blind Source Separation
(Preprint)

Andrzej CICHOCKI
Rafal ZDUNEK
Anh Huy PHAN
Shun-ichi AMARI

.

RIKEN Brain Science Institute, JAPAN
Warsaw University of Technology,
Systems Research Institute PAN, POLAND
Wroclaw University of Technology, POLAND

Tokyo

NONNEGATIVE MATRIX AND TENSOR FACTORIZATIONS

APPLICATIONS TO EXPLORATORY MULTI-WAY DATA ANALYSIS AND BLIND SOURCE SEPARATION

Andrzej Cichocki

Laboratory for Advanced Brain Signal Processing, Riken Brain Science Institute, Japan; and Warsaw University of Technology and Systems Research Institute, PAN, Poland

Rafal Zdunek

Institute of Telecommunications, Teleinformatics and Acoustics, Wroclaw University of Technology, Poland; and Riken Brain Science Institute, Japan

Anh Huy Phan

Laboratory for Advanced Brain Signal Processing, Riken Brain Science Institute, Japan

Shun-ichi Amari

Research Unit for Mathematical Neuroscience, Riken Brain Science Institute, Japan

WILEY

A John Wiley and Sons, Ltd, Publication

This edition first published 2009

Registered office
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA, 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

Library of Congress Cataloging-in-Publication Data

Non-negative matrix and tensor factorizations : applications to exploratory multiway data analysis and blind source separation / Andrzej Cichocki . . . [et al.].
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-74666-0 (cloth)
1. Computer algorithms. 2. Data mining. 3. Machine learning 4. Data structures (Computer science) I. Cichocki, Andrzej.
QA76.9.A43N65 2009
005.1–dc22 2009016049

A catalogue record for this book is available from the British Library.

ISBN: 9780470746660

Set in 9/11 pt Times by Thomson Digital, Noida, India
Printed in Singapore by Fabulous

Contents

Preface *xvii*

Glossary of Symbols and Abbreviations *xxiii*

1 Introduction – Problem Statements and Models *1*

1.1 Blind Source Separation and Linear Generalized Component Analysis *2*

1.2 Matrix Factorization Models with Nonnegativity and Sparsity Constraints *8*

1.2.1 Why Nonnegativity and Sparsity Constraints? *8*

1.2.2 Basic NMF Model *9*

1.2.3 Symmetric NMF *10*

1.2.4 Semi-Orthogonal NMF *11*

1.2.5 Semi-NMF and Nonnegative Factorization of Arbitrary Matrix *11*

1.2.6 Three-factor NMF *11*

1.2.7 NMF with Offset (Affine NMF) *14*

1.2.8 Multi-layer NMF *15*

1.2.9 Simultaneous NMF *15*

1.2.10 Projective and Convex NMF *16*

1.2.11 Kernel NMF *17*

1.2.12 Convolutive NMF *17*

1.2.13 Overlapping NMF *19*

1.3 Basic Approaches to Estimate Parameters of Standard NMF 20
1.3.1 Large-Scale NMF 23
1.3.2 Non-uniqueness of NMF and Techniques to Alleviate the Ambiguity Problem 24
1.3.3 Initialization of NMF 26
1.3.4 Stopping Criteria 28
1.4 Tensor Properties and Basis of Tensor Algebra 28
1.4.1 Tensors (Multi-way Arrays) – Preliminaries 28
1.4.2 Subarrays, Tubes and Slices 29
1.4.3 Unfolding – Matricization 29
1.4.4 Vectorization 32
1.4.5 Outer, Kronecker, Khatri-Rao and Hadamard Products 33
1.4.6 Mode-n Multiplication of Tensor by Matrix and Tensor by Vector, Contracted Tensors Product 36
1.4.7 Special Forms of Tensors 41
1.5 Tensor Decompositions and Factorizations 42
1.5.1 Why Multi-way Array Decompositions and Factorizations? 43
1.5.2 PARAFAC and Nonnegative Tensor Factorization 45
1.5.3 NTF1 Model 51
1.5.4 NTF2 Model 53
1.5.5 Individual Differences in Scaling (INDSCAL) and Implicit Slice Canonical Decomposition Model (IMCAND) 56
1.5.6 Shifted PARAFAC and Convolutive NTF 57
1.5.7 Nonnegative Tucker Decompositions 59
1.5.8 Block Component Decompositions 63
1.5.9 Block-Oriented Decompositions 67
1.5.10 PARATUCK2 and DEDICOM Models 68
1.5.11 Hierarchical Tensor Decomposition 71
1.6 Discussion and Conclusions 72
Appendix Uniqueness Conditions for Three-way Tensor Factorizations 72
Appendix Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) with Sparsity and/or Nonnegativity Constraints 73
B.1 Standard SVD and PCA 74
B.2 Sparse PCA 76
B.3 Nonnegative PCA 77
Appendix Determining a True Number of Components 78

Appendix Nonnegative Rank Factorization Using Wedderborn Theorem – Estimation of the Number of Components 81

References 83

2 Similarity Measures and Generalized Divergences 95
2.1 Error-induced Distance and Robust Regression Techniques 97
2.2 Robust Estimation 98
2.3 Csiszár Divergences 104
2.4 Bregman Divergence 111
2.4.1 Bregman Matrix Divergences 119
2.5 Alpha-Divergences 120
2.5.1 Asymmetric Alpha-Divergences 120
2.5.2 Symmetric Alpha-Divergences 127
2.6 Beta-Divergences 129
2.7 Gamma-Divergences 133
2.8 Divergences Derived from Tsallis and Rényi Entropy 135
2.8.1 Concluding Remarks 140
Appendix Information Geometry, Canonical Divergence, and Projection 141
A.0.2 Space of Probability Distributions 141
A.0.3 Geometry of Space of Positive Measures 145
Appendix Probability Density Functions for Various Distributions 148

References 151

3 Multiplicative Algorithms for NMF with Sparsity Constraints 155
3.1 Extended ISRA and EMML Algorithms: Regularization and Sparsity 156
3.1.1 Multiplicative NMF Algorithms Based on the Squared Euclidean Distance 157
3.1.2 Multiplicative NMF Algorithms Based on Kullback-Leibler I-Divergence 164
3.2 Multiplicative Algorithms Based on Alpha-Divergence 168
3.2.1 Multiplicative Alpha NMF Algorithm 168
3.2.2 Generalized Multiplicative Alpha NMF Algorithms 172
3.3 Alternating SMART: Simultaneous Multiplicative Algebraic Reconstruction Technique 173
3.3.1 Alpha SMART Algorithm 173
3.3.2 Generalized SMART algorithms 177

3.4 Multiplicative NMF Algorithms Based on Beta-Divergence 180
3.4.1 Multiplicative Beta NMF algorithm 180
3.4.2 Multiplicative Algorithm Based on the Itakura-Saito Distance 181
3.4.3 Generalized Multiplicative Beta Algorithm for NMF 183
3.5 Algorithms for Semi-orthogonal NMF and Orthogonal Three-Factor NMF 184
3.6 Multiplicative Algorithms for Affine NMF 186
3.7 Multiplicative Algorithms for Convolutive NMF 187
3.7.1 Multiplicative Algorithm for Convolutive NMF Based on Alpha-Divergence 189
3.7.2 Multiplicative Algorithm for Convolutive NMF Based on Beta-Divergence 189
3.7.3 Efficient Implementation of CNMF Algorithm 192
3.8 Simulation Examples for Standard NMF 193
3.9 Examples for Affine NMF 202
3.10 Music Analysis and Decomposition Using Convolutive NMF 205
3.11 Discussion and Conclusions 215
Appendix Fast Algorithms for Large-scale Data 216
A.1 Random Block-wise Processing Approach – Large Scale NMF 216
A.2 Multi-layer Procedure 217
A.3 Parallel Processing 217
Appendix Performance Evaluation 217
B.1 Signal-to-Interference-Ratio - SIR 218
B.2 Peak Signal-to-Noise-Ratio (PSNR) 219
Appendix Convergence Analysis of the Multiplicative Alpha NMF Algorithm 221
Appendix MATLAB Implementation of the Multiplicative NMF Algorithms 223
D.1 Alpha Algorithm 223
D.2 SMART Algorithm 225
D.3 ISRA Algorithm for aNMF 227
Appendix Additional MATLAB Functions 228
E.1 Multi-layer NMF 228
E.2 MC Analysis with Distributed Computing Tool 229

References 231

4 ALS and Related Algorithms for NMF and SCA Problems 237

4.1 Standard ALS Algorithm 238
4.1.1 Multiple Linear Regression – Vectorized Version of ALS Update Formulas 240
4.1.2 Weighted ALS 241
4.2 Methods for Improving Performance and Convergence Speed of ALS Algorithms 242
4.2.1 ALS Algorithm for Large-Scale NMF 242
4.2.2 ALS Algorithm with Line-Search 243
4.2.3 Acceleration of ALS Algorithm via Simple Regularization 243
4.3 ALS Algorithm with Flexible and Generalized Regularization Terms 244
4.3.1 ALS with Tikhonov Type Regularization Terms 244
4.3.2 ALS Algorithms with Sparsity Control and Decorrelation 246
4.4 Combined Generalized Regularized ALS Algorithms 247
4.5 Wang-Hancewicz Modified ALS Algorithm 248
4.6 Implementation of Regularized ALS Algorithms for NMF 249
4.7 HALS Algorithm and its Extensions 250
4.7.1 Projected Gradient Local Hierarchical Alternating Least Squares (HALS) Algorithm 250
4.7.2 Extensions and Implementations of the HALS Algorithm 252
4.7.3 Fast HALS NMF Algorithm for Large-Scale Problems 254
4.7.4 HALS NMF Algorithm with Sparsity, Smoothness and Uncorrelatedness Constraints 256
4.7.5 HALS Algorithm for Sparse Component Analysis and Flexible Component Analysis 258
4.7.6 Simplified HALS Algorithm for Distributed and Multi-task Compressed Sensing 263
4.7.7 Generalized HALS-CS Algorithm 268
4.7.8 Generalized HALS Algorithms Using Alpha-Divergence 270
4.7.9 Generalized HALS Algorithms Using Beta-Divergence 271
4.8 Simulation Results 273
4.8.1 Underdetermined Blind Source Separation Examples 273
4.8.2 NMF with Sparseness, Orthogonality and Smoothness Constraints 274
4.8.3 Simulations for Large Scale NMF 278
4.8.4 Illustrative Examples for Compressed Sensing 278

4.9 Discussion and Conclusions 287
Appendix MATLAB Source Code for ALS Algorithm 291
Appendix MATLAB Source Code for Regularized ALS Algorithms 292
Appendix MATLAB Source Code for Mixed ALS-HALS Algorithms 295
Appendix MATLAB Source Code for HALS CS Algorithm 298
Appendix Additional MATLAB Functions 300

References 303

5 Projected Gradient Algorithms 309
5.1 Oblique Projected Landweber (OPL) Method 311
5.2 Lin's Projected Gradient (LPG) Algorithm with Armijo Rule 311
5.3 Barzilai-Borwein Gradient Projection for Sparse Reconstruction (GPSR-BB) 314
5.4 Projected Sequential Subspace Optimization (PSESOP) 316
5.5 Interior Point Gradient (IPG) Algorithm 316
5.6 Interior Point Newton (IPN) Algorithm 318
5.7 Regularized Minimal Residual Norm Steepest Descent Algorithm (RMRNSD) 321
5.8 Sequential Coordinate-Wise Algorithm (SCWA) 324
5.9 Simulations 326
5.10 Discussions 334
Appendix Stopping Criteria 334
Appendix MATLAB Source Code for Lin's PG Algorithm 336

References 339

6 Quasi-Newton Algorithms for Nonnegative Matrix Factorization 343
6.1 Projected Quasi-Newton Optimization 344
6.1.1 Projected Quasi-Newton for Frobenius Norm 344
6.1.2 Projected Quasi-Newton for Alpha-divergence 346
6.1.3 Projected Quasi-Newton for Beta-divergence 351
6.1.4 Practical Implementation 353
6.2 Gradient Projection Conjugate Gradient 354
6.3 FNMA algorithm 356
6.4 NMF with Quadratic Programming 359
6.4.1 Nonlinear Programming 360
6.4.2 Quadratic Programming 361
6.4.3 Trust-region Subproblem 363

6.4.4 Updates for A 365
6.5 Hybrid Updates 367
6.6 Numerical Results 368
6.7 Discussions 370
Appendix Gradient and Hessian of Cost Functions 373
Appendix MATLAB Source Codes 374

References 385

7 Multi-Way Array (Tensor) Factorizations and Decompositions 389
7.1 Learning Rules for the Extended Three-way NTF1 Problem 390
7.1.1 Basic Approaches for the Extended NTF1 Model 390
7.1.2 ALS Algorithms for NTF1 393
7.1.3 Multiplicative Alpha and Beta Algorithms for the NTF1 Model 394
7.1.4 Multi-layer NTF1 Strategy 395
7.2 Algorithms for Three-way Standard and Super Symmetric Nonnegative Tensor Factorization 396
7.2.1 Multiplicative NTF Algorithms Based on Alpha- and Beta-Divergences 398
7.2.2 Simple Alternative Approaches for NTF and SSNTF 403
7.3 Nonnegative Tensor Factorizations for Higher-Order Arrays 404
7.3.1 Alpha NTF Algorithm 407
7.3.2 Beta NTF Algorithm 408
7.3.3 Fast HALS NTF Algorithm Using Squared Euclidean Distance 409
7.3.4 Generalized HALS NTF Algorithms Using Alpha- and Beta-Divergences 412
7.3.5 Tensor Factorization with Additional Constraints 415
7.4 Algorithms for Nonnegative and Semi-Nonnegative Tucker Decompositions 416
7.4.1 Higher Order SVD (HOSVD) and Higher Order Orthogonal Iteration (HOOI) Algorithms 417
7.4.2 ALS Algorithm for Nonnegative Tucker Decomposition 420
7.4.3 HOSVD, HOOI and ALS Algorithms as Initialization Tools for Nonnegative Tensor Decomposition 420
7.4.4 Multiplicative Alpha Algorithms for Nonnegative Tucker Decomposition 422

7.4.5 Beta NTD Algorithm 424
7.4.6 Local ALS Algorithms for Nonnegative TUCKER Decompositions 426
7.4.7 Semi-Nonnegative Tucker Decomposition 430
7.5 Nonnegative Block-Oriented Decomposition 430
7.5.1 Multiplicative Algorithms for NBOD 431
7.6 Multi-level Nonnegative Tensor Decomposition - High Accuracy Compression and Approximation 432
7.7 Simulations, Illustrative Examples and Applications 433
7.7.1 Experiments for Nonnegative Tensor Factorizations 434
7.7.2 Experiments for Nonnegative Tucker Decomposition 442
7.7.3 Experiments for Nonnegative Block-Oriented Decomposition 449
7.7.4 Multi-Way Analysis of High Density Array EEG – Classification of Event Related Potentials 451
7.7.5 Application of Tensor Decompositions in BCI 462
7.7.6 Image and Video Applications 468
7.8 Discussion and Conclusions 472
Appendix Evaluation of Interactions and Relationships Among Hidden Components for NTD model 473
Appendix Computation of a Reference Tensor 475
Appendix Trilinear and Direct Trilinear Decompositions for Efficient Initialization 477
Appendix MATLAB Source Code for Alpha NTD Algorithm 479
Appendix MATLAB Source Code for Beta NTD Algorithm 481
Appendix MATLAB Source Code for HALS NTD Algorithm 483
Appendix MATLAB Source Code for ALS NTF1 Algorithm 485
Appendix MATLAB Source Code for ISRA BOD Algorithm 486
Appendix Additional MATLAB functions 487

References 489

8 Selected Applications 497
8.1 Clustering 497
8.1.1 Semi-Binary NMF 498
8.1.2 NMF vs. Spectral Clustering 499
8.1.3 Clustering with Convex NMF 502
8.1.4 Application of NMF to Text Mining 503
8.1.5 Email Surveillance 505
8.2 Classification 508

8.2.1 Musical Instrument Classification *508*
8.2.2 Image Classification *509*
8.3 Spectroscopy *513*
8.3.1 Raman Spectroscopy *514*
8.3.2 Fluorescence Spectroscopy *514*
8.3.3 Hyperspectral Imaging *516*
8.3.4 Chemical Shift Imaging *521*
8.4 Application of NMF for Analyzing Microarray Data *523*
8.4.1 Gene Expression Classification *523*
8.4.2 Analysis of Time Course Microarray Data *526*

References *535*

Index *546*

Preface

Signal processing, data analysis and data mining are pervasive throughout science and engineering. Extracting an interesting knowledge from experimental raw datasets, measurements, observations and understanding complex data has become an important challenge and objective. Often datasets collected from complex phenomena represent the integrated result of several inter-related variables or they are combinations of underlying latent components or factors. Such datasets can be first decomposed or separated into the components that underlie them in order to discover structures and extract hidden information. In many situations, the measurements are gathered and stored as data matrices or multi-way arrays (tensors), and described by linear or multi-linear models.

Approximative low-rank matrix and tensor factorizations or decompositions play a fundamental role in enhancing the data and extracting latent components. A common thread in various approaches for noise removal, model reduction, feasibility reconstruction, and Blind Source Separation (BSS) is to replace the original data by a lower dimensional approximate representation obtained via a matrix or multi-way array factorization or decomposition. The notion of a matrix factorization arises in a wide range of important applications and each matrix factorization makes a different assumption regarding component (factor) matrices and their underlying structures, so choosing the appropriate one is critical in each application domain. Very often the data, signals or images to be analyzed are nonnegative (or partially nonnegative), and sometimes they also have sparse or smooth representation. For such data, it is preferable to take these constraints into account in the analysis to extract nonnegative and sparse/smooth components or factors with physical meaning or reasonable interpretation, and thereby avoid absurd or unpredictable results. Classical tools cannot guarantee to maintain the nonnegativity.

In this research monograph, we provide a wide survey of models and algorithmic aspects of Nonnegative Matrix Factorization (NMF), and its various extensions and modifications, especially the Nonnegative Tensor Factorization (NTF) and the Nonnegative Tucker Decomposition (NTD). In the NTF and NTD approaches high-dimensional data, such as hyper-spectral or medi-

cal images are factored or decomposed directly and they are approximated by a sum of rank-one nonnegative tensors. The motivation behind NMF, NTF and NTD is that besides the dimensionality reduction sought in many applications, the underlying data ensemble is nonnegative and can be better modeled and interpreted by means of nonnegative and, preferably, also sparse or smooth components.

The notions of NMF, NTF and NTD play a major role in a wide range of important applications, including bioinformatics, micro-array analysis, neuroscience, text mining, image understanding, air pollution research, chemometrics, and spectral data analysis. Nonnegative matrix and tensor factorizations and decompositions have many other applications, such as linear sparse coding, image classification, clustering, neural learning process, sound recognition, remote sensing, and object characterization. For example, NMF/NTF processing permits the detection alternative or context-dependent patterns of gene expression in complex biological systems and especially to recover meaningful biological information from cancer-related microarray data. We believe that a potential impact of the NMF and its extensions on scientific advancements might be as great as the Independent Component Analysis (ICA) or the Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). In contrast to ICA or SVD/PCA approaches, NMF/NTF and NTD techniques, if successively realized, may improve interpretability and visualization of large-scale data while maintaining the physical feasibility more closely.

Researchers from various research fields are interested in different, usually very diverse aspects, of NMF and NTF. For example, neuroscientists and biologists need reliable methods and techniques which can extract or separate useful information from superimposed biomedical data corrupted by a large level of noise and interference, for example, by using non-invasive recordings of human brain activities targeted at understanding the ability of the brain to sense, recognize, store and recall patterns and comprehending crucial elements of learning: association, abstraction and generalization. A second group of researchers: engineers and computer scientists, are fundamentally interested in developing and implementing flexible and efficient algorithms for specific practical engineering and scientific applications. A third group of researchers: mathematicians and physicists, have an interest in the development of a fundamental theory to understand mechanisms, properties and abilities of the developed algorithms, and their generalizations to more complex and sophisticated models. The interactions among such groups permits a real progress to be made in this very interdisciplinary research devoted to NMF/NTF and NTD, and each group benefits from the others.

The theory built up around NMF, NTF and NTD is so extensive and the applications are so numerous that we are, of course, not able to cover all of them. Our selection and treatment of material reflects our background and our own research interests and results in this fascinating area over the last five years.

The book provides a wide coverage of the models and algorithms for nonnegative matrix factorizations and tensor decompositions both from a theoretical and practical point of view. The main objective is to derive and implement in MATLAB efficient and relatively simple iterative algorithms that work well in practice for real-world data. In fact, almost all the algorithms presented in the book have been implemented in MATLAB and extensively tested. We have attempted to present the concepts, models and algorithms in general or flexible forms to stimulate a reader to be creative in visualizing new approaches and adopt methods or algorithms for their specific needs and applications.

In *Chapter 1* we describe the basic NMF models and their extensions, and formulate the fundamental problems related to the calculation of component (factor) matrices. A special emphasis is given to basic properties and mathematical operations for multi-way arrays, also called

multi-dimensional matrices or tensors. Chapter 1 introduces the basic linear and multi-linear models for matrix factorizations and tensor decompositions, and formulates the fundamental analytical framework for the solution of the problems posed in this book. The workhorse is the NMF algorithms for sparse representations of data, and its extensions, including the multi-layer NMF, semi-NMF, sparse NMF, tri-NMF, symmetric NMF, orthogonal NMF, non-smooth NMF (nsNMF), overlapping NMF, convolutive NMF (CNMF), and large-scale NMF. Our particular emphasis is on the NMF and semi-NMF models and their extensions to multi-way models (i.e., multi-linear models which perform multi-way array (tensor) decompositions) with nonnegativity and sparsity constraints, especially NTD, together with nonnegative and semi-nonnegative tensor factorizations that are mostly based on a family of Tucker and PARAFAC models.

In *Chapter 2*, we give an overview and discuss properties of a large family of generalized and flexible divergences or similarity distances between two nonnegative sequences or patterns. They are formulated for probability distributions used in the development of novel algorithms for NMF and NTF. Information theory, convex analysis, and information geometry play key roles in the formulation of the divergences. The scope of these results is vast since the generalized divergence functions and their variants include quite a large number of useful loss functions, including those based on relative entropies, generalized Kullback-Leibler or I-divergence, Hellinger distance, Jensen-Shannon divergence, J-divergence, Pearson and Neyman Chi-squared divergences, triangular discrimination, and arithmetic-geometric Taneya divergence. Many of these measures belong to the class of Alpha-divergences and Beta-divergences, and have been applied successfully in the disciplines such as signal processing, pattern recognition, probability distributions, information theory, finance and economics. In the following chapters we will apply such divergences as cost functions (possibly with additional constraints and regularization terms) to derive novel multiplicative and additive projected gradient and fixed-point algorithms. They provide working solutions for the problems where nonnegative latent (hidden) components can be generally statistically dependent, and satisfy some other conditions or additional constraints such as sparsity or smoothness.

In *Chapter 3*, we introduce a wide family of iterative multiplicative algorithms for NMF and related problems, subject to additional constraints such as sparsity and/or smoothness. Although a standard multiplicative update rule for NMF achieves a sparse representation of its components, we can impose a control over the sparsity of the components by designing a suitable cost function with additional penalty terms. In this chapter we consider a wide class of cost functions or divergences, leading to generalized multiplicative algorithms with regularization and/or penalty terms. Such relaxed forms of the multiplicative NMF algorithms usually provide better performance and convergence speed, and allows us to extract the desired components uniquely up to the scale and permutation ambiguities. As special cases we introduce the multiplicative algorithms for Alpha and Beta divergences, squared Hellinger, Pearson's Chi-squared, and Itakura-Saito distances.

In *Chapter 4*, we derive and give an overview of the Alternating Least Squares algorithms referred to as the ALS algorithms for NMF, semi-NMF, and multi-layer NMF. This is important as many existing NMF techniques are prohibitively slow and inefficient, especially for large-scale problems. For such problems a promising approach is to apply the ALS algorithms with the regularization and/or extended line search techniques. A special emphasis in this chapter is put on various regularization and penalty terms together with local learning rules. By incorporating the regularization and penalty terms into the weighted Frobenius norm, we show that it is possible to achieve sparse, orthogonal, or smooth representations, thus helping to obtain a desired global solution. The main objective of this chapter is to develop efficient and robust Regularized ALS (RALS) algorithms. For this purpose, we use several approaches from constrained opti-

mization and regularization theory, and in addition introduce several heuristic algorithms. The proposed algorithms are characterized by improved efficiency and often very good convergence properties, especially for large-scale problems.

In *Chapter 5*, we present a wide class of Projected Gradient (PG) algorithms and compare their performance, especially for large-scale problems. In contrast to the multiplicative NMF algorithms discussed in Chapter 3, the PG algorithms have additive updates, and provide an approximate solution to the Non-Negative Least Squares (NNLS) problems. Chapter 5 focuses on the following PG algorithms: Oblique Projected Landweber (OPL), projected gradients with Armijo rule, Barzilai-Borwein gradient projection, Projected Sequential Subspace Optimization (PSESOP), Interior-Point Newton (IPN), Minimal Residual Norm Steepest Descent (MRNSD), and Sequential Coordinate-Wise Algorithm (SCWA).

In *Chapter 6*, we introduce learning algorithms for NMF, using second-order approximations, that is, the information about the Hessian and gradient of a cost function. Using the information about the curvature of the cost function, which is intimately related to second-order derivatives, the convergence can be considerably accelerated. This, however, also introduces many related practical problems that must be addressed prior to applying learning algorithms. For example, the Hessian must be positive-definite to ensure the convergence of approximations to a local minimum of a specific cost function. Unfortunately, this is not guaranteed using the NMF alternating minimization rule, and we need to resort to some suitable Hessian approximation techniques. In addition, the Hessian values may be very large, and of a severely ill-conditioned nature (in particular for large-scale problems), which gives rise to many difficult problems related to its inversion. Moreover, the nonlinear projections may be performed in many ways, similarly to the PG and steepest descent algorithms. This chapter provides a comprehensive study of the solutions to the above-mentioned problems. We also give some heuristics on the selection of a cost function and the related regularization terms which restrict the area of feasible solutions, and help the algorithms to converge to the global minimum of a specific cost function. In particular, we discuss the simplest approach to the projected quasi-Newton optimization using the Levenberg-Marquardt regularization of the Hessian. We then extend the discussion to more sophisticated second-order algorithms that iteratively update only the strictly positive (inactive) variables. The example includes the Gradient Projection Conjugate Gradient (GPCG). Furthermore, as a special case of the second-order method, we present one Quadratic Programming (QP) method that solves a QP problem using the trust-region subproblem algorithm. The QP problem is formulated from the Tikhonov regularized squared Euclidean cost function extended with a logarithmic barrier function to satisfy nonnegativity constraints. The BSS experiments demonstrate the high efficiency of the proposed algorithms.

In *Chapter 7*, we attempt to extend and generalize the results and algorithms from the previous chapters for the NTF and NTD models. In fact, almost all the NMF algorithms described in the earlier chapters can be extended or generalized to the various nonnegative tensor factorizations and decompositions formulated in Chapter 1. However, in this chapter we mainly focus on NTF, that is, PARAFAC with nonnegativity constraints and NTD. In order to make this chapter as self-contained as possible, we re-introduce some concepts and derive several novel and efficient algorithms for nonnegative and semi-nonnegative tensor (multi-way arrays) factorizations and decompositions. Our particular emphasis is on a detailed treatment of the generalized cost functions, including Alpha- and Beta-divergences. Based on these cost functions, several classes of algorithms are introduced, including: (1) multiplicative updating; (2) ALS; and (3) Hierarchical ALS (HALS). These algorithms are then incorporated into multi-layer hierarchical networks in order to improve their performance. A special emphasis is given on the ways to impose nonnegativity or semi-nonnegativity, together with optional constraints such as orthogo-

nality, sparsity and/or smoothness. The developed algorithms are tested for several applications such as denoising, compression, feature extraction, clustering, EEG data analysis, brain computer interface and video tracking. To understand the material in this chapter it would be helpful to be familiar with the previous chapters, especially Chapters 1, 3 and 4.

Finally, in *Chapter 8*, we briefly discuss the selected applications of NMF and multi-dimensional array decompositions, with a special emphasis on these applications to which the algorithms described in the previous chapters are applicable. We review the following applications: data clustering, text mining, email surveillance, musical instrument classification, face recognition, handwritten digit recognition, texture classification, Raman spectroscopy, fluorescence spectroscopy, hyper-spectral imaging, chemical shift imaging, and gene expression classification.

The book is partly a textbook and partly a research monograph. It is a textbook because it gives the detailed introduction to the basic models and algorithms of nonnegative and sparse matrix and tensor decompositions. It is simultaneously a monograph because it presents many new results, methods, ideas, models, further developments and implementation of efficient algorithms which are brought together and published in this monograph for the first time. As a result of its twofold character the book is likely to be of interest to graduate and postgraduate students, engineers and scientists working in the field of biomedical engineering, data analysis, data mining, multidimensional data visualization, signal/image processing, mathematics, computer science, finance, economics, optimization, geophysics, and neural computing. Furthermore, the book may also be of interest to researchers working in different areas of science, because a number of the results and concepts have been included which may be advantageous for their further research. One can read this book through sequentially but it is not necessary since each chapter is essentially self-contained, with as few cross references as possible. So, browsing is encouraged.

Acknowledgments

The authors would like to express their appreciation and gratitude to a number of researchers who helped them in a variety of ways, directly and also indirectly, in the development of this book.

First of all, we would like to express our sincere gratitude to Professor Susumu Tonegawa and Professor Keiji Tanaka directors of RIKEN Brain Science Institute for creating a great scientific environment for multidisciplinary research and promotion international scientific collaboration.

Although part of this book is derived from the research activities of the authors over the past five years on this subject, many influential results and approaches are developed in collaboration with our colleagues and researchers from the RIKEN Brain Science Institute, JAPAN, and several worldwide universities.

Many of them have made important and influential contributions. Special thanks and gratitude go to Professors Jonathon Chambers from Loughborough University - UK, Robert Plemmons from Wake Forest University USA, Seungjin Choi, Hyekyoung Lee and Yong-Deok Kim from Computer Science and Engineering at Pohang University of Science and Technology (POSTECH) - Korea, Danilo Mandic from Imperial College London, Saeid Sanei from Cardiff University - UK, Liqing Zhang, Qibin Zhao and Qiang Wu from Shanghai Jiao Tong University - China, Sergio A. Cruces-Alvarez from E.S. Ingenieros, University of Seville - Spain and Cesar Caiafa from Universidad de Buenos Aires - Argentina.

Those whose works have had some inspiration and impact in our book, and are reflected in this research monograph include Tamara G. Kolda, Brett W. Bader, Evrim Acar from Sandia National Laboratories in Livermore, Ivan Oseledets and Eugene Tyrtyshnikov from Russian Academy of Sciences, Morten Morup from the Technical University of Denmark and Lieven De Lathauwer from from the Katholieke Universiteit Leuven.

Over various phases of writing this book, several people have kindly agreed to read and comment on parts or all of the manuscript. We are very grateful for the insightful comments and suggestions to Danilo Mandic, Jonathon Chambers, Saeid Sanei, Evrim Acar Ataman, Cesar Caiafa, Sidney Lehky, Nicolas Gillis, Yoshikazu Washizawa, Noboru Murata, Ivica Kopriva, Abd-Krim Seghouane, Scott Douglas, Sergio Cruces, Wenwu Wang, Zhaoshui He, John Stutz and Paul Sajda.

Finally, we must acknowledge the help and understanding of our families and friends during the past two years while we carried out this challenging project.

A. CICHOCKI, R. ZDUNEK, A.H. PHAN, AND S. AMARI

May 15, 2009, Tokyo, JAPAN

Glossary of Symbols

Principal Symbols

$\mathbb{R}_+$	nonnegative real number
$\mathbb{R}^n$	n-dimensional real vector space
$\mathbf{A} = [a_{ij}] \in \mathbb{R}^{I \times J}$	I by J matrix (mixing, basis or dictionary matrix)
a_{ij}	ij-th element of the matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$
I	number of observations, sensors or available time series
J	number of components or common factors
$\mathbf{A}^{-1}$	inverse of a nonsingular matrix $\mathbf{A}$
$\mathbf{A}^{\dagger}$	Moore-Penrose pseudo-inverse of a matrix $\mathbf{A}$
$\mathbf{A} \geq \mathbf{0}$	nonnegative matrix $\mathbf{A}$ with all entries $a_{ij} \geq 0$
$\mathbf{A} \geq \boldsymbol{\varepsilon}$	positive matrix $\mathbf{A}$ with entries $a_{ij} \geq \varepsilon$, where ε is small positive constant
$\mathbf{X} \in \mathbb{R}^{J \times T}$	matrix representing hidden components
$\mathbf{Y} \in \mathbb{R}^{I \times T}$	matrix representing input data, e.g., measurements
$\mathbf{E} \in \mathbb{R}^{I \times T}$	matrix representing error or noise
$\arg\min_{\theta} J(\theta)$	denotes the value of θ that minimizes $J(\theta)$
x_i	i-th element of vector $\boldsymbol{x}$

$y_{it} = y_i(t)$	element of time series $y_i(t)$ for a time instant t or it-th entry of $\mathbf{Y}$
$[\mathbf{AX}]_j$	j-th column vector of the matrix $\hat{\mathbf{Y}} = \mathbf{AX}$
$[\mathbf{AX}]_{it}$	it-th element of the matrix $\hat{\mathbf{Y}} = \mathbf{AX}$
$\underline{\mathbf{A}} \in \mathbb{R}^{I \times J \times Q}$	three-way array (third-order tensor) representing basis matrices (slices)
a_{ijq}	ijq-th element of a three-way array $\underline{\mathbf{A}} \in \mathbb{R}^{I \times J \times Q}$
$\underline{\mathbf{G}} \in \mathbb{R}^{J \times R \times P}$	third-order core tensor representing links between factor matrices
g_{jrp}	jrp-th element of a three-way array $\underline{\mathbf{G}} \in \mathbb{R}^{J \times R \times P}$
$\underline{\mathbf{X}} \in \mathbb{R}^{J \times T \times Q}$	three-way array representing hidden components
x_{jtq}	jtq-th element of a three-way array $\underline{\mathbf{X}} \in \mathbb{R}^{J \times T \times Q}$
$\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$	three-way array (a third-order tensor) representing input (observed) data
y_{itq}	itq-th element of the three-way array $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$
$\underline{\mathbf{E}} \in \mathbb{R}^{I \times T \times Q}$	three-way array (third-order tensor) representing error or noise
e_{itq}	the itq-th element of the three-way array $\underline{\mathbf{E}} \in \mathbb{R}^{I \times T \times Q}$
$\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$	Nth order tensor representing usually input (observed) data
$\mathbf{Y}_{(n)}$	n-mode matricized (unfolded) version of $\underline{\mathbf{Y}}$
$\underline{\mathbf{Y}}_{i_n = p}$	subtensor of the tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ obtained by fixing the n-th index to some value p
$\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \cdots \times J_N}$	N-th order core tensor
$\mathbf{A}^{(n)}$	the n-th factor (loading matrix) of N-th order tensor
$\boldsymbol{a}_j^{(n)}$	j-th column vector of factor $\mathbf{A}^{(n)}$
$\{\mathbf{A}\}$	set of factors (loading matrices) $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
$\{\boldsymbol{a}_j\}$	set of vectors $\boldsymbol{a}_j^{(1)}, \boldsymbol{a}_j^{(2)}, \ldots, \boldsymbol{a}_j^{(N)}$
$D(\boldsymbol{p} \Vert \boldsymbol{q})$	divergence between two nonnegative 1D sequences: $\boldsymbol{p} = \{p_t\}$ and $\boldsymbol{q} = \{q_t\}$
$D(\mathbf{P} \Vert \mathbf{Q})$	divergence between two nonnegative 2D sequences: $\mathbf{P} = \{p_{it}\}$ and $\mathbf{Q} = \{q_{it}\}$
$D(\underline{\mathbf{P}} \Vert \underline{\mathbf{Q}})$	divergence between two nonnegative three-way arrays: $\underline{\mathbf{P}} = \{p_{itq}\}$ and $\underline{\mathbf{Q}} = \{q_{itq}\}$
$\mathbf{D}$	diagonal scaling matrix
$\det(\mathbf{A})$	determinant of matrix $\mathbf{A}$
$\mathbf{1}$	vector or matrix of suitable dimension with all ones
$\mathbf{1}_{I \times J} \in \mathbb{R}^{I \times J}$	I by J matrix with all ones

$\exp\{\cdot\}$	exponential function
$E\{\cdot\}$	expectation operator
$\mathbf{I}$	identity matrix
$\mathbf{I}_n$	identity matrix of dimension $n \times n$
$J(\mathbf{X})$	penalty or regularization term of a cost function
ln	natural logarithm (equivalent to log)
k	k-th alternating step
K	total number of alternating steps
Q	number of frontal slices, trials, experiments or subjects
I, T, Q	dimensions of different modes in three-way array $\underline{\mathbf{Y}}$
$p(\boldsymbol{x})$ or $p_x(\boldsymbol{x})$	probability density function (p.d.f.) of $\boldsymbol{x}$
$p_y(\boldsymbol{y})$	probability density function (p.d.f.) of $\boldsymbol{y}(t)$
$\mathbf{P}$	permutation matrix
$\text{sign}(x)$	sign function (= 1 for $x > 0$ and = -1 for $x < 0$)
t	continuous or discrete time
$\text{tr}(\mathbf{A})$	trace of matrix $\mathbf{A}$
$\lvert x \rvert$	absolute value (magnitude) of x
$\lVert \boldsymbol{x} \rVert_p$	p-norm (length) of the vector $\boldsymbol{x}$, where $p = 0, 1, 2, \ldots, \infty$
δ_{ij}	Kronecker delta
η	learning rate for discrete-time algorithms
$\lambda_{max}(\mathbf{A})$	maximal eigenvalue of matrix $\mathbf{A}$
$\lambda_{min}(\mathbf{A})$	minimal eigenvalue of matrix $\mathbf{A}$
σ^2	variance
ω	over-relaxation parameter $0 < \omega \leq 2$
∇	gradient operator
$\nabla_{x_i} D$	gradient of D with respect to variable x_i
$\nabla_{\mathbf{X}} D$	gradient of cost function D with respect to matrix $\mathbf{X}$
$[\boldsymbol{x}]_+ = \max\{0, \boldsymbol{x}\}$	nonlinear "half-wave rectifying" projection replacing negative values of $\boldsymbol{x}$ (element-wise) by zero or by a small positive value ε
$[\cdot]^\dagger$	superscript symbol for Moore-Penrose pseudo-inversion of a matrix
$[\cdot]^T$	transpose of matrix or vector
$\langle \cdot \rangle$	inner product or average operator

$:=$ or $\triangleq$	left hand variable is defined by the right hand variable
$\circledast$ or $.*$	Hadamard product
$\oslash$ or $./$	element-wise division
$\mathbf{X}^{[.\alpha]}$	rise to the power α each element of matrix $\mathbf{X}$
$\circ$	outer product
$\odot$	Khatri-Rao product
$\otimes$	Kronecker product ($\mathbf{A} \otimes \mathbf{B} := [a_{ij}\mathbf{B}]$)
$\times_n$	$n-mode$ product of a tensor by matrix
$\bar{\times}_n$	$n-mode$ product of a tensor by vector
$\mathbf{A} \sqcup_n \mathbf{B}$	concatenation of two tensors along the n-th mode
$\mathbf{A}^{\odot}$	$\mathbf{A}^{(N)} \odot \mathbf{A}^{(N-1)} \odot \cdots \odot \mathbf{A}^{(1)}$
$\mathbf{A}^{\odot -n}$	$\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}$
$\mathbf{A}^{\otimes}$	$\mathbf{A}^{(N)} \otimes \mathbf{A}^{(N-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}$
$\mathbf{A}^{\otimes -n}$	$\mathbf{A}^{(N)} \otimes \cdots \times \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}$
$\underline{\mathbf{G}} \times \{\mathbf{A}\}$	$\underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}$
$\underline{\mathbf{G}} \times_{-n} \{\mathbf{A}\}$	$\underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \cdots \times_{n-1} \mathbf{A}^{(n-1)} \times_{n+1} \mathbf{A}^{(n+1)} \cdots \times_N \mathbf{A}^{(N)}$

Abbreviations

ALS	Alternating Least Squares
HALS	Hierarchical Alternating Least Squares
RALS	Regularized Alternating Least Squares
i.i.d.	independent identically distributed
BOD	Block Oriented Decmposition
BSE	Blind Signal Extraction
BSS	Blind Signal Separation
BSD	Blind Signal Deconvolution
EVD	Eigenvalue Decomposition
ICA	Independent Component Analysis
IIR	Infinite Impulse Response
SIR	Signal-to-Interference-Ratio

PSNR	Peak Signal-to-Noise-Ratio
JAD	Joint approximative diagonalization
LMS	Least Mean Squares
KL	Kullback Leibler divergence
MCA	Minor Component Analysis
MBD	Multichannel Blind Deconvolution
MED	Maximum Entropy Distribution
MoCA	Morphological Component Analysis
MIMO	Multiple-Input, Multiple-Output
NMF	Nonnegative Matrix Factorization
aNMF	affine NMF
CNMF	Convolutive NMF
nsNMF	non-smooth NMF
NTD	Nonnegative Tucker Decomposition
NTF	Nonnegative Tensor Factorization
CP	CANDECOMP/PARAFAC
PARAFAC	Parallel Factor Analysis (equivalent to CP)
CANNDECOPM	Canonical Decomposition (equivalent to PARAFAC or CP)
PCA	Principal Component Analysis
SCA	Sparse Component Analysis
SOD	Slice Oriented Decomposition
SVD	Singular Value Decomposition

1

Introduction – Problem Statements and Models

Matrix factorization is an important and unifying topic in signal processing and linear algebra, which has found numerous applications in many other areas. This chapter introduces basic linear and multi-linear[1] models for matrix and tensor factorizations and decompositions, and formulates the analysis framework for the solution of problems posed in this book. The workhorse in this book is Nonnegative Matrix Factorization (NMF) for sparse representation of data and its extensions including the multi-layer NMF, semi-NMF, sparse NMF, tri-NMF, symmetric NMF, orthogonal NMF, non-smooth NMF (nsNMF), overlapping NMF, convolutive NMF (CNMF), and large-scale NMF. Our particular emphasis is on NMF and semi-NMF models and their extensions to multi-way models (i.e., multi-linear models which perform multi-way array (tensor) decompositions) with nonnegativity and sparsity constraints, including, Nonnegative Tucker Decompositions (NTD), Constrained Tucker Decompositions, Nonnegative and semi-nonnegative Tensor Factorizations (NTF) that are mostly based on a family of the TUCKER, PARAFAC and PARATUCK models.

As the theory and applications of NMF, NTF and NTD are still being developed, our aim is to produce a unified, state-of-the-art framework for the analysis and development of efficient and robust algorithms. In doing so, our main goals are to:

1. Develop various working tools and algorithms for data decomposition and feature extraction based on nonnegative matrix factorization (NMF) and sparse component analysis (SCA) approaches. We thus integrate several emerging techniques in order to estimate physically, physiologically, and neuroanatomically meaningful sources or latent

[1] A function in two or more variables is said to be multi-linear if it is linear in each variable separately.

(hidden) components with morphological constraints. These constraints include nonnegativity, sparsity, orthogonality, smoothness, and semi-orthogonality.

2. Extend NMF models to multi-way array (tensor) decompositions, factorizations, and filtering, and to derive efficient learning algorithms for these models.

3. Develop a class of advanced blind source separation (BSS), unsupervised feature extraction and clustering algorithms, and to evaluate their performance using *a priori* knowledge and morphological constraints.

4. Develop computational methods to efficiently solve the bi-linear system $\mathbf{Y} = \mathbf{AX} + \mathbf{E}$ for noisy data, where $\mathbf{Y}$ is an input data matrix, $\mathbf{A}$ and $\mathbf{X}$ represent unknown matrix factors to be estimated, and the matrix $\mathbf{E}$ represents error or noise (which should be minimized using suitably designed cost function).

5. Describe and analyze various cost functions (also referred to as (dis)similarity measures or divergences) and apply optimization criteria to ensure robustness with respect to uncertainty, ill-conditioning, interference and noise distribution.

6. Present various optimization techniques and statistical methods to derive efficient and robust learning (update) rules.

7. Study what kind of prior information and constraints can be used to render the problem solvable, and illustrate how to use this information in practice.

8. Combine information from different imaging modalities (e.g., electroencephalography (EEG), magnetoencephalography (MEG), electromyography (EMG), electrooculography (EOG), functional magnetic resonance imaging (fMRI), positron emission tomography (PET)), in order to provide data integration and assimilation.

9. Implement and optimize algorithms for NMF, NTF and NTD together with providing pseudo-source codes and/or efficient source codes in MATLAB, suitable for parallel computing and large-scale-problems.

10. Develop user-friendly toolboxes which supplement this book: NMFLAB and MULTIWAY-LAB for potential applications to data analysis, data mining, and blind source separation.

Probably the most useful and best understood matrix factorizations are the Singular Value Decomposition (SVD), Principal Component Analysis (PCA), and LU, QR, and Cholesky decompositions (see Appendix). In this book we mainly focus on nonnegativity and sparsity constraints for factor matrices. We shall therefore attempt to illustrate why nonnegativity and sparsity constraints play a key role in our investigations.

1.1 BLIND SOURCE SEPARATION AND LINEAR GENERALIZED COMPONENT ANALYSIS

Blind source separation (BSS) and related methods, e.g., independent component analysis (ICA), employ a wide class of unsupervised learning algorithms and have found important applications across several areas from engineering to neuroscience [26]. The recent trends in blind source

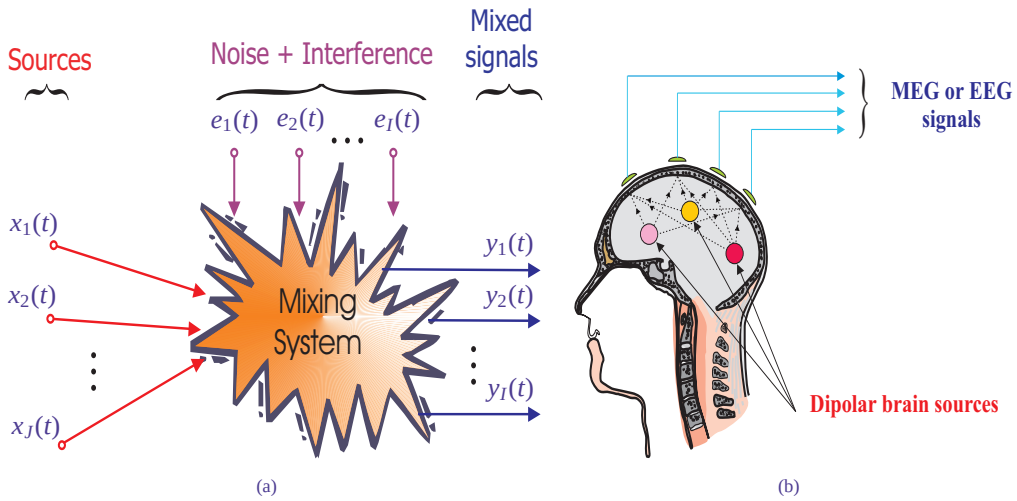


Fig. 1.1 (a) General model illustrating blind source separation (BSS), (b) Such models are exploited, for example, in noninvasive multi-sensor recording of brain activity using EEG (electroencephalography) or MEG (magnetoencephalography). It is assumed that the scalp sensors (e.g., electrodes, magnetic or optical sensors) pick up a superposition of neuronal brain sources and non-neuronal sources (noise or physiological artifacts) related, for example, to movements of eyes and muscles. Our objective is to identify the individual signals coming from different areas of the brain.

separation and generalized (flexible) component analysis (GCA) are to consider problems in the framework of matrix factorization or more general multi-dimensional data or signal decomposition with probabilistic generative models and exploit *a priori* knowledge about true nature, morphology or structure of latent (hidden) variables or sources such as nonnegativity, sparseness, spatio-temporal decorrelation, statistical independence, smoothness or lowest possible complexity. The goal of BSS can be considered as estimation of true physical sources and parameters of a mixing system, while the objective of GCA is to find a new reduced or hierarchical and structured component representation for the observed (sensor) data that can be interpreted as physically or physiologically meaningful coding or blind signal decomposition. The key issue is to find such a transformation or coding which has true physical meaning and interpretation.

Throughout this book we discuss some promising applications of BSS/GCA in analyzing multi-modal, multi-sensory data, especially brain data. Furthermore, we derive some efficient unsupervised learning algorithms for linear blind source separation, and generalized component analysis using various criteria, constraints and assumptions.

Figure 1.1 illustrates a fairly general BSS problem also referred to as blind signal decomposition or blind source extraction (BSE). We observe records of I sensor signals $\boldsymbol{y}(t) = [y_1(t), y_2(t), \ldots, y_I(t)]^T$ coming from a MIMO (multiple-input/multiple-output) mixing and filtering system, where t is usually a discrete time sample,[2] and $(\cdot)^T$ denotes transpose of a vector. These signals are usually a superposition (mixture) of J unknown source signals $\boldsymbol{x}(t) = [x_1(t), x_2(t), \ldots, x_J(t)]^T$ and noises $\boldsymbol{e}(t) = [e_1(t), e_2(t), \ldots, e_I(t)]^T$. The primary objective is to estimate all the primary

[2]Data are often represented not in the time domain but in the complex frequency or time-frequency domain, so, the index t may have a different meaning and can be multi-dimensional.

source signals $x_j(t) = x_{jt}$ or only some of them with specific properties. This estimation is usually performed based only on the output (sensor, observed) signals $y_{it} = y_i(t)$.

In order to estimate sources, sometimes we try first to identify the mixing system or its inverse (unmixing) system and then estimate the sources. Usually, the inverse (unmixing) system should be adaptive in such a way that it has some tracking capability in a nonstationary environment. Instead of estimating the source signals directly by projecting observed signals using the unmixing system, it is often more convenient to identify an unknown mixing and filtering system (e.g., when the unmixing system does not exist, especially when the system is underdetermined, i.e., the number of observations is lower than the number of source signals with $I < J$) and simultaneously estimate the source signals by exploiting some *a priori* information about the source signals and applying a suitable optimization procedure.

There appears to be something magical about blind source separation since we are estimating the original source signals without knowing the parameters of the mixing and/or filtering processes. It is difficult to imagine that one can estimate this at all. In fact, without some *a priori* knowledge, it is not possible to *uniquely* estimate the original source signals. However, one can usually estimate them up to certain indeterminacies. In mathematical terms these indeterminacies and ambiguities can be expressed as arbitrary scaling and permutation of the estimated source signals. These indeterminacies preserve, however, the waveforms of original sources. Although these indeterminacies seem to be rather severe limitations, in a great number of applications these limitations are not crucial, since the most relevant information about the source signals is contained in the temporal waveforms or time-frequency patterns of the source signals and usually not in their amplitudes or the order in which they are arranged in the system output.[3]

The problem of separating or extracting source signals from a sensor array, without knowing the transmission channel characteristics and the sources, can be expressed briefly as a number of related BSS or GCA methods such as ICA and its extensions: Topographic ICA, Multi-way ICA, Kernel ICA, Tree-dependent Component Analysis, Multi-resolution Subband Decomposition -ICA [77], [41], [28], [29], Non-negative Matrix Factorization (NMF) [93], [120], [35], Sparse Component Analysis (SCA) [96], [95], [141], [70], [72], and Multi-channel Morphological Component Analysis (MCA) [13] (see Figure 1.2).

The mixing and filtering processes of the unknown input sources $x_j(t)$, $(j = 1, 2, \ldots, J)$ may have different mathematical or physical models, depending on the specific applications [77], [4]. Most linear BSS models in their simplest forms can be expressed algebraically as some specific forms of matrix factorization: Given observation (often called sensor or input data matrix) $\mathbf{Y} = [y_{it}] = [\boldsymbol{y}(1), \ldots, \boldsymbol{y}(T)] \in \mathbb{R}^{I \times T}$ perform the matrix factorization (see Figure 1.3(a)):

$$\boxed{\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{E},} \tag{1.1}$$

where $\mathbf{A} \in \mathbb{R}^{I \times J}$ represents the unknown basis matrix or mixing matrix (depending on the application), $\mathbf{E} \in \mathbb{R}^{I \times T}$ is an unknown matrix representing errors or noises, $\mathbf{X} = [x_{jt}] = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)] \in \mathbb{R}^{J \times T}$ contains the corresponding latent (hidden) components that give the contribution of each basis vector, T is the number of available samples, I is the number of observations and J is the number of sources or components. In general, the number of source signals J is unknown and can be larger, equal or smaller than the number of observations. The above model

[3] For some models, however, there is no guarantee that the estimated or extracted signals have exactly the same waveforms as the source signals, and then the requirements must be sometimes further relaxed to the extent that the extracted waveforms are distorted (i.e., time delayed, filtered or convolved) versions of the primary source signals.

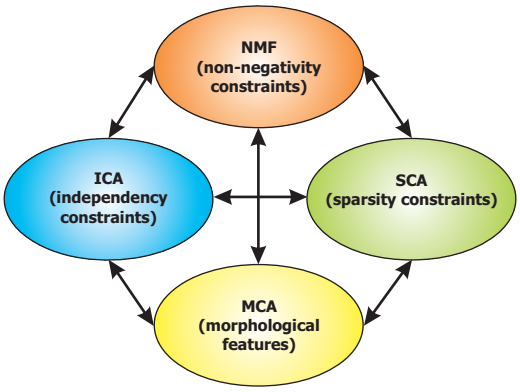


Fig. 1.2 Four basic component analysis methods: Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), Sparse Component Analysis (SCA) and Morphological Component Analysis (MCA).

can be written in an equivalent scalar (element-wise) form (see Figure 1.3(b)):

$$y_{it} = \sum_{j=1}^{J} a_{ij}\, x_{jt} + e_{it} \qquad \text{or} \qquad y_i(t) = \sum_{j=1}^{J} a_{ij}\, x_j(t) + e_i(t). \tag{1.2}$$

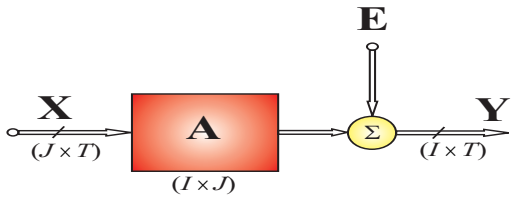
Usually, the latent components represent unknown source signals with specific statistical properties or temporal structures. The matrices usually have clear statistical properties and meanings. For example, the rows of the matrix $\mathbf{X}$ that represent sources or components should be statistically independent for ICA, sparse for SCA [96], [95], [70], [69], [72], nonnegative for NMF, or have other specific and additional morphological properties such as sparsity, smoothness, continuity, or orthogonality in GCA [29], [13], [26].

In some applications the mixing matrix $\mathbf{A}$ is ill-conditioned or even singular. In such cases, some special models and algorithms should be applied. Although some decompositions or matrix factorizations provide an exact reconstruction of the data (i.e., $\mathbf{Y} = \mathbf{AX}$), we shall consider here factorizations which are approximative in nature. In fact, many problems in signal and image processing can be solved in terms of matrix factorization. However, different cost functions and imposed constraints may lead to different types of matrix factorization. In many signal processing applications the data matrix $\mathbf{Y} = [\boldsymbol{y}(1), \boldsymbol{y}(2) \ldots, \boldsymbol{y}(T)] \in \mathbb{R}^{I \times T}$ is represented by vectors $\boldsymbol{y}(t) \in \mathbb{R}^I$ $(t = 1, 2, \ldots, T)$ for a set of discrete time instants t as multiple measurements or recordings. As mentioned above, the compact aggregated matrix equation (1.1) can be written in a vector form as a system of linear equations (see Figure 1.4(a)), that is,

$$\boldsymbol{y}(t) = \mathbf{A}\, \boldsymbol{x}(t) + \boldsymbol{e}(t), \qquad (t = 1, 2, \ldots, T), \tag{1.3}$$

where $\boldsymbol{y}(t) = [y_1(t), y_2(t), \ldots, y_I(t)]^T$ is a vector of the observed signals at the discrete time instant t whereas $\boldsymbol{x}(t) = [x_1(t), x_2(t), \ldots, x_J(t)]^T$ is a vector of unknown sources at the same time instant. The problems formulated above are closely related to the concept of linear inverse problems or more generally, to solving a large ill-conditioned system of linear equations (overdetermined or underdetermined), where it is required to estimate vectors $\boldsymbol{x}(t)$ (also in some cases to identify a matrix $\mathbf{A}$) from noisy data [87], [26], [32]. Physical systems are often contaminated by noise,

(a)



(b)

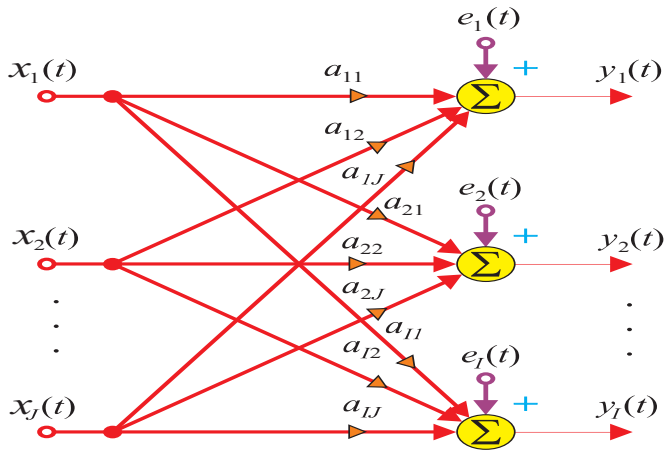


Fig. 1.3 Basic linear instantaneous BSS model: (a) Block diagram, (b) detailed model.

thus, our task is generally to find an optimal and robust solution in a noisy environment. Wide classes of extrapolation, reconstruction, estimation, approximation, interpolation and inverse problems can be converted into minimum norm problems of solving underdetermined systems of linear equations (1.3) for $J > I$ [87], [26].[4] It is often assumed that only the sensor vectors $\boldsymbol{y}(t)$ are available and we need to estimate parameters of the unmixing system online. This enables us to perform indirect identification of the mixing matrix $\mathbf{A}$ (for $I \geq J$) by estimating the separating matrix $\mathbf{W} = \hat{\mathbf{A}}^{\dagger}$, where the symbol $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudo-inverse and simultaneously estimate the sources. In other words, for $I \geq J$ the original sources can be estimated by the linear transformation

$$\hat{\boldsymbol{x}}(t) = \mathbf{W}\,\boldsymbol{y}(t), \qquad (t = 1, 2, \ldots, T). \tag{1.4}$$

Although many different BSS criteria and algorithms are available, most of them exploit various diversities[5] or constraints imposed for estimated components and/or mixing matrices such as mutual independence, nonnegativity, sparsity, smoothness, predictability or lowest complexity. More sophisticated or advanced approaches use combinations or integration of various diversi-

[4] Generally speaking, in signal processing applications, an overdetermined ($I > J$) system of linear equations (1.3) describes filtering, enhancement, deconvolution and identification problems, while the underdetermined case describes inverse and extrapolation problems [32], [26].

[5] By diversities we mean usually different morphological characteristics or features of the signals.

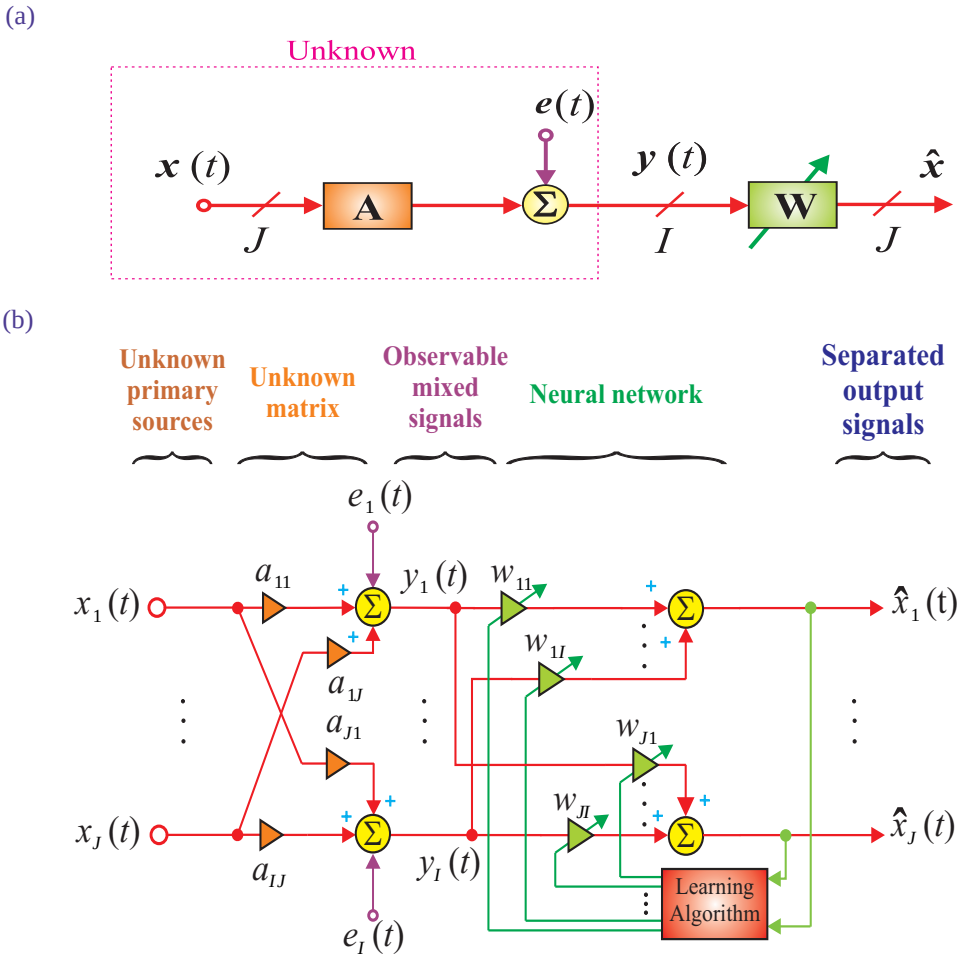


Fig. 1.4 Blind source separation using unmixing (inverse) model: (a) block diagram, and(b) detailed model.

ties, in order to separate or extract sources with various constraints, morphology, structures or statistical properties and to reduce the influence of noise and undesirable interferences [26].

All the above-mentioned BSS methods belong to a wide class of unsupervised learning algorithms. Unsupervised learning algorithms try to discover a structure underlying a data set, extract of meaningful features, and find useful representations of the given data. Since data can always be interpreted in many different ways, some knowledge is needed to determine which features or properties best represent our true latent (hidden) components. For example, PCA finds a low-dimensional representation of the data that captures most of its variance. On the other hand, SCA tries to explain data as a mixture of sparse components (usually, in the time-frequency domain), and NMF seeks to explain data by parts-based localized additive representations (with nonnegativity constraints).

Generalized component analysis algorithms, i.e., a combination of ICA, SCA, NMF, and MCA, are often considered as pure mathematical formulas, powerful, but rather mechanical

procedures. There is an illusion that there is not very much left for the user to do after the machinery has been optimally implemented. However, the successful and efficient use of such tools strongly depends on *a priori* knowledge, common sense, and appropriate use of the preprocessing and postprocessing tools. In other words, it is the preprocessing of data and postprocessing of models where expertise is truly needed in order to extract and identify physically significant and meaningful hidden components.

1.2 MATRIX FACTORIZATION MODELS WITH NONNEGATIVITY AND SPARSITY CONSTRAINTS

1.2.1 Why Nonnegativity and Sparsity Constraints?

Many real-world data are nonnegative and the corresponding hidden components have a physical meaning only when nonnegative. In practice, both nonnegative and sparse decompositions of data are often either desirable or necessary when the underlying components have a physical interpretation. For example, in image processing and computer vision, involved variables and parameters may correspond to pixels, and nonnegative sparse decomposition is related to the extraction of relevant parts from the images [93], [94]. In computer vision and graphics, we often encounter multi-dimensional data, such as images, video, and medical data, one type of which is MRI (magnetic resonance imaging). A color image can be considered as 3D nonnegative data, two of the dimensions (rows and columns) being spatial, and the third one being a color plane (channel) depending on its color space, while a color video sequence can be considered as 4D nonnegative data, time being the fourth dimension. A sparse representation of the data by a limited number of components is an important research problem. In machine learning, sparseness is closely related to feature selection and certain generalizations in learning algorithms, while nonnegativity relates to probability distributions. In economics, variables and data such as volume, price and many other factors are nonnegative and sparse. Sparseness constraints may increase the efficiency of a portfolio, while nonnegativity both increases efficiency and reduces risk [143], [122]. In microeconomics, household expenditures in different commodity/service groups are recorded as a relative proportion. In information retrieval, documents are usually represented as relative frequencies of words in a prescribed vocabulary. In environmental science, scientists investigate a relative proportion of different pollutants in water or air [11]. In biology, each coordinate axis may correspond to a specific gene and the sparseness is necessary for finding local patterns hidden in data, whereas the nonnegativity is required to give physical or physiological meaning. This is also important for the robustness of biological systems, where any observed change in the expression level of a specific gene emerges from either positive or negative influence, rather than a combination of both, which partly cancel each other [94], [143].

It is clear, however, that with constraints such as sparsity and nonnegativity some of the explained variance (FIT) may decrease. In other words, it is natural to seek a trade-off between the two goals of interpretability (making sure that the estimated components have physical or physiological sense and meaning) and statistical fidelity (explaining most of the variance of the data, if the data are consistent and do not contain too much noise). Generally, compositional data (i.e., positive sum of components or real vectors) are natural representations when the variables (features) are essentially the probabilities of complementary and mutually exclusive events. Furthermore, note that NMF is an additive model which does not allow subtraction; therefore it

often quantitatively describes the parts that comprise the entire entity. In other words, NMF can be considered as a part-based representation in which a zero-value represents the absence and a positive number represents the presence of some event or component. Specifically, in the case of facial image data, the additive or part-based nature of NMF has been shown to result in a basis of facial features, such as eyes, nose, and lips [93]. Furthermore, matrix factorization methods that exploit nonnegativity and sparsity constraints usually lead to estimation of the hidden components with specific structures and physical interpretations, in contrast to other blind source separation methods.

1.2.2 Basic NMF Model

NMF has been investigated by many researchers, e.g. Paatero and Tapper [113], but it has gained popularity through the works of Lee and Seung published in Nature and NIPS [93], [94]. Based on the argument that the nonnegativity is important in human perception they proposed simple algorithms (often called the Lee-Seung algorithms) for finding nonnegative representations of nonnegative data and images.

The basic NMF problem can be stated as follows: Given a nonnegative data matrix $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$ (with $y_{it} \geq 0$ or equivalently $\mathbf{Y} \geq \mathbf{0}$) and a reduced rank J ($J \leq \min(I, T)$), find two nonnegative matrices $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} = \mathbf{B}^T = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J]^T \in \mathbb{R}_+^{J \times T}$ which factorize $\mathbf{Y}$ as well as possible, that is (see Figure 1.3):

$$\boxed{\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{E} = \mathbf{A}\mathbf{B}^T + \mathbf{E},} \tag{1.5}$$

where the matrix $\mathbf{E} \in \mathbb{R}^{I \times T}$ represents approximation error.[6] The factors $\mathbf{A}$ and $\mathbf{X}$ may have different physical meanings in different applications. In a BSS problem, $\mathbf{A}$ plays the role of mixing matrix, while $\mathbf{X}$ expresses source signals. In clustering problems, $\mathbf{A}$ is the basis matrix, while $\mathbf{X}$ denotes the weight matrix. In acoustic analysis, $\mathbf{A}$ represents the basis patterns, while each row of $\mathbf{X}$ expresses time points (positions) when sound patterns are activated.

In standard NMF we only assume nonnegativity of factor matrices $\mathbf{A}$ and $\mathbf{X}$. Unlike blind source separation methods based on independent component analysis (ICA), here we do not assume that the sources are independent, although we will introduce other assumptions or constraints on $\mathbf{A}$ and/or $\mathbf{X}$ later. Notice that this symmetry of assumptions leads to a symmetry in the factorization: we could just as easily write $\mathbf{Y}^T \approx \mathbf{X}^T\mathbf{A}^T$, so the meaning of "source" and "mixture" in NMF are often somewhat arbitrary.

The NMF model can also be represented as a special form of the bilinear model (see Figure 1.5):

$$\boxed{\mathbf{Y} = \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{b}_j + \mathbf{E} = \sum_{j=1}^{J} \boldsymbol{a}_j \boldsymbol{b}_j^T + \mathbf{E},} \tag{1.6}$$

where the symbol $\circ$ denotes the outer product of two vectors. Thus, we can build an approximate representation of the nonnegative data matrix $\mathbf{Y}$ as a sum of rank-one nonnegative matrices $\boldsymbol{a}_j \boldsymbol{b}_j^T$. If such decomposition is exact (i.e., $\mathbf{E} = \mathbf{0}$) then it is called the Nonnegative Rank

[6] Since we usually operate on column vectors of matrices (in order to avoid a complex or confused notation) it is often convenient to use the matrix $\mathbf{B} = \mathbf{X}^T$ instead of the matrix $\mathbf{X}$.

Factorization (NRF) [53]. Among the many possible series representations of data matrix $\mathbf{Y}$ by nonnegative rank-one matrices, the smallest integer J for which such a nonnegative rank-one series representation is attained is called the nonnegative rank of the nonnegative matrix $\mathbf{Y}$ and it is denoted by $\text{rank}_+(\mathbf{Y})$. The nonnegative rank satisfies the following bounds [53]:

$$\text{rank}(\mathbf{Y}) \leq \text{rank}_+(\mathbf{Y}) \leq \min\{I, T\}. \tag{1.7}$$

It should be noted that an NMF is not necessarily an NRF in the sense that the latter demands the exact factorization whereas the former is usually only an approximation.

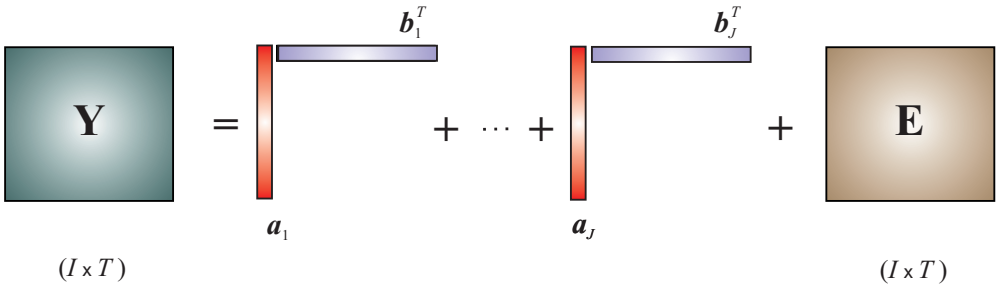


Fig. 1.5 Bilinear NMF model. The nonnegative data matrix $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$ is approximately represented by a sum or linear combination of rank-one nonnegative matrices $\mathbf{Y}^{(j)} = \boldsymbol{a}_j \circ \boldsymbol{b}_j = \boldsymbol{a}_j \boldsymbol{b}_j^T \in \mathbb{R}_+^{I \times T}$.

Although the NMF can be applied to BSS problems for nonnegative sources and nonnegative mixing matrices, its application is not limited to BSS and it can be used in various and diverse applications far beyond BSS (see Chapter 8). In many applications we require additional constraints on the elements of matrices $\mathbf{A}$ and/or $\mathbf{X}$, such as smoothness, sparsity, symmetry, and orthogonality.

1.2.3 Symmetric NMF

In the special case when $\mathbf{A} = \mathbf{B} \in \mathbb{R}_+^{I \times J}$ the NMF is called a symmetric NMF, given by

$$\mathbf{Y} = \mathbf{A}\mathbf{A}^T + \mathbf{E}. \tag{1.8}$$

This model is also considered equivalent to Kernel K-means clustering and Laplace spectral clustering [50].

If the exact symmetric NMF ($\mathbf{E} = \mathbf{0}$) exists then a nonnegative matrix $\mathbf{Y} \in \mathbb{R}_+^{I \times I}$ is said to be completely positive (CP) and the smallest number of columns of $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ satisfying the exact factorization $\mathbf{Y} = \mathbf{A}\mathbf{A}^T$ is called the cp-rank of the matrix $\mathbf{Y}$, denoted by $\text{rank}_{cp}(\mathbf{Y})$. If $\mathbf{Y}$ is CP, then the upper bound estimate of the cp-rank is given by [53]:

$$\text{rank}_{cp}(\mathbf{Y}) \leq \frac{\text{rank}(\mathbf{Y})(\text{rank}(\mathbf{Y}) + 1)}{2} - 1, \tag{1.9}$$

provided $\text{rank}(\mathbf{Y}) > 1$.

1.2.4 Semi-Orthogonal NMF

The semi-orthogonal NMF can be defined as

$$\mathbf{Y} = \mathbf{AX} + \mathbf{E} = \mathbf{AB}^T + \mathbf{E}, \tag{1.10}$$

subject to nonnegativity constraints $\mathbf{A} \geq \mathbf{0}$ and $\mathbf{X} \geq \mathbf{0}$ (component-wise) and an additional orthogonality constraint: $\mathbf{A}^T\mathbf{A} = \mathbf{I}_J$ or $\mathbf{X}\mathbf{X}^T = \mathbf{I}_J$.

Probably the simplest and most efficient way to impose orthogonality onto the matrix $\mathbf{A}$ or $\mathbf{X}$ is to perform the following transformation after each iteration

$$\boxed{\mathbf{A} \leftarrow \mathbf{A}\left[\mathbf{A}^T\mathbf{A}\right]^{-1/2}, \qquad \text{or} \qquad \mathbf{X} \leftarrow \left[\mathbf{X}\mathbf{X}^T\right]^{-1/2}\mathbf{X}.} \tag{1.11}$$

1.2.5 Semi-NMF and Nonnegative Factorization of Arbitrary Matrix

In some applications the observed input data are unsigned (unconstrained or bipolar) as indicated by $\mathbf{Y} = \mathbf{Y}_{\pm} \in \mathbb{R}^{I \times T}$ which allows us to relax the constraints regarding nonnegativity of one factor (or only specific vectors of a matrix). This leads to approximative semi-NMF which can take the following form

$$\mathbf{Y}_{\pm} = \mathbf{A}_{\pm}\mathbf{X}_{+} + \mathbf{E}, \qquad \text{or} \qquad \mathbf{Y}_{\pm} = \mathbf{A}_{+}\mathbf{X}_{\pm} + \mathbf{E}, \tag{1.12}$$

where the subscript in $\mathbf{A}_{+}$ indicates that a matrix is forced to be nonnegative.

In Chapter 4 we discuss models and algorithms for approximative factorizations in which the matrices $\mathbf{A}$ and/or $\mathbf{X}$ are restricted to contain nonnegative entries, but the data matrix $\mathbf{Y}$ may have entries with mixed signs, thus extending the range of applications of NMF. Such a model is often referred to as Nonnegative Factorization (NF) [58], [59].

1.2.6 Three-factor NMF

Three-factor NMF (also called the tri-NMF) can be considered as a special case of the multi-layer NMF and can take the following general form [52], [51]

$$\boxed{\mathbf{Y} = \mathbf{ASX} + \mathbf{E},} \tag{1.13}$$

where nonnegativity constraints are imposed to all or only to the selected factor matrices: $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{S} \in \mathbb{R}^{J \times R}$, and/or $\mathbf{X} \in \mathbb{R}^{R \times T}$.

It should be noted that if we do not impose any additional constraints to the factors (besides nonnegativity), the three-factor NMF can be reduced to the standard (two-factor) NMF by the transformation $\mathbf{A} \leftarrow \mathbf{AS}$ or $\mathbf{X} \leftarrow \mathbf{SX}$. However, the three-factor NMF is not equivalent to the standard NMF if we apply special constraints or conditions as illustrated by the following special cases.

1.2.6.1 **Orthogonal Three-Factor NMF** Orthogonal three-factor NMF imposes additional constraints upon the two matrices $\mathbf{A}^T\mathbf{A} = \mathbf{I}_J$ and $\mathbf{X}\mathbf{X}^T = \mathbf{I}_R$ while the matrix $\mathbf{S}$ can be an arbitrary unconstrained matrix (i.e., it has both positive and negative entries) [52], [51].

For uni-orthogonal three-factor NMF only one matrix $\mathbf{A}$ or $\mathbf{X}$ is orthogonal and all three matrices are usually nonnegative.

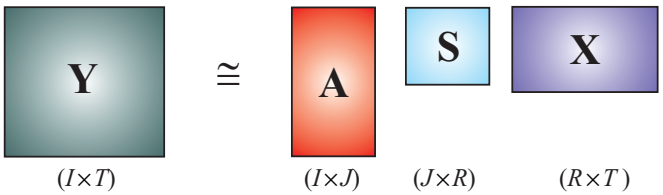


Fig. 1.6 Illustration of three factor NMF (tri-NMF). The goal is to estimate two matrices $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{R \times T}$, assuming that the matrix $\mathbf{S} \in \mathbb{R}^{J \times R}$ is given, or to estimate all three factor matrices $\mathbf{A}, \mathbf{S}, \mathbf{X}$ subject to additional constraints such as orthogonality or sparsity.

1.2.6.2 **Non-Smooth NMF** Non-smooth NMF (nsNMF) was proposed by Pascual-Montano *et al.* [114] and is a special case of the three-factor NMF model in which the matrix $\mathbf{S}$ is fixed and known, and is used for controlling the sparsity or smoothness of the matrix $\mathbf{X}$ and/or $\mathbf{A}$. Typically, the smoothing matrix $\mathbf{S} \in \mathbb{R}^{J \times J}$ takes the form:

$$\mathbf{S} = (1 - \Theta)\, \mathbf{I}_J + \frac{\Theta}{J}\, \mathbf{1}_{J \times J}, \tag{1.14}$$

where $\mathbf{I}_J$ is $J \times J$ identity matrix and $\mathbf{1}_{J \times J}$ is the matrix of all ones. The scalar parameter $0 \leq \Theta \leq 1$ controls the smoothness of the matrix operator $\mathbf{S}$. For $\Theta = 0$, $\mathbf{S} = \mathbf{I}_J$, the model reduces to the standard NMF and for $\Theta \to 1$ strong smoothing is imposed on $\mathbf{S}$, causing increased sparseness of both $\mathbf{A}$ and $\mathbf{X}$ in order to maintain the faithfulness of the model.

1.2.6.3 **Filtering NMF** In many applications it is necessary to impose some kind of filtering upon the rows of the matrix $\mathbf{X}$ (representing source signals), e.g., lowpass filtering to perform smoothing or highpass filtering in order to remove slowly changing trends from the estimated components (source signals). In such cases we can define the filtering NMF as

$$\boxed{\mathbf{Y} = \mathbf{AXF} + \mathbf{E},} \tag{1.15}$$

where $\mathbf{F}$ is a suitably designed (prescribed) filtering matrix. In the case of lowpass filtering, we usually perform some kind of averaging in the sense that every sample value x_{jt} is replaced by a weighted average of that value and the neighboring value, so that in the simplest scenario the smoothing lowpass filtering matrix $\mathbf{F}$ can take the following form:

$$\mathbf{F} = \begin{bmatrix} 1/2 & 1/3 & 0 & & & 0 \\ 1/2 & 1/3 & 1/3 & & & 0 \\ & 1/3 & 1/3 & 1/3 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & 1/3 & 1/3 & 1/2 \\ 0 & & & 0 & 1/3 & 1/2 \end{bmatrix} \in \mathbb{R}^{T \times T}. \tag{1.16}$$

A standard way of performing highpass filtering is equivalent to an application of a first-order differential operator, which means (in the simplest scenario) just replacing each sample value by the difference between the value at that point and the value at the preceding point. For example, a highpass filtering matrix can take following form (using the first order or second order discrete

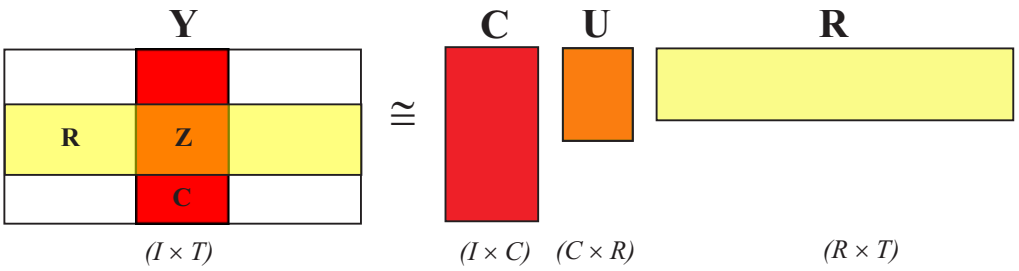


Fig. 1.7 Illustration of CUR decomposition. The objective is to select such rows and columns of data matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$ which provide the best approximation. The matrix $\mathbf{U}$ is usually the pseudo-inverse of the matrix $\mathbf{Z} \in \mathbb{R}^{R \times C}$, i.e., $\mathbf{U} = \mathbf{Z}^{\dagger}$. For simplicity of graphical illustration, we have assumed that the joint R rows and the joint C columns of the matrix $\mathbf{Y}$ are selected.

difference forms):

$$\mathbf{F} = \begin{bmatrix} 1 & -1 & 0 & & & 0 \\ -1 & 2 & -1 & & & 0 \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & -1 & 2 & -1 \\ 0 & & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{T \times T}. \tag{1.17}$$

Note that since the matrix $\mathbf{S}$ in the nsNMF and the matrix $\mathbf{F}$ in filtering NMF are known or designed in advance, almost all the algorithms known for the standard NMF can be straightforwardly extended to the nsNMF and Filtering NMF, for example, by defining new matrices $\mathbf{A} \triangleq \mathbf{AS}$, $\mathbf{X} \triangleq \mathbf{SX}$, or $\mathbf{X} \triangleq \mathbf{XF}$, respectively.

