Scalable Low-rank Matrix and Tensor Decomposition on Graphs

THÈSE N^{O} 7958 (2017)

PRÉSENTÉE LE 8 DÉCEMBRE 2017
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE TRAITEMENT DES SIGNAUX 2
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Nauman SHAHID

acceptée sur proposition du jury:

Prof. J.-Ph. Thiran, président du jury
Prof. P. Vandergheynst, directeur de thèse
Prof. Y. Wiaux, rapporteur
Prof. J. A. Fessler, rapporteur
Dr D. Van De Ville, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

To my mom and dad…

Acknowledgements

I wish to express my gratitude to a number of people, without whom this thesis would not have been possible.

First, I would like to thank my supervisor Prof. Pierre Vandergheynst for introducing me to the field of Graph Signal Processing and helping me with all the challenges associated with the field. For inspiring discussions, mentoring and for steering me in the right direction whenever it was needed. For teaching me the value of my work, for encouraging me to be independent by establishing a continuous improvement strategy. He taught me endurance and to keep a positive attitude which was needed for various parts of this work. He continues to provide inspiration and it has been a true honor to be his student over the past four years.

I would like to thank Dr. Xavier Bresson, Dr. Vassilis Kalofolias and Dr. Benjamin Ricaud for helping me to kick-start my PhD and helping me with all the small and big challenges that I encountered in the beginning of my PhD.

I would also like to thank Dr. Nathanael Perraudin and Dr. Gilles Puy for co-authoring a few important papers of my PhD. They helped me expand my knowledge in the field of graph signal processing and Compressed Sensing.

I am also grateful to Faisal Mahmood from Okinawa Institute of Science & Technology in Japan, who introduced me to the field of biomedical signal processing, more specifically, computational tomography, where my work has an impact. Collaboration with him broadened my inter-disciplinary knowledge.

I would also like to thank Keith Smith, a visiting PhD student from The University of Edinburgh. Working with him on the early detection of Alzheimer's disease helped me understand the challenges faced by clinicians and doctors in the real life scenarios.

I would also like to thank Franceso Grassi, an exchange PhD student from Politecnico di Torino. We have worked together for over a year in the field of low-rank tensor decomposition using graphs. Francesco has not only helped me in professional capacity but has also proved to be a great friend.

Finally, I would like to thank Rodrigo Pena, a PhD student from LTS2, who proof-read my thesis and helped me to make it better.

Abstract

In many signal processing, machine learning and computer vision applications, one often has to deal with high dimensional and big datasets such as images, videos, web content, etc. The data can come in various forms, such as univariate or multivariate time series, matrices or high dimensional tensors. The goal of the data mining community is to reveal the hidden linear or non-linear structures in the datasets. Over the past couple of decades matrix factorization, owing to its intrinsic association with dimensionality reduction has been adopted as one of the key methods in this context.

One can either use a single linear subspace to approximate the data (the standard Principal Component Analysis (PCA) approach) or a union of low dimensional subspaces where each data class belongs to a different subspace. In many cases, however, the low dimensional data follows some additional structure. Knowledge of such structure is beneficial, as we can use it to enhance the representativity of our models by adding structured priors. A nowadays standard way to represent pairwise affinity between objects is by using graphs. The introduction of graph-based priors to enhance matrix factorization models has recently brought them back to the highest attention of the data mining community. Representation of a signal on a graph is well motivated by the emerging field of signal processing on graphs, based on notions of spectral graph theory. The underlying assumption is that high-dimensional data samples lie on or close to a smooth low-dimensional manifold. Interestingly, the underlying manifold can be represented by its discrete proxy, i.e. a graph.

A primary limitation of the state-of-the-art low-rank approximation methods is that they do not generalize for the case of non-linear low-rank structures. Furthermore, the standard low-rank extraction methods for many applications, such as low-rank and sparse decomposition, are computationally cumbersome.

We argue, that for many machine learning and signal processing applications involving big data, an approximate low-rank recovery suffices. Thus, in this thesis we present solutions to the above two limitations by presenting a new framework for scalable but approximate low-rank extraction which exploits the hidden structure in the data using the notion of graphs.

First, we present a novel signal model, called 'Multilinear low-rank tensors on graphs (MLRTG)' which states that a tensor can be encoded as a multilinear combination of the low-frequency graph eigenvectors, where the graphs are constructed along the various modes of the tensor. Since the graph eigenvectors have the interpretation of *non-linear* embedding of a dataset on the low-dimensional manifold, we propose a method called 'Graph Multilinear SVD (GMLSVD)' to recover PCA based linear subspaces from these eigenvectors. Finally, we propose a plethora of highly scalable matrix and tensor based approximation methods for low-rank extraction which implicitly or explicitly make use of the GMLSVD framework. The core idea is to replace the expensive iterative SVD operations by updating the linear subspaces from the fixed non-linear ones via low-cost operations. We present applications in low-rank and sparse decomposition and clustering of the low-rank features to evaluate all the proposed methods. Our detailed theoretical analysis shows that the approximation error of the proposed framework depends on the spectral properties of the graph Laplacians.

Key words: Principal Component Analysis, graphs, low-rank and sparse decomposition, clustering, low-rank tensors

Résumé

Dans de nombreuses applications de traitement du signal, de l'apprentissage automatique et de la vision par ordinateur, il faut souvent traiter des jeux de données de grande dimension et de grande taille, tel que des images, des vidéos, du contenu Web, etc. Les données peuvent se présenter sous diverses formes, telles que de séries temporelles univariées ou multivariées, des matrices, ou des tenseurs à haute dimension. Le but de la communauté d'exploration des données est de révéler les structures cachées, linéaires ou non-linéaires, dans les jeux de données. Au cours des dernières décennies, la factorisation de matrices, en raison de son association intrinsèque avec la réduction de dimension, a été adoptée comme l'une des méthodes clés dans ce contexte.

On peut soit utiliser un seul sous-espace linéaire pour approcher les données (l'approche standard de l'analyse en composantes principales (PCA)), soit une union de sous-espaces à faible dimension où chaque classe de données appartient à un sous-espace différent. Dans de nombreux cas, cependant, les données à faible dimension ont une structure supplémentaire. La connaissance de cette structure est bénéfique, car nous pouvons l'utiliser pour améliorer la représentativité de nos modèles en ajoutant des structures à priori. Un mode standard de nos jours pour représenter l'affinité par paires entre les objets est en utilisant des graphes. L'introduction des informations à priori fondés sur des graphes pour améliorer les modèles the factorisation matricielle les a récemment ramenés à la plus haute attention de la communauté d'exploration des données. La représentation d'un signal sur un graphe est bien motivée par le champ émergent du traitement du signal sur des graphes, qui est basé sur des notions de la théorie spectrale des graphes. L'hypothèse sous-jacente est que les échantillons de données à haute dimension se situent sur ou près d'une variété lisse à faible dimension. Fait intéressant, la variété sous-jacente peut être représenté de façon discrète par un graphe.

Une limitation primaire des méthodes actuelles d'approximation de rang faible est qu'elles ne se généralisent pas pour le cas de structures non-linéaires de rang faible. En outre, les méthodes standard d'extraction de rang faible, telles que les décompositions à rang faible et parcimonieuse, ont une lourde complexité algorithmique pour de nombreuses applications.

Nous soutenons que pour de nombreuses applications d'apprentissage automatique et de traitement de signal impliquant de grandes données, une récupération approximative de rang faible suffit. Ainsi, dans cette thèse, nous présentons des solutions aux deux limitations ci-dessus en présentant un nouveau cadre pour l'extraction approximative mais scalable de rang faible, qui exploite la structure cachée dans les données en utilisant la notion de graphes.

Tout d'abord, nous présentons un nouveau modèle de signal, appelé «tenseurs de rang faible multilinéaire sur les graphes (MLRTG)», qui indique qu'un tenseur peut être codé en tant que combinaison multi-linéaire des vecteurs propres de basse fréquence des graphes, où les graphes sont construits le long des différents modes du tenseur. Étant donné que les vecteurs propres du graphe ont l'interprétation de plongement non-linéaire d'un ensemble de données sur une variété à faible dimension, nous proposons une méthode appelée «Graph Multilinear SVD (GMLSVD)» pour récupérer des sous-espaces linéaires basés sur PCA à partir de ces vecteurs propres. Enfin, nous proposons une pléthore de problèmes inverses basés sur les matrices et les tenseurs, avec haute capacité à monter en charge, pour l'extraction de rang faible qui utilisent implicitement ou explicitement le cadre GMLSVD. L'idée principale est de remplacer les coûteuses opérations itératives SVD en mettant à jour les sous-espaces linéaires, à partir des espaces non-linéaires fixes, par des opérations peu coûteuses. Nous présentons des applications sur des décompositions de rang faible et parcimonieuse, et sur regroupement des features de rang faible pour évaluer toutes les méthodes proposées. Notre analyse théorique détaillée montre que l'erreur d'approximation du cadre proposé dépend des propriétés spectrales des Laplacians des graphes.

Mots-clés : Analyse en composantes principales, graphes, décompositions de rang faible et parcimonieuse, regroupement, tenseurs de rang faible.

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ADMM	Alternating Direction Method of Multipliers
ALS	Alternating Least Squares
BCI	Brain Computer Interface
CP	Candecomp
EEG	Electroencephalogram
FMRI	Functional Magnetic Resonance Imaging
FRPCAG	Fast Robust Principal Component Analysis on Graphs
GCT	Graph Core Tensor
GFT	Graph Fourier Transform
GLPCA	Graph Laplacian Principal Component Analysis
GLRR	Graph regularized Low-Rank Representation
GLRSR	Graph regularized Low-Rank and Sparse Representation
GMLSVD	Graph Multilinear Singular Value Decomposition
GNMF	Graph regularized Non-negative Matrix Factorization
GTD	Graph Tucker Decomposition
HOSVD	Higher Order Singular Value Decomposition
HSI	Hyper Spectral Image / Imaging
LE	Laplacian Eigenmaps
LLP	Locality Preserving Projections
LRR	Low-Rank Representation
MLRTG	Multilinear Low-rank Tensor on Graphs
MLSVD	Multilinear Singular Value Decomposition
MMF	Manifold Matrix Factorization
MRI	Magnetic Resonance Imaging
NCuts	Normalized Cuts
NMF	Non-negative Matrix Factorization
PCA	Principal Component Analysis
RGLPCA	Robust Graph Laplacian Principal Component Analysis
RPCA	Robust Principal Component Analysis
RPCAG	Robust Principal Component Analysis on Graphs
SVD	Singular Value Decomposition
TRPCA	Tensor Robust Principal Component Analysis
TRPCAG	Tensor Robust Principal Component Analysis on Graphs
TSVD	Truncated Singular Value Decomposition

Contents

Acknowledgements **i**

Abstract (English/Français/Deutsch) **iii**

Abbreviations **vii**

List of figures **xiii**

List of tables **xix**

I Introduction, Background & Motivation 1

1 Introduction & Motivation **3**

- 1.1 Low-rank Feature Extraction & Dimensionality Reduction 3
  - 1.1.1 Applications of Low-rank Approximation 4
  - 1.1.2 Low-rank Matrix & Tensors 5
- 1.2 Linear Low-rank Matrix Approximation Methods 6
  - 1.2.1 Standard Principal Component Analysis (PCA) 6
  - 1.2.2 Low-rank & Sparse Decomposition: Robust PCA 8
- 1.3 Low-rank Tensor Approximation Methods 8
  - 1.3.1 Multilinear SVD 8
  - 1.3.2 Alternating Least Squares for Candecomp Decomposition 9
  - 1.3.3 Tensor Robust PCA 9
- 1.4 From Linear Subspaces to Manifolds 10
- 1.5 Graphs 12
  - 1.5.1 Graph nomenclature 12
  - 1.5.2 What does a graph portray? 13
  - 1.5.3 Graph construction: A k_{nn}-nearest neighbor graph 13
  - 1.5.4 Graph embedding: Non-linear dimensionality reduction 14
- 1.6 Motivation: Non-linear Low-rank Structure & Scalability 14
- 1.7 Goals of this thesis 17
- 1.8 Proposed Methodology & Thesis Contributions 18
  - 1.8.1 Part I: Introduction, Background & Motivation 19
  - 1.8.2 Part II: Proposed Low-rank Decomposition Model 19
  - 1.8.3 Part III: Proposed Approximation Methods & Applications 20

2 Related Work & Terminology 23
2.1 Terminology & Notation . . . 23
2.1.1 Vectors & Matrices . . . 23
2.1.2 Tensors . . . 24
2.1.3 Vector, Matrix & Tensor Operators . . . 24
2.1.4 Low-rank Matrix . . . 25
2.1.5 Low-rank Tensor . . . 25
2.1.6 Graphs . . . 25
2.2 Graph Regularized PCA: A Review . . . 27
2.2.1 Factorized Models . . . 28
2.2.2 Non-factorized Models . . . 30
2.2.3 Multiple subspace models: Low-rank Representation (LRR) . . . 30
2.3 Graph Regularized Tensor PCA . . . 31
2.3.1 Regularization on all the factors . . . 31
2.3.2 Regularization on one of the factors . . . 32
2.3.3 Other Methods . . . 32
2.4 Scalable matrix and tensor methods . . . 33
2.5 Conclusion . . . 33

II Proposed Low-rank Decomposition Model 35

3 Multilinear Low-Rank Tensors on Graphs 37
3.1 Properties of Low-rank Signal Models . . . 37
3.1.1 Matrix based model: PCA & RPCA . . . 37
3.1.2 Tensor based models: MLSVD & TRPCA . . . 39
3.2 Proposed Low-rank Model: Multilinear Low-rank Tensors on Graphs (MLRTG) . . . 39
3.3 Analysis of MLRTG Framework . . . 41
3.3.1 Assumption: The eigen-gap of a cartesian product graph . . . 41
3.3.2 Eigen-gap implies ‘Approximate Graph Stationarity’ & ‘k-clusterable’ nature of MLRTG . . . 43
3.3.3 Properties of Multilinear Low-rank Tensors on Graphs (MLRTG) . . . 47
3.4 Approximate MLRTG . . . 48
3.5 Examples of Exact and Approximate MLRTG . . . 50
3.5.1 Methods to generate MLRTG . . . 50
3.5.2 From exact to approximate MLRTG . . . 53
3.5.3 Real tensors . . . 54
3.6 Understanding MLRTG . . . 56
3.6.1 The graph of features provides a basis for data . . . 56
3.6.2 The graph of samples provides embedding for data . . . 58
3.6.3 Multi-dimensional embedding in a space of product of manifolds . . . 59
3.6.4 Graph Core Tensor: The Joint Graph Fourier Transform . . . 59
3.7 Conclusion . . . 60

4 From graph basis to PCA basis: Graph Multilinear SVD 61
4.1 Graph Multilinear SVD (GMLSVD) . . . 62
4.1.1 Derivation from Low-frequency power concentration property of MLRTG . . . 63
4.1.2 Derivation from MLSVD algorithm . . . 65
4.1.3 Algorithm for GMLSVD . . . 67

4.1.4 Complexity . . . 67
4.1.5 Properties of GMLSVD . . . 67
4.2 Examples . . . 68
4.3 Choice of graph multilinear rank . . . 70
4.4 GMLSVD application: Tensor Compression . . . 72
4.5 Conclusion . . . 72

III Proposed Approximation Methods & Applications 75

5 Fast Robust PCA on Graphs (FRPCAG) 77
5.1 Introduction to Fast Robust PCA on Graphs . . . 78
5.2 Optimization Problem . . . 79
5.3 Connections with the state-of-the-art . . . 80
5.4 Algorithm . . . 81
5.5 Computational Complexity . . . 81
5.6 Theoretical Analysis of FRPCAG . . . 83
5.6.1 Main Theorem . . . 84
5.6.2 Why FRPCAG gives a low-rank solution? Implications of the Theoretical Analysis . 85
5.6.3 Conclusion . . . 88
5.7 Studying the model: Working Examples of FRPCAG . . . 88
5.7.1 Low-rank recovery, singular value penalization and subspace alignment . . . 88
5.7.2 Clustering, singular value penalization and subspace alignment . . . 89
5.7.3 Bi-clustering Example on ORL dataset . . . 92
5.8 Experiments . . . 93
5.8.1 Clustering . . . 96
5.8.2 Low-rank & Sparse Decomposition . . . 99
5.8.3 Computational Time . . . 99
5.9 Generalized Fast Robust PCA on Graphs (GFRPCAG) . . . 101
5.9.1 Re-visiting the Approximation Error of FRPCAG . . . 101
5.9.2 Design of deterministic graph filters ‘g’ to improve approximation error . . . 103
5.10 Conclusions . . . 105

6 Compressive PCA on Graphs 107
6.1 Contributions . . . 108
6.2 A glimpse of Compressive PCA on Graphs . . . 109
6.3 RIP for MLRTG . . . 109
6.4 Compressed Low-Rank Matrix . . . 111
6.4.1 Graphs for Compressed data . . . 112
6.4.2 FRPCAG on the Compressed Data . . . 112
6.5 Decoders for low-rank recovery . . . 113
6.5.1 Ideal Decoder . . . 113
6.5.2 Alternate Decoder . . . 114
6.5.3 Approximate Decoder . . . 115
6.5.4 Other Approximate Decoders . . . 120
6.6 Decoder for clustering . . . 121
6.7 Complexity, Memory & Convergence . . . 123
6.8 Experimental Results . . . 127
6.8.1 Clustering . . . 127

6.8.2 Low-rank and sparse decomposition . . . 134
6.9 Conclusion . . . 135

7 Tensor Low-rank Decomposition on Graphs 137
7.1 The General Framework . . . 139
7.1.1 Step 1: Weighted Nuclear Norm Problem . . . 140
7.1.2 Step 2: Computationally feasible weighted nuclear norm problem . . . 141
7.1.3 Step 3: Extension for tensors . . . 143
7.2 Proposed Approximation Problems . . . 143
7.2.1 Graph Core Tensor Pursuit . . . 144
7.2.2 Tensor Robust PCA on Graphs . . . 145
7.3 Properties of our framework . . . 146
7.4 Theoretical Analysis . . . 147
7.5 Experimental Results . . . 150
7.5.1 Experiments on GMLSVD . . . 152
7.5.2 Low-rank & sparse decomposition . . . 155
7.5.3 Robust Subspace Recovery . . . 159
7.5.4 The effect of parameters on GMLSVD . . . 159
7.6 Conclusion . . . 161

8 Discussion & Conclusions 163
8.1 This thesis in a nutshell . . . 163
8.2 Comparison of approximation problems . . . 166
8.3 Discussion on the choice of appropriate method . . . 167
8.4 Short-comings & Future Directions . . . 168
8.5 Further Applications . . . 170
8.5.1 Matrix Completion . . . 170
8.5.2 Tensor non-linear embedding & clustering . . . 171
8.5.3 Tensor Segmentation . . . 172
8.5.4 Low-Rank Recovery from Random Projections . . . 173
8.5.5 Dynamic MRI Reconstructions . . . 175
8.5.6 Discriminative Feature Learning for Tensors . . . 176
8.5.7 Low-Rank Tensors for Convolutional Neural Networks . . . 177

A Appendices 179
A.1 Proof of Lemma 3.3.1 . . . 179
A.2 Proof of Lemma 3.3.2 . . . 180
A.3 Proof of Theorem 5.6.1 . . . 180
A.4 Proof of theorem 6.3.1 . . . 181
A.5 Proof of Theorem 6.5.1 . . . 183
A.6 Proof of Theorem 6.5.2 . . . 183
A.7 Proof of Theorem 6.5.3 . . . 185
A.8 Proof of Lemma 6.5.1 . . . 186
A.8.1 Proof of Theorem 7.4.1 . . . 186

Bibliography 198

Curriculum Vitae 199

List of Figures

1.1 Three different classes of points in a 3-dimensional linear subspace. The left plot shows that the principal directions $\boldsymbol{U}_r$ are robust to Gaussian noise $\pm\boldsymbol{\eta}$, i.e, they follow the variance of the data points. The right plot shows that in the case of sparse noise (a few strong outliers here), the principal directions $\boldsymbol{U}_r$ deviate significantly from the data trend. 7

1.2 A few images from two different classes of the AT&T dataset and their arbitrary mapping on a 3-dimensional linear and non-linear surface. Clearly, a non-linear surface is a better fit to the low-dimensional point. 11

1.3 This figure shows the recovery of underlying clean low-rank structure of the noisy 2D or 3D manifolds using the Robust PCA framework [1]. RPCA, which is a linear subspace recovery algorithm fails in all the scenario as it tries to reduce the whole data to a point at the center of the manifold in an attempt to exactly recover the manifold. The last column shows the approximate desired behavior for each of the cases. 15

1.4 A summary of the main goals, applications and evaluation schemes used in this thesis. . 17

1.5 A summary of the contributions and organization of this thesis. 22

3.1 The proposed low-rank model for tensors and matrices. 40

3.2 A summary of our analysis on MLRTG. 42

3.3 Examples of GSC for 1) joint approximate graph stationarity (left), 2) MLRTG (middle) and 3) approximate MLRTG (right). The difference between the three concepts is illustrated in terms of the concentration of energy in the specific regions of the GSC. 49

3.4 A summary of the variation of properties from exact to approximate MLRTG. 52

3.5 Examples of approximate multilinear low-rank tensors on graphs. Each row shows the full tensor, the SVD of each of its flattened modes, graph spectral covariance (eq. (3.4)) of mode 1 $\boldsymbol{\Gamma}^1$ and the low-rank power concentration $\hat{s}(\boldsymbol{\Gamma})$ (eq. (3.6)) for each of the modes, from left to right. Clearly, the concentration of energy in the top corner of GSC shows that each of these tensors is multilinear low-rank on graphs. 55

3.6 Studying the number 3 of USPS. Left: Covariance eigenvectors associated with the 16 highest eigenvalues. Right: Laplacian eigenvectors associated to the 16 smallest non-zero eigenvalues. Because of stationarity, Laplacian eigenvectors are similar to the covariance eigenvectors. Bottom: $\boldsymbol{\Gamma} = \boldsymbol{P}^{1\top}\boldsymbol{C}\boldsymbol{P}^1$ in dB. Note the diagonal shape of the matrix implying that $\boldsymbol{P}^1$ is aligned with the eigenvectors of $\boldsymbol{C}$. 57

3.7 Studying the full USPS dataset. Left: Covariance eigenvectors associated with the 16 highest eigenvalues. Right: Laplacian eigenvectors associated to the 16 smallest non-zero eigenvalues. Because of stationarity, Laplacian eigenvectors are similar to the covariance eigenvectors. Bottom: $\boldsymbol{\Gamma} = \boldsymbol{P}^{1\top}\boldsymbol{\Omega}\boldsymbol{P}^1$ in dB. Note the diagonal shape of the matrix implying that $\boldsymbol{P}^1$ is aligned with the eigenvectors of $\boldsymbol{\Omega}$. 58

4.1 GSC of the mode 1 (time) of the FMRI tensor. 63

4.2 A comparison of the first two singular vectors of the exact MLRTG $\boldsymbol{Y}$ and those estimated by the GMLSVD method. The left most column shows the first two singular vectors in the left (upper) and right (lower) subspaces obtained via SVD. The red line corresponds to the 1st singular vector and the blue to the 2nd. The middle plot shows the first four eigenvectors (black, red, blue green) of the row and column graphs. The rightmost column shows the resulting left (upper) and right (lower) singular vectors obtained by linear combination of the graph eigenvectors, i.e., by the GMLSVD method. 68

4.3 A comparison of the 1st singular vectors obtained via GSVD and SVD for various 2D and 3D real face tensors. In each row, the leftmost plot shows an example image from the database, the middle plot shows the 1st singular vector of mode 1 obtained via GSVD and the right plot shows the 1st singular vector of mode 1 obtained via SVD. It can be clearly seen that the singular vectors determined by GSVD and SVD are equivalent. 69

4.4 Visualization of the singular vector and singular values alignment for each of the three modes of hyperspectral face tensor. 70

4.5 Increase in the singular vector and singular values alignment and decrease in the normalized reconstruction error for mode 1 of an artificial tensor w.r.t increasing graph multilinear rank k. 71

4.6 Qualitative and quantitative results for the compression of various hyperspectral tensors from the Stanford database. 73

5.1 Main idea of Fast Robust PCA on Graphs. 79

5.2 A study of the error of approximation $\|\boldsymbol{Y}_0 - \boldsymbol{X}\|_F^2 / \|\boldsymbol{Y}_0\|_F^2$ for a data generated from graph eigenvectors. We plot the low-rank matrix $\boldsymbol{X}$, the alignment order $s(\boldsymbol{\Gamma}^r)$, $s(\boldsymbol{\Gamma}^c)$ and $\hat{s}(\boldsymbol{\Gamma}^c)$, $\hat{s}(\boldsymbol{\Gamma}^r)$ ($k = 10$), the singular values $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$, the coherency between $(\boldsymbol{U}, \boldsymbol{U}_0)$, $(\boldsymbol{V}, \boldsymbol{V}_0)$ 90

5.3 The matrices $\boldsymbol{V}\boldsymbol{V}^\top$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$, $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ and the corresponding clustering errors obtained for different values of the weights on the two graph regularization terms for 1000 samples of MNIST dataset (digits 0 and 1). The block diagonal structure of $\boldsymbol{V}\boldsymbol{V}^\top$ becomes more clear by increasing γ_c and γ_r with a thresholding of the singular values in $\boldsymbol{\Sigma}$. Furthermore, the sparse structures of $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$ and $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ towards the rightmost corners show that the number of left and right singular vectors (weighted by singular values) which align with the eigenvectors of the Laplacians $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ go on decreasing with increasing γ_c and γ_r. This shows that the two graphs help in attaining a low-rank structure with a low clustering error. 91

5.4 A comparison of singular values of the low-rank matrix obtained via our model, RPCA and RPCAG. The experiments were performed on the ORL dataset with different levels of block occlusions. The parameters corresponding to the minimum validation clustering error for each of the model were used. 92

5.5 The clustering of features for ORL dataset. The top five rows show the five different feature clusters across five different faces. The last row shows the actual face which is a sum of all the feature clusters. 93

5.6 The checkerboard pattern of the low-rank representation of ORL dataset, obtained by FRPCAG. 94

5.7 A comparison of clustering error of our model with various dimensionality reduction models. The image data sets include: 1) ORL 2) CMU PIE 3) COIL20 and 4) YALE. The compared models are: k-means, NCuts, LE [2], PCA, GLPCA [3], NMF [4], GNMF [5], MMF [6], 9) RPCA [1], FRPCAG(A), FRPCAG(B) and FRPCAG(C). Two types of corruptions are introduced in the data: 1) Block occlusions and 2) Random missing values. NCut, LE, GLPCA, MMF and GNMF are evaluated with a good sample graph $\mathbb{G}^c$. FRPCAG(A) corresponds to our model evaluated with a good sample and a good feature graph, FRPCAG(B) to a good sample graph and a noisy feature graph and FRPCAG(C) to a noisy sample and feature graph. NMF and GNMF require non-negative data so they were not evaluated for the USPS and MFeat datasets because they are negative as well. MFeat is a non-image dataset so it is not evaluated with block occlusions. For MNIST both $\mathbb{G}^c$ and $\mathbb{G}^r$ are noisy. 97

5.8 Principal Components of 1000 samples of digits 0 and 1 of the MNIST dataset in 2D space. For this experiment all the digits were corrupted randomly with 15% missing pixels. Our proposed model (lower right) attains a good separation between the digits which is comparable and even better than other state-of-the-art dimensionality reduction models. 98

5.9 Static background separation from three videos. Each row shows the actual frame (left), recovered static low-rank background using RPCA, RPCAG and our proposed model. The first row corresponds to the video of a restaurant food counter, the second row to the shopping mall lobby and the third to an airport lobby. In all the three videos the moving people belong to the sparse component. Thus, our model is able to accurately separate the static portion from the three frames as good as the RPCAG. Our model converged in less than 2 minutes for each of the three videos, whereas RPCA and RPCAG converged in more than 45 minutes. 100

5.10 The manifold shrinking effect of a 2D manifold corrupted with noise. The first row shows the recovery of manifold with increasing values of FRPCAG (with one graph only) and the second row shows the singular values. Clearly, the 3^{rd} singular value does not become zero with higher penalization which degrades the lower singular values and the manifold begins to shrink. 102

5.11 The alignment of the singular vectors with the singular vectors of clean data and the corresponding shrinking of the singular values with the increasing graph regularization parameter. The first row shows the original and noisy data, where the blue vectors show the first two singular vectors of the clean data and the red ones for the noisy data. The second row shows how the singular vectors (green) learned with the FRPCAG eq. 5.1 are being aligned with the singular vectors of the clean data. All the singular vectors are scaled by their singular values. Too much regularization shrinks the singular vectors. . . . 103

5.12 Example of functions. Here $b = 0.4$. 105

6.1 Box plots for clustering error over 10 random sampling runs of CPCA (10,1) for the full USPS dataset (256 × 10000) with increasing levels of Gaussian noise. For each run CPCA is evaluated for the full parameter grid $\gamma_r^{'}, \gamma_c^{'} \in (0, 30)$ and the minimum clustering error is considered. A slight increase in the average clustering error with Gaussian noise shows that CPCA is quite robust to sampling and noise. 132

6.2 Static background separation from videos using different PCA based models. The first row corresponds to a frame from the video of a shopping mall lobby, the second row to a restaurant food counter and the third row to an airport lobby. The leftmost plot in each row shows the actual frame, the other 5 show the recovered low-rank using RPCA, RPCAG, FRPCAG and CPCA with two different uniform downsampling schemes. 134

6.3 A quality comparison of various low-rank decoders discussed in this work. 134

6.4 A comparison of the quality of low-rank frames for the shopping mall video (1st row of Fig. 6.2) extracted using the approximate decoder (6.18) for different downsampling factors on the pixels and frames. It is obvious that the quality of low-rank remains intact even with higher downsampling factors. 136

7.1 Performance comparison of GSVD and GMLSVD with SVD and MLSVD for 2D artificial (1st row), 3D artificial (2nd row) and 4D real BCI, FMRI, Hyperspectral and EEG tensors (3rd to 6th rows) under different SNR scenarios. Clearly GSVD / GMLSVD outperforms SVD / MLSVD in terms of computation time and error for big and noisy datasets. 153

7.2 Singular values (left plot) of GSVD, SVD and clean data and singular vector alignment of GSVD and clean data (right plot) for a 2D matrix corrupted with Gaussian noise (SNR 5dB). Clearly GSVD eliminates the effect of noise from lower singular values and aligns the first few singular vectors appropriately. 154

7.3 A visualization of the clean, noisy and GMLSVD recovered BCI and hyperspectral face tensors along with a study of the singular values and subspace vectors of various modes. First row shows the BCI tensor and second row shows the singular values recovered for this tensor for various modes. Clearly, the singular values of the recovered tensor (black) are well aligned with those of the clean tensor (blue). Note, how GMLSVD eliminates the effect of noise from the lower singular values. The rightmost plot shows the alignment between the singular vectors of the recovered and clean tensors. This alignment is calculated as $|V_1^\top U_1|$, where V_1 are the first mode singular vectors obtained via GMLSVD and U_1 are the first mode singular vectors of the clean tensor. Again, note that the first few singular vectors are very well aligned. Third and fourth rows show the same type of results for hyperspectral face tensor. 156

7.4 Performance analysis of TRPCAG for 2D artificial (1st row) and 3D artificial (2nd row) with varying levels of sparse noise, 2D airport lobby (3rd row) and 2D shopping mall (4th row) video tensor. The artificial tensors have a size 100 and rank 10 along each mode. A core tensor of size 30 along each mode is used for the artificial tensors and 100×50 for the video tensors. The leftmost plots in 1st two rows show the ℓ_2 reconstruction errors, the middle plots show the ℓ_2 reconstruction error for 30 singular values and the right plots show the subspace angles for the 1st five subspace vectors. For the 2D real video datasets, the rightmost frame shows the result obtained via TRPCAG. The computation time for different methods is written on the top of frames. Clearly TRPCAG performs quite well while significantly reducing the computation time. 157

7.5 Singular values of GSVD, SVD, FRPCAG, RPCA, TRPCAG and clean data for a 2D matrix which is corrupted uniformly (10%) with sparse noise of standard deviation 0.1. Clearly TRPCAG eliminates the effect of noise from lower singular values. In addition note that under sparse noise, the 1st singular value deviates significantly. This effect is also eliminated via TRPCAG. 158

7.6 TRPCAG performance for recovering the low-rank point clouds from the sparse noise. Actual point cloud (left), noisy point cloud (middle), recovered via TRPCAG (right). 158

7.7 Low-rank recovery for a 3D video of dimension $1920 \times 1080 \times 500$ and size 1.5GB via TRPCAG. Using a core size of $100 \times 100 \times 50$, TRPCAG converged in less than 3 minutes. . 159

7.8 Robust recovery of subspace structures via TRPCAG. The leftmost plots show a clean and sparsely corrupted sample image from various datasets. For each of the datasets, other plots in the 1st row show the 1st singular vector of the clean data and those recovered by various low-rank recovery methods, SVD, FRPCAG, RPCA, RPCAG and TRPCAG and the 2nd row shows the alignment of the 1st 20 singular vecors recovered by these methods with those of the clean tensor. 160
7.9 The effect of regularization parameter γ, number of nearest neighbors k_{nn} and the power q in $f_\mu(\mathbf{\Lambda}_k)$ (the weights of nuclear norm) on the performance of GMLSVD. The experiments are performed a 100 × 100 matrix of rank 10 with Gaussian noise to attain a SNR of 5dB. 161

8.1 This thesis in a nutshell. 165
8.2 Initial results for our matrix completion approach of eq. 8.1. 170
8.3 A result of tensor clustering algorithm on the COIL20 dataset. 172
8.4 Segmentation result on the hyperspectral face tensor from Stanford database. 173
8.5 Segmentation result on the Geneva hyperspectral tensor. 174
8.6 The reconstruction of a 64 × 64 Modified Shepp-Logan phantom from 20% projections using eq.(8.2). 175

List of Tables

3.1 Properties of exact and approximate artificial and real MLRTG. 53

3.2 Example artificial and real exact and approximate MLRTG. 53

5.1 Details of the datasets used for clustering experiments in this work. 94

5.2 Range of parameter values for each of the models considered in this work. r is the rank or dimension of subspace, λ is the weight associated with the sparse term for Robust PCA framework [1] and γ is the parameter associated with the graph regularization term. . . . 96

5.3 Computation times (in seconds) for graphs $\mathbb{G}^c$, $\mathbb{G}^r$, FRPCAG, RPCAG, RPCA and the number of iterations to converge for different datasets. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. ∞ indicates that the algorithm did not converge in 4 hours. 99

6.1 Summary of CPCA and its computational complexity for a dataset $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$. Throughout we assume that $k_{nn}, r, k, \rho_r, \rho_c, p \ll n$. 123

6.2 A summary and computational complexities of all the decoders proposed in this work. The Lancoz method used here is presented in [7]. 125

6.3 Computational complexity of all the models considered in this work 126

6.4 Details of the datasets used for clustering experiments in this work. 127

6.5 Clustering error of USPS datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue. NMF and GNMF require non-negative data so they are not evaluated for USPS because the features of the USPS dataset are also negative. 129

6.6 Clustering error of ORL datasets for different models. The best results per column are highlighted in bold and the 2nd best in blue. 130

6.7 Clustering error of CMU PIE and YALE datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue. 131

6.8 Clustering error and computational times of FRPCAG and CPCA on MNIST large dataset (784 × 70,000). 132

6.9 Computation times (in seconds) for graphs G_r, G_c, FRPCAG, CPCA, RPCAG, RPCA and the number of iterations to converge for different datasets. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. An ∞ indicates that the algorithm did not converge in 4 hours. 132

6.10 Preservation of the rank of the datasets in the compressed low-rank $\tilde{\boldsymbol{X}}$ determined by solving FRPCAG (6.4). 133

6.11 Variation of clustering error of CPCA with different uniform downsampling schemes / factors across rows and columns of the USPS dataset (256 × 10,000). 133

6.12 Computational time in seconds of RPCA, RPCAG, FRPCAG and CPCA for low-rank recovery of different videos in Fig. 6.2. 135

Introduction, Background & Motivation

Part I

1 Introduction & Motivation

In the modern era of data explosion, many problems in signal and image processing, machine learning and pattern recognition require dealing with very high dimensional datasets, such as images, videos and web content. The data mining community often strives to reveal natural associations or hidden structures in the data. This task can be important for many different reasons. For example, many machine learning algorithms involve training complex models on datasets which are very high dimensional. When the number of samples is not enough, the classifiers can easily overfit in the high dimensional space. Thus, it is important to extract useful low-dimensional features to feed a classifier. Similar problem arises in signal processing, where one might want to perform the analysis of high dimensional images. In such a scenario, it is always important to search for hidden structures in the dataset, which could lead to a reduction in the dimension or useful feature extraction from the dataset.