1.2.6.4 **CGR/CUR Decomposition** In the CGR, also recently called CUR decomposition, a given data matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$ is decomposed as follows [61], [60], [55], [101], [100]:

$$\boxed{\mathbf{Y} = \mathbf{CUR} + \mathbf{E},} \tag{1.18}$$

where $\mathbf{C} \in \mathbb{R}^{I \times C}$ is a matrix constructed from C selected columns of $\mathbf{Y}$, $\mathbf{R} \in \mathbb{R}^{R \times T}$ consists of R rows of $\mathbf{Y}$ and matrix $\mathbf{U} \in \mathbb{R}^{C \times R}$ is chosen to minimize the error $\mathbf{E} \in \mathbb{R}^{I \times T}$. The matrix $\mathbf{U}$ is often the pseudo-inverse of a matrix $\mathbf{Z} \in \mathbb{R}^{R \times C}$, i.e., $\mathbf{U} = \mathbf{Z}^{\dagger}$, which is defined by the intersections of the selected rows and columns (see Figure 1.7). Alternatively, we can compute a core matrix $\mathbf{U}$ as $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{Y}\mathbf{R}^{\dagger}$, but in this case knowledge of the whole data matrix $\mathbf{Y}$ is necessary.

Since typically, $C << T$ and $R << I$, our challenge is to find a matrix $\mathbf{U}$ and select rows and columns of $\mathbf{Y}$ so that for the fixed number of columns and rows the error cost function $\|\mathbf{E}\|_F^2$ is minimized. It was proved by Goreinov *et al.* [60] that for $R = C$ the following bounds can be theoretically achieved:

$$\|\mathbf{Y} - \mathbf{CUR}\|_{\max} \leq (R+1)\,\sigma_{R+1}, \tag{1.19}$$

$$\|\mathbf{Y} - \mathbf{CUR}\|_F \leq \sqrt{1 + R(T-R)}\,\sigma_{R+1}, \tag{1.20}$$

where $\|\mathbf{Y}\|_{\max} = \max_{it}\{|y_{it}|\}$ denotes max norm and σ_r is the r-th singular value of $\mathbf{Y}$.

Without loss of generality, let us assume that the first C columns and the first R rows of the matrix $\mathbf{Y}$ are selected so the matrix is partitioned as follows:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{bmatrix} \in \mathbb{R}^{I \times T}, \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{Y}_{11} \\ \mathbf{Y}_{21} \end{bmatrix} \in \mathbb{R}^{I \times C}, \quad \mathbf{R} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \end{bmatrix} \in \mathbb{R}^{R \times T}, \tag{1.21}$$

then the following bound is obtained [61]

$$\|\mathbf{Y} - \mathbf{C}\mathbf{Y}_{11}^{\dagger}\mathbf{R}\|_F \leq \gamma_R \, \sigma_{R+1}, \tag{1.22}$$

where $\gamma_R = \min\left\{ \sqrt{(1 + \|\mathbf{Y}_{21}\mathbf{Y}_{11}^{\dagger}\|_F^2)}, \sqrt{(1 + \|\mathbf{Y}_{11}^{\dagger}\mathbf{Y}_{12}\|_F^2)} \right\}$. This formula allows us to identify optimal columns and rows in sequential manner [22]. In fact, there are several strategies for the selection of suitable columns and rows. The main principle is to select columns and rows that exhibit high "statistical leverage" and provide the best low-rank fit of the data matrix [60], [55].

In the special case, assuming that $\mathbf{U}\mathbf{R} = \mathbf{X}$, we have CX decomposition:

$$\boxed{\mathbf{Y} = \mathbf{C}\mathbf{X} + \mathbf{E}.} \tag{1.23}$$

The CX and CUR (CGR) decompositions are low-rank matrix decompositions that are explicitly expressed in terms of a small number of actual columns and/or actual rows of the data matrix and they have recently received increasing attention in the data analysis community, especially for nonnegative data due to many potential applications [60, 55, 100, 101]. The CUR decomposition has an advantage that components (factor matrices $\mathbf{C}$ and $\mathbf{R}$) are directly obtained from rows and columns of data matrix $\mathbf{Y}$, preserving desired properties such as nonnegativity or sparsity. Because they are constructed from actual data elements, CUR decomposition is often more easily interpretable by practitioners of the field from which the data are drawn (to the extent that the original data points and/or features are interpretable) [100].

1.2.7 NMF with Offset (Affine NMF)

In NMF with offset (also called affine NMF, aNMF), our goal is to remove the base line or DC bias from the matrix $\mathbf{Y}$ by using a slightly modified NMF model:

$$\boxed{\mathbf{Y} = \mathbf{A}\mathbf{X} + \boldsymbol{a}_0\mathbf{1}^T + \mathbf{E},} \tag{1.24}$$

where $\mathbf{1} \in \mathbb{R}^T$ is a vector of all ones and $\boldsymbol{a}_0 \in \mathbb{R}_+^I$ is a vector which is selected in such a way that the matrix $\mathbf{X}$ is zero-grounded, that is, with a possibly large number of zero entries in each row (or for noisy data close to zero entries). The term $\mathbf{Y}_0 = \boldsymbol{a}_0\mathbf{1}^T$ denotes offset, which together with nonnegativity constraint often ensures the sparseness of factored matrices. The main role of the offset is to absorb the constant values of a data matrix, thereby making the factorization sparser and therefore improving (relaxing) conditions for the uniqueness of NMF (see next sections). Chapter 3 will demonstrates the affine NMF with multiplicative algorithms. However, in practice, the offsets are not the same and perfectly constant in all data sources. For image data, due to illumination flicker, the intensities of offset regions vary between images. Affine NMF with the model (1.24) fails to decompose such data. The Block-Oriented Decomposition (BOD1) model presented in section (1.5.9) will help us resolving this problem.

1.2.8 Multi-layer NMF

In multi-layer NMF the basic matrix $\mathbf{A}$ is replaced by a set of cascaded (factor) matrices. Thus, the model can be described as (see Figure 1.8)

$$\mathbf{Y} = \mathbf{A}^{(1)}\mathbf{A}^{(2)}\cdots\mathbf{A}^{(L)}\mathbf{X} + \mathbf{E}. \tag{1.25}$$

Since the model is linear, all the matrices can be merged into a single matrix $\mathbf{A}$ if no special constraints are imposed upon the individual matrices $\mathbf{A}^{(l)}, (l = 1, 2, \ldots, L)$. However, multi-layer NMF can be used to considerably improve the performance of standard NMF algorithms due to distributed structure and alleviating the problem of local minima.

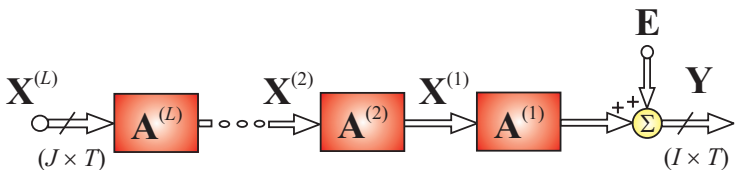


Fig. 1.8 Multilayer NMF model. In this model the global factor matrix $\mathbf{A} = \mathbf{A}^{(1)}\mathbf{A}^{(2)}\cdots\mathbf{A}^{(L)}$ has distributed representation in which each matrix $\mathbf{A}^{(l)}$ can be sparse.

To improve the performance of the NMF algorithms (especially for ill-conditioned and badly-scaled data) and to reduce the risk of converging to local minima of a cost function due to non-convex alternating minimization, we have developed a simple hierarchical multi-stage procedure [39], [38], [27], [37] combined with a multi-start initialization, in which we perform a sequential decomposition of nonnegative matrices as follows. In the first step, we perform the basic approximate decomposition $\mathbf{Y} \cong \mathbf{A}^{(1)}\mathbf{X}^{(1)} \in \mathbb{R}^{I\times T}$ using any available NMF algorithm. In the second stage, the results obtained from the first stage are used to build up a new input data matrix $\mathbf{Y} \leftarrow \mathbf{X}^{(1)}$, that is, in the next step, we perform a similar decomposition $\mathbf{X}^{(1)} \cong \mathbf{A}^{(2)}\mathbf{X}^{(2)} \in \mathbb{R}^{J\times T}$, using the same or different update rules. We continue our decomposition taking into account only the last obtained components. The process can be repeated for an arbitrary number of times until some stopping criteria are satisfied. Thus, our multi-layer NMF model has the form:

$$\mathbf{Y} \cong \mathbf{A}^{(1)}\mathbf{A}^{(2)}\cdots\mathbf{A}^{(L)}\mathbf{X}^{(L)}, \tag{1.26}$$

with the final results $\mathbf{A} = \mathbf{A}^{(1)}\mathbf{A}^{(2)}\cdots\mathbf{A}^{(L)}$ and $\mathbf{X} = \mathbf{X}^{(L)}$. Physically, this means that we build up a distributed system that has many layers or cascade connections of L mixing subsystems. The key point in this approach is that the learning (update) process to find parameters of matrices $\mathbf{X}^{(l)}$ and $\mathbf{A}^{(l)}, (l = 1, 2, \ldots, L)$ is performed sequentially, layer-by-layer, where each layer is randomly initialized with different initial conditions. We have found that the hierarchical multi-layer approach can improve performance of most NMF algorithms discussed in this book [33], [27], [36].

1.2.9 Simultaneous NMF

In simultaneous NMF (siNMF) we have available two or more linked input data matrices (say, $\mathbf{Y}_1$ and $\mathbf{Y}_2$) and the objective is to decompose them into nonnegative factor matrices in such a way that one of a factor matrix is common, for example, (which is a special form of the

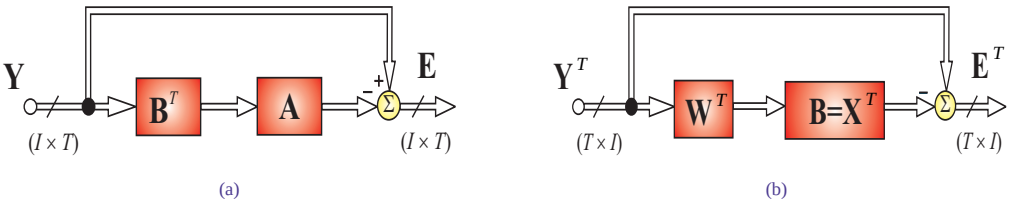


Fig. 1.9 (a) Illustration of Projective NMF (typically, $\mathbf{A} = \mathbf{B} = \mathbf{W}$) and (b) Convex NMF.

Nonnegative Tensor Factorization NTF2 model presented in Section 1.5.4),

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{A}_1\mathbf{X} + \mathbf{E}_1, \\ \mathbf{Y}_2 &= \mathbf{A}_2\mathbf{X} + \mathbf{E}_2. \end{aligned} \tag{1.27}$$

Such a problem arises, for example, in bio-informatics if we combine gene expression and transcription factor regulation [8]. In this application the data matrix $\mathbf{Y}_1 \in \mathbb{R}^{I_1 \times T}$ is the expression level of gene t in a data sample i_1 (i.e., the index i_1 denotes samples, while t stands for genes) and $\mathbf{Y}_2 \in \mathbb{R}^{I_2 \times T}$ is a transcription matrix (which is 1 whenever transcription factor i_2 regulates gene t).

1.2.10 Projective and Convex NMF

A projective NMF model can be formulated as the estimation of sparse and nonnegative matrix $\mathbf{W} \in \mathbb{R}_+^{I \times J}$, $I > J$, which satisfies the matrix equation

$$\mathbf{Y} = \mathbf{W}\mathbf{W}^T\mathbf{Y} + \mathbf{E}. \tag{1.28}$$

In a more general nonsymmetric form the projective NMF involves estimation of two nonnegative matrices: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{B} \in \mathbb{R}_+^{I \times J}$ in the model (see Figure 1.9(a)):

$$\mathbf{Y} = \mathbf{A}\mathbf{B}^T\mathbf{Y} + \mathbf{E}. \tag{1.29}$$

This may lead to the following optimization problem:

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}^T\mathbf{Y}\|_F^2, \qquad \text{s.t.} \;\; \mathbf{A} \geq \mathbf{0}, \;\; \mathbf{B} \geq \mathbf{0}. \tag{1.30}$$

The projective NMF is similar to the subspace PCA. However, it involves nonnegativity constraints.

In the convex NMF proposed by Ding, Li and Jordan [51], we assume that the basis vectors $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J]$ are constrained to be convex combinations of the data input matrix $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T]$. In other words, we require that the vectors $\boldsymbol{a}_j$ lie within the column space of the data matrix $\mathbf{Y}$, i.e.:

$$\boldsymbol{a}_j = \sum_{t=1}^{T} w_{tj}\boldsymbol{y}_t = \mathbf{Y}\boldsymbol{w}_j \qquad \text{or} \qquad \mathbf{A} = \mathbf{Y}\mathbf{W}, \tag{1.31}$$

where $\mathbf{W} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} = \mathbf{B}^T \in \mathbb{R}_+^{J \times T}$. Usually each column in $\mathbf{W}$ satisfies the sum-to-one constraint, i.e., they are unit length in terms of the ℓ_1-norm. We restrict ourselves to convex combinations of the columns of $\mathbf{Y}$. The convex NMF model can be written in the matrix form as[7]

$$\mathbf{Y} = \mathbf{Y}\mathbf{W}\mathbf{X} + \mathbf{E} \tag{1.32}$$

and we can apply the transpose operator to give

$$\mathbf{Y}^T = \mathbf{X}^T\mathbf{W}^T\mathbf{Y}^T + \mathbf{E}^T. \tag{1.33}$$

This illustrates that the convex NMF can be represented in a similar way to the projective NMF (see Figure 1.9(b)). The convex NMF usually implies that both nonnegative factors $\mathbf{A}$ and $\mathbf{B} = \mathbf{X}^T$ tend to be very sparse.

The standard cost function (squared Euclidean distance) can be expressed as

$$\|\mathbf{Y} - \mathbf{Y}\,\mathbf{W}\,\mathbf{B}^T\|_F^2 = \mathrm{tr}(\mathbf{I} - \mathbf{B}\,\mathbf{W}^T)\,\mathbf{Y}^T\,\mathbf{Y}\,(\mathbf{I} - \mathbf{W}\,\mathbf{B}^T) = \sum_{j=1}^{J} \lambda_j \,\|\,\boldsymbol{v}_j^T\,(\mathbf{I} - \mathbf{W}\,\mathbf{B}^T)\,\|_2^2, \tag{1.34}$$

where λ_j is the positive j-th eigenvalue (a diagonal entry of diagonal matrix $\mathbf{\Lambda}$) and $\boldsymbol{v}_j$ is the corresponding eigenvector for the eigenvalue decomposition: $\mathbf{Y}^T\mathbf{Y} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \sum_{j=1}^{J} \lambda_j \boldsymbol{v}_j \boldsymbol{v}_j^T$. This form of NMF can also be considered as a special form of the kernel NMF with a linear kernel defined as $\mathbf{K} = \mathbf{Y}^T\mathbf{Y}$.

1.2.11 Kernel NMF

The convex NMF leads to a natural extension of the kernel NMF [97], [91], [119]. Consider a mapping $\boldsymbol{y}_t \rightarrow \phi(\boldsymbol{y}_t)$ or $\mathbf{Y} \rightarrow \phi(\mathbf{Y}) = [\phi(\boldsymbol{y}_1), \phi(\boldsymbol{y}_2), \ldots, \phi(\boldsymbol{y}_T)]$, then the kernel NMF can be defined as

$$\phi(\mathbf{Y}) \cong \phi(\mathbf{Y})\,\mathbf{W}\,\mathbf{B}^T. \tag{1.35}$$

This leads to the minimization of the cost function:

$$\|\phi(\mathbf{Y}) - \phi(\mathbf{Y})\mathbf{W}\,\mathbf{B}^T\|_F^2 = \mathrm{tr}(\mathbf{K}) - 2\,\mathrm{tr}(\mathbf{B}^T\,\mathbf{K}\,\mathbf{W}) + \mathrm{tr}(\mathbf{W}^T\,\mathbf{K}\,\mathbf{W}\,\mathbf{B}^T\mathbf{B}), \tag{1.36}$$

which depends only on the kernel $\mathbf{K} = \phi^T(\mathbf{Y})\phi(\mathbf{Y})$.

1.2.12 Convolutive NMF

The Convolutive NMF (CNMF) is a natural extension and generalization of the standard NMF. In the Convolutive NMF, we process a set of nonnegative matrices or patterns which are horizontally shifted (or time delayed) versions of the primary matrix $\mathbf{X}$ [125]. In the simplest form

[7] In general, the convex NMF applies to both nonnegative data and mixed sign data which can be written symbolically as $\mathbf{Y}_\pm = \mathbf{Y}_\pm\mathbf{W}_+\mathbf{X}_+ + \mathbf{E}$.

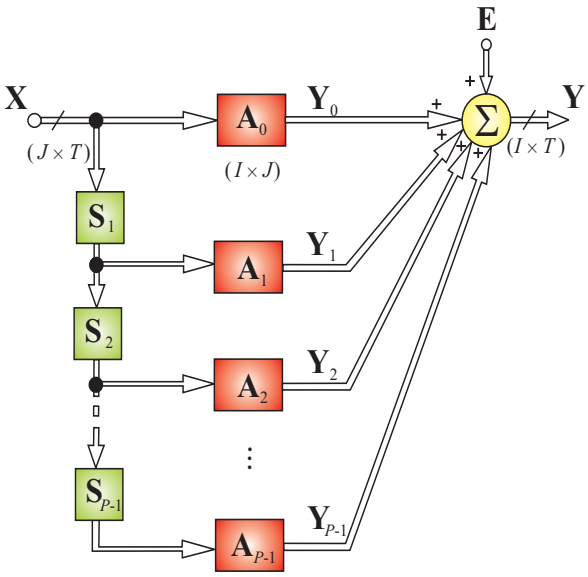


Fig. 1.10 Illustration of Convolutive NMF. The goal is to estimate the input sources represented by nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ (typically, $T >> I$) and to identify the convoluting system, i.e., to estimate a set of nonnegative matrices $\{\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{P-1}\}$ ($\mathbf{A}_p \in \mathbb{R}_+^{I \times J}$, $p = 0, 1, \ldots, P-1$) knowing only the input data matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$. Each operator $\mathbf{S}_p = \mathbf{T}_1$ ($p = 1, 2, \ldots, P-1$) performs a horizontal shift of the columns in $\mathbf{X}$ by one spot.

the CNMF can be described as (see Figure 1.10)

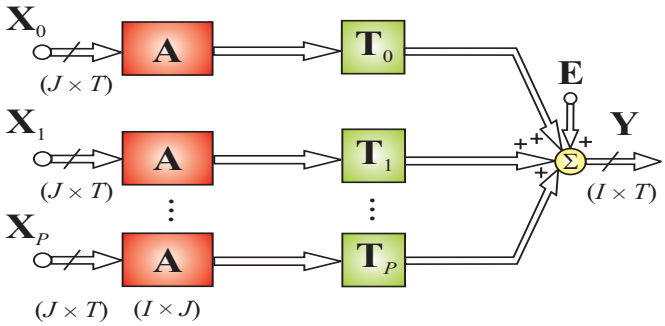
$$\mathbf{Y} = \sum_{p=0}^{P-1} \mathbf{A}_p \overset{p\rightarrow}{\mathbf{X}} + \mathbf{E}, \tag{1.37}$$

where $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$ is a given input data matrix, $\mathbf{A}_p \in \mathbb{R}_+^{I \times J}$ is a set of unknown nonnegative basis matrices, $\mathbf{X} = \overset{0\rightarrow}{\mathbf{X}} \in \mathbb{R}_+^{J \times T}$ is a matrix representing primary sources or patterns, $\overset{p\rightarrow}{\mathbf{X}}$ is a shifted by p columns version of $\mathbf{X}$. In other words, $\overset{p\rightarrow}{\mathbf{X}}$ means that the columns of $\mathbf{X}$ are shifted to the right p spots (columns), while the entries in the columns shifted into the matrix from the outside are set to zero. This shift (time-delay) is performed by a basic operator illustrated in Figure 1.10 as $\mathbf{S}_p = \mathbf{T}_1$. Analogously, $\overset{\leftarrow p}{\mathbf{Y}}$ means that the columns of $\mathbf{Y}$ are shifted to the left p spots. These notations will also be used for the shift operations of other matrices throughout this book (see Chapter 3 for more detail). Note that, $\overset{0\rightarrow}{\mathbf{X}} = \overset{\leftarrow 0}{\mathbf{X}} = \mathbf{X}$.
The shift operator is illustrated by the following example:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \overset{1\rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 4 & 5 \end{bmatrix}, \quad \overset{2\rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 4 \end{bmatrix}, \quad \overset{\leftarrow 1}{\mathbf{X}} = \begin{bmatrix} 2 & 3 & 0 \\ 5 & 6 & 0 \end{bmatrix}.$$

In the Convolutive NMF model, temporal continuity exhibited by many audio signals can be expressed more efficiently in the time-frequency domain, especially for signals whose frequen-

(a)



(b)

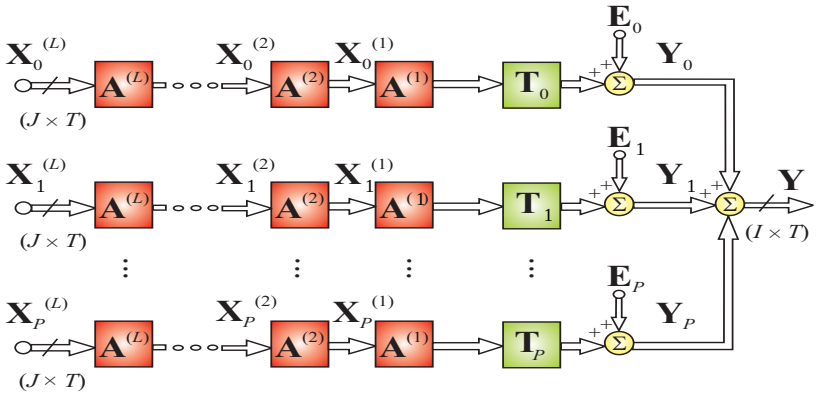


Fig. 1.11 (a) Block diagram schema for overlapping NMF, (b) Extended Multi-layer NMF model.

cies vary with time. We will present several efficient and extensively tested algorithms for the CNMF model in Chapter 3.

1.2.13 Overlapping NMF

In Convolutive NMF we perform horizontal shift of the columns of the matrix $\mathbf{X}$. In some applications, such as in spectrogram decomposition, we need to perform different transformations by shift vertically the rows of the matrix $\mathbf{X}$. For example, the observed data may be represented by a linear combination of horizontal bars or features and modeled by transposing the CNMF model (1.37) as

$$\mathbf{Y} \approx \sum_{p=0}^{P} (\overrightarrow{\mathbf{X}}^{p})^T \mathbf{A}_p^T = \sum_{p=0}^{P} (\mathbf{X}\underrightarrow{\mathbf{T}}_p)^T \mathbf{A}_p^T = \sum_{p=0}^{P} \underrightarrow{\mathbf{T}}_p^T \mathbf{X}^T \mathbf{A}_p^T, \tag{1.38}$$

where $\underrightarrow{\mathbf{T}}_p \triangleq \mathbf{T}_p$ is the horizontal-shift matrix operator such that $\overrightarrow{\mathbf{X}}^{p} = \mathbf{X}\underrightarrow{\mathbf{T}}_p$ and $\overleftarrow{\mathbf{X}}^{p} = \mathbf{X}\underleftarrow{\mathbf{T}}_p$. For

example, for the fourth-order identity matrix this operator can take the following form

$$\mathbf{T}_{\underset{\rightarrow}{1}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_{\underset{\rightarrow}{2}} = \mathbf{T}_{\underset{\rightarrow}{1}}\mathbf{T}_{\underset{\rightarrow}{1}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_{\underset{\leftarrow}{1}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Transposing the horizontal shift operator $\mathbf{T}_{\underset{\rightarrow}{p}} := \mathbf{T}_p$ gives us the vertical shift operator $\mathbf{T}_{\uparrow p} = \mathbf{T}^T_{\underset{\leftarrow}{p}}$ and $\mathbf{T}_{\downarrow p} = \mathbf{T}^T_{\underset{\rightarrow}{p}}$, in fact, we have $\mathbf{T}_{\uparrow p} = \mathbf{T}_{\underset{\rightarrow}{p}}$ and $\mathbf{T}_{\downarrow p} = \mathbf{T}_{\underset{\leftarrow}{p}}$.

It is interesting to note that by interchanging the role of matrices $\mathbf{A}$ and $\mathbf{X}$, that is, $\mathbf{A} \triangleq \mathbf{X}$ and $\mathbf{X}_p \triangleq \mathbf{A}_p$, we obtain the overlapping NMF introduced by Eggert *et al.* [56] and investigated by Choi *et al.* [81], which can be described as (see Fig. 1.11(a))

$$\boxed{\mathbf{Y} \cong \sum_{p=0}^{P} \mathbf{T}_{\uparrow p}\mathbf{A}\mathbf{X}_p.} \tag{1.39}$$

Figure 1.11(b) illustrates the extended multi-layer overlapping NMF (by analogy to the standard multi-layer NMF in order to improve the performance of the overlapping NMF). The overlapping NMF model can be considered as a modification or variation of the CNMF model, where transform-invariant representations and sparseness constraints are incorporated [56], [81].

1.3 BASIC APPROACHES TO ESTIMATE PARAMETERS OF STANDARD NMF

In order to estimate factor matrices $\mathbf{A}$ and $\mathbf{X}$ in the standard NMF, we need to consider the similarity measure to quantify a difference between the data matrix $\mathbf{Y}$ and the approximative NMF model matrix $\widehat{\mathbf{Y}} = \mathbf{A}\mathbf{X}$. The choice of the similarity measure (also referred to as distance, divergence or measure of dissimilarity) mostly depends on the probability distribution of the estimated signals or components and on the structure of data or a distribution of noise. The simplest and most often used measure is based on Frobenius norm:

$$D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \frac{1}{2}||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2, \tag{1.40}$$

which is also referred to as the squared Euclidean distance. It should be noted that the above cost function is convex with respect to either the elements of the matrix $\mathbf{A}$ or the matrix $\mathbf{X}$, but not both.[8] Alternating minimization of such a cost leads to the ALS (Alternating Least Squares) algorithm which can be described as follows:

1. Initialize $\mathbf{A}$ randomly or by using a specific deterministic strategy.

[8] Although the NMF optimization problem is not convex, the objective functions are separately convex in each of the two factors $\mathbf{A}$ and $\mathbf{X}$, which implies that finding the optimal factor matrix $\mathbf{A}$ corresponding to a fixed matrix $\mathbf{X}$ reduces to a convex optimization problem and vice versa. However, the convexity is lost as soon as we try to optimize factor matrices simultaneously [59].

2. Estimate $\mathbf{X}$ from the matrix equation $\mathbf{A}^T\mathbf{A}\mathbf{X} = \mathbf{A}^T\mathbf{Y}$ by solving

$$\min_{\mathbf{X}} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \frac{1}{2}||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2, \qquad \text{with fixed} \qquad \mathbf{A}. \tag{1.41}$$

3. Set all the negative elements of $\mathbf{X}$ to zero or some small positive value.

4. Estimate $\mathbf{A}$ from the matrix equation $\mathbf{X}\mathbf{X}^T\mathbf{A}^T = \mathbf{X}\mathbf{Y}^T$ by solving

$$\min_{\mathbf{A}} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \frac{1}{2}||\mathbf{Y}^T - \mathbf{X}^T\mathbf{A}^T||_F^2, \qquad \text{with fixed} \qquad \mathbf{X}. \tag{1.42}$$

5. Set all negative elements of $\mathbf{A}$ to zero or some small positive value ε.

The above ALS algorithm can be written in the following form:[9]

$$\mathbf{X} \leftarrow \max\left\{\varepsilon,\ (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}\right\} = [\mathbf{A}^\dagger\mathbf{Y}]_+, \tag{1.43}$$

$$\mathbf{A} \leftarrow \max\left\{\varepsilon,\ \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\right\} = [\mathbf{Y}\mathbf{X}^\dagger]_+, \tag{1.44}$$

where $\mathbf{A}^\dagger$ is the Moore-Penrose inverse of $\mathbf{A}$, ε is a small constant (typically, 10^{-16}) to enforce positive entries. Various additional constraints on $\mathbf{A}$ and $\mathbf{X}$ can be imposed.

Today the ALS method is considered as a basic "workhorse" approach, however it is not guaranteed to converge to a global minimum nor even a stationary point, but only to a solution where the cost functions cease to decrease [85], [11]. Moreover, it is often not sufficiently accurate. The ALS method can be dramatically improved and its computational complexity reduced as it will be shown in Chapter 4.

It is interesting to note that the NMF problem can be considered as a natural extension of a Nonnegative Least Squares (NLS) problem formulated as the following optimization problem: given a matrix $\mathbf{A} \in \mathbb{R}^{I\times J}$ and a set of observed values given by the vector $\boldsymbol{y} \in \mathbb{R}^I$, find a nonnegative vector $\boldsymbol{x} \in \mathbb{R}^J$ which minimizes the cost function $J(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{y} - \mathbf{A}\boldsymbol{x}||_2^2$, i.e.,

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \frac{1}{2}||\boldsymbol{y} - \mathbf{A}\,\boldsymbol{x}||_2^2, \tag{1.45}$$

subject to $\boldsymbol{x} \geq \mathbf{0}$. There is a large volume of literature devoted to the NLS problems which will be exploited and adopted in this book.

Another frequently used cost function for NMF is the generalized Kullback-Leibler divergence (also called the I-divergence) [94]:

$$D_{KL}(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \sum_{it}\left(y_{it}\ \ln\frac{y_{it}}{[\mathbf{A}\mathbf{X}]_{it}} - y_{it} + [\mathbf{A}\mathbf{X}]_{it}\right). \tag{1.46}$$

Most existing approaches minimize only one kind of cost function by alternately switching between sets of parameters. In this book we adopt a more general and flexible approach in which instead of one cost function we rather exploit two or more cost functions (with the same

[9] Note that the max operator is applied element-wise, that is, each element of a matrix is compared with scalar parameter ε.

global minima); one of them is minimized with respect to $\mathbf{A}$ and the other one with respect to $\mathbf{X}$. Such an approach is fully justified as $\mathbf{A}$ and $\mathbf{X}$ may have different distributions or different statistical properties and therefore different cost functions can be optimal for them.

Algorithm 1.1: Multi-layer NMF using alternating minimization of two cost functions

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data, J: rank of approximation
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ such that some given cost functions are minimized.

```
begin
    X = Y,  A = I
    for l = 1 to L do
        Initialize randomly A_(l) and X_(l)        [a]
        repeat
            A_(l) = arg min_{A_(l) ≥ 0} {D_1 (X || A_(l) X_(l))}  for fixed X_(l)
            X_(l) = arg min_{X_(l) ≥ 0} {D_2 (X || A_(l) X_(l))}  for fixed A_(l)
        until a stopping criterion is met                  /* convergence condition */
        X = X_(l)
        A ← A A_(l)
    end
end
```

[a] Instead of random initialization, we can use ALS or SVD based initialization, see Section 1.3.3.

Algorithm 1.1 illustrates such a case, where the cost functions $D_1(\mathbf{Y}||\mathbf{AX})$ and $D_2(\mathbf{Y}||\mathbf{AX})$ can take various forms, e.g.: I-divergence and Euclidean distance [49], [35] (see Chapter 2).

We can generalize this concept by using not one or two cost functions but rather a set of cost functions to be minimized sequentially or simultaneously. For $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_J]$ and $\mathbf{B} = \mathbf{X}^T = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J]$, we can express the squared Euclidean cost function as

$$\begin{aligned} J(\boldsymbol{a}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_J, \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J) &= \frac{1}{2}\|\mathbf{Y} - \mathbf{A}\mathbf{B}^T\|_F^2 \\ &= \frac{1}{2}\|\mathbf{Y} - \sum_{j=1}^{J} \boldsymbol{a}_j \boldsymbol{b}_j^T\|_F^2. \end{aligned} \tag{1.47}$$

An underlying idea is to define a residual (rank-one approximated) matrix (see Chapter 4 for more detail and explanation)

$$\mathbf{Y}^{(j)} \triangleq \mathbf{Y} - \sum_{p \neq j} \boldsymbol{a}_p \boldsymbol{b}_p^T \tag{1.48}$$

and alternately minimize the set of cost functions with respect to the unknown variables $\boldsymbol{a}_j, \boldsymbol{b}_j$:

$$D_A^{(j)}(\boldsymbol{a}) = \frac{1}{2}\|\mathbf{Y}^{(j)} - \boldsymbol{a}\, \boldsymbol{b}_j^T\|_F^2, \qquad \text{for a fixed } \boldsymbol{b}_j, \tag{1.49a}$$

$$D_B^{(j)}(\boldsymbol{b}) = \frac{1}{2}\|\mathbf{Y}^{(j)} - \boldsymbol{a}_j\, \boldsymbol{b}^T\|_F^2, \qquad \text{for a fixed } \boldsymbol{a}_j, \tag{1.49b}$$

for $j = 1, 2, \ldots, J$ subject to $\boldsymbol{a} \geq \mathbf{0}$ and $\boldsymbol{b} \geq \mathbf{0}$, respectively.

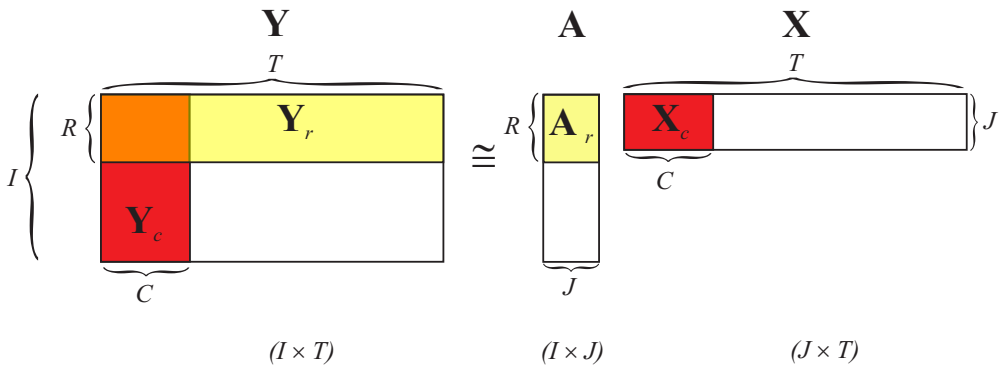


Fig. 1.12 Conceptual illustration of block-wise data processing for large-scale NMF. Instead of processing the whole matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$, we can process much smaller block matrices $\mathbf{Y}_c \in \mathbb{R}^{I \times C}$ and $\mathbf{Y}_r \in \mathbb{R}^{R \times T}$ and corresponding factor matrices $\mathbf{X}_c \in \mathbb{R}^{J \times C}$ and $\mathbf{A}_r \in \mathbb{R}^{R \times J}$ with $C << T$ and $R << I$. For simplicity of graphical illustration, we have assumed that the first R rows and the first C columns of the matrices $\mathbf{Y}$, $\mathbf{A}$ and $\mathbf{X}$ are selected.

1.3.1 Large-Scale NMF

In many applications, especially in dimension reduction applications the data matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$ can be very large (with millions of entries), but it can be approximately factorized using a rather smaller number of nonnegative components (J), that is, $J << I$ and $J << T$. Then the problem $\mathbf{Y} \approx \mathbf{AX}$ becomes highly redundant and we do not need to use information about all entries of $\mathbf{Y}$ in order to estimate precisely the factor matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{X} \in \mathbb{R}^{J \times T}$. In other words, to solve the large-scale NMF problem we do not need to know the whole data matrix but only a small random part of it. As we will show later, such an approach can outperform considerably the standard NMF methods, especially for extremely overdetermined systems.

In this approach, instead of performing large-scale factorization

$$\mathbf{Y} = \mathbf{AX} + \mathbf{E},$$

we can consider a two set of linked factorizations using much smaller matrices, given by

$$\mathbf{Y}_r = \mathbf{A}_r\mathbf{X} + \mathbf{E}_r, \quad \text{for fixed (known)} \quad \mathbf{A}_r, \tag{1.50}$$

$$\mathbf{Y}_c = \mathbf{A}\mathbf{X}_c + \mathbf{E}_c, \quad \text{for fixed (known)} \quad \mathbf{X}_c, \tag{1.51}$$

where $\mathbf{Y}_r \in \mathbb{R}_+^{R \times T}$ and $\mathbf{Y}_c \in \mathbb{R}_+^{I \times C}$ are the matrices constructed from the selected rows and columns of the matrix $\mathbf{Y}$, respectively. Analogously, we can construct the reduced matrices: $\mathbf{A}_r \in \mathbb{R}^{R \times J}$ and $\mathbf{X}_c \in \mathbb{R}^{J \times C}$ by using the same indices for the columns and rows as those used for the construction of the data sub-matrices $\mathbf{Y}_c$ and $\mathbf{Y}_r$. In practice, it is usually sufficient to choose: $J < R \leq 4J$ and $J < C \leq 4J$.

In the special case, for the squared Euclidean distance (Frobenius norm), instead of alternately minimizing the cost function

$$D_F(\mathbf{Y} \,\|\, \mathbf{AX}) = \frac{1}{2} \|\mathbf{Y} - \mathbf{AX}\|_F^2 , \tag{1.52}$$

we can minimize sequentially the two cost functions:

$$D_F(\mathbf{Y}_r \,||\, \mathbf{A}_r\mathbf{X}) = \frac{1}{2}||\mathbf{Y}_r - \mathbf{A}_r\mathbf{X}||_F^2, \qquad \text{for fixed} \qquad \mathbf{A}_r, \tag{1.53}$$

$$D_F(\mathbf{Y}_c \,||\, \mathbf{A}\mathbf{X}_c) = \frac{1}{2}||\mathbf{Y}_c - \mathbf{A}\mathbf{X}_c||_F^2, \qquad \text{for fixed} \qquad \mathbf{X}_c. \tag{1.54}$$

The minimization of these cost functions with respect to $\mathbf{X}$ and $\mathbf{A}$, subject to nonnegativity constraints, leads to the simple ALS update formulas for the large-scale NMF:

$$\boxed{\mathbf{X} \leftarrow \left[\mathbf{A}_r^{\dagger}\mathbf{Y}_r\right]_+ = \left[(\mathbf{A}_r^T\mathbf{A}_r)^{-1}\mathbf{A}_r\mathbf{Y}_r\right]_+, \qquad \mathbf{A} \leftarrow \left[\mathbf{Y}_c\mathbf{X}_c^{\dagger}\right]_+ = \left[\mathbf{Y}_c\mathbf{X}_c^T(\mathbf{X}_c\mathbf{X}_c^T)^{-1}\right]_+.} \tag{1.55}$$

A similar strategy can be applied for other cost functions and details will be given in Chapter 3 and Chapter 4.

There are several strategies to choose the columns and rows of the input data matrix [15], [100], [22], [66], [67]. The simplest scenario is to choose the first R rows and the first C columns of the data matrix $\mathbf{Y}$ (see Figure 1.12) or select them randomly using a uniform distribution. An optional strategy is to select randomly rows and columns from the set of all rows and columns with probability proportional to their relevance, e.g., with probability proportional to square of Euclidean ℓ_2-norm of rows and columns, i.e., $||\underline{y}_i||_2^2$ and $||y_t||_2^2$, respectively. Another heuristic option is to choose those rows and columns that provide the largest ℓ_p-norm. For noisy data with uncorrelated noise, we can construct new columns and rows as a local average (mean values) of some specific numbers of the columns and rows of raw data. For example, the first selected column is created as an average of the first M columns, the second column is an average of the next M columns, and so on; the same procedure applies for rows. Another strategy is to select optimal rows and columns using optimal CUR decomposition [22].

1.3.2 Non-uniqueness of NMF and Techniques to Alleviate the Ambiguity Problem

Usually, we perform NMF using the alternating minimization scheme (see Algorithm 1.1) of a set given objective functions. However, in general, such minimization does not guarantee a unique solution (neglecting unavoidable scaling and permutation ambiguities). Even the quadratic function with respect to both sets of arguments $\{\mathbf{A}\}$ and $\{\mathbf{X}\}$ may have many local minima, which makes NMF algorithms suffer from rotational indeterminacy (ambiguity). For example, consider the quadratic function:

$$D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = ||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2 = ||\mathbf{Y} - \mathbf{A}\mathbf{R}^{-1}\mathbf{R}\mathbf{X}||_F^2 = ||\mathbf{Y} - \tilde{\mathbf{A}}\tilde{\mathbf{X}}||_F^2. \tag{1.56}$$

There are many ways to select a rotational matrix $\mathbf{R}$ which is not necessarily nonnegative or not necessarily a generalized permutation matrix,[10] so that the transformed (rotated) $\tilde{\mathbf{A}} \neq \mathbf{A}$ and $\tilde{\mathbf{X}} \neq \mathbf{X}$ are nonnegative. Here, it is important to note that the inverse of a nonnegative matrix is nonnegative if and only if it is a generalized permutation matrix [118]. If we assume that $\mathbf{R} \geq 0$ and $\mathbf{R}^{-1} \geq 0$ (element-wise) which are sufficient conditions for the nonnegativity of the transform matrices $\mathbf{A}\mathbf{R}^{-1}$ and $\mathbf{R}\mathbf{X}$, then $\mathbf{R}$ must be a generalized permutation (also called mono-

[10] Generalized permutation matrix is a matrix with only one nonzero positive element in each row and each column.

mial) matrix, i.e., $\mathbf{R}$ can be expressed as a product of a nonsingular positive definite diagonal matrix and a permutation matrix. It is intuitively easy to understand that if the original matrices $\mathbf{X}$ and $\mathbf{A}$ are sufficiently sparse only a generalized permutation matrix $\mathbf{P} = \mathbf{R}$ can satisfy the nonnegativity constraints of any transform matrices and NMF is unique.

To illustrate rotational indeterminacy consider the following mixing and source matrices:

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 7 & 2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, \tag{1.57}$$

which give the output

$$\mathbf{Y} = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \mathbf{A}\mathbf{X} = \begin{bmatrix} 3x_1(t) + 2x_2(t) \\ 7x_1(t) + 2x_2(t) \end{bmatrix}. \tag{1.58}$$

It is clear that there exists another nonnegative decomposition which gives us the following components:

$$\mathbf{Y} = \begin{bmatrix} 3x_1(t) + 2x_2(t) \\ 7x_1(t) + 2x_2(t) \end{bmatrix} = \tilde{\mathbf{A}}\tilde{\mathbf{X}} = \begin{bmatrix} 0 & 1 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ 3x_1(t) + 2x_2(t) \end{bmatrix}, \tag{1.59}$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & 1 \\ 4 & 1 \end{bmatrix}, \quad \tilde{\mathbf{X}} = \begin{bmatrix} x_1(t) \\ 3x_1(t) + 2x_2(t) \end{bmatrix} \tag{1.60}$$

are new nonnegative components which do not come from the permutation or scaling indeterminacies.

However, incorporating some sparsity or smoothness measures to the objective function is sufficient to solve the NMF problem uniquely (up to unavoidable scale and permutation indeterminacies). The issues related to sparsity measures for NMF have been widely discussed [76], [54], [73], [36], [39], [144], and are addressed in almost all chapters in this book.

When no prior information is available, we should perform normalization of the columns in $\mathbf{A}$ and/or the rows in $\mathbf{X}$ to help mitigate the effects of rotation indeterminacies. Such normalization is usually performed by scaling the columns $\boldsymbol{a}_j$ of $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_J]$ as follows:

$$\mathbf{A} \leftarrow \mathbf{A}\mathbf{D}_A, \quad \text{where} \quad \mathbf{D}_A = \text{diag}(\|\boldsymbol{a}_1\|_p^{-1}, \|\boldsymbol{a}_2\|_p^{-1}, \ldots, \|\boldsymbol{a}_J\|_p^{-1}), \qquad p \in [0, \infty). \tag{1.61}$$

Heuristics based on extensive experimentations show that best results can be obtained for $p = 1$, i.e., when the columns of $\mathbf{A}$ are normalized to unit ℓ_1-norm. This may be justified by the fact that the mixing matrix should contain only a few dominant entries in each column, which is emphasized by the normalization to the unit ℓ_1-norms.[11] The normalization (1.61) for the alternating minimization scheme (Algorithm 1.1) helps to alleviate many numerical difficulties, like numerical instabilities or ill-conditioning, however, it makes searching for the global minimum more complicated.

Moreover, to avoid rotational ambiguity of NMF, the rows of $\mathbf{X}$ should be sparse or zero-grounded. To achieve this we may apply some preprocessing, sparsification, or filtering of the

[11] In the case when the columns of $\mathbf{A}$ and rows of $\mathbf{X}$ are both normalized, the standard NMF model $\mathbf{Y} \approx \mathbf{A}\mathbf{X}$ is converted to a three-factor NMF model $\mathbf{Y} \approx \mathbf{A}\mathbf{D}\mathbf{X}$, where $\mathbf{D} = \mathbf{D}_A\mathbf{D}_X$ is a diagonal scaling matrix.

input data. For example, we may remove the baseline from the input data $\mathbf{Y}$ by applying the affine NMF instead of the regular NMF, that is,

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \boldsymbol{a}_0\mathbf{1}_T^T + \mathbf{E}, \tag{1.62}$$

where $\boldsymbol{a}_0 \in \mathbb{R}_+^I$ is a vector selected in such a way that the unbiased matrix $\hat{\mathbf{Y}} = \mathbf{Y} - \boldsymbol{a}_0\mathbf{1}_T^T \in \mathbb{R}_+^{I\times T}$ has many zeros or close to zero entries (see Chapter 3 for algorithms).

In summary, in order to obtain a unique NMF solution (neglecting unavoidable permutation and scaling indeterminacies), we need to enforce at least one of the following techniques:

1. Normalize or filter the input data $\mathbf{Y}$, especially by applying the affine NMF model (1.62), in order to make the factorized matrices zero-grounded.
2. Normalize the columns of $\mathbf{A}$ and/or the rows of $\mathbf{X}$ to unit length.
3. Impose sparsity and/or smoothness constraints to the factorized matrices.

1.3.3 Initialization of NMF

The solution and convergence provided by NMF algorithms usually highly depends on initial conditions, i.e., its starting guess values, especially in a multivariate context. Thus, it is important to have efficient and consistent ways for initializing matrices $\mathbf{A}$ and/or $\mathbf{X}$. In other words, the efficiency of many NMF strategies is affected by the selection of the starting matrices. Poor initializations often result in slow convergence, and in certain instances may lead even to an incorrect or irrelevant solution. The problem of selecting appropriate starting initialization matrices becomes even more complicated for large-scale NMF problems and when certain structures or constraints are imposed on the factorized matrices involved. As a good initialization for one data set may be poor for another data set, to evaluate the efficiency of an initialization strategy and the algorithm we should perform uncertainty analysis such as Monte Carlo simulations. Initialization in NMF plays a key role since the objective function to be minimized may have many local minima, and the intrinsic alternating minimization in NMF is nonconvex, even though the objective function is strictly convex with respect to one set of variables. For example, the quadratic function:

$$D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = ||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2$$

is strictly convex in one set of variables, either $\mathbf{A}$ or $\mathbf{X}$, but not in both. The issues of initialization in NMF have been widely discussed in the literature [3], [82], [92], [14].
As a rule of thumb, we can obtain a robust initialization using the following steps:

1. First, we built up a search method for generating R initial matrices $\mathbf{A}$ and $\mathbf{X}$. This could be based on random starts or the output from a simple ALS NMF algorithm. The parameter R depends on the number of required iterations (typically, 10-20 is sufficient).
2. Run a specific NMF algorithm for each set of initial matrices and with a fixed but small number of iterations (typically, 10-20). As a result, the NMF algorithm provides R initial estimates of the matrices $\mathbf{A}^{(r)}$ and $\mathbf{X}^{(r)}$.
3. Select the estimates (denoted by $\mathbf{A}^{(r_{\min})}$ and $\mathbf{X}^{(r_{\min})}$) corresponding to the lowest value of the cost function (the best likelihood) among the R trials as initial values for the final factorization.

Algorithm 1.2: Multi-start initialization

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data,
J: rank of approximation , R: number of restarts,
K_{init}, K_{fin}: number of alternating steps for initialization and completion
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ such that a given cost function is minimized.

```
begin
    parfor r = 1 to R do                          /* process in parallel mode */
        Initialize randomly A^(0) or X^(0)
        {A^(r), X^(r)} ← nmf_algorithm(Y, A^(0), X^(0), K_init)
        d_r = D(Y||A^(r)X^(r))                    /* compute the cost value */
    endfor
    r_min = arg min_{1≤r≤R} d_r
    {A, X} ← nmf_algorithm(Y, A^(r_min), X^(r_min), K_fin)
end
```

In other words, the main idea is to find good initial estimates ("candidates") with the following multi-start initialization algorithm:

Thus, the multi-start initialization selects the initial estimates for $\mathbf{A}$ and $\mathbf{X}$ which give the steepest decrease in the assumed objective function $D(\mathbf{Y}||\mathbf{AX})$ via alternating steps. Usually, we choose the generalized Kullback-Leibler divergence $D_{KL}(\mathbf{Y}||\mathbf{AX})$ for checking the convergence results after K_{init} initial alternating steps. The initial estimates $\mathbf{A}^{(0)}$ and $\mathbf{X}^{(0)}$ which give the lowest values of $D_{KL}(\mathbf{Y}||\mathbf{AX})$ after K_{init} alternating steps are expected to be the most suitable candidates for continuing the alternating minimization. In practice, for $K_{init} \geq 10$, the algorithm works quite efficiently.

Throughout this book, we shall explore various alternative methods for the efficient initialization of the iterative NMF algorithms and provide supporting pseudo-source codes and MATLAB codes; for example, we use extensively the ALS-based initialization technique as illustrated by the following MATLAB code:

Listing 1.1 Basic initializations for NMF algorithms.

```
function [Ainit,Xinit] = NMFinitialization(Y,J,inittype)
% Y     :      nonnegative matrix
% J     :      number of components
% inittype     1 {random}, 2 {ALS}, 3 {SVD}
[I,T] = size(Y);
Ainit = rand(I,J);
Xinit = rand(J,T);

switch inittype
    case 2 % ALS
        Ainit = max(eps,(Y*Xinit')*pinv(Xinit*Xinit'));
        Xinit = max(eps,pinv(Ainit'*Ainit)*(Ainit'*Y));
    case 3  %SVD
        [Ainit,Xinit] = lsvNMF(Y,J);
end
Ainit = Ainit*bsxfun(@rdivide,Ainit,sum(Ainit));
end
```

1.3.4 Stopping Criteria

There are several possible stopping criteria for the iterative algorithms used in NMF:

- The cost function achieves a zero-value or a value below a given threshold ε, for example,

$$D_F^{(k)}(\mathbf{Y} \,\|\, \hat{\mathbf{Y}}^{(k)}) - \left\|\mathbf{Y} - \hat{\mathbf{Y}}^{(k)}\right\|_F^2 \leq \varepsilon\,. \tag{1.63}$$

- There is little or no improvement between successive iterations in the minimization of a cost function, for example,

$$D_F^{(k+1)}(\hat{\mathbf{Y}}^{(k+1)} \,\|\, \hat{\mathbf{Y}}^{(k)}) = \left\|\hat{\mathbf{Y}}^{(k)} - \hat{\mathbf{Y}}^{(k+1)}\right\|_F^2 \leq \varepsilon, \tag{1.64}$$

  or

$$\frac{|D_F^{(k)} - D_F^{(k-1)|}|}{D_F^{(k)}} \leq \varepsilon\,. \tag{1.65}$$

- There is little or no change in the updates for factor matrices $\mathbf{A}$ and $\mathbf{X}$.
- The number of iterations achieves or exceeds a predefined maximum number of iterations.

In practice, the iterations usually continue until some combinations of stopping conditions are satisfied. Some more advanced stopping criteria are discussed in Chapter 5.

1.4 TENSOR PROPERTIES AND BASIS OF TENSOR ALGEBRA

Matrix factorization models discussed in the previous sections can be naturally extended and generalized to multi-way arrays, also called multi-dimensional matrices or simply tensor decompositions.[12]

1.4.1 Tensors (Multi-way Arrays) – Preliminaries

A tensor is a multi-way array or multi-dimensional matrix. The order of a tensor is the number of dimensions, also known as ways or modes. Tensor can be formally defined as

Definition 1.1 (Tensor) *Let $I_1, I_2, \ldots, I_N \in \mathbb{N}$ denote index upper bounds. A tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of order N is an N-way array where elements $y_{i_1 i_2 \cdots i_n}$ are indexed by $i_n \in \{1, 2, \ldots, I_n\}$ for $1 \leq n \leq N$.*

Tensors are obviously generalizations of vectors and matrixes, for example, a third-order tensor (or three-way array) has three modes (or indices or dimensions) as shown in Figure 1.13. A zero-order tensor is a scalar, a first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three and higher are called higher-order tensors (see Figure 1.14).

[12] The notion of tensors used in this book should not be confused with field tensors used in physics and differential geometry, which are generally referred to as tensor fields (i.e., tensor-valued functions on manifolds) in mathematics [85]. Examples include, stress tensor, moment-of inertia tensor, Einstein tensor, metric tensor, curvature tensor, Ricci tensor.

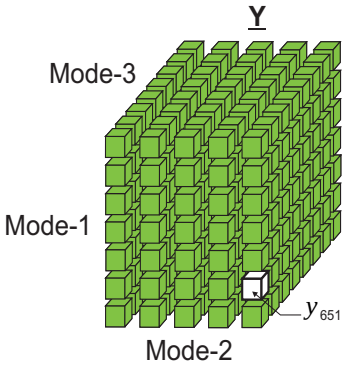


Fig. 1.13 A three-way array (third-order tensor) $\underline{\mathbf{Y}} \in \mathbb{R}^{7\times5\times8}$ with elements y_{itq}.

Generally, tensors are denoted by an underlined capital boldface letters, e.g., $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. In contrast, matrices are denoted by boldface capital letters, e.g., $\mathbf{Y}$; vectors are denoted by boldface lowercase letters, e.g., columns of the matrix $\mathbf{A}$ by $\boldsymbol{a}_j$ and scalars are denoted by lowercase letters, e.g., a_{ij}. The i-th entry of a vector $\boldsymbol{a}$ is denoted by a_i, and the (i, j)-th element of a matrix $\mathbf{A}$ by a_{ij}. Analogously, the element (i, t, q) of a third-order tensor $\mathbf{Y} \in \mathbb{R}^{I\times T\times Q}$ is denoted by y_{itq}. The values of indices are typically ranging from 1 to their capital version, e.g., $i = 1, 2, \ldots, I;\ t = 1, 2, \ldots, T; q = 1, 2, \ldots, Q$.

1.4.2 Subarrays, Tubes and Slices

Subtensors or subarrays are formed when a subset of the indices is fixed. For matrices, these are the rows and columns. A colon is used to indicate all elements of a mode in the style of MATLAB. Thus, the j-th column of a matrix $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J]$ is formally denoted by $\boldsymbol{a}_{:j}$; likewise, the j-th row of $\mathbf{X}$ is denoted by $\underline{\boldsymbol{x}}_j = \boldsymbol{x}_{j:}$.

Definition 1.2 (Tensor Fiber) *A tensor fiber is a one-dimensional fragment of a tensor, obtained by fixing all indices except for one.*

A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $\boldsymbol{y}_{:tq}, \boldsymbol{y}_{i:q}$, and $\boldsymbol{y}_{it:}$, respectively (see Figure 1.15). Note that fibers are always assumed to be oriented as column vectors [85].

Definition 1.3 (Tensor Slice) *A tensor slice is a two-dimensional section (fragment) of a tensor, obtained by fixing all indices except for two indices.*

Figure 1.16 shows the horizontal, lateral, and frontal slices of a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times T\times Q}$, denoted respectively by $\mathbf{Y}_{i::}$, $\mathbf{Y}_{:t:}$ and $\mathbf{Y}_{::q}$ (see also Figure 1.17). Two special subarrays have more compact representations: the j-th column of matrix $\mathbf{A}$, $\boldsymbol{a}_{:j}$, may also be denoted as $\boldsymbol{a}_j$, whereas the q-th frontal slice of a third-order tensor, $\mathbf{Y}_{::q}$ may also be denoted as $\mathbf{Y}_q$, $(q = 1, 2, \ldots, Q)$.

1.4.3 Unfolding – Matricization

It is often very convenient to represent tensors as matrices or to represent multi-way relationships and a tensor decomposition in their matrix forms. Unfolding, also known as matricization or

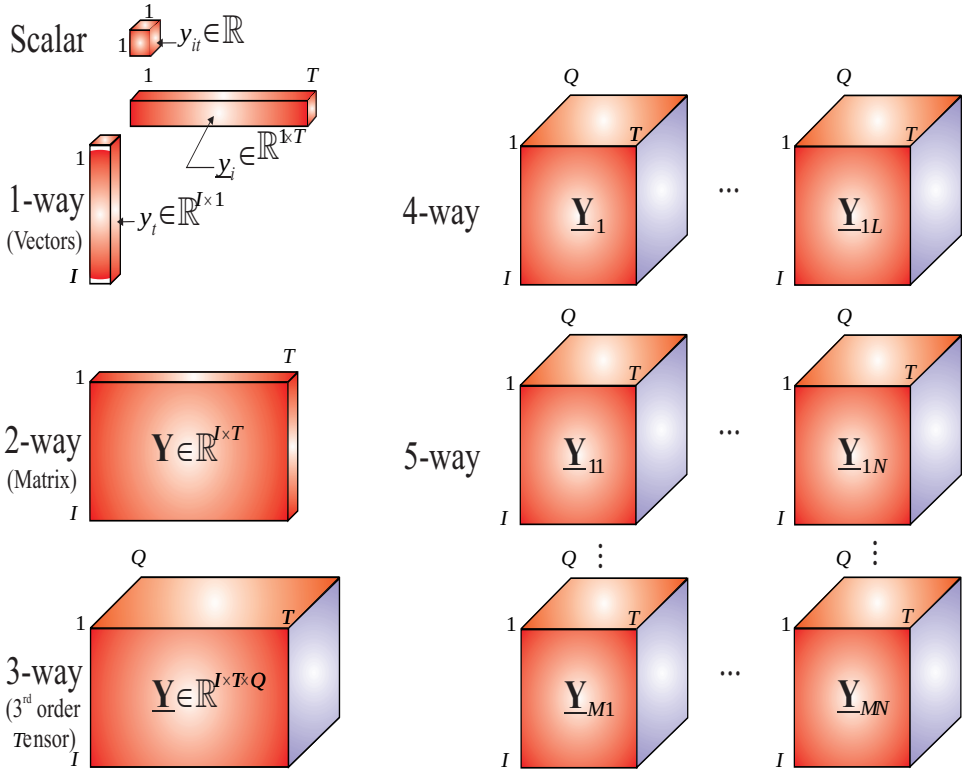


Fig. 1.14 Illustration of multi-way data: zero-way tensor = scalar, 1-way tensor = row or column vector, 2-way tensor = matrix, N-way tensor = higher-order tensors. The 4-way and 5-way tensors are represented here as a set of the three-way tensors.

flattening, is a process of reordering the elements of an N-th order tensor into a matrix. There are various ways to order the fibers of tensors, therefore, the unfolding process is not unique. Since the concept is easy to understand by examples, Figures 1.18, 1.19 and 1.20 illustrate the various unfolding processes of a three-way array. For example, for a third-order tensor we can arrange frontal, horizontal and lateral slices in row-wise and column-wise ways. Generally speaking, the unfolding of an N-th order tensor can be understood as the process of the construction of a matrix containing all the mode-n vectors of the tensor. The order of the columns is not unique and in this book it is chosen in accordance with [85] and based on the following definition:

Definition 1.4 (Unfolding) *The mode-n unfolding of tensor* $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *is denoted by*[13] $\mathbf{Y}_{(n)}$ *and arranges the mode-n fibers into columns of a matrix. More specifically, a tensor element* $(i_1, i_2, \ldots, i_N)$ *maps onto a matrix element* (i_n, j)*, where*

$$j = 1 + \sum_{p \neq n} (i_p - 1) J_p, \quad \text{with} \quad J_p = \begin{cases} 1, & \text{if } p = 1 \text{ or if } p = 2 \text{ and } n = 1, \\ \prod_{m \neq n}^{p-1} I_m, & \text{otherwise.} \end{cases} \tag{1.66}$$

[13] We use the Kolda - Bader notations [85].

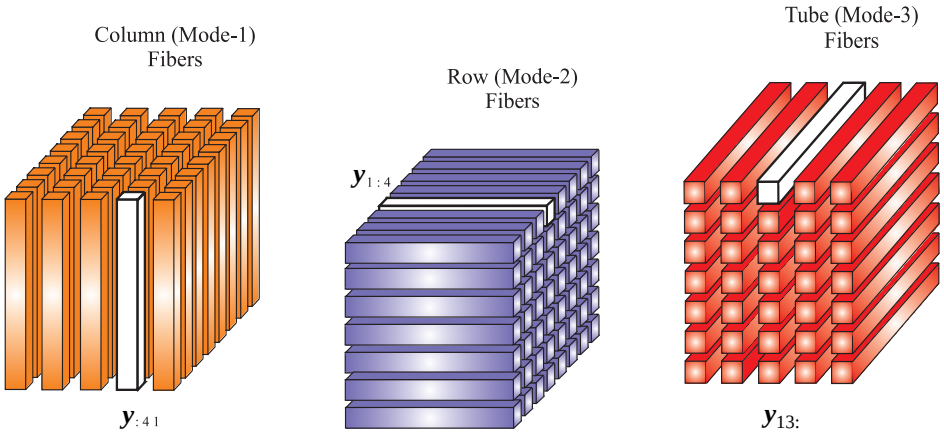


Fig. 1.15 Fibers: for a third-order tensor $\underline{\mathbf{Y}} = [y_{itq}] \in \mathbb{R}^{I \times T \times Q}$ (all fibers are treated as column vectors).

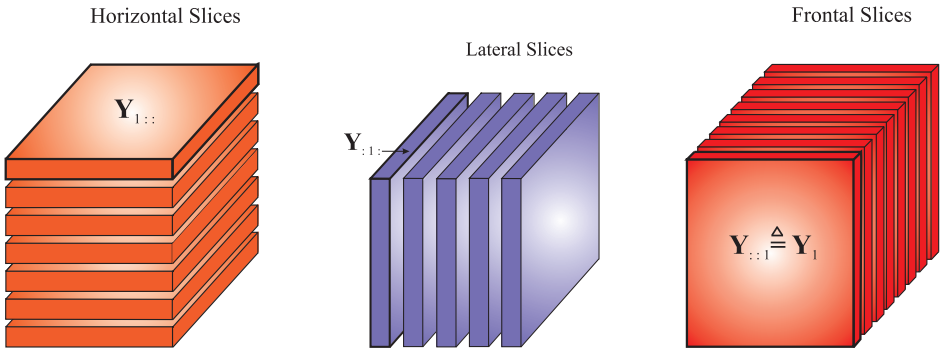


Fig. 1.16 Slices for a third-order tensor $\underline{\mathbf{Y}} = [y_{itq}] \in \mathbb{R}^{I \times T \times Q}$.

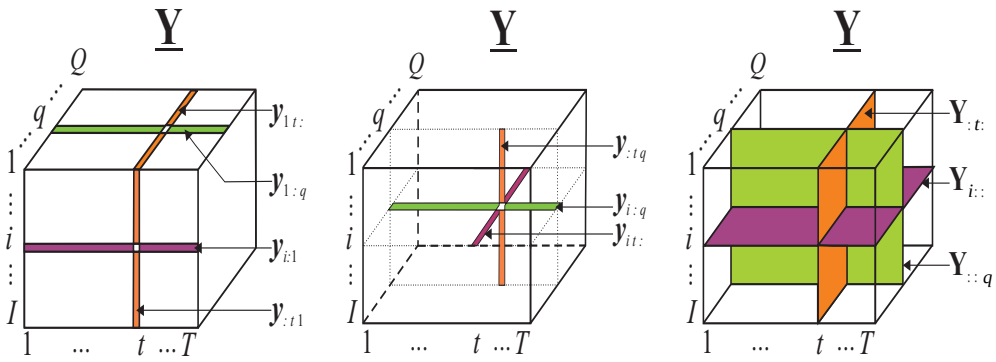


Fig. 1.17 Illustration of subsets (subarrays) of a three-way tensor and basic tensor notations of tubes and slices.

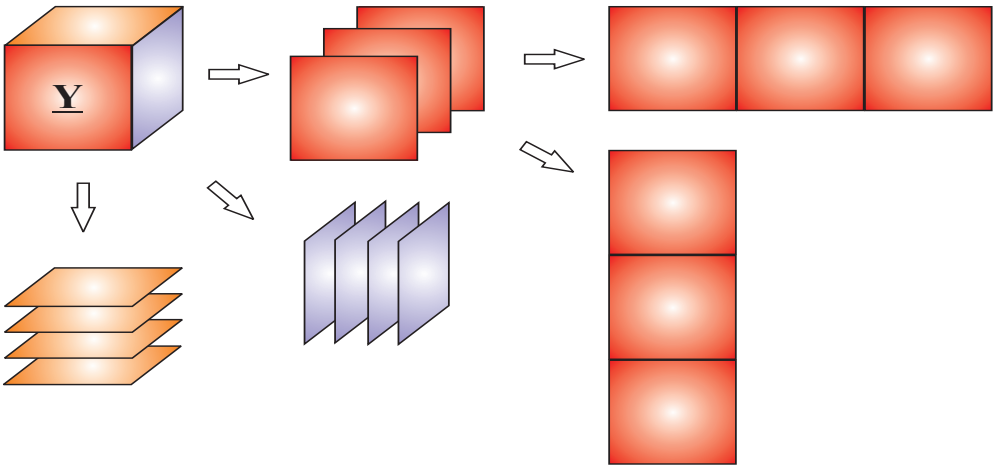


Fig. 1.18 Illustration of row-wise and column-wise unfolding (flattening, matricizing) of a third-order tensor.

Observe that in the mode-n unfolding the mode-n fibers are rearranged to be the columns of the matrix $\mathbf{Y}_{(n)}$.

More generally, a subtensor of the tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, denoted by $\underline{\mathbf{Y}}_{(i_n=j)}$, is obtained by fixing the n-th index to some value j. For example, a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with entries y_{i_1,i_2,i_3} and indices (i_1, i_2, i_3) has a corresponding position (i_n, j) in the mode-n unfolded matrix $\mathbf{Y}_{(n)}$ $(n = 1, 2, 3)$ as follows

- mode-1: $j = i_2 + (i_3 - 1)I_2$,
- mode-2: $j = i_1 + (i_3 - 1)I_1$,
- mode-3: $j = i_1 + (i_2 - 1)I_1$.

Note that mode-n unfolding of a tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ also represents mode-1 unfolding of its permuted tensor $\underline{\tilde{\mathbf{Y}}} \in \mathbb{R}^{I_n \times I_1 \cdots \times I_{n-1} \times I_{n+1} \cdots \times I_N}$ obtained by permuting its modes to obtain the mode-1 be I_n.

1.4.4 Vectorization

It is often convenient to represent tensors and matrices as vectors, whereby vectorization of matrix $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T] \in \mathbb{R}^{I \times T}$ is defined as

$$\boldsymbol{y} = \text{vec}(\mathbf{Y}) = \left[\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, \ldots, \boldsymbol{y}_T^T\right]^T \in \mathbb{R}^{IT}. \tag{1.67}$$

The vec-operator applied on a matrix $\mathbf{Y}$ stacks its columns into a vector. The reshape is a reverse function to vectorization which converts a vector to a matrix. For example, $\text{reshape}(\boldsymbol{y}, I, T) \in \mathbb{R}^{I \times T}$ is defined as (using MATLAB notations and similar to the reshape MATLAB function):

$$\text{reshape}(\boldsymbol{y}, I, T) = [\boldsymbol{y}(1 : I), \boldsymbol{y}(I + 1 : 2I), \ldots, \boldsymbol{y}((T - 1)I : IT)] \in \mathbb{R}^{I \times T}. \tag{1.68}$$

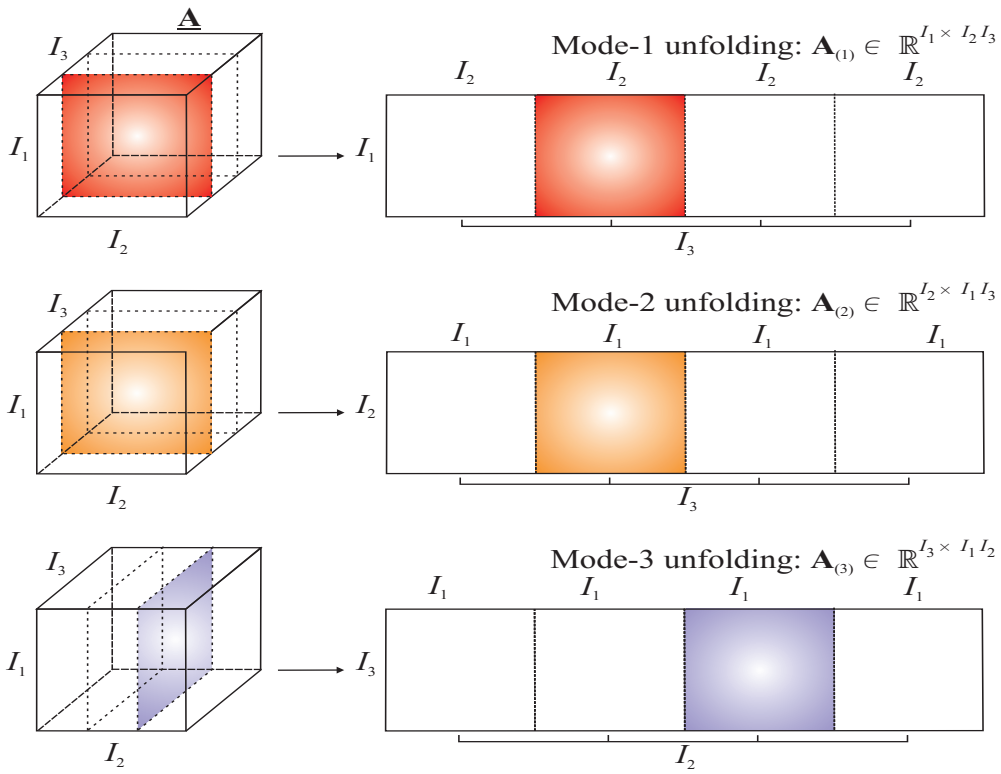


Fig. 1.19 Unfolding (matricizing) of a third-order tensor. The tensor can be unfolded in three ways to obtain matrices comprising its mode-1, mode-2 and mode-3 vectors.

Analogously, we define the vectorization of a tensor $\underline{\mathbf{Y}}$ as a vectorization of the associated mode-1 unfolded matrix $\mathbf{Y}_{(1)}$. For example, the vectorization of the third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ can be written in the following form

$$\text{vec}(\underline{\mathbf{Y}}) = \text{vec}(\mathbf{Y}_{(1)}) = \left[\text{vec}(\mathbf{Y}_{::1})^T, \text{vec}(\mathbf{Y}_{::2})^T, \ldots, \text{vec}(\mathbf{Y}_{::Q})^T\right]^T \in \mathbb{R}^{ITQ}. \tag{1.69}$$

Basic properties of the vec-operators include (assuming that matrices are appropriate sizes):

$$\text{vec}(c\,\mathbf{A}) = c\,\text{vec}(\mathbf{A}), \tag{1.70}$$
$$\text{vec}(\mathbf{A} + \mathbf{B}) = \text{vec}(\mathbf{A}) + \text{vec}(\mathbf{B}), \tag{1.71}$$
$$\text{vec}(\mathbf{A})^T\,\text{vec}(\mathbf{B}) = \text{trace}(\mathbf{A}^T\mathbf{B}), \tag{1.72}$$
$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B}). \tag{1.73}$$

1.4.5 Outer, Kronecker, Khatri-Rao and Hadamard Products

Several special matrix products are important for representation of tensor factorizations and decompositions.

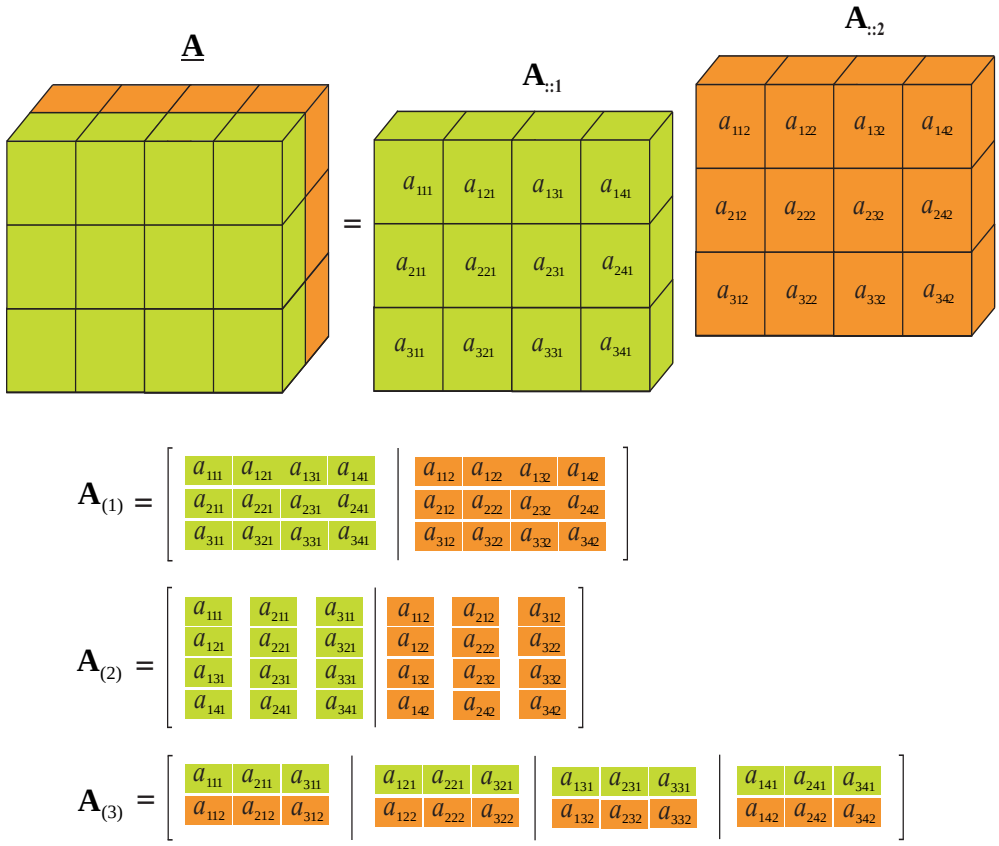


Fig. 1.20 Example of unfolding the third-order tensor in mode-1, mode-2 and mode-3.

1.4.5.1 **Outer Product** The outer product of the tensors $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{\mathbf{X}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ is given by

$$\underline{\mathbf{Z}} = \underline{\mathbf{Y}} \circ \underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_M}, \tag{1.74}$$

where

$$z_{i_1, i_2, \ldots, i_N, j_1, j_2, \ldots, j_M} = y_{i_1, i_2, \ldots, i_N} \, x_{j_1, j_2, \ldots, j_M}. \tag{1.75}$$

Observe that, the tensor $\underline{\mathbf{Z}}$ contains all the possible combinations of pair-wise products between the elements of $\underline{\mathbf{Y}}$ and $\underline{\mathbf{X}}$.

As special cases, the outer product of two vectors $\boldsymbol{a} \in \mathbb{R}^I$ and $\boldsymbol{b} \in \mathbb{R}^J$ yields a rank-one matrix

$$\mathbf{A} = \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T \in \mathbb{R}^{I \times J} \tag{1.76}$$

and the outer product of three vectors: $\boldsymbol{a} \in \mathbb{R}^I$, $\boldsymbol{b} \in \mathbb{R}^J$ and $\boldsymbol{c} \in \mathbb{R}^Q$ yields a third-order rank-one tensor:

$$\underline{\mathbf{Z}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c} \in \mathbb{R}^{I \times J \times Q}, \tag{1.77}$$

where

$$z_{ijq} = a_i\, b_j\, c_q. \tag{1.78}$$

1.4.5.2 **Kronecker Product** The Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{I\times J}$ and $\mathbf{B} \in \mathbb{R}^{T\times R}$ is a matrix denoted as $\mathbf{A}\otimes\mathbf{B} \in \mathbb{R}^{IT\times JR}$ and defined as (see the MATLAB function `kron`):

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}\,\mathbf{B} & a_{12}\,\mathbf{B} & \cdots & a_{1J}\,\mathbf{B} \\ a_{21}\,\mathbf{B} & a_{22}\,\mathbf{B} & \cdots & a_{2J}\,\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\,\mathbf{B} & a_{I2}\,\mathbf{B} & \cdots & a_{IJ}\,\mathbf{B} \end{bmatrix} \tag{1.79}$$

$$= \begin{bmatrix} \boldsymbol{a}_1\otimes\boldsymbol{b}_1 & \boldsymbol{a}_1\otimes\boldsymbol{b}_2 & \boldsymbol{a}_1\otimes\boldsymbol{b}_3 & \cdots & \boldsymbol{a}_J\otimes\boldsymbol{b}_{R-1} & \boldsymbol{a}_J\otimes\boldsymbol{b}_R \end{bmatrix}. \tag{1.80}$$

For any given three matrices $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ of (appropriate size), where $\mathbf{B}$ and $\mathbf{C}$ have the same size, the following properties hold:

$$(\mathbf{A}\otimes\mathbf{B})^T = \mathbf{A}^T\otimes\mathbf{B}^T, \tag{1.81}$$

$$(\mathbf{A}\otimes\mathbf{B})^\dagger = \mathbf{A}^\dagger\otimes\mathbf{B}^\dagger, \tag{1.82}$$

$$\mathbf{A}\otimes(\mathbf{B}+\mathbf{C}) = (\mathbf{A}\otimes\mathbf{B}) + (\mathbf{A}\otimes\mathbf{C}), \tag{1.83}$$

$$(\mathbf{B}+\mathbf{C})\otimes\mathbf{A} = (\mathbf{B}\otimes\mathbf{A}) + (\mathbf{C}\otimes\mathbf{A}), \tag{1.84}$$

$$(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = \mathbf{AC}\otimes\mathbf{BD}, \tag{1.85}$$

$$c\,(\mathbf{A}\otimes\mathbf{B}) = (c\,\mathbf{A})\otimes\mathbf{B} = \mathbf{A}\otimes(c\,\mathbf{B}). \tag{1.86}$$

It should be mentioned that, in general, the outer product of vectors yields a tensor whereas the Kronecker product gives a vector. For example, for the three vectors $\boldsymbol{a}\in\mathbb{R}^J$, $\boldsymbol{b}\in\mathbb{R}^T$, $\boldsymbol{c}\in\mathbb{R}^Q$ their three-way outer product $\underline{\mathbf{Y}} = \boldsymbol{a}\circ\boldsymbol{b}\circ\boldsymbol{c} \in \mathbb{R}^{I\times T\times Q}$ is a third-order tensor with the entries $y_{itq} = a_j b_t c_q$, while the three-way Kronecker product of the same vectors is a vector $\text{vec}(\underline{\mathbf{Y}}) = \boldsymbol{c}\otimes\boldsymbol{b}\otimes\boldsymbol{a} \in \mathbb{R}^{ITQ}$.

1.4.5.3 **Hadamard Product** The Hadamard product of two equal-size matrices is the element-wise product denoted by $\circledast$ (or $.*$ for MATLAB notation) and defined as

$$\mathbf{A}\circledast\mathbf{B} = \begin{bmatrix} a_{11}\,b_{11} & a_{12}\,b_{12} & \cdots & a_{1J}\,b_{1J} \\ a_{21}\,b_{21} & a_{22}\,b_{22} & \cdots & a_{2J}\,b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\,b_{I1} & a_{I2}\,b_{I2} & \cdots & a_{IJ}\,b_{IJ} \end{bmatrix}. \tag{1.87}$$

1.4.5.4 **Khatri-Rao Product** For two matrices $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I\times J}$ and $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J] \in \mathbb{R}^{T\times J}$ with the same number of columns J, their Khatri-Rao product, denoted by $\odot$, performs the following operation:

$$\mathbf{A}\odot\mathbf{B} = [\boldsymbol{a}_1\otimes\boldsymbol{b}_1 \;\; \boldsymbol{a}_2\otimes\boldsymbol{b}_2 \;\cdots\; \boldsymbol{a}_J\otimes\boldsymbol{b}_J] \tag{1.88}$$

$$= \left[\text{vec}(\boldsymbol{b}_1\boldsymbol{a}_1^T) \;\; \text{vec}(\boldsymbol{b}_2\boldsymbol{a}_2^T) \;\cdots\; \text{vec}(\boldsymbol{b}_J\boldsymbol{a}_J^T)\right] \in \mathbb{R}^{IT\times J}. \tag{1.89}$$

The Khatri-Rao product is:

- associative
$$\mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) = (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C}, \tag{1.90}$$
- distributive
$$(\mathbf{A} + \mathbf{B}) \odot \mathbf{C} = \mathbf{A} \odot \mathbf{C} + \mathbf{B} \odot \mathbf{C}, \tag{1.91}$$
- non-commutative
$$\mathbf{A} \odot \mathbf{B} \neq \mathbf{B} \odot \mathbf{A}, \tag{1.92}$$
- its cross-product simplifies into
$$(\mathbf{A} \odot \mathbf{B})^T (\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^T \mathbf{A} \circledast \mathbf{B}^T \mathbf{B}, \tag{1.93}$$
- and the Moore-Penrose pseudo-inverse can be expressed as
$$(\mathbf{A} \odot \mathbf{B})^\dagger = [(\mathbf{A} \odot \mathbf{B})^T (\mathbf{A} \odot \mathbf{B})]^{-1} (\mathbf{A} \odot \mathbf{B})^T = [(\mathbf{A}^T \mathbf{A}) \circledast (\mathbf{B}^T \mathbf{B})]^{-1} (\mathbf{A} \odot \mathbf{B})^T, \tag{1.94}$$
$$((\mathbf{A} \odot \mathbf{B})^T)^\dagger = (\mathbf{A} \odot \mathbf{B})[(\mathbf{A}^T \mathbf{A}) \circledast (\mathbf{B}^T \mathbf{B})]^{-1}. \tag{1.95}$$

1.4.6 Mode-n Multiplication of Tensor by Matrix and Tensor by Vector, Contracted Tensors Product

To multiply a tensor by a matrix, we need to specify which mode of the tensor is multiplied by the columns (or rows) of a matrix (see Figure 1.21 and Table 1.1).

Definition 1.5 (mode-n tensor matrix product) *The mode-n product* $\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_n \mathbf{A}$ *of a tensor* $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ *and a matrix* $\mathbf{A} \in \mathbb{R}^{I_n \times J_n}$ *is a tensor* $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times I_n \times J_{n+1} \times \cdots \times J_N}$*, with elements*

$$y_{j_1, j_2, \ldots, j_{n-1}, i_n, j_{n+1}, \ldots, j_N} = \sum_{j_n=1}^{J_n} g_{j_1, j_2, \ldots, J_N} \, a_{i_n, j_n}. \tag{1.96}$$

The tensor-matrix product can be applied successively along several modes, and it is commutative, that is

$$(\underline{\mathbf{G}} \times_n \mathbf{A}) \times_m \mathbf{B} = (\underline{\mathbf{G}} \times_m \mathbf{B}) \times_n \mathbf{A} = \underline{\mathbf{G}} \times_n \mathbf{A} \times_m \mathbf{B}, \qquad (m \neq n). \tag{1.97}$$

The repeated (iterated) mode-n tensor-matrix product for matrices $\mathbf{A}$ and $\mathbf{B}$ of appropriate dimensions can be simplified as

$$(\underline{\mathbf{G}} \times_n \mathbf{A}) \times_n \mathbf{B} = \underline{\mathbf{G}} \times_n (\mathbf{B}\mathbf{A}). \tag{1.98}$$

For $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and a set of matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, their multiplication in all possible modes $(n = 1, 2, \ldots, N)$ is denoted as

$$\underline{\mathbf{G}} \times \{\mathbf{A}\} = \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}, \tag{1.99}$$

and the resulting tensor has dimension $I_1 \times I_2 \times \cdots \times I_N$. Multiplication of a tensor with all but one mode is denoted as

$$\underline{\mathbf{G}} \times_{-n} \{\mathbf{A}\} = \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \cdots \times_{n-1} \mathbf{A}^{(n-1)} \times_{n+1} \mathbf{A}^{(n+1)} \cdots \times_N \mathbf{A}^{(N)} \tag{1.100}$$

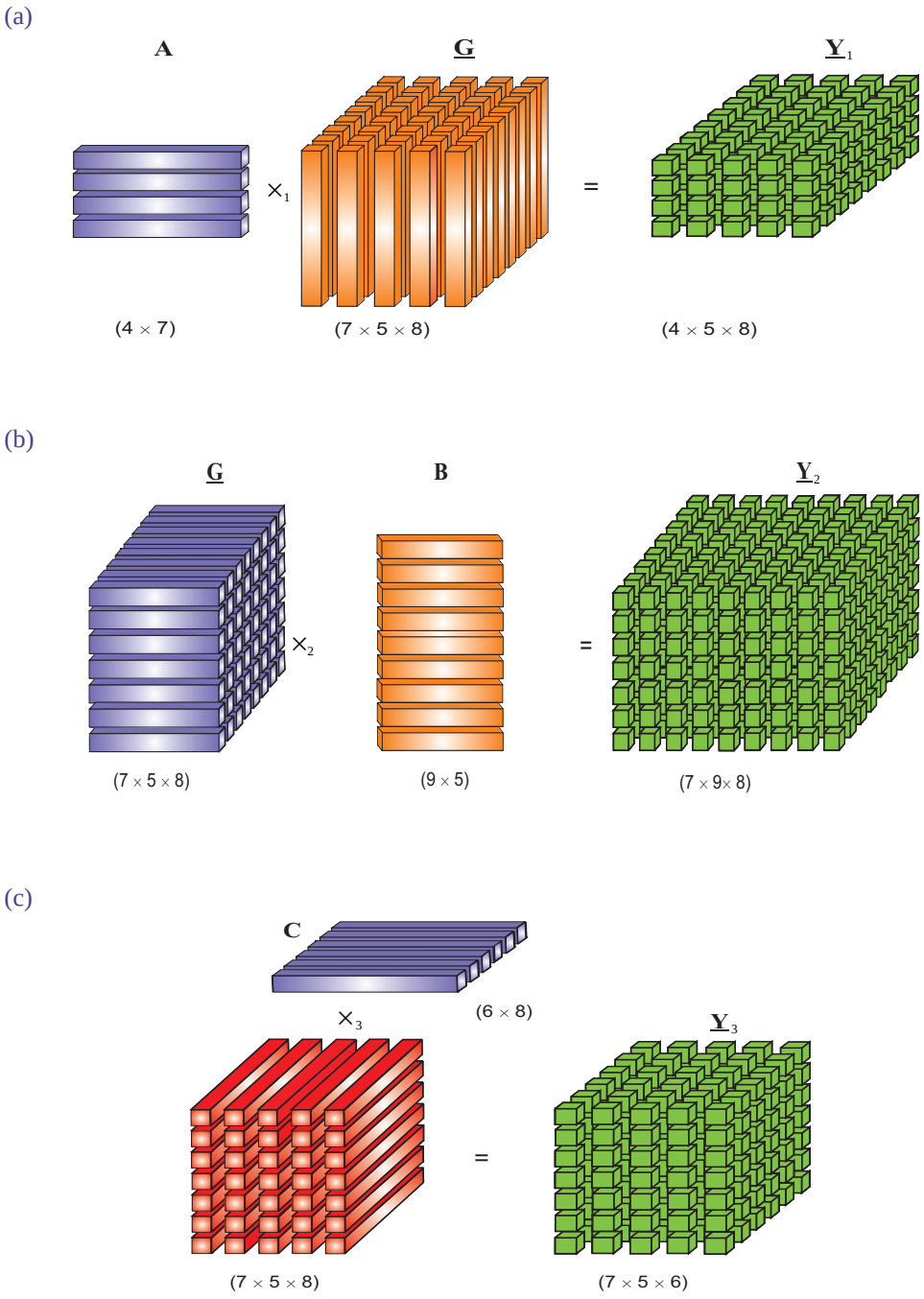


Fig. 1.21 Illustration of the mode-n multiplications of a third-order tensor by matrices. (a) mode-1 multiplication $\underline{\mathbf{Y}}_1 = \underline{\mathbf{G}} \times_1 \mathbf{A}$, (b) mode-2 multiplication $\underline{\mathbf{Y}}_2 = \underline{\mathbf{G}} \times_2 \mathbf{B}$, (c) mode-3 multiplication $\underline{\mathbf{Y}}_3 = \underline{\mathbf{G}} \times_3 \mathbf{C}$.

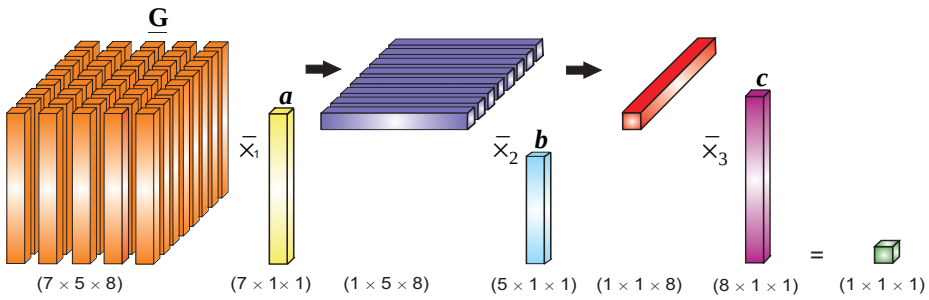


Fig. 1.22 Illustration of mode-n multiplication of a third-order tensor $\underline{\mathbf{G}}$ by vectors, yielding scalar $y = \underline{\mathbf{G}} \,\bar{\times}_1\, \boldsymbol{a} \,\bar{\times}_2\, \boldsymbol{b} \,\bar{\times}_3\, \boldsymbol{c}$. Note that the dimension of the result is reduced by one. For example, multiplying a three-way (a third-order) tensor by a vector in mode-1 results in a 2-way tensor (a matrix).

giving a tensor of dimension $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$. The above notation is adopted from [85].

It is not difficult to verify that these operations satisfy the following properties

$$\left[\underline{\mathbf{G}} \times \{\mathbf{A}\}\right]_{(n)} = \mathbf{A}^{(n)} \mathbf{G}_{(n)} \left[\mathbf{A}^{(N)} \otimes \mathbf{A}^{(N-1)} \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \cdots \otimes \mathbf{A}^{(1)}\right]^T. \tag{1.101}$$

Definition 1.6 (mode-n tensor-vector product) *The mode-n multiplication of a tensor* $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *by a vector* $\boldsymbol{a} \in \mathbb{R}^{I_n}$ *is denoted by*[14]

$$\underline{\mathbf{Y}} \,\bar{\times}_n\, \boldsymbol{a} \tag{1.102}$$

and has dimension $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$, *that is,*

$$\underline{\mathbf{Z}} = \underline{\mathbf{Y}} \,\bar{\times}_n\, \boldsymbol{a} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}, \tag{1.103}$$

Element-wise, we have

$$z_{i_1, i_2, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} y_{i_1, i_2, \ldots, i_N}\, a_{i_n}. \tag{1.104}$$

It is also possible to multiply a tensor by a vector in more than one mode. Multiplying a three-way tensor by vectors in the two modes results in a 1-way tensor (a vector); multiplying it in all modes results in a scalar. We can exchange the order of multiplication by the following rule:

$$\underline{\mathbf{Y}} \,\bar{\times}_m\, \boldsymbol{a} \,\bar{\times}_n\, \boldsymbol{b} = (\underline{\mathbf{Y}} \,\bar{\times}_m\, \boldsymbol{a}) \,\bar{\times}_n\, \boldsymbol{b} = (\underline{\mathbf{Y}} \,\bar{\times}_n\, \boldsymbol{b}) \,\bar{\times}_m\, \boldsymbol{a}, \qquad \text{for } m < n. \tag{1.105}$$

[14] A bar over the operator $\times$ indicates a contracted product.

Table 1.1 Rules for the mode-n multiplication of tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J\times R\times P}$ with matrices $\mathbf{A} \in \mathbb{R}^{I\times J}$, $\mathbf{B} \in \mathbb{R}^{T\times R}$, and $\mathbf{C} \in \mathbb{R}^{Q\times P}$ and with vectors: $\boldsymbol{a} \in \mathbb{R}^{J}$, $\boldsymbol{b} \in \mathbb{R}^{R}$ and $\boldsymbol{c} \in \mathbb{R}^{P}$.

Mode-n product	Matricized version	Vectorized version
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \in \mathbb{R}^{I\times R\times P}$ $y_{irp} = \sum_{j=1}^{J} g_{jrp}\, a_{ij}$	$\mathbf{Y}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}$	$\text{vec}(\mathbf{Y}_{(1)}) = (\mathbf{I} \otimes \mathbf{A})\text{vec}(\mathbf{G}_{(1)})$
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_2 \mathbf{B} \in \mathbb{R}^{J\times T\times P}$ $y_{jtp} = \sum_{r=1}^{R} g_{jrp}\, b_{tr}$	$\mathbf{Y}_{(2)} = \mathbf{B}\mathbf{G}_{(2)}$	$\text{vec}(\mathbf{Y}_{(2)}) = (\mathbf{I} \otimes \mathbf{B})\text{vec}(\mathbf{G}_{(2)})$
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_3 \mathbf{C} \in \mathbb{R}^{J\times R\times Q}$ $y_{jrq} = \sum_{p=1}^{P} g_{jrp}\, c_{qp}$	$\mathbf{Y}_{(3)} = \mathbf{C}\mathbf{G}_{(3)}$	$\text{vec}(\mathbf{Y}_{(3)}) = (\mathbf{I} \otimes \mathbf{C})\text{vec}(\mathbf{G}_{(3)})$
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \,\bar{\times}_1\, \boldsymbol{a} \in \mathbb{R}^{R\times P}$ $y_{rp} = \sum_{j=1}^{J} g_{jrp}\, a_{j}$	$\mathbf{Y}_{(1)} = \boldsymbol{a}^T\mathbf{G}_{(1)}$	$\text{vec}(\mathbf{Y}_{(1)}) = (\mathbf{I} \otimes \boldsymbol{a}^T)\text{vec}(\mathbf{G}_{(1)})$ $\text{vec}(\mathbf{Y}_{(1)}) = \mathbf{G}_{(1)}^T\, \boldsymbol{a}$
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \,\bar{\times}_2\, \boldsymbol{b} \in \mathbb{R}^{J\times P}$ $y_{jp} = \sum_{r=1}^{R} g_{jrp}\, b_{r}$	$\mathbf{Y}_{(2)} = \boldsymbol{b}^T\mathbf{G}_{(2)}$	$\text{vec}(\mathbf{Y}_{(2)}) = (\mathbf{I} \otimes \boldsymbol{b}^T)\text{vec}(\mathbf{G}_{(2)})$ $\text{vec}(\mathbf{Y}_{(2)}) = \mathbf{G}_{(2)}^T\, \boldsymbol{b}$
$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \,\bar{\times}_3\, \boldsymbol{c} \in \mathbb{R}^{J\times R}$ $y_{jp} = \sum_{p=1}^{P} g_{jrp}\, c_{p}$	$\mathbf{Y}_{(3)} = \boldsymbol{c}^T\mathbf{G}_{(3)}$	$\text{vec}(\mathbf{Y}_{(3)}) = (\mathbf{I} \otimes \boldsymbol{c}^T)\text{vec}(\mathbf{G}_{(3)})$ $\text{vec}(\mathbf{Y}_{(3)}) = \mathbf{G}_{(3)}^T\, \boldsymbol{c}$

For example, the mode-n multiplication of a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J\times R\times P}$ by vectors $\boldsymbol{a} \in \mathbb{R}^{J}$, $\boldsymbol{b} \in \mathbb{R}^{R}$ and $\boldsymbol{c} \in \mathbb{R}^{P}$ can be expressed as (see Figure 1.22 and Table 1.1)

$$z = \underline{\mathbf{G}} \,\bar{\times}_1\, \boldsymbol{a} \,\bar{\times}_2\, \boldsymbol{b} \,\bar{\times}_3\, \boldsymbol{c} = \sum_{j=1}^{J}\sum_{r=1}^{R}\sum_{p=1}^{P} g_{jrp}\, a_j\, b_r\, c_p.$$

More generally, for $\underline{\mathbf{G}} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$ and $\boldsymbol{a}^{(n)} \in \mathbb{R}^{J_n}$, the multiplication by all vectors in all modes ($n = 1, 2, \ldots, N$) gives a scalar:

$$y = \underline{\mathbf{G}} \,\bar{\times}_1\, \boldsymbol{a}^{(1)} \,\bar{\times}_2\, \boldsymbol{a}^{(2)} \,\cdots\bar{\times}_N\, \boldsymbol{a}^{(N)} = \underline{\mathbf{G}} \,\bar{\times}\, \{\boldsymbol{a}\} \in \mathbb{R} \tag{1.106}$$

whereas multiplication in every mode except mode-n results in a vector $\boldsymbol{x}$ of length J_n:

$$\begin{aligned}\boldsymbol{x} &= \underline{\mathbf{G}} \,\bar{\times}_1\, \boldsymbol{a}^{(1)} \,\cdots \bar{\times}_{n-1}\, \boldsymbol{a}^{(n-1)} \,\bar{\times}_{n+1}\, \boldsymbol{a}^{(n+1)} \,\cdots\, \bar{\times}_N\, \boldsymbol{a}^{(N)} \\ &= \mathbf{G}_{(n)} \left(\boldsymbol{a}^{(N)} \otimes \cdots \otimes \boldsymbol{a}^{(n+1)} \otimes \boldsymbol{a}^{(n-1)} \otimes \cdots \otimes \boldsymbol{a}^{(1)}\right) = \underline{\mathbf{G}} \,\bar{\times}_{-n}\, \{\boldsymbol{a}\} \in \mathbb{R}^{J_n}. \end{aligned} \tag{1.107}$$

Also note that multiplication in every mode except mode-n and mode-m, results in a matrix of size $J_n \times J_m$.

A matrix $\mathbf{G}\,(I \times J)$ can be considered as a third-order tensor $\underline{\mathbf{G}}$ in which the 3rd dimension is 1 $(I \times J \times 1)$, and its matricized versions in each mode are given by

$$\left[\underline{\mathbf{G}}\right]_{(1)} = \left[\underline{\mathbf{G}}\right]_{(2)}^T = \mathbf{G}, \tag{1.108}$$

$$\left[\underline{\mathbf{G}}\right]_{(3)} = \text{vec}\,(\mathbf{G})^T\,. \tag{1.109}$$

The mode-3 product of the tensor $\underline{\mathbf{G}}$ with a vector $\boldsymbol{a}$ is exactly the outer product of $\mathbf{G}$ and $\boldsymbol{a}$.

$$\underline{\mathbf{G}} \times_3 \boldsymbol{a} = \underline{\mathbf{G}} \circ \boldsymbol{a}. \tag{1.110}$$

Definition 1.7 The scalar product *(or inner product) of two tensors* $\underline{\mathbf{A}}, \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2, \times \cdots \times I_N}$ *of the same order is denoted by* $\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle$ *and is computed as a sum of element-wise products over all the indices, that is,*

$$c = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} b_{i_1,i_2,\ldots,i_N} a_{i_1,i_2,\ldots,i_N} \in \mathbb{R}. \tag{1.111}$$

The scalar product allows us to define the higher-order Frobenius norm of a tensor $\underline{\mathbf{A}}$ as

$$\|\underline{\mathbf{A}}\|_F = \sqrt{\langle \underline{\mathbf{A}}, \underline{\mathbf{A}} \rangle} = \sqrt{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} a_{i_1,i_2,\ldots,i_N}^2}, \tag{1.112}$$

whereas the ℓ_1-norm of a tensor is defined as

$$\|\underline{\mathbf{A}}\|_1 = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} |a_{i_1,i_2,\ldots,i_N}|. \tag{1.113}$$

Definition 1.8 The contracted product *of two tensors* $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_M \times J_1 \times \cdots \times J_N}$ *and* $\underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_M \times K_1 \times \cdots \times K_P}$ *along the first M modes is a tensor of size* $J_1 \times \cdots \times J_N \times K_1 \times \cdots \times K_P$*, given by*

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{1,\ldots,M;1,\ldots,M}(j_1, \ldots, j_N, k_1, \ldots, k_P) = \sum_{i_1=1}^{I_1} \cdots \sum_{i_M=1}^{I_M} a_{i_1,\ldots,i_M,j_1,\ldots,j_N}\; b_{i_1,\ldots,i_M,k_1,\ldots,k_P}. \tag{1.114}$$

The remaining modes are ordered such that those from $\underline{\mathbf{A}}$ come before $\underline{\mathbf{B}}$. The arguments specifying the modes of $\underline{\mathbf{A}}$ and those of $\underline{\mathbf{B}}$ for contraction need not be consecutive. However, the sizes of the corresponding dimensions must be equal. For example, the contracted tensor product along the mode-2 of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{3\times4\times5}$, and the mode-3 of a tensor $\underline{\mathbf{B}} \in \mathbb{R}^{7\times8\times4}$ returns a tensor $\underline{\mathbf{C}} = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{2;3} \in \mathbb{R}^{3\times5\times7\times8}$.

The contracted tensor product of $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$ along the same M modes simplifies to

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{1,\ldots,M;1,\ldots,M} = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{1,\ldots,M}, \tag{1.115}$$

whereas the contracted product of tensors $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ along all modes except the mode-n is denoted as

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{-n} = \mathbf{A}_{(n)} \mathbf{B}_{(n)}^T \in \mathbb{R}^{I_n \times J_n}, \qquad (I_k = J_k, \quad \forall k \neq n). \tag{1.116}$$

The tensor-vector, tensor-matrix and scalar multiplications can be expressed in a form of contracted product. For example, the contracted product along the mode-n of the tensor $\underline{\mathbf{A}}$ and the mode-2 of matrix $\mathbf{C} \in \mathbb{R}^{J \times I_n}$ can be obtained by permuting the dimensions of the mode-n product of $\underline{\mathbf{A}}$ and $\mathbf{C}$

$$\langle \underline{\mathbf{A}}, \mathbf{C} \rangle_{n;2} = \langle \underline{\mathbf{A}}, \mathbf{C}^T \rangle_{n;1} = \texttt{permute}(\underline{\mathbf{A}} \times_n \mathbf{C}, [1, \ldots, n-1, n+1, \ldots, N, n]). \tag{1.117}$$

We also have

$$\langle \mathbf{C}, \underline{\mathbf{A}} \rangle_{2;n} = \langle \mathbf{C}^T, \underline{\mathbf{A}} \rangle_{1;n} = \texttt{permute}(\underline{\mathbf{A}} \times_n \mathbf{C}, [n, 1, \ldots, n-1, n+1, \ldots, N]). \tag{1.118}$$

For two tensors of the same dimension, their contracted product along all their modes is their inner product

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle_{1,\ldots,N} = \langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle. \tag{1.119}$$

In a special case of $M = 0$, the contracted product becomes the outer product of two tensors.

1.4.7 Special Forms of Tensors

Tensors can take special forms or structures. For instance, often a tensor is sparse or symmetric.

1.4.7.1 **Rank-One Tensor** Using the outer product, the rank of tensor can be defined as follows (see Figure 1.23)

Definition 1.9 (Rank-one tensor) *A tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of order N has rank-one if it can be written as an outer product of N vectors i.e.,*

$$\underline{\mathbf{Y}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)}, \tag{1.120}$$

where $\boldsymbol{a}^{(n)} \in \mathbb{R}^{I_n}$ and $y_{i_1,i_2,\ldots,i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \cdots a_{i_N}^{(N)}$. The rank of a tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as the minimal number of rank-one tensors $\underline{\mathbf{Y}}_1, \ldots, \underline{\mathbf{Y}}_R$ such that $\underline{\mathbf{Y}} = \sum_{r=1}^{R} \underline{\mathbf{Y}}_r$.

This outer product is often computed via the Khatri-Rao product or the Kronecker product based on the following relation

$$\text{vec}(\underline{\mathbf{Y}}) = \text{vec}(\mathbf{Y}_{(1)}) = \text{vec}(\boldsymbol{a}^{(1)}(\boldsymbol{a}^{(N)} \odot \cdots \odot \boldsymbol{a}^{(2)})^T) = \boldsymbol{a}^{(N)} \odot \cdots \odot \boldsymbol{a}^{(2)} \odot \boldsymbol{a}^{(1)}. \tag{1.121}$$

Rank-one tensors have many interesting properties and play an important role in multi-way analysis [145], [138], [139], [31], [68], [98], [30], [115]. In general, rank of a higher-order tensor is defined as the minimal number of rank-one tensors whose linear combination yields $\underline{\mathbf{Y}}$. Such a representation of a tensor by a linear combination of rank-one tensors is just a CANonical

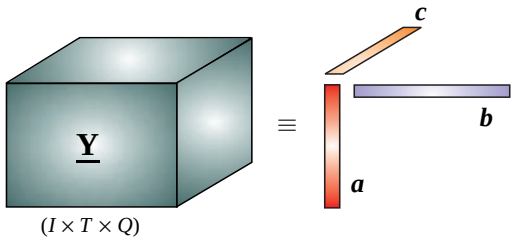


Fig. 1.23 Rank-one third-order tensor: $\underline{\mathbf{Y}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c} \in \mathbb{R}^{I \times T \times Q}$, $\boldsymbol{a} \in \mathbb{R}^{I}$, $\boldsymbol{b} \in \mathbb{R}^{T}$, $\boldsymbol{c} \in \mathbb{R}^{Q}$.

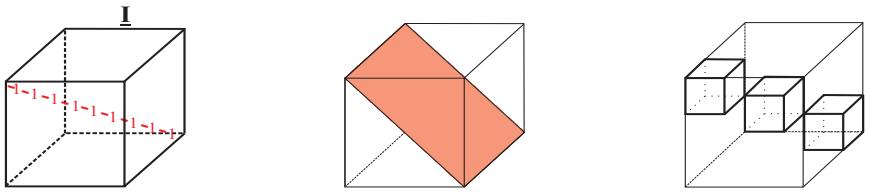


Fig. 1.24 Special forms of third-order tensors: (a) Super-Identity cube tensor $\underline{\mathbf{I}}$, (b) sparse tensor with diagonal frontal slices, which can be mathematically expressed as $\underline{\mathbf{I}} \times_3 \mathbf{C}$, and (c) block diagonal tensor.

DECOMPposition (CANDECOMP) or PARAFAC (PARAllel FACtor decomposition) which preserves the uniqueness under some mild conditions [90].

1.4.7.2 **Symmetric and Super-Symmetric Tensors** For the particular case when all the N vectors $\boldsymbol{a}^{(j)}$ are equal to a vector $\boldsymbol{g}$, their outer product is called a supersymmetric rank-one tensor.[15] A super-symmetric tensor has the same dimension in every mode.

Tensors can also only be (partially) symmetric in two or more modes. For example, a three-way tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times I \times Q}$ is symmetric in modes one and two if all its frontal slices are symmetric, i.e., $\mathbf{Y}_q = \mathbf{Y}_q^T, \forall q = 1, 2, \ldots, Q$.

1.4.7.3 **Diagonal Tensors** An N-th order cubical tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is diagonal if its elements $y_{i_1, i_2, \ldots, i_N} \neq 0$ only if $i_1 = i_2 = \cdots = i_N$ (see Figure 1.24(a)). We use $\underline{\mathbf{I}}$ to denote the cubical identity tensor with ones on the superdiagonal and zeros elsewhere. This concept can be generalized or extended as illustrated in Figures 1.24(b) and 1.24(c).

1.5 TENSOR DECOMPOSITIONS AND FACTORIZATIONS

Many modern applications generate large amounts of data with multiple aspects and high dimensionality for which tensors (i.e., multi-way arrays) provide a natural representation. These

[15] In general, by analogy to symmetric matrices a higher-order tensor is called supersymmetric if its entries are invariant under any permutation of their indices.

include text mining, clustering, Internet traffic, telecommunication records, and large-scale social networks.

Tensor decompositions and factorizations were initiated by Hitchcock in 1927 [74], and later developed by Cattelin in 1944 [85] and by Tucker in 1966 [135], [136]. These concepts and approaches received more attention after Carroll and Chang [24], [25] proposed the Canonical Decomposition (CANDECOMP) and independently Harshman [62], [63], [64] proposed an equivalent model called the PARAFAC (Parallel Factor Analysis) in 1970.

Möck rediscovered the PARAFAC when tackling a neuroscience problem of event related potentials (ERP) in the context of brain imaging [104]. These foundations for tensor factorizations and decompositions also include results on the uniqueness of tensor factorizations and some recommendations on how to choose the number of components. The subsequent contributions put Möck's results in the framework proposed by Harshman [62], Kruskal [89] and Carroll and Chang [24].

Most of the early results devoted to tensor factorizations and decompositions appeared in the psychometrics literature. Appellof, Davidson and Bro are credited as being the first to use tensor decompositions (in 1981-1998) in chemometrics, which have since become extremely popular in that field (see, e.g., [7], [17], [16], [19], [2], [85]). In parallel with the developments in psychometrics and chemometrics, there was a great deal of interest in decompositions of bilinear forms in the field of algebraic complexity [85], [80].

Although some tensor decomposition models have been proposed long time ago, they have recently attracted the interest of researchers working in mathematics, signal processing, data mining, and neuroscience. This probably explains why available mathematical theory seldom deals with the computational and algorithmic aspects of tensor decompositions, together with many still unsolved fundamental problems.

Higher-order tensor decompositions are nowadays frequently used in a variety of fields including psychometrics, chemometrics, image analysis, graph analysis, and signal processing. Two of the most commonly used decompositions are the Tucker decomposition and PARAFAC (also known as CANDECOMP or simply CP) which are often considered (thought of) as higher-order generalizations of the matrix singular value decomposition (SVD) or principal component analysis (PCA). In this book, we superimpose different constraints such as nonnegativity, sparsity or smoothness, and generally such an analogy is no longer valid.

In this chapter we formulate the models and problems for three-way arrays. Extension for arbitrary N-th order tensors will be given in Chapter 7.

1.5.1 Why Multi-way Array Decompositions and Factorizations?

Standard matrix factorizations, such as PCA/SVD, ICA, NMF, and their variants, are invaluable tools for feature selection, dimensionality reduction, noise reduction, and data mining [26]. However, they have only two modes or 2-way representations (say, space and time), and their use is therefore limited. In many applications the data structures often contain higher-order ways (modes) such as trials, task conditions, subjects, and groups together with the intrinsic dimensions of space, time, and frequency. For instance, a sequence of trials may lead to a large stream of data encompassing many dimensions: space, time-frequency, subjects, trials, and conditions [5], [86].

Clearly the "flat-world view" provided by 2-way matrix factorizations (ICA, NMF, SCA) may be insufficient and it is natural to use tensor decomposition approaches. This way all dimensions or modes are retained by virtue of multi-linear models which often produce unique and physically meaningful components. For example, studies in neuroscience often involve multi-

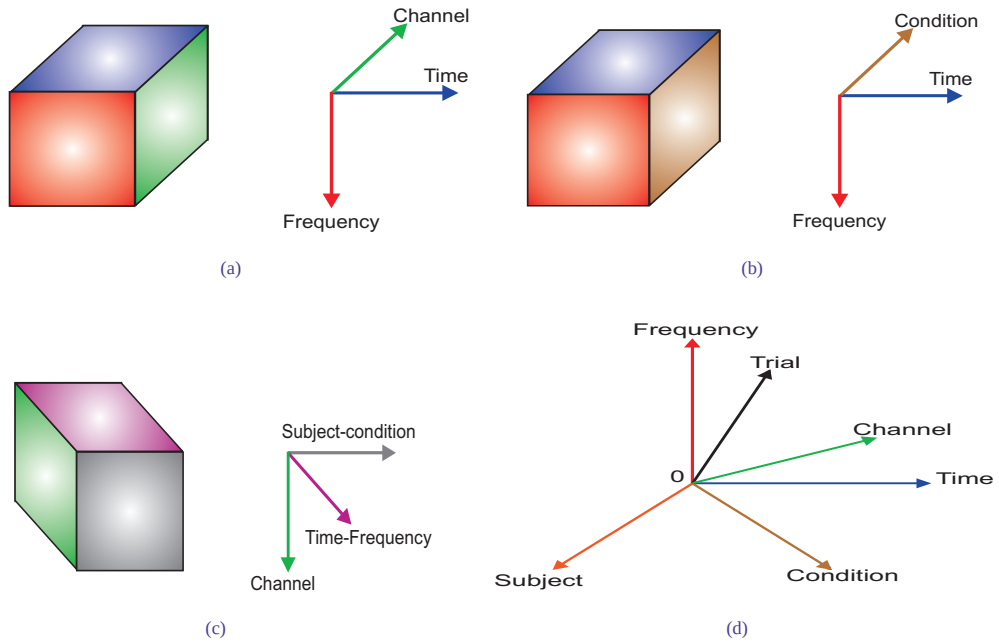


Fig. 1.25 Illustration of various possible arrangements (organization) of three-way and multi-way multichannel EEG/MEG data.

ple subjects (people or animals) and trials leading to experimental data structures conveniently represented by multiway arrays or blocks of three-way data. If the data for every subject were analyzed separately by extracting a matrix or slice from a data block we would lose the covariance information among subjects. To discover hidden components within the data and retain the integrative information, the analysis tools should reflect the multi-dimensional structure of the data.

The multi-way analysis (tensor factorizations and decompositions) is a natural choice, for instance, in EEG studies as it provides convenient multi-channel and multi-subject time-frequency-space sparse representations, artifacts rejection in the time-frequency domain, feature extraction, multi-way clustering and coherence tracking. Our main objective here is to decompose the multichannel time-varying EEG signals into multiple components with distinct modalities in the space, time, and frequency domains in order to identify among them the components common across these different domains, which at the same time are discriminative across different conditions (see Figure 1.25). The two most popular decomposition/factorization models for N-th order tensors are the Tucker model and the more restricted PARAFAC model. Especially, NMF and NTF in conjunction with sparse coding, have recently been given much attention due to their easy interpretation and meaningful representation. NTF has been used in numerous applications in environmental analysis, food studies, pharmaceutical analysis and in chemistry in general (see [18], [2], [85] for review).

As a result of such tensor decompositions, the inherent structures of the recorded brain signals usually become enhanced and better exposed. Further operations performed on these components can remove redundancy and achieve compact sparse representations. There are at least

two possible operations we can perform. First, the extracted factors or hidden latent components can be grouped (clustered) together and represented collectively in a lower dimensional space to extract features and remove redundancy. Second, the components can be simply pruned if they are correlated with a specific mental task. With the addition of extra dimensions it is possible to investigate topography and time and frequency patterns in one analysis. The resulting components can be described not only by the topography and the time-frequency signature but also by the relative contribution from different subjects or conditions. Regarding an application to brain signal analysis, various oscillatory activities within the EEG may overlap, however, the sparse and nonnegative tensor representation by means of the time-frequency-space transformation makes it possible in many cases to isolate each oscillatory behavior well, even when these activities are not well-separated in the space-time (2-way) domain.

Recent development in high spatial density arrays of EEG signals involve multi-dimensional signal processing techniques (referred to as multi-way analysis (MWA), multi-way-array (tensor) factorization/decomposition, dynamic tensor analysis (DTA), or window-based tensor analysis (WTA)). These can be employed to analyze multi-modal and multichannel experimental EEG/MEG and fMRI data [5], [102], [140].

1.5.2 PARAFAC and Nonnegative Tensor Factorization

The PARAFAC[16] can be formulated as follows (see Figures 1.26 and 1.27 for graphical representations).

Given a data tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ and the positive index J, find three-component matrices, also called loading matrices or factors, $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I \times J}$, $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J] \in \mathbb{R}^{T \times J}$ and $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_J] \in \mathbb{R}^{Q \times J}$ which perform the following approximate factorization:

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j + \underline{\mathbf{E}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] + \underline{\mathbf{E}}, \tag{1.122}$$

or equivalently in the element-wise form

$$y_{itq} = \sum_{j=1}^{J} a_{ij} b_{tj} c_{qj} + e_{itq}. \tag{1.123}$$

The symbol $\widehat{\underline{\mathbf{Y}}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a shorthand notation for the PARAFAC factorization, and $\boldsymbol{a}_j = [a_{ij}] \in \mathbb{R}^I$, $\boldsymbol{b}_j = [b_{tj}] \in \mathbb{R}^T$, and $\boldsymbol{c}_j = [c_{qj}] \in \mathbb{R}^Q$ are respectively the constituent vectors of the corresponding factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

The PARAFAC algorithms decompose a given tensor into a sum of multi-linear terms (in this case tri-linear), in a way analogous to the bilinear matrix decomposition. As discussed before, unlike SVD, PARAFAC usually does not impose any orthogonality constraints. A model which imposes nonnegativity on factor matrices is called the NTF (Nonnegative Tensor Factorization) or Nonnegative PARAFAC. A nonnegative version of PARAFAC was first introduced by Carroll *et al.* [25]. Later, more efficient approaches were developed by Bro [16], [6], [80], based on the modified NLS and Paatero [112], [111] who generalized his earlier 2-way positive matrix

[16] Also called the CANDECOMP (Canonical Decomposition) or CP decomposition or simply CPD.

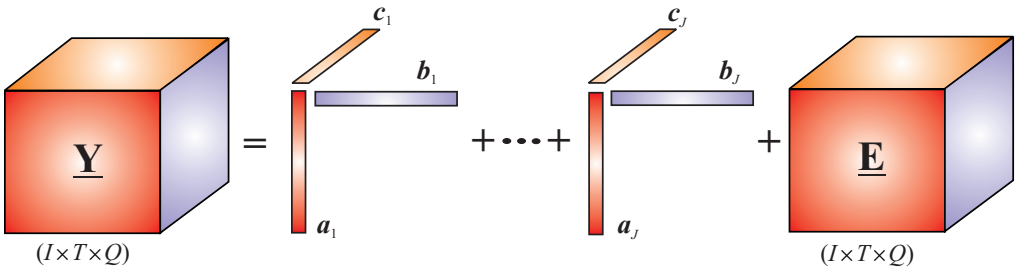


Fig. 1.26 A graphical representation of the third-order PARAFAC as a sum of rank-one tensors. All the vectors $\{\boldsymbol{a}_j, \boldsymbol{b}_j, \boldsymbol{c}_j\}$ are treated as column vectors of factor matrices and are linked for each index j via the outer product operator, that is, $\underline{\mathbf{Y}} = \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j + \mathbf{E}$ or equivalently in a compact form $\underline{\mathbf{Y}} = \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \mathbf{E}$. (In this model not all vectors are normalized to unit length).

factorization (PMF) method to the three-way PARAFAC model, referring to the result as PMF3 (three-way positive matrix factorization). Although such constrained nonnegativity based model may not match perfectly the input data (i.e., it may have larger residual errors $\mathbf{E}$ than the standard PARAFAC without any constraints) such decompositions are often very meaningful and have physical interpretation [30], [115], [116].

It is often convenient to assume that all vectors have unit length so that we can use the modified Harshman's PARAFAC model given by [62], [63]

$$\boxed{\underline{\mathbf{Y}} = \sum_{j=1}^{J} \lambda_j \, \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j + \underline{\mathbf{E}} \cong [\![\boldsymbol{\lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}]\!],} \tag{1.124}$$

or in equivalent element-wise form

$$\boxed{y_{itq} = \sum_{j=1}^{J} \lambda_j \, a_{ij} \, b_{tj} \, c_{qj} + e_{itq},} \tag{1.125}$$

where λ_j are scaling factors and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_J]^T$. Figure 1.28 illustrates the above model and its alternative equivalent representations. The basic PARAFAC model can be represented in compact matrix forms upon applying unfolding representations of the tensor $\underline{\mathbf{Y}}$:

$$\boxed{\begin{aligned} \mathbf{Y}_{(1)} &\cong \mathbf{A}\,\boldsymbol{\Lambda}\,(\mathbf{C} \odot \mathbf{B})^T, \qquad (1.126)\\ \mathbf{Y}_{(2)} &\cong \mathbf{B}\,\boldsymbol{\Lambda}\,(\mathbf{C} \odot \mathbf{A})^T, \qquad (1.127)\\ \mathbf{Y}_{(3)} &\cong \mathbf{C}\,\boldsymbol{\Lambda}\,(\mathbf{B} \odot \mathbf{A})^T, \qquad (1.128) \end{aligned}}$$

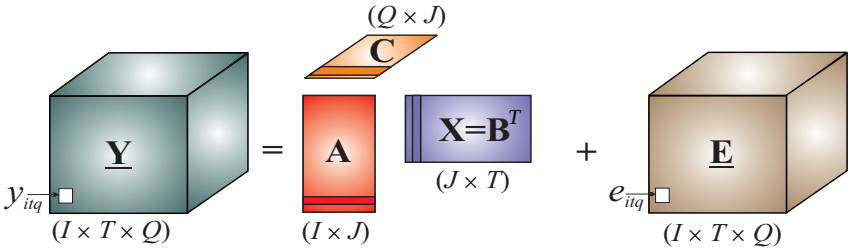
where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ and $\odot$ means the Khatri-Rao product.

Using the mode-n multiplication of a tensor by a matrix, we have

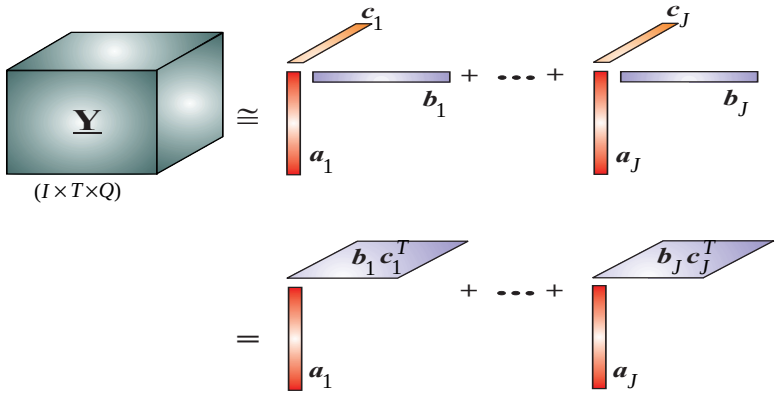
$$\boxed{\underline{\mathbf{Y}} = \underline{\boldsymbol{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}},} \tag{1.129}$$

where $\underline{\boldsymbol{\Lambda}} \in \mathbb{R}^{J \times J \times J}$ is diagonal cubical tensor with nonzero elements λ_j on the superdiagonal. In other words, within Harshman's model for the core tensor all but the superdiagonal elements

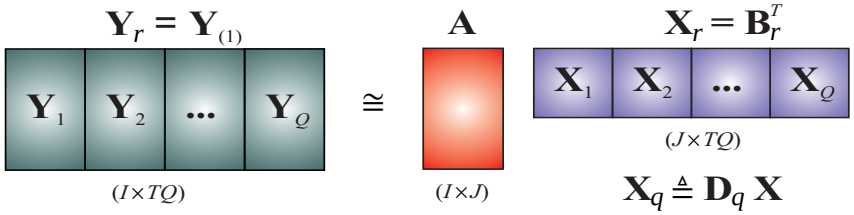
(a)



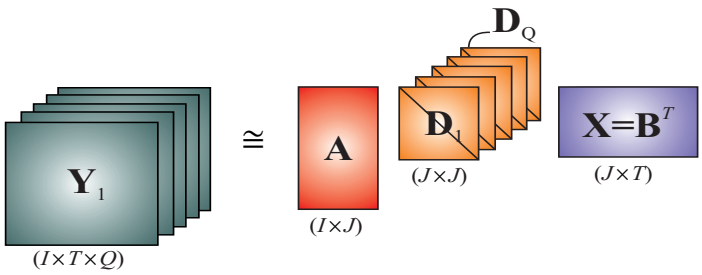
(b)



(c)



(d)



$$\mathbf{Y}_q = \mathbf{A}\mathbf{D}_q\mathbf{X}, \quad (q = 1, 2, \ldots, Q)$$

Fig. 1.27 The alternative representations of the third-order PARAFAC model: (a) as a set of three matrices using a scalar representation (see Eq. (1.123)), (b) as a set of vectors using summation of rank-one tensors expressed by the outer products of the vectors (see Eq. (1.122)), (c) decomposition into two matrices using row-wise unfolding and (d) representation by frontal slices (see Eq. (1.134)) The tensor $\underline{\mathbf{D}} \in \mathbb{R}^{J\times J\times Q}$ has diagonal frontal slices $\mathbf{D}_q \in \mathbb{R}^{J\times J}$, so we can write $\underline{\mathbf{Y}} \approx \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B}$.

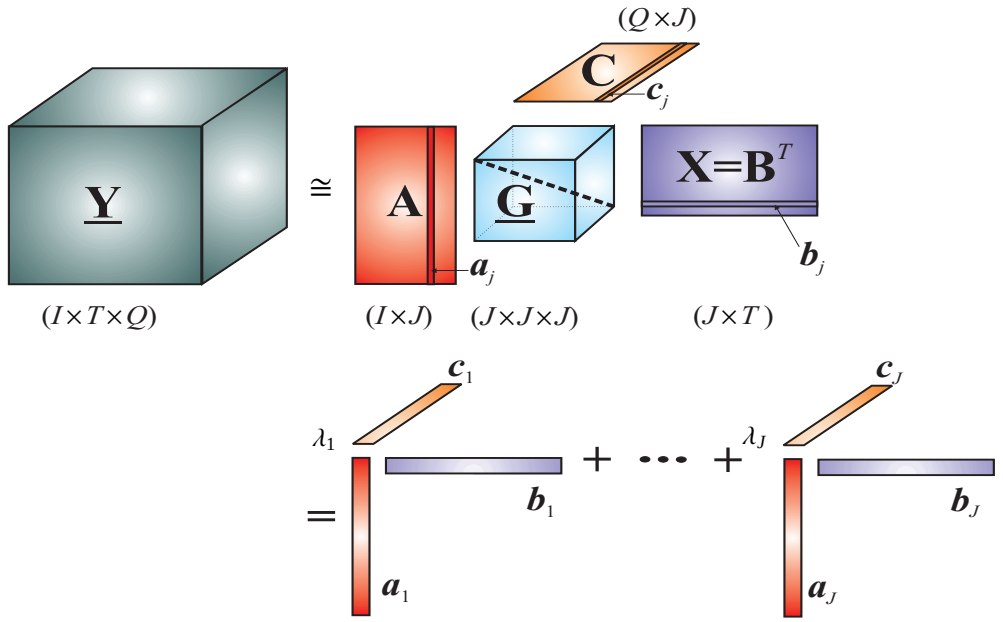


Fig. 1.28 Harshman's PARAFAC model with a superdiagonal core tensor $\underline{\mathbf{G}} = \underline{\mathbf{\Lambda}} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_J) \in \mathbb{R}^{J\times J\times J}$ for the third-order tensor $\underline{\mathbf{Y}} \cong \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{j=1}^{J} \lambda_j \, \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j$. (In this model all vectors are normalized to unit length).

vanish (see Figure 1.28). This also means that PARAFAC can be considered as a special case of the Tucker3 model in which the core tensor is a cubical superdiagonal or super-identity tensor, i.e., $\underline{\mathbf{G}} = \underline{\mathbf{\Lambda}} \in \mathbb{R}^{J\times J\times J}$ with $g_{jjj} \neq 0$.

Another form of the PARAFAC model is the vectorized form given by

$$\boxed{\text{vec}\,(\underline{\mathbf{Y}}) \cong (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\boldsymbol{\lambda}.} \tag{1.130}$$

The three-way PARAFAC model can be also described by using frontal, lateral and horizontal slices as follows[17]

$$\mathbf{Y}_{::q} \cong \mathbf{A}\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{B}^T, \tag{1.131}$$
$$\mathbf{Y}_{:t:} \cong \mathbf{A}\,\mathbf{D}_t(\boldsymbol{b}_{t:})\,\mathbf{C}^T, \tag{1.132}$$
$$\mathbf{Y}_{i::} \cong \mathbf{B}\,\mathbf{D}_i(\boldsymbol{a}_{i:})\,\mathbf{C}^T, \tag{1.133}$$

where $\mathbf{D}_i(\boldsymbol{a}_{i:})$, $\mathbf{D}_t(\boldsymbol{b}_{t:})$ and $\mathbf{D}_q(\boldsymbol{c}_{q:})$ are diagonal matrices which take the i-th, t-th and q-th row of the matrices $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, respectively, and produce diagonal matrices by placing the corresponding row on the main diagonal.

[17] Such a representation does not exist for higher-order tensors where $N > 3$.

Table 1.2 Mathematical formulations of the standard PARAFAC model for a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times T\times Q}$ with factor matrices: $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I\times J}$, $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J] \in \mathbb{R}^{T\times J}$ and $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_J] \in \mathbb{R}^{Q\times J}$. There are two optional and equivalent representations: one in which we have a linear combination of rank-one tensors with unit length vectors and a second in which the vectors $\boldsymbol{a}_j \in \mathbb{R}^I$ and $\boldsymbol{b}_j \in \mathbb{R}^T$ have a unit ℓ_2-norm and the scaling factors λ_j are absorbed by the non-normalized vectors $\boldsymbol{c}_j \in \mathbb{R}^Q$.

Operator	Formulation
Outer products	$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j + \underline{\mathbf{E}}, \quad (\text{s.t. } \|\boldsymbol{a}_j\|_2 = \|\boldsymbol{b}_j\|_2 = 1, \ \forall j)$ $\underline{\mathbf{Y}} = \sum_{j=1}^{J} \lambda_j \, \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j + \underline{\mathbf{E}}, \quad (\text{s.t. } \|\boldsymbol{a}_j\|_2 = \|\boldsymbol{b}_j\|_2 = \|\boldsymbol{c}_j\|_2 = 1, \ \forall j)$
Scalar	$y_{itq} = \sum_{j=1}^{J} a_{ij} \, b_{tj} \, c_{qj} + e_{itq}$ $y_{itq} = \sum_{j=1}^{J} \lambda_j \, a_{ij} \, b_{tj} \, c_{qj} + e_{itq}$
Mode-n multiplications	$\underline{\mathbf{Y}} = \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}}$ $\underline{\mathbf{Y}} = \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}}$
Slice representations	$\mathbf{Y}_{::q} = \mathbf{A} \ \text{diag}(\boldsymbol{c}_{q:}) \ \mathbf{B}^T + \mathbf{E}_{::q} = \mathbf{A} \ \mathbf{D}_q \ (\boldsymbol{c}_{q:}) \ \mathbf{B}^T + \mathbf{E}_{::q}$ $\mathbf{Y}_{i::} = \mathbf{B} \ \text{diag}(\boldsymbol{a}_{i:}) \ \mathbf{C}^T + \mathbf{E}_{i::} = \mathbf{B} \ \mathbf{D}_i \ (\boldsymbol{a}_{i:}) \ \mathbf{C}^T + \mathbf{E}_{i::}$ $\mathbf{Y}_{:t:} = \mathbf{A} \ \text{diag}(\boldsymbol{b}_{t:}) \ \mathbf{C}^T + \mathbf{E}_{:t:} = \mathbf{A} \ \mathbf{D}_t \ (\boldsymbol{b}_{t:}) \ \mathbf{C}^T + \mathbf{E}_{:t:}$
Vectors	$\text{vec}(\underline{\mathbf{Y}}) = \text{vec}(\mathbf{Y}_{(1)}) = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A}) \ \boldsymbol{\lambda} + \text{vec}(\mathbf{E}_{(1)})$
Kronecker products	$\mathbf{Y}_{(1)} = \mathbf{A} \ \mathbf{D}_{(1)}(\boldsymbol{c}_q) \ (\mathbf{I} \otimes \mathbf{B})^T + \mathbf{E}_{(1)}$
Khatri-Rao products	$\mathbf{Y}_{(1)} = \mathbf{A} \ (\mathbf{C} \odot \mathbf{B})^T + \mathbf{E}_{(1)}; \quad \mathbf{Y}_{(1)} = \mathbf{A} \ \mathbf{\Lambda}(\mathbf{C} \odot \mathbf{B})^T + \mathbf{E}_{(1)}$ $\mathbf{Y}_{(2)} = \mathbf{B} \ (\mathbf{C} \odot \mathbf{A})^T + \mathbf{E}_{(2)}; \quad \mathbf{Y}_{(2)} = \mathbf{B} \ \mathbf{\Lambda}(\mathbf{C} \odot \mathbf{A})^T + \mathbf{E}_{(2)}$ $\mathbf{Y}_{(3)} = \mathbf{C} \ (\mathbf{B} \odot \mathbf{A})^T + \mathbf{E}_{(3)}; \quad \mathbf{Y}_{(3)} = \mathbf{C} \ \mathbf{\Lambda}(\mathbf{B} \odot \mathbf{A})^T + \mathbf{E}_{(3)}.$

In particular, it is convenient to represent the three-way PARAFAC model in terms of the frontal slices, as $\mathbf{Y}_q = \mathbf{Y}_{::q}$ of $\underline{\mathbf{Y}}$

$$\boxed{\mathbf{Y}_q \cong \mathbf{A}\mathbf{D}_q\mathbf{B}^T,} \tag{1.134}$$

where a matrix $\mathbf{D}_q := \mathbf{D}_q(\boldsymbol{c}_{q:})$ is the diagonal matrix based on the q-th row of $\mathbf{C}$.

The above representation of the PARAFAC model has striking similarity to the Approximative Joint Diagonalization method, thus having a clear interpretation for the BSS problem, where the matrix $\mathbf{A}$ represents a mixing matrix, $\mathbf{X} = \mathbf{B}^T$ represents unknown sources, and $\mathbf{C}$ represents

a scaling matrix [45], [44]. In fact, the PARAFAC factorization can be reformulated as simultaneous diagonalization of a set of matrices, which often leads to fast and reliable way to compute this factorization.

The PARAFAC model has some severe limitations as it represents observed data by common factors utilizing the same number of components (columns). In other words, we do not have enough degrees of freedom as compared to other models. Moreover, the PARAFAC approximation may be ill-posed and may lead to unstable estimation of its components. The next sections discuss more flexible models.

1.5.2.1 **Basic Approaches to Solve NTF Problem** In order to compute the nonnegative component matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ we usually apply constrained optimization approach as by minimizing a suitable design cost function. Typically, we minimize (with respect the component matrices) the following global cost function

$$D_F(\underline{\mathbf{Y}} \,||\, [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) = ||\underline{\mathbf{Y}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]||_F^2 + \alpha_{\mathbf{A}} \, ||\mathbf{A}||_F^2 + \alpha_{\mathbf{B}} \, ||\mathbf{B}||_F^2 + \alpha_{\mathbf{C}} \, ||\mathbf{C}||_F^2, \tag{1.135}$$

subject to nonnegativity constraints, where $\alpha_{\mathbf{A}}, \alpha_{\mathbf{B}}, \alpha_{\mathbf{C}}$ are nonnegative regularization parameters.

There are at least three different approaches for solving this optimization problem. The first approach is to use a vectorized form of the above cost function in the form $J(\boldsymbol{x}) = \text{vec}(\underline{\mathbf{Y}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) = \mathbf{0}$ and employ the Nonnegative Least Squares (NLS) approach. Such a method was first applied for NTF by Paatero [111] and also Tomasi and Bro [133], [134]. The Jacobian of such function can be of large size $ITQJ \times (I + T + Q)$, yielding very high computation cost.

In the second approach, Acar, Kolda and Dunlavy propose to optimize the cost function simultaneously with respect to all variables using a modern nonlinear conjugate gradient optimization technique [1]. However, such a cost function is generally not convex and is not guaranteed to obtain the optimal solution although results are very promising.

The most popular approach is to apply the ALS technique (see Chapter 4 for more detail). In this approach we compute the gradient of the cost function with respect to each individual component matrix (assuming that the others are fixed and independent):

$$\nabla_{\mathbf{A}} D_F = -\mathbf{Y}_{(1)} \, (\mathbf{C} \odot \mathbf{B}) + \mathbf{A} \, [(\mathbf{C}^T \mathbf{C}) \circledast (\mathbf{B}^T \mathbf{B}) + \alpha_{\mathbf{A}} \, \mathbf{I}], \tag{1.136}$$

$$\nabla_{\mathbf{B}} D_F = -\mathbf{Y}_{(2)} \, (\mathbf{C} \odot \mathbf{A}) + \mathbf{B} \, [(\mathbf{C}^T \mathbf{C}) \circledast (\mathbf{A}^T \mathbf{A}) + \alpha_{\mathbf{B}} \, \mathbf{I}], \tag{1.137}$$

$$\nabla_{\mathbf{C}} D_F = -\mathbf{Y}_{(3)} \, (\mathbf{B} \odot \mathbf{A}) + \mathbf{C} \, [(\mathbf{B}^T \mathbf{B}) \circledast (\mathbf{A}^T \mathbf{A}) + \alpha_{\mathbf{C}} \, \mathbf{I}]. \tag{1.138}$$

By equating the gradient components to zero and applying the nonlinear projection to maintain nonnegativity of components we obtain efficient and relatively simple nonnegative ALS update rules for the NTF:

$$\mathbf{A} \leftarrow \left[\mathbf{Y}_{(1)} \, (\mathbf{C} \odot \mathbf{B}) \, [(\mathbf{C}^T \mathbf{C}) \circledast (\mathbf{B}^T \mathbf{B}) + \alpha_{\mathbf{A}} \, \mathbf{I}]^{-1}\right]_+, \tag{1.139}$$

$$\mathbf{B} \leftarrow \left[\mathbf{Y}_{(2)} \, (\mathbf{C} \odot \mathbf{A}) \, [(\mathbf{C}^T \mathbf{C}) \circledast (\mathbf{A}^T \mathbf{A}) + \alpha_{\mathbf{B}} \, \mathbf{I}]^{-1}\right]_+, \tag{1.140}$$

$$\mathbf{C} \leftarrow \left[\mathbf{Y}_{(3)} \, (\mathbf{B} \odot \mathbf{A}) \, [(\mathbf{B}^T \mathbf{B}) \circledast (\mathbf{A}^T \mathbf{A}) + \alpha_{\mathbf{C}} \, \mathbf{I}]^{-1}\right]_+. \tag{1.141}$$

In Chapter 4 and Chapter 6 we prove that the ALS algorithms are special cases of a quasi-Newton method that implicitly employ information about the gradient and Hessian of a cost function. The main advantage of ALS algorithms is high convergence speed and its scalability for large-scale problems.

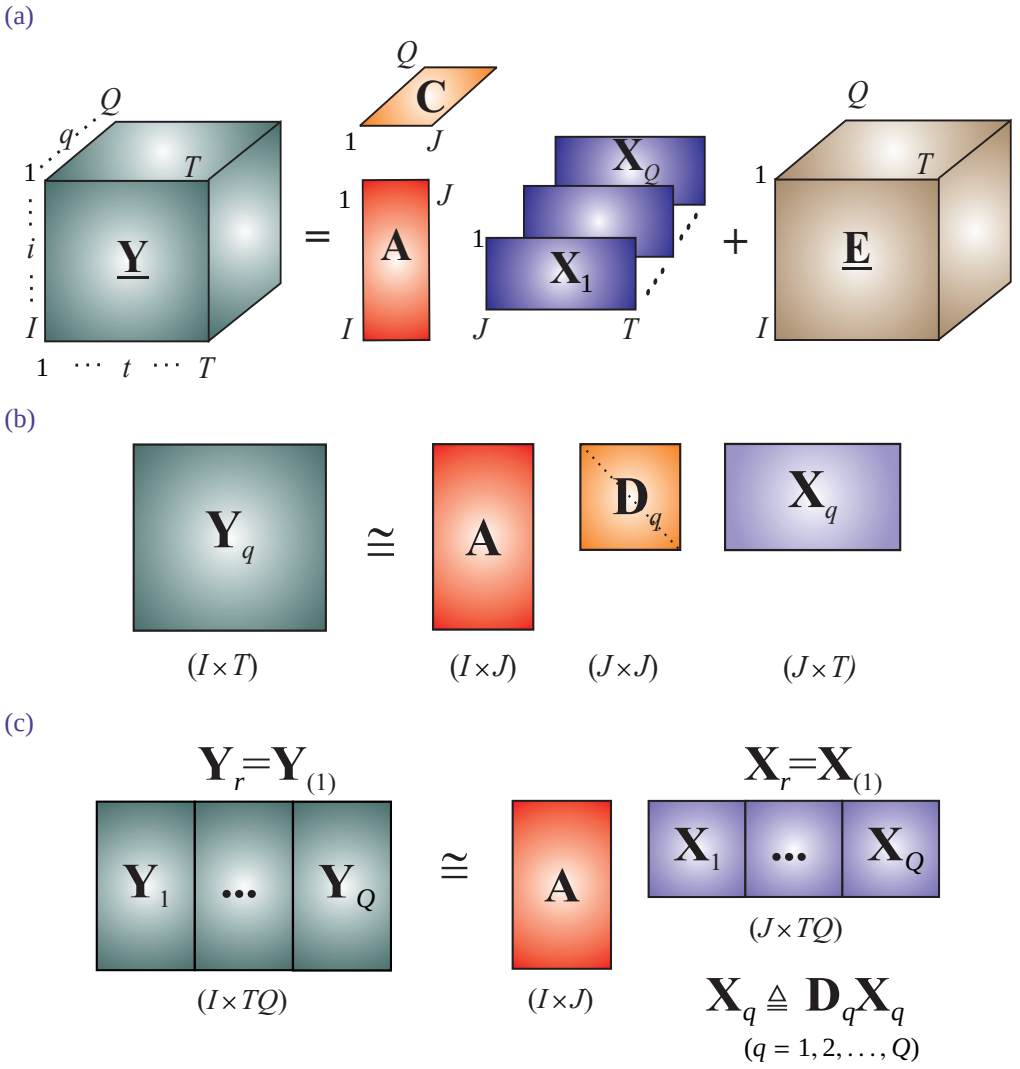


Fig. 1.29 (a) NTF1 model that (approximately) decomposes tensor $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I \times T \times Q}$ into a set of nonnegative matrices $\mathbf{A} = [a_{ij}] \in \mathbb{R}_+^{I \times J}, \mathbf{C} \in \mathbb{R}_+^{Q \times J}$ and $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q\}$, $\mathbf{X}_q = [x_{jtq}] \in \mathbb{R}_+^{J \times T}$, and $\underline{\mathbf{E}} \in \mathbb{R}^{I \times T \times Q}$ is a tensor representing errors. Matrices $\mathbf{X}_q$ are frontal slices of the tensor $\underline{\mathbf{X}} \in \mathbb{R}^{J \times T \times Q}$, typically with $J << I$. (b) Equivalent representation using joint diagonalization of frontal slices, where $\mathbf{D}_q = \text{diag}(\boldsymbol{c}_q)$ are diagonal matrices. (c) Global matrix representation using row-wise unfolding of the tensor; the sub-matrices are defined as $\mathbf{X}_q \triangleq \mathbf{D}_q \mathbf{X}_q$, $(q = 1, 2, \ldots, Q)$.

1.5.3 NTF1 Model

Figure 1.29 illustrates the basic 3D NTF1 model, which is an extension of the NTF model [34]. A given data (observed) tensor $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I \times T \times Q}$ is decomposed into a set of matrices $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{C} \in \mathbb{R}_+^{Q \times J}$, as well as a third-order tensor with reduced dimension ($J < I$), for which the frontal

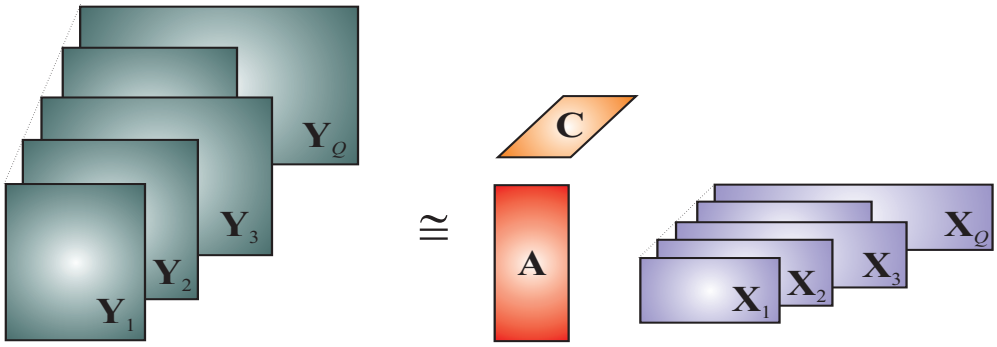


Fig. 1.30 Extended NTF1 model for a three-way array. The goal is to estimate the set of non-negative matrices $\mathbf{A}, \mathbf{C}$ and $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q\}$.

slices $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_Q\}$ have nonnegative entries. This three-way NTF1 model is given by

$$\mathbf{Y}_q = \mathbf{A}\mathbf{D}_q\mathbf{X}_q + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q), \tag{1.142}$$

where $\mathbf{Y}_q = \mathbf{Y}_{::q} \in \mathbb{R}_+^{I\times T}$ are the frontal slices of $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I\times T\times Q}$, Q is the number of the frontal slices, $\mathbf{A} = [a_{ij}] \in \mathbb{R}_+^{I\times J}$ is the basis (mixing matrix) representing common factors, $\mathbf{D}_q \in \mathbb{R}_+^{J\times J}$ is a diagonal matrix that holds the q-th row of matrix $\mathbf{C} \in \mathbb{R}_+^{Q\times J}$ in its main diagonal, $\mathbf{X}_q = [x_{jtq}] \in \mathbb{R}_+^{J\times T}$ is matrix representing the sources (or hidden components), and $\mathbf{E}_q = \mathbf{E}_{::q} \in \mathbb{R}^{I\times T}$ is the q-th vertical slice of the tensor $\underline{\mathbf{E}} \in \mathbb{R}^{I\times T\times Q}$ representing the errors or noise depending on the application. Typically, for BSS problems $T >> I \geq Q > J$.

We wish to estimate the set of matrices $\mathbf{A}$, $\mathbf{C}$, and $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q\}$ subject to nonnegativity constraints (and other constraints such as sparseness and/or smoothness), given only the observed data $\underline{\mathbf{Y}}$. Since the diagonal matrices $\mathbf{D}_q$ are scaling matrices, they can be absorbed into the matrices $\mathbf{X}_q$ by introducing the row-normalized matrices $\mathbf{X}_q := \mathbf{D}_q\mathbf{X}_q$, thus giving $\mathbf{Y}_q = \mathbf{A}\mathbf{X}_q + \mathbf{E}_q$. Therefore, in the multi-way BSS applications only the matrix $\mathbf{A}$ and the set of scaled source matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q$ need be estimated.

For applications where the observed data are incomplete or have different dimensions for each frontal slice (as shown in Figure 1.30) the model can be described as

$$\mathbf{Y}_q = \mathbf{A}\mathbf{D}_q\mathbf{X}_q + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q), \tag{1.143}$$

where $\mathbf{Y}_q \in \mathbb{R}_+^{I\times T_q}$ are the frontal slices of the irregular tree-dimensional array, $\mathbf{X}_q = [x_{jtq}] \in \mathbb{R}_+^{J\times T_q}$ are matrices representing sources (or hidden components), and $\mathbf{E}_q = \mathbf{E}_{::q} \in \mathbb{R}^{I\times T}$ is the q-th vertical slice of the multi-way array comprising errors.

1.5.4 NTF2 Model

The dual model to the NTF1 is referred to as the 3D NTF2 (by analogy to the PARAFAC2 model[18] [78], [17], [105], [126], [88], see Figure 1.31).

A given tensor $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I \times T \times Q}$ is decomposed into a set of matrices $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_Q\}$, $\mathbf{X} = \mathbf{B}^T$ and $\mathbf{C}$ with nonnegative entries, by the three-way NTF2 model as

$$\mathbf{Y}_q = \mathbf{A}_q \mathbf{D}_q \mathbf{X} + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q), \tag{1.144}$$

where $\mathbf{Y}_q = \mathbf{Y}_{::q} = [y_{itq}] \in \mathbb{R}_+^{I \times T}$ are the frontal slices of $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I \times T \times Q}$, Q is the number of frontal slices, $\mathbf{A}_q = [a_{ijq}] \in \mathbb{R}_+^{I \times J}$ are the basis (mixing) matrices, $\mathbf{D}_q \in \mathbb{R}_+^{J \times J}$ is a diagonal matrix that holds the q-th row of $\mathbf{C} \in \mathbb{R}_+^{Q \times J}$ in its main diagonal, $\mathbf{X} = [x_{jt}] \in \mathbb{R}_+^{J \times T}$ is a matrix representing latent sources (or hidden components or common factors), and $\mathbf{E}_q = \mathbf{E}_{::q} \in \mathbb{R}^{I \times T}$ is the q-th frontal slice of a tensor $\underline{\mathbf{E}} \in \mathbb{R}^{I \times T \times Q}$ comprising error or noise depending on the application. The goal is to estimate the set of matrices $\{\mathbf{A}_q\}$, $(q = 1, 2, \ldots, Q)$, $\mathbf{C}$ and $\mathbf{X}$, subject to some nonnegativity constraints and other possible natural constraints such as sparseness and/or smoothness. Since the diagonal matrices $\mathbf{D}_q$ are scaling matrices they can be absorbed into the matrices $\mathbf{A}_q$ by introducing column-normalization, that is, $\mathbf{A}_q := \mathbf{A}_q \mathbf{D}_q$. In BSS applications, therefore, only the matrix $\mathbf{X}$ and the set of scaled matrices $\mathbf{A}_1, \ldots, \mathbf{A}_Q$ need be estimated. This, however, comes at a price, as we may lose the uniqueness of the NTF2 representation ignoring the scaling and permutation ambiguities. The uniqueness can still be preserved by imposing nonnegativity and sparsity constraints.

The NTF2 model is similar to the well-known PARAFAC2 model[19] with nonnegativity constraints and to the Tucker models described in the next section [105], [126], [88]. In a special case, when all matrices $\mathbf{A}_q$ are identical, the NTF2 model can be simplified into the ordinary PARAFAC model (see Section 1.5.2) with the nonnegativity constraints described by

$$\mathbf{Y}_q = \mathbf{A} \mathbf{D}_q \mathbf{X} + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q). \tag{1.145}$$

As shown in Figure 1.32(a) the NTF2 model can be extended to the decomposition of multi-way arrays with different dimensions using the simultaneous factorizations

$$\mathbf{Y}_q = \mathbf{A}_q \mathbf{D}_q \mathbf{X} + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q), \tag{1.146}$$

where $\mathbf{Y} \in \mathbb{R}_+^{I_q \times T}$, $\mathbf{X} \in \mathbb{R}_+^{J \times T}$, $\mathbf{C} \in \mathbb{R}_+^{Q \times J}$, $\mathbf{A}_q \in \mathbb{R}_+^{I_q \times J}\}$, $\mathbf{E}_q = \mathbf{E}_{::q} \in \mathbb{R}^{I_q \times T}$ is the q-th frontal slice of a three-way array (of the same dimensions as the data array) and $\mathbf{D}_q \in \mathbb{R}_+^{J \times J}$ is a diagonal matrix that holds the q-th row of $\mathbf{C}$ in its main diagonal. Using the transformation $\mathbf{X}_q := \mathbf{D}_q \mathbf{X}_q$, we can convert the NTF2 problem to the standard (2-way) NMF problem:

$$\bar{\mathbf{Y}} = \bar{\mathbf{A}} \mathbf{X} + \bar{\mathbf{E}}, \tag{1.147}$$

[18] In fact the NTF2 model be can obtained form NTF1 model via simple permutation of tensors and matrices. However, since the frontal slices $\mathbf{A}_q$ and $\mathbf{X}_q$ of the core tensors have different physical interpretations we discuss these models separately.

[19] In the PARAFAC2 model we usually assume that $\mathbf{A}_q^T \mathbf{A}_q = \mathbf{\Phi} \in \mathbb{R}^{J \times J}$, $\forall q$ (i.e., it is required that the matrix product $\mathbf{A}_q$ with its transpose is invariant for all frontal slices of a core three-way tensor $\underline{\mathbf{A}}$).

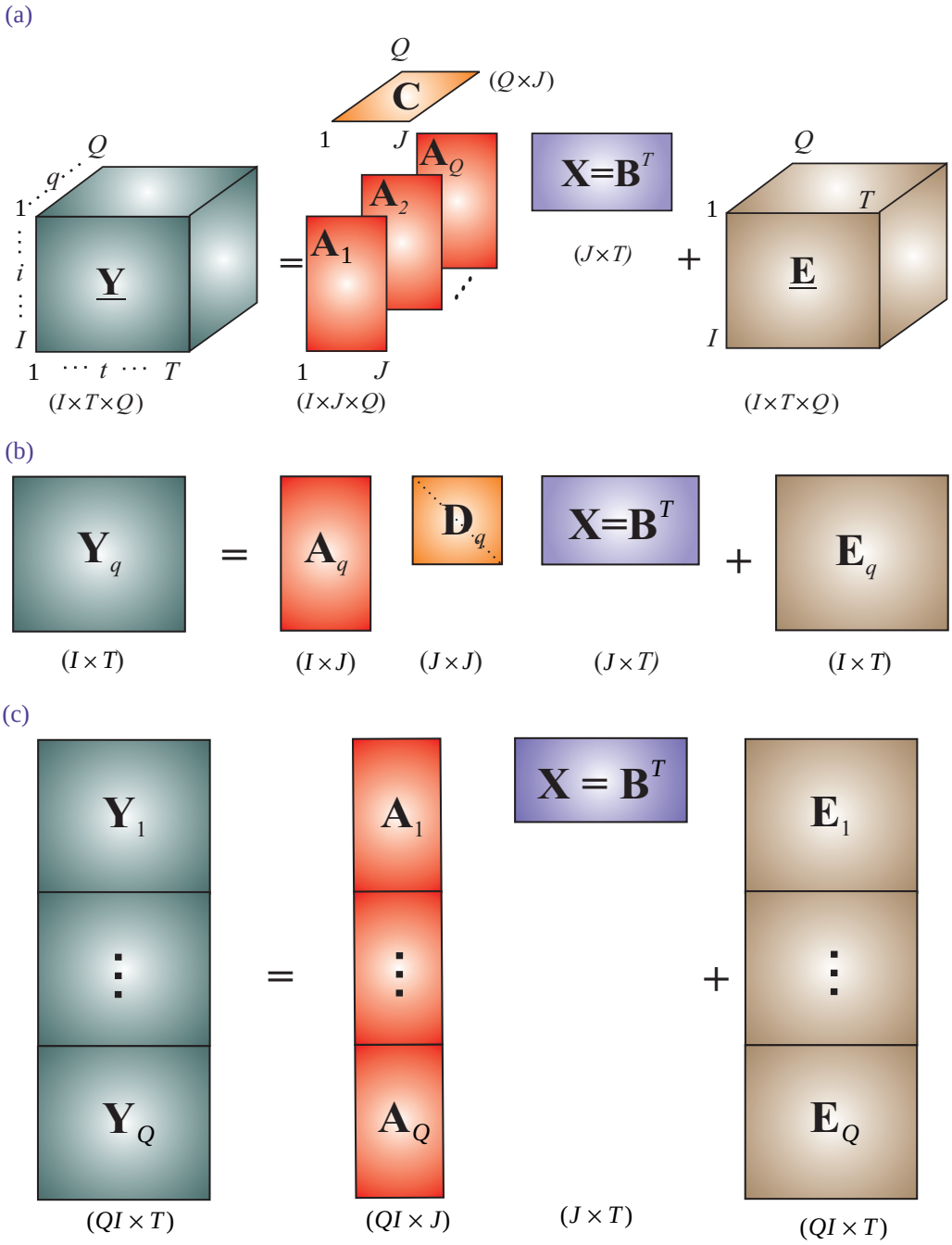


Fig. 1.31 (a) NTF2 model in which a third-order tensor is decomposed into a set of nonnegative matrices: $\{\mathbf{A}_1, \ldots, \mathbf{A}_Q\}$, $\mathbf{C}$, and $\mathbf{X}$. (b) Equivalent representation in which the frontal slices of a tensor are factorized by a set of nonnegative matrices. (c) Global matrix representation using column-wise unfolding with sub-matrices $\mathbf{A}_q \triangleq \mathbf{A}_q\mathbf{D}_q$.

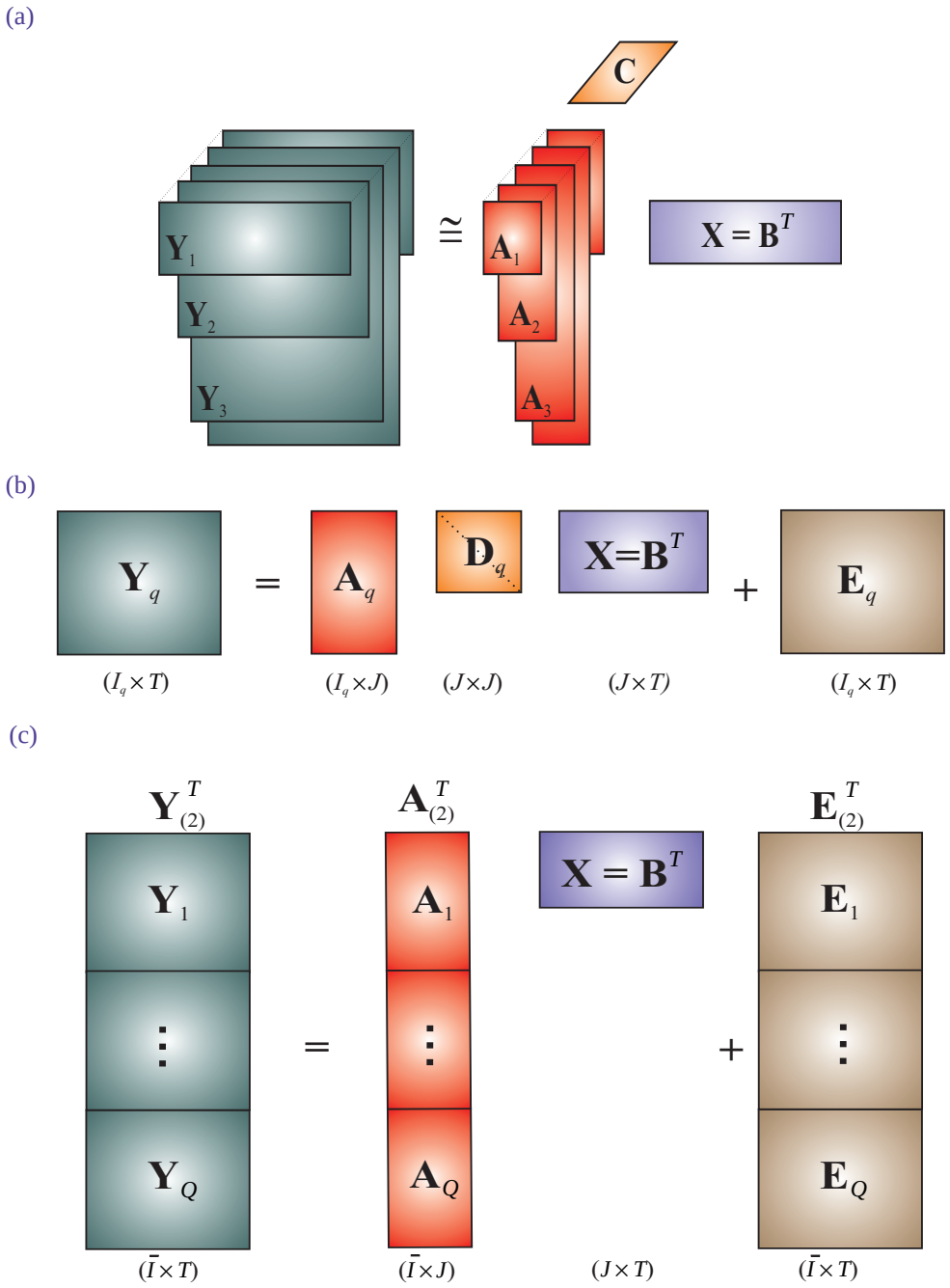


Fig. 1.32 (a) Extended NTF2 model. (b) An equivalent representation in which the frontal slices of a three-way array are factorized by a set of nonnegative matrices. (c) Global matrix representation using column-wise unfolding with sub-matrices $\mathbf{A}_q \triangleq \mathbf{A}_q\mathbf{D}_q$.

where $\bar{\mathbf{Y}} = \mathbf{Y}_{(2)}^T = [\mathbf{Y}_1; \mathbf{Y}_2; \ldots; \mathbf{Y}_Q] \in \mathbb{R}_+^{\bar{I} \times T}$, $\bar{\mathbf{A}} = \mathbf{A}_{(2)}^T = [\mathbf{A}_1; \mathbf{A}_2; \ldots; \mathbf{A}_Q] \in \mathbb{R}_+^{\bar{I} \times J}$, $\bar{\mathbf{E}} = \mathbf{E}_{(2)}^T = [\mathbf{E}_1; \mathbf{E}_2; \ldots; \mathbf{E}_Q] \in \mathbb{R}_+^{\bar{I} \times T}$ and $\bar{I} = \sum_q I_q$.