We first provide a concise introduction to a specific type of feature extraction process, known as low-rank feature extraction. Then, we argue, with some simple examples that this process falls under the broad class of *linear* dimensionality reduction. Finally, we discuss some short-comings of the state-of-the-art low-rank approximation methods and build a motivation for the work done in the context of this thesis. More specifically, we argue that the state-of-the-art low-rank feature extraction methods cannot recover non-linear low-rank features in a scalable manner. In fact, extracting such features would involve exploiting pairwise relationship between different features and samples of the data matrix which can reveal the underlying geometry of the low-dimensional space. This thesis, as explained in more detail in Section 1.7, provides a framework - a signal model and associated low-rank recovery problems, based on the notion of 'graphs', which can recover low-rank features for a wide variety of datasets in a structured and scalable manner.

1.1 Low-rank Feature Extraction & Dimensionality Reduction

Consider a set of gray-scale images of the same object captured under fixed lighting conditions with a moving camera, or a set of hand-written digits with small rotations. Given that the image has p pixels, each such data sample is represented by a vector in $\mathbb{R}^p$. Since each image is just a small perturbation of the other images, there are many common features in the entire set of images. Simply put, most of the data variance can be explained only with a few common features among the set of images. Thus, the intrinsic dimensionality of the space of all images of the same object captured with small perturbations is much lower than p. One could then use a dimensionality reduction or a compression framework to analyze this dataset. Alternatively, when the data is noisy, the above problem of analyzing the set of images can also be modeled as determining the closest noiseless approximation which can capture maximum variance of the dataset.

Due to many common features among the images, it is possible to model each image as a linear combination of a few basic templates. These templates serve as ‘dictionary’ elements, which can be combined to approximate the entire dataset. As the number of such possible templates is far less than the dimension of the dataset, the approximation is known as a low-rank approximation.

1.1.1 Applications of Low-rank Approximation

Low-rank approximation spans a broad variety of applications. Before diving into the core of mathematical details, we present a glimpse of a few applications here:

Surveillance: Static low-rank: Consider a real world surveillance application where a camera is installed in an airport lobby and it collects frames with with a fixed time interval. As the structure of this lobby does not change over time, one would like to extract the moving people or objects from the static lobby. Such an application falls under the low-rank extraction framework.

Surveillance: Dynamic low-rank: Now consider the same camera installed outside in the open air where the lightning is changing over time. Assume there are short periods of sunlight interleaved by cloudy periods and even rain. As compared to the lobby example where all these conditions were constant, this is a more challenging environment. However, the overall problem can still be seen as extracting dynamic people and or rain from the static background over different intervals of the video. Thus, this application falls under the low-rank approximation framework as well.

Multiclass low-rank: As another example, consider a big database of faces of several people collected from a social media platform. Assume further that the faces are corrupted by different types of noise and errors. Furthermore, the database consists of different facial expressions. A more reasonable assumption for such a dataset would be to group the images based on the people using some notion of pairwise similarity and then perform a low-rank approximation to extract the clean faces from noise and errors.

Topic extraction: Now consider a set of text messages collected from a social network about different topics. One would like to analyze the most common topics in the entire set of messages. A more interesting analysis would be to identify the communities of people in the social network among which a particular topic is more popular. Again, the messages are highly corrupted by noise due to the use of slang and unstructured sentences. In such a case one definitely needs to use the notion of similarity between the people in the network to identify the low-rank component, i.e., most commonly used words and topics and as well as analyze the community structure.

Space-time low-rank: Consider a set of MRI scans (2D or 3D), collected from the brain of a subject, which vary slowly over time. Since the brain responds to stimuli, it exhibits a dynamic activity over time. However, the underlying brain structure does not change over time. One would like to separate the clean low-rank components from dynamic activity which might correspond to some kind of abnormal condition. Furthermore, the acquired scans might be corrupted with noise due to limitations of device or the movement of patient, etc. This is a typical example of a space-time real world dataset which has gained significant attention of the research community. For such a dataset, the notion of functional connectivity between different regions of the brain plays an important role in addition to the structural connectivity. Moreover, the acquired images can be related in time. Thus, it is important to consider the space-time relationship while extracting the low-rank representation.

Space-time-frequency low-rank: Consider Brain Computer Interface application, where a set of time series is collected from electrodes on an EEG cap, mounted over the head of a subject. The EEG signals have a higher time resolution, therefore, it is common to perform a time-frequency analysis of these signals to obtain an information rich dataset. Thus the entire dataset has space-time-frequency features. As EEG is well-known for its highly dynamic and noisy nature, one would like to extract the clean signal from the noise. Assuming that EEG signals correspond to a specific set of brain activities, it is reasonable to argue that this dataset is low-rank. Therefore, one would like to separate low-rank component from noise.

Point-cloud denoising: Point clouds, both static and dynamic, span a plethora of applications in the field of computer vision. However, they are often corrupted by noise and errors. Assuming that a dynamic 2D or 3D point-cloud is just a small set of variations of the pose, point-cloud denoising can be done via low-rank approximation.

1.1.2 Low-rank Matrix & Tensors

So far, we have discussed 'dimensionality reduction ' and 'low-rank approximation' as two alternatives for modeling a set of images with many common features. Now we begin to formalize our discussion and show that the latter approach, i.e, low-rank approximation falls under the broad category of the former, i.e, dimensionality reduction. The key to connecting the two approaches is matrix factorization. Over the past couple of decades matrix factorization has been adopted as one of the key methods in the context of dimensionality reduction. Given a data matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$ with n p-dimensional data vectors, the matrix factorization can be stated as determining factors $\boldsymbol{U} \in \mathbb{R}^{p \times r}$ and $\boldsymbol{Q} \in \mathbb{R}^{r \times n}$ such that $\boldsymbol{Y} \approx \boldsymbol{U}\boldsymbol{Q}^\top$ under different constraints on $\boldsymbol{U}$ and $\boldsymbol{Q}$.

Low-rank Matrix

While, the state-of-the-art methods have proposed several constraints, such as non-negativity, sparsity, joint non-negativity and sparsity, etc. on $\boldsymbol{U}$ and $\boldsymbol{Q}$, our goal in this thesis is to deal with the special case where $\boldsymbol{U}_r \in \mathbb{R}^{p \times r}$ and $\boldsymbol{Q}_r \in \mathbb{R}^{n \times r}$ are thin rectangular matrices, i.e, $r \ll p < n$. Let $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{Q}_r^\top$, then such an approximation of $\boldsymbol{Y}$ is known as a low-rank approximation. Hence, low-rank approximation is equivalent to dimension reduction under the above mentioned constraints on $\boldsymbol{U}, \boldsymbol{Q}$. It is common to call $\boldsymbol{U}_r$ and $\boldsymbol{Q}_r$ as 'factors' of the low-rank matrix $\boldsymbol{X}$.

Low-rank Tensor

Since the beginning, we have only considered the notion of a dataset in the context of a matrix. However, a matrix is not always the most desirable and flexible arrangement of a dataset. In the modern era of data deluge, one might be tempted to arrange multi-dimensional data in the form of high dimensional entities, instead of 2D matrices. Such high dimensional entities are known as *tensors*. We represent tensors with bold calligraphic letters $\boldsymbol{\mathcal{Y}}$. Consider the example of a set of EEG signals collected from the scalp of a few subjects using a 128 electrode apparatus, over different trials. It is straight-forward to note that it is crucial to store and analyze this dataset in the form of a 4D tensor, where channels, time, trials and subjects constitute different dimensions. One cannot come up with simple ways to store such data in matrices without compromising the dependencies between various dimensions.

Similarly to matrices, tensors can also be low-rank. Several of the low-rank datasets, we mentioned earlier can be arranged in the form of tensors. Consider the static and dynamic surveillance video example, where we have access to frames collected over time. This dataset, naturally arises in the form of a 3D tensor, where the three dimensions correspond to the x and y spatial dimensions and the time axis. Similarly, the space-time MRI low-rank example can also be treated as a 3D or 4D MRI tensor.

Unfortunately, the rank of a tensor is not unique, therefore we use the notion of multilinear rank [8]. For a tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, its μ^{th} matricization / flattening $\boldsymbol{Y}^{\mu}$ is a re-arrangement into a matrix such that $\boldsymbol{Y}^1 \in \mathbb{R}^{n_1 \times n_2 n_3}$. Simply put, a tensor is multilinear low-rank, if each of the matrices $\boldsymbol{Y}^{\mu}$ formed by flattening its various modes is low-rank. Thus, a multilinear low-rank approximation $\mathcal{X}$ of a 3D tensor $\mathcal{Y}$ is given as:

$$\operatorname{vec}(\mathcal{X}) = (\boldsymbol{U}^1_{r_1} \otimes \boldsymbol{U}^2_{r_2} \otimes \boldsymbol{U}^3_{r_3}) \operatorname{vec}(\mathcal{S}),$$

where $\mathcal{S} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is called the *core tensor*, $\otimes$ denotes the Kronecker product, $\operatorname{vec}(\cdot)$ denotes the vectorization operation and $\boldsymbol{U}^{\mu}_{r_\mu}$ are the factors or singular vectors, similar to those obtained by PCA. The tensor $\mathcal{X}$ is said to be low-rank with a multilinear rank (r_1, r_2, r_3).

1.2 Linear Low-rank Matrix Approximation Methods

For a data matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, the signal model for low-rank approximation can be stated as follows:

$$\boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ is the noise and $\boldsymbol{X}^*$ is the low-rank approximation which follows $\boldsymbol{X}^* = \boldsymbol{U}_r \boldsymbol{Q}_r^\top$, where $\boldsymbol{U}_r \in \mathbb{R}^{p \times r}$ and $\boldsymbol{Q}_r \in \mathbb{R}^{n \times r}$ and $r \ll p < n$. Depending on the different types of noise $\boldsymbol{\eta}$, we can classify the low-rank approximation techniques, discussed in this thesis into the following two types:

1. Standard Principal Component Analysis (PCA), when $\boldsymbol{\eta}$ is Gaussian.
2. Low-rank and sparse decomposition via Robust PCA, when $\boldsymbol{\eta}$ is sparse or Laplacian distributed.

Throughout this thesis, we will be mainly concerned with the above two types of low-rank approximation techniques in the context of various applications. Therefore, we provide a detailed explanation of these techniques below.

1.2.1 Standard Principal Component Analysis (PCA)

Standard PCA [9] is a classical matrix factorization technique to determine the low-rank approximation of a matrix $\boldsymbol{Y}$, in the presence of Gaussian noise. The goal is to decompose a matrix $\boldsymbol{Y}$ into a product of factors $\boldsymbol{U}_r, \boldsymbol{Q}_r$ by solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{U}_r, \boldsymbol{Q}_r} \|\boldsymbol{Y} - \boldsymbol{U}_r \boldsymbol{Q}_r^\top\|_F^2 \\ &\quad \text{s.t } \boldsymbol{U}_r \in \mathbb{R}^{p \times r}, \quad \boldsymbol{Q}_r \in \mathbb{R}^{n \times r}, \quad \boldsymbol{U}_r^\top \boldsymbol{U}_r = \boldsymbol{I}_r \end{aligned} \tag{1.1}$$

Note that in addition to the matrices $\boldsymbol{U}_r, \boldsymbol{Q}_r$ being thin, there exists an additional orthonormality constraint on the columns of $\boldsymbol{U}_r$. Assuming that each of the columns of the matrix $\boldsymbol{Y}$ consists of a p-dimensional data sample, the factor $\boldsymbol{U}_r \in \mathbb{R}^{p\times r}$ serves as a dictionary or 'principal directions' of the matrix $\boldsymbol{Y}$. Thus, it is possible to linearly combine the r atoms of this orthonormal dictionary to approximate every p-dimensional data sample in $\boldsymbol{Y}$. The linear combination is given by the rows of the matrix $\boldsymbol{Q}_r$, commonly known as 'principal components' in the standard PCA literature. Note that each row of matrix $\boldsymbol{Q}_r$ is an r-dimensional embedding of the columns of $\boldsymbol{Y}$ in an r-dimensional space, characterized by $\boldsymbol{U}_r$. Finally, the number of columns of $\boldsymbol{U}_r$ and $\boldsymbol{Q}_r$, i.e, r is known as the rank of the low-rank approximation of the data matrix $\boldsymbol{Y}$.

Although, the optimization problem in eq. 1.1 is non-convex due to the presence of the orthonormality constraint, its solution is given by a truncated singular value decomposition operation (TSVD). More specifically, let the full SVD of $\boldsymbol{Y}$ be:

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ is a diagonal matrix of singular values which are sorted in the non-increasing order, $\boldsymbol{U}$ the left singular vectors and $\boldsymbol{V}$, the right singular vectors. Then, the rank r solution to eq. 1.1 is given by truncating $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ to the first r columns, i.e,

$$\boldsymbol{X} = \boldsymbol{U}_r\boldsymbol{Q}_r^\top = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top, \quad \text{where} \quad \boldsymbol{Q}_r = \boldsymbol{\Sigma}_r\boldsymbol{V}_r$$

A primary limitation of PCA is that it can tolerate only Gaussian noise. However, for many real world applications, such as those mentioned earlier, the noise is sparse. For example, the dynamic people in the surveillance video example and sparse dynamic activity in the MRI and EEG examples. Consider the arbitrary example in Fig. 1.1 where we show 3 different classes of low-dimensional points in a 3-dimensional linear subspace. The left plot shows that the principal directions $\boldsymbol{U}_r$ are robust to Gaussian noise $\pm\boldsymbol{\eta}$, i.e, they follow the variance of the data points. The right plot shows that in the case of sparse noise (a few strong outliers here), the principal directions $\boldsymbol{U}_r$ deviate significantly from the data trend.

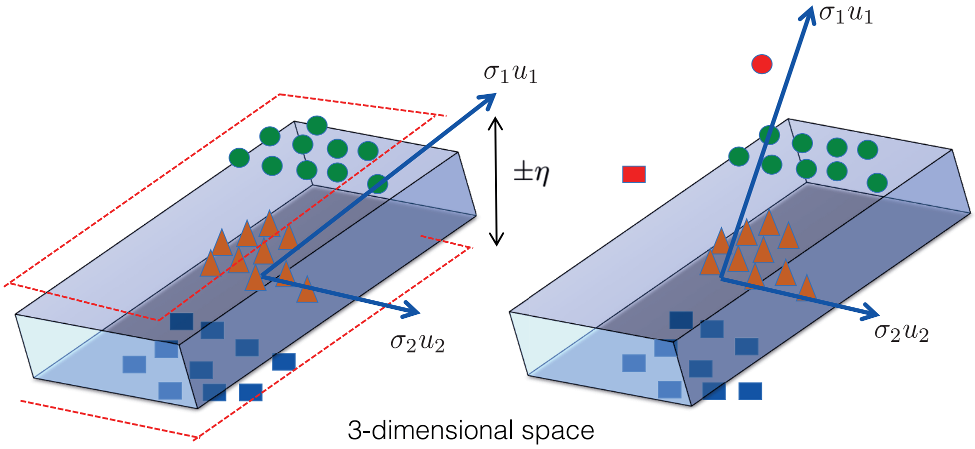


Figure 1.1 – Three different classes of points in a 3-dimensional linear subspace. The left plot shows that the principal directions $\boldsymbol{U}_r$ are robust to Gaussian noise $\pm\boldsymbol{\eta}$, i.e, they follow the variance of the data points. The right plot shows that in the case of sparse noise (a few strong outliers here), the principal directions $\boldsymbol{U}_r$ deviate significantly from the data trend.

1.2.2 Low-rank & Sparse Decomposition: Robust PCA

Applications which involve the separation of sparse and low-rank components of a data matrix, such as the video surveillance, MRI and EEG examples discussed earlier, fall under the category of 'Low-rank and sparse decomposition'. Candes et. al [1] proposed Robust PCA (RPCA), which can be used to model robustness to sparse noise:

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* + \lambda \|\boldsymbol{S}\|_1 \\ &\text{s.t. } \boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{S}, \end{aligned} \tag{1.2}$$

where $\boldsymbol{X}$ is the low-rank matrix, modeled by the nuclear norm operator $\|\cdot\|_*$ and $\boldsymbol{S}$ models the sparse noise. The nuclear norm, a sum of singular values of a matrix, is a convex relaxation to the rank operator, similarly to ℓ_1 norm being a convex relaxation to the ℓ_0 norm. The solution of the nuclear norm operator is given by the singular value soft-thresholding operation and thus requires an explicit computation of the SVD of $\boldsymbol{X}$. Since, SVD is an inherent step in the above optimization problem, one can see this as a problem similar to the standard PCA, except that the data fidelity term involves an ℓ_1 norm.

In contrast to the standard PCA, the rank r of $\boldsymbol{X}$ is not explicitly specified. Instead, it is controlled by the regularization parameter λ. Smaller the λ, bigger is the rank of $\boldsymbol{X}$. Due to the presence of an ℓ_1 norm, which is non-differentiable, it is not possible to compute a closed form solution of eq. 1.2. Thus, one starts with a matrix $\boldsymbol{X}$ of full rank, performs an SVD, thresholds the singular values of $\boldsymbol{X}$, performs the minimization corresponding to the ℓ_1 norm and iterates until convergence. Hence, the rank of $\boldsymbol{X}$ gradually reduces in every iteration of the algorithm. Furthermore, note that unlike the TSVD based solution for PCA, the higher singular values are also thresholded in this algorithm. Finally, the algorithm requires the computation of a full SVD of $\boldsymbol{X}$ is in every iteration. For a matrix of size $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, this costs $\mathcal{O}(p^2 n)$, where $p < n$, which can be computationally cumbersome when both n and p are large.

1.3 Low-rank Tensor Approximation Methods

For a tensor $\mathcal{Y}$, the signal model for low-rank approximation can be stated as:

$$\mathcal{Y} = \mathcal{X}^* + \boldsymbol{\eta}.$$

Again, depending on $\boldsymbol{\eta}$ to be either Gaussian or sparse noise, we will consider two types of methods.

1.3.1 Multilinear SVD

Multilinear SVD (MLSVD) [10], [11] is an alternative to standard PCA for the case of tensors. In standard MLSVD, one aims to decompose a tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ into factors $\boldsymbol{U}^{\mu}_{r_\mu} \in \mathbb{R}^{n_\mu \times r_\mu}$ which are linked by a core $\mathcal{S} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$. This decomposition is known as **Tucker Decomposition** [12], [13], [14], [15], and can be mathematically stated as follows:

$$\operatorname{vec}(\mathcal{X}) = (\boldsymbol{U}^1_{r_1} \otimes \boldsymbol{U}^2_{r_2} \otimes \boldsymbol{U}^3_{r_3}) \operatorname{vec}(\mathcal{S}), \tag{1.3}$$

where $\mathscr{X}$ is the low-rank tensor approximation of $\mathscr{Y}$. This can be achieved by solving the Alternating Least Squares (ALS) problem [11] which iteratively computes the SVD of every mode of $\mathscr{Y}$ until the fit $\| \operatorname{vec}(\mathscr{Y}) - (\boldsymbol{U}^1_{r_1} \otimes \boldsymbol{U}^2_{r_2} \otimes \boldsymbol{U}^3_{r_3}) \operatorname{vec}(\mathscr{S}) \|_2^2$ stops to improve.

$$\begin{aligned} &\min_{\boldsymbol{U}^\mu, \forall\, \mu, \mathscr{S}} \left\| \operatorname{vec}(\mathscr{Y}) - (\boldsymbol{U}^1_{r_1} \otimes \boldsymbol{U}^2_{r_2} \otimes \boldsymbol{U}^3_{r_3}) \operatorname{vec}(\mathscr{S}) \right\|_2^2 \\ &\quad \text{s.t } {\boldsymbol{U}^{\mu}}^{\top}_{r_\mu} \boldsymbol{U}^\mu_{r_\mu} = \boldsymbol{I}_{r_\mu}, \quad \forall\, \mu \end{aligned} \tag{1.4}$$

Just like the standard PCA, the factors $\boldsymbol{U}^\mu_{r_\mu}$ are orthornormal, however, the core tensor $\mathscr{S}$ is not necessarily diagonal like the singular value matrix $\boldsymbol{\Sigma}$. For the case of a 2D tensor, MLSVD reduces to standard PCA for a matrix.

1.3.2 Alternating Least Squares for Candecomp Decomposition

Candecomp (CP) decomposition [11] is the alternative decomposition for low-rank tensors. For a tensor $\mathscr{Y} \in \mathbb{R}^{n \times n \times n}$, it is defined as:

$$\mathscr{X} = \sum_{i=1}^{r} \boldsymbol{U}^1(i) \odot \boldsymbol{U}^2(i) \odot \boldsymbol{U}^3(i), \tag{1.5}$$

where $\mathscr{X}$ is the low-rank tensor approximation of $\mathscr{Y}$ and $\odot$ denotes the outer-product. Note that in contrast to the Tucker decomposition, CP approximates a tensor as a sum of rank-1 tensors. However, it is limited in its modeling power, as it associates a unique rank r to reach of the modes of the tensor. CP decomposition can be attained by solving the following Alternating Least Squares (ALS) problem,

$$\min_{\boldsymbol{U}^\mu, \forall\, \mu} \left\| \mathscr{Y} - \sum_{i=1}^{r} \boldsymbol{U}^1(i) \odot \boldsymbol{U}^2(i) \odot \boldsymbol{U}^3(i) \right\|_F^2. \tag{1.6}$$

The above optimization problem is non-convex. Furthermore, CP decomposition is known not to provide a tight relaxation for the rank of a tensor [11]. Throughout this thesis, we will work with Tucker decomposition.

1.3.3 Tensor Robust PCA

Similarly, to the case of matrices, one needs a method for low-rank and sparse decomposition of tensors, when $\boldsymbol{\eta}$ is sparse. Tensor Robust PCA (TRPCA) [16], [17], [18] based methods have been used for this purpose which can be stated as:

$$\min_{\mathscr{X}} \left\| \operatorname{vec}(\mathscr{Y}) - \operatorname{vec}(\mathscr{X}) \right\|_1 + \gamma \sum_{\mu} \| \boldsymbol{X}^\mu \|_*. \tag{1.7}$$

Like the RPCA algorithm [1] on a matrix , such methods need the notion of nuclear norm on a tensor. Tensor nuclear norm $\sum_\mu \|\boldsymbol{X}^\mu\|_*$ [19], which minimizes the sum of nuclear norms on each of the matrices $\boldsymbol{X}^\mu$ formed by flattening the various modes of a tensor is used for this purpose. For a 3D tensor $\mathscr{Y} \in \mathbb{R}^{n \times n \times n}$, this costs $\mathscr{O}(n^4)$ per iteration, which is computationally prohibitive, even for n as small as

100.

1.4 From Linear Subspaces to Manifolds

Now that we have reviewed low-rank approximation methods for matrices and tensors, we are ready to switch gears and discuss the short-comings of these methods. While, tensor based methods are quite an important subject of this thesis, throughout this chapter, in order to motivate the utility of this work in a simple manner, we limit our discussion to the matrix methods only.

Both, the standard PCA and Robust PCA fit a single subspace $\boldsymbol{U}_r$ that characterizes the low-dimensional data by capturing maximum variance. This is a good enough assumption about the slowly varying images. Thus, a single subspace model, like PCA might work very well for the applications of static low-rank separation from dynamic foreground.

Having described the single subspace models, we now move on to more complicated applications. For the case of dynamic low-rank surveillance video example, it might not be possible to extract the low-rank component via a single linear subspace model. One would need something slightly more complex than this simple assumption. Another example where the linear subspace assumption might fail is the case of a movie where the scenes change over time. A more reasonable assumption would be to target a dynamically changing low-rank representation over short bursts of frames. This would require some sort of grouping between the frames in time. The same holds for our multiclass low-rank face database example. A linear subspace model would simply fail to extract a low-rank representation from such a database of faces because it would not take into account the subgroups of the data samples.

In order to make our discussion more concrete, we consider the example of AT&T dataset of faces of various people with some pose variations. A few example images from this dataset are shown in Fig. 1.2. Each image consists of approximately 10,000 pixels, hence lies in a 10000 dimensional space. Note that each of the classes of images undergoes a pose variation, thus, a single linear subspace model would be unable to capture these variations. Furthermore, the images belong to different classes and each class possesses a different structure from the other. It might be reasonable to assume here that a highly non-linear subspace (manifold) can capture such variations. Fig. 1.2 shows an arbitrary mapping of the three different classes of AT&T dataset on a 3D space. The left plot shows the mapping on a linear subspace and the right plot fits an arbitrary non-linear surface to the data points. Clearly, the non-linear surface is a better fit for such points than the linear one. Such a non-linear surface is known as a manifold. Formally, a manifold is a low-dimensional non-linear subspace, embedded in the high dimensional space. Different parts of the manifold would model different classes of images and the non-linearity within each class would take into account the variational effect of the images. We call such a set of images or a dataset to follow a *non-linear low-rank structure*.

In fact, many real world datasets have a linear or non-linear low-rank structure in a very low-dimensional space. Unfortunately, one often has very little or no information about the geometry of the space, resulting in a highly under-determined recovery problem. Under certain circumstances, as already discussed, state-of-the-art algorithms like PCA and RPCA provide an exact recovery for linear low-rank structures but at the expense of highly inscalable algorithms which use nuclear norm. However, the case of non-linear structures remains unresolved. Thus, there is a dire need to revisit the problem of low-rank recovery from a totally different perspective, involving manifolds and non-linear structures.

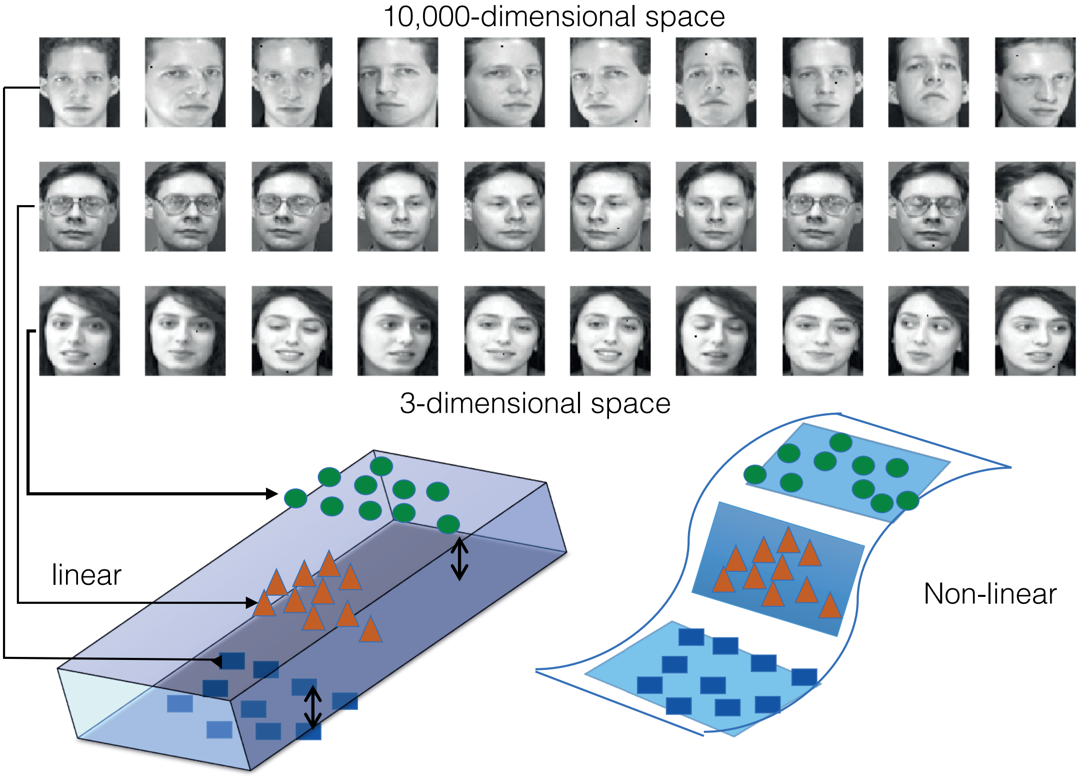


Figure 1.2 – A few images from two different classes of the AT&T dataset and their arbitrary mapping on a 3-dimensional linear and non-linear surface. Clearly, a non-linear surface is a better fit to the low-dimensional point.

1.5 Graphs

In the previous section we discussed the need of manifolds and non-linear surfaces to represent several real world datasets. A manifold is a continuous, smooth, low-dimensional surface, embedded in a high dimensional space. It is imperative to have a mechanism for the discrete characterization of this continuous surface. Given a dataset $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, one can view it as n-samples drawn from a p-dimensional space, where the intrinsic dimensionality of these samples is $r \ll p < n$. Simply put, we have access to a few samples on the manifold and we want to characterize this manifold with these n-samples. Graphs [20] are the key to modeling manifolds.

1.5.1 Graph nomenclature

A graph is a tuple $\mathbb{G} = \{\mathbb{V}, \mathbb{E}, W\}$ where $\mathbb{V}$ is a set of vertices, $\mathbb{E}$ a set of edges, and $\boldsymbol{W} : \mathbb{V} \times \mathbb{V} \to \mathbb{R}_+$ a weight function. The vertices are indexed from $1, \dots, |\mathbb{V}|$ and each entry of the weight matrix $\boldsymbol{W} \in \mathbb{R}_+^{|\mathbb{V}| \times |\mathbb{V}|}$ contains the weight of the edge connecting the corresponding vertices: $\boldsymbol{W}(i,j) = \boldsymbol{W}(v_i, v_j)$. If there is no edge between two vertices, the weight is set to 0. We assume $\boldsymbol{W}$ is symmetric, non-negative and with zero diagonal. We denote by $i \leftrightarrow j$ that node v_i is connected to node v_j. For a vertex $v_i \in \mathbb{V}$, the degree $\boldsymbol{d}(i)$ is defined as the sum of the weights of incident edges: $\boldsymbol{d}(i) = \sum_{j \leftrightarrow i} \boldsymbol{W}(i,j)$.

In this framework, a graph signal is defined as a function $\boldsymbol{s} : \mathbb{V} \to \mathbb{R}$ assigning a value to each vertex. It is convenient to consider a signal $\boldsymbol{s}$ as a vector of size $|\mathbb{V}|$ with the i^{th} component representing the signal value at the i^{th} vertex. The graph signal can be a scalar, a vector, a matrix or even a tensor residing on the node of the graph, depending on the application. For a signal $\boldsymbol{s}$ on the graph $\mathbb{G}$, the gradient $\nabla_{\mathbb{G}} : \mathbb{R}^{|\mathbb{V}|} \to \mathbb{R}^{|\mathbb{E}|}$ is defined as

$$\nabla_{\mathbb{G}} \boldsymbol{s}(i,j) = \sqrt{\boldsymbol{W}(i,j)} \left(\frac{\boldsymbol{s}(j)}{\sqrt{\boldsymbol{d}(j)}} - \frac{\boldsymbol{s}(i)}{\sqrt{\boldsymbol{d}(i)}} \right),$$

where we consider only the pair $\{i,j\}$ when $i \leftrightarrow j$. For a signal $\boldsymbol{c}$ living on the graph edges, the adjoint of the gradient $\nabla_{\mathbb{G}}^* : \mathbb{R}^{|\mathbb{E}|} \to \mathbb{R}^{|\mathbb{V}|}$, called divergence can be written as

$$\nabla_{\mathbb{G}}^* \boldsymbol{c}(i) = \sum_{i \leftrightarrow j} \sqrt{\boldsymbol{W}(i,j)} \left(\frac{1}{\sqrt{\boldsymbol{d}(i)}} \boldsymbol{c}(i,i) - \frac{1}{\sqrt{\boldsymbol{d}(j)}} \boldsymbol{c}(i,j) \right).$$

The Laplacian corresponds to the second order derivative and its definition arises from $\boldsymbol{L}\boldsymbol{s} := \nabla_{\mathbb{G}}^* \nabla_{\mathbb{G}} \boldsymbol{s}$. Finally, the graph can be characterized using a normalized or unnormalized graph Laplacian. The normalized graph Laplacian $\boldsymbol{L}_n$ defined as

$$\boldsymbol{L}_n = \boldsymbol{D}^{-\frac{1}{2}} (\boldsymbol{D} - \boldsymbol{W}) \boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}}$$

where $\boldsymbol{D}$ is the diagonal degree matrix with diagonal entries $\boldsymbol{D}(i,i) = \boldsymbol{d}(i)$ and $\boldsymbol{I}$ the identity matrix. The un-normalized Laplacian is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$.

Since the Laplacian $\boldsymbol{L}$ or $\boldsymbol{L}_n$ is by construction always a symmetric positive semi-definite operator, it always possesses a complete set of orthonormal eigenvectors which we denote with $\boldsymbol{P}$. For convenience, and without loss of generality, we order the set of eigenvalues $\boldsymbol{\Lambda}$ as follows: $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{n-1} = \lambda_{max}$. When the graph is connected, there is only one zero eigenvalue. In fact, the multiplicity of the zero eigenvalue is equal to the number of connected components. See for example [21] for more details on spectral graph theory.

1.5.2 What does a graph portray?

As already described, a graph $\mathbb{G}$ is characterized by its weight matrix $\boldsymbol{W}$ and a Lapalcian $\boldsymbol{L}$. Each entry $\boldsymbol{W}(i,j)$ of the weight matrix contains information about the strength of connectivity between the nodes i and j. The stronger is this weight, the smaller is the distance between the nodes on the manifold. Thus, pairwise relationships between the nodes of a graph give us information about the geometry of the manifold [2]. Furthermore, the Laplacian $\boldsymbol{L}$ of a graph $\mathbb{G}$ is analogous to the Laplace Beltrami operator on the manifolds, as shown in the seminal work of Belkin et. al. [2]. Thus, a graph can be used as a way to discretely characterize a continuous manifold.

1.5.3 Graph construction: A k_{nn}-nearest neighbor graph

So far we have argued that a manifold can be characterized via a graph $\mathbb{G}$. For a given application, the data can either naturally arise in the form of a graph, such as for the case of a social network, where each node models a user and the edges between different users depict that they are in each others' social circle. However, for many applications, the data is unstructured, i.e, it does not come with a natural graph. One might want to construct a graph from the data in such cases.

There can be many ways to construct graphs, see e.g, Ch.4 of [22] or Section 2 of [2]. However, for the purpose of this thesis, we are concerned with a k_{nn}-nearest neighbor graph $\mathbb{G}$. Consider the example of a data matrix $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$, then, assuming that each column of $\boldsymbol{Y}$ is a signal on the graph node, a k_{nn}-nearest neighbors graph needs to be built between the columns of the matrix $\boldsymbol{Y}$. Such a graph can be built using a three-step strategy as follows:

1. **Nearest neighbors search:** In the first step the search for the closest neighbors for all the samples is performed using some distance metric. Depending on the application and dataset, several distance metrics are possible. For example, a Euclidean distance metric is a good choice for a graph between images and a correlation distance for a graph between time series. Thus, each $\boldsymbol{Y}(i)$ is connected to its k_{nn} nearest neighbors $\boldsymbol{Y}(j)$, selected based on the smallest distance, resulting in $|\mathbb{E}|$ number of connections. Note that for our specific example of n-node graph, $|\mathbb{E}| \approx nk_{nn}$.

2. **Weight matrix computation:** In the second step the graph weight matrix $\boldsymbol{W}$ is computed as

$$\boldsymbol{W}(i,j) = \begin{cases} \exp\left(-\frac{\|(\boldsymbol{Y}(i)-\boldsymbol{Y}(j)\|_2^2}{\sigma^2}\right) & \text{if } \boldsymbol{Y}(i) \text{ is connected to } \boldsymbol{Y}(j) \\ 0 & \text{otherwise.} \end{cases}$$

   The parameter σ can be set experimentally as the average distance of the connected samples.

3. **Graph Laplacian:** Finally, the third step consists of constructing the normalized graph Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, or the unnormalized one $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is the degree matrix. This procedure has a complexity of $\mathcal{O}(n|\mathbb{E}|)$.

1.5.4 Graph embedding: Non-linear dimensionality reduction

Given a graph $\mathbb{G}$, it is often of interest to obtain the low-dimensional embedding of the data on a manifold, such as the one visualized in Fig. 1.2. In contrast to the standard PCA for a dataset, a graph embedding, computed from a data graph provides a non-linear r-dimensional embedding of the data on the manifold, where $r \ll n$. Hence, a graph embedding corresponds to non-linear dimensionality reduction. Several state-of-the-art methods exist for this purpose, which include but are not limited to Laplacian Eigenmaps (LE) [2], Locality Preserving Projections (LPP) [23] and Locally Linear Embedding (LLE) [24]. While, we do not describe these methods in detail here, we point out that a majority of these methods rely on the r-Eigenvalue Decomposition (EVD) of a normalized or an unnormalized Laplacian. For a Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, constructed between the columns of a matrix $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$, the EVD is given as:

$$\boldsymbol{L} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top,$$

where the eigenvalues $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$. The first r eigenvectors $\boldsymbol{P}_r$ contain information about the geometry of the data on an r-dimensional manifold.

It is also interesting to point out that the complete set of eigenvectors $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ correspond to the graph Fourier atoms [20]. One can then perform a Graph Fourier Transform (GFT) for a matrix $\boldsymbol{Y}$ as follows:

$$\hat{\boldsymbol{Y}} = \boldsymbol{P}^\top \boldsymbol{Y}.$$

The embedding methods, based on the graph eigenvectors are not limited to 2D matrices only. One of the first extensions for tensors appeared in [25] as Tensor Subspace Analysis. Several variants of this approach have been proposed for unsupervised and supervised dimensionality reduction since then. For example, the authors of [26] proposed a discriminant LLE method for tensors which uses two graphs, one to ensure the same behavior as LLE and the other to enforce class strucure between the embedding. The authors of [27] propose a multilinear graph embedding method and its kernel variant which uses the same two types of graphs as [26]. The authors of [28] propose a tensor based patch alignment method for hyperspectral data and those of [29] propose a semi-supervised framework for tensor based graph embedding.

1.6 Motivation: Non-linear Low-rank Structure & Scalability

Having described the linear and non-linear faces of dimensionality reduction in terms of PCA and graphs, we are now ready to have a first glimpse at the type of problems targeted in this thesis and the motivation behind them. For this purpose we recall:

- Examples of linear and non-linear low-rank datasets from Section 1.1.1. We argued why a single linear subspace might not be a satisfactory assumption for many datasets.
- PCA based applications can be computationally cumbersome: For a matrix of size $p \times n$, Robust PCA scales with $\mathcal{O}(p^2 n)$ per iteration. This problem is even more severe for tensors. For a tensor $\mathcal{Y} \in \mathbb{R}^{n\times n\times n}$, TRPCA scales as $\mathcal{O}(n^4)$.
- Given an r-dimensional graph embedding, there is no method to obtain r-dimensional principal components (PCA based embedding) or r-dimensional linear subspace. A computationally inexpensive method for this task can be used to perform inexpensive subspace updates in the

linear low-rank approximation problems such as Robust PCA, eliminating the use of expensive SVD updates.

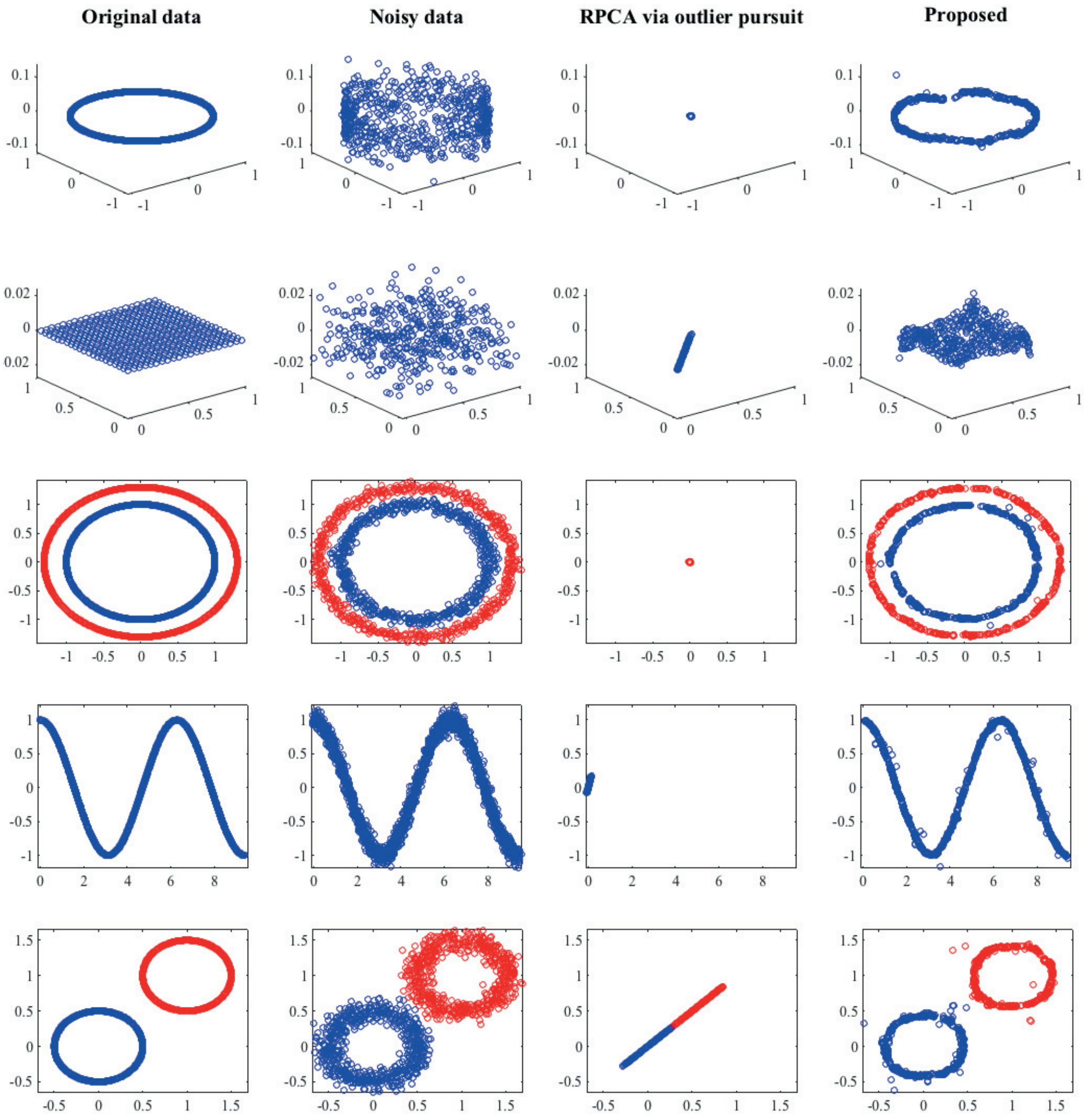


Figure 1.3 – This figure shows the recovery of underlying clean low-rank structure of the noisy 2D or 3D manifolds using the Robust PCA framework [1]. RPCA, which is a linear subspace recovery algorithm fails in all the scenario as it tries to reduce the whole data to a point at the center of the manifold in an attempt to exactly recover the manifold. The last column shows the approximate desired behavior for each of the cases.

While, the examples presented in Section 1.1.1 are cumbersome to visualize, we base our exposition here on a set of very simple non-linear low-rank datasets, sampled from a manifold. In the first column of Fig. 1.3, we present some examples of 1 or 2 dimensional manifolds embedded in a 2 or 3 dimensional space. Given a noisy version of the samples in the space, the goal is to: 1) de-noise and recover the underlying low-rank manifold if the data is low-rank, 2) simply de-noise the dataset and recover the clean manifold if it is not low-rank.

We try to perform this task using the state-of-the-art algorithm Robust PCA (with an ℓ_2 data fidelity term) [1] from the regime of linear dimensionality reduction. This model has the tendency to *exactly* recover the linear low-rank structure of the data. In very simple words the algorithm decomposes a dataset into a low-rank and a noisy matrix.

The first two rows of Fig. 1.3 show two different manifolds lying in a 2D-space. We add noise in the third dimension and create an artificial 3D data from these manifolds. The goal here is to de-noise and recover the low-rank manifold in 2D. The Robust PCA (RPCA) framework, which is a linear subspace recovery algorithm totally fails in this scenario. It can be seen that it tries to reduce all the data points to a single point in the center of the manifold.

Next, consider the datasets shown in the third, fourth and fifth rows of Fig. 1.3. Here, the manifolds are not low-rank, i.e, they only lie in the 2D space and the noise is introduced in the same dimensions. Apparently, this task is easier than the first one because it only involves de-noising. Again we use RPCA to recover the manifold. We observe it fails in this task as well.

The last column of Fig. 1.3 shows what we would approximately like to achieve for each of these datasets. Clearly RPCA is far from achieving the goal. One can simply argue that RPCA can only extract linear structure from the data so it is supposed to fail in such a scenario. The argument is entirely true. It is possible to use some non-linear dimensionality reduction algorithm, like the LE framework but it would recover the low rank structure of the manifold *in a low-dimensional space*.

To be more precise, given the principal components and principal directions, it is straight-forward to do the inverse operation, i.e, go back to the data domain. However, given a graph and embedding, one cannot go back to the data domain, due to non-linearity of the operation. Thus PCA has an interpretation both in the data space and low-dimensional space, however, it does not capture the non-linearity of a manifold.

Thus, currently we do not have algorithms that would simultaneously de-noise the data, recover the manifold and as well as extract the low-rank structures in the data space itself. We need a framework to recover low dimensional manifold structure in the original space of the data points. It should be highly scalable to deal with the datasets consisting of millions of samples that are corrupted with noise and errors. We should also be able to explore relationship between graph based non-linear embedding and PCA based linear embedding. This thesis broadly deals with a class of algorithms which solve such problems, all under one framework, using simple, yet very powerful tools, i.e, graphs.

1.7 Goals of this thesis

We now present the problem statement and goals of this thesis in a formal manner.

OUR GOALS: *Given a high dimensional big dataset (matrix or tensor) that follows some unknown linear or non-linear structure in the low-dimensional space,*

1. *devise an inexpensive method to recover the linear embedding from the non-linear embedding for this dataset.*
2. *recover the underlying linear or non-linear low-rank structure in the data space via highly scalable and efficient algorithms, which makes implicit or explicit use of the above method.*

Note that the two goals are inter-related, specially for the case of linear low-rank approximation problems, such as Robust PCA. Such a method uses an SVD in every iteration to update the linear subspaces. If we devise an inexpensive method to update the linear embedding from the fixed non-linear embedding, then we can use it to avoid doing an SVD in every iteration of Robust PCA. This will become more clear in the later chapters of the thesis.

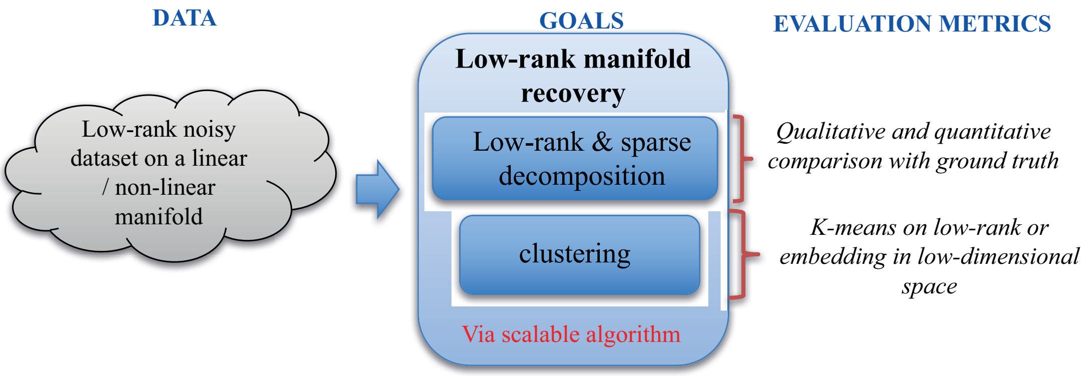

DATA
GOALS
EVALUATION METRICS
Low-rank manifold recovery
Low-rank noisy dataset on a linear / non-linear manifold
Low-rank & sparse decomposition
clustering
Via scalable algorithm
Qualitative and quantitative comparison with ground truth
K-means on low-rank or embedding in low-dimensional space

Figure 1.4 – A summary of the main goals, applications and evaluation schemes used in this thesis.

Low-rank matrix and tensor approximation span a wide variety of applications, such as matrix and tensor completion [30], [31], [32], [33], [34], [35], [36] , [37], [38], [39], [40], [41], convolutional neural networks and sparse coding [42], [43], [44], [45], learning the data structure via multiple subspaces [46], latent low-rank representation [47], [48], transfer learning [49], subspace clustering [50], dynamical system modeling [51], such as human behavior modeling [52], dynamic Magnetic Resonance Imaging Reconstructions [53], [54] and many others. However, throughout this thesis, we will use the following two applications to evaluate our methods in terms of their linear and non-linear manifold modeling capabilities.

1. **Low-rank and sparse decomposition** for several datasets. Recall from our previous discussion that scalability is a primary issue for standard methods, such as matrix and tensor based Robust PCA, which involve iterative SVD in every iteration. We will propose scalable but approximate methods for this purpose and perform a qualitative evaluation of the recovered low-rank matrix

and tensors in this context. The importance of this application is already evident from the discussion in terms of various examples in the previous sections of this chapter.

Furthermore, due to the lack of non-linear benchmark datasets, these experiments will be compared with other linear state-of-the-art methods on the standard benchmark datasets. *Hence, this application will highlight the linear modeling capability of our proposed approximation problems.*

2. **Clustering** of the data in data space or low-dimensional space. We will perform a quantitative evaluation of the clustering quality of our method by performing K-means on the low-rank matrix or the low-dimensional embedding. It is crucial to point out here that the clustering is not a standard application of PCA, because PCA is just a feature extraction method. However, clustering experiments had been widely adopted as a standard procedure to demonstrate the quality of the feature extraction methods [55], [56], [6], [57], [5], [58], [3]. Furthermore, a theoretical analysis of our proposed methods for low-rank and sparse decomposition will show that clustering is a by-product of our methods.

   Recall that clustering corresponds to partitioning of data on a union of low-dimensional subspaces or on a smooth low-dimensional manifold. *Therefore, this application will highlight the non-linear manifold modeling capability of our proposed approximation problems.*

Fig. 1.4 visualizes the goals, applications and evaluation metrics for this thesis with a simple illustration.

1.8 Proposed Methodology & Thesis Contributions

In order to achieve the goals of this thesis, we look at the low-rank approximation problem in an entirely novel way, from the perspectives of graphs. Unlike many other state-of-the-art methods, discussed in Chapter 2, which incorporate graph regularization into the PCA framework, to get the best of linear and non-linear worlds, we study the relationship between linear and non-linear embeddings. More specifically, we explore if it is possible to extract low-rank structures without explicitly enforcing low-rank constraints, via graphs only. Based on our study, we propose a new low-rank signal model, which uses graph information, in the form of graph eigenvectors to encode or factorize a low-rank matrix or tensor. Based on this model, we propose a plethora of approximation problems which use graph eigenvectors or graph regularization to approximate low-rank matrices and tensors. Thus, we have at our disposal, the following ingredients:

- k_{nn}-nearest neighbor graphs
- First k graph eigenvectors
- graph regularization prior

It is important to remark here that we compromise the exactness of the recovery provided by the state-of-the-art algorithms, such as PCA and Robust PCA. While, it seems to be a drawback of our work, we recall that this is the price that we have to pay to make our algorithms scalable, a key goal of this thesis. Dropping the low-rank constraints either alleviates the non-convexity or the computational complexity of low-rank and sparse decomposition algorithms. This will be made clear in the following chapters. Furthermore, we also recall Fig. 1.3, where we saw that even though our proposed framework (rightmost column) is approximate, it does recover non-linear low-rank structures much better than the

PCA based methods in certain cases. Although, the benefit of scalability and non-linear low-rankness, comes at the price of loss of exactness, compromising on the exactness of recovery might be useful for many applications.

Finally, the use of graphs is not only limited to recover non-linear low-rank structures but is also tied to the scalabiltiy aspect of our proposed solutions. We will make this connection clear in the later chapters of this thesis. Simply, put, our proposed solutions exploit structure or the intrinsic non-linearity along the different dimensions of the datasets and this property automatically comes with the scalability of these algorithms as well. Thus, *joint incorporation of structure and scalability* is the primary reason we choose graphs over a variety of other methods available in the literature (Chapter 2).

This thesis is divided into three different parts and each part contains a set of chapters as follows:

1.8.1 Part I: Introduction, Background & Motivation

This part of the thesis comprises of Chapters 1 and 2. Chapter 1 served as an introduction to the thesis and laid down the foundation in terms of linear and non-linear dimensionality reduction. The pros and cons of both were discussed. It was argued that low-rank approximation falls under the broad category of dimension reduction. Several examples of low-rank approximation were presented, both for the cases of matrices and tensors. The goals of this thesis, i.e, to recover linear and non-linear low-rank structures and study the relationship between linear and non-linear embedding were stated. Finally, it was stated that we would like to achieve the goals by using various graph based ingredients, such as graph eigenvectors and graph priors, without the explicit use of low-rank constraints, for the purpose of scalability.

Chapter 2: Related Work & Terminology

Chapter 2 introduces important notation and terminology that will be used throughout the thesis. It also provides a detailed literature review on the different types of low-rank approximation methods for matrices and tensors, which incorporate structure in the form of graph. The short-comings of each of these methods are discussed.

1.8.2 Part II: Proposed Low-rank Decomposition Model

Part II of the thesis comprises of Chapters 3 and 4 and corresponds to the main contributions of this thesis.

Chapter 3: Multilinear Low-rank Tensors on Graphs

Chapter 3 introduces our novel low-rank matrix and tensor decomposition model ‘Multilinear low-rank tensor on graphs’ (MLRTG) which states that a low-rank tensor can be encoded as a multilinear combination of the graph eigenvectors. The eigen-gap assumption underlying MLRTG and its implications such as approximate graph stationarity, low-frequency power concentration and k-clusterable nature of the tensor along each of the modes are discussed in detail. The signal model is justified with a few real world examples and methods to generate artificial MLRTG are discussed. Finally, the utility of constructing graphs across the different modes of the tensor is explained.

Chapter 4: From graph basis to PCA basis: Graph Multilinear SVD

Chapter 4 introduces the notion of Graph Multilinear SVD - an alternative to the standard SVD method, using the MLRTG framework of Chapter 3. The GMLSVD method is derived both intuitively by considering an example of a real world low-rank tensor and as well as mathematically from the MLSVD algorithm. This method provides a mechanism to extract PCA based linear embedding from the graph based non-linear embedding. As the non-linear graph based embedding is fixed, the linear embedding can be updated via low-cost operations on it. This is useful for defining a plethora of graph based matrix and tensor low-rank approximation problems in the next part of the thesis.

1.8.3 Part III: Proposed Approximation Methods & Applications

Based on the MLRTG and GMLSVD framework, this part, which comprises of Chapters 5, 6 and 7 presents three different matrix and tensor based scalable approximation problems for the applications of low-rank and sparse decomposition and clustering of datasets in low-dimensional space. Our proposed approximation problems fall into two categories:

- Those which make implicit use of the GMLSVD framework via graph Dirichlet / smoothness priors (explained later).
- Those which make explicit use of the GMLSVD framework, i.e., the graph eigenvectors of each of the modes of the tensor.

Chapters 5 and 6 present the matrix based approximation problems Fast Robust PCA on Graphs (FRPCAG) and Compressive PCA (CPCA), which are meant for the cases when the number of samples n and the features p are very large and one can afford to lose the exactness of the representation for gain in speed. In this context, these methods fall under the first category, as computing the graph eigenvectors and tuning the size of subspaces can be cumbersome. Chapter 7 presents tensor based approximation problems for which n is not very large but n^d, where d is the number of modes of the tensor is large. Such methods can afford to explicitly use the eigenvectors as their computation is not expensive and scales linearly with the size of the tensor mode (n).

Chapter 5: Fast Robust PCA on Graphs

Fast Robust PCA on Graphs (FRPCAG) is the first of the series of approximation problems for low-rank and sparse decomposition of matrices. As already mentioned earlier, the method implicitly makes use of the GMLSVD framework via a dual graph filtering approach to jointly filter the rows and columns of a matrix and extract an approximate low-rank representation, without the use of a nuclear norm operator. A detailed theoretical analysis has been carried out which quantifies the approximation error in terms of the eigen-gap of the row and column graph Laplacians. The method scales linearly with the number of elements in the matrix. Experiments on static background separation from dynamic foreground and clustering of the low-rank features are presented.

Chapter 6: Compressive PCA on Graphs

Compressive PCA on graphs (CPCA) is a highly scalable extension of FRPCAG for matrices which involves randomly sampling the rows and columns of the data matrix based on novel graph based coherence measures. Similarly to FRPCAG, the method brings the GMLSVD framework into use implicitly. A restricted isometry property is introduced for MLRTG which quantifies the number of rows and column or the nodes of the graphs to be sampled. FRPCAG is then run on the sub-sampled matrix and the full low-rank matrix is decoded on the row and column graphs using a variety of decoders. The approximate decoders presented in this chapter are of specific interest because they exploit certain heuristics to decode the subspaces of low-rank matrix in a highly parallel manner without the explicit need of parameter tuning. Approximation error for the decoders is characterized in terms of the eigen-gap of the Laplacians. Experiments on low-rank and sparse decomposition and clustering are presented.

Chapter 7: Tensor Low-rank Decomposition on Graphs

A primary short-coming of FRPCAG and CPCA is the manifold shrinking effect which is a consequence of the non-linear penalization of the singular values of the matrix. Extending FRPCAG for tensors is also computationally expensive. This chapter introduces a new set of approximation problems for tensors, which explicitly bring into use the MLRTG and GMLSVD frameworks and overcome the high computational complexity and manifold shrinking effect of FRPCAG. Two approximation problems, Graph Core Tensor Pursuit (GCTP) and Tensor Robust PCA on Graphs (TRPCAG) are introduced for the recovery of low-rank tensors from Gaussian and sparse noise. A detailed theoretical analysis and experimental comparison of the method is performed with the previously proposed methods. It has been proved that making an explicit use of the graph eigenvectors (GMLSVD framework) is equivalent to introducing structured priors in the approximation problems.

An summary of the contributions of this thesis is presented in Fig. 1.5.

Chapter 8: Discussion & Conclusions

Chapter 8 concludes the thesis with an overview of the entire set of contributions and compares all the approximation problems under a unified framework. Limitations of each of the methods are summarized and future work is motivated based on the discussion. Finally, many different application based extensions of the proposed framework are presented.

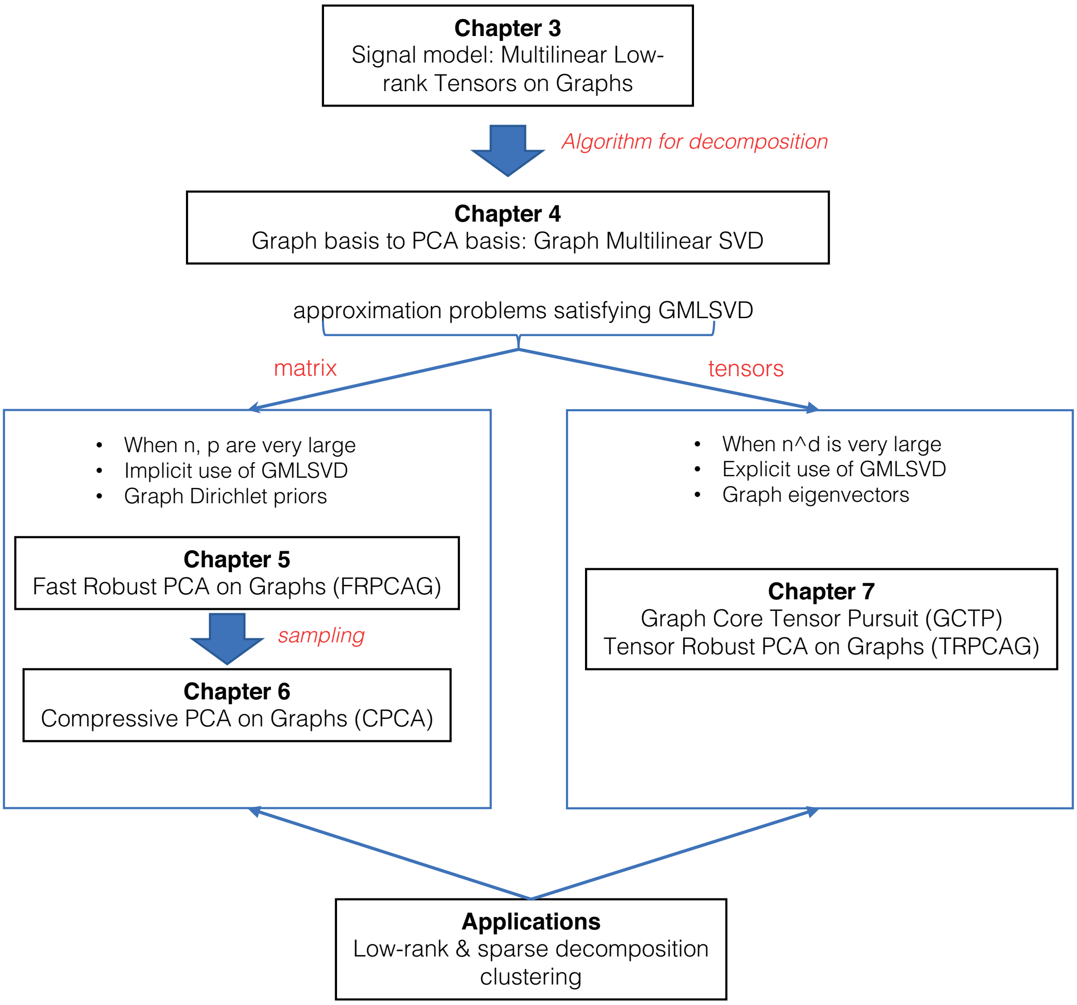

Chapter 3
Signal model: Multilinear Low-rank Tensors on Graphs
Algorithm for decomposition
Chapter 4
Graph basis to PCA basis: Graph Multilinear SVD
approximation problems satisfying GMLSVD
matrix
tensors
When n, p are very large
Implicit use of GMLSVD
Graph Dirichlet priors
Chapter 5
Fast Robust PCA on Graphs (FRPCAG)
sampling
Chapter 6
Compressive PCA on Graphs (CPCA)
When n^d is very large
Explicit use of GMLSVD
Graph eigenvectors
Chapter 7
Graph Core Tensor Pursuit (GCTP)
Tensor Robust PCA on Graphs (TRPCAG)
Applications
Low-rank & sparse decomposition
clustering

Figure 1.5 – A summary of the contributions and organization of this thesis.

2 Related Work & Terminology

In Chapter 1, we pointed out that one of our two goals is to propose non-linear low-rank approximation algorithms by taking advantage of PCA and graphs. Although, we mentioned that we will drop the low-rank constraints to achieve scalability, there are several state-of-the-art methods which combine PCA and graphs from the perspective of enhancing the performance of PCA. Another important aspect of our proposed framework is its scalability. Several methods have been proposed in the last decade which tend to scale up the standard low-rank decomposition methods. These methods should be studied here before we go into the details of the contributions of thesis. Some of the methods mentioned here will also be compared for the purpose of low-rank and sparse decomposition and clustering applications with our proposed methods.

First, we describe the basic terminology and notation that will be used throughout the thesis.

2.1 Terminology & Notation

Throughout this thesis, we will follow a few conventions as stated below. Some of these have already been described in Chapter 1. More specific convention and complex notation will be introduced as needed.

2.1.1 Vectors & Matrices

- We represent vectors with small bold letters $\boldsymbol{y}$, matrices with bold capital letters, such as $\boldsymbol{Y}$ and tensors with bold caligraphic letters, such as $\mathcal{Y}$.
- The term 'dataset' will broadly refer to either a matrix or a tensor, unless explicitly stated.
- Throughout, we refer to a matrix $\boldsymbol{Y}$ as a 2D tensor and vice versa.
- We denote the i^{th} column of $\boldsymbol{Y}$ as $\boldsymbol{Y}(i)$ and the i^{th} row of $\boldsymbol{Y}$ as $\boldsymbol{Y}^\top(i)$.
- We represent the (i,j) entry of a matrix $\boldsymbol{Y}$ as $\boldsymbol{Y}(i,j)$.
- We denote the first r columns of a matrix $\boldsymbol{Y}$ as $\boldsymbol{Y}(:,1:r) = \boldsymbol{Y}_r$.
- Unless stated otherwise, a data matrix $\boldsymbol{Y}$ is always assumed to be of the size $p \times n$, where p is the number of features or pixels and n is the number of samples and $p < n$. Hence, for a data matrix

$\boldsymbol{Y}$, we always assume that the data samples are vectorized and stacked in the columns of $\boldsymbol{Y}$.