1.5.5 Individual Differences in Scaling (INDSCAL) and Implicit Slice Canonical Decomposition Model (IMCAND)

Individual Differences in Scaling (INDSCAL) was proposed by Carroll and Chang [24] in the same paper in which they introduced CANDECOMP, and is a special case of the three-way PARAFAC for third-order tensors that are symmetric in two modes.

The INDSCAL imposes the constraint that the first two factor matrices in the decomposition are the same, that is,

$$\underline{\mathbf{Y}} \cong \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{a}_j \circ \boldsymbol{c}_j, \tag{1.148}$$

or equivalently

$$\underline{\mathbf{Y}} \cong \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{C}, \tag{1.149}$$

where $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times I \times Q}$ with $y_{itq} = y_{tiq}, i = 1, \ldots, I, t = 1, \ldots, I, q = 1, \ldots, Q$, and $\underline{\mathbf{I}}$ is the cubic identity tensor. The goal is to find an optimal solution for matrices $\mathbf{A}$ and $\mathbf{C}$, subject to the nonnegativity and sparsity constraints.

In data mining applications third-order input tensors $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times I \times Q}$ may have a special form where each frontal slice $\mathbf{Y}_q$ is the product of two matrices which are typically the matrix $\mathbf{X}_q \in \mathbb{R}^{I \times T}$ and its transpose $\mathbf{X}_q^T$, thus yielding

$$\mathbf{Y}_q = \mathbf{X}_q \mathbf{X}_q^T, \qquad (q = 1, 2, \ldots, Q). \tag{1.150}$$

Such a model is called the IMplicit Slice Canonical Decomposition (IMSCAND) Model (see Figure 1.33) where the PARAFAC or Tucker decompositions of the tensor $\underline{\mathbf{Y}}$ are performed implicitly, that is, by using matrices $\mathbf{X}_q$ and not directly the elements y_{itq} (which do not need to be stored on computer) [121].

For example, these slice matrices may represent covariance matrices in signal processing, whereas in text mining (clustering of scientific publications from a set of SIAM journals) slices $\mathbf{Y}_q$ are document by document matrices and may have the following meanings [121]:

- $\mathbf{Y}_1$ = similarity between names of authors,
- $\mathbf{Y}_2$ = similarity between words in the abstract,
- $\mathbf{Y}_3$ = similarity between author-specified keywords,
- $\mathbf{Y}_4$ = similarity between titles,
- $\mathbf{Y}_5$ = co-citation information,
- $\mathbf{Y}_6$ = co-reference information.

The first four slices are formed from feature-document matrices for the specified similarity. If there exists no similarity between two documents, then the corresponding element in a slice is

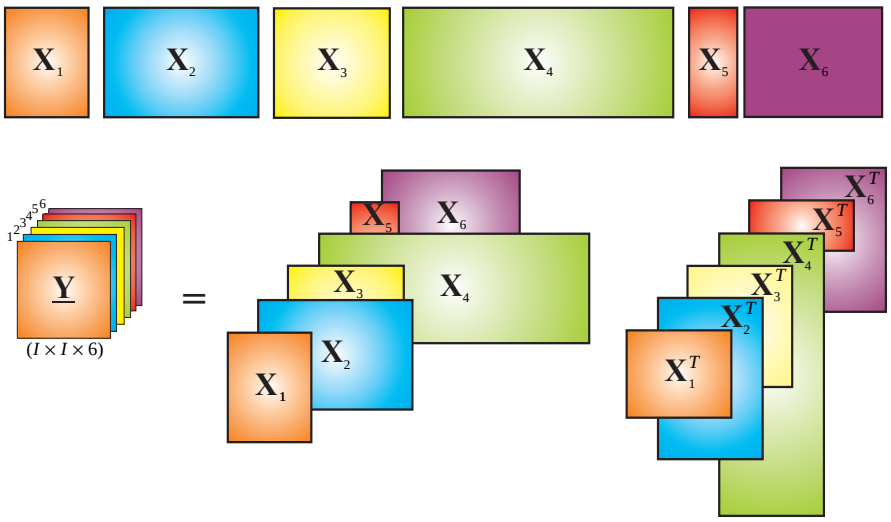


Fig. 1.33 Illustration of IMplicit Slice Canonical Decomposition (IMSCAND). The frontal slices $\mathbf{Y}_q$ are not stored directly but rather represented by a set of matrices $\mathbf{X}_q$ as $\mathbf{Y}_q = \mathbf{X}_q \mathbf{X}_q^T$ for $q = 1, 2, \ldots, 6$.

nonzero. For the fifth slice, the element y_{it5} indicates the number of papers that both documents i and t cite. Whereas the element y_{it6} on the sixth slice is the number of papers cited by both documents i and t.

1.5.6 Shifted PARAFAC and Convolutive NTF

Harshman *et al.* [64] introduced the shifted PARAFAC (S-PARAFAC) in order to deal with shift factors in sequential data such as time series or spectra data. For example, the S-PARAFAC for mode-2 can be described for each entry y_{itq} as

$$y_{itq} = \sum_{j=1}^{J} a_{ij}\, b_{(t+s_{qj})j}\, c_{qj} + e_{itq}, \tag{1.151}$$

where the shift parameter s_{qj} gives the shift at column j of the factor $\mathbf{B}$. We can rewrite this model for frontal slices

$$\mathbf{Y}_q = \mathbf{A}\,\mathbf{D}_q\,\mathcal{S}_{s_q}(\mathbf{B})^T + \mathbf{E}_q, \qquad (q = 1, \ldots, Q), \tag{1.152}$$

where the shift operator (or function) $\mathcal{S}_{s_q}(\mathbf{B})$ shifts all elements in each column of the matrix $\mathbf{B}$ by amount s_{qj}. The vector $\boldsymbol{s}_q$ is a vector of J shift values taken from row q of the (implied) shift matrix $\mathbf{S} \in \mathbb{R}^{Q \times J}$, and the matrix $\mathbf{D}_q$ is a diagonal matrix containing the q-th row of $\mathbf{C}$. One limitation of S-PARAFAC is that it only considers one-dimensional shifts, typically time, but does not handle two-dimensional shifts that might be encountered in neuroimages of brain scans [2], [107]

Table 1.3 Basic description of PARAFAC (CP) and NTF (if we impose additional nonnegativity constraints) family models. Some models are expressed in matrix and/or scalar notations to make it easier understand the differences and compare them with standard PARAFAC. For Shifted NTF (S-NTF), s_{qj} represents the shift at column q for the j-th factor. For Convolutive NTF (CNTF), s is used usually to capture the shifts in the frequency spectrogram.

Model	Description
Nonnegative PARAFAC (NTF)	$y_{itq} = \sum_{j=1}^{J} a_{ij}\, b_{tj}\, c_{qj} + e_{itq}$ $\mathbf{Y}_q = \mathbf{A}\mathbf{D}_q\mathbf{B}^T + \mathbf{E}_q = \sum_{j=1}^{J} c_{qj}\boldsymbol{a}_j\, \boldsymbol{b}_j^T + \mathbf{E}_q$
NTF1	$y_{itq} = \sum_{j=1}^{J} a_{ij}\, b_{tjq}\, c_{qj} + e_{itq}$ $\mathbf{Y}_q = \mathbf{A}\,\mathbf{D}_q\,\mathbf{B}_q^T + \mathbf{E}_q = \sum_{j=1}^{J} c_{qj}\,\boldsymbol{a}_j\,(\boldsymbol{b}_j^{(q)})^T + \mathbf{E}_q$
NTF2	$y_{itq} = \sum_{j=1}^{J} a_{ijq}\, b_{tj}\, c_{qj} + e_{itq}$ $\mathbf{Y}_q = \mathbf{A}_q\,\mathbf{D}_q\,\mathbf{B}^T + \mathbf{E}_q = \sum_{j=1}^{J} c_{qj}\,\boldsymbol{a}_j^{(q)}\,\boldsymbol{b}_j^T + \mathbf{E}_q$
Shifted NTF (S-NTF)	$y_{itq} = \sum_{j=1}^{J} a_{ij}\, b_{(t+s_{qj})j}\, c_{qj} + e_{itq}$ $\mathbf{Y}_q = \mathbf{A}\,\mathbf{D}_q\,\mathcal{S}_{s_q}(\mathbf{B})^T + \mathbf{E}_q$
Convolutive NTF (CNTF)	$y_{itq} = \sum_{j=1}^{J}\sum_{s=1}^{S} a_{ij}\, b_{(t-s+1)j}\, c_{qjs} + e_{itq}$ $\mathbf{Y}_q = \mathbf{A}\sum_{s=1}^{S}\mathbf{D}_q^{(s)}\,(\mathbf{T}_{\uparrow(s-1)}\mathbf{B})^T + \mathbf{E}_q$
C2NTF	$y_{itq} = \sum_{j=1}^{J}\sum_{s=1}^{S}\sum_{r=1}^{R} a_{ij}\, b_{(t-s+1)jr}\, c_{(q-r+1)js} + e_{itq}$
INDSCAL	$y_{itq} = \sum_{j=1}^{J} a_{ij}\, a_{tj}\, c_{qj} + e_{itq}$

Another extension of PARAFAC is Convolutive PARAFAC (CPARAFAC or CNTF) which is a generalization of CNMF to multiway spectral data. Morup and Schmidt [107] introduced this model with the name Sparse Nonnegative Tensor 2D Deconvolution (SNTF2D). The single convolutive NTF on mode-2 and mode-3 and with rank-J for the nonnegative tensor $\underline{\mathbf{Y}} \in \mathbb{R}_+^{I \times T \times Q}$ returns a factor matrix $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ on the first dimension, a factor matrix $\mathbf{B} \in \mathbb{R}_+^{T \times J}$ on the second dimension, and a set of S factor matrices or tensor $\underline{\mathbf{C}} \in \mathbb{R}_+^{Q \times J \times S}$, and can be expressed as follows

$$y_{itq} = \sum_{j=1}^{J} \sum_{s=1}^{S} a_{ij}\, b_{(t-s+1)j}\, c_{qjs} + e_{itq}. \tag{1.153}$$

For $S = 1$, CPARAFAC (CNTF) simplifies to PARAFAC (NTF). Matrix representation of this model via frontal slices $\mathbf{Y}_q$, $(q = 1, \ldots, Q)$ is given by

$$\mathbf{Y}_q \cong \mathbf{A} \sum_{s=0}^{S-1} \mathbf{D}_q^{(s+1)} (\mathbf{T}_{\uparrow(s)} \mathbf{B})^T = \mathbf{A} \sum_{s=0}^{S-1} \mathbf{D}_q^{(s+1)} \mathbf{B}^T \mathbf{T}_{\underset{\rightarrow}{s}} \tag{1.154}$$

where $\mathbf{D}_q^{(s)}$ is a diagonal matrix containing the fiber $\boldsymbol{c}_{q:s}$, and the shift operators $\mathbf{T}_{\uparrow(s)}$, $\mathbf{T}_{\underset{\rightarrow}{s}}$ are defined in Section 1.2.12. In the tensor form the CNTF can be described as

$$\underline{\mathbf{Y}} = \sum_{s=1}^{S} \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{T}_{\uparrow(s-1)} \mathbf{B} \times_3 \mathbf{C}_s + \underline{\mathbf{E}}. \tag{1.155}$$

The CPARAFAC can be extended to the double convolutive model (C2PARAFAC or C2NTF) as

$$y_{itq} \cong \sum_{j=1}^{J} \sum_{s=1}^{S} \sum_{r=1}^{R} a_{ij}\, b_{(t-s+1)jr}\, c_{(q-r+1)js}, \tag{1.156}$$

where the second factor in Equation (1.155) is no longer a matrix but a tensor of size $(T \times J \times R)$.

1.5.7 Nonnegative Tucker Decompositions

The Tucker decomposition, also called the Tucker3 or best rank (J, R, P) approximation, can be formulated as follows[20] [135], [136]:

Given a third-order data tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ and three positive indices $\{J, R, P\} << \{I, T, Q\}$, find a core tensor $\mathbf{G} = [g_{jrp}] \in \mathbb{R}^{J \times R \times P}$ and three component matrices called factor or loading matrices or factors: $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I \times J}$, $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_R] \in \mathbb{R}^{T \times R}$, and $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_P] \in \mathbb{R}^{Q \times P}$, which perform the following approximate decomposition:

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{jrp}\, (\boldsymbol{a}_j \circ \boldsymbol{b}_r \circ \boldsymbol{c}_p) + \underline{\mathbf{E}} \tag{1.157}$$

[20] The Tucker3 model with orthogonal factors is also known as three-way PCA (principal component analysis). The model can be naturally extended to N-way Tucker decomposition of arbitrary N-th order tensor.

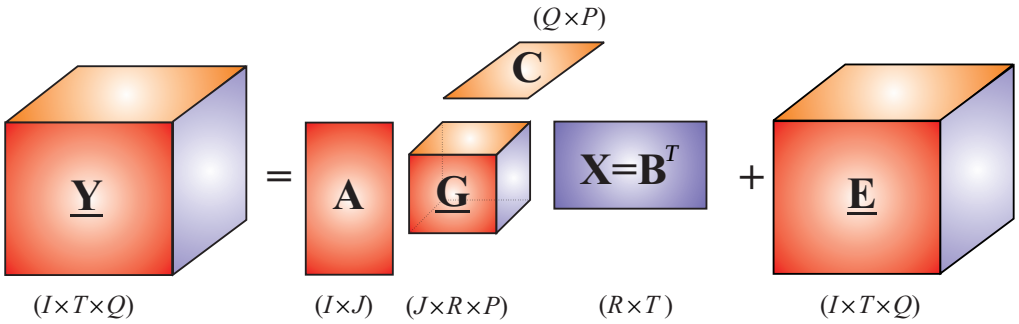


Fig. 1.34 Tucker3 model is a weighted sum of the outer product of three vectors (factors) stored as columns of component matrices $\mathbf{A} \in \mathbb{R}^{I\times J}, \mathbf{B} = \mathbf{X}^T \in \mathbb{R}^{T\times R}$ and $\mathbf{C} \in \mathbb{R}^{Q\times P}$. The core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J\times R\times P}$ defines a linking structure between the set of components and J, R, and P denote the number of components. In order to achieve uniqueness for the Tucker models it is necessary to impose additional constraints such as sparsity and nonnegativity.

or equivalently in the element-wise form

$$y_{itq} = \sum_{j=1}^{J}\sum_{r=1}^{R}\sum_{p=1}^{P} g_{jrp}\, a_{ij}\, b_{tr}\, c_{qp} + e_{itq}, \tag{1.158}$$

where $\boldsymbol{a}_j \in \mathbb{R}^I$, $\boldsymbol{b}_j \in \mathbb{R}^T$, and $\boldsymbol{c}_j \in \mathbb{R}^Q$, (that is, the vectors within the associated component (factor) matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$,), and g_{jrp} are scaling factors which are the entries of a core tensor $\underline{\mathbf{G}} = [g_{jrp}] \in \mathbb{R}^{J\times R\times P}$.

The original Tucker model makes the assumption of orthogonality of the factor matrices (in analogy to SVD), [79], [17], [84], [83], [117], [106]. We will, however, ignore these constraints. By imposing nonnegativity constraints the problem of estimating the component matrices and a core tensor is converted into a generalized NMF problem called the Nonnegative Tucker Decomposition (NTD) (see Chapter 7 for details). The first implementations of Tucker decomposition with nonnegativity constraints together with a number of other constraints were given by Kiers, Smilde and Bro in [79], [17]. The NTD imposes nonnegativity constraints for all component matrices and a core tensor, while a semi-NTD (in analogy to semi-NMF) imposes nonnegativity constraints to only some components matrices and/or some elements of the core tensor.

There are several equivalent mathematical descriptions for the Tucker model (see Table 1.4). It can be expressed in a compact matrix form using mode-n multiplications

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}} = [\![\underline{\mathbf{G}}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] + \underline{\mathbf{E}}, \tag{1.159}$$

where $\widehat{\underline{\mathbf{Y}}} = [\![\underline{\mathbf{G}}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is the shorthand notation for the Tucker3 tensor decomposition.

Using the unfolding approach we can obtain matrix forms expressed compactly by the Kronecker products:

$$\mathbf{Y}_{(1)} \cong \mathbf{A}\, \mathbf{G}_{(1)}\, (\mathbf{C} \otimes \mathbf{B})^T, \tag{1.160}$$

$$\mathbf{Y}_{(2)} \cong \mathbf{B}\, \mathbf{G}_{(2)}\, (\mathbf{C} \otimes \mathbf{A})^T, \tag{1.161}$$

Table 1.4 Formulations of the Tucker3 model for a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times T\times Q}$ with the core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J\times R\times P}$ and the factor matrices: $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I\times J}$, $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_R] \in \mathbb{R}^{T\times R}$, and $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_P] \in \mathbb{R}^{Q\times P}$.

Operator	Mathematical Formula
Outer product	$\underline{\mathbf{Y}} = \sum_{j=1}^{J}\sum_{r=1}^{R}\sum_{p=1}^{P} g_{jrp}\, \boldsymbol{a}_j \circ \boldsymbol{b}_r \circ \boldsymbol{c}_p + \underline{\mathbf{E}}$
Scalar	$y_{itq} = \sum_{j=1}^{J}\sum_{r=1}^{R}\sum_{p=1}^{P} g_{jrp}\, a_{ij}\, b_{tr}\, c_{qp} + e_{itq}$
Mode-n multiplications	$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}}$
Slice representation	$\mathbf{Y}_q = \mathbf{A}\mathbf{H}_q\mathbf{B}^T + \mathbf{E}_q, \quad (q = 1, 2, \ldots, Q)$ $\mathbf{H}_q = \sum_{p=1}^{P} c_{qp}\mathbf{G}_p, \quad \mathbf{G}_p \triangleq \mathbf{G}_{::p} \in \mathbb{R}^{J\times R}$
Vector	$\text{vec}(\underline{\mathbf{Y}}) = \text{vec}(\mathbf{Y}_{(1)}) \cong (\mathbf{C}\otimes\mathbf{B}\otimes\mathbf{A})\,\text{vec}(\underline{\mathbf{G}})$
Kronecker product	$\mathbf{Y}_{(1)} \cong \mathbf{A}\,\mathbf{G}_{(1)}\,(\mathbf{C}\otimes\mathbf{B})^T$ $\mathbf{Y}_{(2)} \cong \mathbf{B}\,\mathbf{G}_{(2)}\,(\mathbf{C}\otimes\mathbf{A})^T$ $\mathbf{Y}_{(3)} \cong \mathbf{C}\,\mathbf{G}_{(3)}\,(\mathbf{B}\otimes\mathbf{A})^T$

$$\mathbf{Y}_{(3)} \cong \mathbf{C}\,\mathbf{G}_{(3)}\,(\mathbf{B}\otimes\mathbf{A})^T. \tag{1.162}$$

It is often convenient to represent the three-way Tucker model in its vectorized forms

$$\text{vec}\,(\mathbf{Y}_{(1)}) \cong \text{vec}\,(\mathbf{A}\mathbf{G}_{(1)}(\mathbf{C}\otimes\mathbf{B})^T) = (\mathbf{C}\otimes\mathbf{B})\otimes\mathbf{A}\,\text{vec}\,(\mathbf{G}_{(1)}), \tag{1.163}$$

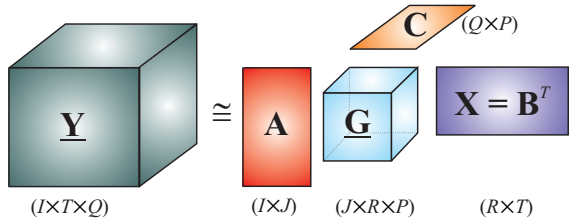
$$\text{vec}\,(\mathbf{Y}_{(2)}) \cong \text{vec}\,(\mathbf{B}\mathbf{G}_{(2)}(\mathbf{C}\otimes\mathbf{A})^T) = (\mathbf{C}\otimes\mathbf{A})\otimes\mathbf{B}\,\text{vec}\,(\mathbf{G}_{(2)}), \tag{1.164}$$

$$\text{vec}\,(\mathbf{Y}_{(3)}) \cong \text{vec}\,(\mathbf{C}\mathbf{G}_{(3)}(\mathbf{B}\otimes\mathbf{A})^T) = (\mathbf{B}\otimes\mathbf{A})\otimes\mathbf{C}\,\text{vec}\,(\mathbf{G}_{(3)}). \tag{1.165}$$

The Tucker model described above is often called the Tucker3 model because a third-order tensor is decomposed into three factor (loading) matrices (say, $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$) and a core tensor $\underline{\mathbf{G}}$. In applications where we have two factor matrices or even only one, the Tucker3 model for a three-way tensor simplifies into the Tucker2 or Tucker1 models (see Table 1.5). The Tucker2 model can be obtained from the Tucker3 model by absorbing one factor by a core tensor (see Figure 1.35(b)), that is,

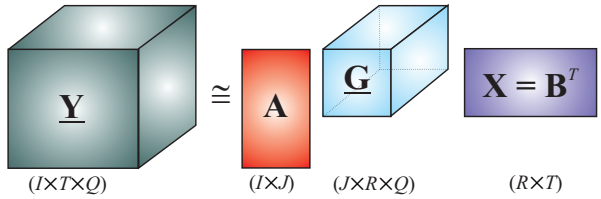
$$\underline{\mathbf{Y}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B}. \tag{1.166}$$

$$y_{itq} = \sum_{j=1}^{J}\sum_{r=1}^{R}\sum_{p=1}^{P} g_{jrp}\, a_{ij}\, b_{tr}\, c_{qp}$$



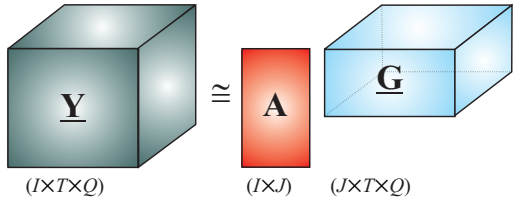
(a) Tucker3

$$y_{itq} = \sum_{j=1}^{J}\sum_{r=1}^{R} g_{jrq}\, a_{ij}\, b_{tr}$$



(b) Tucker2

$$y_{itq} = \sum_{j=1}^{J} g_{jtq}\, a_{ij}$$



(c) Tucker1

Fig. 1.35 Summary of the three related Tucker decompositions.

For the Tucker1 model we have only one factor matrix (while two others are absorbed by a core tensor) which is described as (see also Figure 1.35(c))

$$\underline{\mathbf{Y}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A}. \tag{1.167}$$

It is interesting to note that the approximation of a tensor by factor matrices and a core tensor often helps to simplify mathematical operations and reduce the computation cost of some operations in multi-linear (tensor) algebra. For example:

$$\begin{aligned} \underline{\mathbf{Y}} = \underline{\mathbf{X}} \,\bar{\times}_3\, \boldsymbol{a} &\approx (\underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}) \,\bar{\times}_3\, \boldsymbol{a} \\ &= (\underline{\mathbf{G}} \,\bar{\times}_3 \mathbf{C}^T \boldsymbol{a}) \times_1 \mathbf{A} \times_2 \mathbf{B} \\ &= \underline{\mathbf{G}}_{Ca} \times_1 \mathbf{A} \times_2 \mathbf{B}, \end{aligned}$$

where $\underline{\mathbf{G}}_{Ca} = \underline{\mathbf{G}} \,\bar{\times}_3\, \mathbf{C}^T \boldsymbol{a}$. Comparison of tensor decomposition models are summarized in Table 1.6.

Table 1.5 Tucker models for a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$.

Model	Description
Tucker1	$y_{itq} = \sum_{j=1}^{J} g_{jtq}\, a_{ij} + e_{itq}$ $\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} + \underline{\mathbf{E}}, \quad (\underline{\mathbf{G}} \in \mathbb{R}^{J \times T \times Q})$ $\mathbf{Y}_q = \mathbf{A}\mathbf{G}_q + \mathbf{E}_q, \quad (q = 1, 2, \ldots, Q)$ $\mathbf{Y}_{(1)} = \mathbf{A}\mathbf{G}_{(1)} + \mathbf{E}_{(1)}$
Tucker2	$y_{itq} = \sum_{j=1}^{J} \sum_{r=1}^{R} g_{jrq}\, a_{ij}\, b_{tr} + e_{itq}$ $\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} + \underline{\mathbf{E}}, \quad (\underline{\mathbf{G}} \in \mathbb{R}^{J \times R \times Q})$ $\underline{\mathbf{Y}} = \sum_{j=1}^{J} \sum_{r=1}^{R} \boldsymbol{a}_j \circ \boldsymbol{b}_r \circ \boldsymbol{g}_{jr} + \underline{\mathbf{E}}$ $\mathbf{Y}_{(3)} = \mathbf{G}_{(3)} (\mathbf{B} \otimes \mathbf{A})^T + \mathbf{E}_{(3)}$
Tucker3	$y_{itq} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{jrp}\, a_{ij}\, b_{tr}\, c_{qp} + e_{itq}$ $\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}}, \quad (\underline{\mathbf{G}} \in \mathbb{R}^{J \times R \times P})$
Shifted Tucker3	$y_{itq} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{jrp}\, a_{(i+s_{tj})j}\, b_{tr}\, c_{qp} + e_{itq}$

1.5.8 Block Component Decompositions

Block Component Decompositions (BCDs) (also called Block Component Models) introduced by De Lathauwer and Nion for applications in signal processing and wireless communications [46], [47], [110], [109], [48] can be considered as a sum of basic subtensor decompositions (see Figure 1.36). Each basic subtensor in this sum has the same kind of factorization or decomposition, typically, Tucker2 or Tucker3 decomposition, and the corresponding components have a similar structure (regarding dimensions, sparsity profile and nonnegativity constraints) as illustrated in Figures 1.36 (a), (b) and (c).

The model shown in Figure 1.36(a), called the BCD rank-$(J_r, 1)$, decomposes a data tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ into a sum of R subtensors $\underline{\mathbf{Y}}^{(r)} \in \mathbb{R}^{I \times T \times Q}$, $(r = 1, 2, \ldots, R)$. Each of the subtensor $\underline{\mathbf{Y}}^{(r)}$ is factorized into three factors $\mathbf{A}_r \in \mathbb{R}^{I \times J_r}$, $\mathbf{B}_r \in \mathbb{R}^{T \times J_r}$ and $\boldsymbol{c}_r \in \mathbb{R}^{Q}$. The mathematical

Table 1.6 Matrix and tensor representations for various factorization models (for most of the models we impose additional nonnegativity constraints).

Model	Matrix Representation	Tensor Representation
NMF	$\mathbf{Y} \cong \mathbf{A}\,\mathbf{X} = \mathbf{A}\,\mathbf{B}^T$	$\mathbf{Y} \cong \mathbf{I} \times_1 \mathbf{A} \times_2 \mathbf{X}^T$
SVD	$\mathbf{Y} \cong \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T = \sum_{r=1}^{R} \sigma_r\, \boldsymbol{u}_r \boldsymbol{v}_r^T$	$\mathbf{Y} \cong \mathbf{\Sigma} \times_1 \mathbf{U} \times_2 \mathbf{V} = \sum_{r=1}^{R} \sigma_r\, \boldsymbol{u}_r \circ \boldsymbol{v}_r$
Three-factor NMF	$\mathbf{Y} \cong \mathbf{A}\,\mathbf{S}\,\mathbf{X} = \mathbf{A}\,\mathbf{S}\,\mathbf{B}^T$	$\mathbf{Y} \cong \mathbf{S} \times_1 \mathbf{A} \times_2 \mathbf{X}^T$
NTF (nonnegative PARAFAC)	$\mathbf{Y}_q \cong \mathbf{A}\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{B}^T$ $(q = 1, 2, \ldots, Q)$	$\underline{\mathbf{Y}} \cong \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{j=1}^{J} \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j$
NTF1	$\mathbf{Y}_q \cong \mathbf{A}\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{B}_q^T = \mathbf{A}\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{X}_q$ $(q = 1, 2, \ldots, Q)$	$\underline{\mathbf{Y}} \cong \underline{\mathbf{X}} \times_1 \mathbf{A} \times_3 \mathbf{C}$
NTF2	$\mathbf{Y}_q \cong \mathbf{A}_q\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{B}^T = \mathbf{A}_q\,\mathbf{D}_q(\boldsymbol{c}_{q:})\,\mathbf{X}$ $(q = 1, 2, \ldots, Q)$	$\underline{\mathbf{Y}} \cong \underline{\mathbf{A}} \times_2 \mathbf{B} \times_3 \mathbf{C}$
Tucker1	$\mathbf{Y}_q \cong \mathbf{A}\,\mathbf{G}_{::q}$ $(q = 1, 2, \ldots, Q)$	$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A}$
Tucker2	$\mathbf{Y}_q \cong \mathbf{A}\,\mathbf{G}_{::q}\,\mathbf{B}^T$ $(q = 1, 2, \ldots, Q)$	$\underline{\mathbf{Y}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} = \sum_{j=1}^{J} \sum_{r=1}^{R} \boldsymbol{a}_j \circ \boldsymbol{b}_r \circ \boldsymbol{g}_{jr}$
Tucker3	$\mathbf{Y}_q \cong \mathbf{A}\mathbf{H}_q\mathbf{B}^T$ $\mathbf{H}_q = \sum_{p=1}^{P} \boldsymbol{c}_{qp}\,\mathbf{G}_{::p}$	$\underline{\mathbf{Y}} \cong \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_2 \mathbf{C} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{jrp}\,(\boldsymbol{a}_j \circ \boldsymbol{b}_r \circ \boldsymbol{c}_p)$

description of this BCD model is given by

$$\underline{\mathbf{Y}} = \sum_{r=1}^{R} \underline{\mathbf{Y}}^{(r)} + \underline{\mathbf{E}} = \sum_{r=1}^{R} (\mathbf{A}_r \mathbf{B}_r^T) \circ \boldsymbol{c}_r + \underline{\mathbf{E}}, \tag{1.168}$$

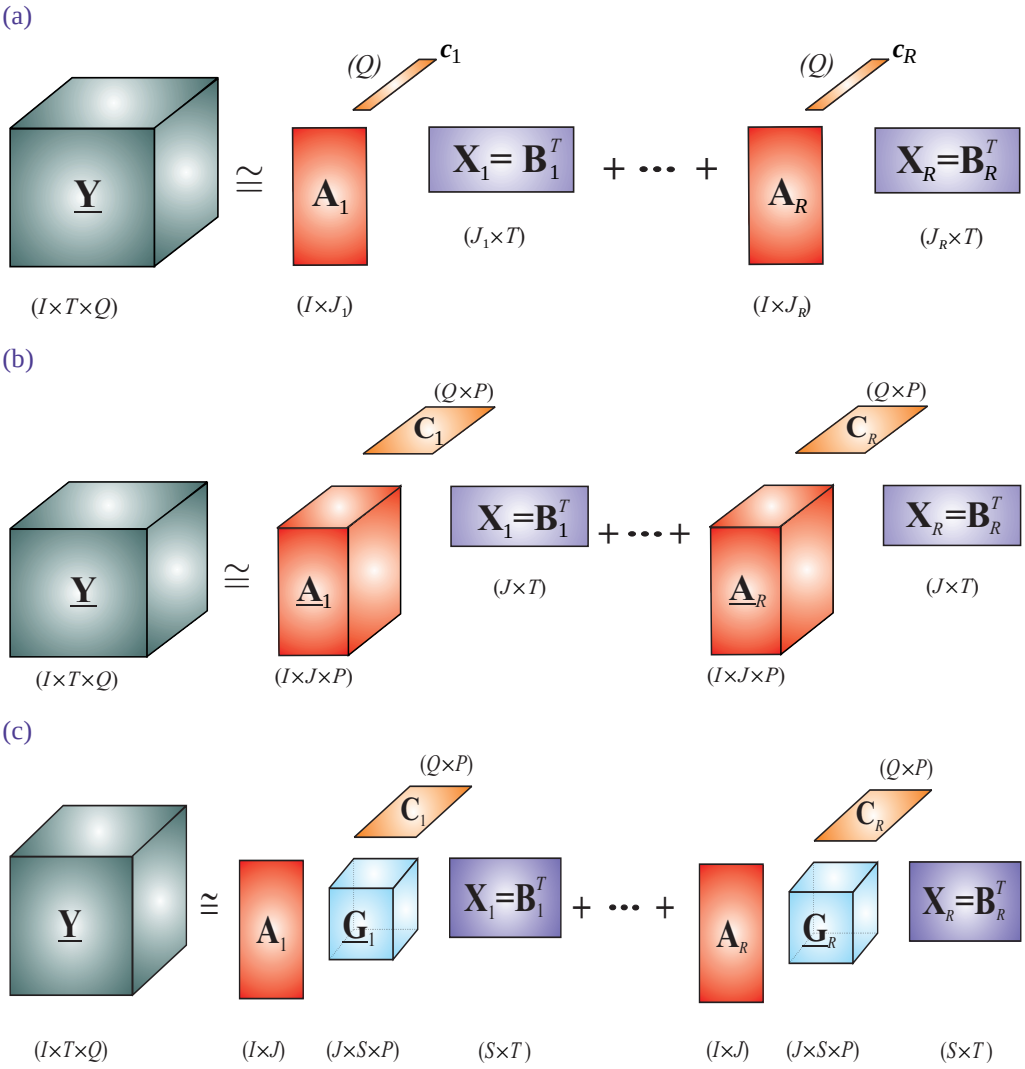


Fig. 1.36 Block Component Decompositions (BCD) for a third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times T\times Q}$: (a) BCD with rank-$(J_r, 1)$, (b) BCD with rank -$(J, P)$ and (c) BCD with rank-(J, S, P).

or equivalently

$$\underline{\mathbf{Y}} = \sum_{r=1}^{R} \underline{\mathbf{A}}_r \times_2 \mathbf{B}_r \times_3 \boldsymbol{c}_r + \underline{\mathbf{E}}, \tag{1.169}$$

where tensors $\underline{\mathbf{A}}_r \triangleq \mathbf{A}_r \in \mathbb{R}^{I\times J_r\times 1}$ are three-way tensors with only one frontal slice. With this notation, each subtensor $\underline{\mathbf{Y}}^{(r)}$ is a Tucker2 model with the core tensor $\underline{\mathbf{A}}_{(r)}$, and factors $\mathbf{B}_r$ and $\boldsymbol{c}_r$. Hence, the BCD rank-$(J_r, 1)$ decomposition can be considered as a sum of simplified Tucker-2 models. The objective is to estimate component matrices $\mathbf{A}_r \in \mathbb{R}^{I\times J_r}$, $\mathbf{B}_r \in \mathbb{R}^{T\times J_r}$, $(r =$

$1, 2, \ldots, R$) and a factor matrix $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_R] \in \mathbb{R}^{Q \times R}$ subject to optional nonnegativity and sparsity constraints.

Using the unfolding approach, the above BCD model can be written in several equivalent forms:

$$\begin{aligned} \mathbf{Y}_{(1)} &\cong \sum_{r=1}^{R} \left[\underline{\mathbf{A}}_r\right]_{(1)} (\boldsymbol{c}_r \otimes \mathbf{B}_r)^T = \sum_{r=1}^{R} \mathbf{A}_r (\boldsymbol{c}_r \otimes \mathbf{B}_r)^T \\ &= \boxed{\left[\mathbf{A}_1 \ \mathbf{A}_2 \ \cdots \ \mathbf{A}_R\right] \left[\boldsymbol{c}_1 \otimes \mathbf{B}_1 \ \boldsymbol{c}_2 \otimes \mathbf{B}_2 \ \cdots \ \boldsymbol{c}_R \otimes \mathbf{B}_R\right]^T}, \end{aligned} \tag{1.170}$$

$$\begin{aligned} \mathbf{Y}_{(2)} &\cong \sum_{r=1}^{R} \mathbf{B}_r \left[\underline{\mathbf{A}}_r\right]_{(2)} (\boldsymbol{c}_r \otimes \mathbf{I}_I)^T = \sum_{r=1}^{R} \mathbf{B}_r \mathbf{A}_r^T (\boldsymbol{c}_r \otimes \mathbf{I}_I)^T \\ &= \sum_{r=1}^{R} \mathbf{B}_r [(\boldsymbol{c}_r \otimes \mathbf{I}_I) \mathbf{A}_r]^T = \sum_{r=1}^{R} \mathbf{B}_r (\boldsymbol{c}_r \otimes \mathbf{A}_r)^T \\ &= \boxed{\left[\mathbf{B}_1 \ \mathbf{B}_2 \ \cdots \ \mathbf{B}_R\right] \left[\boldsymbol{c}_1 \otimes \mathbf{A}_1 \ \boldsymbol{c}_2 \otimes \mathbf{A}_2 \ \cdots \ \boldsymbol{c}_R \otimes \mathbf{A}_R\right]^T}, \end{aligned} \tag{1.171}$$

$$\begin{aligned} \mathbf{Y}_{(3)} &\cong \sum_{r=1}^{R} \boldsymbol{c}_r \left[\underline{\mathbf{A}}_r\right]_{(3)} (\mathbf{B}_r \otimes \mathbf{I}_I)^T = \sum_{r=1}^{R} \boldsymbol{c}_r \text{vec}\,(\mathbf{A}_r)^T (\mathbf{B}_r \otimes \mathbf{I}_I)^T \\ &= \sum_{r=1}^{R} \boldsymbol{c}_r [(\mathbf{B}_r \otimes \mathbf{I}_I) \text{vec}\,(\mathbf{A}_r)]^T = \sum_{r=1}^{R} \boldsymbol{c}_r \text{vec}\left(\mathbf{A}_r \mathbf{B}_r^T\right)^T \\ &= \boxed{\mathbf{C}\left[\text{vec}\left(\mathbf{A}_1 \mathbf{B}_1^T\right) \ \text{vec}\left(\mathbf{A}_2 \mathbf{B}_2^T\right) \ \cdots \ \text{vec}\left(\mathbf{A}_R \mathbf{B}_R^T\right)\right]^T}. \end{aligned} \tag{1.172}$$

A simple and natural extension of the model BCD rank-$(J_r, 1)$ assumes that the tensors $\underline{\mathbf{A}}_r \in \mathbb{R}^{I \times J \times P}$ contains P (instead of one) frontal slices of size $(I \times J)$ (see Figure 1.36(b)). This model is referred to as the BCD with rank-(J, P) and is described as follows

$$\boxed{\underline{\mathbf{Y}} = \sum_{r=1}^{R} \left(\underline{\mathbf{A}}_r \times_2 \mathbf{B}_r \times_3 \mathbf{C}_r\right) + \underline{\mathbf{E}}.} \tag{1.173}$$

The objective is to find a set of R tensors $\underline{\mathbf{A}}_r \in \mathbb{R}^{I \times J \times P}$, a tensor $\underline{\mathbf{B}} \in \mathbb{R}^{T \times J \times R}$, and a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{Q \times P \times R}$. By stacking tensors $\underline{\mathbf{A}}_r$ along their third dimension, we form a common tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I \times J \times PR}$. The mode-1 matricization of this BCD model gives an equivalent matrix factorization model [110], [48]:

$$\mathbf{Y}_{(1)} \cong \sum_{r=1}^{R} \mathbf{A}_{r(1)} (\mathbf{C}_r \otimes \mathbf{B}_r)^T \tag{1.174}$$

$$= \boxed{\mathbf{A}_{(1)} \left[\mathbf{C}_1 \otimes \mathbf{B}_1 \ \mathbf{C}_2 \otimes \mathbf{B}_2 \ \cdots \ \mathbf{C}_R \otimes \mathbf{B}_R\right]^T}. \tag{1.175}$$

The most general, BCD rank-(J, S, P) model is formulated as a sum of R Tucker3 models of corresponding factors $\mathbf{A}_r \in \mathbb{R}^{I\times J}$, $\mathbf{B}_r \in \mathbb{R}^{T\times S}$, $\mathbf{C}_r \in \mathbb{R}^{Q\times P}$ and core tensor $\underline{\mathbf{G}}_r \in \mathbb{R}^{J\times S\times P}$, and described in a compact form as (see Figure 1.36(c)):

$$\boxed{\underline{\mathbf{Y}} = \sum_{r=1}^{R} \left(\underline{\mathbf{G}}_r \times_1 \mathbf{A}_r \times_2 \mathbf{B}_r \times_3 \mathbf{C}_r\right) + \underline{\mathbf{E}}.} \tag{1.176}$$

This model can be also converted in a similar way to a matrix factorization model with set of constrained component matrices.

1.5.9 Block-Oriented Decompositions

A natural extension of the tensor decomposition models discussed in the previous sections will be a decomposition which uses sum of subtensors factorized the data tensor along different modes. Such decompositions will be referred to as Block-Oriented Decompositions (BODs). The key distinction between BOD and BCD models is that subtensors in a BCD model attempt to explain the data tensor in the same modes while BOD models exploit at least two up to all possible separate modes for each subtensors. For example, using Tucker2 model the corresponding BOD2 model can be formulated as follows

$$\underline{\mathbf{Y}} \cong \underline{\mathbf{G}}_1 \times_1 \mathbf{A}_1 \times_2 \mathbf{B}_1 + \underline{\mathbf{G}}_2 \times_1 \mathbf{A}_2 \times_3 \mathbf{C}_1 + \underline{\mathbf{G}}_3 \times_2 \mathbf{B}_2 \times_3 \mathbf{C}_2, \tag{1.177}$$

where core tensors and factor matrices have suitable dimensions.

Analogously, we can define a simpler BOD1 model which is based on the Tucker1 models (see Figure 1.37):

$$\boxed{\underline{\mathbf{Y}} \cong \underline{\mathbf{H}} \times_1 \mathbf{A} + \underline{\mathbf{L}} \times_2 \mathbf{B} + \underline{\mathbf{F}} \times_3 \mathbf{C},} \tag{1.178}$$

where tensors $\underline{\mathbf{H}} \in \mathbb{R}^{R_1\times T\times Q}$, $\underline{\mathbf{L}} \in \mathbb{R}^{I\times R_2\times Q}$, $\underline{\mathbf{F}} \in \mathbb{R}^{I\times T\times R_3}$ are core tensors in the Tucker1 models with mode-n, $n = 1, 2, 3$, $\mathbf{A} \in \mathbb{R}^{I\times R_1}$, $\mathbf{B} \in \mathbb{R}^{T\times R_2}$ and $\mathbf{C} \in \mathbb{R}^{Q\times R_3}$ are corresponding factors. The objective is to find three core tensors $\underline{\mathbf{H}}$, $\underline{\mathbf{L}}$, $\underline{\mathbf{F}}$ and three corresponding factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. This model is also called the Slice Oriented Decomposition (SOD) which was recently proposed and investigated by Caiafa and Cichocki [23], and may have various mathematical and graphical representations.

Remark 1.1 *The main motivation to use BOD1 model is to eliminate offset in a data tensor and provide unique and meaningful representation of the extracted components. The BOD1 can be considered as a generalization or extension of the affine NMF model presented in section (1.2.7). As we will show in Chapter 7 the BOD1 model can resolve the problem related with offset degraded by flicker, occlusion or discontinuity.*

Using matricization approach we obtain for BOD1 model several equivalent matrix factorization models:

$$\begin{aligned}
\mathbf{Y}_{(1)} &\cong \mathbf{A}\,\mathbf{H}_{(1)} + \mathbf{L}_{(1)}\,(\mathbf{I}_Q \otimes \mathbf{B})^T + \mathbf{F}_{(1)}\,(\mathbf{C} \otimes \mathbf{I}_T)^T \\
&= \boxed{\left[\mathbf{A}\ \mathbf{L}_{(1)}\ \mathbf{F}_{(1)}\right]\left[\mathbf{H}_{(1)}\ \mathbf{I}_Q \otimes \mathbf{B}^T\ \mathbf{C}^T \otimes \mathbf{I}_T\right],} \\
\mathbf{Y}_{(2)} &\cong \mathbf{H}_{(2)}\,(\mathbf{I}_Q \otimes \mathbf{A})^T + \mathbf{B}\,\mathbf{L}_{(2)} + \mathbf{F}_{(2)}\,(\mathbf{C} \otimes \mathbf{I}_I)^T
\end{aligned} \tag{1.179}$$

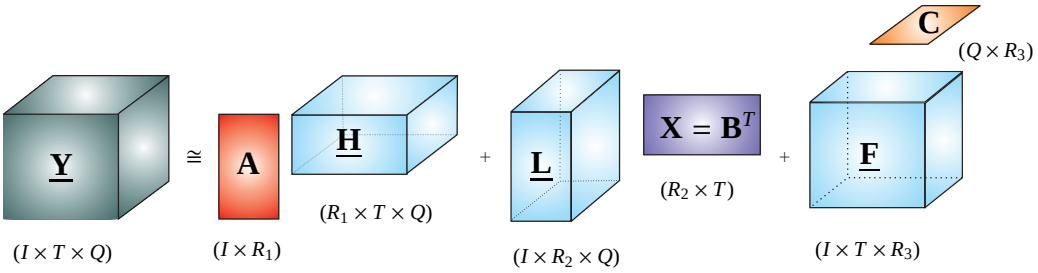


Fig. 1.37 Illustration of the Block-Oriented Decomposition (BOD1) for a third-order tensor. Three Tucker1 models express the data tensor along each modes. Typically, core tensors $\underline{\mathbf{H}} \in \mathbb{R}^{R_1 \times T \times Q}$, $\underline{\mathbf{L}} \in \mathbb{R}^{I \times R_2 \times Q}$, $\underline{\mathbf{F}} \in \mathbb{R}^{I \times T \times R_3}$ have much smaller dimensions than a data tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$, i.e., $R_1 << I$, $R_2 << T$, and $R_3 << Q$.

$$= \boxed{\left[\mathbf{H}_{(2)}\ \mathbf{B}\ \mathbf{F}_{(2)}\right]\left[\mathbf{I}_Q \otimes \mathbf{A}^T\ \mathbf{L}_{(2)}\ \mathbf{C}^T \otimes \mathbf{I}_I\right]}, \tag{1.180}$$

$$\mathbf{Y}_{(3)} \cong \mathbf{H}_{(3)}\,(\mathbf{I}_T \otimes \mathbf{A})^T + \mathbf{L}_{(3)}\,(\mathbf{B} \otimes \mathbf{I}_I)^T + \mathbf{C}\,\mathbf{F}_{(3)}$$

$$= \boxed{\left[\mathbf{H}_{(3)}\ \mathbf{L}_{(3)}\ \mathbf{C}\right]\left[\mathbf{I}_T \otimes \mathbf{A}^T\ \mathbf{B}^T \otimes \mathbf{I}_I\ \mathbf{F}_{(3)}\right]}. \tag{1.181}$$

These matrix representations allow us to compute core tensors and factor matrices via matrix factorization.

Similar, but more sophisticated BOD models can be defined based on a restricted Tucker3 model [128] and also PARATUCK2 or DEDICOM models (see the next section).

1.5.10 PARATUCK2 and DEDICOM Models

The PARATUCK2, developed by Harshman and Lundy [65] is a generalization of the PARAFAC model, that adds some of the flexibility of Tucker2 model while retaining some of PARAFAC's uniqueness properties. The name PARATUCK2 indicates its similarity to both the PARAFAC and the Tucker2 model. The PARATUCK2 model performs decomposition of an arbitrary third-order tensor (see Figure 1.38(a)) $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ as follows

$$\boxed{\mathbf{Y}_q = \mathbf{A}\,\mathbf{D}_q^{(A)}\,\mathbf{R}\,\mathbf{D}_q^{(B)}\,\mathbf{B}^T + \mathbf{E}, \qquad (q = 1, 2, \ldots, Q),} \tag{1.182}$$

where $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{T \times P}$, $\mathbf{R} \in \mathbb{R}^{J \times P}$, $\mathbf{D}_q^{(A)} \in \mathbb{R}^{J \times J}$ and $\mathbf{D}_q^{(B)} \in \mathbb{R}^{P \times P}$ are diagonal matrices representing the q-th frontal slices of the tensors $\underline{\mathbf{D}}^{(A)} \in \mathbb{R}^{J \times J \times Q}$, and $\underline{\mathbf{D}}^{(B)} \in \mathbb{R}^{P \times P \times Q}$, respectively. In fact, tensor $\underline{\mathbf{D}}^{(A)}$ is formed by a matrix $\mathbf{U} \in \mathbb{R}^{J \times Q}$ whose columns are diagonals of the corresponding frontal slices and tensor $\underline{\mathbf{D}}^{(B)}$ is constructed from a matrix $\mathbf{V} \in \mathbb{R}^{P \times Q}$.

$$\mathbf{D}_q^{(A)} = \text{diag}(\boldsymbol{u}_q), \quad \mathbf{D}_q^{(B)} = \text{diag}(\boldsymbol{v}_q). \tag{1.183}$$

The j-th row $\underline{\boldsymbol{u}}_j$ (or $\underline{\boldsymbol{v}}_j$) gives the weights of participation for the corresponding component $\boldsymbol{a}_j$ in the factor $\mathbf{A}$ (or $\boldsymbol{b}_j$ in $\mathbf{B}$) with respect to the third dimension. The terms $\mathbf{D}_q^{(A)}\mathbf{R}\mathbf{D}_q^{(B)T}$ correspond to frontal slices of the core tensor $\underline{\mathbf{G}}$ of a Tucker2 model, but due to the restricted structure of the core tensor compared to Tucker2 uniqueness is retained. The core tensor $\underline{\mathbf{G}}$ can be described

as

$$\begin{aligned}\text{vec}\left(\mathbf{G}_q\right) &= \text{vec}\left(\mathbf{D}_q^{(A)}\mathbf{R}\mathbf{D}_q^{(B)}\right) &&= \text{vec}\left(\text{diag}(\boldsymbol{u}_q)\mathbf{R}\,\text{diag}(\boldsymbol{v}_q)^T\right)\\ &= \left(\text{diag}(\boldsymbol{v}_q)\otimes\text{diag}(\boldsymbol{u}_q)\right)\text{vec}\left(\mathbf{R}\right) &&= \text{diag}\left(\boldsymbol{v}_q\otimes\boldsymbol{u}_q\right)\text{vec}\left(\mathbf{R}\right)\\ &= \left(\boldsymbol{v}_q\otimes\boldsymbol{u}_q\right)\circledast\text{vec}\left(\mathbf{R}\right),\end{aligned} \tag{1.184}$$

or simply via frontal slices

$$\mathbf{G}_q = \left(\boldsymbol{u}_q\,\boldsymbol{v}_q^T\right)\circledast\mathbf{R}\,, \qquad (q=1,2,\ldots,Q). \tag{1.185}$$

This leads to the mode-3 matricization of the core tensor $\underline{\mathbf{G}}$ having following form

$$\begin{aligned}\mathbf{G}_{(3)} &= \left[\text{vec}\left(\mathbf{G}_1\right),\ldots,\text{vec}\left(\mathbf{G}_R\right)\right]^T\\ &= \left[\left(\boldsymbol{v}_1\otimes\boldsymbol{u}_1\right)\circledast\text{vec}\left(\mathbf{R}\right),\ldots,\left(\boldsymbol{v}_R\otimes\boldsymbol{u}_R\right)\circledast\text{vec}\left(\mathbf{R}\right)\right]^T\\ &= \left[\boldsymbol{v}_1\otimes\boldsymbol{u}_1,\ldots,\boldsymbol{v}_R\otimes\boldsymbol{u}_R\right]^T\circledast\left[\text{vec}\left(\mathbf{R}\right),\ldots,\text{vec}\left(\mathbf{R}\right)\right]^T\\ &= \left(\mathbf{V}\odot\mathbf{U}\right)^T\circledast\mathbf{T}_{(3)}\\ &= \mathbf{Z}_{(3)}\circledast\mathbf{T}_{(3)}\end{aligned} \tag{1.186}$$

or

$$\underline{\mathbf{G}} = \underline{\mathbf{Z}}\circledast\underline{\mathbf{T}}, \tag{1.187}$$

where $\underline{\mathbf{Z}}\in\mathbb{R}^{J\times P\times Q}$ is a rank-Q PARAFAC tensor represented by two factors $\mathbf{U}$ and $\mathbf{V}$ (the third factor for the mode-3 is identity matrix $\mathbf{I}_Q$), that is

$$\underline{\mathbf{Z}} = \underline{\mathbf{I}}\times_1\mathbf{U}\times_2\mathbf{V}, \tag{1.188}$$

and $\underline{\mathbf{T}}\in\mathbb{R}^{J\times P\times Q}$ is a tensor with identical frontal slices expressed by the matrix $\mathbf{R}$: $\mathbf{T}_q = \mathbf{R}$, $\forall q$. Equation (1.187) indicates that the core tensor $\underline{\mathbf{G}}$ is the Hadamard product of the PARAFAC tensor and a special (constrained) tensor with identity frontal slices. In other words the PARATUCK2 can be considered as the Tucker2 model in which the core tensor has special PARAFAC decomposition as illustrated in Figure 1.38(a). The PARATUCK2 model is well suited for a certain class of multi-way problems that involve interactions between factors.

Figure 1.38(b) illustrates a special form of PARATUCK2 called the three-way DEDICOM (inxDEcomposition into DIrectional COMponents) model [9], [10], [85]. In this case for a given symmetric third-order data tensor $\underline{\mathbf{Y}}\in\mathbb{R}^{I\times I\times Q}$ with frontal slices $\mathbf{Y}_q$ the simplified decomposition is:

$$\boxed{\mathbf{Y}_q = \mathbf{A}\,\mathbf{D}_q\,\mathbf{R}\,\mathbf{D}_q\,\mathbf{A}^T + \mathbf{E}, \qquad (q=1,2,\ldots,Q),} \tag{1.189}$$

where $\mathbf{A}\in\mathbb{R}^{I\times J}$ is a matrix of loadings, $\mathbf{D}_q$ is a diagonal matrix representing the q-th frontal slice of the tensor $\underline{\mathbf{D}}\in\mathbb{R}^{J\times J\times Q}$, and $\mathbf{R}\in\mathbb{R}^{J\times J}$ is an asymmetric matrix.

The three-way DEDICOM model can be considered as natural extension of the 2-way DEDICOM model [131]:

$$\boxed{\mathbf{Y} = \mathbf{A}\mathbf{R}\mathbf{A}^T + \mathbf{E},} \tag{1.190}$$

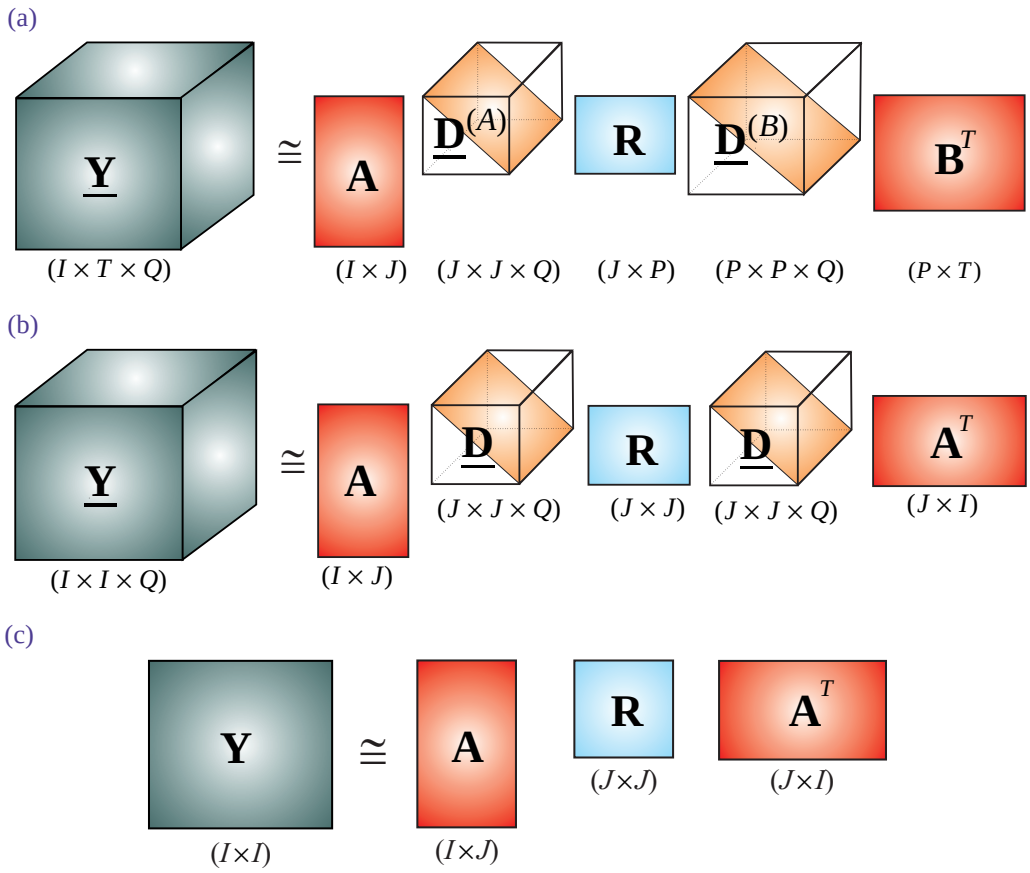


Fig. 1.38 (a) PARATUCK2 model performing decomposition of tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times T\times Q}$. (b) DEDICOM model for a symmetric third-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times I\times Q}$, (c) DEDICOM model for a square data matrix (usually, we assume that a matrix $\mathbf{A}$ is orthogonal).

where $\mathbf{Y} \in \mathbb{R}^{I\times I}$ is a given data matrix (generally asymmetric), and $\mathbf{E}$ is a matrix representing error not explained by the model. The goal is to estimate the best-fitting matrices: $\mathbf{A} \in \mathbb{R}^{I\times J}$ and $\mathbf{R} \in \mathbb{R}^{J\times J}$. To achieve this goal we usually perform the following optimization problem:

$$\min_{\mathbf{A},\mathbf{R}} \|\mathbf{Y} - \mathbf{A}\mathbf{R}\mathbf{A}^T\|_F^2. \tag{1.191}$$

The matrix $\mathbf{A} \in \mathbb{R}_+^{I\times J}$ comprises loadings or weights (with $J < I$), and the square matrix $\mathbf{R}$ is a matrix that represents the asymmetric relationships for the latent dimensions of $\mathbf{A}$.

This uniqueness of the three-way DEDICOM gives plausibility to the factors making them a valid description with a high confidence that they can explain more variance than convenient rotated 2-way solutions [85].

It should be noted that is some close relationships between PARAFAC2 and three-way PARATUCK2 and DEDICOM models. In fact the PARATUCK2 model can be derived from PARAFAC2 model by performing PARAFAC factorization of a tensor $\underline{\mathbf{A}}$. For the PARFAAC2 model the matrix $\mathbf{R} = \mathbf{H}$ is dense, symmetric matrix (usually positive definite), while the matrix $\mathbf{R}$ in

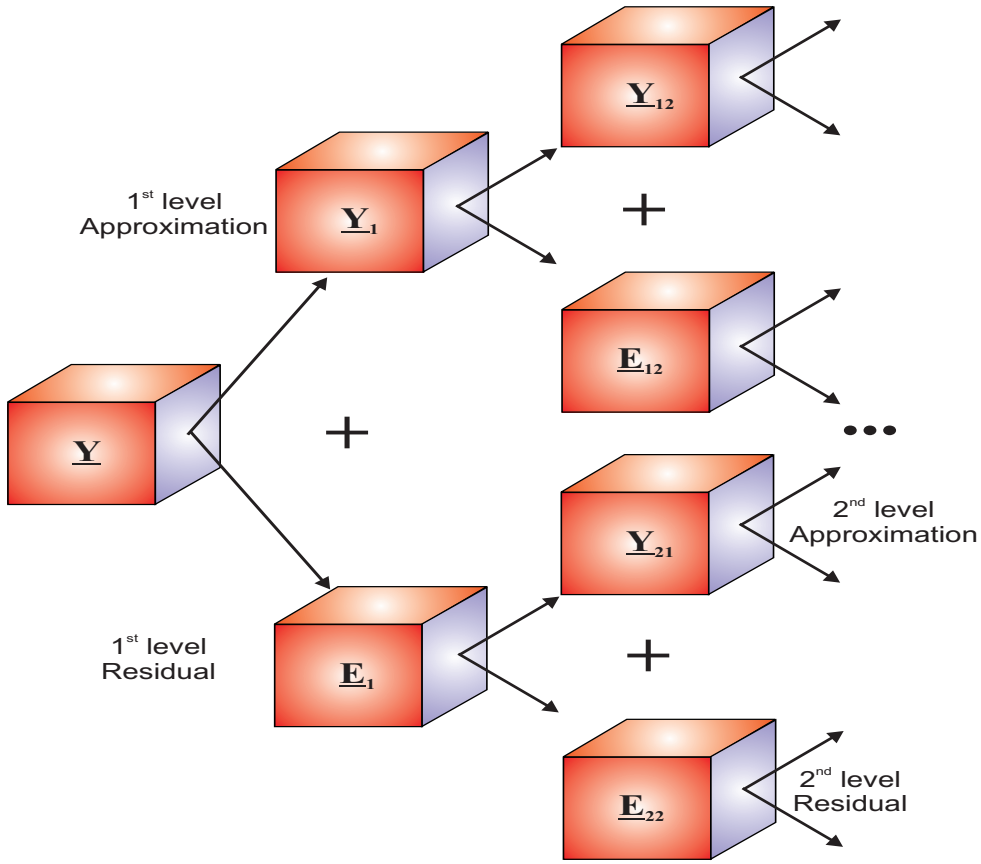


Fig. 1.39 Graphical illustration of hierarchical tensor decomposition.

the PARATUCK2 or DEDICOM model is a dense, generally asymmetric matrix that captures asymmetric relationships.

1.5.11 Hierarchical Tensor Decomposition

Recently, multi-linear models based on tensor approximation have received much attention as tools for denoising, reduction or compression as they have the potential to produce more compact representations of multi-dimensional data than traditional dimensionality reduction methods. We will exploit the aforementioned characteristics of visual 3D data and develop an analysis and a representation technique based on a hierarchical tensor-based transformation. In this technique, a multi-dimensional dataset is transformed into a hierarchy of signals to reflect multi-scale structures present in the multi-way data. The signal at each level of the hierarchy is further divided into a number of tensors with smaller spatial support to expose spatial inhomogeneity structures. To achieve a highly compact representation these smaller dimension tensors are further transformed and pruned using a tensor approximation technique [142].

It is interesting to note that the hierarchical scheme is very similar to the BCD model discussed in section (1.5.8). A source (data) tensor is expressed as a sum of multiple tensor decomposition models. However, the hierarchical technique is much more simpler than BCD model. For BCD model, all factors in all subtensors are simultaneously estimated, hence, constraints imposed on these factors such as nonnegative can be assured during the estimation process. However, BCD increases the complexity of the algorithms, especially for very large-scale data set. For a specified data, we can choose an acceptable trade-off between simplicity and accuracy.

1.6 DISCUSSION AND CONCLUSIONS

In this chapter we have presented a variety of different models, graphical and mathematical representations for NMF, NTF, NTD and the related matrix/tensor factorizations and decompositions. Our emphasis has been on the formulation of the problems and establishing relationships and links among different models. Each model usually provides a different interpretation of the data and may have different applications. Various equivalent representations have been presented which will serve as a basis for the development of learning algorithms throughout this book.

It has been highlighted that constrained models with nonnegativity and sparsity constraints for real-world data cannot provide a perfect fit to the observed data (i.e., they do not explain as much variance in the input data and may have larger residual errors) as compared to unconstrained factorization and decomposition models. They, however, often produce more meaningful physical interpretations. Although nonnegative factorizations/decompositions already exhibit some degree of sparsity, the combination of both constraints enables a precise control of sparsity.

Appendix 1.A. Uniqueness Conditions for Three-way Tensor Factorizations

The most attractive feature of the PARAFAC model is its uniqueness property. Kruskal [89] has proved that, for fixed error tensor $\mathbf{E}$, the vectors $\boldsymbol{a}_j$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_j$ of component matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are unique up to unavoidable scaling and permutation of columns,[21] provided that

$$k_A + k_B + k_C \geq J + 2, \tag{A.1}$$

where k_A, k_B, k_C denote the k-ranks of the component matrices. The k-rank of a matrix is the largest number k such that every subset of k columns of the matrix is linearly independent [130].

Kruskal's uniqueness condition was generalized to N-th order tensors with $N > 3$ by Sidropoulos and Bro [124]. A more accessible proof of the uniqueness condition (A.1) for the PARAFAC model was given by Stegeman and Sidiropoulos [129]. For the case where one of the component matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ has full column rank, weaker uniqueness conditions than (A.1) have been derived by Jiang and Sidiropoulos, De Lathauwer, and Stegeman (e.g., see [127], [130]). For example, if a component matrix $\mathbf{C} \in \mathbb{R}^{Q \times J}$ is full column rank, and $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{T \times J}$ have

[21] If a PARAFAC solution is unique up to these indeterminacies, it is called essentially unique. Two PARAFAC solutions that are identical up to the essential uniqueness indeterminacies will be called equivalent.

k-rank at least 2, then Kruskal's condition $k_A + k_B \geq J + 2$ implies uniqueness of a PARAFAC solution [127].

It should be noted that the above conditions are only valid for the unrestricted PARAFAC model. If we impose additional constraints such as nonnegativity, sparsity, orthogonality the conditions for uniqueness can be relaxed[22] and they can be different [128], [21]. For example, the NTF, NTF1, NTF2 and NTD models are unique (i.e., without rotational ambiguity) if component matrices and/or core tensor are sufficiently sparse.

Appendix 1.B. Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) with Sparsity and/or Nonnegativity Constraints

SVD and PCA are widely used tools, for example, in medical image analysis for dimension reduction, model building, and data understanding and exploration. They have applications in virtually all areas of science, machine learning, image processing, engineering, genetics, neurocomputing, chemistry, meteorology, computer networks, to name just a few, where large data sets are encountered. If $\mathbf{Y} \in \mathbb{R}^{I \times T}$ is a data matrix encoding T samples of I variables, with I being large, PCA aims at finding a few linear combinations of these variables, called the principal components, which point in orthogonal directions explaining as much of the variance in the data as possible. The purpose of principal component analysis PCA is to derive a relatively small number of uncorrelated linear combinations (principal components) of a set of random zero-mean variables while retaining as much of the information from the original variables as possible. Among the objectives of Principal Components Analysis are the following.

1. Dimensionality reduction.
2. Determination of linear combinations of variables.
3. Feature selection: the choosing of the most useful variables.
4. Visualization of multi-dimensional data.
5. Identification of underlying variables.
6. Identification of groups of objects or of outliers.

The success of PCA/SVD is due to two main optimal properties: Principal components sequentially capture the maximum variability of $\mathbf{Y}$ thus guaranteeing minimal information loss, and they are mutually uncorrelated. Despite the power and popularity of PCA, one key drawback is its lack of sparseness (i.e., factor loadings are linear combinations of all the input variables), yet sparse representations are generally desirable since they aid human understanding (e.g., with gene expression data), reduce computational costs and promote better generalization in learning algorithms. In other words, the standard principal components (PCs) can sometimes be difficult to interpret, because they are linear combinations of all the original variables. To facilitate better interpretation, sparse and/or nonnegative PCA estimate modified PCs with sparse and/or

[22] Moreover, by imposing the nonnegativity or orthogonality constraints the PARAFAC has an optimal solution, i.e., there is no risk for degenerate solutions [98]. Imposing nonnegativity constraints makes degenerative solutions impossible since no factor can counteract the effect of another factor and usually improves convergence since the search space is greatly reduced.

nonnegative eigenvectors, i.e. loadings with very few nonzero and possibly nonnegative entries. We use the connection of PCA with SVD of the data matrix and extract the PCs through solving a low rank matrix approximation problem. Regularization penalties are usually incorporated to the corresponding minimization problem to enforce sparsity and/or nonnegativity in PC loadings [143], [123].

Standard PCA is essentially the same technique as SVD but usually obtained using slightly different assumptions. Usually, in PCA we use normalized data with each variable centered and possibly normalized by the standard deviation.

B.1 STANDARD SVD AND PCA

At first let us consider basic properties of standard SVD and PCA. The SVD of a data matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$ assuming without loss of generality that $T > I$ leads to the following matrix factorization

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{j=1}^{J} \sigma_j \, \boldsymbol{u}_j \, \boldsymbol{v}_j^T, \tag{B.1}$$

where the matrix $\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_I] \in \mathbb{R}^{I \times I}$ contains the I left singular vectors, $\boldsymbol{\Sigma} \in \mathbb{R}_+^{I \times T}$ with nonnegative elements on the main diagonal representing the singular values σ_j and the matrix $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T] \in \mathbb{R}^{T \times T}$ represents the T right singular vectors called the loading factors. The nonnegative quantities σ_j, sorted as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_J > \sigma_{J+1} = \sigma_{J+2} = \cdots = \sigma_I = 0$ can be shown to be the square roots of the eigenvalues of the data covariance matrix $\mathbf{Y}\mathbf{Y}^T \in \mathbb{R}^{I \times I}$. The term $\boldsymbol{u}_j \boldsymbol{v}_j^T$ is an $I \times T$ rank-one matrix called often the j-th eigenimage of $\mathbf{Y}$. Orthogonality of the SVD expansion ensures that the left and right singular vectors are orthogonal, i.e., $\boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij}$ and $\boldsymbol{v}_i^T \boldsymbol{v}_j = \delta_{ij}$,with δ_{ij} the Kronecker function (or equivalently $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$). In many applications, it is most practical to work with the truncated form of the SVD where only the first $P < J$, (where J is a rank of $\mathbf{Y}$ with $J < I$) singular values are used so that

$$\mathbf{Y} \cong \mathbf{U}_P \, \boldsymbol{\Sigma}_P \, \mathbf{V}_P^T = \sum_{j=1}^{P} \sigma_j \, \boldsymbol{u}_j \, \boldsymbol{v}_j^T, \tag{B.2}$$

where $\mathbf{U}_P = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_P] \in \mathbb{R}^{I \times P}$, $\boldsymbol{\Sigma}_P = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_P\}$ and $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_P] \in \mathbb{R}^{T \times P}$. This is no longer an exact decomposition of the data matrix $\mathbf{Y}$, but according to the Eckart-Young theorem it is the best rank-P approximation in the least-squares sense and it is still unique (neglecting signs of vectors ambiguity) if the singular values are distinct.

Approximation of the matrix $\mathbf{Y}$ by a rank-one matrix $\sigma \boldsymbol{u}\boldsymbol{v}^T$ of two unknown vectors $\boldsymbol{u} = [u_1, u_2, \ldots, u_I]^T \in \mathbb{R}^I$ and $\boldsymbol{v} = [v_1, v_2, \ldots, v_T]^T \in \mathbb{R}^T$ normalized to unit length with a scaling constant term σ can be presented as follows:

$$\mathbf{Y} = \sigma \, \boldsymbol{u} \, \boldsymbol{v}^T + \mathbf{E}, \tag{B.3}$$

where $\mathbf{E} \in \mathbb{R}^{I \times T}$ is a matrix of the residual errors e_{it}. In order to compute the unknown vectors we minimize the squared Euclidean error as [99]

$$J_1 = \|\mathbf{E}\|_F^2 = \sum_{it} e_{it}^2 = \sum_{i=1}^{I} \sum_{t=1}^{T} (y_{it} - \sigma \, u_i \, v_t)^2. \tag{B.4}$$

The necessary conditions for minimization of (B.4) are obtained by equating gradients to zero:

$$\frac{\partial J_1}{\partial u_i} = -2\sigma \sum_{t=1}^{T} (y_{it} - \sigma\, u_i\, v_t) v_t = 0, \tag{B.5}$$

$$\frac{\partial J_1}{\partial v_t} = -2\sigma \sum_{i=1}^{I} (y_{it} - \sigma\, u_i\, v_t) u_i = 0, \tag{B.6}$$

These equations can be expressed as follows:

$$\sum_{t=1}^{T} y_{it}\, v_t = \sigma\, u_i \sum_{t=1}^{T} v_t^2, \qquad \sum_{i=1}^{I} y_{it}\, u_i = \sigma\, v_t \sum_{i=1}^{I} u_i^2. \tag{B.7}$$

Taking into account that the vectors are normalized to unit length, that is, $\boldsymbol{u}^T\boldsymbol{u} = \sum_{i=1}^{I} u_i^2 = 1$ and $\boldsymbol{v}^T\boldsymbol{v} = \sum_{t=1}^{T} v_t^2 = 1$, we can write the above equations in a compact matrix form as

$$\mathbf{Y}\,\boldsymbol{v} = \sigma\,\boldsymbol{u}, \qquad \mathbf{Y}^T\,\boldsymbol{u} = \sigma\,\boldsymbol{v} \tag{B.8}$$

or equivalently (by substituting one of Eqs. (B.8) into another)

$$\mathbf{Y}^T\mathbf{Y}\,\boldsymbol{v} = \sigma^2\boldsymbol{v}, \qquad \mathbf{Y}\mathbf{Y}^T\boldsymbol{u} = \sigma^2\boldsymbol{u}, \tag{B.9}$$

which are classical eigenvalue problems which estimate the maximum eigenvalue $\lambda_1 = \sigma_1^2 = \sigma_{max}^2$ with the corresponding eigen vectors $\boldsymbol{u}_1 = \boldsymbol{u}$ and $\boldsymbol{v}_1 = \boldsymbol{v}$. The solutions of these problems give the best first rank-one approximation of Eq. (B.1).

One of the most important results for nonnegative matrices is the following [75]:

Theorem B.1 (Perron-Frobenius) *For a square nonnegative matrix* $\mathbf{Y}$ *there exists a largest modulus eigenvalue of* $\mathbf{Y}$ *which is nonnegative and a corresponding nonnegative eigenvector.*

The eigenvector satisfying the Perron-Frobenius theorem is usually referred to as the Perron vector of a nonnegative matrix. For a rectangular nonnegative matrix, a similar result can be established for the largest singular value and its corresponding singular vector:

Theorem B.2 *The leading singular vectors:* $\boldsymbol{u}_1$, *and* $\boldsymbol{v}_1$ *corresponding to the largest singular value* $\sigma_{max} = \sigma_1$ *of a nonnegative matrix* $\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ *(with* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_I$*) are nonnegative.*

Based on this observation, it is straightforward to compute the best rank-one NMF approximation $\sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T$, this idea can be extended to approximate a higher-order NMF. If we compute the rank-one NMF and subtract it from the original matrix $\hat{\mathbf{Y}}_1 = \mathbf{Y} - \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T$, the input data matrix will no longer be nonnegative, however, all negative elements can be forced to be zero or are positive and the procedure can be repeated [12]. In order to estimate the next singular values and the corresponding singular vectors, we may apply a deflation approach, that is,

$$\mathbf{Y}_j = \mathbf{Y}_{j-1} - \sigma_j\boldsymbol{u}_j\boldsymbol{v}_j^T, \qquad (j = 1, 2, \ldots, J), \tag{B.10}$$

where $\mathbf{Y}_0 = \mathbf{Y}$. Solving the same optimization problem (B.4) for the residual matrix $\mathbf{Y}_j$ yields the set of consecutive singular values and corresponding singular vectors. Repeating reduction of the matrix yields the next set of the solution until the deflation matrix $\mathbf{Y}_{J+1}$ becomes the zero.

Using the property of the orthogonality of the eigenvectors, and the equality $\boldsymbol{u}^T\mathbf{Y}\boldsymbol{v} = \boldsymbol{v}^T\mathbf{Y}\boldsymbol{u} = \sigma$ we can estimate the precision of the matrix approximation with the first $P \leq J$ pairs of singular vectors [99]:

$$\begin{aligned} \|\mathbf{Y} - \sum_{j=1}^{P} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T\|_F^2 &= \sum_{i=1}^{I}\sum_{t=1}^{T}(y_{it} - \sum_{j=1}^{J} \sigma_j u_{ij} v_{jt})^2 \\ &= \|\mathbf{Y}\|_F^2 - \sum_{j=1}^{P} \sigma_j^2, \end{aligned} \tag{B.11}$$

and the residual error reduces exactly to zero with the number of singular values equal to the matrix rank, that is, for $P = J$. Thus, we can write for the rank-J matrix:

$$\|\mathbf{Y}\|_F^2 = \sum_{j=1}^{J} \sigma_j^2. \tag{B.12}$$

It is interesting to note that (taking into account that $\sigma u_i = \sum_{t=1}^{T} y_{it} v_t / \sum_{t=1}^{T} v_t^2$ (see Eqs. (B.7)) the cost function (B.4) can be expressed as [99]

$$\begin{aligned} J_1 = \|\mathbf{E}\|_F^2 &= \sum_{i=1}^{I}\sum_{t=1}^{T}(y_{it} - \sigma\, u_i\, v_t)^2 \\ &= \sum_{i=1}^{I}\sum_{t=1}^{T} y_{it}^2 - 2\sum_{i=1}^{I}(\sigma\, u_i)\sum_{t=1}^{T} y_{it} v_t) + \sum_{i=1}^{I}(\sigma\, u_i)^2 \sum_{t=1}^{T} v_t^2 \\ &= \sum_{i=1}^{I}\sum_{t=1}^{T} y_{it}^2 - \frac{\sum_{i=1}^{I}(\sum_{t=1}^{T} y_{it} v_t)^2}{\sum_{t=1}^{T} v_t^2}. \end{aligned} \tag{B.13}$$

In matrix notation the cost function can be written as

$$\|\mathbf{E}\|_F^2 = \|\mathbf{Y}\|_F^2 - \frac{\boldsymbol{v}^T\mathbf{Y}^T\mathbf{Y}\boldsymbol{v}}{\|\boldsymbol{v}\|_2^2} = \|\mathbf{Y}\|_F^2 - \sigma^2, \tag{B.14}$$

where the second term is called the Rayleigh quotient. The maximum value of the Rayleigh quotient is exactly equal to the maximum eigenvalue $\lambda_1 = \sigma_1^2$.