2.1.2 Tensors

- In contrast to a data matrix, we do not associate the notion of features and samples to different tensor dimensions. We call the different dimensions of a tensor as 'modes'.
- We always work with 3D data tensors $\mathcal{Y}$ of size $n \times n \times n$, i.e, equal number of samples in all the dimensions or modes of the tensor. The concepts can be extended in a straight-forward manner for higher dimensional tensors with arbitrary number of samples in every dimension.
- **Tensor Mode Indices:** Let $\mathbb{I} = \{1,2,3\}$, then, we use $\mu \in \mathbb{I}$ to refer to an index from this set. We also write $\mathbb{I}_{-\mu}$ to refer to all indices except μ.
- **Tensor Size:** Let $n_{\mathbb{I}} = \prod_{\mu \in \mathbb{I}} n_\mu = n^3$ and $n_{-\mu} = \prod_{\mu \in \mathbb{I}_{-\mu}} n_\mu = n^2$.
- **Tensor Rank:** We will use r_μ and k_μ for the multilinear rank of tensors and lower frequency indices of the Laplacians. A tensor is said to be multilinear low-rank if it is low-rank along each of its modes [59]. Throughout we will assume without the loss of generality that $k_\mu \geq r_\mu$ and $k_1 = k_2 = k_3 = k$, as well as $r_1 = r_2 = r_3 = r$. Also let $k_{\mathbb{I}} = \prod_{\mu \in \mathbb{I}} k_\mu = k^3$, $r_{\mathbb{I}} = \prod_{\mu \in \mathbb{I}} r_\mu = r^3$. Similarly, $k_{-\mu} = k^2$ and $r_{-\mu} = r^2$ can be defined.
- **Vectorization:** For a tensor $\mathcal{Y}$ its vectorization is denoted as $\text{vec}(\mathcal{Y}) \in \mathbb{R}^{n^3}$.
- **Tensor Matricization:** The μ^{th} matricization or flattening of $\mathcal{Y}$ is its arrangement into a matrix $\boldsymbol{Y}^\mu \in \mathbb{R}^{n \times n^2}$.
- For a tensor $\mathcal{Y}$, its i^{th} slice along mode μ is a sub-tensor formed by fixing the i^{th} index along the μ^{th} mode and is denoted as $\mathcal{Y}^\mu_i$.

2.1.3 Vector, Matrix & Tensor Operators

Throughout this thesis, we use the following notation for vector, matrix and tensor operators:

- vector ℓ_2 norm: $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i \|\boldsymbol{x}(i)\|^2}$
- vector ℓ_1 norm: $\|\boldsymbol{x}\|_1$
- Matrix Frobenius norm as the sum of ℓ_2 norms of the rows or columns: $\|\boldsymbol{X}\|_F^2 = \sum_i \|\boldsymbol{X}(i)\|_2^2$
- Matrix ℓ_1 norm as the sum of ℓ_1 norms of rows or columns: $\|\boldsymbol{X}\|_1 = \sum_i \|\boldsymbol{X}(i)\|_1$
- Matrix nuclear norm as the sum of the singular values of a matrix: $\|\boldsymbol{X}\|_* = \sum_{i=1}^{p} \sigma_i$.
- Tensor nuclear norm as the sum of matrix nuclear norms of each of the matrices $\boldsymbol{X}^\mu$ formed from $\mathcal{X}$: $\sum_\mu \|\boldsymbol{X}^\mu\|_*$.
- Kronecker product of the two matrices with $\otimes$.
- Cartesian product of the two matrices with $\times$.
- Hadamard product of the two matrices with $\circ$.

2.1.4 Low-rank Matrix

The definition of low-rank matrix and multilinear low-rank tensor has been provided in Chapter 1, Sections 1.2 and 1.3. Here, we present general convention that we will maintain throughout the thesis.

- The low-rank approximation of a matrix / 2D tensor $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$ is represented by $\boldsymbol{X} \in \mathbb{R}^{p\times n}$.
- The rank of a matrix is unique and is represented by r, where $r \ll p < n$.
- Referring to Chapter 1, Section 1.2.1, the low-rank approximation $\boldsymbol{X}$ of a matrix $\boldsymbol{Y}$ can also be given as $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{Q}_r^\top$, where $\boldsymbol{U}_r \in \mathbb{R}^{p\times r}$ and $\boldsymbol{Q}_r \in \mathbb{R}^{n\times r}$.
- Given the SVD of $\boldsymbol{X}$, i.e, $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top$, $\boldsymbol{Q}_r = \boldsymbol{\Sigma}_r \boldsymbol{V}_r$.
- The singular values $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ (as $p < n$) are sorted in the non-increasing order $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$.
- We will call $\boldsymbol{U}$ and $\boldsymbol{Q}$ above as ’factors’ or ’principal directions’ and ’principal components’, alternatively.
- Unless stated otherwise, the factors $\boldsymbol{U}$ and $\boldsymbol{Q}$ are always obtained via the standard PCA optimization problem eq. 1.1.

2.1.5 Low-rank Tensor

- The low-rank approximation of a tensor $\mathscr{Y} \in \mathbb{R}^{n\times n\times n}$ is represented by $\mathscr{X} \in \mathbb{R}^{n\times n\times n}$.
- The multilinear rank of a tensor [11] is represented as a tuple (r_1, r_2, r_3).
- We assume that $r_1 = r_2 = r_3 = r$. Furthermore, we also assume $r \ll n$.
- The low-rank approximation $\mathscr{X}$ of $\mathscr{Y}$ can also be given as $\text{vec}(\mathscr{X}) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\text{vec}(\mathscr{Z})$, where $\mathscr{Z} \in \mathbb{R}^{r\times r\times r}$, $\text{vec}(\cdot)$ denotes the vectorization operation and $\otimes$ denotes the Kronecker product of matrices. This is known as the **Tucker Decomposition** [12], [13], [14], [15].
- We call $\boldsymbol{U}_r^\mu \in \mathbb{R}^{n\times r}$, $\forall\, \mu$ above as ’factors’ of the decomposition.
- We do not associate the notion of principal directions and principal components with the tensor factors.
- Unless stated otherwise, the tensor factors are always obtained via MLSVD [11], as described in detail in Chapter 1, Section 1.3.1.

2.1.6 Graphs

Throughout this thesis, we refer to graphs as k_{nn}-nearest neighbor graphs, represented as $\mathbb{G}$. A general introduction to graphs and graph construction was presented in Section 1.5. Here we specifically refer to a few conventions we follow in the context of graphs constructed from data matrices and tensors.

- Unless stated otherwise we use the unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is the degree matrix and $\boldsymbol{W}$ is the adjacency matrix, using the Gaussian kernel, as defined in Section 1.5.3.

- For a matrix $\boldsymbol{Y}$, the graph $\mathbb{G}$ can be either constructed between the rows or the columns of matrix $\boldsymbol{Y}$. This will be explicitly specified on a case-by-case basis.
- We will refer to the graph Laplacian among the rows of a matrix as $\boldsymbol{L}^1$ or $\boldsymbol{L}^r$ and among the columns of the matrix as $\boldsymbol{L}^2$ or $\boldsymbol{L}^c$.
- Throughout we will assume that the graphs and their corresponding Laplacians are always constructed from the rows or columns of the original data matrix $\boldsymbol{Y}$.
- For a tensor $\mathcal{Y}$, there might be several ways to construct graphs. However, we specifically refer to graphs constructed between the rows of matrices $\boldsymbol{Y}^\mu$, formed by flattening the μ^{th} mode of $\mathcal{Y}$. Such graphs and their corresponding Laplacians are represented as $\mathbb{G}^\mu$ and $\boldsymbol{L}^\mu$. The significance of this way of constructing graphs will be justified later in the thesis.
- We will often refer to the eigenvalue decomposition of a graph Laplacian $\boldsymbol{L} \in \mathbb{R}^{n \times n}$. This is given as $\boldsymbol{L} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top$, where the eigenvalues $\boldsymbol{\Lambda}$ are sorted in the non-increasing order $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$.
- In the context of tensors, the eigenvalue decomposition of $\boldsymbol{L}^\mu$, computed from $\boldsymbol{Y}^\mu$ is denoted as $\boldsymbol{P}^\mu \boldsymbol{\Lambda}^\mu \boldsymbol{P}^{\mu\top}$.
- Very often, we will work with the set of smallest k-Laplacian eigenvectors and eigenvalues. We will denote this set as $(\boldsymbol{P}_k, \boldsymbol{\Lambda}_k)$.
- We will also refer to the eigenvectors and eigenvalues above k, i.e, $(\bar{\boldsymbol{P}}^\mu_k \in \mathbb{R}^{n \times (n-k)}, \bar{\boldsymbol{\Lambda}}^\mu_k \in \mathbb{R}^{(n-k) \times (n-k)})$, as the complement set.
- We will denote by $\boldsymbol{P}^{-\mu}_{k^2} \in \mathbb{R}^{n^2 \times k^2}$, the Kronecker product of all the smallest k eigenvector matrices, except $\boldsymbol{P}^\mu_k$, i.e, $\boldsymbol{P}^{-1}_{k^2} = \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k$.

Note that k_{nn} is different from k, where the former stands for nearest neighbors and latter for the smallest eigenvectors of the graph.

Graph Dirichlet Energy

Similarly to the ℓ_2 norm which is commonly used to characterize / model the smoothness of signals in Euclidean domain, graph Dirichlet energy [20] can be used to do so for signals defined on the nodes of a graph. For a 1D signal $\boldsymbol{x}$, the smoothness w.r.t to a Laplacian $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is given as:

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}.$$

This extends in a straight-forward manner for Laplacians defined for matrices of the form $\boldsymbol{X} \in \mathbb{R}^{p \times n}$:

$$\mathrm{tr}(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top),$$

where $\mathrm{tr}(\cdot)$ is the matrix trace operator.

Fast Computation of Graphs: The FLANN library

For big or high dimensional datasets, the exact k_{nn}-nearest neighbors strategy, described in Section 1.5.3 can become computationally cumbersome ($\mathcal{O}(n^2)$, for n nodes). In such a scenario the computations can be made efficient using the FLANN library (Fast Library for Approximate Nearest Neighbors searches in high dimensional spaces) [60]. The quality of the graphs constructed using this strategy is slightly worse as compared to the exact strategy, due to the approximate nearest neighbor search method. However, for a graph of n nodes, the computationally complexity is as low as $\mathcal{O}(n \log n)$ [61]. Throughout this work we construct our graphs using FLANN irrespective of the size of the datasets. It is important to point out here that one can tune certain parameters of FLANN to trade-off between the approximation / speed of the construction and the exactness of the nearest-neighbor search. We use the Graph Signal Porcessing Toolbox GSPBOX [62] by Perraudin et al. which supports FLANN for all types of graph computations.

Fast Computation of first 'k' Laplacian Eigenvectors

The computation of the first 'k' smallest eigenvectors of large sparse Laplacians lies at the heart of our proposed framework. We rely on the notion that this computation is relatively inexpensive, i.e, scales linearly with the number of nodes in the graph. Thus, for a 2D tensor $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$, the cost of computing the first k eigenvectors of a Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is $\mathcal{O}(nk^2)$, where $k \ll n$. Nevertheless, we point out and briefly discuss the state-of-the-art for the decomposition of large sparse matrices and encourage the use of these techniques.

Many of the existing classical eigensolvers for sparse matrices employ the single-vector version of iterative algorithms, such as the power method and Lanczos algorithm [63]. The Lanczos algorithm iteratively constructs the basis of the Krylov subspace to obtain the eigendecomposition, which has been extensively applied in the well known eigensolvers, e.g., eigs in Matlab (ARPACK) [64] and PROPACK [65]. However, a primary constraint of the single-vector iterative algorithms is that they can only compute the leading eigenvalue/eigenvector or have difficulty in computing multiplicities of eigenvalues. In contrast, the block version of iterative algorithms using multiple starting vectors, such as the randomized SVD [66] and block Lanczos [63], avoid such problems by utilizing efficient matrix-matrix operations with better caching behavior. Very recently the authors of [67] have introduced a multiscale spectral decomposition approach which first clusters the graph into smaller clusters whose spectral decomposition can be computed efficiently and independently. The union of the subspaces of the clustered graphs have significant overlap with the subspace of the original graph if the graph is appropriately clustered. For a matrix of the size mentioned above, the single core version of this algorithm still poses a complexity of $\mathcal{O}(nk^2)$. However, this algorithm can parallelized for each of the c clusters to gain a significant reduction in the computational complexity $\mathcal{O}(n\frac{k^2}{c})$ with a negligible performance loss. Such methods can be used for computation of Laplacian eigenvectors in the context of this thesis.

2.2 Graph Regularized PCA: A Review

A very common method to incorporate structure in the standard PCA problem is to use manifold or graph regularization. Several state-of-the-art PCA based methods have proposed to do this in various formulations. These methods can be divided into three types depending on the type of graph regularization used:

1. Factorized models: graph regularization on factors
2. Non-factorized models: graph regularization on the low-rank matrix
3. Multiple subspace models

Before describing the graph regularized PCA based methods, we point out that most of these methods use a graph regularization w.r.t graph of samples. Thus, for a matrix of size $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$, the graph Laplacian is constructed between the columns of the matrix $\boldsymbol{Y}$, i.e, between the data samples. Most of the methods presented below, hence, incorporate graph regularization to improve the embedding of samples in low-dimensional space. An exception to this convention in the regime of low-rank matrix approximation methods is the work of Kalofolias et. al [33]. This work incorporates a dual graph regularization, i.e, a graph between the samples and another between the features of the data matrix $\boldsymbol{Y}$. We briefly review a few of these methods.

2.2.1 Factorized Models

These models are of the form of standard PCA, i.e, they determine the low-rank approximation of a data matrix $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$ via a matrix factorization approach. With a small variation of constraints, these methods incorporate graph structure via an explicit regularization on the principal components or the embedding of data in low-dimensional space.

Graph Laplacian PCA (GLPCA)

The authors of [3] proposed graph Laplacian PCA (GLPCA) which solves the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{U}_r,\boldsymbol{Q}_r} \|\boldsymbol{Y} - \boldsymbol{U}_r\boldsymbol{Q}_r^\top\|_F^2 + \gamma\,\mathrm{tr}(\boldsymbol{Q}_r^\top \boldsymbol{L}\boldsymbol{Q}_r) \\ &\quad \text{s.t } \boldsymbol{Q}_r^\top\boldsymbol{Q}_r = \boldsymbol{I}_r, \end{aligned} \tag{2.1}$$

where the Laplacian $\boldsymbol{L}$ of the k_{nn}-graph is computed from the columns of the data matrix $\boldsymbol{Y}$. Note that the orthornomality constraint is on $\boldsymbol{Q}_r$, thus, the model assumes that the data embedding is orthonormal. This also results in a closed form solution of the problem, determined by the first r eigenvectors of a generalized Laplacian. The authors show that the solution to this problem is a linear combination of the solution of standard PCA and Laplacian embedding. Finally, note that the model is non-convex due to the orthonormality constraint, nevertheless, it has a unique solution. The computational complexity of this method is equivalent to that of a r-Eigenvalue Decomposition of a Laplacian: $\mathcal{O}(nr^2)$.

Robust Graph Laplacian PCA (RGLPCA)

The GLPCA model above is not robust to outliers and gross errors in the dataset. Thus, in the same work, the authors of [3] also propose a robust variant of GLPCA, which they call Robust Graph Laplacian PCA (RGLPCA). RGLPCA solves the following problem:

$$\begin{aligned} &\min_{\boldsymbol{U}_r,\boldsymbol{Q}_r} \|\boldsymbol{Y} - \boldsymbol{U}_r\boldsymbol{Q}_r^\top\|_{2,1} + \gamma\,\mathrm{tr}(\boldsymbol{Q}_r^\top \boldsymbol{L}\boldsymbol{Q}_r) \\ &\quad \text{s.t } \boldsymbol{Q}_r^\top\boldsymbol{Q}_r = \boldsymbol{I}_r \end{aligned} \tag{2.2}$$

The $\ell_{2,1}$ norm above, makes the problem robust to outliers in the data samples. However, RGLPCA does not possess a closed form solution like GLPCA and hence results in a non-convex optimization problem due to the orthonormality constraint. The computational complexity of this method is equivalent to that of an r-Eigenvalue Decomposition of a Laplacian in every iteration of the algorithm: $\mathcal{O}(Inr^2)$, where I is the number of iterations.

Manifold Matrix Factorization (MMF)

The authors of [6] introduced Manifold Regularized Matrix Factorization (MMF), in the same spirit as [3] . However, a slight variation of the problem results in an entirely different framework.

$$\begin{aligned}&\min_{\boldsymbol{U}_r,\boldsymbol{Q}_r} \|\boldsymbol{Y}-\boldsymbol{U}_r\boldsymbol{Q}_r^\top\|_F^2+\gamma\,\mathrm{tr}(\boldsymbol{Q}_r^\top\boldsymbol{L}\boldsymbol{Q}_r)\\&\quad \text{s.t } \boldsymbol{U}_r^\top\boldsymbol{U}_r=\boldsymbol{I}_r\end{aligned}\tag{2.3}$$

In contrast to [3], the optimization problem above has an orthonormality constraint on $\boldsymbol{U}_r$, i.e, principal directions instead of $\boldsymbol{Q}_r$ (principal components). Hence, the problem structure resembles very closely with that of standard PCA. It scales as $\mathcal{O}(nr^2)$.

Multiple Manifold Matrix Factorization (MMMF)

In the spirit similar to the MMF model above, the authors of [56] proposed an ensemble of graph regularization terms, called multiple manifold matrix factorization (MMMF). MMMF solves the following optimization problem.

$$\begin{aligned}&\min_{\boldsymbol{U}_r,\boldsymbol{Q}_r,\boldsymbol{\alpha}} \|\boldsymbol{Y}-\boldsymbol{U}_r\boldsymbol{Q}_r^\top\|_F^2+\gamma\,\mathrm{tr}\big(\boldsymbol{Q}_r^\top(\sum_i\boldsymbol{\alpha}(i)\boldsymbol{L}^i)\boldsymbol{Q}_r\big)+\beta\|\boldsymbol{\alpha}\|_2^2\\&\quad \text{s.t } \boldsymbol{U}_r^\top\boldsymbol{U}_r=\boldsymbol{I}_r\\&\quad \sum_i\boldsymbol{\alpha}(i)=1\end{aligned}\tag{2.4}$$

The idea behind MMMF is that one can use a sparse linear combination of multiple graphs constructed via different methods to make the regularization term more robust to noisy data. However, the best sparse combination is determined during the optimization process, which makes the problem non-convex. The computational complexity of this method is more than that of MMF, as it requires minimizing the MMF problem in every iteration.

Multiple Hypergraph Regularized Matrix Factorization (MHMF)

A hypergraph [68] is an important graph structure, that is capable of modeling more complictaed relationships between the nodes, via multiple edge sharing among a set of nodes. A straight-forward extension of the MMMF model for hypergraphs was proposed by the authors of [57]. MHMF solves the following optimization problem

$$\begin{aligned}&\min_{\boldsymbol{U}_r,\boldsymbol{Q}_r,\boldsymbol{\alpha}} \|\boldsymbol{Y}-\boldsymbol{U}_r\boldsymbol{Q}_r^\top\|_F^2+\gamma\,\mathrm{tr}\big(\boldsymbol{Q}_r^\top(\sum_i\boldsymbol{\alpha}(i)\boldsymbol{L}_h^i)\boldsymbol{Q}_r\big)+\beta\|\boldsymbol{\alpha}\|_2^2\\&\quad \text{s.t } \boldsymbol{U}_r^\top\boldsymbol{U}_r=\boldsymbol{I}_r\\&\quad \sum_i\boldsymbol{\alpha}(i)=1\end{aligned}\tag{2.5}$$

where $\boldsymbol{L_h}$ is the hypergraph Laplacian. The problem is non-convex and the computational complexity of this algorithm is the same as MMMF.

2.2.2 Non-factorized Models

Unlike the factorized models above, which involve a graph regularization on the factors, a few models explicitly involve graph regularization on the low-rank matrix approximation.

Manifold Matrix Factorization (MMF)

Although, the MMF [6] model falls under the category of factorized models, with a simple change of variables it can be tailored to possess a closed form solution and becomes more interpretable. Note that for $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{Q}_r^\top$

$$\operatorname{tr}(\boldsymbol{Q}_r^\top \boldsymbol{L} \boldsymbol{Q}_r) = \operatorname{tr}(\boldsymbol{U}_r \boldsymbol{Q}_r^\top \boldsymbol{L} \boldsymbol{Q}_r \boldsymbol{U}_r^\top) = \operatorname{tr}(\boldsymbol{X} \boldsymbol{L} \boldsymbol{X}^\top),$$

holds due to invariance property of trace under cyclic permutation. Thus, the problem formulation of MMF [6] is equivalent to a graph regularization on the columns of low-rank approximation.

Matrix Completion on Graphs

The authors of [33] propose matrix completion on graphs, as a method to fill in the missing entries of a low-rank matrix via a dual graph regularization approach on the low-rank matrix $\boldsymbol{X}$.

$$\min_{\boldsymbol{X}} \frac{1}{2} \|\boldsymbol{M} \circ (\boldsymbol{Y} - \boldsymbol{X})\|_F^2 + \gamma \|\boldsymbol{X}\|_* + \frac{\gamma_r}{2} \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X}) + \frac{\gamma_c}{2} \operatorname{tr}(\boldsymbol{X} \boldsymbol{L}^c \boldsymbol{X}^\top), \tag{2.6}$$

where $\boldsymbol{M}$ is the binary mask matrix, which contains ones for the known and zeros for the unknown entries. Although the above problem is specifically for matrix completion, it can also be seen as a low-rank matrix approximation in the context of simple PCA when $\boldsymbol{M}$ is the matrix of all ones. This model is convex but it scales as $\mathcal{O}(Inp^2)$, where I is the number of iterations.

2.2.3 Multiple subspace models: Low-rank Representation (LRR)

The third type, proposed by the authors of [69] treats the low rank representation as a coefficient learning problem, where each data sample can be represented as a low rank combination of the atoms of a fixed overcomplete dictionary. Commonly known as Low-rank Representation model (LRR), it solves the following optimization problem:

$$\min_{\boldsymbol{C_x}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{C_x}\|_F^2 + \lambda \|\boldsymbol{C_x}\|_*, \tag{2.7}$$

where $\boldsymbol{C_x} \in \mathbb{R}^{n \times n}$ is the low-rank coefficient matrix for $\boldsymbol{Y}$ and $\boldsymbol{D} \in \mathbb{R}^{p \times n}$ is the representation dictionary. The authors propose to use the data matrix $\boldsymbol{Y}$ itself as a dictionary. Thus, the goal of the above problem is to determine the low-rank coefficients (columns) of $\boldsymbol{C_x}$ for self representing the training matrix $\boldsymbol{Y}$. Due to the low-rankness constraint, the model ends up learning the coefficients $\boldsymbol{C_x}$ in a union of subspaces.

Graph regularized LRR (GLRR)

The authors of [70] propose to incorporate graph regularization to the LRR framework, known as GLRR. They propose the following optimization problem:

$$\min_{\boldsymbol{C_x}} \|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C_x}\|_F^2 + \lambda \|\boldsymbol{C_x}\|_* + \gamma \operatorname{tr}(\boldsymbol{C_x}\boldsymbol{L}\boldsymbol{C_x}^\top), \tag{2.8}$$

Thus, it assumes that the low rank coefficients (not the principal components) lie on a smooth manifold. This model (GLRR) is a subspace clustering model and it can be considered as an extension of PCA in a union of low-dimensional subspaces. However, the recovered codes cannot be treated as principal components or the low rank representation. This model is convex but it scales as $\mathcal{O}(In^3)$.

Graph Regularized Low-rank & Sparse Representation (GLRSR)

As GLRR can be used for subspace clustering and sparse subspace clustering is a widely used technique, the authors of [71] propose to combine low-rank and sparse subspace clustering representations. They also propose to incorporate graph regularization to the jointly low-rank and sparse codes to improve the clustering property. Their proposed optimization problem is as following:

$$\min_{\boldsymbol{C_x}} \|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C_x}\|_F^2 + \lambda \|\boldsymbol{C_x}\|_* + \beta \|\boldsymbol{C_x}\|_1 + \gamma \operatorname{tr}(\boldsymbol{C_x}\boldsymbol{L}\boldsymbol{C_x}^\top), \tag{2.9}$$

The computational complexity of this method is the same as GLRR. Hence, it does not scale well for big datasets.

2.3 Graph Regularized Tensor PCA

Tensor PCA refers to the standard MLSVD based algorithm to determine the Tucker Decomposition of a tensor, as explained in Section 1.3.1. We saw how state-of-the-art graph regularized PCA methods incorporate graph priors along the factor that corresponds to sample embedding. However, for the case of a tensor, it is non-intuitive to assign the notion of samples and features to various modes. Thus, for the case of tensors, the methods which use a graph regularization can be divided into two categories:

1. those which use graphs on all the factors of the decomposition and

2. those which use a single or multiple graphs between the tensor entities and impose regularization on one of the factors.

A handful of tensor based methods have also appeared in the past which incorporate such graph regularization. However, none of the proposed methods exactly fall under the tensor PCA framework because they do not incorporate the orthonormality constraint on the factors.

2.3.1 Regularization on all the factors

Chen et al. [72] study the tensor completion problem by incorporating graph regularization into the tensor decomposition framework as factor priors. Although, they study tensor completion, the optimization problem can be written in a general manner for a 3D tensor $\mathcal{Y}$ as following:

$$\min_{\boldsymbol{U}_r^1,\boldsymbol{U}_r^2,\boldsymbol{U}_r^3,\mathcal{Z}} \sum_{\mu} \lambda_\mu \|\boldsymbol{U}_r^\mu\|_* + \gamma \operatorname{tr}\left((\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)^\top \boldsymbol{L}(\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\right) + \beta \|\operatorname{vec}(\mathcal{Z})\|_2^2$$
$$\text{s.t } \operatorname{vec}(\mathcal{Y}) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\operatorname{vec}(\mathcal{Z}), \tag{2.10}$$

where the nuclear norm $\|\cdot\|_*$ on the factors $\boldsymbol{U}_r^\mu$ guarantees their low-rankness. The above problem is highly non-convex and computationally intensive, as for a tensor $\mathcal{Y} \in \mathbb{R}^{n\times n\times n}$, the Laplacian $\boldsymbol{L} \in \mathbb{R}^{n^3\times n^3}$ is constructed between the elements of the vectorized tensor. The construction of such a Laplacian alone scales as $\mathcal{O}(n^6)$. The optimization problem, on the contrary, scales as $\mathcal{O}(In^3)$ in every iteration due to the trace term. However, the authors propose a manifold downsampling approach which alleviates the complexity to some extent.

This class of methods assumes that the vectorized tensor is smooth on a product of manifolds in the low-dimensional space. While this assumption seems to lead to a computationally cumbersome problem, our proposed tensor methods will alleviate this problem. Also note that the problem formulation above follows the Tucker Decomposition model.

2.3.2 Regularization on one of the factors

Wang et. al [73] proposed Laplacian regularized non-negative tensor factorization method. The method does not fall under the PCA category, nevertheless, we discuss it here because it belongs to an interesting class of problems. In contrast to the above problem, where the tensor is assumed to be smooth on a product of low-dimensional manifolds, this method is more similar to the assumptions made in the GLPCA based method (discussed earlier), i.e, the embedding of tensor entities in the low-dimensional space. Each tensor is composed of a set of small tensors, each of which corresponds to a data sample. The graph based smoothness is then modeled only on the factor corresponding to this embedding. The proposed optimization problem is as follows:

$$\min_{\boldsymbol{U}_r^1,\boldsymbol{U}_r^2,\boldsymbol{U}_r^3} \left\| \mathcal{Y} - \sum_i \boldsymbol{U}_r^1(i) \otimes \boldsymbol{U}_r^2(i) \otimes \boldsymbol{U}_r^3(i) \right\|_F^2 + \gamma \operatorname{tr}(\boldsymbol{U}_r^{3\top} \boldsymbol{L} \boldsymbol{U}_r^3)$$
$$\text{s.t } \boldsymbol{U}_r^\mu \geq 0, \tag{2.11}$$

where it is assumed that the 3rd tensor dimension corresponds to the tensor data samples and the regularization is only on the factor $\boldsymbol{U}_r^3$. Furthermore, note that the above decomposition follows CP-decomposition and does not possess a unique solution.

2.3.3 Other Methods

Other state-of-the-art low-rank tensor based methods have been proposed which are simple extensions of the above methods. For example, Lui et al. [74], [75] studied the tensor geometry for gesture recognition based on Grassmann manifold, where they consider action videos as the tensor objects, and they are factorized using HOSVD. [48] and [76] proposed a discriminative low-rank tensor factorization for action classification. A supervised low-rank tensor factorization framework was proposed in [77]. Inspired by similar ideas, a manifold regularized non-negative tucker decomposition was proposed in [78].

2.4 Scalable matrix and tensor methods

Several techniques have been proposed in the past to overcome the computational cost of the matrix and tensor based methods. A primary class of such algorithms is the randomization algorithms. Randomized techniques come into play to deal with the scalability problem associated with very high dimensional data (the case of large p) [66, 79–86] using the tools of compression [87]. These works improve upon the computational complexity by reducing only the *data dimension* p but still scale in the same manner w.r.t n.

The case of large n can be tackled by using the sampling schemes accompanied with Nystrom method [88]. However, this method works efficiently only for low-rank kernel matrices and does not recover the low-rank data matrix itself. Furthermore, note that the kernel based methods are very different from graph based methods. Although, a graph adjacency matrix and a kernel matrix have close connections with each other as far as construction method is concerned, they have very different interpretations. For example, consider the case of a Gaussian weight matrix for a k_{nn}-nearest neighbor graph. The adjacency matrix $\boldsymbol{W}$ for this case is equal to a Gaussian / RBF kernel if $k_{nn} \to n$. However, when $k_{nn} \ll n$, the graph $\mathbb{G}$ and its corresponding adjacency matrix $\boldsymbol{W}$ characterize a low-dimensional manifold, whereas a dense Gaussian kernel characterizes the dot product of data samples in an infinite dimensional space. Thus, the two matrices have entirely different intuitions and methods from the kernel sampling literature cannot be used for graphs.

Finally, scalable extensions of LRR such as [89] exist but they focus only on the subspace clustering application. Recently, Aravkin et. al [90] proposed to speed-up RPCA and ease the parameter selection problem, however, the variational approach does not qualify to represent the non-linear low-rank structures.

2.5 Conclusion

State-of-the-art low-rank matrix and tensor decomposition methods tend to incorporate structure in the approximation problems by using different forms of graph regularization. This mechanism, in some cases results in an increase in the complexity of the algorithms. Furthermore, the techniques to make these methods scalable involve sampling or randomization / projection on a low-dimensional space, which cannot be used for graph based methods. Our proposed framework, as presented in the next chapters of this thesis, models the scalabilty and structure jointly and recovers low-rank linear or non-linear datasets by making use of the graph based priors only.

Part II

Proposed Low-rank Decomposition Model

3 Multilinear Low-Rank Tensors on Graphs

Given a noisy low-rank tensor $\mathscr{Y} \in \mathbb{R}^{n \times n \times n}$ or a matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, our goal is to recover the low-rank tensor $\mathscr{X}$ / matrix $\boldsymbol{X}$, i.e:

$$\mathscr{Y} = \mathscr{X}^* + \boldsymbol{\eta} \quad , \quad \boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ models noise. Similarly to the standard low-rank approximation problems, which follow a low-rank signal model, our goal is to propose a signal model and then a set of generalized approximation problems based on the signal model, regardless of the type of noise. However, in the later chapters of this thesis, we will consider applications for the case when $\boldsymbol{\eta}$ corresponds to Gaussian and Sparse / Laplacian noise.

3.1 Properties of Low-rank Signal Models

Before we begin to describe our proposed signal model, we point out here the two desired properties of a general low-rank signal model.

1. Singular vector determination, i.e, singular vectors provide a basis for each of the data modes
2. Singular value thresholding

Recall the various matrix-and-tensor-based PCA methods presented in Sections 1.2 & 1.3. Each of the methods possesses these properties. We revisit and build more on these properties here:

3.1.1 Matrix based model: PCA & RPCA

Given a noisy matrix $\boldsymbol{Y}$ and its full SVD $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, PCA approximates the low-rank matrix $\boldsymbol{X}$, with a rank r, by truncating the singular vectors $\boldsymbol{U}, \boldsymbol{V}$ and singular values $\boldsymbol{\Sigma}$.

$$\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top$$

Thus, PCA obeys the two properties, i.e, it determines the singular vectors and values, directly from those of the original data matrix $\boldsymbol{Y}$ via truncation. Note that the above solution obtained by the truncation of higher singular values and vectors is not the best one as noise might also be present on the singular values below r. Thus, we can write the low-rank matrix $\boldsymbol{X}$ as:

$$\boldsymbol{X} = \boldsymbol{X}^* + \bar{\boldsymbol{X}}^* = \boldsymbol{U}_r(\boldsymbol{\Sigma}_r^* + \bar{\boldsymbol{\Sigma}}^*)\boldsymbol{V}_r^\top = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^*\boldsymbol{V}_r^\top + \boldsymbol{U}_r\bar{\boldsymbol{\Sigma}}_r^*\boldsymbol{V}_r^\top,$$

where, $\boldsymbol{\eta} = \boldsymbol{U}_r\bar{\boldsymbol{\Sigma}}_r^*\boldsymbol{V}_r^\top + \bar{\boldsymbol{U}}_r\bar{\boldsymbol{\Sigma}}_r\bar{\boldsymbol{V}}_r^\top$, i.e, the sum of noise on the lower singular values and higher singular values.

Starting with the full SVD of $\boldsymbol{Y}$ and keeping into consideration $\sigma_1 \geq \cdots \geq \sigma_r \geq \cdots \geq \sigma_p$, we can write the SVD of $\boldsymbol{Y}$ as:

$$\boldsymbol{Y} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top + \bar{\boldsymbol{U}}_r\bar{\boldsymbol{\Sigma}}_r\bar{\boldsymbol{V}}_r^\top,$$

where $\bar{\boldsymbol{U}}_r \in \mathbb{R}^{p\times(p-r)}, \bar{\boldsymbol{V}}_r \in \mathbb{R}^{n\times(p-r)}$ is the set of singular vectors beyond r and $\bar{\boldsymbol{\Sigma}} \in \mathbb{R}^{(p-r)\times(p-r)}$ is a diagonal matrix of singular values above r. The above equation can be re-written as:

$$\boldsymbol{Y} = \boldsymbol{U}_r\boldsymbol{A} + \bar{\boldsymbol{U}}_r\bar{\boldsymbol{A}},$$

where $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ and $\bar{\boldsymbol{A}} \in \mathbb{R}^{(p-r)\times n}$. Hence, we can state the following:

$$\boldsymbol{Y}(i) \in span(\boldsymbol{U}) \;\forall\; i \quad \text{and} \quad \|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F \implies \boldsymbol{X}(i) \in span(\boldsymbol{U}_r) \;\forall\; i.$$

Thus, the columns of a low-rank matrix belong to the span of first r left singular vectors. In a similar manner one can also state the following:

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{V}_r^\top + \bar{\boldsymbol{B}}\bar{\boldsymbol{V}}_r^\top,$$

where $\boldsymbol{B} \in \mathbb{R}^{p\times r}$ and $\bar{\boldsymbol{B}} \in \mathbb{R}^{p\times(p-r)}$. This leads to:

$$\boldsymbol{Y}^\top(i) \in span(\boldsymbol{V}) \;\forall\; i \quad \text{and} \quad \|\bar{\boldsymbol{B}}\|_F \ll \|\boldsymbol{B}\|_F \implies \boldsymbol{X}^\top(i) \in span(\boldsymbol{V}_r) \;\forall\; i.$$

Thus, to summarize, low-rank signal models behind PCA can be stated as:

$$\begin{aligned}
\boldsymbol{Y} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top + \bar{\boldsymbol{U}}_r\bar{\boldsymbol{\Sigma}}_r\bar{\boldsymbol{V}}_r^\top \quad &, \quad \|\bar{\boldsymbol{\Sigma}}_r\|_F \ll \|\boldsymbol{\Sigma}_r\|_F \implies \boldsymbol{X} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top \\
\boldsymbol{Y} = \boldsymbol{U}_r\boldsymbol{A} + \bar{\boldsymbol{U}}_r\bar{\boldsymbol{A}} \quad &, \quad \|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F \implies \boldsymbol{X} = \boldsymbol{U}_r\boldsymbol{A} \\
\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{V}_r^\top + \bar{\boldsymbol{B}}\bar{\boldsymbol{V}}_r^\top \quad &, \quad \|\bar{\boldsymbol{B}}\|_F \ll \|\boldsymbol{B}\|_F \implies \boldsymbol{X} = \boldsymbol{B}\boldsymbol{V}_r^\top
\end{aligned}$$

Since Robust PCA performs an SVD in every iteration along with the soft-thresholding of singular values, it can remove noise from the lower singular values and thus has the tendency to recover $\boldsymbol{X}^*$. Thus, the signal model behind RPCA can be stated as following.

$$\boldsymbol{Y} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r^* \boldsymbol{V}_r^\top + \boldsymbol{U}_r \bar{\boldsymbol{\Sigma}}_r^* \boldsymbol{V}_r^\top + \bar{\boldsymbol{U}}_r \bar{\boldsymbol{\Sigma}}_r \bar{\boldsymbol{V}}_r^\top \quad , \quad \|\bar{\boldsymbol{\Sigma}}_r^*\|_F \ll \|\boldsymbol{\Sigma}_r^*\|_F, \|\bar{\boldsymbol{\Sigma}}_r\|_F \ll \|\boldsymbol{\Sigma}_r^*\|_F \implies \boldsymbol{X}^* = \boldsymbol{U}_r \boldsymbol{\Sigma}_r^* \boldsymbol{V}_r^\top$$

$$\boldsymbol{Y} = \boldsymbol{U}_r \boldsymbol{A}^* + \boldsymbol{U}_r \bar{\boldsymbol{A}}^* + \bar{\boldsymbol{U}}_r \bar{\boldsymbol{A}} \quad , \quad \|\bar{\boldsymbol{A}}^*\|_F \ll \|\boldsymbol{A}^*\|_F, \|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}^*\|_F \implies \boldsymbol{X}^* = \boldsymbol{U}_r \boldsymbol{A}^*$$

$$\boldsymbol{Y} = \boldsymbol{B}^* \boldsymbol{V}_r^\top + \bar{\boldsymbol{B}}^* \boldsymbol{V}_r^\top + \bar{\boldsymbol{B}} \bar{\boldsymbol{V}}_r^\top \quad , \quad \|\bar{\boldsymbol{B}}^*\|_F \ll \|\boldsymbol{B}^*\|_F, \|\bar{\boldsymbol{B}}\|_F \ll \|\boldsymbol{B}^*\|_F \implies \boldsymbol{X}^* = \boldsymbol{B}^* \boldsymbol{V}_r^\top$$

3.1.2 Tensor based models: MLSVD & TRPCA

For a noisy tensor $\mathscr{Y}$, MLSVD determines the low-rank approximation $\mathscr{X}$ via Tucker Decomposition, as described in Section 1.3.1.

$$\mathrm{vec}(\mathscr{X}) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\mathrm{vec}(\mathscr{S}).$$

Similarly to PCA, MLSVD is not an exact recovery method as it only thresholds the higher singular values, thus:

$$\mathscr{X} = \mathscr{X}^* + \bar{\mathscr{X}}^*.$$

$$\text{and} \quad \mathrm{vec}(\mathscr{X}) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\mathrm{vec}(\mathscr{S}^*) + (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\mathrm{vec}(\bar{\mathscr{S}}^*)$$

For the noisy tensor $\mathscr{Y}$, this can be written as:

$$\mathrm{vec}(\mathscr{Y}) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\mathrm{vec}(\mathscr{S}^*) + (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\,\mathrm{vec}(\bar{\mathscr{S}}^*) + (\bar{\boldsymbol{U}}_r^1 \otimes \bar{\boldsymbol{U}}_r^2 \otimes \bar{\boldsymbol{U}}_r^3)\,\mathrm{vec}(\bar{\mathscr{S}}),$$

$$\text{where} \quad \|\mathrm{vec}(\bar{\mathscr{S}})^*\|_2 \ll \|\mathrm{vec}(\mathscr{S}^*)\|_2, \|\mathrm{vec}(\bar{\mathscr{S}})\|_2 \ll \|\mathrm{vec}(\mathscr{S}^*)\|_2$$

For the Tucker Decomposition model, each of the modes $\boldsymbol{X}^\mu$ of the tensor $\mathscr{X}$ satisfies the same properties as standard PCA. Thus

1. $\boldsymbol{X}^\mu(i) \in span(\boldsymbol{U}_r^\mu)\ \forall\ i, \forall\ \mu$.
2. The singular values of a mode are determined by truncating the singular values of $\boldsymbol{Y}^\mu$ to first r values.
3. In addition, the fibres, i.e, slices of $\mathscr{S}$ also follow the property: $\|\mathscr{S}_1^\mu\|_F \geq \|\mathscr{S}_2^\mu\|_F \geq \cdots \geq \|\mathscr{S}_n^\mu\|_F,\ \forall\ \mu$. Thus, the norm of fibres along each mode has a singular value interpretation. Note that this property reduces to the ordering property of singular values for a 2D tensor.

Since TRPCA performs an SVD in every iteration for each of the tensor modes along with the soft-thresholding of singular values, the signal model behind TRPCA is similar to that behind RPCA.

3.2 Proposed Low-rank Model: Multilinear Low-rank Tensors on Graphs (MLRTG)

Inspired by the signal models governing the standard low-rank approximation problems, described in the previous section, we now present our novel model. Throughout this section, we treat a matrix as a 2D tensor and present a general model for matrices and tensors.

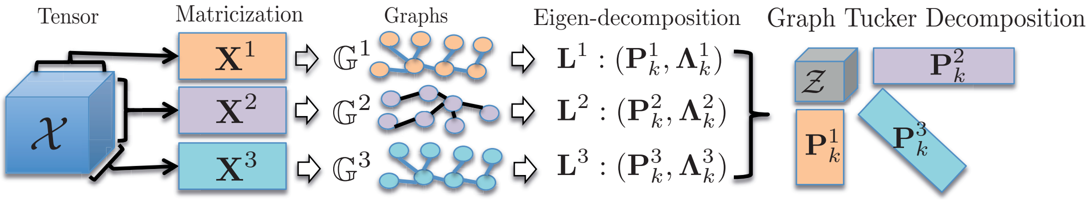


Figure 3.1 – The proposed low-rank model for tensors and matrices.

Definition 3.2.1 (**Multilinear Low-rank Tensor on Graphs**). *A tensor $\mathscr{X}^* \in \mathbb{R}^{n\times n\times n}$ is said to be Multilinear low-rank on graphs if it can be encoded as a multilinear combination of the smallest k Laplacian eigenvectors $\boldsymbol{P}_k^\mu$, where $k \ll n$, of the graphs constructed from the rows of the flattened matrices $\boldsymbol{X}^\mu$, $\forall\ \mu$. We denote the set of such tensors as $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$.*

Definition 3.2.2 (**Multilinear Graph Rank**). *A tensor $\mathscr{X}^* \in \mathbb{R}^{n\times n\times n}$ is said to have a multilinear rank (k,k,k), where $k \ll n$. The tuple (k,k,k) is known as the "Mutlilinear Graph Rank" of $\mathscr{X}^*$.*

Definition 3.2.3 (**Graph Tucker Decomposition**). *A tensor $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ can be expressed as a multilinear combination of the Laplacian eigenvectors $\boldsymbol{P}_k^\mu$ and a core tensor $\mathscr{Z}^* \in \mathbb{R}^{k\times k\times k}$ as:*

$$\operatorname{vec}(\mathscr{X}^*) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\operatorname{vec}(\mathscr{Z}^*), \tag{3.1}$$

which we call as Graph Tucker Decomposition (GTD). The core tensor $\mathscr{Z}^* \in \mathbb{R}^{k\times k\times k}$ is called the "Graph Core Tensor (GCT)".

Definitions 3.2.1 to 3.2.3 imply that for any $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ and $\boldsymbol{Z}^{*\mu} \in \mathbb{R}^{k\times k^2}$, μ^{th} matricization $\boldsymbol{X}^{*\mu}$ satisfies:

$$\boldsymbol{X}^{*\mu} = \boldsymbol{P}_k^\mu \boldsymbol{Z}^{*\mu} \boldsymbol{P}_{k^2}^{-\mu\top}, \quad \forall\ \mu,$$

where $\boldsymbol{Z}^{*\mu}$ is the μ^{th} matricization of $\mathscr{Z}^*$ and $\boldsymbol{P}_{k^2}^{-\mu} \in \mathbb{R}^{n^2\times k^2}$ is the Kronecker product of all the smallest k eigenvector matrices, except $\boldsymbol{P}_k^\mu$. The main idea of the proposed model is given in Fig. 3.1.

We mention two important points about the above definition

1. The flattened modes of a multilinear tensor can be expressed as follows:
   (a) Let $\boldsymbol{A}^{*\mu} = \boldsymbol{Z}^{*\mu}\boldsymbol{P}_{k^2}^{-\mu}$, then,

   $$\boldsymbol{X}^{*\mu} = \boldsymbol{P}_k^\mu \boldsymbol{A}^{*\mu} \implies \boldsymbol{X}^{*\mu}(i) \in span(\boldsymbol{P}_k^\mu).$$

   Thus, every column of $\boldsymbol{X}^{*\mu}$ belongs to the span of the smallest k eigenvectors of the Laplacian basis.

   (b) Let $\boldsymbol{A}^{*-\mu} = \boldsymbol{P}_k^\mu \boldsymbol{Z}^{*\mu}$, then,

   $$\boldsymbol{X}^{*\mu} = \boldsymbol{A}^{*-\mu}\boldsymbol{P}_{k^2}^{-\mu} \implies \boldsymbol{X}^{*\mu\top}(i) \in span(\boldsymbol{P}_{k^2}^{-\mu}).$$

   Thus, the rows of $\boldsymbol{X}^{*\mu}$ belong to the span of the lowest eigenvectors of the basis $\boldsymbol{P}_{k^2}^{-\mu}$.
2. Altogether, a tensor can be expressed as a multilinear encoding of the graph eigenvectors of all the modes. The GCT $\mathscr{Z}^*$ (multilinear core) has the interpretation of a very structured joint graph fourier transform of $\mathscr{X}^*$. The properties of $\mathscr{Z}^*$ will be explored later in this thesis.

The above three definitions raise a number of questions about the choice of k, its relationship with the multilinear rank (r,r,r) of the tensor, the relationship of the factors $\boldsymbol{P}_k^{\mu}$ with those of the Tucker Decomposition $\boldsymbol{U}_r^{\mu}$ and the relationship of the core tensor obtained by standard MLSVD $\mathscr{S}^* \in \mathbb{R}^{r \times r \times r}$ with GCT $\mathscr{X}^* \in \mathbb{R}^{k \times k \times k}$. While we will get back to answering each of these questions in detail in this chapter, we first present the above definitions for the case of 2D tensors, i.e, matrices.

Remark 3.2.1 (**Matrices**)**.** *Below we present simplified versions of above definitions for the case of 2D tensors, i.e, matrices.*

- *Unlike a tensor, a matrix possesses a unique rank, therefore, the notion of Multilinear Graph Rank* (k,k,k) *reduces to that of a "Graph rank k" for a matrix.*
- *The Graph Tucker Decomposition of a matrix* $\boldsymbol{X}$ *can be given as a simplified version of eq. 3.1:*

  $$\boldsymbol{X}^* = \boldsymbol{P}_k^1 \boldsymbol{Z}^* \boldsymbol{P}_k^{2\top}, \tag{3.2}$$

  where $\boldsymbol{Z}^* \in \mathbb{R}^{k \times k}$*. Note that unlike the standard SVD of* $\boldsymbol{X}^*$*, the matrix* $\boldsymbol{Z}^*$ *is not necessarily diagonal.*

3.3 Analysis of MLRTG Framework

Our MLRTG framework states that one can encode a low-rank tensor as a multilinear combination of smallest k Laplacian eigenvectors. In this section we justify the use of 'k' Laplacian eigenvectors. We perform a step-by-step analysis in the following order:

1. As a first step we reveal the underlying 'eigen-gap' assumption for the MLRTG framework. In order to define the eigen-gap assumption in terms of tensors, we introduce several other definitions, which are essential to its understanding.
2. Next, we point out two implications of this assumption for MLRTG: 1) approximate graph stationarity and 2) 'k-clusterable' nature of MLRTG in each of the modes of the tensor. We show that eigen-gap implies graph stationarity and viceversa only if the graph is dense. As we are concerned with sparse graphs, therefore, we rule out this possibility and define the notion of 'approximate stationarity' for our framework. Next, we focus on the 'k-clusterable' nature of MLRTG, which implies eigen-gap for special cases of sparse graphs. Throughout the rest of this thesis, we will rely on the existence of approximate graph stationarity and 'k-clusterable' nature of MLRTG along each of the tensor modes.
3. In the final step we summarize with a highlight of the properties of MLRTG. Fig. 3.2 presents a concise summary of our discussion in this and the next few sections of this chapter.

3.3.1 Assumption: The eigen-gap of a cartesian product graph

Our proposed MLRTG framework relies on the existence of an 'eigen-gap condition'. In order to understand this condition, we first define the notion of separable eigenvector decomposition of a graph Laplacian as follows:

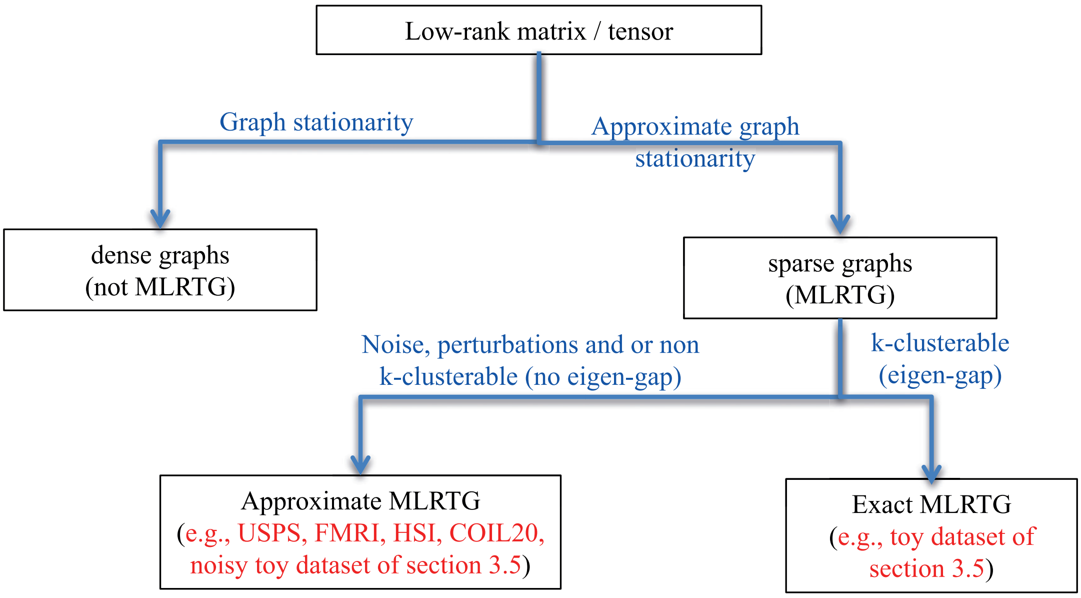

Low-rank matrix / tensor
Graph stationarity
Approximate graph stationarity
dense graphs (not MLRTG)
sparse graphs (MLRTG)
Noise, perturbations and or non k-clusterable (no eigen-gap)
k-clusterable (eigen-gap)
Approximate MLRTG (e.g., USPS, FMRI, HSI, COIL20, noisy toy dataset of section 3.5)
Exact MLRTG (e.g., toy dataset of section 3.5)

Figure 3.2 – A summary of our analysis on MLRTG.

Separable Eigenvector Decomposition of a graph Laplacian: The eigenvector decomposition of a combinatorial (unnormalized) Laplacian $\boldsymbol{L}$ of a graph $\mathbb{G}$ can be written as:

$$\boldsymbol{L} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top = \boldsymbol{P}_k\boldsymbol{\Lambda}_k\boldsymbol{P}_k^\top + \bar{\boldsymbol{P}}_k\bar{\boldsymbol{\Lambda}}_k\bar{\boldsymbol{P}}_k^\top,$$

where $\boldsymbol{P}_k \in \mathbb{R}^{n\times k}, \bar{\boldsymbol{P}}_k \in \mathbb{R}^{n\times(n-k)}, \boldsymbol{\Lambda}_k \in \mathbb{R}^{k\times k}, \bar{\boldsymbol{\Lambda}}_k \in \mathbb{R}^{(n-k)\times(n-k)}$ denote the low and high frequency eigenvectors and eigenvalues in $\boldsymbol{P}, \boldsymbol{\Lambda}$.

Definition 3.3.1 (**Eigen-gap of a graph**). *A graph Laplacian* $\boldsymbol{L}$ *with an eigenvalue decomposition* $\boldsymbol{L} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top$ *is said to have an eigen-gap if there exists a* $k > 0$*, such that* $\lambda_k \ll \lambda_{k+1}$ *and* $\lambda_k/\lambda_{k+1} \approx 0$.

Next, we guide the reader step-by-step through our analysis by defining the following terms:

- Eigen-gap of a graph
- Cartesian product of graphs
- Eigen-gap of a cartesian product graph

Cartesian Product of Graphs

Definition 3.3.2 (**Cartesian product**). *Suppose we have two graphs* $\mathbb{G}^1(\mathbb{V}^1, \mathbb{E}^1, \boldsymbol{W}^1, \boldsymbol{D}^1)$ *and* $\mathbb{G}^2(\mathbb{V}^2, \mathbb{E}^2, \boldsymbol{W}^2, \boldsymbol{D}^2)$ *where the tuple represents (vertices, edges, adjacency matrix, degree matrix). The Cartesian product* $\mathbb{G} = \mathbb{G}^1 \times \mathbb{G}^2$ *is a graph such that the vertex set is the Cartesian product* $\mathbb{V} = \mathbb{V}^1 \times \mathbb{V}^2$ *and the edges are set according to the following rules: any two vertices* (v_1^1, v_1^2) *and* (v_2^1, v_2^2) *are adjacent in* $\mathbb{G}$ *if and only if either*

- $v_1^1 = v_2^1$ *and* v_1^2 *is adjacent with* v_2^2 *in* $\mathbb{G}^2$

- $v_1^2 = v_2^2$ *and* v_1^1 *is adjacent with* v_2^1 *in* $\mathbb{G}^1$.

The adjacency matrix of the Cartesian product graph $\mathbb{G}$ *is given by the matrix Cartesian product:*

$$\boldsymbol{W} = \boldsymbol{W}^1 \times \boldsymbol{W}^2 = \boldsymbol{W}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{W}^2$$

Lemma 3.3.1. *The degree matrix* $\boldsymbol{D}$ *of the graph* $\mathbb{G} = \mathbb{G}^1 \times \mathbb{G}^2$ *which satisfies definition 3.3.2 is given as*

$$\boldsymbol{D} = \boldsymbol{D}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{D}^2 = \boldsymbol{D}^1 \times \boldsymbol{D}^2$$

The ***combinatorial*** *Laplacian of* $\mathbb{G}$ *is:*

$$\boldsymbol{L} = \boldsymbol{L}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{L}^2 = \boldsymbol{L}^1 \times \boldsymbol{L}^2$$

Proof. For the proof please see Appendix A.1. □

Now, for our purpose we define the eigen-gap of the Cartesian Product Graph as follows:

Definition 3.3.3 (Eigen-gap of a Cartesian Product Graph). *A cartesian product graph as defined in 3.3.2, is said to have an eigen-gap if there exist k, such that,* $\max\{\lambda_k^1, \lambda_k^2\} \ll \min\{\lambda_{k+1}^1, \lambda_{k+1}^2\}$, *where* λ_k^μ *denotes the* k^{th} *eigenvalue of the* μ^{th} *graph Laplacian* $\boldsymbol{L}^\mu$.

Consequence of 'Eigen-gap assumption': Separable eigenvector decomposition of a cartesian product graph

The eigen gap assumption (definition 3.3.3) is important to define the notion of the 'Separable eigenvector decomposition of a cartesian product graph'. We define the eigenvector decomposition of a Laplacian of cartesian product of two graphs. The definition can be extended in a straight-forward manner for more than two graphs as well. For the sake of generalization of this notion, we assume here that $n_1 \neq n_2$ and $k_1 \neq k_2$.

Lemma 3.3.2 (Separable Eigenvector Decomposition of a Cartesian Product Graph). *For a cartesian product graph as defined in 3.3.2, the eigenvector decomposition can be written in a separable form as:*

$$\boldsymbol{L} = (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{\Lambda}^1 \times \boldsymbol{\Lambda}^2)(\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)^\top = \boldsymbol{P}_k \boldsymbol{\Lambda}_k \boldsymbol{P}_k^\top + \bar{\boldsymbol{P}}_k \bar{\boldsymbol{\Lambda}}_k \bar{\boldsymbol{P}}_k^\top = \boldsymbol{P} \boldsymbol{\Lambda} \boldsymbol{P}^\top,$$

where $\boldsymbol{P}_k \in \mathbb{R}^{n_1 n_2 \times k_1 k_2}, \bar{\boldsymbol{P}}_k \in \mathbb{R}^{n_1 n_2 \times (n_1-k_1)(n_2-k_2)}, \boldsymbol{\Lambda}_k \in \mathbb{R}^{k_1 k_2 \times k_1 k_2}, \bar{\boldsymbol{\Lambda}}_k \in \mathbb{R}^{(n_1-k_1)(n_2-k_2) \times (n_1-k_1)(n_2-k_2)}$ *denote the low and high frequency eigenvectors and eigenvalues in* $\boldsymbol{P}, \boldsymbol{\Lambda}$.

Proof. Please refer to Appendix A.2. □

3.3.2 Eigen-gap implies 'Approximate Graph Stationarity' & 'k-clusterable' nature of MLRTG

Graph Stationarity

The notion of graph stationarity was introduced in [91]. This an extension of the classical notion of the stationarity of signals to the signals defined on the nodes of a graph $\mathbb{G}$. To keep our exposition simple, we introduce stationarity in an informal manner and then state one definition from [91] which directly relates to our work. First we state two conditions for the stationarity of a graph signal $\boldsymbol{X} \in \mathbb{R}^{p\times n}$:

1. The first moment is constant over the vertex set $\mathbb{V}$.

2. The covariance matrix is invariant to localization of the vertices of the graph.

The second condition can be explained in a more simpler manner in terms of the Fourier interpretation. Recall that for the notion of time stationarity of a random process, the second condition above is equivalent to stating that the frequencies of the random process are uncorrelated, i.e:

$$Cov(FFT(\boldsymbol{X})) = E[\boldsymbol{F}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{F}] = |\boldsymbol{F}^\top \boldsymbol{\Omega}_X \boldsymbol{F}| = \boldsymbol{\Gamma}_t,$$

where $\boldsymbol{F}$ is the discrete Fourier matrix and $\boldsymbol{\Gamma}_t$ is a diagonal matrix. The matrix $\boldsymbol{\Gamma}_t$ is called the 'Spectral Covariance' of $\boldsymbol{X}$. For the case of graph signal $\boldsymbol{X}$, the above condition can be stated in terms of Graph Fourier Transform (GFT) as follows:

$$Cov(GFT(\boldsymbol{X})) = E[\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{P}] = |\boldsymbol{P}^\top \boldsymbol{\Omega} \boldsymbol{P}| = \boldsymbol{\Gamma},$$

where $\boldsymbol{\Gamma}$ is called the 'Graph Spectral Covariance' matrix [91]. In simple words, $\boldsymbol{\Omega}$ is diagonalizable by the eigenvectors of the graph Laplacian $\boldsymbol{L}$.

Now let $\boldsymbol{\Omega}^\mu \in \mathbb{R}^{n\times n}$ be the sample column covariance of $\boldsymbol{X}^\mu$, the μ^{th} matricization of $\mathscr{X}$, then we define the **Graph Spectral Covariance matrix** (GSC) $\boldsymbol{\Gamma}^\mu$ for $\boldsymbol{X}^\mu$, as [91]:

$$\boldsymbol{\Gamma}^\mu = \boldsymbol{P}^{\mu\top} \boldsymbol{\Omega}^\mu \boldsymbol{P}^\mu \tag{3.3}$$

From [91] we also know that for a Graph Wide Sense Stationary (GWSS) process:

$$s(\boldsymbol{\Gamma}^\mu) = \frac{\|\operatorname{diag}(\boldsymbol{\Gamma}^\mu)\|_F}{\|\boldsymbol{\Gamma}^\mu\|_F} = 1, \tag{3.4}$$

which we call as *GSC alignment order*. In fact, the more close is $s_r(\boldsymbol{\Gamma}^\mu)$ to 1, the more stationary is the process on the graph $\mathbb{G}^\mu$. Thus, for our purposes we use the above definition of GWSS which is equivalent to the original definition in [91].

Graph stationarity for a low-rank matrix implies eigen-gap only if the graph is dense

So far we discussed the notion of graph stationarity for a general matrix $\boldsymbol{X}$. However, our primary interest is to study low-rank matrices. We now show that 'graph stationarity' for a low-rank matrix $\boldsymbol{X}$ implies the existence of eigen-gap assumption only if the graph $\mathbb{G}$ is dense.

Consider a low-rank matrix $\boldsymbol{X}$ with a rank r. Let the SVD of $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top$, then the empirical column covariance $\boldsymbol{\Omega}$ can be given as:

$$\boldsymbol{\Omega} = \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r^2 \boldsymbol{U}_r^\top.$$

Now, graph stationarity for this matrix implies:

$$\boldsymbol{P}^\top \boldsymbol{U}_r \boldsymbol{\Sigma}_r^2 \boldsymbol{U}_r^\top \boldsymbol{P} = \boldsymbol{\Gamma},$$

where $\boldsymbol{\Gamma}$ is a diagonal matrix. This implies that

$$\boldsymbol{U}_r \boldsymbol{\Sigma}_r^2 \boldsymbol{U}_r^\top = \boldsymbol{P}\boldsymbol{\Gamma}\boldsymbol{P}^\top.$$

Note that the L.H.S of this expression is the covariance $\boldsymbol{\Omega}$ and the R.H.S is a function of the Laplacian $g(\boldsymbol{L})$, i.e.,

$$\boldsymbol{\Omega} = g(\boldsymbol{L}).$$

As $\boldsymbol{P} = \boldsymbol{U}_r$, the only possibility is that $\boldsymbol{\Gamma} = \boldsymbol{\Sigma}^2$. Thus,

$$g(\boldsymbol{L}) = \boldsymbol{P}\boldsymbol{\Gamma}\boldsymbol{P}^\top = \boldsymbol{P} g(\boldsymbol{\Lambda})\boldsymbol{P}^\top = \boldsymbol{P}[g(\boldsymbol{\Lambda}_r)|g(\bar{\boldsymbol{\Lambda}}_r)]\boldsymbol{P}^\top,$$

where $\boldsymbol{\Lambda}_r$ denotes the first r eigenvalues and $\bar{\boldsymbol{\Lambda}}_r$ denotes the complement / higher eigenvalues above r. Assuming the graph eigenvalues $\boldsymbol{\Lambda}$ are sorted in the increasing order, we have,

$$g(\boldsymbol{\Lambda}_r) = (\boldsymbol{\Sigma}_r^{-2})^{-1}.$$

To define the complete set of eigenvalues $\boldsymbol{\Lambda}$, we append a small constant $\epsilon \ll 1$ and $\epsilon \ll \sigma_r$ to the diagonal entries above r in the matrix $\boldsymbol{\Sigma}$. Thus,

$$g(\boldsymbol{\Lambda}) = (\boldsymbol{\Sigma}^{-2})^{-1} \implies g = (\cdot)^{-1} \quad \text{and} \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-2}.$$

The above expression clearly implies that the graph eigenvalues are given as the squared inverse of the covariance singular values. This implies

$$\boldsymbol{L} = (\boldsymbol{\Omega} + \epsilon \boldsymbol{I})^{-1}.$$

As $\sigma_{r+1} \ll \sigma_r$, this implies $\lambda_r \ll \lambda_{r+1}$, which is the eigen-gap assumption. However, note that the graph computed by inverting a covariance matrix is not sparse.

Thus, for a low-rank matrix, the existence of graph stationarity implies an eigen-gap and vice versa only if the graph loses its sparsity property. We point out here explicitly that sparsity is one of the key properties of graphs for modeling locally smooth signals [92]. Therefore, one cannot afford to lose this property in a graph. Recently the work of Pavez et. al [93] has shown the connections between a generalized Laplacian and precision matrices. They also present a sparse graph Learning framework as a precision matrix learning estimation problem.

Consequence: Approximate Graph Stationarity

In practice, a signal is never perfectly stationary on the graph for the following reasons:

- For a signal to be perfectly stationary on graphs, the singular vectors of its covariance are equivalent to the eigenvectors of the graph. This is never the case in practice as a k_{nn}-nearest neighbor graph is computed differently from a covariance matrix.
- For a low-rank matrix, graph stationarity can only be true if the graph is not sparse. This counters

the basic local smoothness assumption in graph signal processing framework.

Nevertheless, the concept of graph stationarity is of vital importance for MLRTG framework. Given the definition of MLRTG, we can relax the strict stationarity assumption and introduce the notion of approximate stationarity.

Definition 3.3.4 (**Approximate Graph Stationarity for a matrix**)**.** *A matrix $\boldsymbol{X}^{\mu} \in \mathbb{R}^{n \times n^2}$ satisfies approximate graph stationarity if its Graph Spectral Covariance alignment order (eq.* (3.4)*) is less than 1, i.e,*

$$s(\boldsymbol{\Gamma}^{\mu}) = \frac{\|\operatorname{diag}(\boldsymbol{\Gamma}^{\mu})\|_F}{\|\boldsymbol{\Gamma}^{\mu}\|_F} < 1, \tag{3.5}$$

Approximate graph stationarity & eigen-gap of sparse graph imply a k-clusterable MLRTG

So far we have argued that for a general low-rank matrix, the satisfaction of graph stationarity implies eigen-gap and vice-versa only if the graph is dense. Since a dense graph is not meaningful for our purpose, we relax the stationarity to 'approximate graph stationarity'. However, the question, i.e., under what conditions does a sparse graph possesses an eigen-gap and the corresponding low-rank matrix possesses approximate graph stationarity remains unanswered. In this section we discuss the k-clusterable property of MLRTG across each of its modes, which leads to an eigen-gap in a sparse graph. Our discussion here is intuitive and based on the conclusions from the work of Belkin et. al. [2].