B.2 SPARSE PCA

For sparse PCA we may employ many alternative approaches [123]. One of the simplest and most efficient approaches is to apply minimization of the following cost function [123]:

$$J_\rho(\tilde{\boldsymbol{v}}) = \|\mathbf{Y} - \boldsymbol{u}\,\tilde{\boldsymbol{v}}^T\|_F^2 + \rho(\tilde{\boldsymbol{v}}), \tag{B.15}$$

where $\tilde{\boldsymbol{v}} = \sigma\boldsymbol{v}$, $\rho(\tilde{\boldsymbol{v}})$ is the additional penalty term which imposes sparsity. Typically, $\rho(|\tilde{\boldsymbol{v}}|) = 2\lambda\|\tilde{\boldsymbol{v}}\|_1$ or $\rho(\tilde{\boldsymbol{v}}) = \lambda^2\|\tilde{\boldsymbol{v}}\|_0$, where λ is the nonnegative coefficient that controls the degree of

sparsity.[23] The cost function can be evaluated in scalar form as follows

$$J_\rho(\tilde{\boldsymbol{v}}) = \sum_i \sum_t (y_{it} - u_i\, \tilde{v}_t)^2 + \sum_t \rho(\tilde{v}_t) = \sum_t \left(\sum_i (y_{it} - u_i\, \tilde{v}_t)^2 + \rho(v_t) \right)$$
$$= \sum_t \left(\sum_i y_{it}^2 - 2\, [\mathbf{Y}^T \boldsymbol{u}]_t\, \tilde{v}_t + \tilde{v}_t^2 + \rho(\tilde{v}_t) \right), \qquad (t = 1, 2, \ldots, T). \tag{B.16}$$

It is not difficult to see (see Chapter 4) that the optimal value of $\tilde{v}_t$ depends on the penalty term $\rho(\tilde{\boldsymbol{v}}_t)$, in particular for $\rho(\tilde{\boldsymbol{v}}_t) = 2\lambda \|\tilde{\boldsymbol{v}}\|_1$ we use soft shrinkage with threshold λ

$$\tilde{v}_t = P_\lambda^{(s)}(x) = \text{sign}(x)[|x| - \lambda]_+ \tag{B.17}$$

and for $\rho(\tilde{\boldsymbol{v}}_t) = \lambda^2 \|\tilde{\boldsymbol{v}}\|_0$ we use hard shrinkage projection

$$\tilde{v}_t = P_\lambda^{(h)}(x) = I(x > \lambda)\, x = \begin{cases} x, & \text{for} \quad |x| \geq \lambda; \\ 0, & \text{otherwise}, \end{cases} \tag{B.18}$$

where $x = [\mathbf{Y}^T \boldsymbol{u}]_t$. This leads to the following iterative algorithm proposed by Shen and Huang [123]

1. Initialize: Apply the regular SVD to data matrix $\mathbf{X}$ and estimate the maximum singular value $\sigma_1 = \sigma_{\max}$ and corresponding singular vectors $\boldsymbol{u} = \boldsymbol{u}_1$ and $\boldsymbol{v} = \boldsymbol{v}_1$, take $\tilde{\boldsymbol{v}}_1 = \sigma_1 \boldsymbol{v}_1$. This corresponds to the best rank-one approximation of $\mathbf{Y} = \mathbf{Y}_0$,

2. Update:

$$\tilde{\boldsymbol{v}}_1 \leftarrow P_\lambda \left(\mathbf{Y}^T \boldsymbol{u}_1 \right), \tag{B.19}$$
$$\tilde{\boldsymbol{u}}_1 \leftarrow (\mathbf{Y}\tilde{\boldsymbol{v}}_1) / \|\mathbf{Y}\tilde{\boldsymbol{v}}_1\|_2, \tag{B.20}$$

3. Repeat Step 2 until convergence,

4. Normalize the vector $\tilde{\boldsymbol{v}}_1$ as $\tilde{\boldsymbol{v}}_1 = \tilde{\boldsymbol{v}}_1 / \|\tilde{\boldsymbol{v}}_1\|_2$.

Note that for $\lambda = 0$ the nonlinear shrinkage function P_λ becomes a linear function and the above procedure simplifies to the well-known standard alternating least squares SVD algorithm. The subsequent pair $\{\boldsymbol{u}_2, \sigma_2 \boldsymbol{v}_2\}$ provides the best rank-one approximation of the corresponding residual matrix $\mathbf{Y}_1 = \mathbf{Y} - \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1$. In other words, subsequent sparse loading vectors $\boldsymbol{v}_j$ can be obtained sequentially via a deflation approach and a rank-one approximation of the residual data matrices $\mathbf{Y}_j$.

B.3 NONNEGATIVE PCA

In some applications it is necessary to incorporate both nonnegativity and/or sparseness constraints into PCA maintaining the maximal variance property of PCA and relaxing orthogonality constraints. The algorithm described in the previous section can be applied almost directly to

[23] We define the degree of sparsity of a PC as the number of zero elements in the corresponding loading vector $\boldsymbol{v}$.

nonnegative PCA by applying a suitable nonnegative shrinkage function. However, in such a case, it should be noted that the orthogonality among vectors $\boldsymbol{v}_j$ is completely lost. If we need to control the orthogonality constraint (to some extent) together with nonnegativity constraints, we may alternatively apply the following optimization problem [143]

$$\max_{\mathbf{V}\geq\mathbf{0}} \frac{1}{2}\|\mathbf{V}_J^T\mathbf{Y}\|_F^2 - \alpha_o\|\mathbf{I} - \mathbf{V}_J^T\mathbf{V}_J\|_F^2 - \alpha_s\|\mathbf{V}_J\|_1, \tag{B.21}$$

subject to nonnegativity constraints $\mathbf{V}_J \geq 0$, where $\|\mathbf{V}_J\|_1 = \mathbf{1}^T\mathbf{V}_J\mathbf{1}$. The nonnegative coefficients α_o and α_s control a level and tradeoff between sparsity and orthogonality.

Alternatively, we can use nonnegative loading parametrization, for example, exponential parametrization [99]

$$v_t = \exp(\gamma_t), \qquad \forall t, \tag{B.22}$$

where γ_t are the estimated parameters. To obtain the loading in the range from zero to one we can use multinomial parametrization

$$v_t = \frac{\exp(\gamma_t)}{\sum_t \exp(\gamma_t)}, \qquad (t = 1, 2, \ldots, T). \tag{B.23}$$

Appendix 1.C. Determining a True Number of Components

Determining the number of components J for the NMF/NTF models, or more generally determining the dimensions of a core tensor, J, R, P, for the Tucker models is very important since the approximately valid model is instrumental in discovering or capturing the underlying structure in the data.

There are several approximative and heuristic techniques for determining the number of components [20], [132], [42], [43], [40], [108], [71]. In an ideal noiseless case when the PARAFAC model is perfectly satisfied, we can apply a specific procedure for calculating the PARAFAC components for $J = 2, 3, \ldots$ until we reach the number of components for which the errors $\mathbf{E} = \mathbf{Y} - \hat{\mathbf{Y}}$ are zero. However, in practice, it is not possible to perfectly satisfy this model. Other proposed methods include: residual analysis, visual appearance of loadings, the number of iterations of the algorithm and core consistency. In this book, we mostly rely on the PCA approach and the core consistency diagnostic developed by Bro and Kiers [20] for finding the number of components and selecting an appropriate (PARAFAC or Tucker) model. The core consistency quantifies the resemblance between the Tucker3 core tensor and the PARAFAC core, which is a super-identity or a superdiagonal core tensor, or in other words, a vector of coefficients. This diagnostic tool suggests whether the PARAFAC model with the specified number of components is a valid model for the data. The core consistency above 90% is often used as an indicator of the trilinear structure of the data, and suggests that the PARAFAC model would be an appropriate model for the data. A core consistency value close to or lower than 50%, on the other hand, would indicate that the PARAFAC-like model is not appropriate. This diagnostic method has been commonly applied in the neuroscience-multi-way literature [57], [103], often together with other diagnostic tools, in order to determine the number of components.

An efficient way is to use the PCA/SVD approach, whereby for a three-way data set we first unfold the tensor as matrices $\mathbf{Y}_{(1)}$ and eventually compute covariance matrix $\mathbf{R}_y = (1/T)\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^T$.

Under the assumption that the power of the signals is larger than the power of the noise, the PCA enables us to divide observed (measured) sensor signals: $\boldsymbol{x}(t) = \boldsymbol{x}_s(t) + \boldsymbol{\nu}(t)$ into two subspaces: the *signal subspace* corresponding to principal components associated with the largest eigenvalues called the principal eigenvalues: $\lambda_1, \lambda_2, ..., \lambda_J$, $(I > J)$ and associated eigenvectors $\mathbf{V}_J = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_J]$ called the principal eigenvectors and the *noise subspace* corresponding to the minor components associated with the eigenvalues $\lambda_{J+1}, ..., \lambda_I$. The subspace spanned by the J first eigenvectors $\boldsymbol{v}_i$ can be considered as an approximation of the noiseless signal subspace. One important advantage of this approach is that it enables not only a reduction in the noise level, but also allows us to estimate the number of sources on the basis of distribution of the eigenvalues. However, a problem arising from this approach is how to correctly set or estimate the threshold which divides eigenvalues into the two subspaces, especially when the noise is large (i.e., the SNR is low). The covariance matrix of the observed data can be written as

$$\begin{aligned} \mathbf{R}_y &= E\{\boldsymbol{y}_t \boldsymbol{y}_t^T\} = [\mathbf{V}_{\mathcal{S}}, \mathbf{V}_{\mathcal{N}}] \begin{bmatrix} \boldsymbol{\Lambda}_{\mathcal{S}} & 0 \\ \mathbf{0} & \boldsymbol{\Lambda}_{\mathcal{N}} \end{bmatrix} [\mathbf{V}_{\mathcal{S}}, \mathbf{V}_{\mathcal{N}}]^T \\ &= \mathbf{V}_{\mathcal{S}} \boldsymbol{\Lambda}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^T + \mathbf{V}_{\mathcal{N}} \boldsymbol{\Lambda}_{\mathcal{N}} \mathbf{V}_{\mathcal{N}}^T, \end{aligned} \tag{C.1}$$

where $\mathbf{V}_{\mathcal{S}} \boldsymbol{\Lambda}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^T$ is a rank-J matrix, $\mathbf{V}_{\mathcal{S}} \in \mathbb{R}^{I \times J}$ contains the eigenvectors associated with J principal (signal+noise subspace) eigenvalues of $\boldsymbol{\Lambda}_{\mathcal{S}} = \text{diag}\{\lambda_1 \geq \lambda_2 \cdots \geq \lambda_J\}$ in a descending order. Similarly, the matrix $\mathbf{V}_{\mathcal{N}} \in \mathbb{R}^{I \times (I-J)}$ contains the $(I - J)$ (noise) eigenvectors that correspond to noise eigenvalues $\boldsymbol{\Lambda}_{\mathcal{N}} = \text{diag}\{\lambda_{J+1}, \ldots, \lambda_I\} = \sigma_e^2 \mathbf{I}_{I-J}$. This means that, theoretically, the $(I - J)$ smallest eigenvalues of $\mathbf{R}_y$ are equal to σ_e^2, so we can determine the dimension of the signal subspace from the multiplicity of the smallest eigenvalues under the assumption that the variance of the noise is relatively low and we have a perfect estimate of the covariance matrix. However, in practice, we estimate the sample covariance matrix from a limited number of samples and the smallest eigenvalues are usually different, so the determination of the dimension of the signal subspace is usually not an easy task.

A crucial problem is to decide how many principal components (PCs) should be retained. A simple ad hoc rule is to plot the eigenvalues in decreasing order and search for an elbow where the signal eigenvalues are on the left side and the noise eigenvalues on the right. Another simple technique is to compute the cumulative percentage of the total variation explained by the PCs and retain the number of PCs that represent, say 95% of the total variation. Such techniques often work well in practice, but their disadvantage is that they need a subjective decision from the user [137]. Many sophisticated methods have been introduced such as a Bayesian model selection method, which is referred to as the Laplace method. It is based on computing the evidence for the data and requires integrating out all the model parameters. Another method is the BIC (Bayesian Information Criterion) method which can be thought of as an approximation of the Laplace criterion.

A simple heuristic method proposed by He and Cichocki [71], [108] computes the Gap (smoothness) index defined as

$$GAP(p) = \frac{\text{var}\left[\{\tilde{\lambda}_i\}_{i=p+1}^{I-1}\right]}{\text{var}\left[\{\tilde{\lambda}_i\}_{i=p}^{I-1}\right]} = \frac{\hat{\sigma}_{p+1}^2}{\hat{\sigma}_p^2}, \qquad (p = 1, 2, \ldots, I-2), \tag{C.2}$$

where $\tilde{\lambda}_i = \lambda_i - \lambda_{i+1}$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_I > 0$ are eigenvalues of the covariance matrix for the noisy data and the sample variance is computed as follows

$$\hat{\sigma}_p^2 = \text{var}\left[\{\tilde{\lambda}_i\}_{i=p}^{I-1}\right] = \frac{1}{I-p}\sum_{i=p}^{I-1}\left(\tilde{\lambda}_i - \frac{1}{I-p}\sum_{i=p}^{I-1}\tilde{\lambda}_i\right)^2. \tag{C.3}$$

The number of components (for each mode) is selected using the following criterion:

$$\widehat{J} = \arg\min_{p=1,2,\ldots,I-3} GAP(p). \tag{C.4}$$

Recently, Ulfarsson and Solo [137] proposed a method called SURE (Steins Unbiased Risk Estimator) which allows the number of PC components to estimate reliably.

The Laplace, BIC and SURE methods are based on the following considerations [137]. The PCA model is given by

$$\boldsymbol{y}_t = \mathbf{A}\boldsymbol{x}_t + \boldsymbol{e}_t + \bar{\boldsymbol{y}} = \boldsymbol{\mu}_t + \boldsymbol{e}_t, \tag{C.5}$$

where $\bar{\boldsymbol{y}} = (1/T)\sum_{t=1}^{T}\boldsymbol{y}_t$ and $\boldsymbol{\mu}_t = \mathbf{A}\boldsymbol{x}_t + \bar{\boldsymbol{y}}$. The maximum-likelihood estimate (MLE) of PCA is given by

$$\widehat{\mathbf{A}} = \mathbf{V}_r(\boldsymbol{\Lambda}_r - \widehat{\sigma}_r^2\mathbf{I}_r)^{1/2}\mathbf{Q}, \qquad \widehat{\sigma}_r^2 = \sum_{j=r+1}^{I}\lambda_j, \tag{C.6}$$

where $\mathbf{Q} \in \mathbb{R}^{r\times r}$ is an arbitrary orthogonal rotation matrix, $\boldsymbol{\Lambda}_r = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ is a diagonal matrix with ordered eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_r$ and $\mathbf{V}_r = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r]$ is a matrix of corresponding eigenvectors of the data covariance matrix:

$$\mathbf{R}_y = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{y}_t - \bar{\boldsymbol{y}})\,(\boldsymbol{y}_t - \bar{\boldsymbol{y}})^T = \mathbf{V}_r\boldsymbol{\Lambda}_r\mathbf{V}_r^T. \tag{C.7}$$

Hence, the estimate for $\boldsymbol{\mu}_t = \mathbf{A}\boldsymbol{x}_t + \bar{\boldsymbol{y}}$ is given by

$$\widehat{\boldsymbol{\mu}}_t = \bar{\boldsymbol{y}} + \sum_{j=1}^{r}\boldsymbol{v}_j\frac{\lambda_j - \widehat{\sigma}_r^2}{\lambda_j}\boldsymbol{v}_j^T(\boldsymbol{y}_t - \bar{\boldsymbol{y}}). \tag{C.8}$$

Ideally, we would like to choose $r = J$ that minimizes the risk function $R_r = E(\|\boldsymbol{\mu}_t - \widehat{\boldsymbol{\mu}}_t\|_2^2)$ which is estimated by the SURE formula

$$\widehat{R}_r = \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{e}_t\|_2^2 + \frac{2\sigma_e^2}{T}\sum_{t=1}^{T}\text{tr}\left(\frac{\partial\widehat{\boldsymbol{\mu}}_t}{\partial\boldsymbol{y}_t^T}\right) - I\sigma_e^2. \tag{C.9}$$

In practice, the SURE algorithm chooses the number of components to minimize the SURE formula, that is,

$$\widehat{J} = \arg\min_{r=1,2,\ldots,I}\widehat{R}_r, \tag{C.10}$$

where the SURE formula is given by [137]

$$\widehat{R}_r = (I-r)\widehat{\sigma}_r^2 + \widehat{\sigma}_r^4 \sum_{j=1}^{r} \frac{1}{\lambda_j} + 2\sigma_e^2(1-\frac{1}{T})r$$
$$-2\sigma_e^2\widehat{\sigma}_r^2(1-\frac{1}{T})\sum_{j=1}^{r}\frac{1}{\lambda_j} + \frac{4(1-1/T)\sigma_e^2\widehat{\sigma}_k^2}{T}\sum_{j=1}^{r}\frac{1}{\lambda_j} + C_r, \tag{C.11}$$

$$C_r = \frac{4(1-1/T)\sigma_e^2}{T}\sum_{j=1}^{r}\sum_{i=r+1}^{I}\frac{\lambda_j-\widehat{\sigma}_r^2}{\lambda_j-\lambda_i} + \frac{2(1-1/T)\sigma_e^2}{T}r(r+1)$$
$$-\frac{2(1-1/T)\sigma_e^2}{T}(I-1)\sum_{j=1}^{r}\left(1-\frac{\widehat{\sigma}_r^2}{\lambda_j}\right), \tag{C.12}$$

$$\sigma^2 = \frac{\text{median}(\lambda_{r+1},\lambda_{r+2},\ldots,\lambda_I)}{F_{\gamma,1}^{-1}(\frac{1}{2})}, \tag{C.13}$$

and $\gamma = T/I$, $F_{\gamma,1}$ denotes the Marchenko-Pastur (MP) distribution function with parameter "γ".

Our extensive numerical experiments indicate that the Laplace method usually outperforms the BIC method while the SURE method can achieve significantly better performance than the Laplace method for NMF, NTF and Tucker models.

Appendix 1.D. Nonnegative Rank Factorization Using Wedderborn Theorem – Estimation of the Number of Components

Nonnegative Rank Factorization (NRF) is defined as exact bilinear decomposition:

$$\mathbf{Y} = \sum_{j=1}^{J} \boldsymbol{a}_j\, \boldsymbol{b}_j^T, \tag{D.1}$$

where $\mathbf{Y} \in \mathbb{R}^{I\times T}$, $\boldsymbol{a}_j \in \mathbb{R}_+^I$ and $\boldsymbol{b}_j \in \mathbb{R}_+^T$.

In order to perform such decomposition (if it exists), we begin with a simple, but far reaching, result first proved by Wedderburn.

Theorem D.3 *Suppose $\mathbf{Y} \in \mathbb{R}^{I\times T}$, $\boldsymbol{a} \in \mathbb{R}^I$ and $\boldsymbol{b} \in \mathbb{R}^T$. Then*

$$rank\left(\mathbf{Y} - \sigma^{-1}\mathbf{Y}\,\boldsymbol{b}\,\boldsymbol{a}^T\,\mathbf{Y}\right) = rank\,(\mathbf{Y}) - 1, \tag{D.2}$$

if and only if $\sigma = \boldsymbol{a}^T\,\mathbf{Y}\,\boldsymbol{b} \neq 0$.

Usually, the Wedderburn theorem is formulated in more general form:

Theorem D.4 *Suppose $\mathbf{Y}_1 \in \mathbb{R}^{I\times T}$, $\boldsymbol{a} \in \mathbb{R}^I$ and $\boldsymbol{b} \in \mathbb{R}^T$. Then the matrix*

$$\mathbf{Y}_2 = \mathbf{Y}_1 - \sigma^{-1}\boldsymbol{a}\boldsymbol{b}^T \tag{D.3}$$

satisfies the rank subtractivity $rank(\mathbf{Y}_2) = rank(\mathbf{Y}_1) - 1$ if and only if there are vectors $\boldsymbol{x} \in \mathbb{R}^T$ and $\boldsymbol{y} \in \mathbb{R}^I$ such that

$$\boldsymbol{a} = \mathbf{Y}_1\boldsymbol{x}, \quad \boldsymbol{b} = \mathbf{Y}_1^T\,\boldsymbol{y}, \quad \sigma = \boldsymbol{y}^T\,\mathbf{Y}_1\,\boldsymbol{x}\,. \tag{D.4}$$

The Wedderburn rank-one reduction formula (D.4) has led to a general matrix factorization process (e.g., the LDU and QR decompositions, the Lanczos algorithm and the SVD are special cases) [53].

The basic idea for the NRF is that, starting with $\mathbf{Y}_1 = \mathbf{Y}$, then so long as $\mathbf{Y}_j$ are nonnegative, we can repeatedly apply the Wedderburn formula to generate a sequence $\{\mathbf{Y}_j\}$ of matrices by defining

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j - (\boldsymbol{y}_j^T \mathbf{Y}_j \boldsymbol{x}_j)^{-1} \mathbf{Y}_j \boldsymbol{x}_j \boldsymbol{y}_j^T \mathbf{Y}_j, \qquad (j = 1, 2, \ldots, J) \tag{D.5}$$

for properly chosen nonnegative vectors satisfying $\boldsymbol{y}_j^T \mathbf{Y}_j \boldsymbol{x}_j \neq 0$. We continue such extraction till the residual matrix $\mathbf{Y}_j$ becomes zero matrix or with negative elements. Without loss of generality we assume that $\boldsymbol{y}_j^T \mathbf{Y}_j \boldsymbol{x}_j = 1$ and consider the following constrained optimization problem [53]

$$\begin{aligned} &\max_{\mathbf{x}_j \in \mathbb{R}_+^I,\, \mathbf{y}_j \in \mathbb{R}_+^T} \min \left(\mathbf{Y}_j - \mathbf{Y}_j \boldsymbol{x}_j \boldsymbol{y}_j^T \mathbf{Y}_j \right) \\ &\text{s.t.} \quad \mathbf{Y}_j \boldsymbol{x}_j \geq 0, \quad \boldsymbol{y}_j^T \mathbf{Y}_j \geq 0, \quad \boldsymbol{y}_j^T \mathbf{Y}_j \boldsymbol{a}_j = 1. \end{aligned} \tag{D.6}$$

There are some available routines for solving the above optimization problem, especially, the MATLAB routine "fminmax" implements a sequential quadratic programming method. It should be noted that this method does not guarantee finding a global solution, but only a suboptimal local solution. On the basis of this idea Dong, Lin and Chu developed an algorithm for the NRF using the Wedderburn rank reduction formula [53].

- Given a nonnegative data matrix $\mathbf{Y}$ and a small threshold of machine $\varepsilon > 0$ set $j = 1$ and $\mathbf{Y}_1 = \mathbf{Y}$.
- Step 1. If $\|\mathbf{Y}_j\| \geq \varepsilon$, go to Step 2. Otherwise, retrieve the following information and stop.

  1. $\text{rank}(\mathbf{Y}) = \text{rank}_+(\mathbf{Y}) = j - 1$.

  2. The NRF of $\mathbf{Y}$ is approximately given by the summation $\mathbf{Y} \cong \sum_{k=1}^{j-1} \mathbf{Y}_k \boldsymbol{x}_k \boldsymbol{y}_k^T \mathbf{Y}_k$ with an error less than ε.
- Step 2. Randomly select a feasible initial value $(\boldsymbol{x}_j^{(0)}, \boldsymbol{y}_j^{(0)})$ satisfying the nonnegativity constraints.
- Step 3. Solve the maximin problem (D.6).
- Step 4. If the objective value at the local maximizer $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ is negative, go to Step 5. Otherwise, do update as follows and go to Step 1.

  1. Define $\mathbf{Y}_{j+1} := \mathbf{Y}_j - \mathbf{Y}_j \boldsymbol{x}_j \boldsymbol{y}_j^T \mathbf{Y}_j$.

  2. Set $j = j + 1$.
- Step 5. Since the algorithm may get stuck at a local minimum try to restart Steps 2 and 3 multiple times. If it is decided within reasonable trials that no initial value can result in nonnegative values, report with caution that the matrix $\mathbf{Y}$ does not have an NRF and stop [53].

References

1. E. Acar, T.G. Kolda, and D.M. Dunlavy. An optimization approach for fitting canonical tensor decompositions. Technical Report SAND2009-0857, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, February 2009.

2. E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21:6–20, 2009.

3. R. Albright, J. Cox, D. Duling, A. N. Langville, and C. D. Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. Technical report, NCSU Technical Report Math 81706, 2006.

4. S. Amari and A. Cichocki. Adaptive blind signal processing - neural network approaches. *Proceedings of the IEEE*, 86:1186–1187, 1998.

5. A.H. Andersen and W.S. Rayens. Structure-seeking multilinear methods for the analysis of fMRI data. *NeuroImage*, 22:728–739, 2004.

6. C.A. Andersson and R. Bro. The N-way toolbox for MATLAB. *Chemometrics Intell. Lab. Systems*, 52(1):1–4, 2000.

7. C.J. Appellof and E.R. Davidson. Strategies for analyzing data from video fluoromatric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53:2053–2056, 1981.

8. L. Badea. Extracting gene expression profiles common to Colon and Pancreatic Adenocarcinoma using simultaneous nonnegative matrix factorization. In *Proceedings of Pacific Symposium on Biocomputing PSB-2008*, pages 267–278, World Scientific, 2008.

9. B.W. Bader, R.A. Harshman, and T.G. Kolda. Pattern analysis of directed graphs using DEDICOM: An application to Enron email. Technical Report SAND2006-7744, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2006.

10. B.W. Bader, R.A. Harshman, and T.G. Kolda. Temporal analysis of semantic graphs using ASALSAN. In *ICDM 2007: Proceedings of the 7th IEEE International Conference on Data Mining*, pages 33–42, October 2007.

11. M. Berry, M. Browne, A. Langville, P. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.

12. M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. In *ICML-2008*, Helsinki, July 2008.

13. J. Bobin, J.L. Starck, J. Fadili, Y. Moudden, and D.L. Donoho. Morphological component analysis: An adaptive thresholding strategy. *IEEE Transactions on Image Processing*, 16(11):2675–2681, 2007.

14. C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41:1350–1362, 2008.

15. C. Boutsidis, M.W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proc. 20-th Annual SODA*, pages 968–977, USA, 2009.

16. R. Bro. PARAFAC. Tutorial and applications. In *Special Issue 2nd Internet Conf. in Chemometrics (INCINC'96)*, volume 38, pages 149–171. Chemom. Intell. Lab. Syst, 1997.

17. R. Bro. *Multi-way Analysis in the Food Industry - Models, Algorithms, and Applications*. PhD thesis, University of Amsterdam, Holland, 1998.

18. R. Bro. Review on multiway analysis in chemistry 2000–2005. *Critical Reviews in Analytical Chemistry*, 36:279–293, 2006.

19. R. Bro and C.A. Andersson. Improving the speed of multiway algorithms - Part II: Compression. *Chemometrics and Intelligent Laboratory Systems*, 42:105–113, 1998.

20. R. Bro and H.A.L. Kiers. A new efficient method for determining the number of components in PARAFAC models. *Journal of Chemometrics*, 17(5):274–286, 2003.

21. A.M. Bruckstein, M. Elad, and M. Zibulevsky. A non-negative and sparse enough solution of an underdetermined linear system of equations is unique. *IEEE Transactions on Information Theory*, 54:4813–4820, 2008.

22. C. Caiafa and A. Cichocki. CUR decomposition with optimal selection of rows and columns. *(submitted)*, 2009.

23. C. Caiafa and A. Cichocki. Slice Oriented Decomposition: A new tensor representation for 3-way data. *(submitted to Journal of Signal Processing)*, 2009.

24. J.D. Carroll and J.J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart–Young decomposition. *Psychometrika*, 35(3):283–319, 1970.

25. J.D. Carroll, G. De Soete, and S. Pruzansky. Fitting of the latent class model via iteratively reweighted least squares CANDECOMP with nonnegativity constraints. In R. Coppi and S. Bolasco, editors, *Multiway data analysis.*, pages 463–472. Elsevier, Amsterdam, The Netherlands, 1989.

26. A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing*. John Wiley & Sons Ltd, New York, 2003.

27. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer, LNAI-4029*, 4029:548–562, 2006.

28. A. Cichocki, S.C. Douglas, and S. Amari. Robust techniques for independent component analysis (ICA) with noisy data. *Neurocomputing*, 23(1–3):113–129, November 1998.

29. A. Cichocki and P. Georgiev. Blind source separation algorithms with matrix constraints. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(1):522–531, January 2003.

30. A. Cichocki and A.H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE (invited paper)*, March 2009.

31. A. Cichocki, A.H. Phan, and C. Caiafa. Flexible HALS algorithms for sparse non-negative matrix/tensor factorization. In *Proc. of 18-th IEEE workshops on Machine Learning for Signal Processing*, Cancun, Mexico, 16–19, October 2008.

32. A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. John Wiley & Sons ltd, New York, 1994.

33. A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization. *Electronics Letters*, 42(16):947–948, 2006.

34. A. Cichocki and R. Zdunek. NTFLAB for Signal Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

35. A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: Family of new algorithms. *Springer, LNCS-3889*, 3889:32–39, 2006.

36. A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2006*, volume 5, pages 621–624, Toulouse, France, May 14–19 2006.

37. A. Cichocki, R. Zdunek, and S.-I. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. *Springer, Lecture Notes on Computer Science, LNCS-4666*, pages 169–176, 2007.

38. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using Alpha and Beta divergencies. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume III, pages 1393–1396, Honolulu, Hawaii, USA, April 15–20 2007.

39. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari. Novel multi-layer nonnegative tensor factorization with sparsity constraints. *Springer, LNCS-4432*, 4432:271–280, April 11–14 2007.

40. J.P.C.L. Da Costa, M. Haardt, and F. Roemer. Robust methods based on the HOSVD for estimating the model order in PARAFAC models. In *Proc. 5-th IEEE Sensor Array and Multich. Sig. Proc. Workshop (SAM 2008)*, pages 510–514, Darmstadt, Germany, July 2008.

41. S. Cruces, A. Cichocki, and L. De Lathauwer. Thin QR and SVD factorizations for simultaneous blind signal extraction. In *Proc. of the European Signal Processing Conference (EUSIPCO*, pages 217–220. Vienna, Austria, 2004.

42. E. Cuelemans and H.A.L. Kiers. Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method. *British Journal of Mathematical and Statistical Psychology*, 59:133–150, 2006.

43. E. Cuelemans and H.A.L. Kiers. Discriminating between strong and weak structures in three-mode principal component analysis. *British Journal of Mathematical and Statistical Psychology*, page (in print), 2008.

44. L. De Lathauwer. Parallel factor analysis by means of simultaneous matrix decompositions. In *Proceedings of the First IEEE International Workshop on Computational Advances in Multi-sensor Adaptive Processing (CAMSAP 2005)*, pages 125–128, Puerto Vallarta, Jalisco State, Mexico, 2005.

45. L. De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 28:642–666, 2006.

46. L. De Lathauwer. Decompositions of a higher-order tensor in block terms – Part I: Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 30(3):1022–1032, 2008. Special Issue on Tensor Decompositions and Applications.

47. L. De Lathauwer. Decompositions of a higher-order tensor in block terms – Part II: Definitions and uniqueness. *SIAM Journal of Matrix Analysis and Applications*, 30(3):1033–1066, 2008. Special Issue on Tensor Decompositions and Applications.

48. L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 30(3):1067–1083, 2008. Special Issue Tensor Decompositions and Applications.

49. I. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Neural Information Proc. Systems*, pages 283–290, Vancouver, Canada, December 2005.

50. C. Ding, X. He, and H.D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. SIAM International Conference on Data Mining (SDM'05)*, pages 606–610, 2005.

51. C. Ding, T. Li, W. Peng, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. Technical Report 60428, Lawrence Berkeley National Laboratory, November 2006.

52. C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *KDD06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 126–135, New York, NY, USA, 2006. ACM Press.

53. B. Dong, M.M. Lin, and M.T. Chu. Nonnegative rank factorization via rank reduction. *http://www4.ncsu.edu/ mtchu/Research/Papers/Readme.html*, page (submitted), 2008.

54. D. Donoho and V. Stodden. When does nonnegative matrix factorization give a correct decomposition into parts? In *Neural Information Processing Systems*, volume 16. MIT press, 2003.

55. P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.

56. J. Eggert, H. Wersing, and E. Koerner. Transformation-invariant representation and NMF. USA, 2004. Proceedings of International Joint Conference Neural Networks.

57. F. Estienne, N. Matthijs, D.L. Massart, P. Ricoux, and D. Leibovici. Multi-way modeling of high-dimensionality electroencephalographic data. *Chemometrics and Intelligent Laboratory Systems*, 58(1):59–72, 2001.

58. N. Gillis and F. Glineur. Nonnegative matrix factorization and underapproximation. In *9th International Symposium on Iterative Methods in Scientific Computing*, Lille, France, 2008. http://www.core.ucl.ac.be/ ngillis/.

59. N. Gillis and F. Glineur. Nonnegative factorization and maximum edge biclique problem. In *submitted*, 2009. http://www.uclouvain.be/en-44508.html.

60. S.A. Goreinov, I.V. Oseledets, D.V. Savostyanov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. How to find a good submatrix. (submitted), 2009.

61. S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Alegebra and Applications*, 261:1–21, 1997.

62. R.A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

63. R.A. Harshman. PARAFAC2: Mathematical and technical notes. *UCLA Working Papers in Phonetics*, 22:30–44, 1972.

64. R.A. Harshman, S. Hong, and M.E. Lundy. Shifted factor analysis - Part I: Models and properties. *Journal of Chemometrics*, 17(7):363–378, 2003.

65. R.A. Harshman and M.E. Lundy. *Research Methods for Multimode Data Analysis*. Praeger, New York, USA, 1984.

66. M. Hasan, F. Pellacini, and K. Bala. Matrix row-column sampling for the many-light problem. In *SIGGRAPH*, page http://www.cs.cornell.edu/ mhasan/, 2007.

67. M. Hasan, E. Velazquez-Armendariz, F. Pellacini, and K. Bala. Tensor clustering for rendering many-light animations. In *Eurographics Symposium on Rendering*, volume 27, page http://www.cs.cornell.edu/ mhasan/, 2008.

68. T. Hazan, S. Polak, and A. Shashua. Sparse image coding using a 3D non-negative tensor factorization. In *Proc. Int. Conference on Computer Vision (ICCV)*, pages 50–57, 2005.

69. Z. He and A. Cichocki. K-EVD clustering and its applications to sparse component analysis. *Lecture Notes in Computer Science*, 3889:438–445, 2006.

70. Z. He and A. Cichocki. An efficient K-hyperplane clustering algorithm and its application to sparse component analysis. *Lecture Notes in Computer Science*, 4492:1032–1041, 2007.

71. Z. He and A. Cichocki. Efficient method for estimating the dimension of Tucker3 model. *Journal of Multivariate Analysis*, page (submitted), 2009.

72. Z. He, S. Xie, L. Zhang, and A. Cichocki. A note on Lewicki-Sejnowski gradient for learning overcomplete representations. *Neural Computation*, 20(3):636–643, 2008.

73. M. Heiler and C. Schnoerr. Controlling sparseness in non-negative tensor factorization. *Springer LNCS*, 3951:56–67, 2006.

74. F.L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7:39–79, 1927.

75. N.-D. Ho, P. Van Dooren, and V.D. Blondel. Descent methods for nonnegative matrix factorization. *Numerical Linear Algebra in Signals, Systems and Control*, 2008.

76. P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

77. A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons Ltd, New York, 2001.

78. H.A.L. Kiers, J.M.F. Ten Berg, and R. Bro. PARAFAC2 Part I. A direct fitting algorithm for the PARAFAC2 model. *Journal of Chemometrics*, 13(3–4):275–294, 1999.

79. H.A.L. Kiers and A.K. Smilde. Constrained three-mode factor analysis as a tool for parameter estimation with second-order instrumental data. *Journal of Chemometrics*, 12:125–147, 1998.

80. H. Kim, L. Eldén, and H. Park. Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares. In *Proceedings of IEEE 7th International Conference on Bioinformatics and Bioengineering (BIBE07)*, volume II, pages 1147–1151, 2007.

81. M. Kim and S. Choi. Monaural music source separation: Nonnegativity, sparseness, and shift-invariance. pages 617–624, USA, Charleston, 2006. Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation (Springer LNCS 3889).

82. Y.-D. Kim and S. Choi. A method of initialization for nonnegative matrix factorization. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume II, pages 537–540, Honolulu, Hawaii, USA, April 15–20 2007.

83. Y.-D. Kim and S. Choi. Nonnegative Tucker Decomposition. In *Proc. of Conf. Computer Vision and Pattern Recognition (CVPR-2007)*, Minneapolis, Minnesota, June 2007.

84. Y.-D. Kim, A. Cichocki, and S. Choi. Nonnegative Tucker Decomposition with Alpha Divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2008*, Nevada, USA, 2008.

85. T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):(in print), September 2009.

86. T.G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM 2008: Proceedings of the 8th IEEE International Conference on Data Mining*, December 2008.

87. K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T.-W. Lee, and T.J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, 2003.

88. P.M. Kroonenberg. *Applied Multiway Data Analysis*. John Wiley & Sons Ltd, New York, 2008.

89. J.B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra Appl.*, 18:95–138, 1977.

90. J.B. Kruskal. Rank, decomposition, and uniqueness for 3-way and N-way arrays. In R. Coppi and S. Bolasco, editors, *Multiway data analysis.*, pages 8–18. Elsevier, Amsterdam, The Netherlands, 1989.

91. B. Kulis, M.A. Sustik, and I.S. Dhillon. Learning low-rank kernel matrices. In *Proc. of the Twenty-third International Conference on Machine Learning (ICML06)*, pages 505–512, July 2006.

92. A. N. Langville, C. D. Meyer, and R. Albright. Initializations for the nonnegative matrix factorization. In *Proc. of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, August 20–23 2006.

93. D.D. Lee and H.S. Seung. Learning of the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

94. D.D. Lee and H.S. Seung. *Algorithms for Nonnegative Matrix Factorization*, volume 13. MIT Press, 2001.

95. Y. Li, S. Amari, A. Cichocki, D. Ho, and S. Xie. Underdetermined blind source separation based on sparse representation. *IEEE Transactions on Signal Processing*, 54:423–437, 2006.

96. Y. Li, A. Cichocki, and S. Amari. Blind estimation of channel parameters and source components for EEG signals: A sparse factorization approach. *IEEE Transactions on Neural Networks*, 17:419–431, 2006.

97. Y. Li, Y. Du, and X. Lin. Kernel-based multifactor analysis for image synthesis and recognition. In *Proc. of 10th IEEE Conference on Computer Vision (ICCV05)*, volume 1, pages 114–119, 2005.

98. L.-H. Lim and P. Comon. Nonnegative approximations of nonnegative tensors. *Journal of Chemometrics*, page (in print), 2009.

99. S. Lipovetsky. PCA and SVD with nonnegative loadings. *Pattern Recognition*, 99(9):(in print), 2009.

100. M.W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. National Academy of Science*, 106:697–702, 2009.

101. M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions and data applications. *SIAM Journal on Matrix Analysis and Applications*, 30:957–987, 2008.

102. E. Martnez-Montes, J.M. Snchez-Bornot, and P.A. Valds-Sosa. Penalized PARAFAC analysis of spontaneous EEG recordings. *Statistica Sinica*, 18:1449–1464, 2008.

103. F. Miwakeichi, E. Martnez-Montes, P. Valds-Sosa, N. Nishiyama, H. Mizuhara, and Y. Yamaguchi. Decomposing EEG data into space−time−frequency components using parallel factor analysis. *NeuroImage*, 22(3):1035–1045, 2004.

104. J. Möck. Topographic components model for event-related potentials and some biophysical considerations. *IEEE Transactions on Biomedical Engineering*, 35:482–484, 1988.

105. M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related EEG. *NeuroImage*, 29(3):938–947, 2006.

106. M. Mørup, L.K. Hansen, and S.M. Arnfred. Algorithms for Sparse Nonnegative Tucker Decompositions. *Neural Computation*, 20:2112–2131, 2008.

107. M. Mørup and M.N. Schmidt. Sparse non-negative tensor factor double deconvolution (SNTF2D)for multi channel time-frequency analysis. Technical report, Technical University of Denmark, DTU, 2006.

108. J. Niesing. *Simultaneous Component and Factor Analysis Methods for Two or More Groups: A Comparative Study*, volume 2nd ed. Leiden: The Netherlands. DSWO Press, Leiden University, 1997, 1997.

109. D. Nion and L. De Lathauwer. A Block Component Model based blind DS-CDMA receiver. *IEEE Transactions on Signal Processing*, 56(11):5567–5579, 2008.

110. D. Nion and L. De Lathauwer. An enhanced line search scheme for complex-valued tensor decompositions. Application in DS-CDMA. *Signal Processing*, 88(3):749–755, 2008.

111. P. Paatero. Least-squares formulation of robust nonnegative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

112. P. Paatero. A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis. *Chemometrics Intelligent Laboratory Systems*, 38(2):223–242, 1997.

113. P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

114. A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmean, and R. Pacual-Marqui. Non-smooth nonnegative matrix factorization (nsNMF). *IEEE Transactions Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006.

115. A.H. Phan and A. Cichocki. Fast and efficient algorithms for nonnegative Tucker decomposition. In *Proc. of The Fifth International Symposium on Neural Networks, Springer LNCS-5264*, pages 772–782, Beijing, China, 24–28, September 2008.

116. A.H. Phan and A. Cichocki. Multi-way nonnegative tensor factorization using fast hierarchical alternating least squares algorithm (HALS). In *Proc. of The 2008 International Symposium on Nonlinear Theory and its Applications*, Budapest, Hungary, 2008.

117. A.H. Phan and A. Cichocki. Local learning rules for nonnegative Tucker decomposition. *(submitted)*, 2009.

118. R.J. Plemmons and R.E. Cline. The generalized inverse of a nonnegative matrix. *Proceedings of the American Mathematical Society*, 31:46–50, 1972.

119. R. Rosipal, M. Girolami, L.J. Trejo, and A. Cichocki. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10:231–243, 2001.

120. P. Sajda, S. Du, and L. Parra. Recovery of constituent spectra using non-negative matrix factorization. In *Proceedings of SPIE*, volume 5207, pages 321–331, 2003.

121. T.M. Selee, T.G. Kolda, W.P. Kegelmeyer, and J.D. Griffin. Extracting clusters from large datasets with multiple similarity measures using IMSCAND. In Michael L. Parks and S. Scott Collis, editors, *CSRI Summer Proceedings 2007, Technical Report SAND2007-7977, Sandia National Laboratories, Albuquerque, NM and Livermore, CA*, pages 87–103, December 2007.

122. A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006.

123. H. Shen and J.-Z Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99:1015–1034, 2008.

124. N.D. Sidiropoulos and R. Bro. On the uniqueness of multilinear decomposition of N-way arrays. *Journal of Chemometrics*, 14:229–239, 2000.

125. P. Smaragdis. Non-negative matrix factor deconvolution; Extraction of multiple sound sources from monophonic inputs. *Lecture Notes in Computer Science*, 3195:494–499, 2004.

126. A. Smilde, R. Bro, and P. Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. John Wiley & Sons Ltd, New York, 2004.

127. A. Stegeman. On uniqueness conditions for Candecomp/Parafac and Indscal with full column rank in one mode. *Linear Algebra and its Applications*, http://www.gmw.rug.nl/ stegeman/:(in print), 2009.

128. A. Stegeman and A.L.F. de Almeida. Uniqueness conditions for a constrained three-way factor decomposition. *SIAM Journal on Matrix Analysis and Applications*, http://www.gmw.rug.nl/ stegeman/:(in print), 2009.

129. A. Stegeman and N.D. Sidiropoulos. On Kruskal's uniqueness condition for the Candecomp/Parafac decomposition. *Linear Algebra and its Applications*, 420:540–552, 2007.

130. A. Stegeman, J.M.F. Ten Berge, and L. De Lathauwer. Sufficient conditions for uniqueness in Candecomp/Parafac and Indscal with random component matrices. *Psychometrika*, 71(2):219–229, 2006.

131. Y. Takane and H.A.L. Kiers. Latent class DEDICOM. *Journal of Classification*, 14:225–247, 1997.

132. M.E. Timmerman and H.A.L. Kiers. Three mode principal components analysis: Choosing the numbers of components and sensitivity to local optima. *British Journal of Mathematical and Statistical Psychology*, 53(1):1–16, 2000.

133. G. Tomasi and R. Bro. PARAFAC and missing values. *Chemometrics Intelligent Laboratory Systems*, 75(2):163–180, 2005.

134. G. Tomasi and R. Bro. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics and Data Analysis*, 50(7):1700–1734, April 2006.

135. L.R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to Mathematical Psychology*, pages 110–127. Holt, Rinehart and Winston, New York, 1964.

136. L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

137. M.O. Ulfarsson and V. Solo. Dimension estimation in noisy PCA with SURE and random matrix theory. *IEEE Transactions on Signal Processing*, 56(12):5804–5816, December 2008.

138. H. Wang and N. Ahuja. Compact representation of multidimensional data using tensor rank-one decomposition. In *Proc. of International Conference on Pattern Recognition.*, volume 1, pages 44–47, 2004.

139. H. Wang and N. Ahuja. A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision*, 76(3):217–229, 2008.

140. Z. Wang, A. Maier, N.K. Logothetis, and H. Liang. Single-trial decoding of bistable perception based on sparse nonnegative tensor decomposition. *Journal of Computational Intelligence and Neuroscience*, 30:1–10, 2008.

141. Y. Washizawa and A. Cichocki. On line K-plane clustering learning algorithm for Sparse Component Analysis. In *Proc. of 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP2006*, pages 681–684, Toulouse, France, May 14-19 2006.

142. Q. Wu, T. Xia, and Y. Yu. Hierarchical tensor approximation of multi-dimensional images. In *14th IEEE International Conference on Image Processing*, volume IV, pages 49–52, San Antonio, 2007.

143. R. Zass and A. Shashua. Nonnegative sparse PCA. In *Neural Information Processing Systems (NIPS)*, Vancuver, Canada, Dec. 2006.

144. R. Zdunek and A. Cichocki. Nonnegative matrix factorization with constrained second-order optimization. *Signal Processing*, 87:1904–1916, 2007.

145. T. Zhang and G.H. Golub. Rank-one approximation to high order tensors. *SIAM Journal of Matrix Analysis and Applications*, 23(2):534–550, 2001.

2

Similarity Measures and Generalized Divergences

In this chapter, we overview and discuss properties of a large family of generalized and flexible divergences or similarity distances between two nonnegative sequences or patterns. They are formulated for probability distributions and for arrays used in Nonnegative Matrix Factorization (NMF) and Nonnegative Tensor Factorizations (NTF). Divergences, or their counter part (dis)similarity measures play an important role in the areas of neural computation, pattern recognition, learning, estimation, inference, and optimization. Generally speaking, they measure a quasi-distance or directed difference between two probability distributions $\boldsymbol{p}$ and $\boldsymbol{q}$ which can also be expressed for unconstrained nonnegative arrays and patterns.

Information theory, convex analysis, and information geometry play key roles in the formulation of divergences [2, 3, 6, 40, 23, 9, 29, 15, 21, 58, 36, 39, 38, 54, 55, 56, 57]. Divergence measures are commonly used to find a distance or difference between two n-dimensional probability distributions[1] $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$ and $\boldsymbol{q} = (q_1, q_2, \ldots, q_n)$. They are called non-normalized measures when they are not normalized to $\sum_{i=1}^{n} p_i = 1$, that is, their total masses are not necessarily unity but an arbitrary positive number.

We are mostly interested in distance-type measures which are separable, thus, satisfying the condition

$$D(\boldsymbol{p} \| \boldsymbol{q}) = \sum_{i=1}^{n} d(p_i, q_i) \geq 0, \qquad \text{which equals zero if and only if } \boldsymbol{p} = \boldsymbol{q} \tag{2.1}$$

[1] Usually, the vector $\boldsymbol{p}$ corresponds to the observed data and the vector $\boldsymbol{q}$ to estimated or expected data which are subject to constraints imposed on the assumed models. For NMF problem $\boldsymbol{p}$ corresponds to the data matrix $\mathbf{Y}$ and $\boldsymbol{q}$ corresponds to estimated matrix $\hat{\mathbf{Y}} = \mathbf{AX}$. An information divergence is a measure of distance between two probability curves. In this chapter, we discuss only one-dimensional probability curves (represented by nonnegative signals or time series). Generalization to two or multidimensional dimensional variables is straightforward; each single subscript is simply replaced by a doubly or triply indexed one.

but are not necessarily symmetric in the sense

$$D(\boldsymbol{p}\|\boldsymbol{q}) = D(\boldsymbol{q}\|\boldsymbol{p}), \tag{2.2}$$

and do not necessarily satisfy the triangular inequality

$$D(\boldsymbol{p}\|\boldsymbol{q}) \leq D(\boldsymbol{p}\|\boldsymbol{z}) + D(\boldsymbol{z}\|\boldsymbol{q}). \tag{2.3}$$

In other words, the distance-type measures under consideration are not necessarily a metric[2] on the space $\mathcal{P}$ of all probability distributions.

The scope of the results presented in this chapter is vast since the generalized divergence functions and their variants include quite a large number of useful loss functions including those based on the Relative entropies, generalized Kullback-Leibler or I-divergence, Hellinger distance, Jensen-Shannon divergence, J-divergence, Pearson and Neyman Chi-squared divergences, Triangular Discrimination and Arithmetic-Geometric (AG) Taneya divergence. Many of these measures belong to the class of Alpha-divergences and Beta-divergences and have been applied successfully in disciplines such as signal processing, pattern recognition, probability distributions, information theory, finance and economics [37]. In the following chapters we will apply such divergences as cost functions (possibly with additional constraints and regularization terms) to derive novel multiplicative and additive projected gradient and fixed point algorithms. These provide working solutions for the problems where nonnegative latent (hidden) components can be generally statistically dependent, and satisfy some other conditions or additional constraints such as sparsity or smoothness.

Section 2.1 addresses the divergences derived from simple component-wise errors (losses), these include the Euclidean and Minkowski metrics. We show that they are related to robust cost functions in Section 2.2. We then study in Section 2.3 the class of Csiszár f-divergences, which are characterized by the invariance and monotonicity properties. This class includes the Alpha-divergence, in particular the Kullback-Leibler divergence. The Bregman type divergences, derived from convex functions, are studied in Section 2.4. We also discuss divergences between positive-definite matrices. An important class of the Beta-divergences belongs to the class of Bregman divergences and is studied in detail in Section 2.6. They do not satisfy the invariance property except for the special case of the KL-divergence. When we extend divergences to positive measures where the total mass $\sum p_i$ is not restricted to unity, the Alpha-divergences belong to the classes of both Csiszár f-divergences and Bregman divergences. They are studied in detail in Section 2.5. Moreover, in Section 2.7 we discuss briefly Gamma-divergences which have "super robust" properties. Furthermore, in Section 2.8 we derive various divergences from Tsallis and Rényi entropy.

The divergences are closely related to the invariant geometrical properties of the manifold of probability distributions. This is a two-way relation: Divergences, in particular the Alpha-divergences, are naturally induced from geometry, and on the other hand divergences give a geometrical structure to the set of probability distributions or positive measures [4]. A brief introduction to information geometry is given in Appendix A. Information geometry provides mathematical tools to analyze families of probability distributions and positive measures. The structure of a manifold of probability distributions is derived from the invariance principle, and it consists of a Riemannian metric derived from the Fisher information matrix, together with

[2]The distance between two pdfs is called a metric if the following conditions hold: $D(\boldsymbol{p}\|\boldsymbol{q}) \geq 0$ with equality iff $\boldsymbol{p} = \boldsymbol{q}$, $D(\boldsymbol{p}\|\boldsymbol{q}) = D(\boldsymbol{q}\|\boldsymbol{p})$ and $D(\boldsymbol{p}\|\boldsymbol{q}) \leq D(\boldsymbol{p}\|\boldsymbol{z}) + D(\boldsymbol{z}\|\boldsymbol{q})$. Distances which are not a metric, are referred to as divergences.

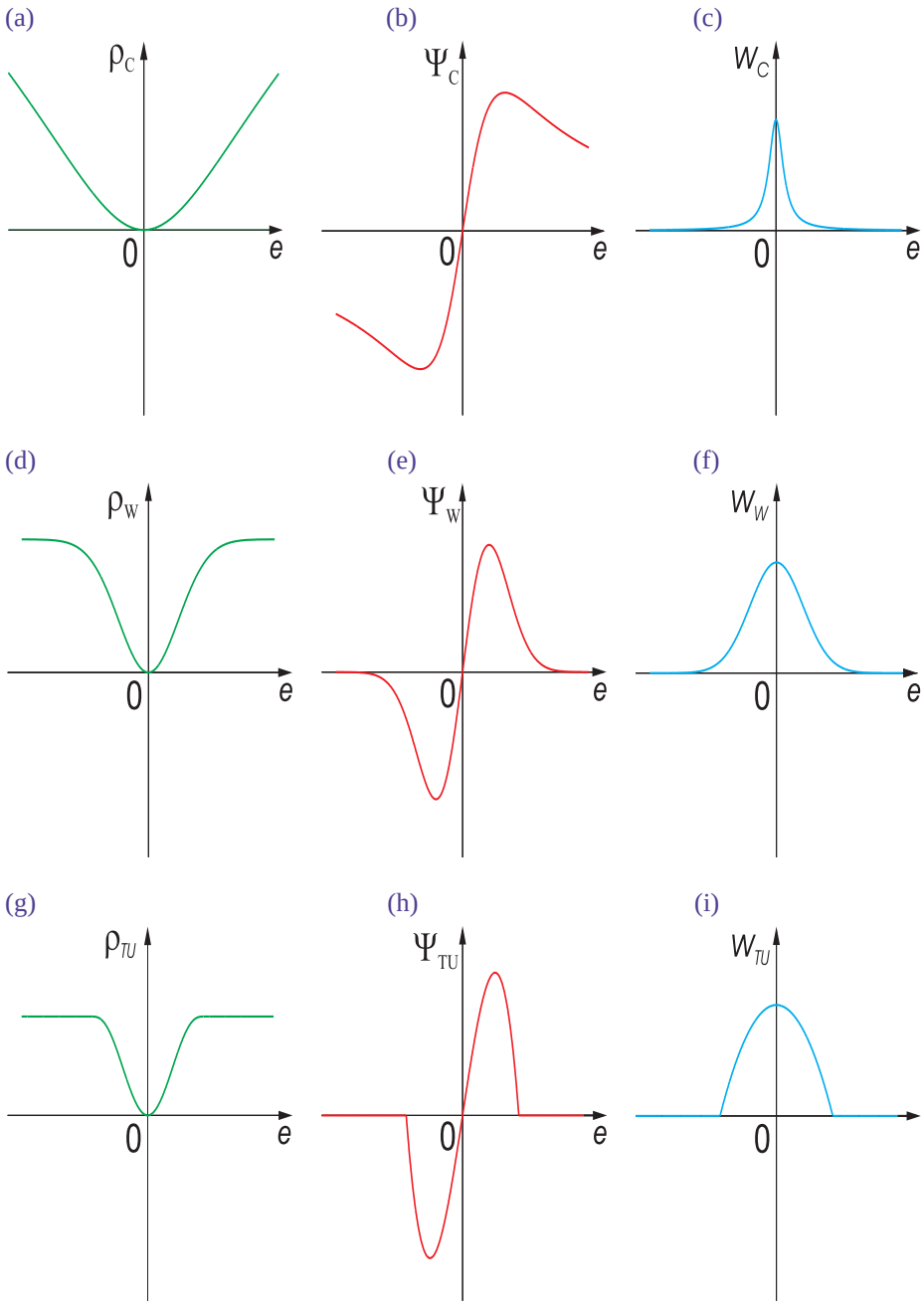


Fig. 2.2 Plots of typical M-functions (continued): (a),(b),(c) Cauchy; (d),(e),(f) Welsh; (g),(h),(i) Tukey. ($\rho(e)$ - loss function, $\Psi(e)$ influence function, and $w(e)$-weight function).

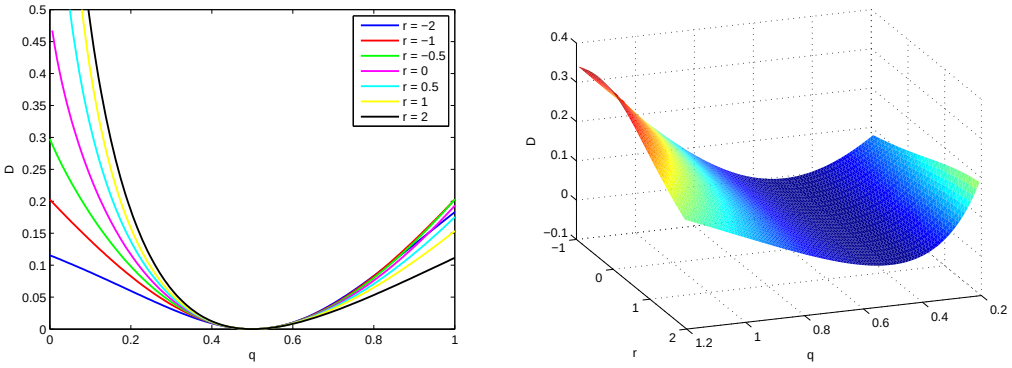


Fig. 2.7 Plots of the Rényi divergence for different values of parameters r (see Eq. (2.105)). All plots are evaluated against $p_i = 0.5$.

where $\widetilde{\boldsymbol{q}} = (\boldsymbol{p} + \boldsymbol{q})/2$.

For the singular values $\alpha = 1$ and $\alpha = 0$, the Alpha-divergences (2.106) can be evaluated as

$$\lim_{\alpha\to 0} D_{Am1}^{(\alpha)}(\boldsymbol{p}\|\tilde{\boldsymbol{q}}) = \sum_i \left(\frac{p_i + q_i}{2} \ln\left(\frac{p_i + q_i}{2p_i}\right) + \frac{p_i - q_i}{2}\right), \tag{2.108}$$

and

$$\lim_{\alpha\to 1} D_{Am1}^{(\alpha)}(\boldsymbol{p}\|\tilde{\boldsymbol{q}}) = \sum_i \left(p_i \ln\left(\frac{2p_i}{p_i + q_i}\right) + \frac{q_i - p_i}{2}\right). \tag{2.109}$$

It is important to consider the following particular cases for (2.107):

1. Triangular Discrimination (TD) (Dacunha-Castelle)

$$D_{Am2}^{(-1)}(\widetilde{\boldsymbol{q}}\|\boldsymbol{p}) = \frac{1}{4} D_T(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{4}\sum_i \frac{(p_i - q_i)^2}{p_i + q_i}. \tag{2.110}$$

2. Relative Jensen-Shannon divergence (Burbea and Rao, Sgarro, Sibson [10, 11, 49])

$$\lim_{\alpha\to 0} D_{Am2}^{(\alpha)}(\widetilde{\boldsymbol{q}}\|\boldsymbol{p}) = D_{RJS}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(p_i \ln\left(\frac{2p_i}{p_i + q_i}\right) - p_i + q_i\right). \tag{2.111}$$

3. Relative Arithmetic-Geometric divergence [50, 51, 52]

$$\lim_{\alpha\to 1} D_{Am2}^{(\alpha)}(\widetilde{\boldsymbol{q}}\|\boldsymbol{p}) = \frac{1}{2} D_{RAG}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left((p_i + q_i) \ln\left(\frac{p_i + q_i}{2p_i}\right) + p_i - q_i\right). \tag{2.112}$$

3. Neyman Chi-squared divergence

$$D^{(2)}_{Am2}(\widetilde{\boldsymbol{q}} \,\|\, \boldsymbol{p}) = \frac{1}{8} D_\chi(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \frac{1}{8} \sum_i \frac{(p_i - q_i)^2}{p_i}. \tag{2.113}$$

2.5.2 Symmetric Alpha-Divergences

The standard Alpha-divergence is asymmetric, that is, $D^{(\alpha)}_A(\boldsymbol{p} \,\|\, \boldsymbol{q}) \neq D^{(\alpha)}_A(\boldsymbol{q} \,\|\, \boldsymbol{p})$. The symmetric Alpha-divergence (Type-1) can be defined as

$$D^{(\alpha)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = D^{(\alpha)}_A(\boldsymbol{p} \,\|\, \boldsymbol{q}) + D^{(\alpha)}_A(\boldsymbol{q} \,\|\, \boldsymbol{p}) = \sum_i \frac{p_i^\alpha q_i^{1-\alpha} + p_i^{1-\alpha} q_i^\alpha - (p_i + q_i)}{\alpha(\alpha - 1)}. \tag{2.114}$$

As special cases, we obtain several well-known symmetric divergences:

1. Symmetric Chi-Squared divergence [27]

$$D^{(-1)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = D^{(2)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \frac{1}{2} D_\chi(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \frac{1}{2} \sum_i \frac{(p_i - q_i)^2 (p_i + q_i)}{p_i q_i}. \tag{2.115}$$

2. J-divergence corresponding to Jeffreys entropy maximization [32, 35]

$$\lim_{\alpha \to 0} D^{(\alpha)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \lim_{\alpha \to 1} D^{(\alpha)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = D_J(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \sum_i (p_i - q_i) \ln\left(\frac{p_i}{q_i}\right). \tag{2.116}$$

3. Squared Hellinger distance [31]

$$D^{(1/2)}_{AS\,1}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = 8 D_H(\boldsymbol{p} \,\|\, \boldsymbol{q}) = 4 \sum_i (\sqrt{p_i} - \sqrt{q_i})^2. \tag{2.117}$$

An alternative wide class of symmetric divergences can be described by the following symmetric Alpha-divergence (Type-2):

$$\begin{aligned} D^{(\alpha)}_{AS\,2}(\boldsymbol{p} \,\|\, \boldsymbol{q}) &= D^{(\alpha)}_A\left(\frac{\boldsymbol{p} + \boldsymbol{q}}{2} \,\|\, \boldsymbol{q}\right) + D^{(\alpha)}_A\left(\frac{\boldsymbol{p} + \boldsymbol{q}}{2} \,\|\, \boldsymbol{p}\right) \\ &= \frac{1}{\alpha(\alpha - 1)} \sum_i \left((p_i^{1-\alpha} + q_i^{1-\alpha}) \left(\frac{p_i + q_i}{2}\right)^\alpha - (p_i + q_i) \right). \end{aligned} \tag{2.118}$$

The above measure admits the following particular cases:

1. Triangular Discrimination

$$D^{(-1)}_{AS\,2}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \frac{1}{2} D_T(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \frac{1}{2} \sum_i \frac{(p_i - q_i)^2}{p_i + q_i}. \tag{2.119}$$

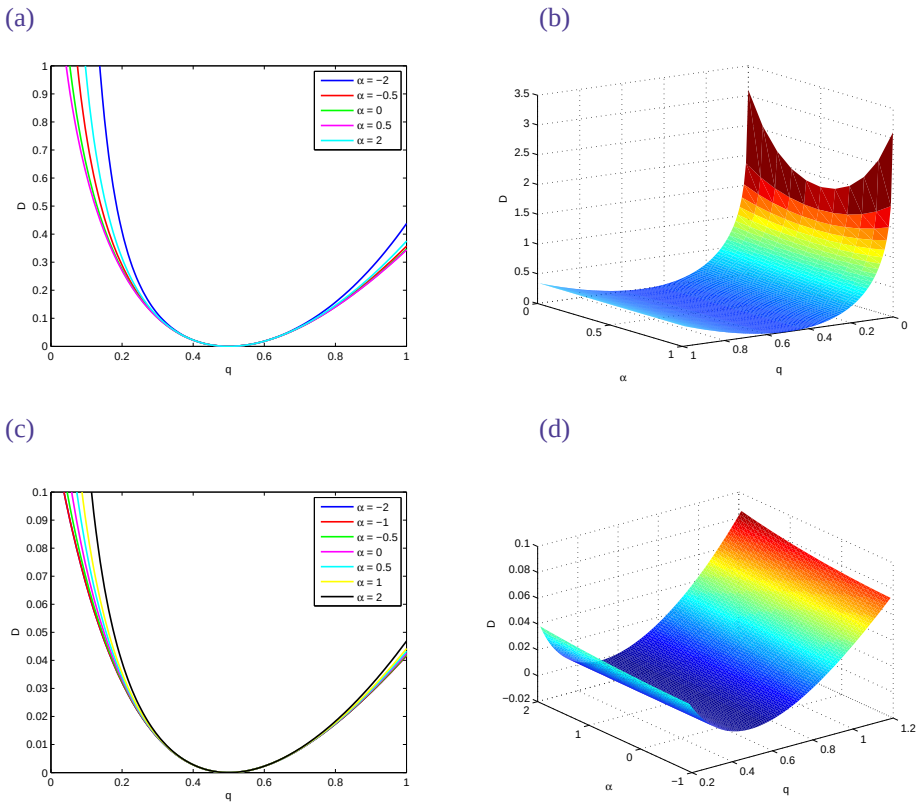


Fig. 2.8 2D and 3D plots of Symmetric Alpha-divergences for different values of parameter α (a)-(b) $D^{(\alpha)}_{AS1}(\boldsymbol{p}\|\boldsymbol{q})$ – Eq. (2.114) and (c)-(d) $D^{(\alpha)}_{AS2}(\boldsymbol{p}\|\boldsymbol{q})$ – Eq. (2.118). (All plots are evaluated against $p_i = 0.5$).

2. Symmetric Jensen-Shannon divergence[11]

$$\lim_{\alpha\to 0} D^{(\alpha)}_{AS2}(\boldsymbol{p}\|\boldsymbol{q}) = D_{JS}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(p_i \ln\left(\frac{2p_i}{p_i+q_i}\right) + q_i \ln\left(\frac{2q_i}{p_i+q_i}\right)\right). \tag{2.120}$$

3. Arithmetic-Geometric divergence [50, 51, 52]

$$\lim_{\alpha\to 1} D^{(\alpha)}_{AS2}(\boldsymbol{p}\|\boldsymbol{q}) = D_{AG}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i (p_i+q_i) \ln\left(\frac{p_i+q_i}{2\sqrt{p_i q_i}}\right). \tag{2.121}$$

[11] The Jensen-Shannon divergence is a symmetrized and smoothed version of the Kullback-Leibler divergence, i.e., it can be interpreted as the average of the Kullback-Leibler divergences to the average distribution. In other words, the Jensen-Shannon divergence is the entropy of the average from which the average of the Shannon entropies is subtracted: $D_{JS} = H_S((\boldsymbol{p}+\boldsymbol{q})/2) - (H_S(\boldsymbol{p}) + H_S(\boldsymbol{q}))/2$, where $H_S(\boldsymbol{p}) = -\sum_i p_i \ln p_i$.

4. Symmetric Chi-squared divergence [27]

$$D^{(2)}_{AS2}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{8}D_{\chi}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{8}\sum_i \frac{(p_i - q_i)^2(p_i + q_i)}{p_i q_i}. \tag{2.122}$$

The above Alpha-divergence is symmetric in its arguments $\boldsymbol{p}$ and $\boldsymbol{q}$, and it is well-defined even if $\boldsymbol{p}$ and $\boldsymbol{q}$ are not absolutely continuous, i.e., D_{AS2} is well-defined even if, for some indexes p_i it vanishes without vanishing q_i or if q_i vanishes without vanishing p_i. It is also lower- and upper-bounded, for example, the Jensen-Shannon divergence is bounded between 0 and 2.

2.6 BETA-DIVERGENCES

The Beta-divergence was introduced by Eguchi, Kano, Minami and also investigated by others [38, 8, 41, 40, 42, 14, 15, 16, 17, 38, 36]. It has a dually flat structure of information geometry, where the Pythagorean theorem holds. However, it is not invariant under a change of the dominating measure, and not invariance monotone for summarization, except for the special case of $\beta = 0$ which gives the KL-divergence.

First let us define the discrete Beta-divergence between two un-normalized density functions: p_i and q_i by

$$D^{(\beta)}_B(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(p_i \frac{p_i^{\beta} - q_i^{\beta}}{\beta} - \frac{p_i^{\beta+1} - q_i^{\beta+1}}{\beta + 1} \right), \tag{2.123}$$

where β is a real number ($\beta \neq 0$ and $\beta \neq -1$).

It is interesting to note that, for $\beta = 1$, we obtain the standard squared Euclidean distance, while for the singular cases $\beta = 0$ and $\beta = -1$ the Beta-divergence has to be defined in limiting cases as $\beta \to 0$ and $\beta \to -1$, respectively.

When these limits are evaluated for $\beta \to 0$, we obtain the generalized Kullback-Leibler divergence (called the I-divergence) defined as[12]

$$D_{KL}(\boldsymbol{p}\|\boldsymbol{q}) = \lim_{\beta\to 0} D^{(\beta)}_B(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(p_i \ln \frac{p_i}{q_i} - p_i + q_i \right), \tag{2.124}$$

whereas for $\beta \to -1$ the Itakura-Saito divergence is obtained as

$$D_{IS}(\boldsymbol{p}\|\boldsymbol{q}) = \lim_{\beta\to -1} D^{(\beta)}_B(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(\ln(\frac{q_i}{p_i}) + \frac{p_i}{q_i} - 1 \right). \tag{2.125}$$

Remark 2.5 *Recently, Févotte, Bertin and Durrieu [29] investigated and emphasized the fundamental properties of Itakura-Saito divergence as follows: "This divergence was obtained by Itakura and Saito (in 1968) from the maximum likelihood (ML) estimation of short-time speech*

[12] It should be noted that $\lim_{\beta\to 0} \frac{p^{\beta}-q^{\beta}}{\beta} = \ln(p/q)$ and $\lim_{\beta\to 0} \frac{p^{\beta}-1}{\beta} = \ln p$.

spectra under autoregressive modeling. It was presented as 'a measure of the goodness of fit between two spectra' and became a standard measure in the speech processing community due to the good perceptual properties of the reconstructed signals. Other important properties of the Itakura-Saito divergence include scale invariance, meaning that low energy components of $\boldsymbol{p}$ *bear the same relative importance as high energy ones. This is relevant to situations where the coefficients of* $\boldsymbol{p}$ *have a large dynamic range, such as in short-term audio spectra. The Itakura-Saito divergence also leads to desirable statistical interpretations of the NMF problem". Furthermore, they explained how under simple Gaussian assumptions NMF can be recast as a maximum likelihood (ML) estimation of matrices* $\mathbf{A}$ *and* $\mathbf{X}$ *and described how IS-NMF can be interpreted as ML of* $\mathbf{A}$ *and* $\mathbf{X}$ *in multiplicative Gamma noise [29, 30].*

Hence, the Beta-divergence can be represented in a more explicit form:

$$D_B^{(\beta)}(\boldsymbol{p}\,\|\,\boldsymbol{q}) = \begin{cases} \sum_i \left(p_i \, \dfrac{p_i^{\beta} - q_i^{\beta}}{\beta} - \dfrac{p_i^{\beta+1} - q_i^{\beta+1}}{\beta+1} \right), & \beta \neq 0, -1, \\ \sum_i \left(p_i \, \ln(\dfrac{p_i}{q_i}) - p_i + q_i \right), & \beta = 0, \\ \sum_i \left(\ln(\dfrac{q_i}{p_i}) + \dfrac{p_i}{q_i} - 1 \right), & \beta = -1. \end{cases} \tag{2.126}$$

As observed by Févotte *et al.* [29] the derivative of the Beta-divergence for separable terms $d(p_i\|q_i)$, with respect to q_i is continuous in $\beta \in \mathbb{R}$ parameter, and can be expressed as

$$\nabla_{q_i} d(p_i\|q_i) = q_i^{\beta+1}(q_i - p_i.) \tag{2.127}$$

It is obvious that that $d(p_i\|q_i)$, as a function of q_i for fixed p_i, has a single minimum at $q_i = p_i$ and that it increases with $|q_i - p_i|$, justifying its relevance as a measure of distortion of dissimilarity [29, 30].

The Beta-divergence smoothly connects the Itakura-Saito distance and the squared Euclidean distance and passes through the KL I-divergence $D_{KL}(\boldsymbol{p}\,\|\,\boldsymbol{q})$. Such a parameterized connection is impossible in the family of the Alpha-divergences.

The Beta-divergence is related to the Tweedie distributions [35, 38, 8]. In probability and statistics, the Tweedie distributions are a family of probability distributions which include continuous distributions such as the normal and gamma, the purely discrete scaled Poisson distribution, and the class of mixed compound Poisson-Gamma distributions which have positive mass at zero, but are otherwise continuous. Tweedie distributions belong to the exponential dispersion model family of distributions, a generalization of the exponential family, which are the response distributions for generalized linear models [35]. Tweedie distributions exist for all real values of β except for $0 < \beta < 1$. Apart from special cases shown in Table 2.8, their probability density function have no closed form. The choice of the parameter β depends on the statistical distribution of data. For example, the optimal choice of the parameter β for the normal distribution is $\beta = 1$, for the gamma distribution it is $\beta = -1$, for the Poisson distribution $\beta = 0$, and for the compound Poisson distribution $\beta \in (-1, 0)$ (see Table 2.8) [14, 15, 16, 17, 38, 40, 41].

Table 2.8 Special cases for Tweedie distributions.

Parameter β	Divergence	Distribution
1	Squared Euclidean distance	Normal
0	KL I-divergence $D_{KL}(\boldsymbol{p}\|\boldsymbol{q})$	Poisson
$(-1, 0)$		Compound Poisson
-1	Dual KL I-divergence $D_{KL}(\boldsymbol{q}\|\boldsymbol{p})$	Gamma
-2	Dual KL I-divergence $D_{KL}(\boldsymbol{q}\|\boldsymbol{p})$	Inverse–Gaussian

The Beta-divergences can be obtained from the Alpha-divergence (2.42) by applying nonlinear transformations:

$$p_i \to p_i^{\beta+1}, \qquad q_i \to q_i^{\beta+1} \quad \alpha = \frac{1}{1+\beta}. \tag{2.128}$$

For example, using these substitutions for (2.42) and assuming that $\alpha = (\beta+1)^{-1}$ we obtain the following divergence

$$D_A^{(\beta)}(\boldsymbol{p}\|\boldsymbol{q}) = (\beta+1)^2 \sum_i \left[\frac{p_i^{\beta+1}}{\beta(\beta+1)} - \frac{p_i q_i^{\beta}}{\beta} + \frac{q_i^{\beta+1}}{\beta+1}\right] \tag{2.129}$$

Observe that, after simple algebraic manipulations and by ignoring the scaling factor $(\beta+1)^2$, we obtain the Beta-divergence defined by Eq. (2.123).

In fact, there exists the same link between the whole family of Alpha-divergences and the family of Beta-divergences (see Table 2.7). For example, we can derive the symmetric Beta divergence from symmetric Alpha-divergence (Type-1) (2.114:

$$\begin{aligned} D_{BS1}^{(\beta)}(\boldsymbol{p}\|\boldsymbol{q}) &= D_B^{(\beta)}(\boldsymbol{p}\|\boldsymbol{q}) + D_B^{(\beta)}(\boldsymbol{q}\|\boldsymbol{p}) \\ &= \frac{1}{\beta}\sum_i \left(p_i^{\beta+1} + q_i^{\beta+1} - p_i q_i^{\beta} - p_i^{\beta} q_i\right), \end{aligned} \tag{2.130}$$

and from symmetric Alpha-divergence (Type-2) (2.118

Table 2.9 The fundamental generalized divergences.

Divergence name	Formula
Alpha-divergence	$D_A^{(\alpha)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\sum_i (p_i^\alpha\, q_i^{1-\alpha} + (\alpha-1)\, q_i - \alpha\, p_i)}{\alpha(\alpha-1)}$
Beta-divergence	$D_B^{(\beta)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\sum_i (p_i^{\beta+1} + \beta\, q_i^{\beta+1} - (\beta+1)\, p_i\, q_i^{\beta})}{\beta(\beta+1)}$
Gamma-divergence	$D_G^{(\gamma)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\ln\left(\sum_i p_i^{\gamma+1}\right) + \gamma \ln\left(\sum_i q_i^{\gamma+1}\right) - (\gamma+1)\ln\left(\sum_i p_i\, q_i^{\gamma}\right)}{\gamma(\gamma+1)}$
Bregman divergence	$D_\phi(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(\phi(p_i) - \phi(q_i) - \frac{d\phi}{dq_i}(p_i - q_i)\right)$
Csiszár f-divergence	$D_f(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i q_i\, f\left(\frac{p_i}{q_i}\right)$
Rényi divergence	$D_R(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \frac{\ln(p_i^r q_i^{1-r} - r\, p_i + (r-1)\, q_i + 1)}{r(r-1)}$
Rényi-type divergence	$D_\psi(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \ln\left(\psi^{-1}\left(p_i \psi\left(\frac{p_i}{q_i}\right)\right)\right)$
Burbea-Rao divergence	$D_{BR}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(\frac{h(p_i) + h(q_i)}{2} - h\left(\frac{p_i + q_i}{2}\right)\right)$

Similarly to the Alpha and Beta-divergences, we can also define the symmetric Gamma-divergence as

$$D_{GS}^{(\gamma)}(\boldsymbol{p}\,\|\,\boldsymbol{q}) = D_G^{(\gamma)}(\boldsymbol{p}\,\|\,\boldsymbol{q}) + D_G^{(\gamma)}(\boldsymbol{q}\,\|\,\boldsymbol{p}) = \frac{1}{\gamma}\, \ln\left[\frac{\left(\sum_i p_i^{1+\gamma}\right)\left(\sum_i q_i^{1+\gamma}\right)}{\left(\sum_i p_i^{\gamma}\, q_i\right)\left(\sum_i p_i\, q_i^{\gamma}\right)}\right]. \tag{2.135}$$

The symmetric Gamma-divergence has similar properties to the asymmetric Gamma-divergence:

Table 2.9 The fundamental generalized divergences.

Divergence name	Formula
Alpha-divergence	$D_A^{(\alpha)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\sum_i (p_i^\alpha\, q_i^{1-\alpha} + (\alpha-1)\, q_i - \alpha\, p_i)}{\alpha(\alpha-1)}$
Beta-divergence	$D_B^{(\beta)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\sum_i (p_i^{\beta+1} + \beta\, q_i^{\beta+1} - (\beta+1)\, p_i\, q_i^\beta)}{\beta(\beta+1)}$
Gamma-divergence	$D_G^{(\gamma)}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{\ln\left(\sum_i p_i^{\gamma+1}\right) + \gamma \ln\left(\sum_i q_i^{\gamma+1}\right) - (\gamma+1)\ln\left(\sum_i p_i\, q_i^\gamma\right)}{\gamma(\gamma+1)}$
Bregman divergence	$D_\phi(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(\phi(p_i) - \phi(q_i) - \frac{d\phi}{dq_i}(p_i - q_i)\right)$
Csiszár f-divergence	$D_f(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i q_i\, f\left(\frac{p_i}{q_i}\right)$
Rényi divergence	$D_R(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \frac{\ln(p_i^r q_i^{1-r} - r\, p_i + (r-1)\, q_i + 1)}{r(r-1)}$
Rényi-type divergence	$D_\psi(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \ln\left(\psi^{-1}\left(p_i \psi\left(\frac{p_i}{q_i}\right)\right)\right)$
Burbea-Rao divergence	$D_{BR}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_i \left(\frac{h(p_i) + h(q_i)}{2} - h\left(\frac{p_i + q_i}{2}\right)\right)$

Similarly to the Alpha and Beta-divergences, we can also define the symmetric Gamma-divergence as

$$D_{GS}^{(\gamma)}(\boldsymbol{p}\,\|\,\boldsymbol{q}) = D_G^{(\gamma)}(\boldsymbol{p}\,\|\,\boldsymbol{q}) + D_G^{(\gamma)}(\boldsymbol{q}\,\|\,\boldsymbol{p}) = \frac{1}{\gamma}\, \ln\left[\frac{\left(\sum_i p_i^{1+\gamma}\right)\left(\sum_i q_i^{1+\gamma}\right)}{\left(\sum_i p_i^\gamma\, q_i\right)\left(\sum_i p_i\, q_i^\gamma\right)}\right]. \tag{2.135}$$

The symmetric Gamma-divergence has similar properties to the asymmetric Gamma-divergence:

References

1. M.S. Ali and S.D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of Royal Statistical Society*, Ser B(28):131–142, 1966.

2. S. Amari. *Differential-Geometrical Methods in Statistics*. Springer Verlag, 1985.

3. S. Amari. Dualistic geometry of the manifold of higher-order neurons. *Neural Networks*, 4(4):443–451, 1991.

4. S. Amari. Information geometry and its applications: Convex function and dually flat manifold. In F. Nielson, editor, *Emerging Trends in Visual Computing*, pages 75–102. Springer Lecture Notes in Computer Science, 2009.

5. S. Amari, K. Kurata, and H. Nagaoka. Information geometry of Boltzman machines. *IEEE Transactions on Neural Networks*, 3:260–271, 1992.

6. S. Amari and H. Nagaoka. *Methods of Information Geometry*. Oxford University Press, New York, 2000.

7. A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D.S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. In *KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 509–514, New York, NY, USA, 2004. ACM Press.

8. A. Basu, I. R. Harris, N.L. Hjort, and M.C. Jones. Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559, 1998.

9. L. Bregman. The relaxation method of finding a common point of convex sets and its application to the solution of problems in convex programming. *Comp. Math. Phys., USSR*, 7:200–217, 1967.

10. J. Burbea and C.R. Rao. Entropy differential metric, distance and divergence measures in probability spaces: A unified approach. *J. Multi. Analysis*, 12:575–596, 1982.

11. J. Burbea and C.R. Rao. On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory*, IT-28:489–495, 1982.

12. N.N. Chentsov. *Statistical Decision Rules and Optimal Inference*. AMS (translated from Russian, Nauka, 1972, New York, NY, 1982.

13. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on a sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

14. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer, LNAI-4029*, 4029:548–562, 2006.