We know from the basic spectral graph theory that multiplicity of the zero eigenvalues of a graph Laplacian represents the number of connected components / communities in the graph. The eigenvectors corresponding to the zero eigenvalues are the cluster indicators. Let us consider the example of a data matrix $\boldsymbol{X}$ which is k-clusterable across its rows and columns, where $k \ll n$. Furthermore, for the simplicity assume that each cluster has q samples, thus the matrix $\boldsymbol{X} \in \mathbb{R}^{kq \times kq}$. The matrix $\boldsymbol{X}$ has a block structure, i.e., each of the $k \times k$ blocks have a constant value. While such a matrix may seem hypothetical, we will show in the next section that it is possible to construct it with the desirable properties. If k_{nn}-nearest neighbor graphs, using Gaussian kernel as a choice for the weight matrices are now constructed among the rows and columns of this matrix, using $k_{nn} \approx q$, then:

$$\boldsymbol{W}(i,j) \approx 1 \quad \text{if } \boldsymbol{X}(i) \in \mathcal{N}(\boldsymbol{X}(j)) \text{ and } \boldsymbol{X}(i), \boldsymbol{X}(j) \in C_l,$$

i.e., the edge between the nodes i and j carry a weight approximately equal to 1 if the two nodes are in the same cluster and i is among the nearest neighbors of j. Arguing in a similar way, no edges will exist between inter-cluster nodes. This would result in a graph with a block diagonal Laplacian structure, such that each of the blocks is dense with similar weights. Recalling our Linear Algebra and Numerical Analysis knowledge, it is possible to say that the eigen-value decomposition of a block diagonal Laplacian, with constant blocks would result in a step function in the eigenvalues, where the step occurs from the k^{th} to $(k+1)^{st}$ eigenvalue. The eigenvectors corresponding to the zero eigenvalues then represent the cluster indicators. The same holds for the graph across other modes as well. This line of argument is similar to the NCut algorithm, where one would like to cut the nodes of a graph into distinct clusters. Note that for this case, the first k-eigenvectors will exactly encode the MLRTG. However, there will be a minimum correlation between the graph eigenvectors and the covariance ones, which would result in an approximate graph stationarity (alignment order < 1) due to the interpretation of the eigenvectors as cluster indicators. Thus, even though approximate stationarity holds, the entire power will still be concentrated in the top $k \times k$ corner of the GSC Γ. We call this as 'low-frequency power concentration'.

To conclude, for a k_{nn}-nearest neighbors graph constructed from a k-clusterable data $\boldsymbol{X}^{\mu}$ one can expect $\lambda^{\mu}_{k}/\lambda^{\mu}_{k+1} \approx 0$ as $\lambda^{\mu}_{k} \approx 0$ and $\lambda^{\mu}_{k} \ll \lambda^{\mu}_{k+1}$. Such a data matrix $\boldsymbol{X}^{\mu}$ is approximately graph stationary and has all the power concentrated in the top $k \times k$ corner of GSC (low frequencies).

3.3.3 Properties of Multilinear Low-rank Tensors on Graphs (MLRTG)

Now that we have studied all the concepts behind MLRTG in detail, we are ready to summarize the properties of MLRTG. First note that the approximate graph stationarity was studied only in the case of a matrix or a specific mode of the tensor. For an MLRTG, we would want this property to hold jointly for all the modes of the tensor. Secondly, it is obvious to note that for an MLRTG, most of the energy of the GSC would be concentrated in the top $k \times k$ entries. Having said that, we are ready to state the properties of MLRTG explicitly.

1. **Property 1: Joint Approximate Graph Stationarity:**

   Definition 3.3.5. *A tensor $\mathscr{X}^{*} \in \mathbb{MLT}(\boldsymbol{P}^{\mu}_{k})$, whose cartesian product graph possesses an eigen-gap (definition 3.3.3), satisfies Joint Approximate Graph Stationarity, i.e, its μ^{th} matricization / flattening $\boldsymbol{X}^{*\mu} \in \mathbb{R}^{n \times n^2}$ satisfies approximate graph stationarity (definition 3.3.4) $\forall, \mu = 1, 2, 3$.*

2. **Property 2: Low Frequency Power Concentration:**

   Definition 3.3.6. *A tensor $\mathscr{X}^{*} \in \mathbb{MLT}(\boldsymbol{P}^{\mu}_{k})$ satisfies low-frequency power concentration property, i.e, power is concentrated in the top entries of the graph spectral covariance matrices $\boldsymbol{\Gamma}^{\mu}, \forall \mu = 1, 2, \cdots, d$.*

$$\hat{s}(\boldsymbol{\Gamma}^{\mu}) = \frac{\|\boldsymbol{\Gamma}^{\mu}(1:k, 1:k)\|_F}{\|\boldsymbol{\Gamma}^{\mu}\|_F} = 1 \tag{3.6}$$

Property 1 above implies automatically that an MLRTG should be simultaneously k-clusterable across all the modes $\boldsymbol{X}^{*\mu}$ of the tensor $\mathscr{X}^{*}$. Intuitively, property 2 makes sense because if $\boldsymbol{X}^{*\mu}$ is low-rank then the covariance $\boldsymbol{\Omega}^{\mu}$ is also low-rank and a diagonalization of $\boldsymbol{\Omega}^{\mu}$ with the complete set of graph eigenvectors $\boldsymbol{P}^{\mu}$ would result in a matrix $\boldsymbol{\Gamma}^{\mu} \in \mathbb{R}^{n \times n}$ which will be zero for all the rows and columns above k.

Summary of the analysis: "MLRTG: Approximate Stationarity in Low-Frequency"

In simple words, a tensor is said to be multilinear low-rank on the graphs constructed from its flattened modes if it can be encoded as a multlinear combination of the first few eigenvectors of the respective graph Laplacians.

It is trivial to note that an exact stationarity assumption means that all the power of GSC is concentrated on the diagonal entries. This in turn implies that the eigenvectors of the Covariance are diagonalizable by those of the Laplacian matrix or the eigenvectors of the two are equivalent. For a matrix $\boldsymbol{X}^{*\mu} \in \mathbb{R}^{n \times n^2}$, the covariance of rows is given as $\boldsymbol{\Omega}^{\mu} \in \mathbb{R}^{n \times n}$ and the left singular vectors of $\boldsymbol{X}^{\mu}$ are the same as the eigenvectors of $\boldsymbol{\Omega}^{\mu}$ (up to a certain numerical precision). Thus, an exact stationarity assumption imposes that the SVD of $\boldsymbol{X}^{\mu}$ gives the eigenvectors of the Laplacian $\boldsymbol{L}^{\mu}$ which never holds. Our framework on the other hand, although prefers most of the energy to be concentrated on the diagonal of GSC, does not necessitate its need. Thus, we allow the power to spill out of the diagonal of the GSC.

This forms the basis of a recovery method for such tensors and the novel tensor decompositions we will introduce in the next chapter, so we define it as following:

Definition 3.3.7 (**Approximate Stationarity in Low-Frequency**)**.** *A tensor $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ is said to be approximately stationary in the lower graph frequencies if it satisfies properties 1 and 2 stated above.*

3.4 Approximate MLRTG

The MLRTG framework relies on computing the graphs $\mathbb{G}^\mu$ from the flattened matrices of $\mathscr{X}^*$. In practice one never has access to $\mathscr{X}^*$ itself, but a noisy version $\mathscr{Y}$ and the graphs $\mathbb{G}^\mu$ have to be constructed from $\boldsymbol{Y}^\mu$. Furthermore, MLRTG requires the eigen-gap condition (definition 3.3.3) to hold. For a k_{nn}-graph constructed from the modes of a noisy tensor, the eigen-gap condition does not hold. It might also be the case, as it is for many real datasets that the data is clusterable and the eigen-gap does not exist. However, such a scenario arises solely due to the choice of a graph construction method, the parameter k_{nn} and the distance metric used.

In order to understand the intuition, we refer again to the example of a hypothetical block low-rank and k-clusterable matrix $\boldsymbol{X}$, its Laplacian, eigenvectors and values from the previous section. If the weights in each of the blocks of the Laplacian are varied, which might result from small perturbations or noise in the data blocks, then the higher eigenvalues show variation. Similarly, if some edges are added between the different graph communities, then the number of zero eigenvalues decrease until a certain point where only one eigenvalue is zero. This occurs when the graph has only one connected component with weakly connected communities. The lower eigenvalues also show a variation until the eigen-gap vanishes. In such a case, when the data is weakly clusterable, one can perform clustering on the first few eigenvectors (excluding the one with zero eigenvalue). These eigenvectors represent the low-dimensional embedding of the data on a non-linear manifold which is characterized by the graph. This is the concept behind the Laplacian Eigenmaps algorithm [2].

An interesting concept that we will verify later in our experiments is that for the case of weakly connected communities, the lowest Laplacian eigenvectors, due to their interpretation as a low-dimensional embedding possess a greater correlation with the eigenvectors of the covariance which results in an increased graph stationarity and a higher alignment order $s(\boldsymbol{\Gamma})$. Thus, they provide more information then just the cluster indication, in contrast to the case of disconnected communities. However, the choice of k becomes much more cumbersome due to the non-existence of eigen-gap. Our experiments will show that greater the k, the better is the low-rank representation. This implies a reduced low-frequency power concentration in the GSC as compared to the case of disconnected communities.

Throughout, the rest of this thesis we will assume that the data is k-clusterable across all the tensor modes. Even when the eigen-gap does not exist, the graph stationarity and low-frequency power concentration properties would hold approximately. The power spilling out of the top corner of GSC can be attributed to the non-existence of the eigen-gap condition and contributes to the approximation error of MLRTG which will studied theoretically in the later chapters of this thesis. Thus, we are now ready to present the following lemma.

Lemma 3.4.1. *For any $\mathscr{Y} = \mathscr{X}^* + \eta \in \mathbb{R}^{n \times n \times n}$, where $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$, η models the noise and errors, the μ^{th} matricization $\boldsymbol{Y}^\mu$ of $\mathscr{Y}$ satisfies*

$$\boldsymbol{Y}^\mu = \boldsymbol{P}_k^\mu \boldsymbol{Z}^{*\mu} \boldsymbol{P}_{k^2}^{-\mu\top} + \bar{\boldsymbol{P}}_k^\mu \bar{\boldsymbol{Z}}^\mu \boldsymbol{P}_{k^2}^{\bar{-}\mu\top} + \eta$$

Now, we can define the notion of "Approximate MLRTG" which corresponds to a more realistic scenario for the real world applications.

Definition 3.4.1 (Approximate MLRTG). *A tensor* $\mathscr{Y} = \mathscr{X}^* + \eta \in \mathbb{R}^{n\times n\times n}$*, where* $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^{\mu})$*,* η *models the noise and errors and* $\boldsymbol{Y}^{\mu} = \boldsymbol{X}^{*\mu} + \eta = \boldsymbol{P}_k^{\mu}\boldsymbol{Z}^{*\mu}\boldsymbol{P}_{k^2}^{-\mu\top} + \bar{\boldsymbol{P}}_k^{\mu}\bar{\boldsymbol{Z}}^{*\mu}\boldsymbol{P}_{k^2}^{\bar{-}\mu\top} + \eta$*, is said to be approximate MLRTG if*

$$\left\|\boldsymbol{X}^{*\mu} - \boldsymbol{P}_k^{\mu}\boldsymbol{Z}^{*\mu}\boldsymbol{P}_{k^2}^{-\mu\top}\right\|_F^2 \ll \epsilon \quad\Longrightarrow\quad \|\bar{\boldsymbol{Z}}^{\mu}\|_F^2 \ll \|\boldsymbol{Z}^{*\mu}\|_F^2.$$

Properties of Approximate MLRTG

Approximate MLRTG satisfies the following properties:

1. its μ^{th} matricization $\boldsymbol{Y}^{\mu} \in \mathbb{R}^{n\times n^2}$ satisfies approximate graph stationarity (definition 3.3.4) $\forall\,\mu = 1,2,3$ and

2. the *approximate* low-frequency power concentration, i.e, most of the power is concentrated in the top entries of the graph spectral covariance matrices $\boldsymbol{\Gamma}^{\mu}, \forall \mu = 1,2,3$.

$$\hat{s}(\boldsymbol{\Gamma}^{\mu}) = \frac{\|\boldsymbol{\Gamma}^{\mu}(1:k,1:k)\|_F^2}{\|\boldsymbol{\Gamma}^{\mu}\|_F^2} \approx 1 \tag{3.7}$$

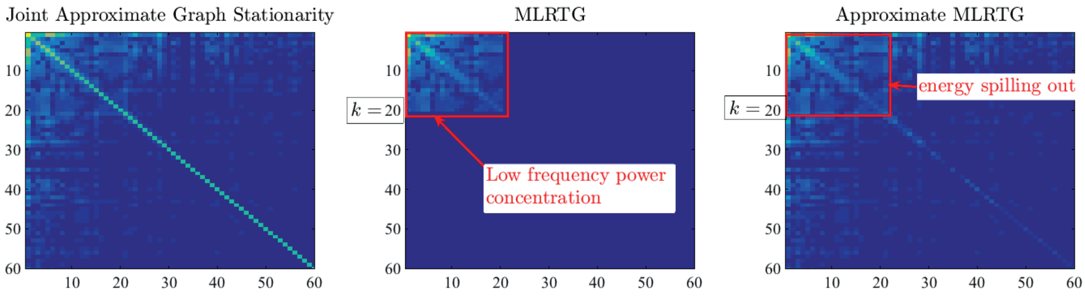


Figure 3.3 – Examples of GSC for 1) joint approximate graph stationarity (left), 2) MLRTG (middle) and 3) approximate MLRTG (right). The difference between the three concepts is illustrated in terms of the concentration of energy in the specific regions of the GSC.

Summary: GSC for MLRTG & Approximate MLRTG

Fig. 3.3 shows examples of GSC matrices for 1) joint approximate graph stationarity (left), 2) MLRTG (middle) and 3) Approximate MLRTG (right). As clear from the figure, Joint approximate graph stationarity corresponds to the concentration of energy on the diagonal of the GSC with some energy spilling out of the diagonal. This condition together with the low frequency power concentration in the first $k = 20$ eigenvectors (middle plot) constitutes the concept of MLRTG. However, MLRTG requires the existence of eigen-gap, as evident from the jump in the energy concentration. This assumption is unrealistic for the graphs constructed from many real world datasets, therefore, some energy spills out of the red box. This gives rise to the notion of approximate MLRTG (the rightmost plot).

Comments

The reader might note that the whole framework relies on the existence of the eigen-gap condition (definition 3.3.3). Such a notion does not necessarily exist for the real data, however, we clarify a few important things right away. The existence of eigen-gap is not strict for our framework, thus, practically, it performs reasonably well for a broad range of applications even when such a gap does not exist. Nevertheless, in the next section we will present several examples of artificial datasets which possess an eigen-gap. 2) We introduce this notion to study the theoretical side of our framework and characterize the approximation error. Experimentally, we will show that choosing a specific number of eigenvectors of the graph Laplacians, without knowing the eigen-gap is good enough for our setup. Thus, at no point throughout this thesis we will present a way to compute this gap. A summary of the properties of exact and approximate MLRTG are presented in Table 3.2.

3.5 Examples of Exact and Approximate MLRTG

In this section we quote several examples of exact and approximate MLRTG to support the theoretical analysis and properties described in the previous sections. While a simple low-rank matrix or tensor qualifies for an approximate MLRTG irrespective of its clusterable nature, one can also generate an approximate MLRTG by introducing noise, perturbations (pose variation, small rotation) to a jointly k-clusterable data across all the modes, as summarized in Fig. 3.2. Our examples consist of both artificial and real matrices and tensors with and without noise and perturbations. Based on Fig. 3.2, we present several ways to construct exact and approximate artificial matrices first. Then, using a few of these methods, we present and analyze a few datasets (Table 3.2).

3.5.1 Methods to generate MLRTG

Method 1: Exact MLRTG

Let $\mathbf{1}_{50}$ be a 50×50 matrix of all ones, then we generate the artificial data matrix $\boldsymbol{Y} \in \mathbb{R}^{200\times 200}$ as follows:

$$\boldsymbol{Y} = \begin{bmatrix} 7\mathbf{1}_{50} & 5\mathbf{1}_{50} & 3\mathbf{1}_{50} & \mathbf{1}_{50} \\ 5\mathbf{1}_{50} & 7\mathbf{1}_{50} & \mathbf{1}_{50} & 3\mathbf{1}_{50} \\ 3\mathbf{1}_{50} & \mathbf{1}_{50} & 7\mathbf{1}_{50} & 5\mathbf{1}_{50} \\ \mathbf{1}_{50} & 3\mathbf{1}_{50} & 5\mathbf{1}_{50} & 7\mathbf{1}_{50} \end{bmatrix},$$

where $\boldsymbol{\eta}$ models a small amount of Gaussian noise. Although matrix $\boldsymbol{Y}$ is symmetric, it does not have to possess this property for our analysis. The above matrix is low-rank, clusterable into 4 clusters across its rows and columns, approximately graph stationarity, possesses the low-frequency power concentration property and the k_{nn}-graphs possess eigen-gaps.

Method 2: Approximate MLRTG: Noise in k-clusterable MLRTG

The simplest method to construct an approximate MLRTG $\boldsymbol{Y}$ is to add noise in the matrix obtained from above method. Depending on the level of noise, $\boldsymbol{Y}$ is approximately low-rank, might retain or lose its k-clusterable property (eigen-gap), more graph stationary than $\boldsymbol{Y}$ obtained via method 1 and possesses approximate low-frequency power concentration property.

Method 3: Approximate MLRTG: From graph eigenvectors

The third method is to construct a tensor directly from the graph eigenvectors. We describe the procedure for a 2D tensor in detail below:

1. Generate a random Gaussian matrix, $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$.
2. Construct a k_{nn} graph $\mathbb{G}^1$ between the rows of $\boldsymbol{Y}$ and compute the combinatorial Laplacian $\boldsymbol{L}^1$.
3. Construct a k_{nn} graph $\mathbb{G}^2$ between the rows of $\boldsymbol{Y}^\top$ and compute the combinatorial Laplacian $\boldsymbol{L}^2$.
4. Compute the first k (where $k \ll n$) eigenvectors and eigenvalues of $\boldsymbol{L}^1$ and $\boldsymbol{L}^2$, $(\boldsymbol{P}^1_k, \boldsymbol{\Lambda}^1_k), (\boldsymbol{P}^2_k, \boldsymbol{\Lambda}^2_k)$.
5. Generate a random matrix of the size $\boldsymbol{X} \in \mathbb{R}^{k\times k}$.
6. Generate the low-rank artificial tensor by using $\text{vec}(\boldsymbol{Y}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k)\,\text{vec}(\boldsymbol{X})$, where $\otimes$ denotes the kronecker product.

Method 4: Graph filtering of a randomly generated tensor

The final method is by filtering a randomly generated tensor with the k_{nn} combinatorial Laplacians constructed from the flattened modes of the tensor. We describe the procedure for a 2D tensor below:

1. Follow steps 1 to 3 of method 3 above.
2. Generate low-rank Laplacians from the eigenvectors: $\hat{\boldsymbol{L}}^1 = \boldsymbol{P}^1_k \boldsymbol{I}_{k\times k} {\boldsymbol{P}^1}^\top_k$ and $\hat{\boldsymbol{L}}^2 = \boldsymbol{P}^2_k \boldsymbol{I}_{k\times k} {\boldsymbol{P}^\top}^2_k$.
3. Filter the matrix $\boldsymbol{Y}$ with these Laplacians to get the low-rank matrix $\boldsymbol{Y}^* = \hat{\boldsymbol{L}}^1 \boldsymbol{Y} \hat{\boldsymbol{L}}^2$.

The tensors produced by the above two methods are low-rank, approximately graph stationary and possess approximate low-frequency power concentration property. However, such tensors do not possess eigen-gaps (not clusterable). Since, the tensors are either generated from Laplacian eigenvectors or by filtering with graphs, the GSC alignment order is high (graph stationarity). Table 3.1 summarizes the properties of the artificial tensors generated by various methods.

It is imperative to point out here that the artificial tensors generated via methods 3 and 4 resemble more closely to the real world MLRTG, as it will be clear from our real world examples of the USPS, FMRI, Hyperspectral Image and COIL20 tensor examples in this section. Such tensors are not necessarily k-clusterable along all the modes and are only approximately low-rank due to noise and perturbations and the k_{nn}-nearest neighbors graphs constructed from them do not possess an eigen-gap. Even though our artificial data examples in this section are based on the tensors generated by methods 1 and 2, in the later chapters of this thesis, when we design the approximation problems, we will present experiments on the tensors generated by methods 3 and 4 along with a plethora of real world datasets. Such datasets do not follow the data model and will pose the necessary challenges to study the true power of our framework and quantify the approximation error.

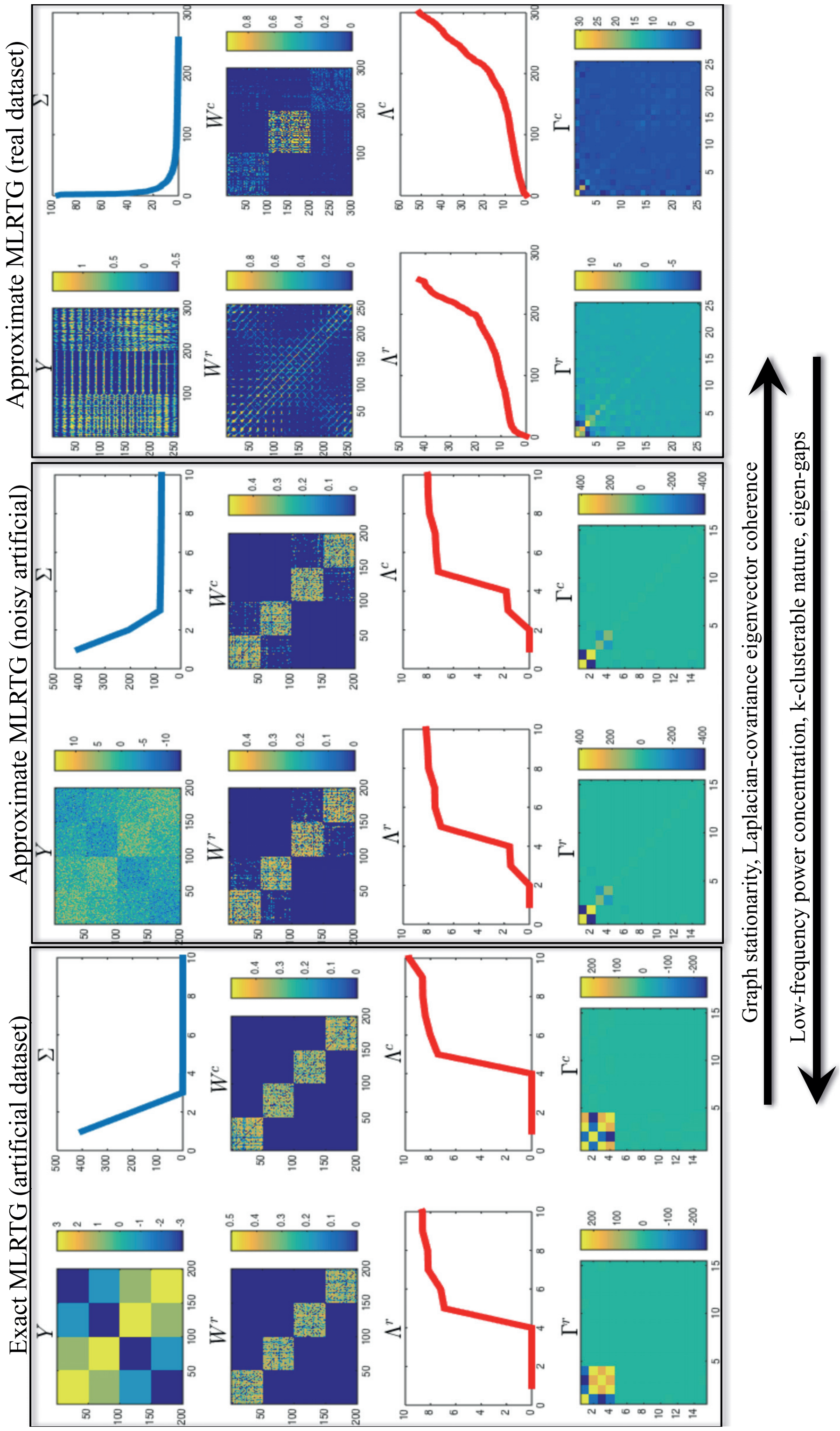


Figure 3.4 – A summary of the variation of properties from exact to approximate MLRTG.

Table 3.1 – Properties of exact and approximate artificial and real MLRTG.

Properties	**Exact MLRTG**	**Approximate MLRTG**		
	method 1	method 2	method 3	method 4
graph stationarity	minimum	medium	high	high
eigenvectors coherence	minimum	medium	high	high
power concentration	exact	high	medium	medium
eigen-gap	✓	✓	×	×
k-clusterable	✓	✓	×	×

Table 3.2 – Example artificial and real exact and approximate MLRTG.

Ingredients	**Exact MLRTG**	**Approximate MLRTG**		
data type	artificial (method 1)	artificial (method 2)	real (USPS)	real (FMRI, HSI, COIL20)
artificial noise	×	✓	×	×
perturbations	×	×	✓	✓
clusterable modes	✓	✓	columns only	×
matrix / tensor	matrix	matrix	matrix	tensor
low-rank	✓	approx.	approx.	approx. multilinear

3.5.2 From exact to approximate MLRTG

Having presented the methods to generate artificial tensors, we are ready to discuss a few examples now. Our examples comprise of two artificial MLRTG, one exact and the other approximate, generated via methods 1 and 2. Furthermore, we also present examples of 4 real datasets, including USPS hand-written digits, FMRI, Hyperspectral Image and COIL20 tensors. The ingredients of each of the tensors in our examples are stated in Table 3.2. Note that the USPS dataset is an interesting example of approximate MLRTG, as only one of its modes is clusterable.

Our goal here is to verify the properties of MLRTG and for that we need to study the following: 1) the low-rankness of the tensor in terms of the singular values of its modes, 2) the clusterable nature, which can be verified by looking at the graph eigenvalues and the community structure of the adjacency matrices and 3) the stationarity and low-frequency power concentration properties which can be verified by visualizing the GSC matrices. We should be able to study how these properties vary from exact to approximate MLRTG, therefore, to study this effect, we focus on the matrices generated via methods 1 and 2 and the USPS dataset of hand-written digits. For each of the three matrices, we construct k_{nn}-graphs $\mathbb{G}^r$ and $\mathbb{G}^c$ along the rows and columns of $\boldsymbol{Y}$, where $k_{nn} = 40$. Next we compute the eigenvectors $\boldsymbol{P}^r, \boldsymbol{P}^c$ and eigenvalues $\boldsymbol{\Lambda}^r, \boldsymbol{\Lambda}^c$ of the Laplacians $\boldsymbol{L}^r, \boldsymbol{L}^c$. Fig. 3.4 presents a summary of our analysis for each of the three datasets (exact on the left, approximate artificial in the middle and approximate real on the right) in terms of the visualization of $\boldsymbol{Y}$, the singular values $\boldsymbol{\Sigma}$, adjacency matrices $\boldsymbol{W}$, graph eigenvalues $\boldsymbol{\Lambda}_k$ and GSC matrices $\boldsymbol{\Gamma}$ for each of the modes.

It is clear to see that as one goes from left to right (exact to approximate MLRTG), the community structure of the MLRTG disrupts, which results in a loss of eigen-gaps of the row and column graphs. The leftmost column shows the case of a purely clusterable data along all the modes with a clear community structure and hence the eigen-gaps of the Laplacians. Since the approximate MLRTG in the middle is just obtained by adding noise to the clusterable matrix $\boldsymbol{Y}$ (method 2), the community structure disrupts slightly. This is evident from the intra-community edges in the adjacency matrices and the decrease in the number of zero eigen-values. The right-most column presents the case of

USPS hand-written dataset, where 100 images (16×16) of each of the digits $0, 1, 2$ are vectorized and stacked in the columns to form the matrix $\boldsymbol{Y} \in \mathbb{R}^{256 \times 300}$. Clearly, USPS has clusterable nature only along the samples (columns), which is obvious from the adjacency matrices. Furthermore, note that the community structure is not pure (no eigen-gap). Even though their is no artificial noise in the dataset, the digits have perturbations, such as small rotations and scaling effect. This results in a Laplacian spectrum which does not possess an eigen-gap.

The fourth row of Fig. 3.4 shows the variation of GSC matrices $\boldsymbol{\Gamma}^r$ and $\boldsymbol{\Gamma}^c$ from exact to approximate MLRTG and it is straight-forward to verify that $\hat{s}(\boldsymbol{\Gamma}^r) = \hat{s}(\boldsymbol{\Gamma}^c) = 1$ for the exact MLRTG and this reduces as we move towards the approximate MLRTG. Due to the existence of an eigen-gap for exact MLRTG, the first k eigenvectors are cluster indicators, therefore, they have minimum coherence with those of the covariance, consequently a lower GSC alignment order $s(\boldsymbol{\Gamma})$ (approximate graph stationarity). This coherence goes on increasing (increasing graph stationarity), as the datasets lose their clustering property. Thus more power is concenrated on the diagonal of the GSC for approximate MLRTG. Finally, note that due to the non-existence of eigen-gap, the choice of k is cumbersome for the approximate MLRTG. We will study this in detail in the next chapters of this thesis.

3.5.3 Real tensors

Finally, we quote three real world tensor examples for different applications to show that such a notion exists in practice, almost always for a low-rank matrix as well as a tensor.

Our three examples consists of: 1) The COIL-20 dataset, the database of objects with pose variations. Each image of this dataset has the dimension 128×128. We arrange the 72 images corresponding to the first object in the form of a cube, hence, $\mathcal{Y} \in \mathbb{R}^{128 \times 128 \times 72}$. Next, we flatten $\mathcal{Y}$ across each of the modes, $\boldsymbol{Y}^1 \in \mathbb{R}^{128 \times 9216}, \boldsymbol{Y}^2 \in \mathbb{R}^{128 \times 9216}, \boldsymbol{Y}^3 \in \mathbb{R}^{72 \times 16384}$ and construct k_{nn}-nearest neighbors graphs between the rows of $\boldsymbol{Y}^1, \boldsymbol{Y}^2, \boldsymbol{Y}^3$, with $k_{nn} = 10$. The combinatorial Laplacians of the three graphs are denoted as $\boldsymbol{L}^1 \in \mathbb{R}^{128 \times 128}, \boldsymbol{L}^2 \in \mathbb{R}^{128 \times 128}, \boldsymbol{L}^3 \in \mathbb{R}^{72 \times 72}$. 2) The Hyperspectral face dataset obtained from the SCIEN [94], $\mathcal{Y} \in \mathbb{R}^{542 \times 333 \times 148}$, where 542 is the y-dimension, 333 is the x-dimension and 148 is the number of channels in the spectrum. We flatten this tensor along all the three modes and construct graphs in the same way as the COIL20 dataset. 3) The resting state and movie watching FMRI dataset of one subject $\boldsymbol{Y} \in \mathbb{R}^{1232 \times 88}$ obtained from [95], [96] with 1232 time points with alternating movie watching and resting intervals and 88 ROIs. A Short Time Fourier Transform with a Hamming window of size 32 time points with a 50% overlap and 128 point DFT was used to convert this dataset into a 3D tensor $\mathcal{Y} \in \mathbb{R}^{129 \times 79 \times 88}$ (frequency, time, ROIs). Although it is non-standard to study FMRI in a time-frequency domain, out goal here was only to increase the dimension of the tensor in a meaningful manner. We perform the same steps for this datasets as well. We do not introduce artificial noise in these tensors but they possess natural perturbations (Table 3.2).

In order to demonstrate that each of the above datasets is MLRTG, we compute the eigenvector decomposition of the Laplacians and plot in Fig. 3.5, from left to right, the original tensor, SVD of flattened modes, Graph Spectral Covariance (GSC) $\boldsymbol{\Gamma}^1, \boldsymbol{\Gamma}^2, \boldsymbol{\Gamma}^3$ (3.3) and its low-frequency power concentration $\hat{s}(\boldsymbol{\Gamma}^1), \hat{s}(\boldsymbol{\Gamma}^2), \hat{s}(\boldsymbol{\Gamma}^3)$ (3.5) for increasing values of k from 1 to 50 (as the exact value of k is unknown). Each row of this figure corresponds to a different dataset. Clearly, all the three tensors are at least approximately low-rank in each of their modes as seen from their respective SVDs. It is interesting to note the same from GSC and power concentration plots which show that most of the energy of the low-rank modes is concentrated in the first few eigenvectors. Thus, it is straight-forward to encode these tensors as a multilinear combination of a few Laplacian eigenvectors.