15. A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: Family of new algorithms. *Springer, LNCS-3889*, 3889:32–39, 2006.

16. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using Alpha and Beta divergencies. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume III, pages 1393–1396, Honolulu, Hawaii, USA, April 15–20 2007.

17. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari. Novel multi-layer nonnegative tensor factorization with sparsity constraints. *Springer, LNCS-4432*, 4432:271–280, April 11–14 2007.

18. N. Cressie and T. Read. Multinomial goodness-of-fit tests. *Journal of Royal Statistical Society B*, 46(3):440–464, 1984.

19. N.A. Cressie and T.C.R. Read. *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Springer, New York, 1988.

20. I. Csiszár. Eine Informations Theoretische Ungleichung und ihre Anwendung auf den Beweis der Ergodizitt von Markoffschen Ketten. *Magyar Tud. Akad. Mat. Kutat Int. Kzl*, 8:85–108, 1963.

21. I. Csiszár. Information measures: A critial survey. In *Transactions of the 7th Prague Conference*, pages 83–86, 1974.

22. I. Csiszár. A geometric interpretation of darroch and ratcliffs generalized iterative scaling. *The Annals of Statistics*, 17(3):1409–1413, 1989.

23. I. Csiszár. Axiomatic characterizations of information measures. *Entropy*, 10:261–273, 2008.

24. I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York, USA, 1981.

25. I. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Neural Information Proc. Systems*, pages 283–290, Vancouver, Canada, December 2005.

26. I.S. Dhillon and J.A. Tropp. Matrix nearness problems with Bregman divergences. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1120–1146, 2007.

27. S.S. Dragomir. *Inequalities for Csiszár f-Divergence in Information Theory*. Victoria University, Melbourne, Australia, 2000. edited monograph.

28. S. Eguchi and Y. Kano. Robustifying maximum likelihood estimation. In *Institute of Statistical Mathematics*, Tokyo, 2001.

29. C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the takura-Saito divergence. With application to music analysis. *Neural Computation*, 21(3):793–830, 2009.

30. C. Févotte and A. T. Cemgil. Nonnegative matrix factorizations as probabilistic inference in composite models. In *In Proc. 17th European Signal Processing Conference (EUSIPCO'09)*, Glagow, Scotland, Augustr 24–28 2009.

31. Y. Fujimoto and N. Murata. A modified EM algorithm for mixture models based on Bregman divergence. *Annals of the Institute of Statistical Mathematics*, 59:57–75, 2007.

32. H. Fujisawa and S. Eguchi. Robust parameter estimation with a small bias against heavy contamination. *Multivariate Analysis*, 99(9):2053–2081, 2008.

33. E. Hellinger. Neue Begründung der Theorie Quadratischen Formen von unendlichen vielen Veränderlichen. *Journal Reine Ang. Math.*, 136:210–271, 1909.

34. H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proc. Roy. Soc. Lon., Ser. A*, 186:453–461, 1946.

35. B. Jorgensen. *The Theory of Dispersion Models*. Chapman and Hall, London, 1997.

36. R. Kompass. A generalized divergence measure for nonnegative matrix factorization. *Neural Computation*, 19(3):780–791, 2006.

37. S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

38. M. Minami and S. Eguchi. Robust blind source separation by Beta-divergence. *Neural Computation*, 14:1859–1886, 2002.

39. T.P. Minka. Divergence measures and message passing. *Microsoft Research Technical Report (MSR-TR-2005)*, 2005.

40. M.N.H. Mollah, S. Eguchi, and M. Minami. Robust prewhitening for ICA by minimizing beta-divergence and its application to FastICA. *Neural Processing Letters*, 25(2):91–110, 2007.

41. M.N.H. Mollah, M. Minami, and S. Eguchi. Exploring latent structure of mixture ica models by the minimum beta-divergence method. *Neural Computation*, 16:166–190, 2006.

42. N. Murata, T. Takenouchi, T. Kanamori, and S. Eguchi. Information geometry of U-Boost and Bregman divergence. *Neural Computation*, 16:1437–1481, 2004.

43. J. Naudats. Generalized exponential families and associated entropy functions. *Entropy*, 10:131–149, 2008.

44. A. Ohara. Possible generalization of Boltzmann-Gibbs statistics. *Journal Statistics Physics*, 52:479–487, 1988.

45. A. Ohara. Geometry of distributions associated with Tsallis statistics and properties of relative entropy minimization. *Physics Letters A*, 370:184–193, 2007.

46. A. Rényi. On measures of entropy and information. In *Proc. 4th Berk. Symp. Math. Statist. and Probl.*, volume 1, pages 547–561, University of California Press, Berkeley, 1961.

47. A.L. Rukhin. Recursive testing of multiple hypotheses: Consistency and efficiency of the Bayes rule. *Ann. Statist.*, 22(2):616–633, 1994.

48. B.D. Sharma and D.P. Mittal. New nonadditive measures of inaccuracy. *J. Math. Sci.*, 10:122–133, 1975.

49. B.D. Sharma and D.P. Mittal. New nonadditive measures of relative information. *J. Comb. Inform. and Syst. Sci.*, 2:122–133, 1977.

50. B.D. Sharma and I.J. Taneja. Entropy of type (α, β) and other generalized additive measures in information theory. *Metrika*, 22:205–215, 1975.

51. R. Sibson. Information radius. *Probability Theory and Related Fields*, 14(2):149–160, June 1969.

52. I.J. Taneja. On measures of information and inaccuarcy. *J. Statist. Phys.*, 14:203–270, 1976.

53. I.J. Taneja. On generalized entropies with applications. In L.M. Ricciardi, editor, *Lectures in Applied Mathematics and Informatics*, pages 107–169. Manchester University Press, England, 1990.

54. I.J. Taneja. New developments in generalized information measures. In P.W. Hawkes, editor, *Advances in Imaging and Electron Physics*, volume 91, pages 37–135. 1995.

55. I. Vajda. *Theory of Statistical Inference and Information*. Kluwer Academic Press, London, 1989.

56. J. Zhang. Divergence function, duality, and convex analysis. *Neural Computation*, 16(1):159–195, 2004.

57. J. Zhang. Referential duality and representational duality on statistical manifolds. In *Proceedings of the Second International Symposium on Information Geometry and its Applications)*, pages 58–67, Tokyo, Japan, 2006.

58. J. Zhang. A note on curvature of a-connections of a statistical manifold. *Annals of the Institute of Statistical Mathematics*, 59:161–170, 2007.

59. J. Zhang and H. Matsuzoe. Dualistic differential geometry associated with a convex function. In *Springer Series of Advances in Mechanics and Mathematics*, pages 58–67, 2008.

60. H. Zhu and R. Rohwer. Information geometric measurements of generalization. Technical Report NCRG/4350, Aston University, Birmingham, UK, August 31 1995.

3

Multiplicative Iterative Algorithms for NMF with Sparsity Constraints

In this chapter we introduce a wide family of iterative multiplicative algorithms for nonnegative matrix factorization (NMF) and related problems, subject to additional constraints such as sparsity and/or smoothness. Although a standard multiplicative update rule for NMF achieves a sparse representation[1] of its factor matrices, we can impose control over the sparsity of the matrices by designing a suitable cost function with additional penalty terms. There are several ways to incorporate sparsity constraints. A simple approach is to add suitable regularization or penalty terms to an optimized cost (loss) function. Another alternative approach is to implement at each iteration step a nonlinear projection (shrinkage) or filtering which increases sparseness of the estimated matrices.

We consider a wide class of cost functions or divergences (see Chapter 2, leading to generalized multiplicative algorithms with regularization and/or penalty terms. Such relaxed forms of the multiplicative NMF algorithms usually provide better performance and convergence speed, and allow us to extract desired unique components. The results included in this chapter give a vast scope as the range of cost functions includes a large number of generalized divergences, such as the squared weighted Euclidean distance, relative entropy, Kullback Leibler I-divergence, Alpha- and Beta-divergences, Bregman divergence and Csiszár f-divergence. As special cases we introduce the multiplicative algorithms for the squared Hellinger, Pearson's Chi-squared, and Itakura-Saito distances.

We consider the basic NMF model

$$\mathbf{Y} \cong \mathbf{A}\mathbf{X}, \tag{3.1}$$

[1] We define sparse NMF $\mathbf{Y} \cong \mathbf{A}\mathbf{X}$ as approximate nonnegative matrix factorization in which both or at least one factor matrix $\mathbf{A}$ or $\mathbf{X}$ is sparse.

for which we construct a set of suitable cost functions $D(\mathbf{Y}||\mathbf{AX})$ which measure the distance between data $y_{it} = [\mathbf{Y}]_{it}$ and the set of estimated parameters $q_{it} = [\hat{\mathbf{Y}}]_{it} = [\mathbf{AX}]_{it} = \sum_{j=1}^{J} a_{ij}x_{jt}$. The multiplicative learning algorithms aim at minimizing a specific cost function or a set of cost functions by alternately updating the parameters a_{ij} while keeping x_{jt} fixed, and then updating the parameters x_{jt} while keeping all a_{ij} fixed. In fact, it is often convenient to estimate a set of parameters $\mathbf{A}$ and $\mathbf{X}$ by the sequential minimization of two different cost functions with the same global minima.

For a large-scale NMF problem (with $T \geq I >> J$) we do not need to store and process large data matrices $\mathbf{Y} \in \mathbb{R}^{I\times T}$ and $\hat{\mathbf{Y}} = \mathbf{AX} \in \mathbb{R}_{+}^{I\times T}$. Instead of the typical alternating minimization of a one global cost function $D(\mathbf{Y}||\mathbf{AX})$ we may perform the following alternating minimization on the subsets:[2]

$$\mathbf{A} = \arg\min_{\mathbf{A}\geq\mathbf{0}} D_1(\mathbf{Y}_c||\mathbf{AX}_c), \quad \text{for fixed} \quad \mathbf{X}, \tag{3.2}$$

$$\mathbf{X} = \arg\min_{\mathbf{X}\geq\mathbf{0}} D_2(\mathbf{Y}_r||\mathbf{A}_r\mathbf{X}), \quad \text{for fixed} \quad \mathbf{A}, \tag{3.3}$$

where $\mathbf{Y}_r \in \mathbb{R}^{R\times T}$ and $\mathbf{Y}_c \in \mathbb{R}^{I\times C}$ comprise respectively the row and column subsets of the matrix $\mathbf{Y}$, whereas $\mathbf{A}_r \in \mathbb{R}_{+}^{R\times J}$ and $\mathbf{X}_c \in \mathbb{R}_{+}^{J\times C}$ are the row and column subsets of matrices $\mathbf{A}$ and $\mathbf{X}$. Typically $R << I$ and $C << T$ (see Chapter 1 for more detail).

All multiplicative learning rules ensure the nonnegativity of the factor matrices. Obviously, all the successive estimates remain positive if the initial estimate is positive. However, if a component of the solution becomes equal to zero, it remains at zero for all the successive iterations. To circumvent this problem, we usually force the values of the estimates a_{ij} and x_{jt} not to be less than a certain small positive value ε (typically, $\varepsilon = 10^{-9}$), called the threshold constraint, which often determines the noise floor, that is, $x_{jt} = \varepsilon$ if $x_{jt} \leq \varepsilon$, or in vector form $\boldsymbol{x} = \max(\boldsymbol{x}, \varepsilon)$. This means that we need to perform the following optimization problem:

$$\mathbf{A} = \arg\min_{\mathbf{A}\geq\boldsymbol{\varepsilon}} D_1(\mathbf{Y}_c||\mathbf{AX}_c), \quad \text{for fixed} \quad \mathbf{X}, \tag{3.4}$$

$$\mathbf{X} = \arg\min_{\mathbf{X}\geq\boldsymbol{\varepsilon}} D_2(\mathbf{Y}_r||\mathbf{A}_r\mathbf{X}), \quad \text{for fixed} \quad \mathbf{A}, \tag{3.5}$$

which lead in fact to Positive Matrix Factorization (PMF).

3.1 EXTENDED ISRA AND EMML ALGORITHMS: REGULARIZATION AND SPARSITY

The most popular algorithms for NMF belong to the class of multiplicative ISRA (Image Space Reconstruction Algorithm) [18, 20, 38, 26] and EMML (Expectation Maximization Maximum Likelihood) [22, 27, 19, 21, 37, 2, 3, 4, 31, 32, 42] update rules (also often referred to as Lee-Seung algorithms[3] [40, 41]). These classes of algorithms have a relative low complexity but are characterized by slow convergence and the risk of converging to spurious local minima. In

[2] Generally, we assume that we minimize sequentially one or two different cost functions with the same global minima, depending on statistical distributions of factor matrices. For simplicity, in this chapter we assume in the most cases that D_1 and D_2 are equal. However, in general the "mixture" or combination of updates rules are possible.

[3] However, since these algorithms have a long history, we refer to them as ISRA and EMML algorithms. They have been developed independently in many fields, including emission tomography, image restoration and astronomical imaging.

this section, we discuss extensions of this class of multiplicative NMF algorithms by imposing additional constraints such as sparsity and smoothness. Moreover, we discuss how to unify and generalize them and how to implement them for large-scale problems.

3.1.1 Multiplicative NMF Algorithms Based on the Squared Euclidean Distance

For $\mathbf{E} = \mathbf{Y} - \mathbf{AX}$ modeled as i.i.d. (independent identically distributed) white Gaussian noise, we can formulate the problem of estimating the matrices $\mathbf{A}$ and $\mathbf{X}$ as that of maximizing the likelihood function:

$$p(\mathbf{Y}|\mathbf{A}, \mathbf{X}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|\mathbf{Y} - \mathbf{AX}\|_F^2}{2\sigma^2}\right), \tag{3.6}$$

subject to $\mathbf{A} \geq \mathbf{0}$ and $\mathbf{X} \geq \mathbf{0}$, element-wise, where σ is the standard deviation of the Gaussian noise.

Maximizing the likelihood is equivalent to minimizing the corresponding negative log-likelihood function, or equivalently, the squared Frobenius norm

$$D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{AX}\|_F^2, \tag{3.7}$$
$$\text{subject to } \; a_{ij} \geq 0, \quad x_{jt} \geq 0, \quad \forall\, i, j, t.$$

Using the gradient descent approach and switching alternatively between the two sets of parameters, we obtain simple multiplicative update formulas (see derivation below):[4]

$$a_{ij} \leftarrow a_{ij} \frac{[\mathbf{Y}\,\mathbf{X}^T]_{ij}}{[\mathbf{A}\,\mathbf{X}\,\mathbf{X}^T]_{ij} + \varepsilon}, \tag{3.8}$$

$$x_{jt} \leftarrow x_{jt} \frac{[\mathbf{A}^T\,\mathbf{Y}]_{jt}}{[\mathbf{A}^T\mathbf{A}\,\mathbf{X}]_{jt} + \varepsilon}. \tag{3.9}$$

The above algorithm (3.8)-(3.9), called often Lee-Seung NMF algorithm can be considered as an extension of the well known ISRA algorithm proposed first by Daube-Witherspoon and Muehllehner [18] and investigated by many researchers, especially, De Pierro and Byrne [20, 19, 21, 37, 3, 42]. The above update rules can be written in compact matrix form as

$$\mathbf{A} \leftarrow \mathbf{A} \circledast \left[(\mathbf{Y}\mathbf{X}^T) \oslash (\mathbf{A}\mathbf{X}\mathbf{X}^T + \varepsilon)\right], \tag{3.10}$$

$$\mathbf{X} \leftarrow \mathbf{X} \circledast \left[(\mathbf{A}^T\mathbf{Y}) \oslash (\mathbf{A}^T\mathbf{A}\mathbf{X} + \varepsilon)\right], \tag{3.11}$$

where $\circledast$ is the Hadamard (components-wise) product and $\oslash$ is element-wise division between two matrices.

[4] Small positive constant ε is usually added to denominators to avoid division by zero.

Remark 3.1 *The ISRA NMF algorithm can be extended to weighted squared Euclidean norm (corresponding to colored Gaussian noise) by minimizing the cost functions*

$$D_{\mathbf{W}}(\mathbf{Y}_c\|\mathbf{A}\mathbf{X}_c) = \frac{1}{2}\,\mathrm{tr}(\mathbf{Y}_c - \mathbf{A}\mathbf{X}_c)^T\mathbf{W}_{\mathbf{A}}(\mathbf{Y}_r - \mathbf{A}\mathbf{X}_c) = \frac{1}{2}\|\mathbf{W}_2(\mathbf{Y}_c - \mathbf{A}\mathbf{X}_c)\|_F^2, \tag{3.12}$$

$$D_{\mathbf{W}}(\mathbf{Y}_r\|\mathbf{A}_r\mathbf{X}) = \frac{1}{2}\,\mathrm{tr}(\mathbf{Y}_r - \mathbf{A}_r\mathbf{X})^T\mathbf{W}_{\mathbf{X}}(\mathbf{Y}_r - \mathbf{A}_r\mathbf{X}) = \frac{1}{2}\|\mathbf{W}_1(\mathbf{Y}_r - \mathbf{A}_r\mathbf{X})\|_F^2, \tag{3.13}$$

where $\mathbf{W}_{\mathbf{A}} = \mathbf{W}_2^T\mathbf{W}_2$ *and* $\mathbf{W}_{\mathbf{X}} = \mathbf{W}_1^T\mathbf{W}_1$ *are symmetric positive-definite weighted matrices, thus giving*

$$\mathbf{A} \leftarrow \mathbf{A} \circledast \left[(\mathbf{W}_{\mathbf{A}}\mathbf{Y}_c\mathbf{X}_c^T) \oslash (\mathbf{W}_{\mathbf{A}}\mathbf{A}\mathbf{X}_c\mathbf{X}_c^T)\right]_+, \tag{3.14}$$

$$\mathbf{X} \leftarrow \mathbf{X} \circledast \left[(\mathbf{A}_r^T\mathbf{W}_{\mathbf{X}}\mathbf{Y}_r) \oslash (\mathbf{A}_r^T\mathbf{W}_{\mathbf{X}}\mathbf{A}_r\mathbf{X})\right]_+. \tag{3.15}$$

In practice, the columns of the matrix $\mathbf{A}$ *should be normalized to the unit* ℓ_p*-norm (typically,* $p = 1$*).*

The original ISRA algorithm is relatively slow, and many heuristic approaches have been proposed to speed it up. For example, a relaxation approach rises the multiplicative coefficients to some power $\omega \in (0, 2]$, that is,

$$a_{ij} \leftarrow a_{ij}\left(\frac{[\mathbf{Y}\,\mathbf{X}^T]_{ij}}{[\mathbf{A}\,\mathbf{X}\,\mathbf{X}^T]_{ij}}\right)^{\omega}, \tag{3.16}$$

$$x_{jt} \leftarrow x_{jt}\left(\frac{[\mathbf{A}^T\,\mathbf{Y}]_{jt}}{[\mathbf{A}^T\mathbf{A}\,\mathbf{X}]_{jt}}\right)^{\omega}, \tag{3.17}$$

in order to achieve faster convergence.

The above learning rules usually provide sparse nonnegative representations of the data, although they do not guarantee the sparsest possible solution (that is, that the solutions contain the largest possible number of zero elements of $\mathbf{X}$ and/or $\mathbf{A}$). Moreover, the solutions are not necessarily unique and the algorithms may converge to local minima. A much better performance (in the sense of convergence) may be achieved by using the multilayer NMF structure as explained in Chapter 1 [10, 15, 17, 11].

To understand the origin of the above update rules, consider the Karush–Kuhn–Tucker (KKT)[5] first-order optimality conditions for NMF [30]:

$$\mathbf{A} \geq \mathbf{0}, \qquad \mathbf{X} \geq \mathbf{0}, \tag{3.18}$$

$$\nabla_{\mathbf{A}}D_F \geq \mathbf{0}, \qquad \nabla_{\mathbf{X}}D_F \geq \mathbf{0}, \tag{3.19}$$

$$\mathbf{A} \circledast \nabla_{\mathbf{A}}D_F = \mathbf{0}, \qquad \mathbf{X} \circledast \nabla_{\mathbf{X}}D_F = \mathbf{0}, \tag{3.20}$$

[5]The Karush–Kuhn–Tucker conditions (also known as the Kuhn–Tucker or the KKT conditions) are contained in a system of equations and inequalities which the solution of a nonlinear programming problem must satisfy when the objective function and the constraint functions are differentiable. The KKT conditions are necessary for a solution in nonlinear programming to be optimal, provided some regularity conditions are satisfied. It is a generalization of the method of Lagrange multipliers to inequality constraints which provides a strategy for finding the minimum of a cost function subject to constraints.

Algorithm 3.5: Multiplicative Beta NMF with Over-relaxation and Sparsity Control

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data, J: rank of approximation, β: order of Beta divergence
ω: over-relaxation, and $\alpha_{\mathbf{A}}, \alpha_{\mathbf{X}}$: sparsity degrees
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ such that cost function (3.146) is minimized.

begin
  initialization for $\mathbf{A}$ and $\mathbf{X}$
  repeat /* update $\mathbf{X}$ and $\mathbf{A}$ */
    select R row indices
    $\mathbf{X} \leftarrow \mathbf{X} \circledast \left(\left[\mathbf{A}_r^T \left(\mathbf{Y}_r \circledast \hat{\mathbf{Y}}_r^{\cdot [\beta-1]} \right) - \alpha_{\mathbf{X}} \mathbf{1}_{J \times T} \right]_+ \oslash \left(\mathbf{A}_r^T \hat{\mathbf{Y}}_r^{\cdot [\beta]} \right) \right)^{\cdot [\omega]}$ /* $\hat{\mathbf{Y}}_r = \mathbf{A}_r \mathbf{X}$ */
    select C column indices
    $\mathbf{A} \leftarrow \mathbf{A} \circledast \left(\left[\left(\mathbf{Y}_c \circledast \hat{\mathbf{Y}}_c^{\cdot [\beta-1]} \right) \mathbf{X}_c^T - \alpha_{\mathbf{A}} \mathbf{1}_{I \times J} \right]_+ \oslash \left(\hat{\mathbf{Y}}_c^{\cdot [\beta]} \mathbf{X}_c^T \right) \right)^{\cdot [\omega]}$ /* $\hat{\mathbf{Y}}_c = \mathbf{A} \mathbf{X}_c$ */
    foreach $\boldsymbol{a}_j$ *of* $\mathbf{A}$ **do** $\boldsymbol{a}_j \leftarrow \boldsymbol{a}_j / \|\boldsymbol{a}_j\|_p$ /* normalize to ℓ_p unit length */
  until *a stopping criterion is met* /* convergence condition */
end

The Itakura-Saito distance is optimal for a Gamma distribution, in other words, it corresponds to maximum likelihood estimation using the Gamma likelihood function. This feature has been investigated recently by C. Févotte *et al.* [29, 30]. To illustrate this, consider the Gamma likelihood of order γ ($\gamma > 0$)

$$L(\mathbf{X}) = \prod_{it} \frac{z_{it}^{-\gamma} y_{it}^{\gamma-1} \exp(-y_{it}/z_{it})}{\Gamma(\gamma)}, \tag{3.159}$$

where $z_{it} = [\mathbf{AX}]_{it}/\gamma$.

The negative log-likelihood is then equal to

$$L_\Gamma(\mathbf{X}) = IT \ln(\Gamma(\gamma)) + \sum_{it} \left(\gamma \ln z_{it} - (\gamma - 1) \ln y_{it} + \frac{y_{it}}{z_{it}} \right). \tag{3.160}$$

Substituting z_{it} and noting that some terms in the above expression do not depend on $\mathbf{A}$ and $\mathbf{X}$, we obtain the Itakura-Saito divergence.

The minimization of the above cost function leads to the following algorithm [27]:

$$\mathbf{X} \leftarrow \mathbf{X} \circledast [(\mathbf{A}^T \mathbf{P}) \oslash (\mathbf{A}^T \mathbf{Q} + \varepsilon)]^{\cdot [\omega]}, \tag{3.161}$$

$$\mathbf{A} \leftarrow \mathbf{A} \circledast [(\mathbf{P} \mathbf{X}^T) \oslash (\mathbf{Q} \mathbf{X}^T + \varepsilon)]^{\cdot [\omega]}, \tag{3.162}$$

$$\mathbf{A} \leftarrow \mathbf{A} \ \operatorname{diag}(\|\boldsymbol{a}_1\|_1^{-1}, \|\boldsymbol{a}_2\|_1^{-1}, \ldots, \|\boldsymbol{a}_J\|_1^{-1}), \tag{3.163}$$

where $\omega \in (0.5, 1)$ is a relaxation parameter and

$$\mathbf{P} = \mathbf{Y} \oslash \hat{\mathbf{Y}}^{\cdot [2]}, \qquad \mathbf{Q} = \hat{\mathbf{Y}}^{\cdot [-1]}, \qquad \hat{\mathbf{Y}} = \mathbf{AX} + \varepsilon. \tag{3.164}$$

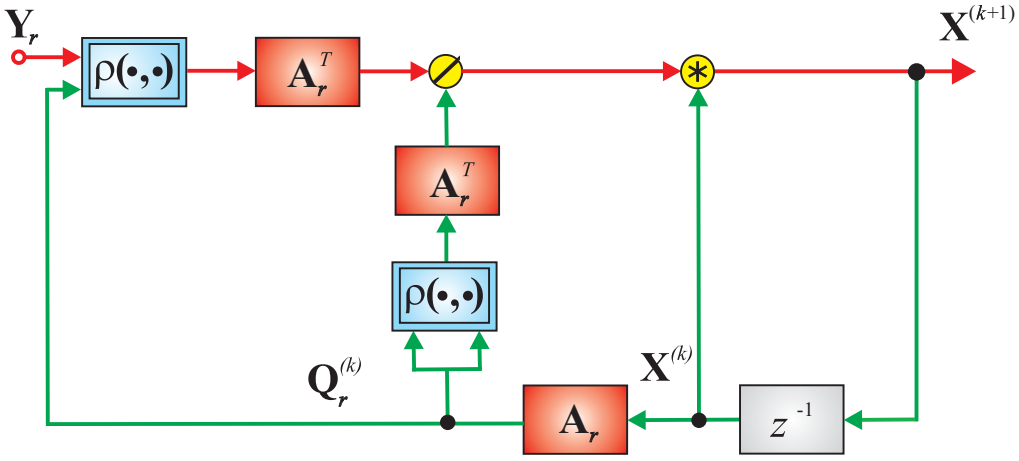


Fig. 3.9 Functional block diagram illustrating the generalized multiplicative Beta NMF algorithm for a large-scale NMF: $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} \circledast (\mathbf{A}_r^T \ \boldsymbol{\rho}(\mathbf{Y}_r, \mathbf{A}_r^T\mathbf{X}^{(k)})) \oslash (\mathbf{A}_r^T \ \boldsymbol{\rho}(\mathbf{A}_r\mathbf{X}^{(k)}, \mathbf{A}_r^T\mathbf{X}^{(k)}))$. A similar block diagram can be formulated for updating matrix $\mathbf{A}$ as: $\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} \circledast (\boldsymbol{\rho}(\mathbf{Y}_c, \mathbf{A}^{(k)}\mathbf{X}_c)\ \mathbf{X}_c^T) \oslash (\boldsymbol{\rho}(\mathbf{A}^{(k)}\mathbf{X}_c, \mathbf{A}^{(k)}\mathbf{X}_c)\ \mathbf{X}_c^T)$.

3.4.3 Generalized Multiplicative Beta Algorithm for NMF

The multiplicative Beta NMF algorithm can be generalized as (see also Figure 3.9) [38, 12, 10, 23]:

$$x_{jt} \leftarrow x_{jt} \frac{\sum_{i \in \mathcal{S}_I} a_{ij}\, \Psi(y_{it}, q_{it})}{\sum_{i \in \mathcal{S}_I} a_{ij}\, \Psi(q_{it}, q_{it})}, \qquad a_{ij} \leftarrow a_{ij} \frac{\sum_{t \in \mathcal{S}_T} x_{jt}\, \Psi(y_{it}, q_{it})}{\sum_{t \in \mathcal{S}_T} x_{jt}\, \Psi(q_{it}, q_{it})}, \tag{3.165}$$

where $q_{it} = [\mathbf{AX}]_{it}$, $\Psi(q,q)$ is a nonnegative nondecreasing function, and $\Psi(y,q)$ may take several different forms, for example:

1. $\Psi(y,q) = y, \quad \Psi(q,q) = q$;
2. $\Psi(y,q) = y/q, \quad \Psi(q,q) = 1$;
3. $\Psi(y,q) = y/q^{\beta}, \quad \Psi(q,q) = q^{1-\beta}$;
4. $\Psi(y,q) = y/(c+q), \quad \Psi(q,q) = q/(c+q)$.

Not all the generalized multiplicative NMF algorithms are expected to work well for any given set of functions and parameters. In practice, in order to ensure stability it is necessary to introduce a suitable scaling and/or a relaxation parameter .[15]

Our main objective here was to unify most existing multiplicative algorithms for the standard NMF problem and to show how to incorporate additional constraints such as sparsity and

[15] It is still an open question and active area of research to decide which algorithms are potentially most useful and practical.

References

1. J. Brown. Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

2. J. Browne and A. De Pierro. A row-action alternative to the EM algorithm for maximizing likelihoods in emission tomography. *IEEE Transactions on Medical Imaging*, 15:687–699, 1996.

3. C.L. Byrne. Accelerating the EMML algorithm and related iterative algorithms by rescaled block-iterative (RBI) methods. *IEEE Transactions on Image Processing*, IP-7:100–109, 1998.

4. C.L. Byrne. Choosing parameters in block-iterative or ordered subset reconstruction algorithms. *IEEE Transactions on Image Processing*, 14(3):321–327, 2005.

5. C.L. Byrne. *Signal Processing: A Mathematical Approach*. A.K. Peters, Publ., Wellesley, MA, 2005.

6. C. Caiafa and A. Cichocki. Slice Oriented Decomposition: A new tensor representation for 3-way data. *(submitted to Journal of Signal Processing)*, 2009.

7. S. Choi. Algorithms for orthogonal nonnegative matrix factorization. Hong-Kong, 2007. Proceedings of the International Joint Conference on Neural Networks (IJCNN-2008).

8. M. Chu and M.M. Lin. Low dimensional polytype approximation and its applications to nonnegative matrix factorization. *SIAM Journal on Scientific Computing*, 30:1131–1151, 2008.

9. A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing*. John Wiley & Sons Ltd, New York, 2003.

10. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer, LNAI-4029*, 4029:548–562, 2006.

11. A. Cichocki and R. Zdunek. NMFLAB for Signal and Image Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

12. A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: Family of new algorithms. *Springer, LNCS-3889*, 3889:32–39, 2006.

13. A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2006*, volume 5, pages 621–624, Toulouse, France, May 14–19 2006.

14. A. Cichocki, R. Zdunek, and S.-I. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. *Springer, Lecture Notes on Computer Science, LNCS-4666*, pages 169–176, 2007.

15. A. Cichocki, R. Zdunek, and S.-I. Amari. Nonnegative matrix and tensor factorization. *IEEE Signal Processing Magazine*, 25(1):142–145, 2008.

16. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using Alpha and Beta divergencies. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume III, pages 1393–1396, Honolulu, Hawaii, USA, April 15–20 2007.

17. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari. Novel multi-layer nonnegative tensor factorization with sparsity constraints. *Springer, LNCS-4432*, 4432:271–280, April 11–14 2007.

18. M.E. Daube-Witherspoon and G. Muehllehner. An iterative image space reconstruction algorithm suitable for volume ECT. *IEEE Transactions on Medical Imaging*, 5:61–66, 1986.

19. A. R. De Pierro. A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography. *IEEE Transactions on Medial Imaging*, 14(1):132–137, March 1995.

20. A.R. De Pierro. On the relation between the ISRA and the EM algorithm for positron emission tomography. *IEEE Transactions on Medial Imaging*, 12(2):328–333, June 1993.

21. A.R. De Pierro and M.E. Beleza Yamagishi. Fast iterative methods applied to tomography models with general gibbs priors. In *Proc. SPIE Conf. Mathematical Modeling, Bayesian Estimation and Inverse Problems*, volume 3816, pages 134–138, 1999.

22. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

23. I. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Neural Information Proc. Systems*, pages 283–290, Vancouver, Canada, December 2005.

24. C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *KDD06: Proceedings of the 12th ACM SIGKDD international conference on*

Knowledge Discovery and Data Mining, pages 126–135, New York, NY, USA, 2006. ACM Press.

25. D. Donoho and V. Stodden. When does nonnegative matrix factorization give a correct decomposition into parts? In *Neural Information Processing Systems*, volume 16. MIT press, 2003.

26. P.P.B. Eggermont and V.N. LaRiccia. Maximum smoothed likelihood density estimation for inverse problems. *Ann. Statist.*, 23(1):199–220, 1995.

27. P.P.B. Eggermont and V.N. LaRiccia. On EM-like algorithms for minimum distance estimation. Technical report, Mathematical Sciences, University of Delaware, 1998.

28. D. Ellis. Spectrograms: Constant-q (log-frequency) and conventional (linear), 05 2004.

29. C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the takura-Saito divergence. With application to music analysis. *Neural Computation*, 21(3):793–830, 2009.

30. C. Févotte and A. T. Cemgil. Nonnegative matrix factorizations as probabilistic inference in composite models. In *In Proc. 17th European Signal Processing Conference (EUSIPCO'09)*, Glagow, Scotland, Augustr 24–28 2009.

31. D. FitzGerald, M. Cranitch, and E. Coyle. Extended nonnegative tensor factorisation models for musical sound source separation. *Computational Intelligence and Neuroscience*, 2008.

32. N. Gillis and F. Glineur. Nonnegative factorization and maximum edge biclique problem. In *submitted*, 2009. http://www.uclouvain.be/en-44508.html.

33. P.J. Green. Bayesian reconstruction from emission tomography data using a modified EM algorithm. *IEEE Transactions on Medical Imaging*, 9:84–93, 1990.

34. T. Hebert and R. Leahy. A generalized EM algorithm for 3-D Bayesian reconstruction from Poisson data using Gibbs priors. *IEEE Transactions on Medical Imaging*, 8:194–202, 1989.

35. N.-D. Ho and P. Van Dooren. Nonnegative matrix factorization with fixed row and column sums. *Linear Algebra and its Applications*, 429(5-6):1020–1025, 2007.

36. L. Kaufman. Maximum likelihood, least squares, and penalized least squares for PET. *IEEE Transactions on Medical Imaging*, 12(2):200–214, 1993.

37. J. Kivinen and M.K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132, 1997.

38. R. Kompass. A generalized divergence measure for nonnegative matrix factorization. *Neural Computation*, 19(3):780–791, 2006.

39. H. Lantéri, M. Roche, and C. Aime. Penalized maximum likelihood image restoration with positivity constraints: multiplicative algorithms. *Inverse Problems*, 18:1397–1419, 2002.

40. H Lantéri, R. Soummmer, and C. Aime. Comparison between ISRA and RLA algorithms: Use of a Wiener filter based stopping criterion. *Astronomy and Astrophysics Supplemantary Series*, 140:235–246., 1999.

41. H. Laurberg and L.K. Hansen. On affine non-negative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2007*, pages 653–656, 2007.

42. D.D. Lee and H.S. Seung. Learning of the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

43. D.D. Lee and H.S. Seung. *Algorithms for Nonnegative Matrix Factorization*, volume 13. MIT Press, 2001.

44. R.M. Lewitt and G. Muehllehner. Accelerated iterative reconstruction for positron emission tomography based on the EM algorithm for maximum-likelihood estimation. *IEEE Transactions on Medical Imaging*, MI-5:16–22, 1986.

45. H. Li, T. Adali, W. Wang, D. Emge, and A. Cichocki. Non-negative matrix factorization with orthogonality constraints and its application to Raman spectroscopy. *The Journal of VLSI Signal Processing*, 48(1-2):83–97, 2007.

46. M. Mørup, M.N. Schmidt, and L.K. Hansen. Shift invariant sparse coding of image and music data. Technical report, 2008.

47. A. Ozerov and C. Févotte. Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. *IEEE Trans. Audio, Speech and Language Processing*, page (in print), 2010.

48. P. Sajda, S. Du, T.R. Brown, R. Stoyanova, D.C. Shungu, X. Mao, and L.C. Parra. Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain. *IEEE Transactions on Medical Imaging*, 23(12):1453–1465, 2004.

49. P. Sajda, S. Du, and L. Parra. Recovery of constituent spectra using non-negative matrix factorization. In *Proceedings of SPIE*, volume 5207, pages 321–331, 2003.

50. M.N. Schmidt. Single-channel source separation using non-negative matrix factorization, 2008.

51. N.N. Schraudolph. Gradient-based manipulation of nonparametric entropy estimates. *IEEE Transactions on Neural Networks*, 15(4):828–837, July 2004.

52. L. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, MI-1:113–122, 1982.

53. P. Smaragdis. Non-negative matrix factor deconvolution; Extraction of multiple sound sources from monophonic inputs. *Lecture Notes in Computer Science*, 3195:494–499, 2004.

54. P. Smaragdis. Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1):1–12, 2007.

55. L. Tong, R.-W. Liu, V.-C. Soon, and Y.-F. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38(5):499–509, 1991.

56. Y. Tsaig and D.L. Donoho. Extensions of compressed sensing. *Signal Processing*, 86(3):549–571, 2006.

57. W. Wang. Squared Euclidean distance based convolutive non-negative matrix factorization with multiplicative learning rules for audio pattern separation. In *Proc. 7th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2007)*, Cairo, Egypt, December 15-18 2007.

58. W. Wang, A. Cichocki, and J. Chambers. Note onset detection via nonnegative factorization of magnitude spectrum. *IEEE Transactions on Signal Processing*, page (in print), 2009.

59. W. Wang, Y. Luo, J. Chambers, and S. Sanei. Non-negative matrix factorization for note onset detection of audio signals. In *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2006)*, pages 447–452, Maynooth, Ireland, September 6–8 2006.

60. W. Wang, Y. Luo, J. Chambers, and S. Sanei. Note onset detection via nonnegative factorization of magnitude spectrum. *EURASIP Journal on Advances in Signal Processing*, ISSN 1687 6172, 2008.

61. K. Wilson, B. Raj, P. Smaragdis, and A. Divakaran. Speech denoising using non-negative matrix factorization with priors. In *In proceedings IEEE International Conference on Audio and Speech Signal Processing*, Las Vegas, Nevada, USA, April 2008.

4

Alternating Least Squares and Related Algorithms for NMF and SCA Problems

In this chapter we derive and overview Alternating Least Squares algorithms referred to as ALS algorithms for Nonnegative Matrix Factorization (NMF) and Sparse Component Analysis (SCA). This is important as many existing NMF/SCA techniques are prohibitively slow and inefficient, especially for very large-scale problems. For such problems a promising approach is to apply the ALS algorithms [47], [2]. Unfortunately, the standard ALS algorithm and its simple modifications suffer from unstable convergence properties, they often return suboptimal solutions, are quite sensitive with respect to noise, and can be relatively slow for nearly collinear data [47], [43], [2], [64].

As explained in Chapter 1 solutions obtained by NMF algorithms may not be unique, and to this end it is often necessary to impose additional constraints (which arise naturally from the data considered) such as sparsity or smoothness. Therefore, special emphasis in this chapter is put on various regularization and penalty terms together with local learning rules in which we update sequentially one-by-one vectors of factor matrices. By incorporating the regularization and penalty terms into the weighted Frobenius norm, we show that it is possible to achieve sparse, orthogonal, or smooth representations thus helping to obtain a desired global solution.

The main objective of this chapter is to develop efficient and robust regularized ALS (RALS) algorithms. For this purpose, we use several approaches from constrained optimization and regularization theory, and introduce in addition several heuristic algorithms. The algorithms are characterized by improved efficiency and very good convergence properties, especially for large-scale problems. The RALS and HALS algorithms were implemented in our NMFLAB/NTFLAB MATLAB Toolboxes, and compared with standard NMF algorithms [19]. Moreover, we have applied the ALS approach for semi-NMF, symmetric NMF, and NMF with orthogonality constraints.

4.1 STANDARD ALS ALGORITHM

Consider the standard NMF model, given by:[1]

$$\mathbf{Y} = \mathbf{AX} + \mathbf{E} = \mathbf{AB}^T + \mathbf{E}, \qquad \mathbf{A} \geq \mathbf{0} \ \text{ and } \ \mathbf{X} \geq \mathbf{0}. \tag{4.1}$$

The problem of estimating the nonnegative elements in $\mathbf{A}$ and $\mathbf{X}$ can be formulated as the minimization of the standard squared Euclidean distance (Frobenius norm):

$$D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}||\mathbf{Y} - \mathbf{AX}||_F^2 = \frac{1}{2}\,\mathrm{tr}(\mathbf{Y} - \mathbf{AX})^T(\mathbf{Y} - \mathbf{AX}), \tag{4.2}$$
$$\text{subject to } \ a_{ij} \geq 0, \quad x_{jt} \geq 0, \quad \forall\, i, j, t.$$

In such a case the basic approach is to perform the alternating minimization or alternating projection: the above cost function can be alternately minimized with respect to the two sets of parameters $\{x_{jt}\}$ and $\{a_{ij}\}$, each time optimizing one set of arguments while keeping the other one fixed [49], [13]. This corresponds to the following set of minimization problems:

$$\mathbf{A}^{(k+1)} = \arg\min_{\mathbf{A}} ||\mathbf{Y} - \mathbf{AX}^{(k)}||_F^2, \qquad \text{s.}t. \ \ \mathbf{A} \geq \mathbf{0}, \tag{4.3}$$

$$\mathbf{X}^{(k+1)} = \arg\min_{\mathbf{X}} ||\mathbf{Y}^T - \mathbf{X}^T[\mathbf{A}^{(k+1)}]^T||_F^2, \qquad \text{s.}t. \ \ \mathbf{X} \geq \mathbf{0}. \tag{4.4}$$

Instead of applying the gradient descent technique, we rather estimate directly the stationary points and thereby exploit the fixed point approach. According to the Karush-Kuhn-Tucker (KKT) optimality conditions, $\mathbf{A}^*$ and $\mathbf{X}^*$ are stationary points of the cost function (4.2) if and only if

$$\mathbf{A}^* \geq \mathbf{0}, \qquad \mathbf{X}^* \geq \mathbf{0}, \tag{4.5}$$

$$\nabla_{\mathbf{A}} D_F(\mathbf{Y}||\mathbf{A}^*\mathbf{X}^*) = \mathbf{A}^*\mathbf{X}^*\mathbf{X}^{*T} - \mathbf{Y}\mathbf{X}^{*T} \geq \mathbf{0}, \qquad \mathbf{A} \circledast \nabla_{\mathbf{A}} D_F(\mathbf{Y}||\mathbf{A}^*\mathbf{X}^*) = \mathbf{0}, \tag{4.6}$$

$$\nabla_{\mathbf{X}} D_F(\mathbf{Y}||\mathbf{A}^*\mathbf{X}^*) = \mathbf{A}^{*T}\mathbf{A}^*\mathbf{X}^* - \mathbf{A}^{*T}\mathbf{Y} \geq \mathbf{0}, \qquad \mathbf{X} \circledast \nabla_{\mathbf{X}} D_F(\mathbf{Y}||\mathbf{A}^*\mathbf{X}^*) = \mathbf{0}. \tag{4.7}$$

Assuming that the factor matrices $\mathbf{A}$ and $\mathbf{X}$ are positive (with zero entries replaced by e.g., $\varepsilon = 10^{-9}$), the stationary points can be found by equating the gradient components to zero:

$$\nabla_{\mathbf{A}} D_F(\mathbf{Y}||\mathbf{AX}) = \frac{\partial D_F(\mathbf{Y}||\mathbf{AX})}{\partial \mathbf{A}} = [-\mathbf{YX}^T + \mathbf{AXX}^T] = \mathbf{0}, \tag{4.8}$$

$$\nabla_{\mathbf{X}} D_F(\mathbf{Y}||\mathbf{AX}) = \frac{\partial D_F(\mathbf{Y}||\mathbf{AX})}{\partial \mathbf{X}} = [-\mathbf{A}^T\mathbf{Y} + \mathbf{A}^T\mathbf{AX}] = \mathbf{0} \tag{4.9}$$

or equivalently in a scalar form

$$\frac{\partial D_F(\mathbf{Y}||\mathbf{AX})}{\partial a_{ij}} = [-\mathbf{YX}^T + \mathbf{AXX}^T]_{ij} = 0, \qquad \forall ij, \tag{4.10}$$

$$\frac{\partial D_F(\mathbf{Y}||\mathbf{AX})}{\partial x_{jt}} = [-\mathbf{A}^T\mathbf{Y} + \mathbf{A}^T\mathbf{AX}]_{jt} = 0, \qquad \forall jt. \tag{4.11}$$

[1] We use a simplified notation: $\mathbf{A} \geq \mathbf{0}$ which is used to denote component-wise relations, that is, $a_{ij} \geq 0$.

Assuming that the estimated components are nonnegative we obtain the simple nonnegative ALS update rules:

$$\mathbf{A} \leftarrow \left[\mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\right]_+ = \left[\mathbf{Y}\mathbf{X}^\dagger\right]_+ , \tag{4.12}$$

$$\mathbf{X} \leftarrow \left[(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}\right]_+ = \left[\mathbf{A}^\dagger\mathbf{Y}\right]_+ , \tag{4.13}$$

where $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudo inverse, and $[x]_+ = \max\{\varepsilon, x\}$ is a half-wave rectifying nonlinear projection to enforce nonnegativity or strictly speaking positive constraints.

In the special case, for a symmetric NMF model, given by

$$\mathbf{Y} = \mathbf{A}\mathbf{A}^T, \tag{4.14}$$

where $\mathbf{Y} \in \mathbb{R}^{I\times I}$ is a symmetric nonnegative matrix, and $\mathbf{A} \in \mathbb{R}^{I\times J}$ with $I \geq J$, we obtain a simplified algorithm

$$\mathbf{A} \leftarrow \left[\mathbf{Y}\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\right]_+ = \left[\mathbf{Y}[\mathbf{A}^T]^\dagger\right]_+ , \tag{4.15}$$

subject to additional normalization of the columns of matrix $\mathbf{A}$.

It is interesting to note that the modified ALS algorithm (4.12) – (4.13) can also be derived from Newton's method based on the second-order gradient descent approach, that is, based not only on the gradient but also on the Hessian.[2] Applying the gradient descent approach, we have

$$\text{vec}\,(\mathbf{X}) \leftarrow \left[\text{vec}\,(\mathbf{X}) - \boldsymbol{\eta}_\mathbf{X}\text{vec}\,(\nabla_\mathbf{X} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}))\right]_+ , \tag{4.16}$$

where $\boldsymbol{\eta}_\mathbf{X}$ is no longer a positive scalar, but a symmetric positive-definite matrix comprising the learning rates, defined as:

$$\boldsymbol{\eta}_\mathbf{X} = \eta_0(\nabla^2_\mathbf{X} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}))^{-1}, \tag{4.17}$$

where $\eta_0 \leq 1$ (typically $\eta_0 = 1$). The gradient and Hessian of cost function (4.2) with respect to $\mathbf{X}$ are given by

$$\nabla_\mathbf{X} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \mathbf{A}^T\mathbf{A}\mathbf{X} - \mathbf{A}\mathbf{Y}, \tag{4.18}$$

$$\nabla^2_\mathbf{X} D_F(\mathbf{Y}||\mathbf{A}\mathbf{X}) = \mathbf{I}_T \otimes \mathbf{A}^T\mathbf{A}. \tag{4.19}$$

Hence, we obtain the learning rule for $\mathbf{X}$:

$$\text{vec}\,(\mathbf{X}) \leftarrow \left[\text{vec}\,(\mathbf{X}) - \eta_0 \left(\mathbf{I}_T \otimes \mathbf{A}^T\mathbf{A}\right)^{-1} \text{vec}\left(\mathbf{A}^T\mathbf{A}\mathbf{X} - \mathbf{A}\mathbf{Y}\right)\right]_+ \tag{4.20}$$

$$= \left[\text{vec}\,(\mathbf{X}) - \eta_0\,\text{vec}\left(\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\mathbf{A}\mathbf{X} - \mathbf{A}\mathbf{Y}\right)\right)\right]_+ \tag{4.21}$$

or in the matrix form as

$$\mathbf{X} \leftarrow \left[(1-\eta_0)\mathbf{X} + \eta_0(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}\right]_+ . \tag{4.22}$$

[2] See details in Chapter 6.

In a similar way, assuming that

$$\boldsymbol{\eta}_{\mathbf{A}} = \eta_0(\nabla^2_{\mathbf{A}} D_F(\mathbf{Y}\|\mathbf{AX}))^{-1} = \eta_0(\mathbf{XX}^T \otimes \mathbf{I}_I)^{-1}, \tag{4.23}$$

we obtain:

$$\text{vec}\,(\mathbf{A}) \leftarrow \left[\text{vec}\,(\mathbf{A}) - \eta_0 \left(\left(\mathbf{XX}^T\right)^{-1} \otimes \mathbf{I}_I\right) \text{vec}\left(\mathbf{AXX}^T - \mathbf{YX}^T\right)\right]_+, \tag{4.24}$$

$$= \left[\text{vec}\,(\mathbf{A}) - \eta_0\, \text{vec}\left(\left(\mathbf{AXX}^T - \mathbf{YX}^T\right)\left(\mathbf{XX}^T\right)^{-1}\right)\right]_+, \tag{4.25}$$

or in the matrix form as

$$\mathbf{A} \leftarrow \left[(1-\eta_0)\mathbf{A} + \eta_0 \mathbf{YX}^T(\mathbf{XX}^T)^{-1}\right]_+ . \tag{4.26}$$

For $\eta_0 = 1$ the above updating rules simplify to the standard ALS illustrating that the ALS algorithm is in fact the Newton method with a relatively good convergence rate since it exploits information not only about gradient but also Hessian.

The main problem with the standard ALS algorithm is that it often cannot escape local minima. In order to alleviate this problem we introduce additional regularization and/or penalty terms and introduce novel cost functions to derive local hierarchical ALS (HALS) algorithms.

4.1.1 Multiple Linear Regression – Vectorized Version of ALS Update Formulas

The minimization problems (4.12)– (4.13) can also be formulated using multiple linear regression by vectorizing matrices, leading to the minimization of the following two cost functions:

$$\min_{\boldsymbol{x}\geq 0} \|\boldsymbol{y} - \bar{\mathbf{A}}\boldsymbol{x}\|_2^2 \tag{4.27}$$

$$\min_{\boldsymbol{a}\geq 0} \|\bar{\boldsymbol{y}} - \bar{\mathbf{X}}\boldsymbol{a}\|_2^2, \tag{4.28}$$

where

$$\begin{aligned} &\boldsymbol{y} = \text{vec}(\mathbf{Y}) \in \mathbb{R}^{IT}, && \bar{\boldsymbol{y}} = \text{vec}(\mathbf{Y}^T) \in \mathbb{R}^{IT}, \\ &\boldsymbol{x} = \text{vec}(\mathbf{X}) \in \mathbb{R}^{JT}, && \bar{\boldsymbol{a}} = \text{vec}(\mathbf{A}^T) \in \mathbb{R}^{IJ}, \\ &\bar{\mathbf{A}} = \text{diag}\{\mathbf{A}, \mathbf{A}, \ldots, \mathbf{A}\} \in \mathbb{R}^{IT\times JT}, && \bar{\mathbf{X}} = \text{diag}\{\mathbf{X}^T, \mathbf{X}^T, \ldots, \mathbf{X}^T\} \in \mathbb{R}^{IJ\times JT}. \end{aligned}$$

The solution to the above optimization problem can be expressed as:

$$\boldsymbol{x} \leftarrow \left[(\bar{\mathbf{A}}^T\bar{\mathbf{A}})^{-1}\bar{\mathbf{A}}^T\boldsymbol{y}\right]_+ , \tag{4.29}$$

$$\bar{\boldsymbol{a}} \leftarrow \left[(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\bar{\boldsymbol{y}}\right]_+ . \tag{4.30}$$

Such representations are not computationally optimal, since for large-scale problems the block-diagonal matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{X}}$ are very large-scale, which makes the inversion of these matrices in each iteration step very time consuming.

normalization after each iterative step, to give a simplified scalar form of the HALS algorithm:

$$b_{tj} \leftarrow \left[\sum_{i=1}^{I} a_{ij}\, y_{it}^{(j)}\right]_+, \qquad a_{ij} \leftarrow \left[\sum_{t=1}^{T} b_{tj}\, y_{it}^{(j)}\right]_+, \tag{4.93}$$

with $a_{ij} \leftarrow a_{ij}/\|\boldsymbol{a}_j\|_2$, where $y_{it}^{(j)} = [\mathbf{Y}^{(j)}]_{it} = y_{it} - \sum_{p\neq j} a_{ip} b_{tp}$.

4.7.2 Extensions and Implementations of the HALS Algorithm

The above simple algorithm can be further extended or improved with respect to the convergence rate and performance by imposing additional constraints such as sparsity and smoothness.

Firstly, observe that the residual matrix $\mathbf{Y}^{(j)}$ can be rewritten as

$$\begin{aligned}\mathbf{Y}^{(j)} &= \mathbf{Y} - \sum_{p\neq j} \boldsymbol{a}_p \boldsymbol{b}_p^T = \mathbf{Y} - \mathbf{A}\mathbf{B}^T + \boldsymbol{a}_j \boldsymbol{b}_j^T, \\ &= \mathbf{Y} - \mathbf{A}\mathbf{B}^T + \boldsymbol{a}_{j-1}\boldsymbol{b}_{j-1}^T - \boldsymbol{a}_{j-1}\boldsymbol{b}_{j-1}^T + \boldsymbol{a}_j \boldsymbol{b}_j^T. \end{aligned} \tag{4.94}$$

It then follows that instead of computing explicitly the residual matrix $\mathbf{Y}^{(j)}$ at each iteration step, we can just perform a smart update [55]. An efficient implementation of the HALS algorithm (4.92) is given in the detailed pseudo-code in Algorithm 4.2.

Algorithm 4.2: HALS

Input: $\mathbf{Y} \in \mathbb{R}_+^{I\times T}$: input data, J: rank of approximation
Output: $\mathbf{A} \in \mathbb{R}_+^{I\times J}$ and $\mathbf{X} = \mathbf{B}^T \in \mathbb{R}_+^{J\times T}$ such that the cost function (4.85) is minimized.

```
begin
    ALS or random nonnegative initialization for A and X = B^T
    foreach a_j of A do  a_j ← a_j/||a_j||_2        /* normalize to ℓ2 unit length */
    E = Y − AB^T                                    /* residue */
    repeat
        for j = 1 to J do
            Y^(j) ← E + a_j b_j^T
            b_j ← [Y^(j)T a_j]_+                    /* update b_j */
            a_j ← [Y^(j) b_j]_+                     /* update a_j */
            a_j ← a_j/||a_j||_2
            E ← Y^(j) − a_j b_j^T                   /* update residue */
        end
    until a stopping criterion is met               /* convergence condition */
end
```

Different cost functions can be used for the estimation of the rows of the matrix $\mathbf{X} = \mathbf{B}^T$ and the columns of matrix $\mathbf{A}$ (possibly with various additional regularization terms [24], [21], [55]). Furthermore, the columns of $\mathbf{A}$ can be estimated simultaneously, and the rows in $\mathbf{X}$ sequentially. In other words, by minimizing the set of cost functions in (4.85) with respect to $\boldsymbol{b}_j$, and simultaneously the cost function (4.2) with normalization of the columns $\boldsymbol{a}_j$ to unit

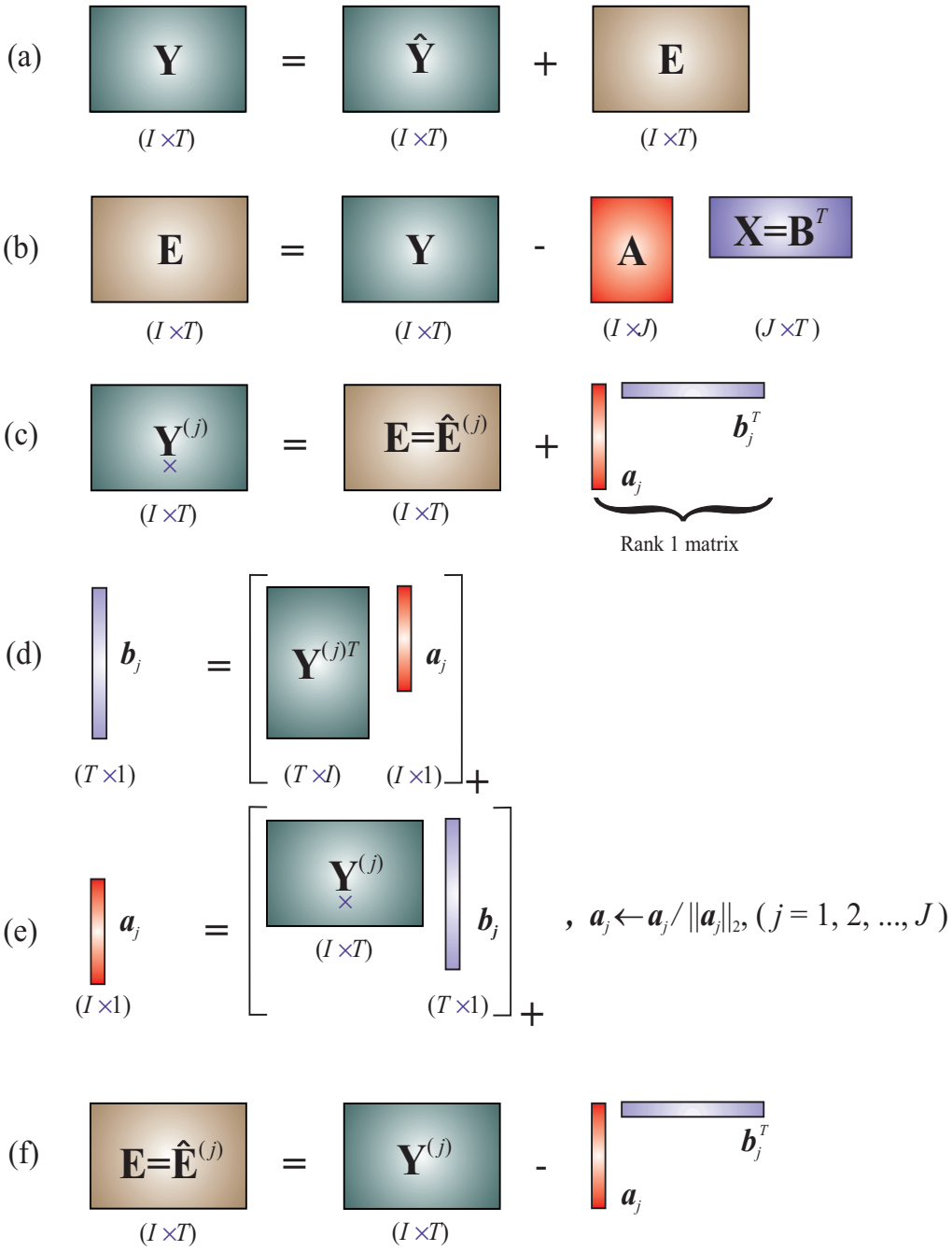


Fig. 4.1 Illustration of the basic (local) HALS algorithm and its comparison with the standard (global) ALS algorithm. In the standard ALS algorithm we minimize the mean squared error of the cost function $\|\mathbf{E}\|_F^2 = \|\mathbf{Y} - \hat{\mathbf{Y}}\|_F^2$, where $\hat{\mathbf{Y}} = \mathbf{A}\mathbf{B}^T$ and the target (desired) data $\mathbf{Y}$ is known and fixed (Figures (a)-(b)). In the HALS algorithm the targets residual matrices $\mathbf{Y}^{(j)}$, $(j = 1, 2, \ldots, J)$ (Figures (c),(f)) are not fixed but they are estimated during an iterative process via the HALS updates (Figures (d)-(e)) and they converge to rank-one matrices.

ℓ_2-norm, we obtain a very efficient NMF learning algorithm in which the individual vectors of $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J]$ are updated locally (column-by-column) and the matrix $\mathbf{A}$ is updated globally using the global nonnegative ALS (all columns $\boldsymbol{a}_j$ simultaneously) (see also [21]):

$$\boldsymbol{b}_j \leftarrow \left[\mathbf{Y}^{(j)\,T}\,\boldsymbol{a}_j\right]_+, \qquad \mathbf{A} \leftarrow \left[\mathbf{Y}\mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\right]_+ = \left[\mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\right]_+, \tag{4.95}$$

with the normalization (scaling) of the columns in $\mathbf{A}$ to unit length in the sense of the ℓ_2-norm after each iteration.

4.7.3 Fast HALS NMF Algorithm for Large-Scale Problems

Alternatively, an even more efficient approach is to perform a factor-by-factor procedure instead of updating column-by-column vectors [55]. This way, from (4.92), we obtain the following update rule for $\boldsymbol{b}_j = \underline{\boldsymbol{x}}_j^T$

$$\begin{aligned}
\boldsymbol{b}_j &\leftarrow \mathbf{Y}^{(j)T}\boldsymbol{a}_j/(\boldsymbol{a}_j^T\boldsymbol{a}_j) = \left(\mathbf{Y} - \mathbf{A}\mathbf{B}^T + \boldsymbol{a}_j\boldsymbol{b}_j^T\right)^T \boldsymbol{a}_j/(\boldsymbol{a}_j^T\boldsymbol{a}_j) \\
&= (\mathbf{Y}^T\boldsymbol{a}_j - \mathbf{B}\mathbf{A}^T\boldsymbol{a}_j + \boldsymbol{b}_j\boldsymbol{a}_j^T\boldsymbol{a}_j)/(\boldsymbol{a}_j^T\boldsymbol{a}_j) \\
&= \left(\left[\mathbf{Y}^T\mathbf{A}\right]_j - \mathbf{B}\left[\mathbf{A}^T\mathbf{A}\right]_j + \boldsymbol{b}_j\boldsymbol{a}_j^T\boldsymbol{a}_j\right)/(\boldsymbol{a}_j^T\boldsymbol{a}_j),
\end{aligned} \tag{4.96}$$

with the nonlinear projection $\boldsymbol{b}_j \leftarrow \left[\boldsymbol{b}_j\right]_+$ at each iteration step to impose the nonnegativity constraints. Since $\|\boldsymbol{a}\|_2^2 = 1$, the learning rule for $\boldsymbol{b}_j$ has a simplified form as

$$\boldsymbol{b}_j \leftarrow \left[\boldsymbol{b}_j + \left[\mathbf{Y}^T\mathbf{A}\right]_j - \mathbf{B}\left[\mathbf{A}^T\mathbf{A}\right]_j\right]_+, \tag{4.97}$$

and analogously, for vector $\boldsymbol{a}_j$:

$$\begin{aligned}
\boldsymbol{a}_j &\leftarrow \mathbf{Y}^{(j)}\,\boldsymbol{b}_j = \left(\mathbf{Y} - \mathbf{A}\mathbf{B}^T + \boldsymbol{a}_j\boldsymbol{b}_j^T\right)\boldsymbol{b}_j \\
&= \mathbf{Y}\boldsymbol{b}_j - \mathbf{A}\mathbf{B}^T\boldsymbol{b}_j + \boldsymbol{a}_j\boldsymbol{b}_j^T\boldsymbol{b}_j \\
&= [\mathbf{Y}\mathbf{B}]_j - \mathbf{A}\left[\mathbf{B}^T\mathbf{B}\right]_j + \boldsymbol{a}_j\boldsymbol{b}_j^T\boldsymbol{b}_j \\
&= \boldsymbol{a}_j\boldsymbol{b}_j^T\boldsymbol{b}_j + [\mathbf{Y}\mathbf{B}]_j - \mathbf{A}\left[\mathbf{B}^T\mathbf{B}\right]_j.
\end{aligned} \tag{4.98}$$

Hence, by imposing the nonnegativity constraints, we finally have

$$\boldsymbol{a}_j \leftarrow \left[\boldsymbol{a}_j\boldsymbol{b}_j^T\boldsymbol{b}_j + [\mathbf{Y}\mathbf{B}]_j - \mathbf{A}\left[\mathbf{B}^T\mathbf{B}\right]_j\right]_+, \tag{4.99}$$

$$\boldsymbol{a}_j \leftarrow \boldsymbol{a}_j/\|\boldsymbol{a}_j\|_2. \tag{4.100}$$

Based on these expressions, the improved and modified HALS NMF algorithm is given in the pseudo-code Algorithm 4.3.

The NMF problem is often highly redundant for $I >> J$, thus, for large-scale problems in order to estimate the vectors $\boldsymbol{a}_j$ and $\boldsymbol{b}_j = \underline{\boldsymbol{x}}_j^T \;\; \forall j$, we can use only some selected vectors and/or rows of the data input matrix $\mathbf{Y}$. For large-scale data and a block-wise update strategy (see

Chapter 1), the fast HALS learning rule for $\boldsymbol{b}_j$ (4.96) can be rewritten as follows

$$\begin{aligned}\boldsymbol{b}_j &\leftarrow \left[\boldsymbol{b}_j + \left[\mathbf{Y}_r^T \mathbf{A}_r\right]_j / \|\tilde{\boldsymbol{a}}_j\|_2^2 - \mathbf{B}\left[\mathbf{A}_r^T \mathbf{A}_r\right]_j / \|\tilde{\boldsymbol{a}}_j\|_2^2\right]_+ \\ &= \left[\boldsymbol{b}_j + \left[\mathbf{Y}_r^T \mathbf{A}_r \mathbf{D}_{A_r}\right]_j - \mathbf{B}\left[\mathbf{A}_r^T \mathbf{A}_r \mathbf{D}_{A_r}\right]_j\right]_+, \end{aligned} \tag{4.101}$$

where $\mathbf{D}_{A_r} = \text{diag}(\|\tilde{\boldsymbol{a}}_1\|_2^{-2}, \|\tilde{\boldsymbol{a}}_2\|_2^{-2}, \ldots, \|\tilde{\boldsymbol{a}}_J\|_2^{-2})$ is a diagonal matrix, and $\tilde{\boldsymbol{a}}_j$ is the j-th column vector of the reduced matrix $\mathbf{A}_r \in \mathbb{R}_+^{R \times J}$.

The update rule for $\boldsymbol{a}_j$ takes a similar form

$$\boldsymbol{a}_j \leftarrow \left[\boldsymbol{a}_j + [\mathbf{Y}_c \mathbf{B}_c \mathbf{D}_{B_c}]_j - \mathbf{A}\left[\mathbf{B}_c^T \mathbf{B}_c \mathbf{D}_{B_c}\right]_j\right]_+, \tag{4.102}$$

where $\mathbf{D}_{B_c} = \text{diag}(\|\tilde{\boldsymbol{b}}_1\|_2^{-2}, \|\tilde{\boldsymbol{b}}_2\|_2^{-2}, \ldots, \|\tilde{\boldsymbol{b}}_J\|_2^{-2})$ and $\tilde{\boldsymbol{b}}_j$ is the j-th column vector of the reduced matrix $\mathbf{B}_c = \mathbf{X}_c^T \in \mathbb{R}_+^{C \times J}$.

Algorithm 4.3: FAST HALS for Large Scale NMF

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data, J: rank of approximation
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} = \mathbf{B}^T \in \mathbb{R}_+^{J \times T}$ such that the cost function (4.85) is minimized.

```
begin
    ALS or random nonnegative initialization for A and X = B^T
    foreach a_j of A do  a_j ← a_j/||a_j||_2        /* normalize to ℓ2 unit length */
    repeat
        W = Y^T A; V = A^T A
        for j = 1 to J do
            b_j ← [b_j + w_j − B v_j]_+                          /* update b_j */
        end
        P = YB; Q = B^T B
        for j = 1 to J do
            a_j ← [a_j q_jj + p_j − A q_j]_+                     /* update a_j */
            a_j ← a_j/||a_j||_2
        end
    until a stopping criterion is met                  /* convergence condition */
end
```

In order to estimate all the vectors $\boldsymbol{a}_j$ and $\underline{\boldsymbol{x}}_j$ we only need to take into account the selected rows and columns of the residual matrices $\mathbf{Y}^{(j)}$ and the input data matrix $\mathbf{Y}$. To estimate precisely all $\boldsymbol{a}_j, \underline{\boldsymbol{x}}_j$, $\forall j$ we need to select at least J rows and columns of $\mathbf{Y}$. Moreover, the computations are performed only for the nonzero elements, thus allowing the computation time to be dramatically reduced for sparse and very large-scale problems. The rows and columns of the data matrix $\mathbf{Y}$ can be selected using different criteria. For example, we can choose only those rows and columns which provide the highest normalized squared Euclidean norms. Alternatively, instead of removing completely some rows and/or columns of $\mathbf{Y}$, we can merge (collapse) them into some clusters by adding them together or computing their averages. In this case we can select the rows and columns uniformly. Recently, extensive research is performed how to choose the optimal number of rows and vectors of data matrix [42], [7], [32], [50], [51], [10].

Appendix 4.D. MATLAB Source Code for HALS CS Algorithm

```
function X = CS_HALS(Y,options)
%
% Hierarchical ALS algorithm for Compressed sensing
% using linear and nonlinear thresholding techniques
% with Hard, Soft, Non negative garrotte, Abramovich,
% N-degree garrotte shrinkage rules
%
% INPUT
% Y        :   T compressed signals of I samples as columns (I × T)
% options  :   struct of optional parameters
%   .A     :   projection matrix
%   .Xinit :   initialization for sources X, zeros matrix is default
%   .J     :   number of samples of original source X
%   .Niter :   number of iteration                                  (1000)
%   .Nrst  :   number of restart                                      (1)
%   .Lmax  :   maximum threshold of lambda                           (20)
%   .Lmin  :   minimum threshold of lambda                            (0)
%   .psitype:  decreasing funtion type for threshold λ                (2)
%              1. shape-preserving piecewise cubic Hermite interpolation
%              (2). exponential function
%   .Hshape:   initial points (at least two)(x,y) y = ψ(x) forming
%              the decreaseing line used for shape-preserving
%              Hermite interpolation.
%              Ex. 2 points (0.2,0.8) and (0.4, 0.2)
%              Hshape = [0.2 .8; .4  .2]
%   .betarate: decreasing speed for exponential decreasing strategy    (4)
%   .shrinkage: shrinkage rule                                         (1)
%              (1). HARD                    2. SOFT
%              3. Non negative garrotte     4  Abramovich's rule
%              5. N-degree garrotte
% OUTPUTS:
% X       :    reconstructed signals (columns)
%
% Copyright 2008 by A.H. Phan and A. Cichocki
% 04/2008
% ########################################################################
%%
[I,T]=size(Y);
defoptions = struct('A',[],'J',I,'Niter',300,...
    'Xinit',[],'psitype','exp','betarate',4,'Hshape',[.2 .8; .8 .2],...
    'Lmax',20,'Lmin',0,'shrinkage',1,'nbrestart',1);
if ~exist('options','var')
    options = struct;
end
[A,J,Niter,X,psitype,betarate,Hshape,Lmax,Lmin,shrinkrule,Nrst] = ...
    scanparam(defoptions,options);
if isempty(X)
    X = zeros(J,T);
end
Hshape = [0 1;Hshape;1 0];
tol = 1e-5;
alpha = .6;                                          % reduction rate
%% Normalization of initial guess
normA = sqrt(sum(A.^2,1));
A = bsxfun(@rdivide,A,normA);
G = A'*A;
```

References

1. F. Abramovich, T. Sapatinas, and B. Silverman. Wavelet thresholding via a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(4):725–749, 1998.

2. R. Albright, J. Cox, D. Duling, A. N. Langville, and C. D. Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. Technical report, NCSU Technical Report Math 81706, 2006.

3. M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. In *ICML-2008*, Helsinki, July 2008.

4. J. Bobin, Y. Moudden, J. Fadili, and J.L. Starck. Morphological diversity and sparsity in blind source separation. In *Proc. ICA-2007*, volume 4666 of *Lecture Notes in Computer Science*, pages 349–356. Springer, 2007.

5. J. Bobin, J.L. Starck, J. Fadili, Y. Moudden, and D.L. Donoho. Morphological component analysis: An adaptive thresholding strategy. *IEEE Transactions on Image Processing*, 16(11):2675–2681, 2007.

6. J. Bobin, J.L. Starck, and R. Ottensamer. Compressed sensing in astronomy. *Submitted to IEEE Journal on Selected Topics in Signal Processing*, February 2008.

7. C. Boutsidis, M.W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proc. 20-th Annual SODA*, pages 968–977, USA, 2009.

8. L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24(6):2350–2383, 1996.

9. J. Buckheit and D. Donoho. Wavelab and reproducible research. Stanford University, Stanford, CA 94305, USA, 1995.

10. C. Caiafa and A. Cichocki. CUR decomposition with optimal selection of rows and columns. *(submitted)*, 2009.

11. E. J. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, February 2006.

12. R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. In *in 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.

13. A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing*. John Wiley & Sons Ltd, New York, 2003.

14. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer, LNAI-4029*, 4029:548–562, 2006.

15. A. Cichocki and A.H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE (invited paper)*, March 2009.

16. A. Cichocki, A.H. Phan, and C. Caiafa. Flexible HALS algorithms for sparse non-negative matrix/tensor factorization. In *Proc. of 18-th IEEE workshops on Machine Learning for Signal Processing*, Cancun, Mexico, 16–19, October 2008.

17. A. Cichocki, A.H. Phan, R. Zdunek, and L.-Q. Zhang. Flexible component analysis for sparse, smooth, nonnegative coding or representation. In *Lecture Notes in Computer Science, LNCS-4984*, volume 4984, pages 811–820. Springer, 2008.

18. A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization. *Electronics Letters*, 42(16):947–948, 2006.

19. A. Cichocki and R. Zdunek. NMFLAB for Signal and Image Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

20. A. Cichocki and R. Zdunek. NTFLAB for Signal Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

21. A. Cichocki and R. Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorizations. *Springer, LNCS-4493*, 4493:793–802, June 3–7 2007.

22. A. Cichocki, R. Zdunek, and S.-I. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. *Springer, Lecture Notes on Computer Science, LNCS-4666*, pages 169–176, 2007.

23. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using Alpha and Beta divergencies. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume III, pages 1393–1396, Honolulu, Hawaii, USA, April 15–20 2007.

24. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari. Novel multi-layer nonnegative tensor factorization with sparsity constraints. *Springer, LNCS-4432*, 4432:271–280, April 11–14 2007.

25. R.R. Coifman and D.L. Donoho. Translation-invariant de-noising. In *Wavelets and Statistics*, pages 125–150. Springer-Verlag, 1995.

26. I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

27. A. de Juan and R. Tauler. Chemometrics applied to unravel multicomponent processes and mixtures: Revisiting latest trends in multivariate resolution. *Analitica Chemica Acta*, 500:195–210, 2003.

28. D.L. Donoho, I. Drori, V. Stodden, and Y. Tsaig. Sparselab. *http://sparselab.stanford.edu/*, December 2005.

29. D.L. Donoho and I.M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *J. Am. Stat. Assoc*, 90(432):1200–1224, 1995.

30. D.L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In *Proceedings of the Conference on Information Sciences and Systems*.

31. D.L. Donoho and Y. Tsaig. Sparse solution of underdetermined linear equations. by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, March 2006. submitted.

32. P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.

33. M.F. Duarte, S. Sarvotham, D. Baron, M.B. Wakin, and R.G. Baraniuk. Distributed compressed sensing of jointly sparse signals. In *Proc. 39th Asilomar Conf. Signals, Sys., Comput.*, pages 1537– 1541, 2005.

34. M. Elad. Why simple shrinkage is still relevant for redundant representations? *IEEE Transactions On Information Theory*, 52:5559–5569, 2006.

35. M. Elad. Optimized projections for compressed sensing. *IEEE Transactions on Signal Processing*, 55(12):5695 – 5702, Dec 2007.

36. Y.C. Eldar. Generalized SURE for exponential families: Applications to regularization. *IEEE Transactions on Signal Processing*, 2008. Forthcoming Articles.

37. M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, Dec 2007.

38. H.-Y. Gao. Wavelet shrinkage denoising using the non-negative Garrote. *Journal of Computational and Graphical Statistics*, 7(4):469–488, 1998.

39. N. Gillis and F. Glineur. Nonnegative matrix factorization and underapproximation. In *9th International Symposium on Iterative Methods in Scientific Computing*, Lille, France, 2008. http://www.core.ucl.ac.be/ ngillis/.

40. N. Gillis and F. Glineur. Nonnegative factorization and maximum edge biclique problem. In *submitted*, 2009. http://www.uclouvain.be/en-44508.html.

41. R. Giryes, M. Elad, and Y.C. Eldar. Automatic parameter setting for iterative shrinkage methods. *Electrical and Electronics Engineers in Israel, 2008. IEEEI 2008. IEEE 25th Convention of*, pages 820–824, 2008.

42. S.A. Goreinov, I.V. Oseledets, D.V. Savostyanov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. How to find a good submatrix. (submitted), 2009.

43. T.M. Hancewicz and J.-H. Wang. Discriminant image resolution: a novel multivariate image analysis method utilizing a spatial classification constraint in addition to bilinear nonnegativity. *Chemometrics and Intelligent Laboratory Systems*, 77:18–31, 2005.

44. N.-D. Ho. *Nonnegative Matrix Factorization - Algorithms and Applications*. These/dissertation, Universite Catholique de Louvain, Belgium, FSA/INMA - Departement d'ingenierie mathematique, 2008.

45. N.-D. Ho, P. Van Dooren, and V.D. Blondel. Descent methods for nonnegative matrix factorization. *Numerical Linear Algebra in Signals, Systems and Control*, 2008.

46. B. Jorgensen. *The Theory of Dispersion Models*. Chapman and Hall, London, 1997.

47. A. N. Langville, C. D. Meyer, and R. Albright. Initializations for the nonnegative matrix factorization. In *Proc. of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, August 20–23 2006.

48. D.D. Lee and H.S. Seung. Learning of the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

49. D.D. Lee and H.S. Seung. *Algorithms for Nonnegative Matrix Factorization*, volume 13. MIT Press, 2001.

50. M.W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. National Academy of Science*, 106:697–702, 2009.

51. M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 327–336, Philadelphia, USA, August 20–23 2006.

52. C. Navasca, L. De Lathauwer, and S. Kinderman. Swamp reducing technique for tensor decomposition. In *Proc. of the European Signal Processing*, EUSIPCO, Lausanne, Switzerland, 2008.

53. M. Nikolova. Minimizers of cost-functions involving nonsmooth data-fidelity terms. application to the processing of outliers. *SIAM Journal on Numerical Analysis*, 40(3):965–994, 2002.

54. D. Nion and L. De Lathauwer. An enhanced line search scheme for complex-valued tensor decompositions. Application in DS-CDMA. *Signal Processing*, 88(3):749–755, 2008.

55. A.H. Phan and A. Cichocki. Multi-way nonnegative tensor factorization using fast hierarchical alternating least squares algorithm (HALS). In *Proc. of The 2008 International Symposium on Nonlinear Theory and its Applications*, Budapest, Hungary, 2008.

56. A.H. Phan, A. Cichocki, and K.S. Nguyen. Simple and efficient algorithm for distributed compressed sensing. In *Machine Learning for Signal Processing*, Cancun, 2008.

57. M. Rajih and P. Comon. Enhanced line search: A novel method to accelerate PARAFAC. Technical report, Laboratoire I3S Sophia Antipolis, France, 2005.

58. M. Rajih and P. Comon. Enhanced line search: A novel method to accelerate PARAFAC. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1128–1147, 2008.

59. J. Shihao, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6), 2008.

60. G. K. Smyth. Fitting Tweedies compound Poisson model to insurance claims data: dispersion modelling. *ASTIN Bulletin*, 32:2002.

61. M.W. Spratling and M.H. Johnson. A feedback model of perceptual learning and categorisation. *Visual Cognition*, 13:129–165, 2006.

62. R. Tauler and A. de Juan. *Multivariate Curve Resolution–Practical Guide to Chemometrics, Chapter 11*. Taylor and Francis Group, 2006.

63. Y. Tsaig and D.L. Donoho. Extensions of compressed sensing. *Signal Processing*, 86(3):549–571, 2006.

64. J.H. Wang, P.K. Hopke, T.M. Hancewicz, and S.-L. Zhang. Application of modified alternating least squares regression to spectroscopic image analysis. *Analytica Chimica Acta*, 476:93–109, 2003.

65. R. Zdunek and A. Cichocki. Nonnegative matrix factorization with constrained second-order optimization. *Signal Processing*, 87:1904–1916, 2007.

5

Projected Gradient Algorithms

In contrast to the multiplicative NMF algorithms discussed in Chapter 3, this class of Projected Gradient (PG) algorithms has additive updates. The algorithms discussed here provide approximate solutions to Non-negative Least Squares (NLS) problems, and are based on the alternating minimization technique:

$$\min_{\boldsymbol{x}_t \geq \mathbf{0}} D_F(\boldsymbol{y}_t \| \mathbf{A}\boldsymbol{x}_t) = \frac{1}{2}\|\boldsymbol{y}_t - \mathbf{A}\boldsymbol{x}_t\|_2^2, \qquad (t = 1, 2, \ldots, T), \tag{5.1}$$

$$\min_{\underline{\boldsymbol{a}}_i \geq \mathbf{0}} D_F(\underline{\boldsymbol{y}}_i \| \mathbf{X}^T \underline{\boldsymbol{a}}_i) = \frac{1}{2}\|\underline{\boldsymbol{y}}_i - \mathbf{X}^T \underline{\boldsymbol{a}}_i\|_2^2, \qquad (i = 1, 2, \ldots, I). \tag{5.2}$$

This can also be written in equivalent matrix forms

$$\min_{x_{jt} \geq 0} D_F(\mathbf{Y} \| \mathbf{A}\mathbf{X}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2, \tag{5.3}$$

$$\min_{a_{ij} \geq 0} D_F(\mathbf{Y}^T \| \mathbf{X}^T \mathbf{A}^T) = \frac{1}{2}\|\mathbf{Y}^T - \mathbf{X}^T \mathbf{A}^T\|_F^2, \tag{5.4}$$

where $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_J] \in \mathbb{R}_+^{I \times J}$, $\mathbf{A}^T = [\underline{\boldsymbol{a}}_1, \ldots, \underline{\boldsymbol{a}}_I] \in \mathbb{R}_+^{J \times I}$, $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T] \in \mathbb{R}_+^{J \times T}$, $\mathbf{X}^T = [\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_J] \in \mathbb{R}_+^{T \times J}$, $\mathbf{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T] \in \mathbb{R}^{I \times T}$, $\mathbf{Y}^T = [\underline{\boldsymbol{y}}_1, \ldots, \underline{\boldsymbol{y}}_I] \in \mathbb{R}^{T \times I}$, and usually $I \geq J$. The matrix $\mathbf{A}$ is assumed to be full-rank, thus providing the existence of a unique solution $\mathbf{X}^* \in \mathbb{R}^{J \times T}$. Since the NNLS problem (5.1) is strictly convex with respect to one set of variables $\{\mathbf{X}\}$, a unique solution exists for any matrix $\mathbf{X}$, and the solution $\boldsymbol{x}_t^*$ satisfies the Karush-Kuhn-Tucker (KKT) conditions:

$$\boldsymbol{x}_t^* \geq \mathbf{0}, \quad \boldsymbol{g}_{\mathbf{X}}(\boldsymbol{x}_t^*) \geq \mathbf{0}, \quad \boldsymbol{g}_{\mathbf{X}}(\boldsymbol{x}_t^*)^T \boldsymbol{x}_t^* = 0 \tag{5.5}$$

or in an equivalent compact matrix form as:

$$\mathbf{X}^* \geq \mathbf{0}, \quad \mathbf{G}_{\mathbf{X}}(\mathbf{X}^*) \geq \mathbf{0}, \quad \mathrm{tr}\{\mathbf{G}_{\mathbf{X}}(\mathbf{X}^*)^T \mathbf{X}^*\} = 0, \tag{5.6}$$

where the symbols $\boldsymbol{g}_{\mathbf{X}}$ and $\mathbf{G}_{\mathbf{X}}$ denote the corresponding gradient vector and gradient matrix:

$$\boldsymbol{g}_{\mathbf{X}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} D_F(\boldsymbol{y}_t \| \mathbf{A}\boldsymbol{x}_t) = \mathbf{A}^T(\mathbf{A}\boldsymbol{x}_t - \boldsymbol{y}_t), \tag{5.7}$$

$$\mathbf{G}_{\mathbf{X}}(\mathbf{X}) = \nabla_{\mathbf{X}} D_F(\mathbf{Y} \| \mathbf{A}\mathbf{X}) = \mathbf{A}^T(\mathbf{A}\mathbf{X} - \mathbf{Y}). \tag{5.8}$$

Similarly, the KKT conditions for the solution $\underline{\boldsymbol{a}}^*$ to (5.2), and the solution $\mathbf{A}^*$ to (5.4) are as follows:

$$\underline{\boldsymbol{a}}_i^* \geq \mathbf{0}, \quad \boldsymbol{g}_{\mathbf{A}}(\underline{\boldsymbol{a}}_i^*) \geq \mathbf{0}, \quad \boldsymbol{g}_{\mathbf{A}}(\underline{\boldsymbol{a}}_i^*)^T \underline{\boldsymbol{a}}_i^* = 0, \tag{5.9}$$

and the corresponding conditions in (5.6) are

$$\mathbf{A}^* \geq \mathbf{0}, \quad \mathbf{G}_{\mathbf{A}}(\mathbf{A}^*) \geq \mathbf{0}, \quad \mathrm{tr}\{\mathbf{A}^* \mathbf{G}_{\mathbf{A}}(\mathbf{A}^*)^T\} = 0, \tag{5.10}$$

where $\boldsymbol{g}_{\mathbf{A}}$ and $\mathbf{G}_{\mathbf{A}}$ are the gradient vector and gradient matrix of the objective function:

$$\boldsymbol{g}_{\mathbf{A}}(\underline{\boldsymbol{a}}_i) = \nabla_{\underline{\boldsymbol{a}}_i} D_F(\underline{\boldsymbol{y}}_i \| \mathbf{X}^T \underline{\boldsymbol{a}}_i) = \mathbf{X}(\mathbf{X}^T \underline{\boldsymbol{a}}_i - \underline{\boldsymbol{y}}_i), \tag{5.11}$$

$$\mathbf{G}_{\mathbf{A}}(\mathbf{A}) = \nabla_{\mathbf{A}} D_F(\mathbf{Y}^T \| \mathbf{X}^T \mathbf{A}^T) = (\mathbf{A}\mathbf{X} - \mathbf{Y})\mathbf{X}^T. \tag{5.12}$$

There are many approaches to solve the minimization problems (5.1) and (5.2), or equivalently (5.3) and (5.4). In this chapter, we shall discuss several projected gradient methods which take a general form of iterative updates:

$$\mathbf{X}^{(k+1)} = \left[\mathbf{X}^{(k)} - \boldsymbol{\eta}_{\mathbf{X}}^{(k)} \mathbf{P}_{\mathbf{X}}^{(k)}\right]_+, \tag{5.13}$$

$$\mathbf{A}^{(k+1)} = \left[\mathbf{A}^{(k)} - \mathbf{P}_{\mathbf{A}}^{(k)} \boldsymbol{\eta}_{\mathbf{A}}^{(k)}\right]_+, \tag{5.14}$$

where $[\mathbf{X}]_+ = \mathcal{P}_\Omega[\mathbf{X}]$ denotes a projection of entries of $\mathbf{X}$ onto a convex "feasible" set $\Omega = \left\{x_{jt} \in \mathbb{R} : x_{jt} \geq 0\right\}$ – namely, the nonnegative orthant $\mathbb{R}_+$ (the subspace of nonnegative real numbers), $\mathbf{P}_{\mathbf{X}}^{(k)}$ and $\mathbf{P}_{\mathbf{A}}^{(k)}$ are descent directions for $\mathbf{X}$ and $\mathbf{A}$ in the k-th inner iterative step, and $\boldsymbol{\eta}_{\mathbf{X}}^{(k)}$ and $\boldsymbol{\eta}_{\mathbf{A}}^{(k)}$ are the learning rate scalars or the diagonal matrices of positive learning rates.

The projection $[\mathbf{X}]_+$ can be performed in many ways.[1] One straightforward way is to replace all the negative entries in $\mathbf{X}$ by zero, or for practical purposes, by a small positive number ε in order to avoid numerical instabilities, thus giving (component-wise)

$$[\mathbf{X}]_+ = \max\{\varepsilon, \mathbf{X}\}. \tag{5.15}$$

Alternatively, it may be more efficient to preserve the nonnegativity of the solutions by an optimal choice of the learning rates $\boldsymbol{\eta}_{\mathbf{X}}^{(k)}$ and $\boldsymbol{\eta}_{\mathbf{A}}^{(k)}$, or by solving least-squares problems subject to the

[1] Although in this chapter we use only the simple nonlinear projection "half-wave rectifying", which replaces negative values by small positive constant ε, the PG algorithms discussed here can be easily adopted to the factors which are upper and/or lower bounded at any specific level, for example, $l_j \leq x_{jt} \leq u_j, \ \forall j$. This can be achieved by applying a suitable projection function $\mathcal{P}_\Omega[x_{jt}]$ which transforms the updated factors to the feasible region.

constraints (5.6) and (5.10). We here present exemplary PG methods which are proven to be very efficient for NMF problems, and are all part of our MATLAB toolbox: NMFLAB/NTFLAB for Signal and Image Processing [9, 30, 8].

5.1 OBLIQUE PROJECTED LANDWEBER (OPL) METHOD

The Landweber method [3] performs gradient descent minimization based on the following iterative scheme:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \eta_{\mathbf{X}}\, \mathbf{G}_{\mathbf{X}}^{(k)}, \tag{5.16}$$

where the descent direction $\mathbf{P}_{\mathbf{X}}^{(k)}$ is replaced with the gradient $\mathbf{G}_{\mathbf{X}}$ given in (5.8), and the range of the learning rate $\eta_{\mathbf{X}} \in (0, \eta_{max})$. This update ensures the asymptotic convergence to the minimum-norm least squares solution, with the convergence radius defined by

$$\eta_{max} = \frac{2}{\lambda_{max}(\mathbf{A}^T\mathbf{A})}, \tag{5.17}$$

where $\lambda_{max}(\mathbf{A}^T\mathbf{A})$ is the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$. Since $\mathbf{A}$ is nonnegative, for its eigenvalues we have $\lambda_{max}(\mathbf{A}^T\mathbf{A}) \leq \max(\mathbf{A}^T\mathbf{A}\mathbf{1}_J)$, where $\mathbf{1}_J = [1, \ldots, 1]^T \in \mathbb{R}^J$, and the modified Landweber method can be expressed as:

$$\mathbf{X}^{(k+1)} = \left[\mathbf{X}^{(k)} - \boldsymbol{\eta}_{\mathbf{X}}\, \mathbf{G}_{\mathbf{X}}^{(k)}\right]_+, \quad \text{where} \quad \boldsymbol{\eta}_{\mathbf{X}} = \text{diag}\{\eta_1, \eta_2, \ldots, \eta_J\}, \quad \eta_j < \frac{2}{(\mathbf{A}^T\mathbf{A}\mathbf{1}_J)_j}. \tag{5.18}$$

One variant of the method is the Oblique Projected Landweber (OPL) method [18], which can be regarded as a particular case of the PG iterative formula (5.13)–(5.14), where at each iterative step the solution obtained by (5.16) is projected onto the feasible set. Based on these, the method can be implemented for the standard NMF problem as shown in Algorithm 5.1.

The MATLAB implementation of the OPL algorithm is given in Listing 5.1.

5.2 LIN'S PROJECTED GRADIENT (LPG) ALGORITHM WITH ARMIJO RULE

A typical representative of PG algorithms in applications to NMF is Chih-Jen Lin's algorithm [21], which is given by the iterative formula (5.13)–(5.14) with $\mathbf{P}_{\mathbf{X}}^{(k)}$ and $\mathbf{P}_{\mathbf{A}}^{(k)}$ expressed by the gradients (5.8) and (5.12), respectively, and the projection rule (5.15). In contrast to the OPL algorithm given in Section 5.1 the learning rates $\eta_{\mathbf{X}}^{(k)}$ and $\eta_{\mathbf{A}}^{(k)}$ in Lin's PG algorithm in the inner iterations are not fixed diagonal matrices, but are scalars computed by inexact estimation techniques. Lin considered two options for estimating the learning rules: the Armijo rule along the projective arc of the algorithm proposed by Bertsekas [5, 4], and the modified Armijo rule.

In the first case, for every inner iterative step of the algorithm, the value of the learning rate $\eta_{\mathbf{X}}^{(k)}$ is given by

$$\eta_{\mathbf{X}}^{(k)} = \beta^{m_k}, \tag{5.19}$$

Algorithm 5.1: OPL-NMF

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data, J: rank of approximation
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ such that the cost functions (5.3) and (5.4) are minimized.

begin
  initialization for $\mathbf{A}$, $\mathbf{X}$
  repeat
    $\mathbf{X} \leftarrow \mathrm{OPL}(\mathbf{Y}, \mathbf{A}, \mathbf{X})$ /* Update $\mathbf{X}$ */
    $\mathbf{A} \leftarrow \mathrm{OPL}(\mathbf{Y}^T, \mathbf{X}^T, \mathbf{A}^T)^T$ /* Update $\mathbf{A}$ */
  until *a stopping criterion is met* /* convergence condition */
end

function $\mathbf{X} = \mathrm{OPL}(\mathbf{Y}, \mathbf{A}, \mathbf{X})$
begin
  $\mathbf{R} = \mathbf{A}^T\mathbf{A}$,
  $\mathbf{Z} = \mathbf{A}^T\mathbf{Y}$,
  $\boldsymbol{\eta}_{\mathbf{X}} = \mathrm{diag}\,(\mathbf{1}_J \oslash (\mathbf{R}\mathbf{1}_J))$
  repeat
    $\mathbf{G}_{\mathbf{X}} = \mathbf{R}\mathbf{X} - \mathbf{Z}$ /* Gradient with respect to $\mathbf{X}$ */
    $\mathbf{X} \leftarrow [\mathbf{X} - \boldsymbol{\eta}_X\, \mathbf{G}_{\mathbf{X}}]_+$ /* Update $\mathbf{X}$ */
  until *a stopping criterion is met* /* convergence condition */
end

Listing 5.1 OPL-NMF algorithm.

```
function [X] = nmf_opl(A,Y,X,no_iter)
%
% INPUTS:
% A – fixed matrix of dimension [I by J]
% Y – data matrix of dimension [I by T]
% X – initial solution matrix of dimension [J by T]
% no_iter – maximum number of iterations
%
% OUTPUTS:
% X – estimated matrix of dimension [J by T]
%
% ========================================================================
R = A'*A; Z = A'*Y; eta = 1./sum(R,2);

for k=1:no_iter
    G = R*X – Z;
    X = max(eps, X – bsxfun(@times,G,eta));
end % for k
```

where m_k is the first nonnegative integer m for which

$$D_F(\mathbf{Y}\|\mathbf{A}\mathbf{X}^{(k+1)}) - D_F(\mathbf{Y}\|\mathbf{A}\mathbf{X}^{(k)}) \leq \sigma\, \mathrm{tr}\left\{\nabla_{\mathbf{X}} D_F(\mathbf{Y}\|\mathbf{A}\mathbf{X}^{(k)})^T(\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)})\right\}, \qquad (5.20)$$

6

Quasi-Newton Algorithms for Nonnegative Matrix Factorization

So far we have discussed the NMF algorithms which perform optimization by searching for stationary points of a cost function based on first-order approximations, that is, using the gradient. In consequence, the additive learning algorithms have the following general form:

$$\mathbf{A} \leftarrow \mathcal{P}_{\Omega}\left[\mathbf{A} - \boldsymbol{\eta}_{\mathbf{A}}\, \nabla_{\mathbf{A}} D(\mathbf{Y}||\mathbf{AX})\right], \tag{6.1}$$

$$\mathbf{X} \leftarrow \mathcal{P}_{\Omega}\left[\mathbf{X} - \boldsymbol{\eta}_{\mathbf{X}} \nabla_{\mathbf{X}} D(\mathbf{Y}||\mathbf{AX})\right], \tag{6.2}$$

where $\mathcal{P}_{\Omega}[\xi]$ denotes the projection of ξ onto the set Ω of feasible solutions,[1] and the learning rates $\boldsymbol{\eta}_{\mathbf{A}}$ and $\boldsymbol{\eta}_{\mathbf{X}}$ are either fixed or iteratively updated scalars or diagonal matrices.

Let us consider a Taylor series expansion

$$\begin{aligned} D(\mathbf{Y}||(\mathbf{A}+\Delta\mathbf{A})\mathbf{X}) &= D(\mathbf{Y}||\mathbf{AX}) + \operatorname{vec}\left(\nabla_{\mathbf{A}} D(\mathbf{Y}||\mathbf{AX})\right)^T \operatorname{vec}(\Delta\mathbf{A}) \\ &+ \frac{1}{2}\operatorname{vec}(\Delta\mathbf{A})^T\, \mathbf{H}_{\mathbf{A}} \operatorname{vec}(\Delta\mathbf{A}) + O\left((\Delta\mathbf{A})^3\right), \end{aligned} \tag{6.3}$$

$$\begin{aligned} D(\mathbf{Y}||\mathbf{A}(\mathbf{X}+\Delta\mathbf{X})) &= D(\mathbf{Y}||\mathbf{AX}) + \operatorname{vec}\left(\nabla_{\mathbf{X}} D(\mathbf{Y}||\mathbf{AX})\right)^T \operatorname{vec}(\Delta\mathbf{X}) \\ &+ \frac{1}{2}\operatorname{vec}(\Delta\mathbf{X})^T\, \mathbf{H}_{\mathbf{X}} \operatorname{vec}(\Delta\mathbf{X}) + O\left((\Delta\mathbf{X})^3\right), \end{aligned} \tag{6.4}$$

where $\mathbf{H}_{\mathbf{A}} = \nabla^2_{\mathbf{A}} D(\mathbf{Y}||\mathbf{AX}) \in \mathbb{R}^{IJ\times IJ}$ and $\mathbf{H}_{\mathbf{X}} = \nabla^2_{\mathbf{X}} D(\mathbf{Y}||\mathbf{AX}) \in \mathbb{R}^{JT\times JT}$ are Hessians with respect to $\mathbf{A}$ and $\mathbf{X}$.

[1] Typically, the set Ω in NMF is the nonnegative orthant of the space of real numbers, that is, $\mathcal{P}_{\Omega}[\xi] = [\xi]_+$, however, other sets can also be used. For example, the updated factors can be bounded by a box rule, that is, $\Omega = \{\xi : l_{min} \le \xi \le u_{max}\}$.

In this chapter, we introduce learning algorithms for the NMF problem using second-order approximations, i.e. the third-order term in the above Taylor series expansion. In consequence, the learning rates $\boldsymbol{\eta}_{\mathbf{A}}$ and $\boldsymbol{\eta}_{\mathbf{X}}$ in (6.1)–(6.2) become the inverses of the Hessian, thus yielding the following projected Newton updating rules:

$$\text{vec}\,(\mathbf{A}) \leftarrow \mathcal{P}_{\Omega}\left[\text{vec}\,(\mathbf{A}) - \mathbf{H}_{\mathbf{A}}^{-1}\,\text{vec}\,(\mathbf{G}_{\mathbf{A}})\right], \tag{6.5}$$

$$\text{vec}\,(\mathbf{X}) \leftarrow \mathcal{P}_{\Omega}\left[\text{vec}\,(\mathbf{X}) - \mathbf{H}_{\mathbf{X}}^{-1}\,\text{vec}\,(\mathbf{G}_{\mathbf{X}})\right], \tag{6.6}$$

where $\mathbf{G}_{\mathbf{A}} = \nabla_{\mathbf{A}} D(\mathbf{Y}\|\mathbf{AX})$ and $\mathbf{G}_{\mathbf{X}} = \nabla_{\mathbf{X}} D(\mathbf{Y}\|\mathbf{AX})$. The symbol vec$(\mathbf{G})$ denotes the vectorized version of the matrix $\mathbf{G} \in \mathbb{R}^{J\times T}$, that is, vec$(\mathbf{G}) = [g_{11}, g_{21}, \ldots, g_{J1}, g_{12}, \ldots, g_{JT}]^T \in \mathbb{R}^{JT}$.

Using the information about the curvature of the cost function, which is intimately related to second-derivatives, the convergence can be considerably accelerated. This, however, also introduces many related practical problems that must be addressed prior to applying learning algorithms. For example, the Hessian $\mathbf{H}_{\mathbf{A}}$ and $\mathbf{H}_{\mathbf{X}}$ must be positive-definite to ensure the convergence of approximations of (6.5)–(6.6) to a local minimum of $D(\mathbf{Y}\|\mathbf{AX})$. Unfortunately, this is not guaranteed using the NMF alternating minimization rule, and we need to resort to some suitable Hessian approximation techniques. In addition, the Hessian values may be very large (especially when updating $\mathbf{X}$) and of severely ill-conditioned nature (in particular for large-scale problems), which gives rise to many difficult problems related to its inversion.

This chapter provides a comprehensive study on the solutions to the above-mentioned problems. We also give some heuristics on the selection of a cost function and related regularization terms which restrict the area of feasible solutions, and help to converge to the global minimum of the cost function.

The layout of the chapter is as follows: first, we discuss the simplest approach to the projected quasi-Newton optimization using the Levenberg-Marquardt regularization of the Hessian. For generality, as a cost function, we consider the Alpha- and Beta-divergences [11, 29, 22] that unify many well-known cost functions (see the details in Chapter 2), we then discuss the reduced quasi-Newton optimization that involves the Gradient Projection Conjugate Gradient (GPCG) algorithm [24, 2, 1, 34], followed by the FNMA method proposed by Kim, Sra and Dhillon [21]. Further, as a special case of the quasi-Newton method, we present one quadratic programming method [35]. The simulations that conclude the chapter are performed on the same benchmark data (mixed signals) as those in other chapters.

6.1 PROJECTED QUASI-NEWTON OPTIMIZATION

In Chapter 2, we have demonstrated that the Bregman, Alpha- or Beta-divergences [15, 11] are particularly useful for dealing with non-Gaussian noisy disturbances. In this section, we discuss the projected quasi-Newton method in the context of application to alternating minimization of these functions. First, the basic computations of the gradient and Hessian matrices are presented, and then the efficient method for computing the inverse to the Hessian is discussed.

6.1.1 Projected Quasi-Newton for Frobenius Norm

First, we will present the projected quasi-Newton method that minimizes the standard Euclidean distance in NMF. This case deserves special attention since normally distributed white noise is

Algorithm 6.1: **QNE-NMF**

Input: $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$: input data, J: rank of approximation
Output: $\mathbf{A} \in \mathbb{R}_+^{I \times J}$ and $\mathbf{X} \in \mathbb{R}_+^{J \times T}$ such that the cost function (6.7) is minimized.

```
1  begin
2      initialization for A, X
3      repeat
4          X ← QNE(Y, A, X)                                  /* Update X */
5          A ← QNE(Y^T, X^T, A^T)^T                          /* Update A */
6      until a stopping criterion is met                     /* convergence condition */
7  end
8  function X = QNE(Y, A, X)
9  begin
10     R = A^† Y
11     repeat
12         X ← [(1 − η_0) X + η_0 R]_+                       /* Update X */
13     until a stopping criterion is met                     /* convergence condition */
14 end
```

a common assumption in practice. For the squared Euclidean distance:

$$D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}||\mathbf{Y} - \mathbf{AX}||_F^2, \tag{6.7}$$

the gradients and Hessians have the following forms:

$$\mathbf{G}_\mathbf{X} = \mathbf{A}^T(\mathbf{AX} - \mathbf{Y}) \in \mathbb{R}^{J \times T}, \qquad \mathbf{G}_\mathbf{A} = (\mathbf{AX} - \mathbf{Y})\mathbf{X}^T \in \mathbb{R}^{I \times J}, \tag{6.8}$$

$$\mathbf{H}_\mathbf{X} = \mathbf{I}_T \otimes \mathbf{A}^T\mathbf{A} \in \mathbb{R}^{JT \times JT}, \qquad \mathbf{H}_\mathbf{A} = \mathbf{X}\mathbf{X}^T \otimes \mathbf{I}_I \in \mathbb{R}^{IJ \times IJ}, \tag{6.9}$$

where $\mathbf{I}_T \in \mathbb{R}^{T \times T}$ and $\mathbf{I}_I \in \mathbb{R}^{I \times I}$ are identity matrices, and the symbol $\otimes$ stands for the Kronecker product.

The update rule (6.6) for $\mathbf{X}$ can be reformulated as follows

$$\begin{aligned} \text{vec}\,(\mathbf{X}) &\leftarrow \mathcal{P}_\Omega\left[\text{vec}\,(\mathbf{X}) - \eta_0\ (\mathbf{H}_\mathbf{X})^{-1}\ \text{vec}\,(\mathbf{G}_\mathbf{X})\right] \\ &= \mathcal{P}_\Omega\left[\text{vec}\,(\mathbf{X}) - \eta_0\ \left(\mathbf{I}_T \otimes \mathbf{A}^T\mathbf{A}\right)^{-1}\ \text{vec}\,(\mathbf{G}_\mathbf{X})\right] \\ &= \mathcal{P}_\Omega\left[\text{vec}\,(\mathbf{X}) - \eta_0\ \left(\mathbf{I}_T \otimes \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\right)\ \text{vec}\,(\mathbf{G}_\mathbf{X})\right], \end{aligned} \tag{6.10}$$

thus it is re-written in the matrix form as

$$\begin{aligned} \mathbf{X} &\leftarrow \mathcal{P}_\Omega\left[\mathbf{X} - \eta_0\ \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\ \mathbf{G}_\mathbf{X}\right] \\ &= \mathcal{P}_\Omega\left[\mathbf{X} - \eta_0\ \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T(\mathbf{AX} - \mathbf{Y})\right] \end{aligned} \tag{6.11}$$

7

Multi-Way Array (Tensor) Factorizations and Decompositions

The problems of nonnegative multi-way array (tensor) factorizations and decompositions arise in a variety of disciplines in the sciences and engineering. They have a wide range of important applications such as in bioinformatics, neuroscience, image understanding, text mining, chemometrics, computer vision and graphics, where tensor factorizations and decompositions can be used to perform factor retrieval, dimensionality reduction, compression, denoising, to mention but a few. For example, in neuroimage processing, images and videos are naturally represented by third-order, or general higher-order tensors. Color video sequences are normally represented by fourth-order tensors, thus requiring three indices for color images and a fourth index for the temporal information.

Almost all NMF algorithms described in the earlier chapters can be extended or generalized to the various nonnegative tensor factorizations and decompositions formulated in Chapter 1. In this chapter we mainly focus on the Nonnegative Tensor Factorization (NTF) (i.e., the PARAFAC with nonnegativity and sparsity constraints), the Nonnegative Tucker Decomposition (NTD) and the Block-Oriented Decomposition (BOD). In order to make this chapter as self-contained as possible, we re-introduce some concepts and derive many efficient heuristic algorithms for nonnegative tensor (multi-way array) factorizations and decompositions. Our particular emphasis is on a detailed treatment of generalized robust cost functions, such as Alpha- and Beta-divergences. Based on these cost functions, several classes of algorithms are introduced, including: (1) multiplicative updating; (2) Alternating Least Squares (ALS); and (3) Hierarchical ALS (HALS). These algorithms are then incorporated into multi-layer networks in order to improve the performance (see also Chapters 3–6), starting from relatively simple third-order nonnegative tensor factorizations through to extensions to arbitrarily high order tensor decompositions.

Practical considerations include the ways to impose nonnegativity or semi-nonnegativity, together with optional constraints such as orthogonality, sparsity and/or smoothness. To follow the material in this chapter it would be helpful to be familiar with Chapters 1, 3 and 4.

7.1 LEARNING RULES FOR THE EXTENDED THREE-WAY NTF1 PROBLEM

Based on the background given in Chapter 1, we shall now introduce practical learning rules for several extended tensor decompositions.

7.1.1 Basic Approaches for the Extended NTF1 Model

Consider the extended NTF1 model with irregular frontal slices, shown in Figure 7.1(a) [25], which can be exploited as follows: "Given a three-way (third-order) tensor formed by a set of matrices $\mathbf{Y}_q \in \mathbb{R}_+^{I \times T_q}$ $(q = 1, 2, \ldots, Q)$, formulate a set of nonnegative and sparse matrices $\mathbf{A} \in \mathbb{R}_+^{I \times J}$, $\mathbf{C} \in \mathbb{R}_+^{Q \times J}$ and $\mathbf{X}_q \in \mathbb{R}_+^{J \times T_q}$ for $q = 1, 2, \ldots, Q$ with reduced dimensions (typically, $J << I < T_q$)".
The extended NTF1 model for a three-way array can be represented in two different mathematical forms, as illustrated in Figures 7.1(b) and 7.1(c). Firstly, it can be described by a set of tri-NMF models:

$$\boxed{\mathbf{Y}_q = \mathbf{A}\mathbf{D}_q\mathbf{X}_q + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q),} \tag{7.1}$$

where $\mathbf{D}_q \in \mathbb{R}_+^{J \times J}$ are diagonal matrices (each diagonal matrix contains the q-th row of matrix $\mathbf{C} \in \mathbb{R}_+^{Q \times J}$ in its main diagonal), $\mathbf{X}_q = [x_{jtq}] \in \mathbb{R}_+^{J \times T_q}$ are matrices representing sources (or hidden components), and matrices $\mathbf{E}_q = [e_{itq}] \in \mathbb{R}^{I \times T_q}$ represent errors or noise depending upon the application. The diagonal matrices $\mathbf{D}_q$ can be considered as scaling matrices and can therefore be absorbed into the matrices $\mathbf{X}_q$ upon defining a new set of matrices as $\mathbf{X}_q \triangleq \mathbf{D}_q\mathbf{X}_q$ (if no additional constraints on the component matrix $\mathbf{C}$ are imposed), to give

$$\boxed{\mathbf{Y}_q = \mathbf{A}\mathbf{X}_q + \mathbf{E}_q, \qquad (q = 1, 2, \ldots, Q).} \tag{7.2}$$

Thus, only the mixing (bases) matrix $\mathbf{A}$ and the set of scaled source matrices $\mathbf{X}_q$ need to be found whereas due to the scaling ambiguity the matrix $\mathbf{C}$ does not need to be calculated explicitly. This also allows us to use row-wise unfolding to convert the NTF1 problem into the standard NMF problem described by the single matrix equation

$$\boxed{\mathbf{Y}_{(1)} = \mathbf{A}\mathbf{X}_{(1)} + \mathbf{E}_{(1)},} \tag{7.3}$$

where $\mathbf{Y}_{(1)} = \bar{\mathbf{Y}} = [\bar{y}_{i\bar{t}}] = [\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_Q] \in \mathbb{R}^{I \times \bar{T}}$; $\mathbf{X}_{(1)} = \bar{\mathbf{X}} = [\bar{x}_{j\bar{t}}] = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q] \in \mathbb{R}^{J \times \bar{T}}$; $\mathbf{E}_{(1)} = [\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_Q] \in \mathbb{R}^{I \times \bar{T}}$, and $\bar{T} = \sum_{q=1}^{Q} T_q$, $\bar{t} = 1, 2, \ldots, \bar{T}$.

Based on the above representations, we have several possible approaches to find (identify) the extended NTF1 model:

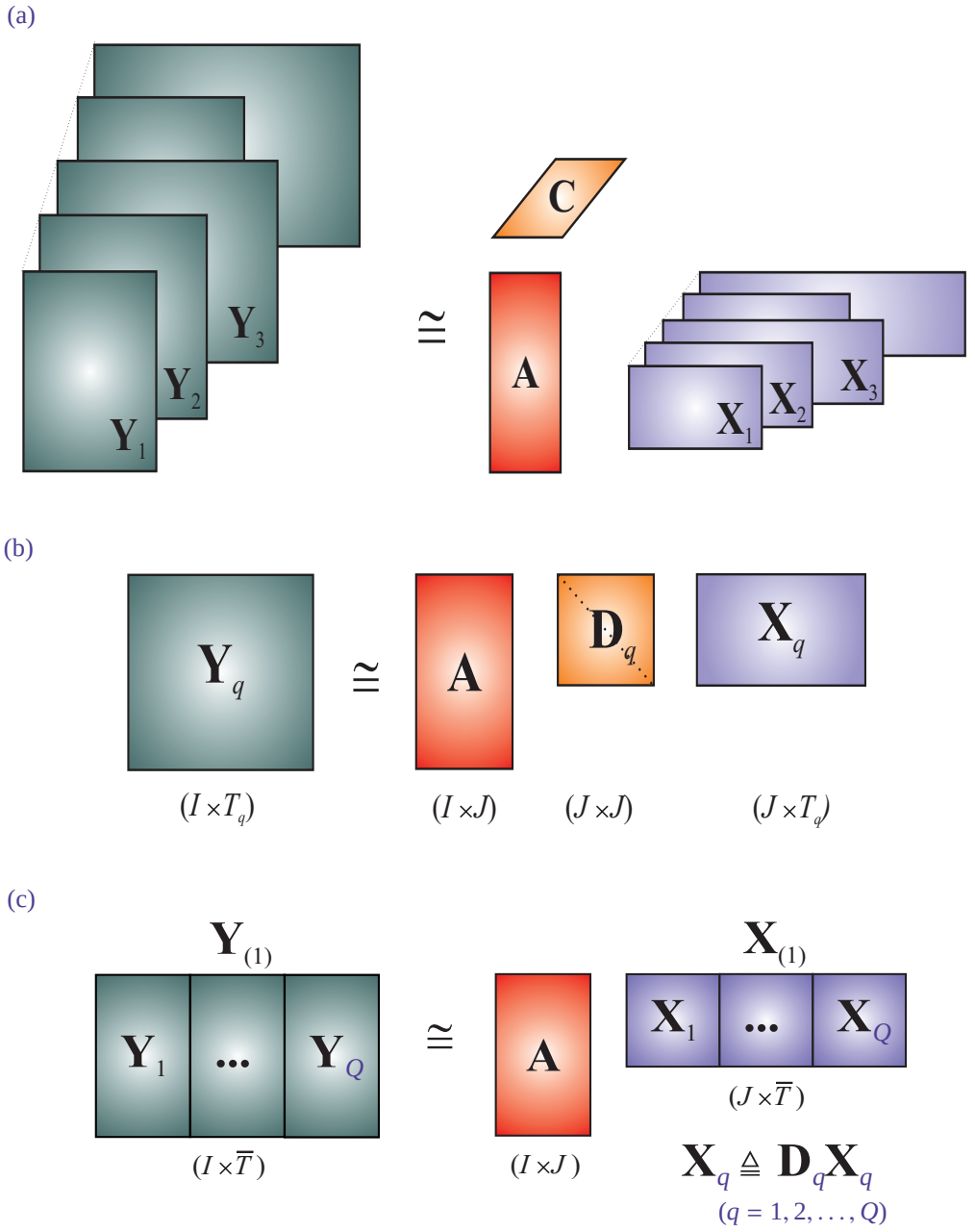


Fig. 7.1 (a) NTF1 model that approximately decomposes a three-way array with irregular frontal slices $\mathbf{Y}_q \in \mathbb{R}_+^{I \times T_q}$ into a set of nonnegative matrices $\mathbf{A} = [a_{ij}] \in \mathbb{R}_+^{I \times J}, \mathbf{C} = [c_{qj}] \in \mathbb{R}_+^{Q \times J}$ and $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_Q\}$, $\mathbf{X}_q = [x_{jtq}] \in \mathbb{R}_+^{J \times T_q}$; ($\mathbf{E}_q \in \mathbb{R}^{I \times T_q}$ represents errors). (b) An equivalent representation using set of three-factor NMF, where $\mathbf{D}_q = \text{diag}(\boldsymbol{c}_q)$ are diagonal matrices. (c) Global matrix representation using row-wise (mode-1) unfolding of the three-way array; in this case the sub-matrices are defined as $\mathbf{X}_q \triangleq \mathbf{D}_q \mathbf{X}_q$, $(q = 1, 2, \ldots, Q)$.

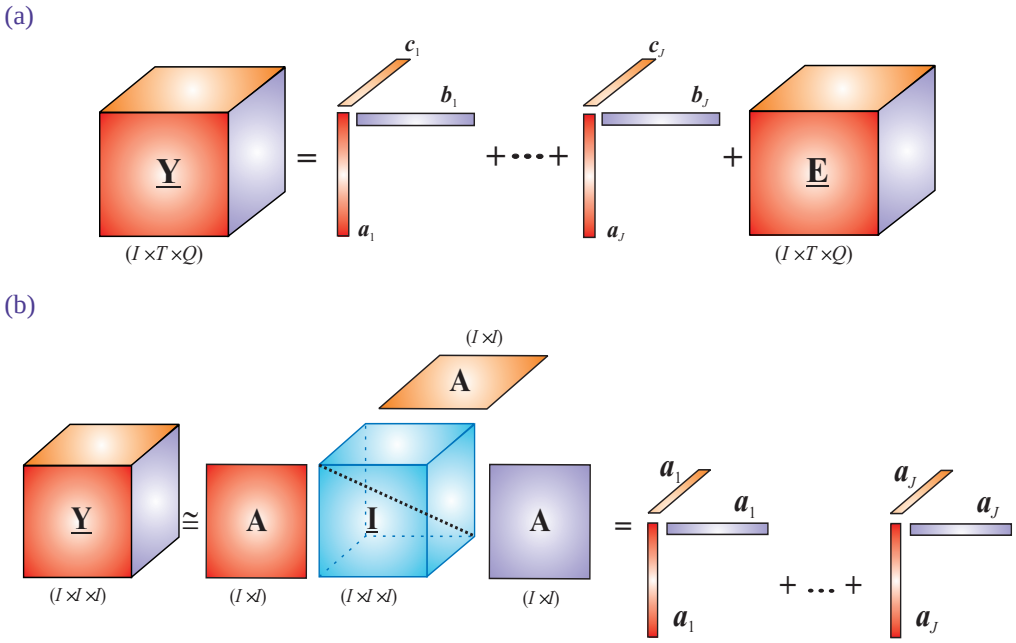


Fig. 7.2 (a) Illustration of the standard nonnegative tensor factorization (NTF) and (b) Super-Symmetric Tensor Nonnegative Factorization (SSNTF) for a third-order tensor by sum of rank-one tensors. The SSNTF is a special case of the NTF for $I = T = Q$ and $\mathbf{A} = \mathbf{B} = \mathbf{C}$ (or equivalently $\boldsymbol{a}_j = \boldsymbol{b}_j = \boldsymbol{c}_j \in \mathbb{R}^I,\ \forall j$).

negative component (factor) matrices: $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_J] \in \mathbb{R}^{I \times J}$, $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_J] \in \mathbb{R}^{T \times J}$ and $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_J] \in \mathbb{R}^{Q \times J}$ (see Figure 7.2 and also Chapter 1)

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \left(\boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j\right) + \underline{\mathbf{E}} = \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}}, \tag{7.30}$$

where $\underline{\mathbf{E}} = \underline{\mathbf{Y}} - \hat{\underline{\mathbf{Y}}} \in \mathbb{R}^{I \times T \times Q}$ is a tensor representing the error.

The goal is to estimate matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ subject to constraints. These include scaling to unit length vectors, nonnegativity, orthogonality, sparseness and/or smoothness of all or some of the columns $\boldsymbol{a}_j$.

A super-symmetric tensor has entries which are invariant under any permutation of the indices. For example, for a third-order super-symmetric tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ (with $I = T = Q$) we have $y_{itq} = y_{iqt} = y_{tiq} = y_{tqi} = y_{qit} = y_{qti}$, and its nonnegative factorization (referred to as the SSNTF) simplifies into the form

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \left(\boldsymbol{a}_j \circ \boldsymbol{a}_j \circ \boldsymbol{a}_j\right) + \underline{\mathbf{E}} = \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{A} + \underline{\mathbf{E}}, \tag{7.31}$$

Algorithm 7.4: Simple HALS NTF

Input: $\underline{\mathbf{Y}}$: input data of size $I_1 \times I_2 \times \cdots \times I_N$, J: number of basis components
Output: N component matrices $\mathbf{A}^{(n)} \in \mathbb{R}_+^{I_n \times J}$ such that the cost functions (7.94) are minimized.

begin
  ALS or random initialization for all factors $\mathbf{A}^{(n)}$
  $\boldsymbol{a}_j^{(n)} \leftarrow \boldsymbol{a}_j^{(n)} / \|\boldsymbol{a}_j^{(n)}\|_2$ for $\forall\, j, n = 1, 2, ..., N-1$ /* normalize to unit length */
  $\underline{\mathbf{E}} = \underline{\mathbf{Y}} - \widehat{\underline{\mathbf{Y}}} = \underline{\mathbf{Y}} - [\![\{\mathbf{A}\}]\!]$ /* residual tensor */
  repeat
    for $j = 1$ *to* J **do**
      $\underline{\mathbf{Y}}^{(j)} = \underline{\mathbf{E}} + [\![\boldsymbol{a}_j^{(1)}, \boldsymbol{a}_j^{(2)}, \ldots, \boldsymbol{a}_j^{(N)}]\!]$
      for $n = 1$ *to* N **do**
        $\boldsymbol{a}_j^{(n)} \leftarrow \left[\mathbf{Y}_{(n)}^{(j)} \left\{\boldsymbol{a}_j\right\}^{\odot_{-n}}\right]_+$ /* See Eqs. (7.99) and (7.100) */
        if $n \neq N$ **then** $\boldsymbol{a}_j^{(n)} \leftarrow \boldsymbol{a}_j^{(n)} / \|\boldsymbol{a}_j^{(n)}\|_2$ /* normalize to unit length */
      end
      $\underline{\mathbf{E}} = \underline{\mathbf{Y}}^{(j)} - [\![\boldsymbol{a}_j^{(1)}, \boldsymbol{a}_j^{(2)}, \ldots, \boldsymbol{a}_j^{(N)}]\!]$
    end
  until *a stopping criterion is met* /* convergence condition */
end

Algorithm 7.5: FAST HALS NTF

Input: $\underline{\mathbf{Y}}$: input data of size $I_1 \times I_2 \times \cdots \times I_N$, J: number of basis components
Output: N factors $\mathbf{A}^{(n)} \in \mathbb{R}_+^{I_n \times J}$ such that the cost functions (7.94) are minimized.

begin
  Nonnegative random or nonnegative ALS initialization for all factors $\mathbf{A}^{(n)}$ /* [a] */
  $\boldsymbol{a}_j^{(n)} \leftarrow \boldsymbol{a}_j^{(n)} / \|\boldsymbol{a}_j^{(n)}\|_2$ for $\forall\, j, n = 1, 2, ..., N-1$ /* normalize to unit length */
  $\mathbf{T}^{(1)} = (\mathbf{A}^{(1)\,T}\mathbf{A}^{(1)}) \circledast \cdots \circledast (\mathbf{A}^{(N)\,T}\mathbf{A}^{(N)})$
  repeat
    $\boldsymbol{\gamma} = \text{diag}(\mathbf{A}^{(N)\,T}\mathbf{A}^{(N)})$
    for $n = 1$ *to* N **do**
      if $n = N$ **then** $\boldsymbol{\gamma} = \mathbf{1}$
      $\mathbf{T}^{(2)} = \mathbf{Y}_{(n)} \{\mathbf{A}^{\odot_{-n}}\}$
      $\mathbf{T}^{(3)} = \mathbf{T}^{(1)} \oslash (\mathbf{A}^{(n)\,T}\mathbf{A}^{(n)})$
      for $j = 1$ *to* J **do**
        $\boldsymbol{a}_j^{(n)} \leftarrow \left[\gamma_j\, \boldsymbol{a}_j^{(n)} + \boldsymbol{t}_j^{(2)} - \mathbf{A}^{(n)}\, \boldsymbol{t}_j^{(3)}\right]_+$
        if $n \neq N$ **then** $\boldsymbol{a}_j^{(n)} = \boldsymbol{a}_j^{(n)} / \left\|\boldsymbol{a}_j^{(n)}\right\|_2$ /* normalize to unit length */
      end
      $\mathbf{T}^{(1)} = \mathbf{T}^{(3)} \circledast (\mathbf{A}^{(n)\,T}\mathbf{A}^{(n)})$
    end
  until *a stopping criterion is met* /* convergence condition */
end

[a] For a three-way tensor, direct trilinear decomposition can be used for initialization.

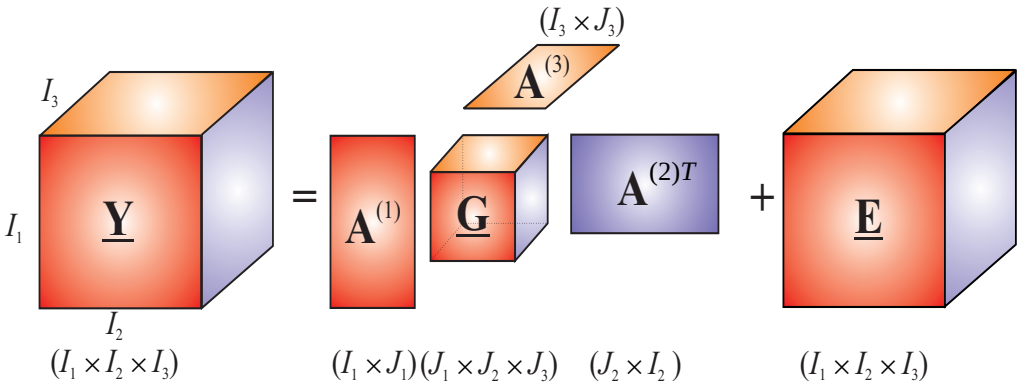


Fig. 7.4 Illustration and notations used for a higher-order Tucker decomposition; the objective here is to find optimal component (common factor) matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ and a core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n}$. We usually impose additional constraints on the component matrices and/or the core tensor such as nonnegativity and sparsity.

where the parameters α_{sp}, α_{sm}, and α_{cr} control respectively the degrees of sparsity, smoothness, and uncorrelatedness, $\gamma_j^{(n)}$ are scaling coefficients defined in (7.98), and $\mathbf{S}$ is a smoothing matrix. These parameters can be different for each factor $\mathbf{A}^{(n)}$.

7.4 ALGORITHMS FOR NONNEGATIVE AND SEMI-NONNEGATIVE TUCKER DECOMPOSITIONS

The higher-order tensor Tucker decomposition is described as a "decomposition of a given N-th order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ into an unknown core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \cdots \times J_N}$ multiplied by a set of N unknown component matrices, $\mathbf{A}^{(n)} = [\boldsymbol{a}_1^{(n)}, \boldsymbol{a}_2^{(n)}, \ldots, \boldsymbol{a}_{J_n}^{(n)}] \in \mathbb{R}^{I_n \times J_n}$ $(n = 1, 2, \ldots, N)$, representing common factors or loadings" [82], [33], [50], [62], [51], [76], [52]

$$\underline{\mathbf{Y}} = \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_N=1}^{J_N} g_{j_1 j_2 \cdots j_N} \, \boldsymbol{a}_{j_1}^{(1)} \circ \boldsymbol{a}_{j_2}^{(2)} \circ \cdots \circ \boldsymbol{a}_{j_N}^{(N)} + \underline{\mathbf{E}} \tag{7.115}$$

$$= \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} + \underline{\mathbf{E}} = \underline{\mathbf{G}} \times \{\mathbf{A}\} + \underline{\mathbf{E}} \tag{7.116}$$

$$= \hat{\underline{\mathbf{Y}}} + \underline{\mathbf{E}}, \tag{7.117}$$

where tensor $\widehat{\underline{\mathbf{Y}}}$ is an approximation of tensor $\underline{\mathbf{Y}}$, and tensor $\underline{\mathbf{E}} = \underline{\mathbf{Y}} - \hat{\underline{\mathbf{Y}}}$ denotes the residual or error tensor (see Figure 7.4). In the next sections, we consider at first a simple Tucker model with orthogonality constraints followed by Tucker models with nonnegativity and sparsity constraints, in which the orthogonality is not necessarily imposed.

2. Compute the residual error tensor $\underline{\mathbf{R}}_1 = \underline{\mathbf{Y}} - \widehat{\underline{\mathbf{Y}}}_1$, and divide it into two parts by threshold values set up by its most frequent values (defined by the mode function): $\underline{\mathbf{R}}_{1up} = \max(\underline{\mathbf{R}}_1, \text{mode}(\underline{\mathbf{R}}_1))$, $\underline{\mathbf{R}}_{1low} = \min(\underline{\mathbf{R}}_1, \text{mode}(\underline{\mathbf{R}}_1))$. Then, we normalize these two tensors $\underline{\mathbf{R}}_{1up}$ and $\underline{\mathbf{R}}_{1low}$ to unit scale [0, 1], and also invert $\underline{\mathbf{R}}_{1low} = \underline{\mathbf{1}} - \underline{\mathbf{R}}_{1low}$.

3. Decompose these two nonnegative residue tensors to get two new approximation tensors $\widehat{\underline{\mathbf{Y}}}_{1up}$ and $\widehat{\underline{\mathbf{Y}}}_{1low}$. Invert and scale these two tensors to the original ranges of their corresponding tensors $\underline{\mathbf{R}}_{1up}$ and $\underline{\mathbf{R}}_{1low}$.

4. Obtain the level-2 approximation tensor $\widehat{\underline{\mathbf{Y}}}_2$ and return to step 2 for the next level.

The residual tensor $\underline{\mathbf{R}}$ does not need to be split if we use the standard or semi-nonnegative Tucker decomposition. Multi-level decomposition allows much smaller errors and higher performance to be achieved. Figure 7.18 illustrates the approximated slices in the multi-level scheme illustrated in Figure 7.6 applied to face representation. The accuracy of approximation increases gradually with the number of decomposition levels (Figures 7.18(b)–7.18(h)). It should be noted that the approximated tensor obtained in the first layer is similar to the low-resolution data of its raw data. In the case of noisy data, to receive the high-resolution details we must make tradeoff between the level of detail and noise. Upon applying denoising to the residue tensors, NTD may become an efficient tool for multi-way restoration and compression. In reconstruction/denoising applications, we take into account an approximation tensor $\widehat{\underline{\mathbf{Y}}}$, but for feature extraction applications, factors $\mathbf{A}^{(n)}$ and core tensor $\underline{\mathbf{G}}$ are analyzed. Another advantage of this scheme is that we can avoid decomposition using a large core tensor, since in hierarchical decomposition the dimension of the core tensor can be much smaller.

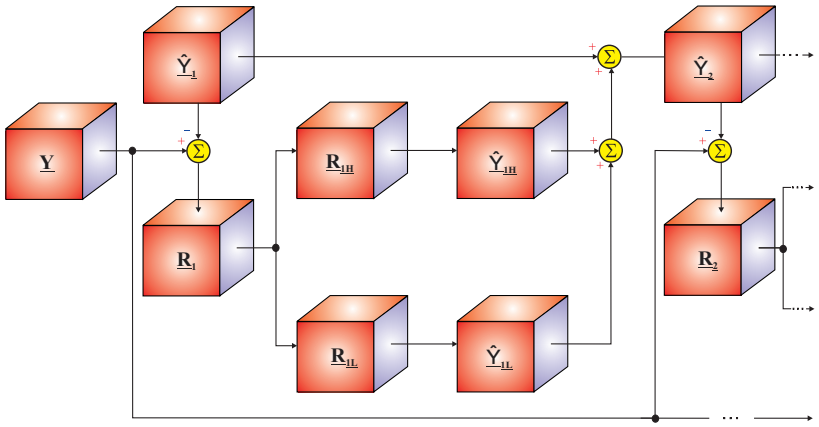
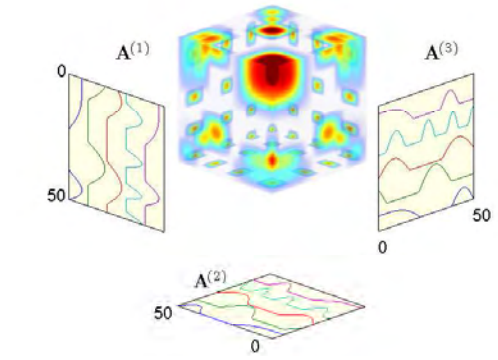


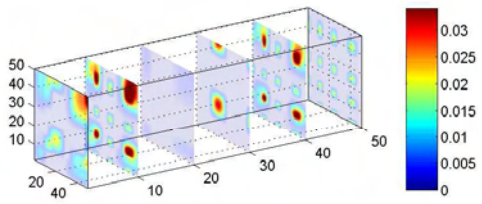
Fig. 7.6 Hierarchical multi-level nonnegative tensor decomposition.

7.7 SIMULATIONS, ILLUSTRATIVE EXAMPLES AND APPLICATIONS

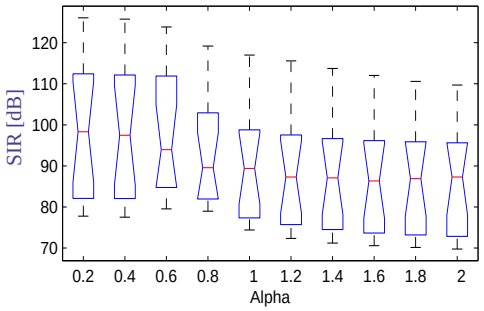
The performance of the algorithms introduced in this chapter are now illustrated with several case studies for various benchmarks and real-world data (see also [68], [66], [19]). For convenience, the case studies are explained through examples which reveal performance and conver-



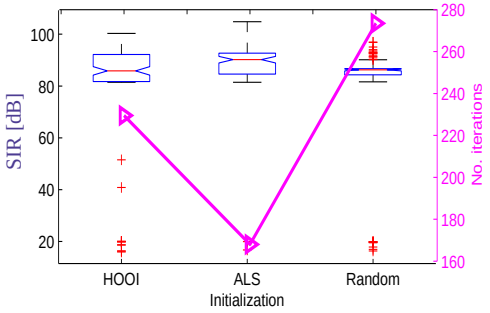
(a) 3D volume of original data tensor with corresponding input factors



(b) Frontal slices of the original data tensor



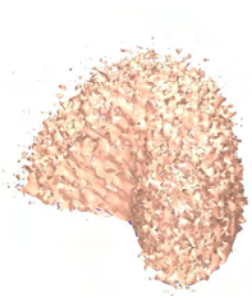
(c) Performance of the Alpha NTF Algorithm



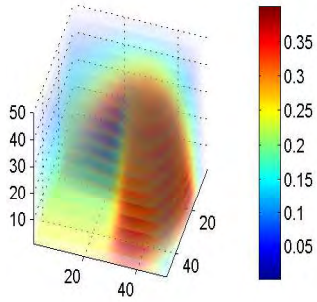
(d) Comparison of initialization techniques (SIR on the left vertical axis, No. iterations on the right vertical axis)

Fig. 7.8 Example 7.2: The data tensor was constructed using five basis waveforms (all three factors are identical) (a) 3D visualization of original tensor with the corresponding factors; (b) Frontal slices of the original tensor (c) SIR distributions obtained by Alpha NTF algorithm with $\alpha = [0.2-2]$; (reconstruction is almost perfect and estimated factors are almost identical to original factors, ignoring scaling and permutation ambiguities); (d) Performance comparison of Alpha NTF algorithm for three initialization techniques: HOOI, ALS and Random mode. (The performance index is Signal to Interference Ratio (SIR), where larger values indicate better performance of teh algorithm).

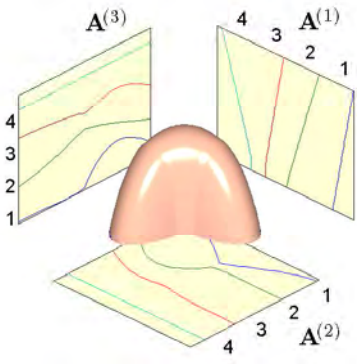
which was corrupted by additive Gaussian noise with SNR = 10 dB (see Figure 7.9(a)). For the data tensor without additive noise (Figure 7.9(b)) the NTF model was able to explain 98.59% of the variation with $J = 4$ components, and 99.20% of the variation with $J = 5$ components. Figure 7.9(c) presents the estimated factors and the corresponding reconstructed data tensor obtained by the Fast HALS NTF algorithm for the data tensor without additive noise. In the next experiment the noisy data tensor (see Figure 7.9(a)) was approximated by the NTF model based on the Alpha and Beta HALS NTF algorithms with smoothness constraints, which gave the FIT value of 96.1% using only $J = 4$ components. Figure 7.9(d) displays the reconstructed data tensor for the estimated components by applying the Beta HALS NTF ($\beta = 2$) algorithm. The



(a) Iso-surface of noisy 3D image

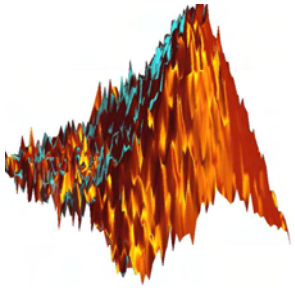


(b) Slice mode representation of the 3D image

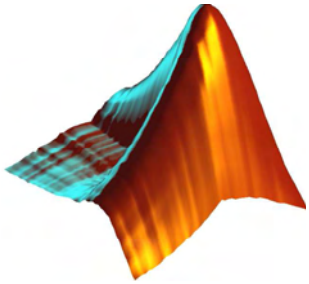


(c) Iso-surface of the reconstructed 3D image using its estimated factors

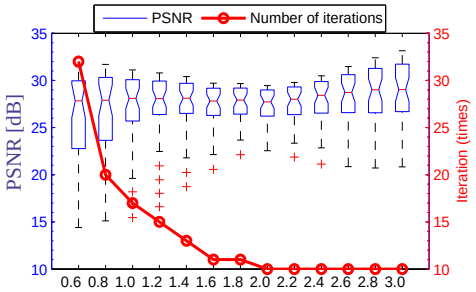
(d) Iso-surface of the reconstructed image using the Beta HALS NTF (PSNR = 27.19 dB)



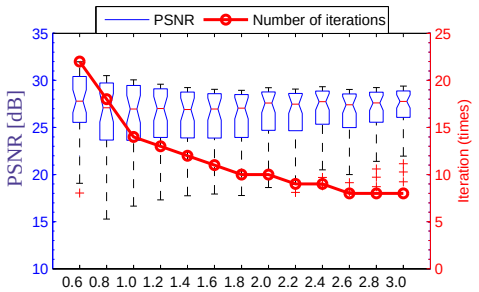
(e) The 40-th noisy slice



(f) The reconstructed 40-th slice using the Beta HALS NTF; PSNR = 27.19 dB



(g) Alpha HALS sNTF with smoothness constraints



(h) Beta HALS sNTF with smoothness constraints

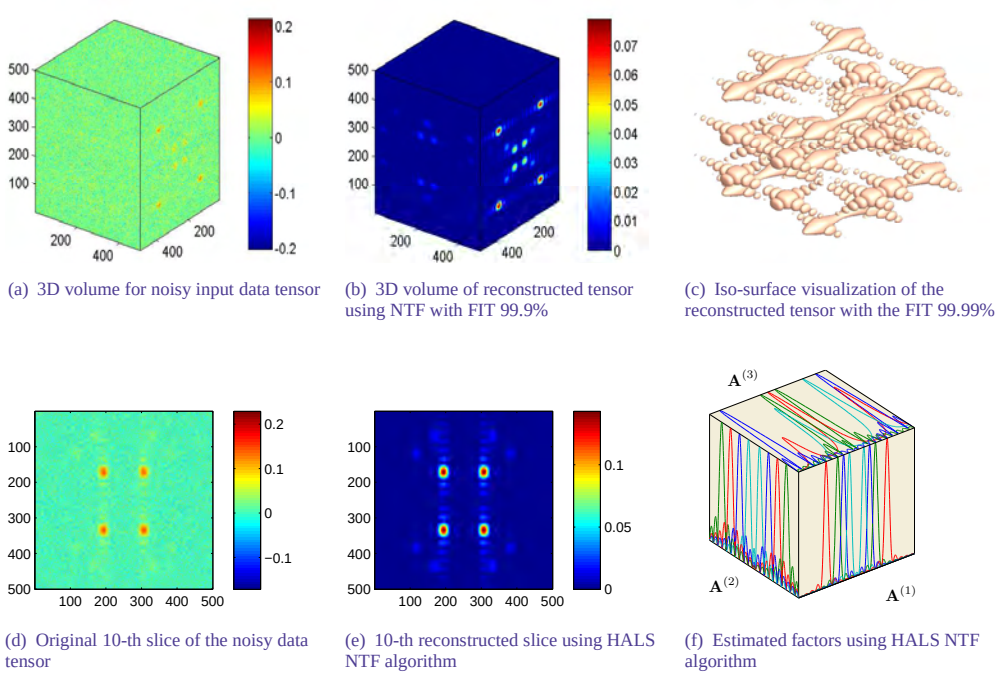
Fig. 7.9 Reconstruction of noisy 3D image for the membrane tensor $\underline{\mathbf{Y}} \in \mathbb{R}_{+}^{51\times51\times40}$ from Example 7.3: (a)-(b) iso-surface and slice mode visualizations of the noisy and "clean" data tensors, (c) estimated factors and the data tensor by the Fast HALS-NTF for the "clean" data tensor (without additive noise), (d) iso-surface visualization of the reconstructed tensor by using Beta-HALS-NTF algorithm; (e)-(f) surface visualizations of the 40-th slices of the noisy data tensor and reconstructed tensor by Beta ($\beta = 2$) HALS NTF algorithm, respectively; (g)-(h) Performance of multiplicative Alpha NTF and Beta NTF algorithms with smoothness constraints.

Listing 7.2 Example 7.2.

```
% Alpha NTF with alpha = 0.2:.2:2 using ALS initialization
randn('state',7256157);
Ainit = mat2cell(rand(sum(In),R),In,R);
alpha = .2:.2:2;
R = 5;
SIR = zeros(3*R,numel(alpha));

for k = 1:numel(alpha)
    options = struct('verbose',1,'tol',1e-6,'maxiters',500,'init',3,...
        'nonlinearproj',1,'alpha',alpha(k),'fixsign',0,'Ainit',{Ainit});
    [Y_alpha,Atemp,Ahat] = parafac_alpha(Y,R,options);
    SIR(:,k) = cell2mat(cellfun(@CalcSIR,A,Ahat,'uni',0));
end

figure
labels = mat2cell(sprintf('%.1f\n',alpha),1,4*ones(1,numel(alpha)))
boxplot(SIR,'label',labels,'notch','on')
xlabel('Alpha');ylabel('SIR [dB]');
```



(a) 3D volume for noisy input data tensor

(b) 3D volume of reconstructed tensor using NTF with FIT 99.9%

(c) Iso-surface visualization of the reconstructed tensor with the FIT 99.99%

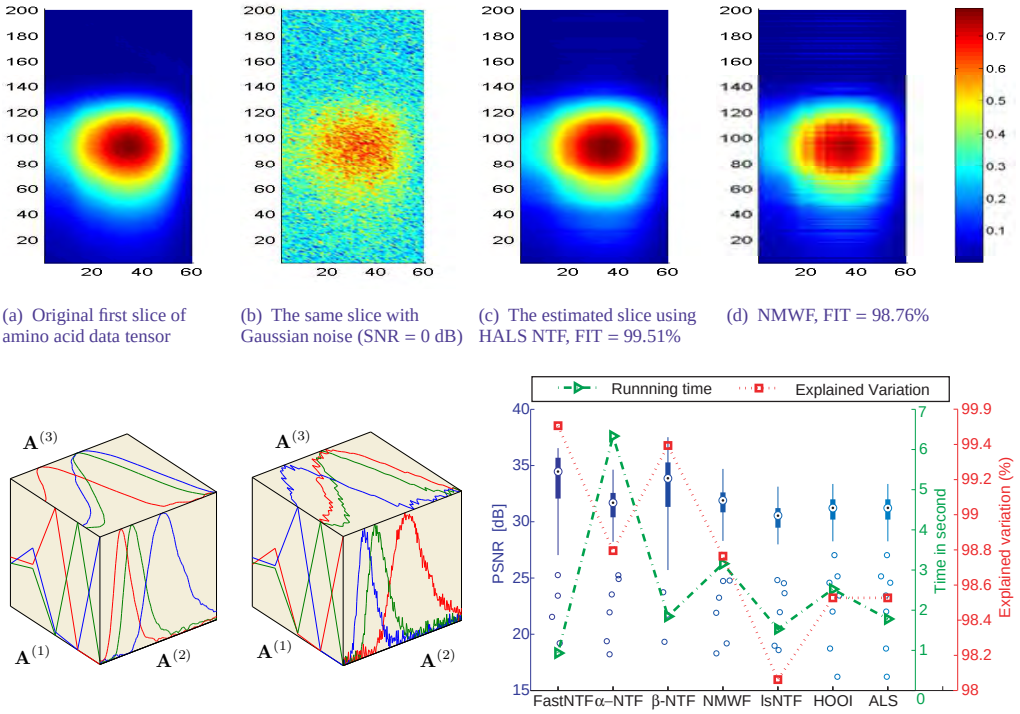
(d) Original 10-th slice of the noisy data tensor

(e) 10-th reconstructed slice using HALS NTF algorithm

(f) Estimated factors using HALS NTF algorithm

Fig. 7.10 NTF for a large-scale data tensor $\underline{\mathbf{Y}} \in \mathbb{R}_{+}^{500\times500\times500}$ degraded by additive Gaussian noise with SNR = 0 dB in Example 7.4.

performance is also illustrated by the visualization of the reconstructed exemplary horizontal 40-th slice (see Figure 7.9(f)) where the original noisy slice is shown as a reference in Figure 7.9(e). In addition, the performance for different values of parameters α and β for the Alpha



(a) Original first slice of amino acid data tensor

(b) The same slice with Gaussian noise (SNR = 0 dB)

(c) The estimated slice using HALS NTF, FIT = 99.51%

(d) NMWF, FIT = 98.76%

(e) Estimated factors using HALS NTF algorithm

(f) The estimated factors using NMWF algorithm

(g) PSNR, running time and FIT for various algorithms

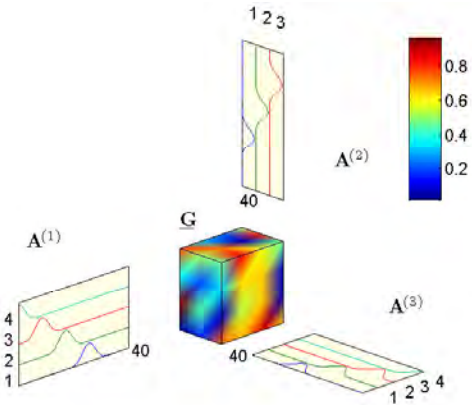
Fig. 7.11 Comparison of performance for various algorithms in Example 7.5. Illustration of estimated factors by the FAST HALS NTF in comparison with the multiplicative NMWF algorithm for three-way tensor factorization of amino acid data. (a)-(b) the first slices of the original amino acid tensor and the same tensor corrupted by large Gaussian noise, (c)-(d) the reconstructed slice using the HALS NTF and NMWF algorithms, (e)-(f) three estimated factors using the HALS NTF and NMWF algorithms (the estimated factors should be as smooth as possible), (g) comparison of distributions of PSNR (dB) for all slices, CPU time (seconds) and the explained variation (FIT %).

Listing 7.3 Example 7.3 generates membrane tensor of 40 slices.

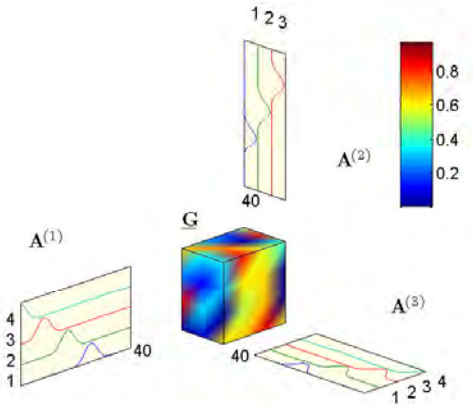
```
L0 = membrane(1,25); L0 = L0 - min(L0(:));
nlayers = 40;
Y = L0(:,:,ones(1,nlayers));
sc = reshape(1:nlayers,[1,1,nlayers]);
Y = bsxfun(@times,Y,sc);
```

and Beta HALS NTF algorithms are illustrated in Figures 7.9(g) and 7.9(h) with Peak Signal to Noise Ratio (PSNR) in the left (blue) axis and number of iterations in the right (red) axis.

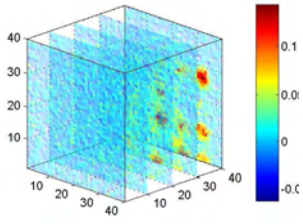
Example 7.4 *Large-scale tensor decomposition*



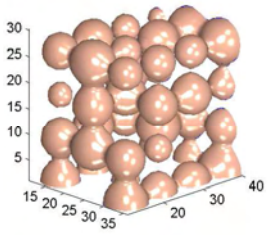
(a) Original core tensor and factors



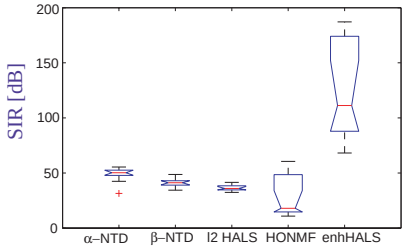
(b) Estimated factors with core tensor using the ℓ_2 HALS NTF algorithm for the data tensor without noise



(c) Frontal slices of the noisy data tensor with SNR = 0 dB



(d) The reconstructed tensor for noisy data with PSNR = 85.49 dB



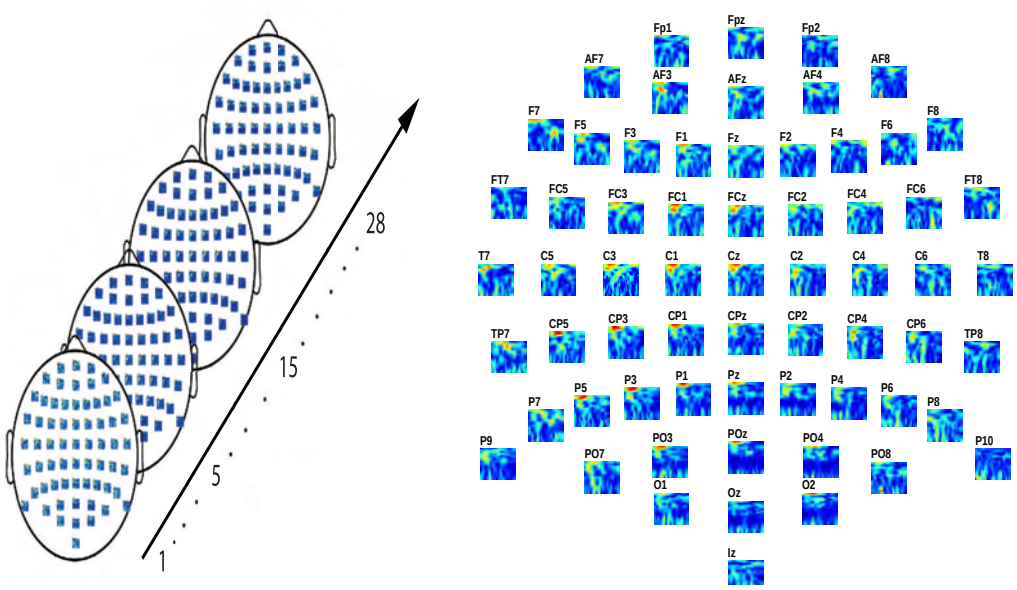
(e) SIR distributions for the reconstructed tensors for various Tucker decomposition algorithms

Fig. 7.14 Illustration of NTD for a data tensor with sparse factors (with and without noise) for Examples 7.8 and 7.9: (a)-(b) original and estimated factors and core tensors; (c) frontal slices of the noisy tensor with SNR = 0 dB; (d) reconstructed tensor by ℓ_2 HALS NTF algorithm (scaled to the unit ℓ_2-norm) with PSNR = 85.49 dB; (e) SIR distributions obtained by Alpha, Beta, ℓ_2 HALS and enhanced HALS NTD algorithms for the data tensor without noise.

The components (columns of factor matrices) were estimated by imposing additional orthogonality constraints and the stopping criterion used was the difference value of the explained variations (with the threshold of 10^{-6}). The estimated factors were initialized by the HOOI algorithm, and the orthogonal parameter λ_{ort} was set to 0.05 for all the estimated factors. Figures 7.14(b) illustrates the estimated component matrices and core tensor obtained by the basic HALS NTD algorithm after being rearranged to match the component order of the original factors. Next, we performed nonnegative Tucker decomposition using the multi-layer model (referred to as enhanced HALS or briefly enhHALS). The estimated core tensor $\underline{\mathbf{G}}$ at each layer was adjusted by the product of the observed tensor and the pseudo-inverse factors $\mathbf{A}^{(n)\dagger}$ as

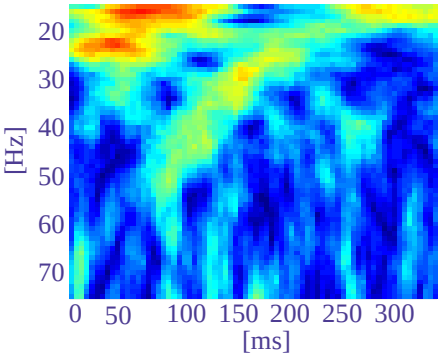
$$\underline{\mathbf{G}} = \underline{\mathbf{Y}} \times_1 \mathbf{A}^{(1)\dagger} \times_2 \mathbf{A}^{(2)\dagger} \cdots \times_N \mathbf{A}^{(N)\dagger} . \tag{7.200}$$

In each layer, the new core tensor and the estimated factors were used for the initialization of the succeeding layer. Using this model, we were able to improve the performance of nonnegative

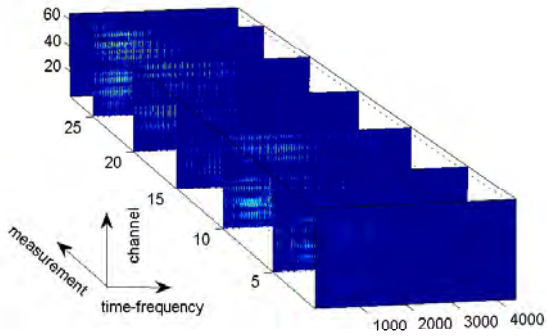


(a) Measurement arrays

(b) Topological montage of 64 time-frequency activities for the first subject: left hand stimuli



(c) A time-frequency plot zoomed in on channel C3

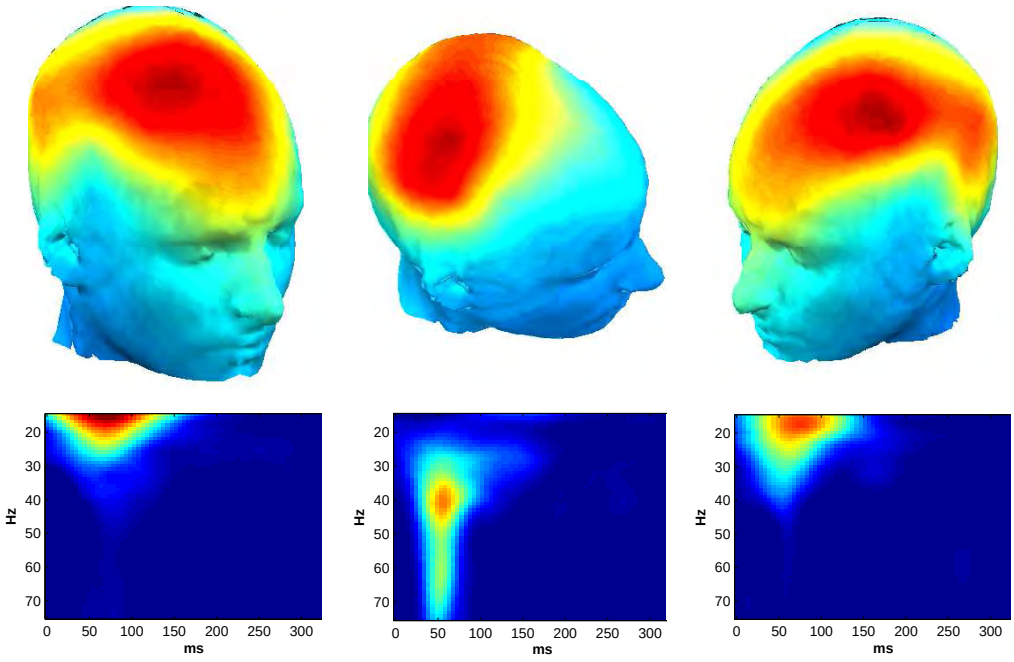


(d) A 3D ITPC tensor of 64 channels× 4392 (time-frequency) × 28 measurements

Fig. 7.22 Visualization of EEG signals `tutorialdataset2.zip` [61] in Example 7.16.

IPTC time-frequency measurements, whereas the performance comparisons are given in Table 7.3. The components of the first factor $\mathbf{A}^{(1)}$ are relative to the location of electrodes, and are used to illustrate the scalp topographic maps (the first row in Figure 7.23); whereas the second factor $\mathbf{A}^{(2)}$ represents the time-frequency spectral maps which were vectorized, and presented in the second row. Each component of these factors corresponds to a specific stimulus (left, right and both hand actions).

Example 7.17 *Decomposition and classification of visual and auditory EEG signals*



(a) Left hand stimuli (b) Gamma activity of both stimuli (c) Right hand stimuli

Fig. 7.23 EEG analysis based on the FAST HALS NTF for Example 7.16, the component matrices are $\mathbf{A}^{(1)}$ for a scalp topographic map (first row), and factor $\mathbf{A}^{(2)}$ for spectral (time-frequency) map (second row) (see [61] for details). Results are consistent with the previous analysis [61] but convergence properties of the FAST HALS NTF algorithm are different.

We illustrate decomposition and classification of EEG signals according to the nature of the stimulus: visual or auditory for the benchmark `EEG_AV_stimuli` [9, 3, 60, 86]. The stimuli were

1. Auditory stimulus with a single tone of 2000 Hz of 30 ms duration.
2. Visual stimulus in the form of a 5×5 checkerboard (600×600 pixels) displayed on a LCD screen (32×25cm). The stimulus duration was also 30 ms.
3. Both the auditory and the visual stimuli simultaneously.

A single class (stimulus type) consisted $N = 25$ trials, and was stored in a separate file. In each trial, EEG signals were recorded from 61 channels (except channels VEOG, HEOG, FP1) during 1.5 seconds after stimulus presentation at a sampling rate of 1 kHz. All the EEG signals in one class formed a third-order tensor of 25 trials $\times$ 1500 time samples $\times$ 61 channels. Hence, the full dataset is a 4-way tensor of size 25 trials $\times$ 1500 samples $\times$ 61 channels $\times$ 3 classes.

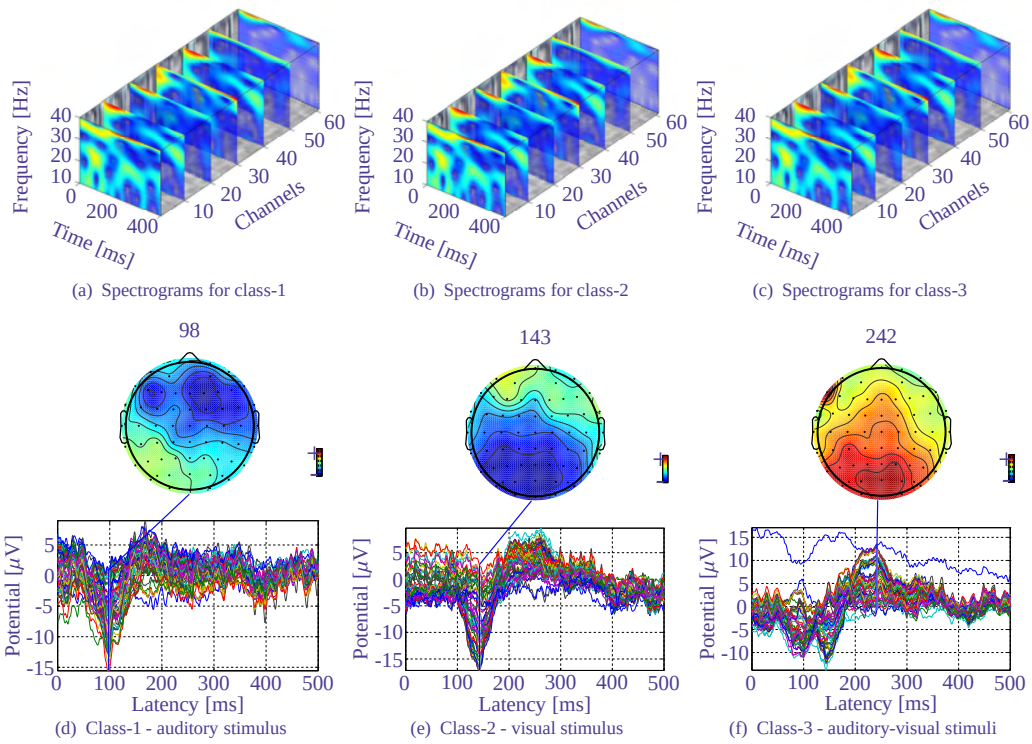
These three types of event related potentials can be classified according to the latency at which their components occur after stimulus presentation. The latency features of the three classes (corresponding to three different conditions) in the time domain can be found in Figures 7.24(d), 7.24(e) and 7.24(f). For instance, there is a P100 peak appearing around at 100 ms

following the stimulus for auditory stimuli as shown in Figure 7.24(d). We applied the nonnegative Tucker decomposition in analyzing the dataset in order to find the complex interactions and relationships between components expressing three modes: channels (space), spectra (time frequency representation), and classes (corresponding to three stimuli).