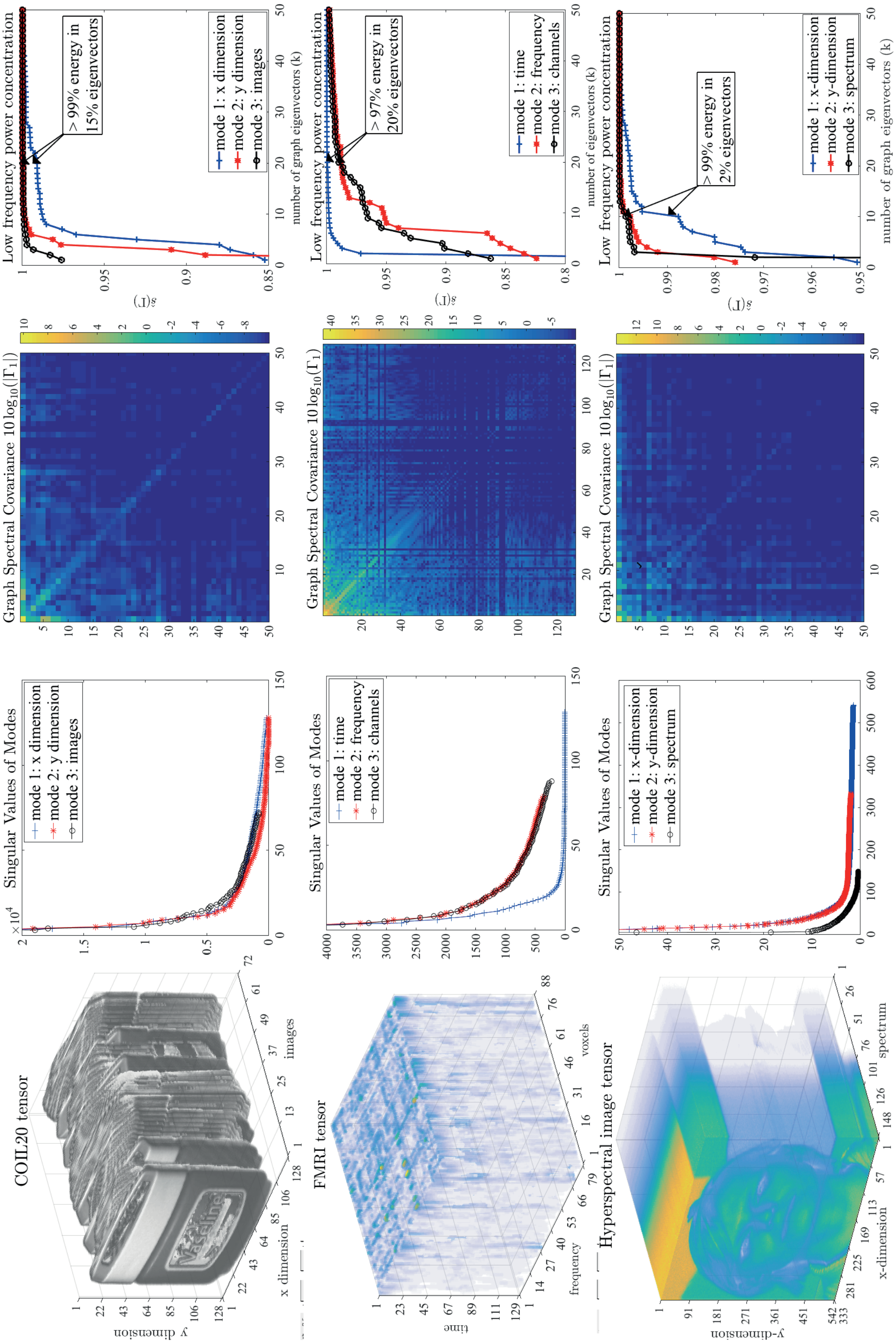


Figure 3.5 – Examples of approximate multilinear low-rank tensors on graphs. Each row shows the full tensor, the SVD of each of its flattened modes, graph spectral covariance (eq. (3.4)) of mode 1 $\mathbf{\Gamma}^1$ and the low-rank power concentration $\hat{s}(\mathbf{\Gamma})$ (eq. (3.6)) for each of the modes, from left to right. Clearly, the concentration of energy in the top corner of GSC shows that each of these tensors is multilinear low-rank on graphs.

3.6 Understanding MLRTG

Our signal model, MLRTG is a highly structured low-rank signal model which can characterize a low-rank matrix in terms of graph eigenvectors. As we will see in the later chapters of this thesis, it is possible to use this model to formulate a number of problems for low-rank recovery of matrices and tensors in a scalable manner. While, we get back to applications in part III of this thesis, we will conclude this chapter by discussing a number of interpretations of MLRTG. This is essential and important to the understanding of MLRTG in simpler manner. Throughout this section, to keep our exposition simple, we will focus on matrix based examples. For a noisy low-rank matrix

$$\boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta},$$

MLRTG can be written as:

$$\boldsymbol{Y} = \boldsymbol{P}^1_k \boldsymbol{Z}^* {\boldsymbol{P}^2}^\top_k + \boldsymbol{\eta},$$

where $\boldsymbol{Z}^* \in \mathbb{R}^{k\times k}$. Below we try to understand the significance of $\boldsymbol{Z}$ and the information provided by Laplacian eigenvectors $\boldsymbol{P}^\mu_k$ in detail. First, we understand what does it mean to construct graphs along each of the modes of a matrix / tensor and the information that it provides.

In order to make this more concrete, we take a simple example of digit 3 from the traditional USPS dataset. We vectorize all the images and form a data matrix $\boldsymbol{Y}$, whose columns consist of different samples of digit 3 from the USPS dataset.

3.6.1 The graph of features provides a basis for data

In order to motivate the need of the graph of features we build the 10 nearest neighbors graph (of features), i.e, a graph between the rows of $\boldsymbol{Y}$ using the FLANN. Fig. 3.6 shows the eigenvectors of the Laplacian denoted by $\boldsymbol{P}^1$. We observe that they have a 3-like shape. In Fig. 3.6, we also plot the eigenvectors associated to the empirical covariance matrix. The empirical covariance matrix is computed as

$$\boldsymbol{\Omega} = \frac{\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}^\top}{n},$$

where n is the number of samples and

$$\tilde{\boldsymbol{Y}} = \boldsymbol{Y} - \mu_{\boldsymbol{Y}} \quad \text{for} \quad \mu_{\boldsymbol{Y}} = \frac{1}{np}\sum_{i=1}^{p}\sum_{j=1}^{n} \boldsymbol{Y}(i,j).$$

This definition is motivated in [91]. We observe that both sets of eigenvectors are similar. This is confirmed by computing the GSC matrix:

$$\boldsymbol{\Gamma} = {\boldsymbol{P}^1}^\top \boldsymbol{\Omega} \boldsymbol{P}^1$$

In order to measure the level of alignment between the orthogonal basis $\boldsymbol{P}^1$ and the one behind $\boldsymbol{C}$, we use the GSC alignment ratio:

$$s_r(\boldsymbol{\Gamma}) = \left(\frac{\sum_\ell \boldsymbol{\Gamma}^2_{\ell,\ell}}{\sum_{\ell_1}\sum_{\ell_2}\boldsymbol{\Gamma}^2_{\ell_1,\ell_2}}\right)^{\frac{1}{2}} = \frac{\|\mathrm{diag}(\boldsymbol{\Gamma})\|_2}{\|\boldsymbol{\Gamma}\|_F}.$$

When the two bases are aligned, the covariance matrix $\boldsymbol{\Omega}$ and the graph Laplacian $\boldsymbol{L}$ are simultaneously diagonalizable, giving a ratio equal to 1. On the contrary, when the bases are not aligned, the ratio is close to $\frac{1}{p}$, where p is the dimension of the dataset. Note that an alternative would be to compute directly the inner product between $\boldsymbol{P}$ and the eigenvectors of $\boldsymbol{\Omega}$. However, using $\boldsymbol{\Gamma}$ we implicitly weight the eigenvectors of $\boldsymbol{\Omega}$ according to their importance given by their corresponding eigenvalues.

In the special case of the digit 3, we obtain a ratio $s_r(\boldsymbol{\Gamma}) = 0.97$, meaning that the main covariance eigenvectors are well aligned with the graph eigenvectors. Fig. 3.6 shows a few eigenvectors of both sets and the matrix $\boldsymbol{\Gamma}$. This effect is in accordance with our previous analysis and has been studied in [91] where the definition of stationary signals on graphs is proposed. A similar idea is also the motivation of the Laplacianfaces algorithm [97]. A closer look at the bottom figure of Fig. 3.6 shows that most of the energy in the diagonal is concentrated in the first few entries. This shows that the first few eigenvectors of the Laplacian are more aligned with the eigenvectors of the covariance matrix. This phenomena implies that digit 3 of the USPS dataset is MLRTG, i.e, only the first few eigenvectors (corresponding to the low eigenvalues) are enough to serve as the features for this dataset.

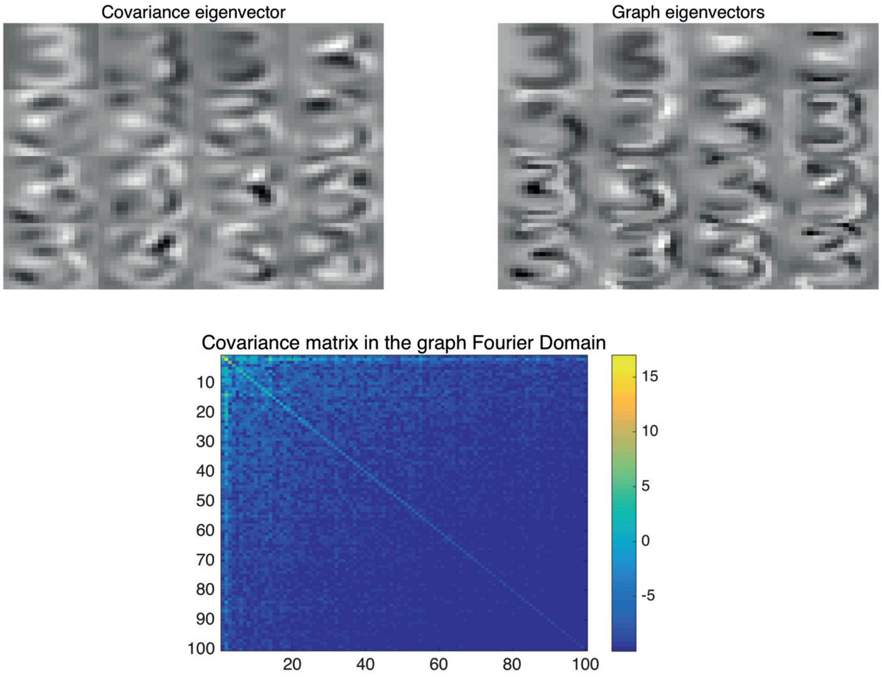


Figure 3.6 – Studying the number 3 of USPS. Left: Covariance eigenvectors associated with the 16 highest eigenvalues. Right: Laplacian eigenvectors associated to the 16 smallest non-zero eigenvalues. Because of stationarity, Laplacian eigenvectors are similar to the covariance eigenvectors. Bottom: $\boldsymbol{\Gamma} = \boldsymbol{P}^{1\top}\boldsymbol{C}\boldsymbol{P}^{1}$ in dB. Note the diagonal shape of the matrix implying that $\boldsymbol{P}^{1}$ is aligned with the eigenvectors of $\boldsymbol{C}$.

Let us analyze how the graph eigenvectors evolve when all digits are taken into account. Fig. 3.7 shows the Laplacian and covariance eigenvectors for the full USPS dataset. Again we observe some alignment: $s_r(\boldsymbol{\Gamma}) = 0.82$.

From this example, we can conclude that every column of a low-rank matrix $\boldsymbol{X}$ lies approximately in the span of the eigenvectors $\boldsymbol{P}^{1}$ of the features graph, where k denotes the eigenvectors corresponding

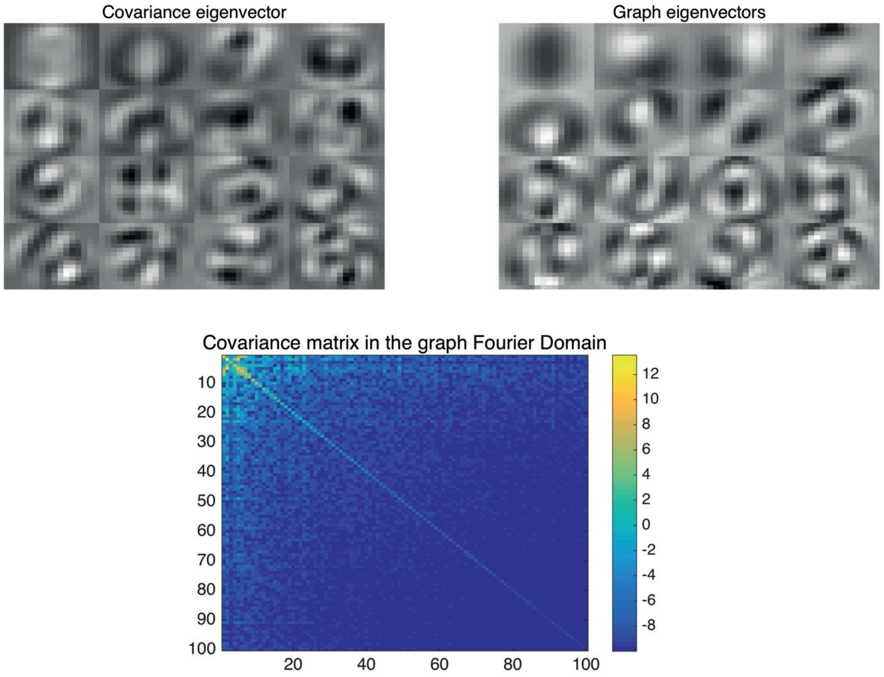


Figure 3.7 – Studying the full USPS dataset. Left: Covariance eigenvectors associated with the 16 highest eigenvalues. Right: Laplacian eigenvectors associated to the 16 smallest non-zero eigenvalues. Because of stationarity, Laplacian eigenvectors are similar to the covariance eigenvectors. Bottom: $\boldsymbol{\Gamma} = \boldsymbol{P}^{1\top}\boldsymbol{\Omega}\boldsymbol{P}^1$ in dB. Note the diagonal shape of the matrix implying that $\boldsymbol{P}^1$ is aligned with the eigenvectors of $\boldsymbol{\Omega}$.

to the smallest k eigenvalues. This is similar to PCA, where a low-rank matrix is represented in the span of the first few principal directions or atoms of the basis. Alternately, the Laplacian eigenvectors are meaningful features for the USPS dataset. Let the eigenvectors $\boldsymbol{P}^1$ of $\boldsymbol{L}^1$ be divided into two sets $(\boldsymbol{P}^1_k \in \mathbb{R}^{p\times k}, \bar{\boldsymbol{P}}^1_k \in \mathbb{R}^{p\times(p-k)})$. Note that the columns of $\boldsymbol{P}^1_k$ contain the eigenvectors corresponding to the low graph frequencies and $\bar{\boldsymbol{P}}^1_k$ contains those corresponding to higher graph frequencies. Then we can write, $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{\eta} = \boldsymbol{X}^* + \bar{\boldsymbol{X}}^* + \boldsymbol{\eta}$, where $\boldsymbol{X}^*$ is the low-rank part and $\boldsymbol{\eta}$ models the noise or corruptions. Thus,

$$\boldsymbol{X} = \boldsymbol{P}^1_k\boldsymbol{A}^* + \bar{\boldsymbol{P}}^1_k\bar{\boldsymbol{A}}^* \quad \text{and}$$

$$\boldsymbol{X}^* = \boldsymbol{P}^1_k\boldsymbol{A}^*$$

where $\boldsymbol{A}^* \in \mathbb{R}^{k\times n}$ and $\bar{\boldsymbol{A}}^* \in \mathbb{R}^{(p-k)\times n}$. From Fig. 3.6 it is also clear that $\|\bar{\boldsymbol{P}}^1_k\bar{\boldsymbol{A}}^*\|_F \ll \|\boldsymbol{P}^1_k\boldsymbol{A}^*\|_F$ for a specific value of k.

3.6.2 The graph of samples provides embedding for data

The smallest eigenvectors of the graph of samples provides an embedding of the data in the low-dimensional space [2]. This has a similar interpretation as the principal components in PCA. We argue that every row of a low-rank matrix lies in the span of the first few eigenvectors of the graph of samples. This is similar to representing every row of the low-rank matrix as the span of the principal components. Thus, the graph of samples $\boldsymbol{L}^2$ encodes a smooth non-linear map towards the principal components of the underlying manifold defined by the graph $\boldsymbol{L}^2$. This is the heart of many algorithms in clustering [98] and dimensionality reduction [2]. In the context of our MLRTG framework, this implies that if the data has a class structure, the graph of samples will enforce the low-rank matrix $\boldsymbol{X}$ to benefit from this class structure. This results in an enhanced clustering of the low-rank signals in a low-dimensional space. Let the eigenvectors $\boldsymbol{P}^2$ of $\boldsymbol{L}^2$ be divided into two sets $(\boldsymbol{P}^2_k \in \mathbb{R}^{n\times k}, \bar{\boldsymbol{P}}^2_k \in \mathbb{R}^{n\times(n-k)})$, where k denotes the eigenvectors in $\boldsymbol{P}^2$ corresponding to the smallest k eigenvalues. Note that the columns of $\boldsymbol{P}^2_k$ contain the eigenvectors corresponding to the low graph frequencies and $\bar{\boldsymbol{P}}^2_k$ contains those corresponding to higher graph frequencies. Then, we write:

$$\boldsymbol{X} = \boldsymbol{B}^* {\boldsymbol{P}^2}^\top_k + \bar{\boldsymbol{B}}^* \bar{\boldsymbol{P}^2}^\top_k \text{ and}$$

$$\boldsymbol{X}^* = \boldsymbol{B}^* {\boldsymbol{P}^2}^\top_k$$

where $\boldsymbol{B}^* \in \mathbb{R}^{p\times k}$ and $\bar{\boldsymbol{B}}^* \in \mathbb{R}^{p\times(n-k)}$. As argued in the previous subsection, $\|\bar{\boldsymbol{B}}^* \bar{\boldsymbol{P}^2}^\top_k\|_F \ll \|\boldsymbol{B}^* {\boldsymbol{P}^2}^\top_k\|_F$.

Thus, MLRTG can be seen as a signal model that encodes multilinear relationship between the graph of features and samples for a low-rank data matrix, in a low-dimensional space.

3.6.3 Multi-dimensional embedding in a space of product of manifolds

Finally, we argue that the eigenvectors of the graph of features and samples jointly provide a multi-dimensional embedding of the matrix or tensor $\boldsymbol{X}$ in a low-dimensional space. To understand this, we first re-write our model in a vectorized form as following:

$$\text{vec}(\boldsymbol{X}^*) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k)\,\text{vec}(\boldsymbol{Z}^*)^\top,$$

where vec(·) denotes the vectorization operation and ⊗ denotes the Kronecker product of the Laplacian eigenvector matrices.

Recall from our previous analysis, that the matrix $\boldsymbol{P}_k = \boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k$ is the matrix of the eigenvectors of the cartesian product of graphs $\mathbb{G}^1$ and $\mathbb{G}^2$. Thus, encoding a matrix $\boldsymbol{X}$ as a multilinear combination of k graph eigenvectors is equivalent to saying that the low-rank signal vec($\boldsymbol{X}$) belongs to the span of the first k^2 Laplacian eigenvectors of a manifold that is the product of row and column manifolds of the matrix. Thus, the matrix $\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \in \mathbb{R}^{n^2\times k^2}$ is a multi-dimensional embedding of $\boldsymbol{X}^*$ in a k^2 dimensional space.

3.6.4 Graph Core Tensor: The Joint Graph Fourier Transform

Having described the importance of constructing graphs across each of the modes of the matrix / tensor, we now move on to explain the role of Graph Core Tensor (GCT).

For a signal $\boldsymbol{x}$ lying on the nodes of a graph $\mathbb{G}$, the Graph Fourier Transform (GFT) is given as $\hat{\boldsymbol{x}} = \boldsymbol{P}^\top \boldsymbol{x}$. Starting from our MLRTG framework,

$$\boldsymbol{X}^* = \boldsymbol{P}^1_k \boldsymbol{Z}^* {\boldsymbol{P}^2}^\top_k,$$

we can re-write it as:

$$\boldsymbol{Z}^* = {\boldsymbol{P}^1}^\top_k \boldsymbol{X}^* \boldsymbol{P}^2_k.$$

The above equation is equivalent to:

$$\text{vec}(\boldsymbol{Z}^*) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k)^\top\,\text{vec}(\boldsymbol{X}^*).$$

Thus $\boldsymbol{Z}^*$ corresponds to the joint GFT of the low-rank matrix $\boldsymbol{X}^*$ w.r.t to the first k eigenvectors only.

3.7 Conclusion

We present a novel low-rank signal model, ‘Multilinear Low-rank Tensors on Graphs’ (MLRTG) which states that a low-rank tensor can be encoded as a multilinear combination of the graph eigenvectors of each of its modes. The multilinear combination is called the Graph Core Tensor (GCT) which has the interpretation of a joint k-fourier transform and encodes all the energy of MLRTG. Any MLRTG satisfies approximate graph stationarity and low-frequency power concentration properties and can be generated from a k-clusertable data along all of its modes which possesses an eigen-gap in the graph constructed from its modes. Many real world low-rank matrices and tensors satisfy the approximate MLRTG criterion, i.e., an approximate low-frequency power concentration property. The proposed framework has the interpretation of providing an embedding of a high dimensional low-rank tensor on a product of low-dimensional manifolds and that of a joint Graph Fourier Transform. A plethora of low-rank approximation problems will be defined based on our proposed model in the next few chapters of this thesis.

4 From graph basis to PCA basis: Graph Multilinear SVD

In Chapter 3, we introduced our novel low-rank signal model for matrices and tensors, MLRTG. Any low-rank matrix or tensor is MLRTG if it can be encoded as a multilinear combination of the first few Laplacian eigenvectors of the graphs constructed between the rows of its modes. We saw that MLRTG is a highly structured signal model and has a rich interpretation as multi-dimensional embedding and joint Graph Fourier Transform. As pointed out in Chapter 1, one of the two main goals of this thesis is to propose scalable and approximate methods for recovering low-rank matrices and tensors and the other goal is to understand the relationship between linear and non-linear embedding of low-rank data in a low-dimensional space. We now argue that both of these goals are inter-related.

Why is it important to understand the connection between linear and non-linear embedding? In the previous chapter we saw that the first few eigenvectors of row and column graphs provide embedding of the data in row and column spaces respectively. We have already pointed out that linear subspace based algorithms, i.e, PCA and its variants, also determine row and column subspaces. The only difference is that graph eigenvectors do so in a non-linear space and PCA in a linear. Since our goal is to propose scalable iterative algorithms for low-rank recovery, one of the possible ways to do this is to fix the basis upfront and refine it iteratively via low-cost operations. Thus, one can avoid expensive SVD operations on the full data matrix in every iteration of the algorithm. Obviously, it is possible to use graph eigenvectors as a basis for rows and columns of the data, but the question of how to refine this non-linear embedding to get a linear one remains unanswered. Thus, our goal in this chapter is to understand this connection. Once, we understand how to determine a linear low-dimensional embedding from a non-linear one, we would be just one step away from defining a plethora of approximation problems for scalable recovery of low-rank matrices and tensors, using graphs.

In this chapter we understand the connection between graph and PCA basis and develop an algorithm, “Graph Multilinear SVD" to compute latter from former.

Recall that for a low-rank tensor

$$\mathcal{Y} = \mathcal{X}^* + \boldsymbol{\eta},$$

where $\mathcal{X}^* \in \mathbb{R}^{n \times n \times n}$ has a multilinear rank (r, r, r) and $\boldsymbol{\eta}$ models noise, its Tucker decomposition can be written as:

$$\operatorname{vec}(\mathcal{X}^*) = (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^2 \otimes \boldsymbol{U}_r^3)\operatorname{vec}(\mathcal{S}^*),$$

where $\boldsymbol{U}_r^\mu \in \mathbb{R}^{n \times r}$ and $\mathcal{S}^* \in \mathbb{R}^{r \times r \times r}$. The standard method to obtain this decomposition is via an

Alternating Least Squares (ALS) method, which is also known as Higher Order SVD or Multilinear SVD (MLSVD) [11]. The ALS method iteratively computes the SVD of every mode of $\mathscr{Y}$ until the fit $\|\operatorname{vec}(\mathscr{Y}) - (\boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^1 \otimes \boldsymbol{U}_r^1)\operatorname{vec}(\mathscr{S})\|_2^2$ stops to improve. This method costs $\mathscr{O}(nr^2)$ per iteration for rank r.

For a tensor $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$, the Graph Tucker Decomposition as proposed in Chapter 3, with a graph multilinear rank (k, k, k) can be written as:

$$\operatorname{vec}(\mathscr{X}^*) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\operatorname{vec}(\mathscr{Z}^*),$$

where $\boldsymbol{P}_k^\mu \in \mathbb{R}^{n \times k}$ are the graph eigenvectors and $\mathscr{Z} \in \mathbb{R}^{k \times k \times k}$ is the Graph Core Tensor (GCT).

Note that our GTD is quite similar in structure to the standard Tucker Decomposition. This similarity in structure raises important questions about the connection between $\mathscr{S}$ and $\mathscr{Z}$, how are the basis $\boldsymbol{U}_r^\mu$ related to the graph eigenvectors $\boldsymbol{P}_k^\mu$ and the relationship between multilinear and graph multilinear ranks r and k. In the next section, we will define the Graph Multilinear SVD method which provides answers to all these questions.

4.1 Graph Multilinear SVD (GMLSVD)

We are now ready to present a graph based multilinear SVD method which will reveal the relationship between core tensor $\mathscr{S}$ and GCT $\mathscr{Z}$ and the vectors $\boldsymbol{U}_r^\mu$ and $\boldsymbol{P}_k^\mu$. First we recall, the following assumptions for any $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ from Chapter 3.

- There exists an eigen-gap for every $\boldsymbol{L}^\mu$, $\forall\ \mu$.
- There exists an eigen-gap in the cartesian product graph $\boldsymbol{L} = \boldsymbol{L}^1 \times \boldsymbol{L}^2 \times \boldsymbol{L}^3$.

In simple words, the above assumptions guarantee that any $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ can be exactly encoded as a multilinear combination of graph eigenvectors $\boldsymbol{P}_k^\mu$. In other words,

$$\operatorname{vec}(\mathscr{Z}^*) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)^\top \operatorname{vec}(\mathscr{X}^*),$$

is a lossless compression of $\mathscr{X}^*$ into $\boldsymbol{P}_k^\mu$ and $\mathscr{Z}^*$. We will get back to this compression in detail later in this chapter. It is possible to derive our GMLSVD algorithm in the following three different ways:

1. Derivation from the ‘Low-frequency power concentration property’ of MLRTG. This is an intuitive derivation and our exposition here is based on an example MLRTG and its stationarity property.
2. Derivation from the MLSVD algorithm, which is a purely mathematical and algorithmic approach.
3. Derivation from an iterative tensor based approximation problem.

We will discuss the first two methods in this Chapter, while the third method will be discussed in Chapter 7.

4.1.1 Derivation from Low-frequency power concentration property of MLRTG

Recall the low-frequency power concentration property of approximate MLRTG from Chapter 3, which states that most of the power is concentrated in the top $k \times k$ corner of the GSC matrix for every mode of the tensor. Further recall several examples of approximate MLRTG from Chapter 3. We will re-visit the example of the FMRI tensor here and derive our GMLSVD algorithm by performing an analysis on the GSC of this tensor. Fig. 4.1 shows the GSC of mode 1 (time) and the power concentration plot of the FMRI tensor. The power concentration curve for mode 1 (blue line) shows that more than 97% power is concentrated in the first 20 eigenvectors of this mode. This corresponds to the top 20×20 corner of the GSC, in Fig. 4.1.

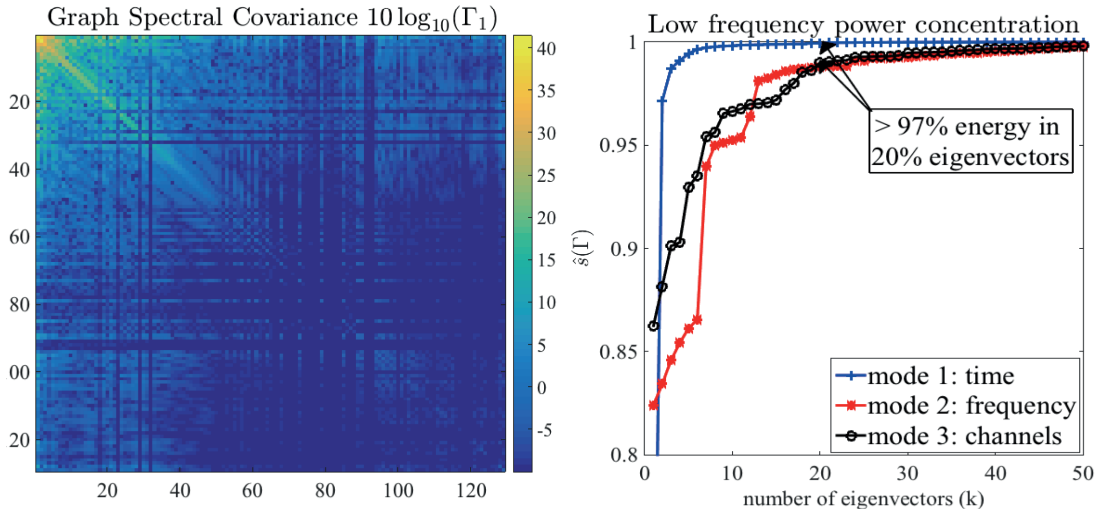


Figure 4.1 – GSC of the mode 1 (time) of the FMRI tensor.

Starting from the Graph Spectral Covariance (GSC), using covariance $\boldsymbol{\Omega}^1 = \boldsymbol{X}^1 \boldsymbol{X}^{1^\top}$ along mode 1 and the eigen decomposition of the covariance $\boldsymbol{\Omega}^1$, we get:

$$\boldsymbol{\Gamma}^1 = \boldsymbol{P}^{1^\top} \boldsymbol{U}^1 \boldsymbol{\Sigma}^2 \boldsymbol{U}^{1^\top} \boldsymbol{P}^1.$$

Dropping the superscript '1' for the ease of notation and using $\boldsymbol{U} = [\boldsymbol{U}_k | \bar{\boldsymbol{U}}_k]$, $\boldsymbol{P} = [\boldsymbol{P}_k | \bar{\boldsymbol{P}}_k]$, and

$$\boldsymbol{\Sigma}^2 = \begin{bmatrix} \boldsymbol{\Sigma}_k^2 & 0 \\ 0 & \bar{\boldsymbol{\Sigma}}_k^2 \end{bmatrix},$$

we get:

$$\boldsymbol{\Gamma} = [\boldsymbol{P}_k | \bar{\boldsymbol{P}}_k]^\top [\boldsymbol{U}_k | \bar{\boldsymbol{U}}_k] \begin{bmatrix} \boldsymbol{\Sigma}_k^2 & 0 \\ 0 & \bar{\boldsymbol{\Sigma}}_k^2 \end{bmatrix} [\boldsymbol{U}_k | \bar{\boldsymbol{U}}_k]^\top [\boldsymbol{P}_k | \bar{\boldsymbol{P}}_k].$$

Now assuming that the matrix $\boldsymbol{X}$ is exactly rank k, then the covariance $\boldsymbol{\Omega}$ is also rank k and $\bar{\boldsymbol{\Sigma}}_k^2 = 0$. Using this, the expression can be expanded as below:

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{P}_k^\top \boldsymbol{U}_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}_k^\top \boldsymbol{P}_k & \boldsymbol{P}_k^\top \boldsymbol{U}_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}_k^\top \bar{\boldsymbol{P}}_k \\ \bar{\boldsymbol{P}}_k^\top \boldsymbol{U}_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}_k^\top \boldsymbol{P}_k & \bar{\boldsymbol{P}}_k^\top \boldsymbol{U}_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}_k^\top \bar{\boldsymbol{P}}_k \end{bmatrix}.$$

Note that the upper left entry of the above expression corresponds to the $k \times k$ top left corner of the GSC

$\boldsymbol{\Gamma}$. As most of the energy / power is concentrated in this corner - 'Low-frequency power concentration property' of MLRTG, it is reasonable to argue that:

$$\|\boldsymbol{\Gamma}\|_F = \|\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{P}\|_F \approx \|\boldsymbol{P}_k^\top \boldsymbol{U}_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}_k^\top \boldsymbol{P}_k\|_F$$

As the low-frequency power concentration and approximate graph stationarity hold *jointly* for all the modes of a tensor, therefore,

$$\|\boldsymbol{\Gamma}^\mu\|_F = \|\boldsymbol{P}^{\mu^\top} \boldsymbol{\Omega}^\mu \boldsymbol{P}^\mu\|_F \approx \|\boldsymbol{P}^{\mu^\top}_k \boldsymbol{U}^\mu_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}^{\mu^\top}_k \boldsymbol{P}^\mu_k\|_F \quad \forall\ \mu.$$

For the ease of understanding, we imagine the case of a 2D-matrix $\boldsymbol{X}$, then the above result can be written as:

$$\|\boldsymbol{\Gamma}^1\|_F = \|\boldsymbol{P}^{1^\top} \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{P}^1\|_F \approx \|\boldsymbol{P}^{1^\top}_k \boldsymbol{U}^1_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}^{1^\top}_k \boldsymbol{P}^1_k\|_F \quad \text{and}$$

$$\|\boldsymbol{\Gamma}^2\|_F = \|\boldsymbol{P}^{2^\top} \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{P}^2\|_F \approx \|\boldsymbol{P}^{2^\top}_k \boldsymbol{U}^2_k \boldsymbol{\Sigma}_k^2 \boldsymbol{U}^{2^\top}_k \boldsymbol{P}^2_k\|_F.$$

As the above two hold jointly, therefore it is possible to conclude the following about the matrix $\boldsymbol{X}$:

$$\|\boldsymbol{P}^{1^\top} \boldsymbol{X} \boldsymbol{P}^2\|_F \approx \|\boldsymbol{P}^{1^\top}_k \boldsymbol{U}^1_k \boldsymbol{\Sigma}_k \boldsymbol{U}^{2^\top}_k \boldsymbol{P}^2_k\|_F = \|\boldsymbol{P}^{1^\top}_k \boldsymbol{X} \boldsymbol{P}^2_k\|_F = \|\boldsymbol{Z}\|_F.$$

Note that the above result is just an alternative definition of MLRTG, thus, it is possible to re-state our MLRTG model:

$$\boldsymbol{X} = \boldsymbol{P}^1_k \boldsymbol{Z} \boldsymbol{P}^{2^\top}_k.$$

Now as $\boldsymbol{X} = \boldsymbol{U}^1_k \boldsymbol{\Sigma}_k \boldsymbol{U}^{2^\top}_k$, so:

$$\|\boldsymbol{P}^1_k \boldsymbol{Z} \boldsymbol{P}^{2^\top}_k - \boldsymbol{U}^1_k \boldsymbol{\Sigma}_k \boldsymbol{U}^{2^\top}_k\|_F^2 \rightarrow 0,$$

only if the matrix $\boldsymbol{Z}$ encodes the singular values of $\boldsymbol{X}$, which can be revealed by a full SVD of $\boldsymbol{X}$. Thus,

$$\|\boldsymbol{P}^1_k \boldsymbol{A}^1_k \hat{\boldsymbol{\Sigma}}_k \boldsymbol{A}^{2^\top}_k \boldsymbol{P}^{2^\top}_k - \boldsymbol{U}^1_k \boldsymbol{\Sigma}_k \boldsymbol{U}^{2^\top}_k\|_F^2 \rightarrow 0 \implies \hat{\boldsymbol{\Sigma}}_k \approx \boldsymbol{\Sigma}_k, \quad \boldsymbol{U}^1_k \approx \boldsymbol{P}^1_k \boldsymbol{A}^1_k \quad \text{and} \quad \boldsymbol{U}^2_k \approx \boldsymbol{P}^2_k \boldsymbol{A}^2_k.$$

The above relationship states that a linear PCA based r-dimensional embedding $\boldsymbol{U}^\mu_r$ for the rows or columns of a matrix $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ is given as a linear combination of the graph eigenvectors $\boldsymbol{P}^\mu_k$. This linear combination is obtained as a result of the left / right singular vectors $\boldsymbol{A}^\mu_k$ that result from the full SVD of the GCT $\boldsymbol{Z} \in \mathbb{R}^{k \times k}$ (for the case of a matrix). Finally note that SVD of $\boldsymbol{Z} \in \mathbb{R}^{k \times k}$ is inexpensive as it costs $\mathcal{O}(k^3)$, where $k \ll n$. We call this method as 'Graph SVD'. It is straight-forward to extend it for the case of a tensor $\mathcal{X}$. In fact, the only difference lies in the SVD step being replaced by the MLSVD step of the GCT $\mathcal{Z} \in \mathbb{R}^{k \times k \times k}$. For the tensor, we call our method as 'Graph MLSVD'. This will become more clear from our mathematical derivation of Graph MLSVD in the next section, from the standard MLSVD.