First, the EEG signals were transformed by the complex Morlet wavelet into the time-frequency spectrograms of 31 frequency bins (10-40 Hz) × 126 time frames (0-500ms) to form a raw data tensor $\underline{\mathbf{W}}$ of 61 channels × 31 frequency bins × 126 time frames × 25 trials × 3 classes. To analyze this spectral tensor, we averaged the wavelet coefficient magnitudes along all the trials as

$$y_{c,f,t,l} = \frac{1}{N}\sum_{n}^{N} \left| w_{c,f,t,n,l} \right|. \tag{7.201}$$

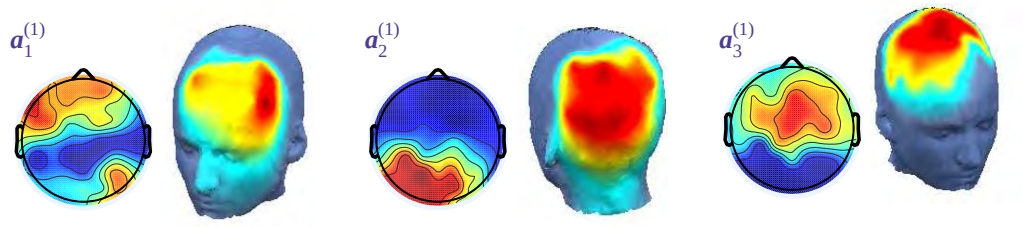
This data tensor corresponds to the time-frequency transformed Event Related Potentials (ERP) [37]. In practice, to reduce the high computational cost, we computed averages of the tensor along 25 trials, to obtain 183 (61 channels × 3 classes) EEG signals for all three classes (each with 1500 time samples); then transformed them to the time-frequency domain. In this way, we avoided wavelet transformation for all 4575 (61 channels × 25 trials × 3 classes) EEG signals. The preprocessed data tensor $\underline{\mathbf{Y}}$ has size of 61 channels × 31 frequency bins × 126 time frames × 3 classes. Figures 7.24(a), 7.24(b), and 7.24(c) show some selected spectra for three stimulus classes.



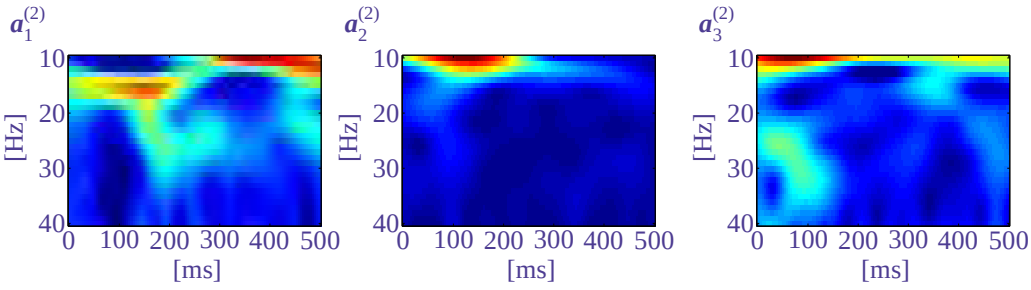
(a) Spectrograms for class-1 (b) Spectrograms for class-2 (c) Spectrograms for class-3

(d) Class-1 - auditory stimulus (e) Class-2 - visual stimulus (f) Class-3 - auditory-visual stimuli

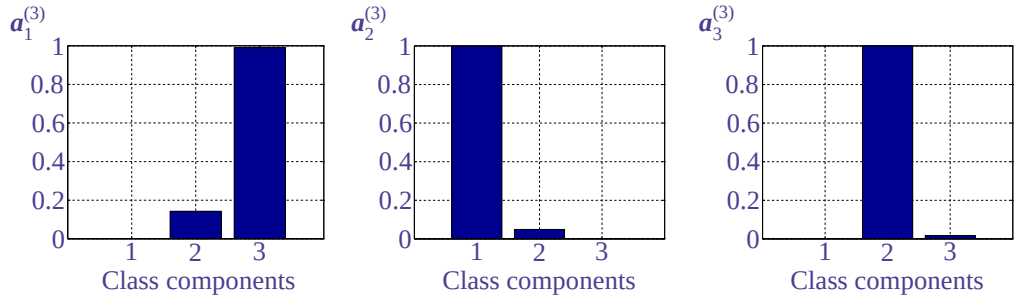
Fig. 7.24 Spectrograms and corresponding ERP for three types of stimuli in Example 7.17: (a)-(d) auditory, (b)-(e) visual, (c)-(f) auditory-visual stimulus.



(a) Topographic and Spherically-splined EEG field maps - factor $\mathbf{A}^{(1)}$



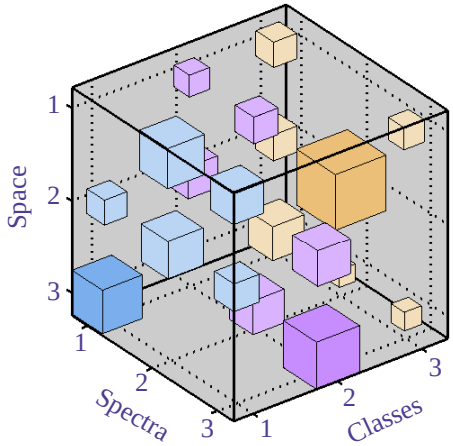
(b) Spectrograms - factor $\mathbf{A}^{(2)}$



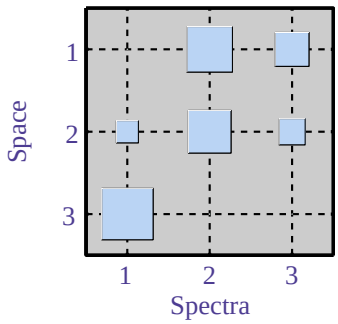
(c) Classes - factor $\mathbf{A}^{(3)}$

Fig. 7.25 Visualization of components of the NTD model in Example 7.17. (a) factor 1 $\mathbf{A}^{(1)}$ characterizes spatial components displayed in topographic maps and sherically-spline EEG field maps; (b) spectral components expressed by factor $\mathbf{A}^{(2)}$; (c) expression of factor $\mathbf{A}^{(3)}$ for 3 classes: component 1 $\boldsymbol{a}_1^{(3)}$ - auditory-visual class, component 2 $\boldsymbol{a}_2^{(3)}$ - Auditory class, and component 3 $\boldsymbol{a}_3^{(3)}$ - visual class-2 in Example 7.17.

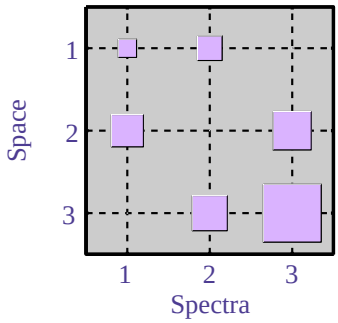
Finally, we reshaped data tensor $\underline{\mathbf{Y}}$ into a third-order nonnegative tensor of the size $61 \times 3906 \times 3$. The Nonnegative Tucker Decomposition model was chosen to decompose the preprocessed tensor data, and the ℓ_2 HALS NTD was selected to extract underlying components. We set the number of components to three, that is, the size of core tensor was $3 \times 3 \times 3$. The listing code 7.12 shows how the data tensor was processed and results were visualized. The three estimated components of the factor $\mathbf{A}^{(1)}$ express spatial activations (channels) distributed over 61 channels. The topographic maps and the corresponding spherically-splined field maps are displayed in Figure 7.25(a). The three basis spectral components $\boldsymbol{a}_{j_2}^{(2)}$, $j_2 = 1, 2, 3$ were reshaped and displayed in Figure 7.25(b). The three components $\boldsymbol{a}_{j_3}^{(3)}$, $j_3 = 1, 2, 3$ indicating the category of stimuli are plotted in Figure 7.25(c). Using such multi-way analysis, the three classes of stimuli were clearly classified, as illustrated by Table 7.6.



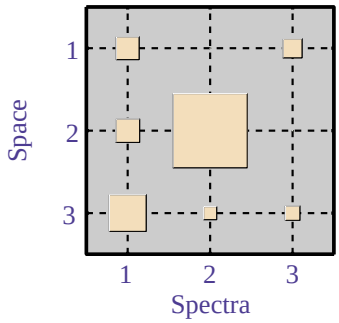
(a) The 3-D Hinton diagram of the core tensor of $\underline{\mathbf{G}}$



(b) Hinton diagram of the frontal slice $\mathbf{G}_1$ (corresponding to class-1)



(c) Hinton diagram of the frontal $\mathbf{G}_2$

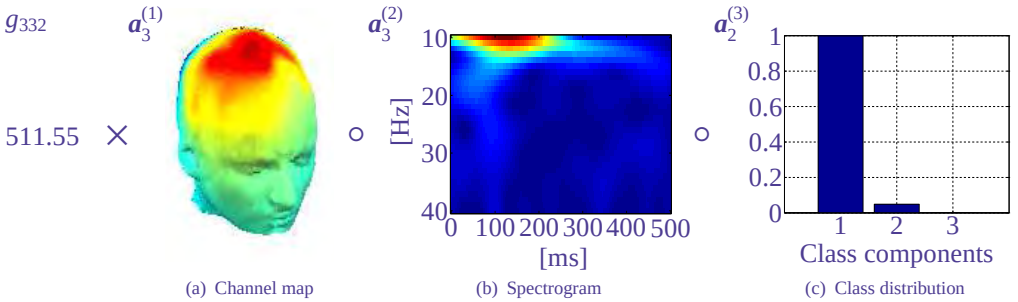


(d) Hinton diagram of the frontal slice $\mathbf{G}_3$

Fig. 7.26 Illustration of the core tensor via Hinton diagrams: (a) full core tensor $\underline{\mathbf{G}}$ in 3-D mode; (b)-(d) Frontal slices $\mathbf{G}_j = \underline{\mathbf{G}}_{:,:,j}$, $j = 1, 2, 3$ express the interaction of the j-th component $\boldsymbol{a}_j^{(3)}$ with components expressing channel and spectrogram in factors $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$: Auditory class-1 - $\boldsymbol{a}_2^{(3)}$ concentrates mainly on coefficient g_{332}, or spreads on $\boldsymbol{a}_3^{(1)}$ by spectrogram $\boldsymbol{a}_3^{(2)}$; Visual class-2 - $\boldsymbol{a}_3^{(3)}$ spreads on $\boldsymbol{a}_2^1$ by spectrogram $\boldsymbol{a}_2^{(2)}$, class-2 - $\boldsymbol{a}_1^{(3)}$ spreads on $\boldsymbol{a}_3^{(1)}$, $\boldsymbol{a}_2^{(1)}$, and $\boldsymbol{a}_1^{(1)}$ by spectrogram $\boldsymbol{a}_1^{(2)}$, $\boldsymbol{a}_2^{(2)}$ and $\boldsymbol{a}_3^{(2)}$. Darker colors (in (a) indicate dominant components for each of the three factors.

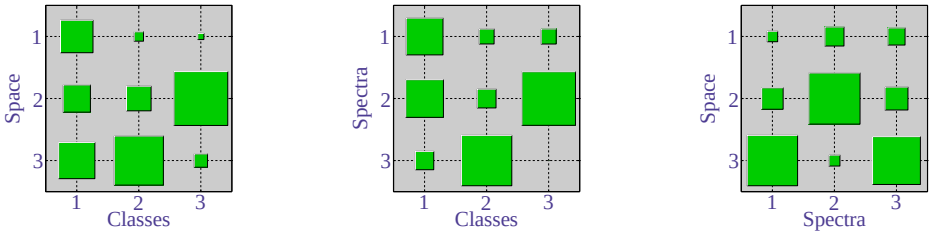
strongly the spatial component $\boldsymbol{a}_{j_1}^{(1)}$ affects the category component $\boldsymbol{a}_{j_3}^{(3)}$ is expressed as

$$JR_{j_3}^{j_1} = \frac{\sum_{j_2=1}^{J_2} g_{j_1 j_2 j_3}^2}{\sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} g_{j_1 j_2 j_3}^2}. \tag{7.202}$$



(a) Channel map (b) Spectrogram (c) Class distribution

Fig. 7.28 Visualization of the three significant components $\boldsymbol{a}_3^{(1)}$, $\boldsymbol{a}_3^{(2)}$, $\boldsymbol{a}_2^{(3)}$ for a dominant rank-one tensor in the NTD for the visual class-2.



(a) *JR* indices between channels and classes (b) *JR* indices between spectrograms and classes (c) *JR* indices between spectrograms and classes

Fig. 7.29 Hinton diagrams of the Joint Rate indices between (a) spatial and category components, (b) spectral and category components and (c) spatial and spectral components.

among components for each class can be evaluated directly from these plots; for example, the $JR_{2/3}^{3/1}$ has the largest value. This means that the the 2nd category component affects primarily the third spatial component, the 3rd category category component correlates mainly with the second spatial component ($JR_{3/3}^{2/1}$), whereas the 1st one spreads over all the three components. In Figure 7.29, we show Hinton diagrams which illustrate the strength of interactions of spatial and spectral components with respect to the category components, and their mutual interactions.

7.7.5 Application of Tensor Decomposition in Brain Computer Interface and Classification of Motor Imagery Tasks

In comprehensive Brain Computer Interface (BCI) studies, the brain data structures often contain higher-order ways (modes) such as trials, tasks conditions, subjects, and groups in addition to the intrinsic dimensions of space, time and frequency. In fact, specific mental tasks or stimuli are often presented repeatedly in a sequence of trials leading to a large volume stream of data encompassing many dimensions: Channels (space), time-frequencies, trials, and conditions [5, 69, 12, 63, 22].

As Figure 7.30 shows, most existing BCI systems use three basic signal-processing blocks. The system applies a preprocessing step to remove noise and artifacts (mostly related to ocular, muscular, and cardiac activities) to enhance the SNR. In the next step, the system performs feature extraction and selection to detect the specific target patterns in brain activity that encode the

user's mental tasks, detect an event-related response, or reflect the subject's motor intentions. The last step is aimed at translating or associating these specific features into useful control (command) signals to be sent to an external device. Ideally, the translator block supports the noncontrol state, because without NC support, all classifier output states are considered intentional. With NC support, the user can control whether or not the output is considered intentional. In the latter case, a self-paced NC state paradigm is monitored continuously, where users can perform specific mental tasks whenever they want.

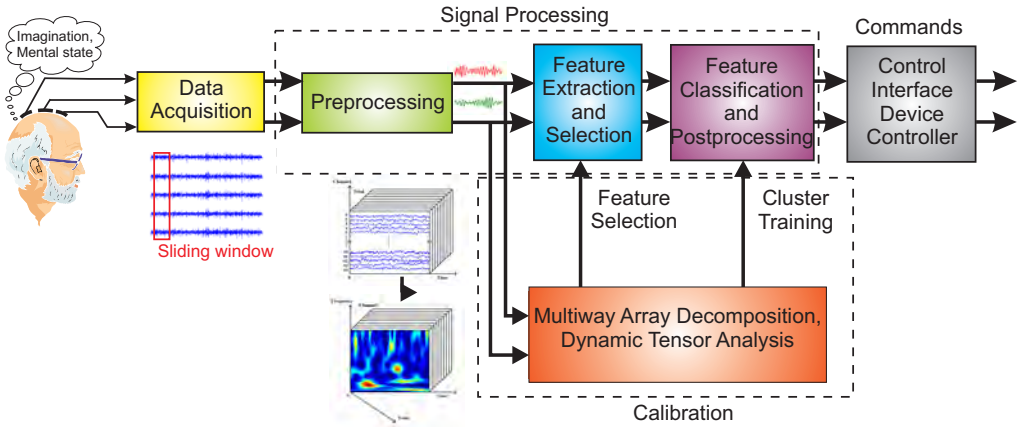


Fig. 7.30 Multistage procedure for online BCI. Preprocessing and feature extraction play a key role in real-time, high-performance BCI systems. In the calibration step, most BCI ERP studies are based on multi-subject and multi-condition analysis. For such scenarios, the tensor decompositions naturally encompass extra modalities such as trials, subjects, conditions, and so on and allow the system to find the dominant sources of activity differences without supervision.

In the preprocessing step, the system can decompose the recorded brain signals into useful signal and noise subspaces using standard techniques (like ICA or nonlinear adaptive filtering). One promising approach to enhance signals, extract significant features, and perform some model reduction is to apply blind source separation techniques, especially multiway blind source separation and multiway array (tensor) decomposition.

A promising and popular approach based on the passive endogenous paradigm is to exploit temporal/spatial changes or spectral characteristics of the sensorimotor rhythm (SMR) oscillations, or mu-rhythm (8-12 Hz) and beta rhythm (18-25 Hz). These oscillations typically decrease during, or immediately before a movement event related desynchronization (ERD). External stimuli-visual, auditory, or somatosensory – drive exogenous BCI tasks, which usually do not require special training. Two often used paradigms are P300 and steady-state visually evoked potentials (SSVEP). P300 is an event-related potential that appears approximately 300 ms after a relevant and rare event. SSVEP uses a flicker stimulus at relatively low frequency (typically, 5-45 Hz).

Another promising and related extension of BCI is to incorporate real-time neuro-feedback capabilities to train subjects to modulate EEG brain patterns and parameters such as ERPs, ERD, SMR, and P300 to meet a specific criterion or learn self-regulation skills where users change their EEG patterns in response to feedback. Such integration of neuro-feedback in BCI is an emerging technology for rehabilitation, but it is also a new paradigm in neuroscience

that might reveal previously unknown brain activities associated with behavior or self-regulated mental states (see Figure 7.31). In a neuro-feedback-modulated response (active endogenous)

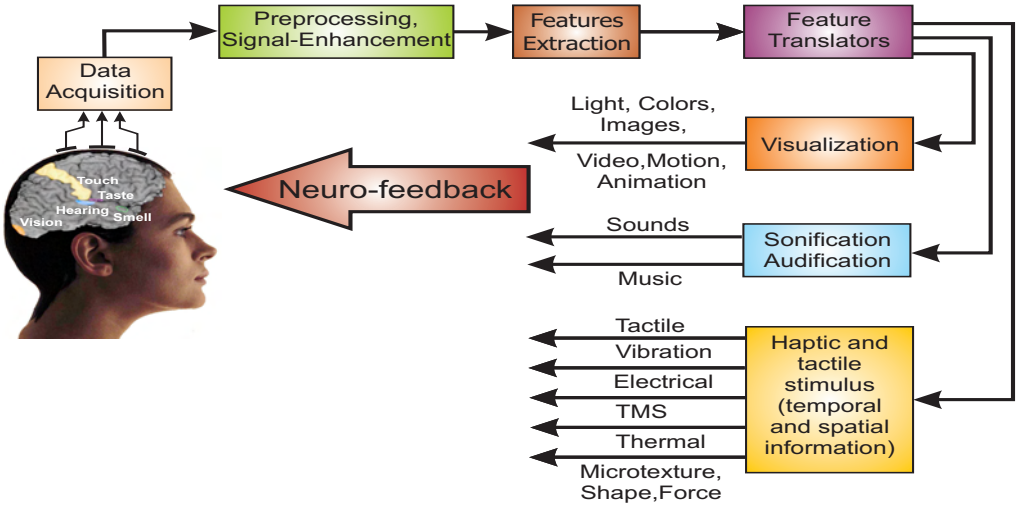
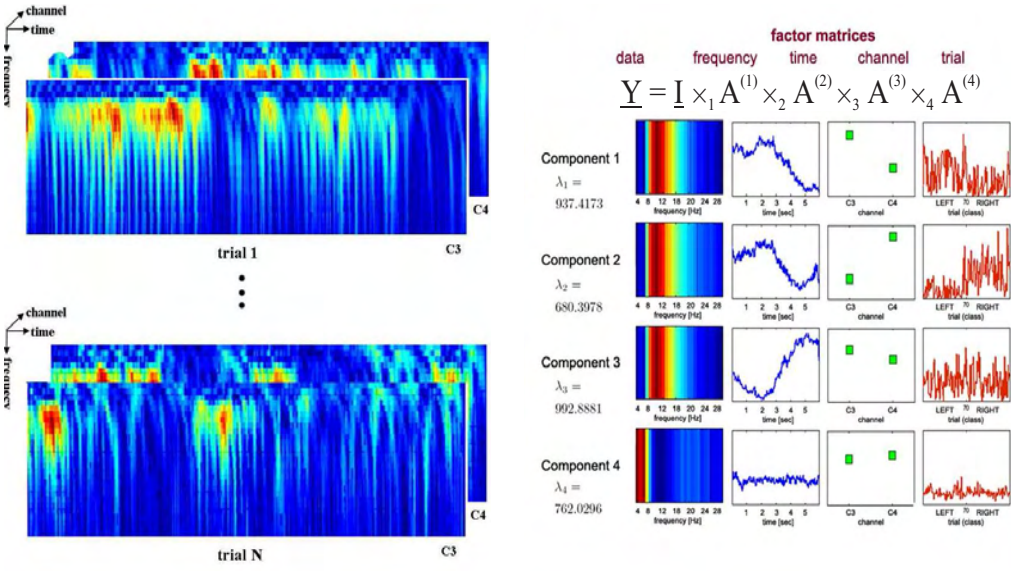


Fig. 7.31 Conceptual BCI system with various kinds of neuro-feedback combined with Human Computer Interactions (HCI). The development of a BCI must handle two learning systems: The computer should learn to discriminate between different complex patterns of brain activity as accurately as possible, and BCI users should learn via different neuro-feedback configurations to modulate and self-regulate or control BCI activity.

BCI paradigm, users learn to generate specific brain waves through various mental strategies while monitoring the outcome of their efforts in near real time. Typically, the user visualizes the preprocessed and translated target brain signal to increase motivation and improve recognition accuracy. However, the successful control of the interface in this way usually requires quite a long process and up to several weeks of training. BCI neuro-feedback in any of these paradigms should be as speedy as possible, which requires fast real-time signal processing algorithms. Recent neurofeedback experiments confirm that performance increases with richer feedback. For example, a simple bar gives lower accuracies than a full immersive 3D dynamic visualization or sonification.

Standard matrix factorizations, such as PCA, SVD, ICA and NMF and their variants, are invaluable tools for BCI feature selection, dimensionality reduction, noise reduction, and mining [5, 69, 12, 63, 46, 56, 57, 59, 58]. However, they have only two modes or 2-way representations (e.g., channels and time) and therefore have severe intrinsic limitations. For such kind of data 2-way matrix factorizations (ICA, NMF) or "flat-world view" may be insufficient for future BCI systems. In order to obtain more natural representations of the original multi-dimensional data structure, it is necessary to use tensor decomposition approaches, since additional dimensions or modes can be retained only in multi-linear models to produce structures that are unique and which admit interpretations that are neurophysiologically meaningful [62, 1].

Recent advances in developing high-spatial density array EEG have called for multi-dimensional signal processing techniques (referred to here as the multi-way analysis (MWA), multi-way array (tensor) factorization/decomposition or dynamic tensor analysis (DTA) or window-based



(a) 4-way tensor based on $N = 140$ three-way EEG spectral tensors

(b) 4 components explain the 4-way EEG tensor

Fig. 7.32 Decomposition of the 4-way time-frequency-spectral EEG data into basic components during motor imaginary tasks.

C4 electrodes during right and left hand motor imagery (70 left-hand trials and 70 right-hand ones). Each trial is represented by the three-way tensor (frequency × time × channel) in Figure 7.32(a). A spectral tensor was factorized into four components displayed in Figure 7.32(b). Component 1 corresponds to left-hand imagery (due to the significantly greater C3 weight than the C4 one), component 2 represents the right-hand imagery and component 3 reflects both left and right hand imagery stimuli. The theta rhythm (4-8 Hz), which is related to concentration is represented by component 4 [57].

Figure 7.33 illustrates the experimental results using the 4-way tensor decomposition of multi-channel (62 electrodes) EEG data (channel, frequency, time, conditions) into four component (factor) matrices in the space (topographic map), frequency, time and class domain shown from left-to-right on this figure. In order to find the most discriminative components for different classes (i.e., left hand and right hand motor imagery), we imposed a sparseness constraint on the class mode. Each row of Figure 7.32(b) represents one component of the factor matrices. From these plots components 4 and 5 are recognized as discriminative components corresponding to the motor imaginary tasks due to their scalp maps covering sensorimotor areas. Component 4 illustrates the ERD/ERS phenomena that indicates in the spatial distribution a larger amplitude on the left hemisphere and lower amplitude for the right hemisphere (see column 1), and the energy of oscillations dominated by the mu rhythm in frequency range mainly 8-12 Hz of mu rhythm (see column 2), and observation of quasi-stationary oscillations through the whole trial duration (see column 3). Hence a larger amplitude is shown for class-1(right-hand imagery) and lower amplitude on class-2 (left-hand imagery) conditions (column 4). Similarly, component 5 shows ERD on the left hemisphere and ERS on the right hemisphere. Components 1 and 3 show

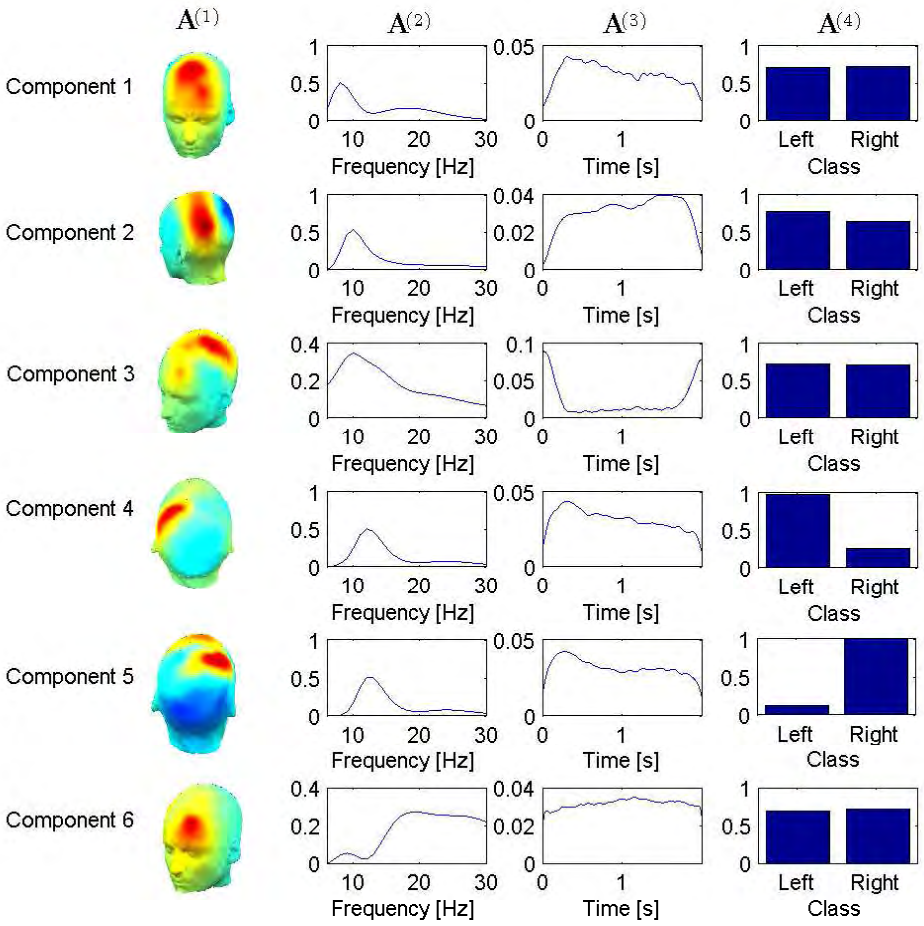


Fig. 7.33 Decomposition of 62 channels EEG signals into basis components. Four columns represent the four factors in the analysis.

the visual evoked potentials caused by the cue stimulus, which have spatial distribution over the visual cortex. Other components represent the existing artifacts (EOG, EMG) and other brain activities uncorrelated with event related potentials.

The multi-way analysis approach and the related concepts (tensor decompositions, especially their extensions to dynamic tensor analysis) presented in this chapter are only a subset of a number of promising and emerging signal processing and data mining tools with potential applications to future BCI systems. The main advantage of the presented approach is its flexibility when dealing with multi-dimensional data and the possibility to enforce various physical constraints.

References

1. E. Acar, C.A. Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23:10–18, 2007.

2. E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21:6–20, 2009.

3. A.H. Andersen and W.S. Rayens. Structure-seeking multilinear methods for the analysis of fMRI data. *NeuroImage*, 22:728–739, 2004.

4. C.A. Andersson and R. Bro. The N-way toolbox for MATLAB. *Chemometrics Intell. Lab. Systems*, 52(1):1–4, 2000.

5. F. Babiloni, A. Cichocki, and S. Gao. Brain computer interfaces: Towards practical implementations and potential applications. *Computational Intelligence and Neuroscience*, 2007.

6. B.W. Bader and T.G. Kolda. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software*, 32(4):635–653, 2006.

7. B.W. Bader and T.G. Kolda. Efficient MATLAB computations with sparse and factored tensors. Technical Report SAND2006-7592, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, December 2006.

8. B.W. Bader and T.G. Kolda. *MATLAB Tensor Toolbox Version 2.2*, January 2007.

9. H. Bakardjian and A. Cichocki. Extraction and classification of common independent components in single-trial crossmodal cortical responses. In *Proceedings of the 5th Annual Meeting of the International Multisensory Research Forum*, pages 26–27, Barcelona, Spain, June 2004.

10. J.R. Bergen, P. An, Th. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. pages 237–252. Springer-Verlag, 1992.

11. M. Berry, M. Browne, A. Langville, P. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.

12. B. Blankertz, M. Kawanabe, R. Tomioka, V. Nikulin, and K.-R. Müller. Invariant common spatial patterns: Alleviating nonstationarities in brain-computer interfacing. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 113–120. MIT Press, Cambridge, MA, 2008.

13. K.S. Booksh and B.R. Kowalski. Error analysis of the generalized rank annihilation method. *Journal of Chemometrics*, 8:45–63, 1994.

14. K.S. Booksh, Z. Lin, Z. Wang, and B.R. Kowalski. Extension of trilinear decomposition method with an application to the flow probe sensor. *Analytical Chemistry*, 66:2561–2569, 1994.

15. R. Bro. PARAFAC. Tutorial and applications. In *Special Issue 2nd Internet Conf. in Chemometrics (INCINC'96)*, volume 38, pages 149–171. Chemom. Intell. Lab. Syst, 1997.

16. C. Caiafa and A. Cichocki. Slice Oriented Decomposition: A new tensor representation for 3-way data. *(submitted to Journal of Signal Processing)*, 2009.

17. J. Chen and Y. Saad. On the tensor SVD and the optimal low rank orthogonal approximation of tensors. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 30:1709–1734, 2009.

18. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer, LNAI-4029*, 4029:548–562, 2006.

19. A. Cichocki and A.H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE (invited paper)*, March 2009.

20. A. Cichocki, A.H. Phan, and C. Caiafa. Flexible HALS algorithms for sparse non-negative matrix/tensor factorization. In *Proc. of 18-th IEEE workshops on Machine Learning for Signal Processing*, Cancun, Mexico, 16–19, October 2008.

21. A. Cichocki, A.H. Phan, R. Zdunek, and L.-Q. Zhang. Flexible component analysis for sparse, smooth, nonnegative coding or representation. In *Lecture Notes in Computer Science, LNCS-4984*, volume 4984, pages 811–820. Springer, 2008.

22. A. Cichocki, Y. Washizawa, T.M. Rutkowski, H. Bakardjian, A.H. Phan, S. Choi, H. Lee, Q. Zhao, L. Zhang, and Y. Li.

23. A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization. *Electronics Letters*, 42(16):947–948, 2006.

24. A. Cichocki and R. Zdunek. NMFLAB for Signal and Image Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

25. A. Cichocki and R. Zdunek. NTFLAB for Signal Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.

26. A. Cichocki and R. Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorizations. *Springer, LNCS-4493*, 4493:793–802, June 3–7 2007.

27. A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: Family of new algorithms. *Springer, LNCS-3889*, 3889:32–39, 2006.

28. A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2006*, volume 5, pages 621–624, Toulouse, France, May 14–19 2006.

29. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using Alpha and Beta divergencies. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07)*, volume III, pages 1393–1396, Honolulu, Hawaii, USA, April 15–20 2007.

30. A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-I. Amari. Novel multi-layer nonnegative tensor factorization with sparsity constraints. *Springer, LNCS-4432*, 4432:271–280, April 11–14 2007.

31. P. Comon. Tensor decompositions: State of the art and applications. In J.G. McWhirter and I.K. Proudler, editors, *Institute of Mathematics and its Applications Conference on Mathematics in Signal Processing*, pages 1–26. Oxford University Press, UK, 2001.

32. L. De Lathauwer. Decompositions of a higher-order tensor in block terms – Part I: Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 30(3):1022–1032, 2008. Special Issue on Tensor Decompositions and Applications.

33. L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications*, 24:1253–1278, 2000.

34. L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-(R1,R2,...,RN) approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

35. L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 30(3):1067–1083, 2008. Special Issue Tensor Decompositions and Applications.

36. L. De Lathauwer and J. Vandewalle. Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_n)$ reduction in multilinear algebra. *Linear Algebra Applications*, 391:31–55, November 2004.

37. A. Delorme and S. Makeig. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods*, 134:9–21, 2004.

38. I. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Neural Information Proc. Systems*, pages 283–290, Vancouver, Canada, December 2005.

39. L. Eldén and B. Savas. A Newton–Grassmann method for computing the best multi-linear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM Journal on Matrix Analysis and Applications*, 31:248–271, 2009.

40. N.M. Faber. On solving generalized eigenvalue problems using matlab. *Journal of Chemometrics*, 11:87–91, 1997.

41. N.M. Faber, L.M.C. Buydens, and G. Kateman. Generalized rank annihilation. iii: practical implementation. *Journal of Chemometrics*, 8:273–285, 1994.

42. M.P. Friedlander and K. Hatz. Computing nonnegative tensor factorizations. *Computational Optimization and Applications*, 23(4):631–647, March 2008.

43. T. Hazan, S. Polak, and A. Shashua. Sparse image coding using a 3D non-negative tensor factorization. In *Proc. Int. Conference on Computer Vision (ICCV)*, pages 50–57, 2005.

44. D. Heeger. Image registration software. Copyright 1997, 2000 by Stanford University, 2000.

45. P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

46. P.R. Kennedy, R.A.E. Bakay, M.M. Moore, K. Adams, and J. Goldwaithe. Direct control of a computer from the human central nervous system. *IEEE Transactions on Rehabilitation Engineering*, 8(2):198–202, June 2000.

47. M. Kim and S. Choi. Monaural music source separation: Nonnegativity, sparseness, and shift-invariance. *Springer LNCS*, 3889:617–624, 2006.

48. Y.-D. Kim and S. Choi. Nonnegative Tucker Decomposition. In *Proc. of Conf. Computer Vision and Pattern Recognition (CVPR-2007)*, Minneapolis, Minnesota, June 2007.

49. Y.-D. Kim, A. Cichocki, and S. Choi. Nonnegative Tucker Decomposition with Alpha Divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2008*, Nevada, USA, 2008.

50. T.G. Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories, 2006.

51. T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):(in print), September 2009.

52. P.M. Kroonenberg. *Applied Multiway Data Analysis*. John Wiley & Sons Ltd, New York, 2008.

53. J.B. Kruskal, R.A. Harshman, and M.E. Lundy. How 3-MFA data can cause degenerate parafac solutions, among other relationships. In *Multiway Data Analysis.*, pages 115–122. North-Holland Publishing Co., Amsterdam, The Netherlands, 1989.

54. A. N. Langville, C. D. Meyer, and R. Albright. Initializations for the nonnegative matrix factorization. In *Proc. of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, August 20–23 2006.

55. D.D. Lee and H.S. Seung. *Algorithms for Nonnegative Matrix Factorization*, volume 13. MIT Press, 2001.

56. H. Lee, A. Cichocki, and S. Choi. Nonnegative matrix factorization for motor imagery EEG classification. *LNCS*, 4132:250–259, 2006.

57. H. Lee, Y.-D. Kim, A. Cichocki, and S. Choi. Nonnegative tensor factorization for continuous EEG classification. *International Journal of Neural Systems*, 17(4):1–13, August 2007.

58. Y.-Q. Li, A. Cichocki, and S. Amari. Blind estimation of channel parameters and source components for EEG signals: a sparse factorization approach. *IEEE Transactions on Neural Networks*, 17(2):419–431, 2006.

59. P. Martinez, H. Bakardjian, and A. Cichocki. Fully online, multi-command brain computer interface with visual neurofeedback using SSVEP paradigm. *Journal of Computational Intelligence and Neuroscience*, 2007, 2007.

60. E. Martnez-Montes, J.M. Snchez-Bornot, and P.A. Valds-Sosa. Penalized PARAFAC analysis of spontaneous EEG recordings. *Statistica Sinica*, 18:1449–1464, 2008.

61. M. Mørup, L.K. Hansen, and S.M. Arnfred. ERPWAVELAB a toolbox for multi-channel analysis of time-frequency transformed event related potentials. *Journal of Neuroscience Methods*, 161:361–368, 2007.

62. M. Mørup, L.K. Hansen, and S.M. Arnfred. Algorithms for Sparse Nonnegative Tucker Decompositions. *Neural Computation*, 20:2112–2131, 2008.

63. K.-R. Müller, M. Krauledat, G. Dornhege, G. Curio, and B. Blankertz. Machine learning and applications for brain-computer interfacing. 4557, 2007.

64. O. Nestares and D.J. Heeger. Robust multiresolution alignment of MRI brain volumes. In *Magnetic Resonance in Medicine, 43:705715*, pages 705–715, 2000.

65. A.H. Phan and A. Cichocki. Fast and efficient algorithms for nonnegative Tucker decomposition. In *Proc. of The Fifth International Symposium on Neural Networks, Springer LNCS-5264*, pages 772–782, Beijing, China, 24–28, September 2008.

66. A.H. Phan and A. Cichocki. Multi-way nonnegative tensor factorization using fast hierarchical alternating least squares algorithm (HALS). In *Proc. of The 2008 International Symposium on Nonlinear Theory and its Applications*, Budapest, Hungary, 2008.

67. A.H. Phan and A. Cichocki. Fast nonnegative Tucker decomposition via HOOI tensor approximation. In *ICONIP-2009 (submitted)*, Bangok, Thailand, December 2009.

68. A.H. Phan and A. Cichocki. Local learning rules for nonnegative Tucker decomposition. *(submitted)*, 2009.

69. P. Sajda, K-R. Müller, and K.V. Shenoy. Brain computer interfaces. *IEEE Signal Processing Magazine*, January 2008.

70. F. Samaria and A.C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, 1994.

71. E. Sanchez and B.R. Kowalski. Tensorial resolution: a direct trilinear decomposition. *J. Chemometrics*, 4:29–45, 1990.

72. M. Sattler, R. Sarlette, and R. Klein. Efficient and realistic visualization of cloth. In *EGSR '03: Proceedings of the 14th Eurographics Symposium on Rendering*, pages 167–177, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

73. B. Savas and L. Eldén. Handwritten digit classification using higher order singular value decomposition. *Pattern Recognition*, 40:993–1003, 2007.

74. A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proc. of the 22-th International Conference on Machine Learning*, Bonn, Germany, 2005.

75. A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006.

76. A. Smilde, R. Bro, and P. Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. John Wiley & Sons Ltd, New York, 2004.

77. J. Sun. *Incremental Pattern Discovery on Streams, Graphs and Tensors*. PhD thesis, CMU-CS-07-149, 2007.

78. J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proc.of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383, 2006.

79. C. Tallon-Baudry, O. Bertrand, C. Delpuech, and J. Pernier. Stimulus specificity of phase-locked and non-phase-locked 40 Hz visual responses in human. *Journal of Neuroscience*, 16 (13):4240–4249, 1996.

80. The BTF Database Bonn. *CEILING Sample*. http://btf.cs.uni-bonn.de/download.html.

81. M.K. Titsias. *Unsupervised Learning of Multiple Objects in Images*. PhD thesis, School of Informatics, University of Edinburgh, June 2005.

82. L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

83. P.D. Turney. Empirical evaluation of four tensor decomposition algorithms. Technical report, National Research Council, Institute for Information Technology, 2007,Technical Report ERB-1152. (NRC 49877).

84. J.H. Wang, P.K. Hopke, T.M. Hancewicz, and S.-L. Zhang. Application of modified alternating least squares regression to spectroscopic image analysis. *Analytica Chimica Acta*, 476:93–109, 2003.

85. W. Wang. Squared Euclidean distance based convolutive non-negative matrix factorization with multiplicative learning rules for audio pattern separation. In *Proc. 7th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2007)*, Cairo, Egypt, December 15-18 2007.

86. Z. Wang, A. Maier, N.K. Logothetis, and H. Liang. Single-trial decoding of bistable perception based on sparse nonnegative tensor decomposition. *Journal of Computational Intelligence and Neuroscience*, 30:1–10, 2008.

87. Q. Wu, T. Xia, and Y. Yu. Hierarchical tensor approximation of multi-dimensional images. In *14th IEEE International Conference on Image Processing*, volume IV, pages 49–52, San Antonio, 2007.

88. R. Zass and A. Shashua. Doubly stochastic normalization for spectral clustering. In *Neural Information Processing Systems (NIPS)*, Vancuver, Canada, Dec. 2006.

8

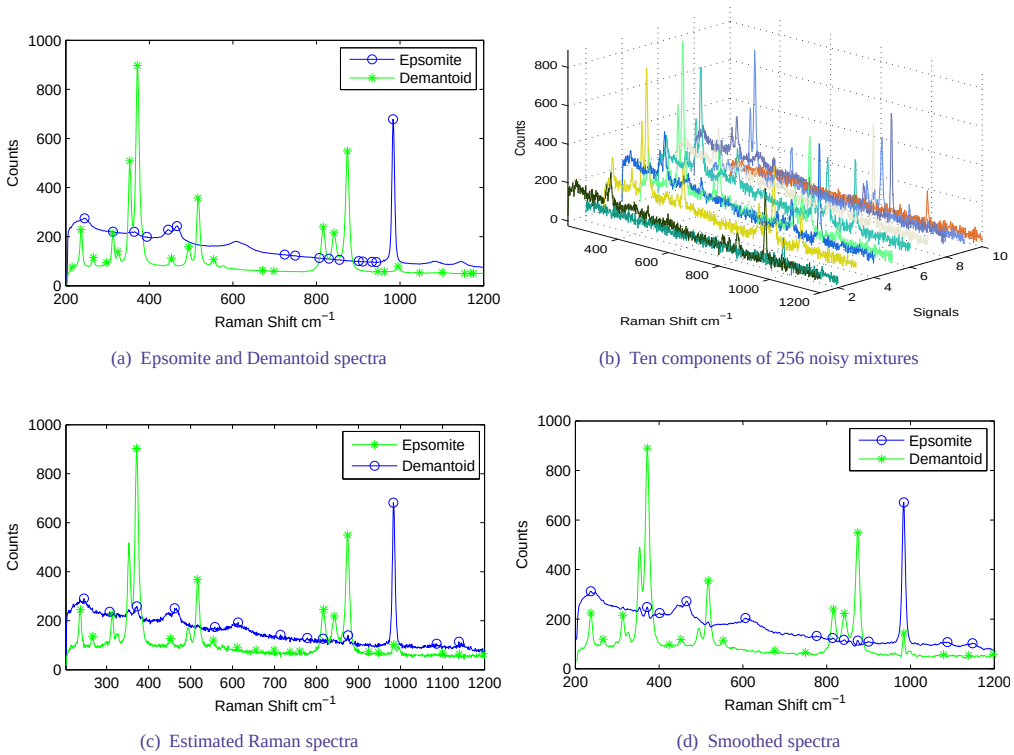
Selected Applications

Early applications of the concept inherited by NMF appeared in the middle 1990s under the name Positive Matrix Factorization (PMF). This kind of factorization was applied by Paatero *et al.* [82] to process environmental data, however, the popularity of NMF significantly increased since Lee and Seung published simple multiplicative NMF algorithms which they applied to image data [62, 63]. At present, NMF and its variants have already found a wide spectrum of applications.

In this chapter, we briefly discuss some selected applications of NMF and multi-dimensional array decompositions, with special emphasis on those applications to which the algorithms described in the previous chapters are applicable. We review the following applications: data clustering [100, 67, 27, 24, 126, 5, 14, 16, 127], text mining [119, 100, 86, 67], email surveillance [9], musical instrument classification [8, 6, 7], face recognition [42, 43, 45, 120, 121, 128, 113], handwritten digit recognition [61], texture classification [89, 87], Raman spectroscopy [95, 65, 75], fluorescence spectroscopy [38, 37, 46], hyperspectral imaging [95, 85, 28, 50, 76, 40, 106], chemical shift imaging [96, 95, 11], and gene expression classification [55, 13, 14, 53, 83, 84, 73, 33, 109].

8.1 CLUSTERING

Data clustering can be regarded as an unsupervised classification of patterns into groups (clusters) that have similar features, as illustrated in Figure 8.1. The data points reside in a 2D space and can be classified into three disjoint groups. The grouping can basically be obtained with hierarchical or partitioning techniques [49]. In hierarchical clustering, a nested series of partitions is performed with varying dissimilarity level whereas partitioned clustering techniques yield a single partition of data for which a given clustering criterion is optimized (usually locally). The former technique is usually more robust but due to its high computational complexity it is not



(a) Epsomite and Demantoid spectra

(b) Ten components of 256 noisy mixtures

(c) Estimated Raman spectra

(d) Smoothed spectra

Fig. 8.9 Raman spectra: (a) target spectra of Epsomite and Demantoid, (b) ten sample components of 256 mixtures, (c) estimated spectra of Epsomite and Demantoid, (d) NMF with smoothness constraints.

the incident ones are emitted. The molecular structure of the species can thereby be determined, by analyzing the spectrum of the emitted light with respect to the frequency of monochromatic light.

Nevertheless, the spectral analysis is not straightforward since the observed spectra are instantaneous mixtures of pure species spectra and other intermediate species spectra. The problem of extracting pure spectra from the mixtures can be formulated in terms of a blind source separation problem, and solved with many algorithms for ICA [18]. Furthermore, considering intrinsic nonnegativity constraints on spectra and their concentrations/abundances, the problem can be solved with NMF, which is more profitable since the separated spectra could be partially statistically dependent.

Assuming the mixed spectra are observed by I sensors and each spectrum has T samples, all the observations can be stored in the observation matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$. Applying NMF to $\mathbf{Y}$ under the assumption that J is the number of constituent spectra, we obtain the abundance matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and the nonnegative matrix $\mathbf{X} \in \mathbb{R}^{J \times T}$ of the pure spectra.

Applications of NMF to fluorescence spectroscopy can be found, e.g., in [38, 37, 78]. Gobinet *et al.* [37] applied NMF to analyze a distribution of some organic compounds such as bound ferulic acid, free ferulic acid, and p-coumaric acid in durum wheat and barley grains. They used a laser scanning microspectrofluorometer to acquire fluorescence signals. A transversal section

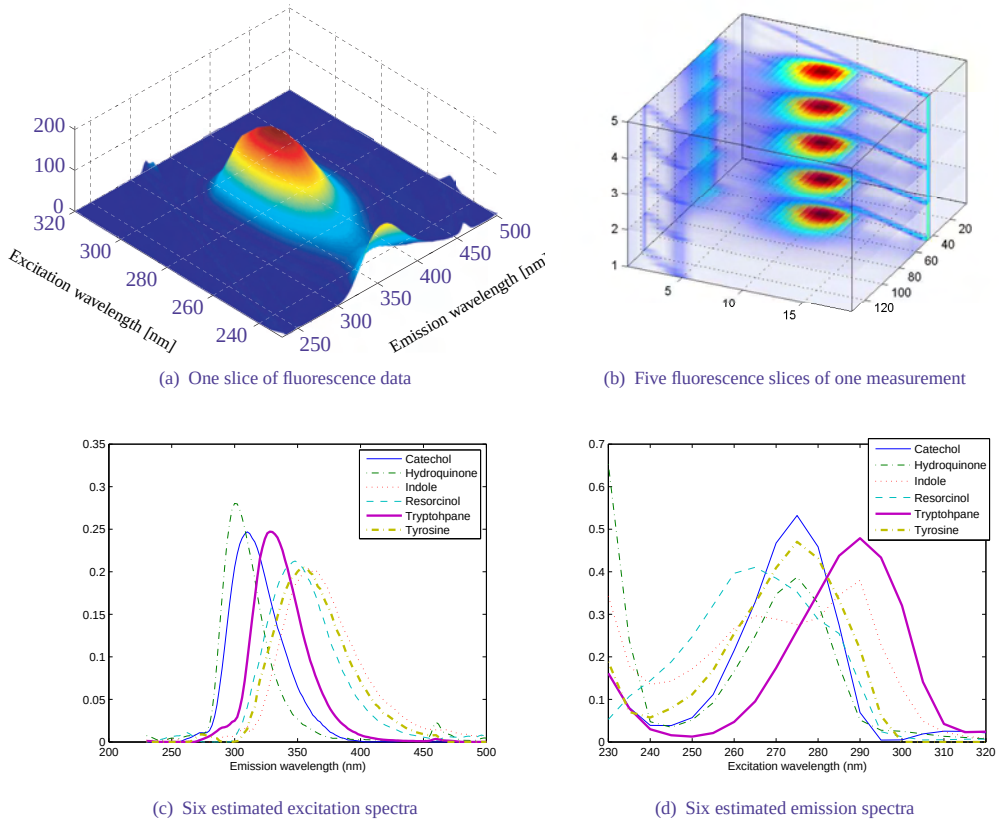
of a wheat grain was scanned with a 365 *nm* laser at a spatial resolution of approximately 1 μm, and the fluorescence signal spectra were measured by a CCD detector in the range of 350 to 670 *nm*. The observed area was discretized into 20×20 pixels, and each observed spectrum was sampled to yield 128 points. Hence, the observations were stored in the matrix $\mathbf{Y} \in \mathbb{R}^{400\times128}$. Applying NMF under the assumption that the number of the constituent compounds is three, they obtained the pure species spectra and the pure species concentration/abundance maps of the corresponding pure species. Each abundance map was obtained by matricization of the corresponding column vector of $\mathbf{A} \in \mathbb{R}^{400\times3}$.

We present an extension of fluorescence spectra using tensor factorization algorithms. The benchmark [91, 12] contains 405 recorded measurements of five replicated fluorescence spectra for a total of six different fluorophores in the dataset: catechol, hydroquinone, indole, resorcinol, tryptohpane and tyrosine. Each spectra was expressed by two factors: Emission (136 wavelengths) and Excitation (19 wavelengths). In total 405×5 Emission-Excitation spectra were recorded. If we vectorize all spectra, the NMF model will be applied to find the six basis components. However, tensor factorization helps to return a much more accurate result. For example, we can factorize the tensor with size of 405 samples $\times$ 136 emission wavelengths $\times$ 19 excitation wavelengths $\times$ 5 replicates into six components. The second and third factors $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ are the Emission and Excitation spectral components, respectively. Figure 8.10(c) and 8.10(d) depict these basis components which correspond to six fluorophores: catechol, hydroquinone, indole, resorcinol, tryptohpane and tyrosine. Another factorization is presented in Figure 8.11 with six basis spectra with size of 136 Emission wavelengths $\times$ 19 Excitation wavelengths. Each 2-D spectrum slice is a composition of the six basis spectra. Figure 8.10(a) is an example of one Emission $\times$ Excitation slice.

8.3.3 Hyperspectral Imaging

Hyperspectral imaging has found many real-life applications [40]. In the mining and oil industries it is mostly used for identifying various minerals or for searching ore or oil fields. In agriculture it is useful for monitoring the development and health control of crops, detection of the chemical composition of plans, and water quality control. Physicists use this technique in electron microscopy, and soldiers for military surveillance.

Hyperspectral imaging remotely maps the object of interest with spectral observations of electromagnetic waves emitted or reflected from the object. Typically, the object of interest is a 2D remote surface from which sunlight is reflected, and a distribution of reflection/absorption rate is reconstructed from the observations. The spectrum of sunlight in a wide range of wavelengths (from ultraviolet to infrared) is measured with a remote array of narrow-band sensors of high spectral resolutions. For example, the Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS) sensors measure spectral signals in the range of $0.4 - 2.45$ μm within 224 contiguous subbands with a spectral resolution of 10*nm*. The spatial resolution depends on the distance from the object of interest to the observation point, and typically, it varies from 4 to 20 meters. This kind of sensor is used in airborne observations; satellite observations (e.g., by NASA's Hyperion sensors) are also commonly-used. The object of interest is discretized, and a spectral characteristic is measured for each pixel, thus the observations are stored in a 3D array. In Figure 8.12(b), we illustrate the La Parguera dataset [94] taken with the Hyperion sensor, atmospheric corrected with the ACORN algorithm and with several bands discarded. The La Parguera region is composed of different types of reefs in shallow and deep water, sea grass (mostly thalassia), mangrove and sand. The true color image of the La Parguera region was collected from the three bands of Red = 634 nm, Green = 545 nm, and Blue = 463 nm (Figure 8.12(a)).



(a) One slice of fluorescence data

(b) Five fluorescence slices of one measurement

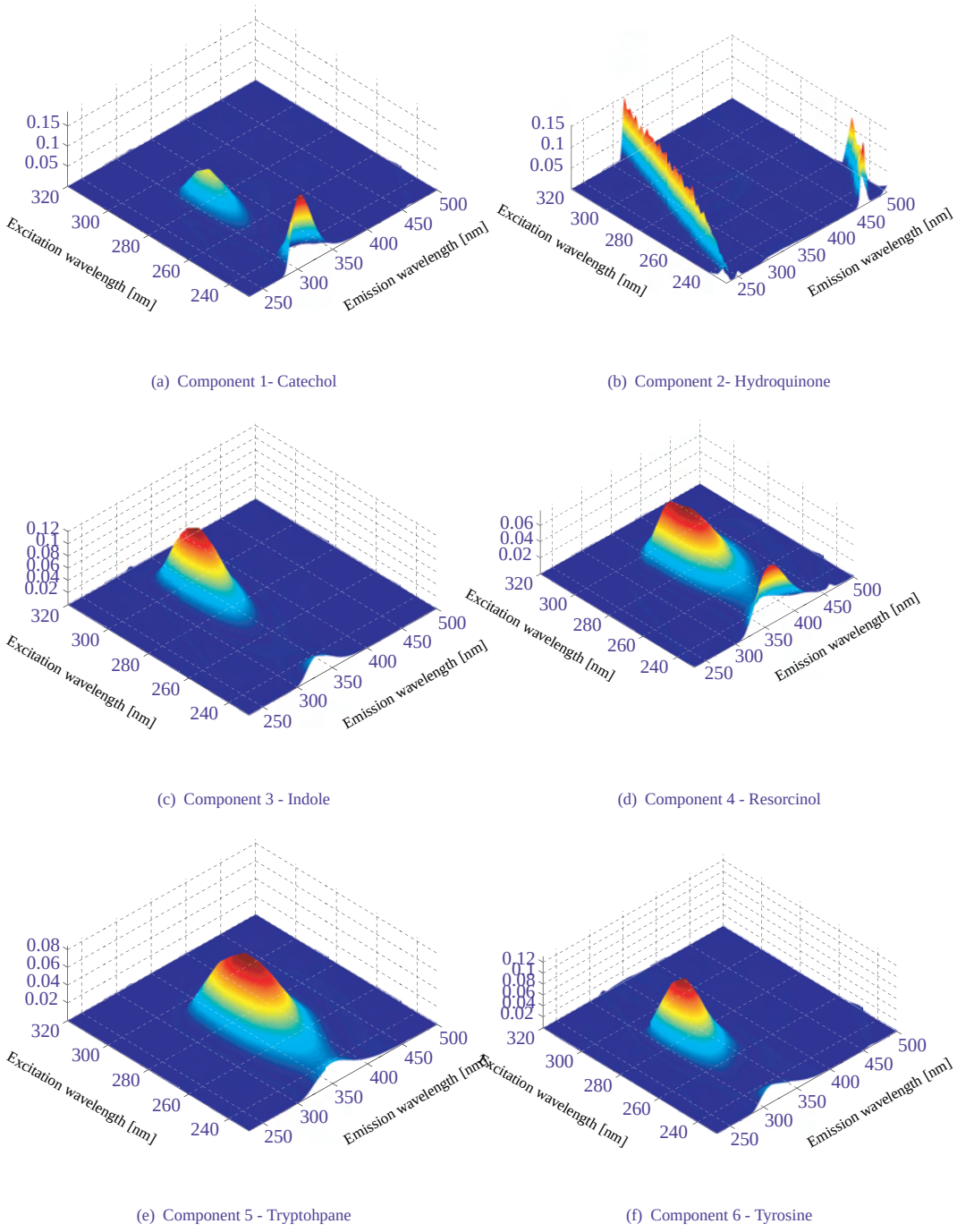
(c) Six estimated excitation spectra

(d) Six estimated emission spectra

Fig. 8.10 Illustration of factorization for fluorescence data of size 405 samples × 136 emission wavelengths × 19 excitation wavelengths × 5 replicates : (a) one spectra slice of the tensor , (b) one 3-D sample with five fluorescence replicates of size 36 emission wavelengths × 19 excitation wavelengths, (c) - (d) estimated excitation and emission spectra of six different fluorophores in the dataset: catechol, hydroquinone, indole, resorcinol, tryptohpane and tyrosine.

Each horizontal slice represents a spatial distribution of a reflection/absorption rate for a given subband. A 3D multi-array is formed from the multiple subband observations. The horizontal slices are divided into pixels, and thus a 3D multi-array of observations is composed from voxels. A plot of the reflection/absorption rate along any vertical line determines the continuous spectrum that identifies the surface material in a given position on the surface.

Unfortunately, the observed spectrum in any position on the surface of interest is practically a superposition of spectra of many underlying materials. This also causes a poor spatial resolution of spectral detectors. A single “pure” material is called the endmember, and the aim of applying NMF to hyperspectral imaging is to extract the spectra of endmembers from the multi-array of mixed spectral observations. Furthermore, having the spectra of endmembers computed, we can estimate the maximum abundance of each endmember in a given position on the surface. A 2D distribution of each corresponding abundance provides complete information for the surveyed area. One should also notice that both abundances and spectra of endmembers are nonnegative



(a) Component 1- Catechol

(b) Component 2- Hydroquinone

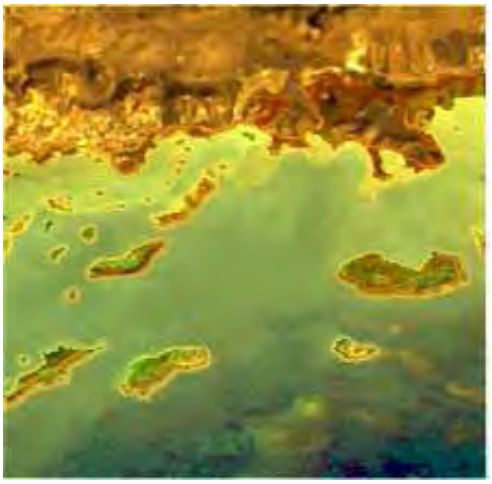
(c) Component 3 - Indole

(d) Component 4 - Resorcinol

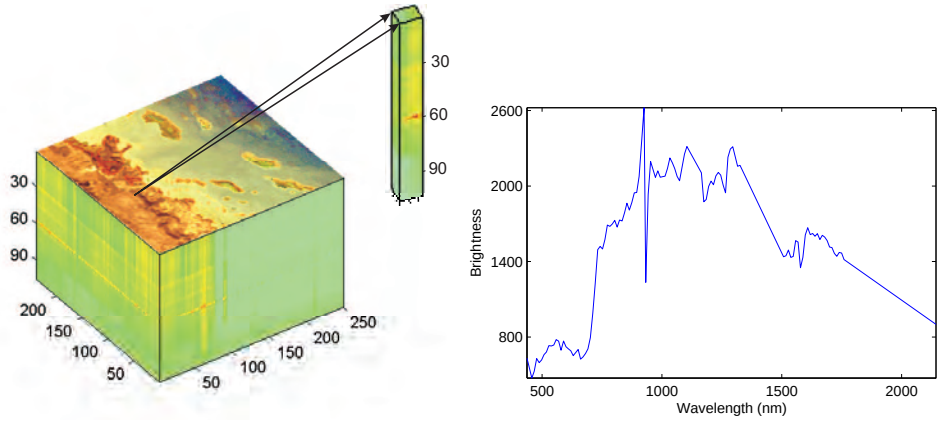
(e) Component 5 - Tryptohpane

(f) Component 6 - Tyrosine

Fig. 8.11 Six spectra components were estimated from the tensor 405 samples × 2584 (emission-excitation) wavelengths × 5 replicates. Slice in Figure 8.10(a) was expressed by addition of these six basis spectra.



(a) True color of the La Parguera image

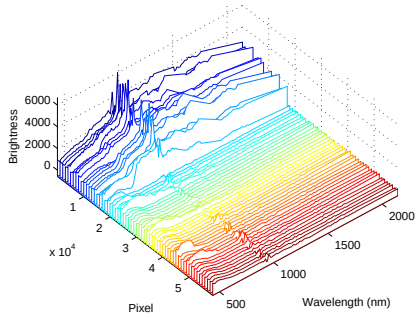


(b) Model of hyperspectra La Parguera image

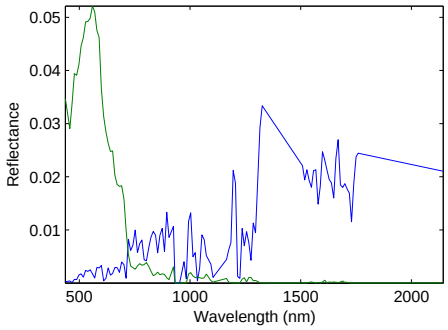
Fig. 8.12 Hyperspectral imagery illustration for the La Parguera hyperspectra image [94], captured with the EO-1 (Hyperion sensor) over La Parguera area Lajas, and Puerto Rico. The dataset processed using ACORN has 106 bands of size 250 × 239 pixels, where multiple subbands form a 3D multi-array. A vertical profile through a single pixel position shows the continuous spectrum assigned to this position. The spectrum identifies the surface material: (a) true color visualization includes bands at Red = 634 nm, Green = 545 nm, and Blue = 463 nm, (b) relative brightness at pixel (45,102) through 106 bands.

curves, and hence, the usage of NMF for this application seems to be reasonable. However, other decomposition methods (e.g., used in ICA) could also be applied [77].

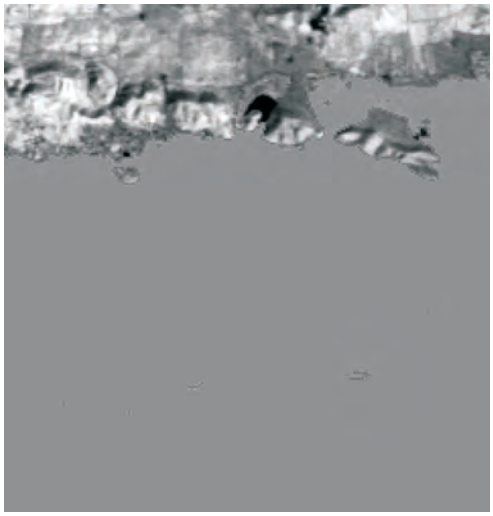
To apply NMF a 3D multi-array should be unfolded to form an observation matrix $\mathbf{Y} \in \mathbb{R}^{I \times T}$ where I is the total number of pixels in one slice, and T is the number of subbands (Figure 8.13(a)). After applying NMF, we obtain the nonnegative matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ of abundances, and the nonnegative matrix $\mathbf{X} \in \mathbb{R}^{J \times T}$ of endmember spectra. The rows in $\mathbf{X}$ correspond to



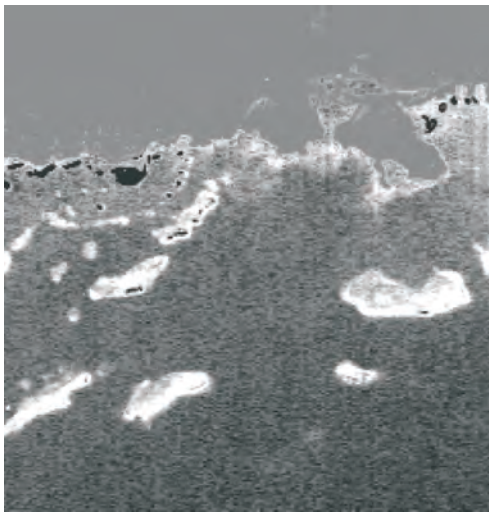
(a) Restructured pixels of 106 bands as NMF data



(b) Two reflectance spectra found by the NMF model



(c) Endmember image 1 - component 1 of NMF



(d) Endmember image 2 - component 2 of NMF

Fig. 8.13 Analysis of hyperspectral images by the NMF model: (a) Pixels was reordered according to wavelengths for analysis, (b) two first reflectance components were estimated by NMF, (c)-(d) two endmember spectra correspond to the reflectance components.

endmembers, and the respective columns in $\mathbf{A}$ refer to the abundances. The spectrum of an endmember is some kind of "fingerprint" or signature of the underlying material and it should be unique and suitable for identification. The abundance vectors are matricized in the reverse way to the used vectorization, and hence each column vector in $\mathbf{A}$ determines a 2D image of the abundance for a given material. Thus, we have J images of distribution of the underlying materials.

For the La Parguera hyperspectra image [94] which has 106 bands of size 250×239 pixels (see Figure 8.13(a)), the mixed region is composed of two major different endmembers which were recovered with NMF and shown in Figure 8.13(b). The corresponding abundance maps are illustrated in Figure 8.13(c) and Figure 8.13(d), where lighter pixels denote higher abundance.

Figure 8.15 presents the results obtained with NMF applied to the 3D ^{31}P CSI database [130]. The plots in Figure 8.15(a) represent the spectra of endmembers, and the abundance maps are illustrated in Figure 8.15(b). As stated in Sajda *et al.* [96], the upper spectrum in Figure 8.15(a) represents muscle tissue whereas the bottom one refers to brain tissue. Consequently, the corresponding abundance maps present distributions of muscular and brain tissues. Indeed, the muscle tissue is distributed near the skull border, which is visible in the top image in Figure 8.15(b), whereas the brain tissue is centered in the interior of the skull.

8.4 APPLICATION OF NMF FOR ANALYZING MICROARRAY DATA

Matrix factorization and decomposition methods have also found many relevant applications in biomedical data processing and analysis. Several works are concerned with application of NMF to gene expression classification [55, 13, 14, 53, 83, 84, 73, 33, 109], mostly in order to classify different types of cancers. Other exemplary applications include muscle identifications in the nervous system [107], classification of PET images [1], and protein fold recognition [80].

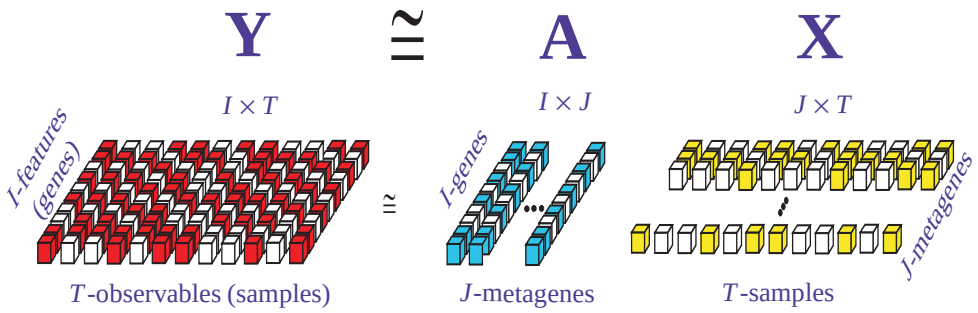
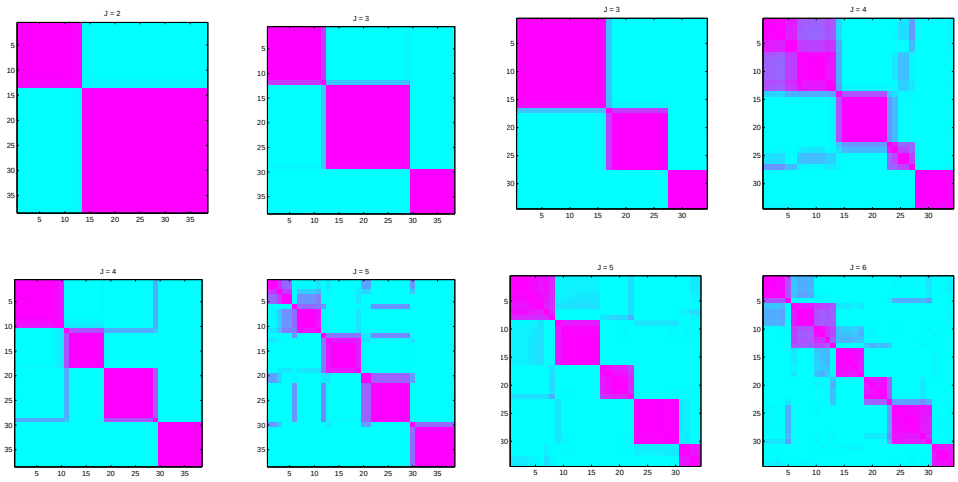


Fig. 8.16 Schematic representation of the NMF model applied to gene-expression matrix $\mathbf{Y}$.

8.4.1 Gene Expression Classification

The aim of applying NMF to the analysis of DNA microarrays is to group genes and experiments according to their similarity in gene expression patterns. The groups of genes that are referred to as *metagenes* (see [13]) capture latent structures in the observed data and may provide biological insight into underlying biological processes and the mechanisms of diseases. Metagenes also provide meaningful information for clustering the genes as well as the related experiments. Nested and partially overlapped clusters can also be identified with the NMF approach. Nested clusters reflect local properties of expression patterns, and overlapping is due to global properties of multiple biological processes (selected genes can participate in many processes). Typically, there are a few metagenes in the observed DNA microarray that may monitor several thousands of genes. Thus, the redundancy in this application is extremely high, which is very profitable for NMF. Furthermore, metagenes and gene expression patterns can often be described by sparse



(a) Leukemia dataset (b) Medulloblastoma dataset

Fig. 8.17 Patch plots of consensus matrices obtained with NMF (a) NMF applied for leukemia dataset, (b) NMF applied for medulloblastoma dataset.

sists of the following samples: 10 classic medulloblastomas, 10 malignant gliomas, 10 rhabdoids, and four normal samples. The goal of using NMF for analyzing the medulloblastoma samples from this dataset is to find genes that are statistically correlated with two basic classes of medulloblastomas: classic and desmoplastic. As reported in [13], only clustering with NMF provides easily interpretable clusters that are shown in Figure 8.17(b). For $J = 5$, one nested cluster is almost entirely related to the samples of the desmoplastic class.

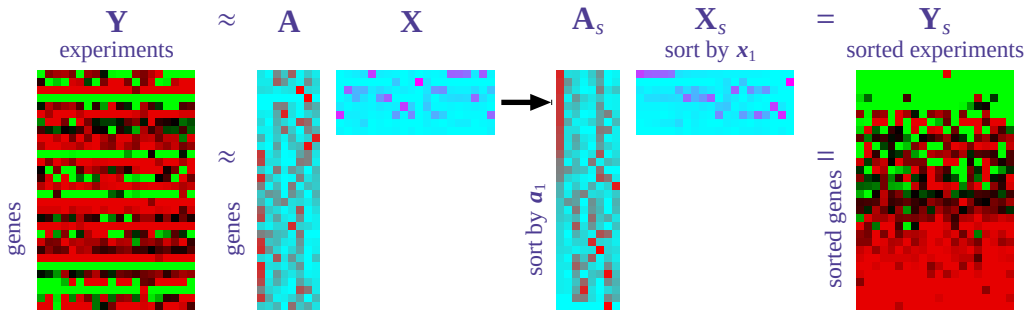
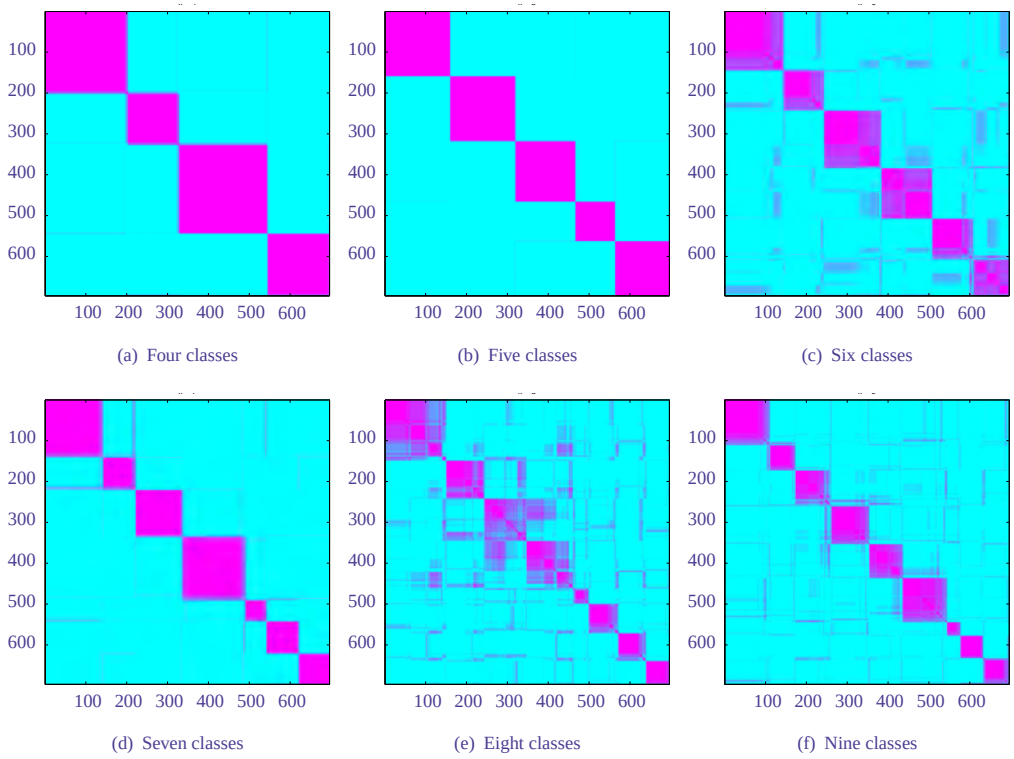


Fig. 8.18 Biclustering using NMF.

Brunet *et al.* [13] successfully applied NMF to the classification of four types of central nervous system embryonal tumors [88]. Moreover, NMF gave much more accurate results than the hierarchical clustering and SOM. The SOM evidently identifies only three classes, merging the malignant glioma and normal samples. Also, the hierarchical clustering misclassifies the



(a) Four classes (b) Five classes (c) Six classes

(d) Seven classes (e) Eight classes (f) Nine classes

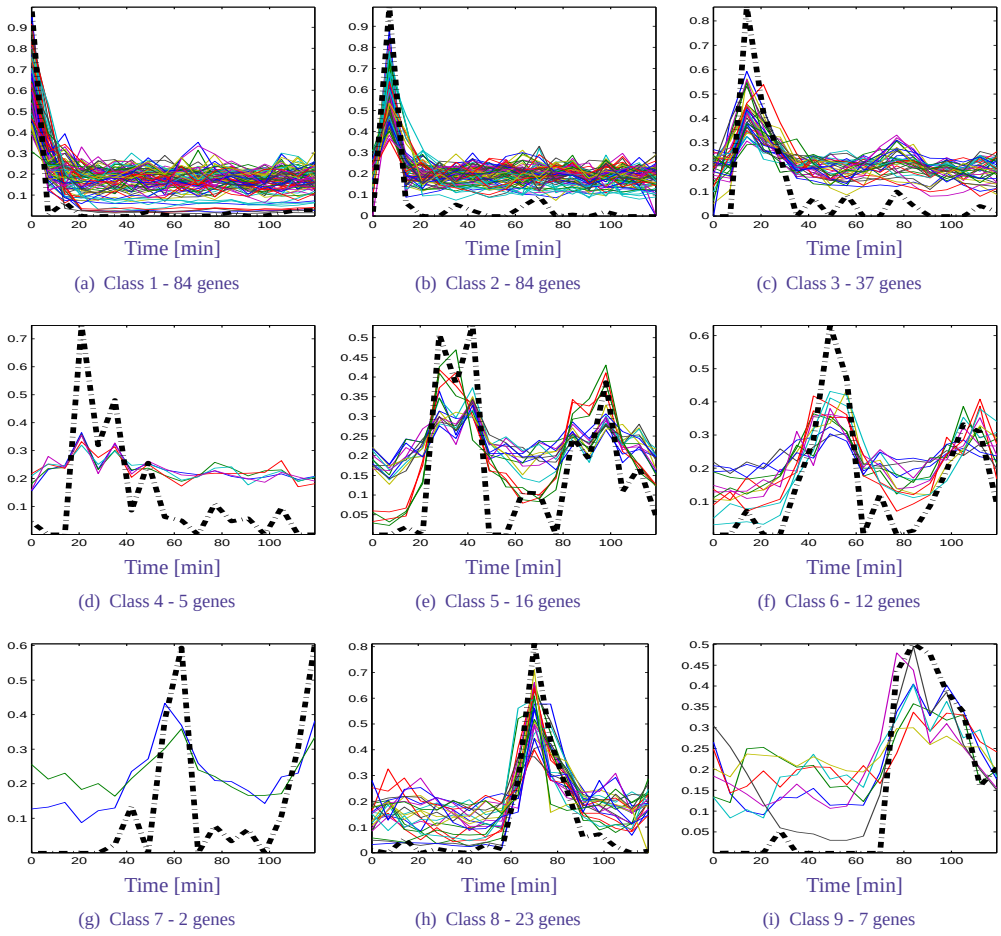
Fig. 8.20 Visualization of consensus matrices for clustering gene expression profiles with 4, 5, . . . , 9 classes from the Alpha-696 dataset. The similarity matrices are measured and reordered from 1000 trials for each case.

to finding such highly correlated genes is to decompose the full dataset ($\mathbf{Y} \in \mathbb{R}_{+}^{6044\times 18}$) but with the given and fixed basis components $\mathbf{X}$. Coefficients in the same column vector of $\mathbf{A}$ reflect the contribution levels of the corresponding basis profiles $\mathbf{X}$ in the analyzed genes $\mathbf{Y}$. Therefore, a group of coefficients with high values in each column of $\mathbf{A}$ will show the set of genes that have a strong correlation with extracted profiles. The procedure for selecting such genes can be summarized as:

- normalize column vectors of $\mathbf{A}$ to unit-length,
- sort all the column vectors of $\mathbf{A}$ in descending order,
- select the highly correlated genes based on their highest coefficients in each column over one specified threshold.

The results of the selected gene profiles for the cases of four and nine component decompositions are shown in Figure 8.21 and 8.22, respectively. The most highly correlated profiles with their gene ID are also listed in Tables 8.5 and 8.6.

The final application will illustrate how to sort gene profiles according to the basis expression profiles. The procedure is performed as follows:



(a) Class 1 - 84 genes (b) Class 2 - 84 genes (c) Class 3 - 37 genes

(d) Class 4 - 5 genes (e) Class 5 - 16 genes (f) Class 6 - 12 genes

(g) Class 7 - 2 genes (h) Class 8 - 23 genes (i) Class 9 - 7 genes

Fig. 8.22 Comparison of nine components extracted by NMF (thick dash lines) with highly correlated genes expression profiles from the Alpha_full dataset. The basis expression components are extracted from the small dataset: Alpha-696. All selected profiles have correlation coefficients $r > 0.8$, and are normalized to unit-length.

(a) Original Alpha-696 (b) Sorted Alpha-696

Fig. 8.23 Sorting the dataset Alpha-696 according to the five extracted basis components. Basis gene expression profiles $\mathbf{X}$ are rearranged in order of appearance of the first major peaks (events), then the columns of the mixing matrix $\mathbf{A}$ are rearranged accordingly, and the coefficients in columns of $\mathbf{A}$ are sorted in descending order. Finally, the genes of the original data are sorted according to both events in $\mathbf{X}$ and weighting in $\mathbf{A}$.