Thus for any $\mathcal{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$, it is possible to compute PCA based linear embedding by rotating / linearly combining graph eigenvectors. The entire procedure can be written in the form of an algorithm, which we will state later in this chapter.

4.1.2 Derivation from MLSVD algorithm

We start from the ALS problem that is used to perform MLSVD.

For a tensor $\mathscr{Y} = \mathscr{X}^* + \boldsymbol{\eta}$, the MLSVD algorithm requires solving the following optimization problem:

$$\begin{aligned}
\min_{\boldsymbol{U}^\mu_r,\mathscr{S},\mathscr{X}} \quad & \|\operatorname{vec}(\mathscr{Y}) - \operatorname{vec}(\mathscr{X})\|_2^2 \\
\text{s.t } & {\boldsymbol{U}^\mu}_r^\top \boldsymbol{U}^\mu_r = \boldsymbol{I}_r \ \forall\ \mu, \\
& \operatorname{vec}(\mathscr{X}) = (\boldsymbol{U}^1_r \otimes \boldsymbol{U}^2_r \otimes \boldsymbol{U}^3_r)\operatorname{vec}(\mathscr{S}).
\end{aligned} \tag{4.1}$$

Our goal is to decompose $\mathscr{Y}$ into factors $\boldsymbol{H}^\mu_k$ and a core tensor $\mathscr{R}$ such that it resembles the Tucker decomposition model, i.e,

$$\operatorname{vec}(\mathscr{X}) = (\boldsymbol{H}^1_k \otimes \boldsymbol{H}^2_k \otimes \boldsymbol{H}^3_k)\operatorname{vec}(\mathscr{R}).$$

However, also recall that for our case $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$, therefore,

$$\operatorname{vec}(\mathscr{X}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\mathscr{Z}).$$

This implies that the factors $\boldsymbol{H}^\mu_k$ have a relationship with the eigenvectors $\boldsymbol{P}^\mu_k$ and the core tensor $\mathscr{R}$ with $\mathscr{Z}$. Thus, the MLSVD problem problem (4.1) for our case can be re-written as following:

$$\begin{aligned}
\min_{\boldsymbol{H}^\mu_k,\mathscr{R},\mathscr{X}} \quad & \|\operatorname{vec}(\mathscr{Y}) - \operatorname{vec}(\mathscr{X})\|_2^2 \\
\text{s.t } & \operatorname{vec}(\mathscr{X}) = (\boldsymbol{H}^1_k \otimes \boldsymbol{H}^2_k \otimes \boldsymbol{H}^3_k)\operatorname{vec}(\mathscr{R}), \\
& \operatorname{vec}(\mathscr{X}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\mathscr{Z}), \\
& {\boldsymbol{H}^\mu}_k^\top \boldsymbol{H}^\mu_k = \boldsymbol{I}_k \ \forall\ \mu
\end{aligned} \tag{4.2}$$

Using $\operatorname{vec}(\mathscr{Y}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\hat{\mathscr{Z}})$, $\operatorname{vec}(\mathscr{X}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\mathscr{Z})$ and $\|(\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\hat{\mathscr{Z}}) - (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\mathscr{Z})\|_2^2 = \|\operatorname{vec}(\hat{\mathscr{Z}}) - \operatorname{vec}(\mathscr{Z})\|_2^2$ we get

$$\begin{aligned}
\min_{\boldsymbol{H}^\mu_k,\mathscr{R},\mathscr{Z}} \quad & \|\operatorname{vec}(\hat{\mathscr{Z}}) - \operatorname{vec}(\mathscr{Z})\|_2^2 \\
\text{s.t } & (\boldsymbol{H}^1_k \otimes \boldsymbol{H}^2_k \otimes \boldsymbol{H}^3_k)\operatorname{vec}(\mathscr{R}) = (\boldsymbol{P}^1_k \otimes \boldsymbol{P}^2_k \otimes \boldsymbol{P}^3_k)\operatorname{vec}(\mathscr{Z}), \\
& {\boldsymbol{H}^\mu}_k^\top \boldsymbol{H}^\mu_k = \boldsymbol{I}_k \ \forall\ \mu.
\end{aligned} \tag{4.3}$$

To simplify this problem further, we need to understand the role played by the orthnormality constraint above. Therefore, we try to analyze it for a simpler case, i.e, a matrix $\boldsymbol{Y}$.

$$
\begin{aligned}
&\min_{\boldsymbol{H}^{\mu}_{k},\boldsymbol{R},\boldsymbol{Z}} \; \|\hat{\boldsymbol{Z}} - \boldsymbol{Z}\|_2^2 \\
&\qquad \text{s.t } \boldsymbol{H}^{1}_{k}\boldsymbol{R}\boldsymbol{H}^{2\top}_{k} = \boldsymbol{P}^{1}_{k}\boldsymbol{Z}\boldsymbol{P}^{2\top}_{k}, \\
&\qquad \boldsymbol{H}^{\mu\top}_{k}\boldsymbol{H}^{\mu}_{k} = \boldsymbol{I}_k \;\forall\; \mu.
\end{aligned} \tag{4.4}
$$

The orthonormality constraint on $\boldsymbol{H}^{\mu}_{k}$ above clearly implies that $\boldsymbol{R}$ is a diagonal matrix of singular values, which is obtained by the SVD of $\boldsymbol{P}^{1}_{k}\boldsymbol{Z}\boldsymbol{P}^{2\top}_{k}$, so we can re-write above eq. as:

$$
\begin{aligned}
&\min_{\boldsymbol{H}^{\mu}_{k},\boldsymbol{R}} \|\hat{\boldsymbol{Z}} - \boldsymbol{P}^{1\top}_{k}\boldsymbol{H}^{1}_{k}\boldsymbol{R}\boldsymbol{H}^{2\top}_{k}\boldsymbol{P}^{2}_{k}\|_2 \\
&\quad \text{s.t } \boldsymbol{H}^{\mu\top}_{k}\boldsymbol{H}^{\mu}_{k} = \boldsymbol{I} \;\forall\; \mu.
\end{aligned} \tag{4.5}
$$

The only way for $\boldsymbol{H}^{\mu}_{k}$ to be orthonormal and $\boldsymbol{R}$ to be diagonal matrix is if $\boldsymbol{H}^{\mu}_{k} = \boldsymbol{P}^{\mu}_{k}\boldsymbol{A}^{\mu}_{k}$, where $\boldsymbol{A}^{\mu}_{k} \in \mathbb{R}^{k\times k}$ is orthonormal. So, we can re-write as:

$$
\begin{aligned}
&\min_{\boldsymbol{A}^{\mu}_{k},\boldsymbol{R}} \|\hat{\boldsymbol{Z}} - \boldsymbol{A}^{1}_{k}\boldsymbol{R}\boldsymbol{A}^{2\top}_{k}\|_2^2 \\
&\quad \text{s.t } \boldsymbol{A}^{\mu\top}_{k}\boldsymbol{A}^{\mu}_{k} = \boldsymbol{I}_k \;\forall\; \mu.
\end{aligned} \tag{4.6}
$$

The solution to above problem is given by the SVD of $\boldsymbol{Z}$. Now, getting back to the tensor case, eq. (4.3) can be written as:

$$
\begin{aligned}
&\min_{\boldsymbol{A}^{\mu}_{k},\mathscr{R}} \|\operatorname{vec}(\hat{\mathscr{Z}}) - (\boldsymbol{A}^{1}_{k}\otimes\boldsymbol{A}^{2}_{k}\otimes\boldsymbol{A}^{3}_{k})\operatorname{vec}(\mathscr{R})\|_2^2 \\
&\quad \text{s.t } \boldsymbol{A}^{\mu\top}_{k}\boldsymbol{A}^{\mu}_{k} = \boldsymbol{I}_k \;\forall\; \mu,
\end{aligned} \tag{4.7}
$$

whose solution is given by the MLSVD of $\mathscr{Z}$. Recall, that $\operatorname{vec}(\mathscr{Z}) = (\boldsymbol{P}^{1}_{k}\otimes\boldsymbol{P}^{2}_{k}\otimes\boldsymbol{P}^{3}_{k})^{\top}\operatorname{vec}(\mathscr{Y})$. The MLSVD can be performed by ALS method, whose complexity is $\mathscr{O}(Ik^4)$, where I is the number of iterations.

Finally, what needs to be shown is that $\boldsymbol{U}^{\mu}_{k} = \boldsymbol{P}^{\mu}_{k}\boldsymbol{A}^{\mu}_{k}$. Let $\underline{\mathscr{X}}$ be the low-rank approximation obtained via MLSVD eq. (4.1) and $\mathscr{X}$ be the one obtained by eq. (4.2). As both $\underline{\mathscr{X}}$ and $\mathscr{X}$ are approximations of $\mathscr{Y}$ and our problems are designed to minimize $\|\operatorname{vec}(\mathscr{Y}) - \operatorname{vec}(\mathscr{X})\|_2^2$ and $\|\operatorname{vec}(\mathscr{Y}) - \operatorname{vec}(\underline{\mathscr{X}})\|_2^2$, this implies $\|\operatorname{vec}(\mathscr{X}) - \operatorname{vec}(\underline{\mathscr{X}})\|_2 \to 0$.

The only way for $\|\operatorname{vec}(\mathscr{X}) - \operatorname{vec}(\underline{\mathscr{X}})\|_2 \to 0$ is if:

$$
\boldsymbol{U}^{\mu}_{k} \to \boldsymbol{H}^{\mu}_{k} \quad \text{and} \quad \mathscr{S} \to \mathscr{R}.
$$

Thus, in order to obtain a rank r orthonormal MLSVD embedding $\boldsymbol{U}^{\mu}_{r}$ from rank k-graph eigenvectors $\boldsymbol{P}^{\mu}_{k}$, one can use:

$$
\boldsymbol{U}^{\mu}_{r} = (\boldsymbol{P}^{\mu}_{k}\boldsymbol{A}^{\mu}_{k})_r,
$$

i.e, the first r columns of the matrix $\boldsymbol{P}^{\mu}_{k}\boldsymbol{A}^{\mu}_{k}$, assuming $k > r$.

Note that in practice, the eigen-gap assumptions do not hold. Therefore, one has to estimate the graph eigen-vectors $\boldsymbol{P}^{\mu}_{k}$ from the noisy tensor $\mathscr{Y}$. Thus, the above equalities hold only approximately. Furthermore, the graph multilinear rank (k, k, k) is not known beforehand. For now, we advise to choose

$k > r$ and discuss this further in the experimental section of this chapter. We defer the quality of this approximation to Chapter 7, where we will re-derive the GMLSVD algorithm from an approximation problem and quantify the approximation error. Obviously, the error would depend on the eigen-gaps of the graph Laplacians.

4.1.3 Algorithm for GMLSVD

For a tensor $\mathcal{Y} = \mathcal{X}^* + \boldsymbol{\eta}$, one can compute GMLSVD in the following steps:

1. Matricize the tensor $\mathcal{Y}$ along each of its modes $\boldsymbol{Y}^\mu$.
2. Compute k_{nn}-graphs $\mathbb{G}^\mu$ with combinatorial Laplacians $\boldsymbol{L}^\mu$ among the rows of $\boldsymbol{Y}^\mu$.
3. Compute the first k eigenvectors $\boldsymbol{P}_k^\mu$ for each of $\boldsymbol{L}^\mu$.
4. Project the tensor $\mathcal{Y}$ on the graph eigenvectors $\boldsymbol{P}_k^\mu$ to get the GCT $\mathcal{Z}$ as following: $\text{vec}(\mathcal{Z}) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)^\top \text{vec}(\mathcal{Y})$.
5. Perform a full MLSVD of $\mathcal{Z} \in \mathbb{R}^{k \times k \times k}$, using the ALS method (eq. (4.7)), to get the singular vectors $\boldsymbol{A}_k^\mu$ and the core tensor $\mathcal{R}$.
6. Set $\boldsymbol{H}_r^\mu = (\boldsymbol{P}_k^\mu \boldsymbol{A}_k^\mu)_r$, i.e, the first r columns of the product $\boldsymbol{P}_k^\mu \boldsymbol{A}_k^\mu$.

Note that for the case of a matrix $\boldsymbol{Y}$, step 5 above reduces to an SVD and the matrix $\boldsymbol{R}$ to a diagonal matrix. Therefore, we call the matrix version of GMLSVD as Graph SVD (GSVD).

4.1.4 Complexity

The complexity of graph construction using FLANN scales with $\mathcal{O}(n^3 \log(n))$, the computation of Laplacian eigenvectors scales with $\mathcal{O}(nk^2)$ and that of the ALS algorithm for the full MLSVD scales with $\mathcal{O}(Ik^4)$, where I is the number of iterations. Thus, the overall complexity of GMLSVD scales with $\mathcal{O}(n^3 \log(n) + nk^2 + Ik^4)$, where $k \ll n$. For a $k > r$, this complexity is more than that of the simple MLSVD algorithm ($\mathcal{O}(nr^2)$). However, we emphasize that the advantage of our GMLSVD method is more prominent in iterative tensor PCA based methods, which require a full SVD in every iteration of the algorithm. Such approximation problems and algorithms will be presented in the part III of this thesis.

4.1.5 Properties of GMLSVD

For any $\mathcal{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$, the core tensor $\mathcal{R}$ obtained from GMLSVD satisfies the same properties as the core tensor obtained from MLSVD. Here we review the following two properties satisfied by the core tensor $\mathcal{R} \in \mathbb{R}^{k \times k \times k}$

- Any two subtensors / slices of the core tensor $\mathcal{R}$ along a μ^{th} mode are orthogonal, i.e, ${\mathcal{R}^{\mu}}_i^\top \mathcal{R}_j^\mu = \mathbf{0}$, $\forall\ i \neq j$.
- The subtensors / slices of the core tensor $\mathcal{R}$ along a μ^{th} mode are ordered in the decreasing order of their Frobenius norms , i.e, $\|\mathcal{R}_1^\mu\|_2 \geq \|\mathcal{R}_2^\mu\|_2 \geq \cdots \geq \|\mathcal{R}_k^\mu\|_2$.

4.2 Examples

Our first example consists of the artificial exact MLRTG which can be generated using method 1 of Section 3.5.1. Recall from Fig. 3.4, that the matrix $\boldsymbol{Y} \in \mathbb{R}^{200\times 200}$ generated by this method has a rank $r = 3$ and is clusterable into $k = 4$ clusters across the rows and columns. We construct the k_{nn}-graphs along the rows and columns using the standard method and $k_{nn} = 40$ and compute the eigenvectors $\boldsymbol{P}^1_4, \boldsymbol{P}^2_4$. Then, we compute the left and right singular vectors and singular values as: $\boldsymbol{Y} = \boldsymbol{U}^1_3 \boldsymbol{\Sigma}_3 \boldsymbol{U}^{2\top}_3$ and run the GMLSVD algorithm to get $\boldsymbol{H}^1_2, \boldsymbol{H}^2_2$. The two rows of Fig. 4.2 plot the actual and estimated singular vectors and the Laplacian eigenvectors. It is clear to see that the estimated singular vectors $\boldsymbol{H}^\mu_2$ are exactly equal to $\boldsymbol{U}^\mu_2$ for this exact MLRTG dataset. These singular vectors are obtained by a linear combination of Laplacian eigenvectors, which are cluster indicators for our example. The singular values obtained by GMLSVD are exactly equal to those of the actual matrix and are not shown here for brevity.

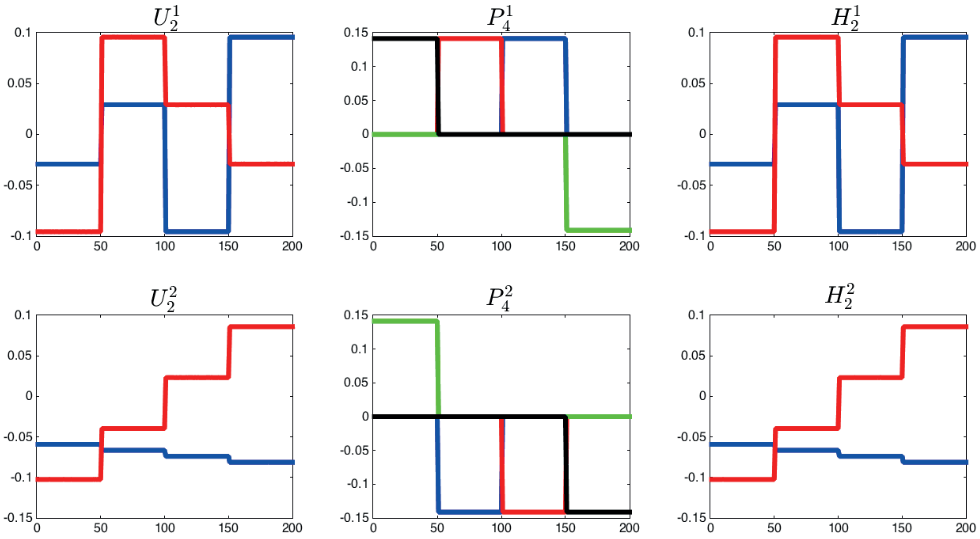


Figure 4.2 – A comparison of the first two singular vectors of the exact MLRTG $\boldsymbol{Y}$ and those estimated by the GMLSVD method. The left most column shows the first two singular vectors in the left (upper) and right (lower) subspaces obtained via SVD. The red line corresponds to the 1st singular vector and the blue to the 2nd. The middle plot shows the first four eigenvectors (black, red, blue green) of the row and column graphs. The rightmost column shows the resulting left (upper) and right (lower) singular vectors obtained by linear combination of the graph eigenvectors, i.e., by the GMLSVD method.

Now, consider the examples of several 2D and 3D datasets as shown in Fig. 4.3. The first four rows of this figure correspond to the case of 2D datasets, where each image is vectorized and stacked in the columns of a matrix. The last row corresponds to the example of a 3D hyperspectral face tensor, where the x and y dimensions are the spatial dimensions and the z-dimension is the spectrum. The leftmost plots in each row show a sample image from the dataset, the middle plot shows the first singular vectors obtained via our GSVD (for matrices) or GMLSVD (for tensors) and the rightmost plot shows the first singular vector obtained via SVD and MLSVD algorithms. It is quite obvious to see that the quality of the first singular vectors obtained by our proposed method is very similar to that of the standard algorithms. More specifically, the first two rows belong to specific classes of images from the benchmark COIL20 dataset. The singular vectors for these images have a different shape from the sample images because the images in these dataset correspond to a 360 degree rotation. The third and fourth rows show the cases for specific classes of benchmark face datasets ORL and CMUPIE. The singular vectors for these datasets look like faces because the images undergo a small pose variation only.

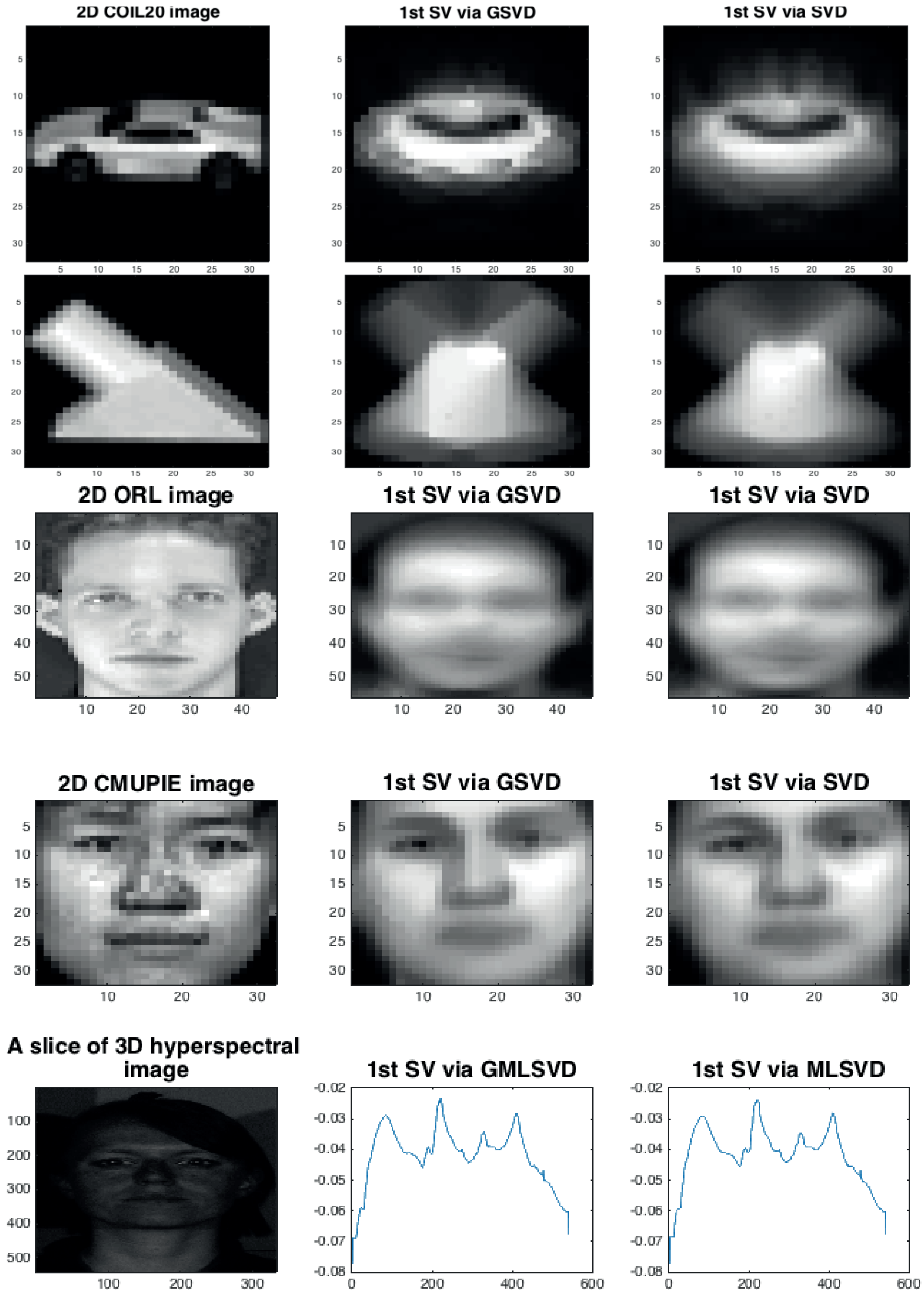


Figure 4.3 – A comparison of the 1st singular vectors obtained via GSVD and SVD for various 2D and 3D real face tensors. In each row, the leftmost plot shows an example image from the database, the middle plot shows the 1st singular vector of mode 1 obtained via GSVD and the right plot shows the 1st singular vector of mode 1 obtained via SVD. It can be clearly seen that the singular vectors determined by GSVD and SVD are equivalent.

Note that this figure only shows the example of first singular vector. Furthermore, the behavior of singular values is also of particular interest for our method. Therefore, we study this in detail for 3D hyperspectral face tensor in Fig. 4.4. In Fig. 4.4, we plot the alignment of the singular vectors obtained via GMLSVD and MLSVD and the singular values for each of the three modes of a hyperspectral face tensor $\mathscr{Y} \in \mathbb{R}^{542\times333\times146}$. For our GMLSVD method, we use a GCT of size $\mathscr{Z} \in \mathbb{R}^{70\times70\times30}$. Let $\boldsymbol{H}_r^{\mu} = (\boldsymbol{P}_k^{\mu}\boldsymbol{A}_k^{\mu})_r$ be the singular vectors obtained via our GMLSVD method and $\boldsymbol{U}_r^{\mu}$ be the ones obtained by the MLSVD method, then, in order to study the alignment of the two types of singular vectors, we plot $|\boldsymbol{U}^{\mu}{}_r^{\top}\boldsymbol{H}_r^{\mu}|$. Since both the singular vectors are orthonormal, the metric should be a diagonal matrix. The first row of Fig. 4.4 shows that the first 20 singular vectors are very well aligned for the x and y dimensions, however, the alignment deteriorates for higher singular vectors. Similar behavior can be observed for the singular values in the second row. Furthermore, note that the alignment is even worse for the spectrum dimension. This phenomenon can be explained in terms of the approximate stationarity of the modes of this tensor on its graphs which was studied in the third row of Fig. 3.5. Clearly, the eigen-gap assumption does not hold for this tensor which results in a leakage of energy outside a given choice of k for each of the modes. In fact, larger the k is, better is the alignment of singular vectors and singular values obtained via both methods. Nevertheless, the alignment of the first few singular vectors and singular values is enough for our approximation purpose. Our experiment shows that a normalized reconstruction ℓ_2 error of 0.05 is obtained with this choice of GCT.

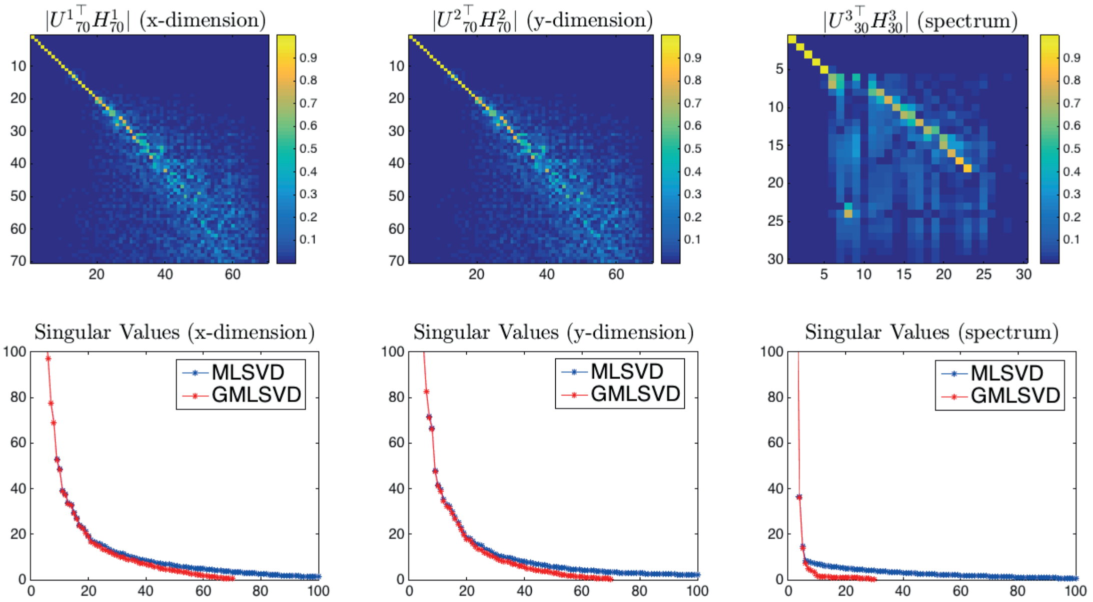


Figure 4.4 – Visualization of the singular vector and singular values alignment for each of the three modes of hyperspectral face tensor.

4.3 Choice of graph multilinear rank

In the previous section we saw that the quality of approximation of the singular vectors and values depends on the size of GCT, which is called the graph multilinear rank. In this section, we study this effect in detail for an artificial low-rank tensor. There can be many methods for the generation of artificial low-rank tensors (Section 3.5.1). We do not consider the clusterable tensor example here as it is very clear from previous chapter that the choice of k for such tensors is made according to the eigen-gap. Furthermore, an exact MLRTG, like the one we discussed in the previous section, does not correspond to a real world scenario. Therefore, in order to understand the effect of the choice of k on our GMLSVD method, we switch to more realistic data generation setup which does not result in an eigen-gap, and requires tuning k, yet possesses approximate graph stationarity and low-frequency power concentration properties. These properties are inherent in the artificial MLRTG generated via methods 3 or 4 of Section 3.5.1.

Fig. 4.5 shows variation in the alignment of first 20 singular vectors $|{\boldsymbol{H}^1}^\top_{20}\boldsymbol{U}^1_{20}|$, where $\boldsymbol{H}$ corresponds to the vectors obtained by GMLSVD and $\boldsymbol{U}$ to the ones obtained by MLSVD and singular values for a tensor $\boldsymbol{Y} \in \mathbb{R}^{200\times 200}$ with a multilinear rank $(r,r) = (10,10)$, generated with the method 3 from Section 3.5.1, for different graph multilinear rank k. Clearly, the alignment increases with an increase in k from 20 to 80 with a decrease in the ℓ_2 reconstruction error of the tensor.

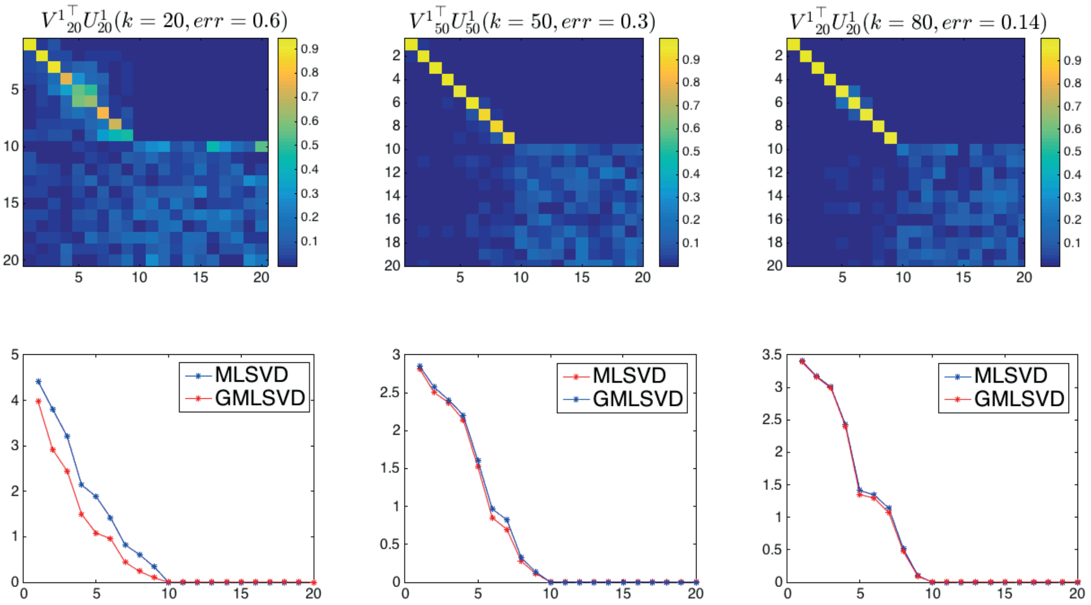


Figure 4.5 – Increase in the singular vector and singular values alignment and decrease in the normalized reconstruction error for mode 1 of an artificial tensor w.r.t increasing graph multilinear rank k.

In chapter 3, we pointed out that for a low-rank matrix of rank r, the eigen-gap assumption only exists if the k_{nn}-graph Laplacian $\boldsymbol{L}$ is constructed by inverting the low-rank empirical covariance matrix. However, such an inversion results in a dense graph, which contradicts the local smoothness assumption used in the graph signal processing framework. Since the eigen-gap assumption does not hold for k_{nn}-graphs constructed from a matrix $\boldsymbol{Y}$, unless the matrix $\boldsymbol{Y}$ has a strong community structure, we recommend to use a $k > r$, but $k \ll n$. Since a bigger k results in an increase in the computational complexity, this also calls for a trade-off between the quality of approximation and the complexity. Finally, we again point out that the advantage of the proposed GMLSVD method over MLSVD is not obvious, from the discussion carried out in this chapter. However, in the next part of this thesis, we will use this framework to define computationally efficient low-rank approximation problems, which will reveal the advantage of our GMLSVD method.

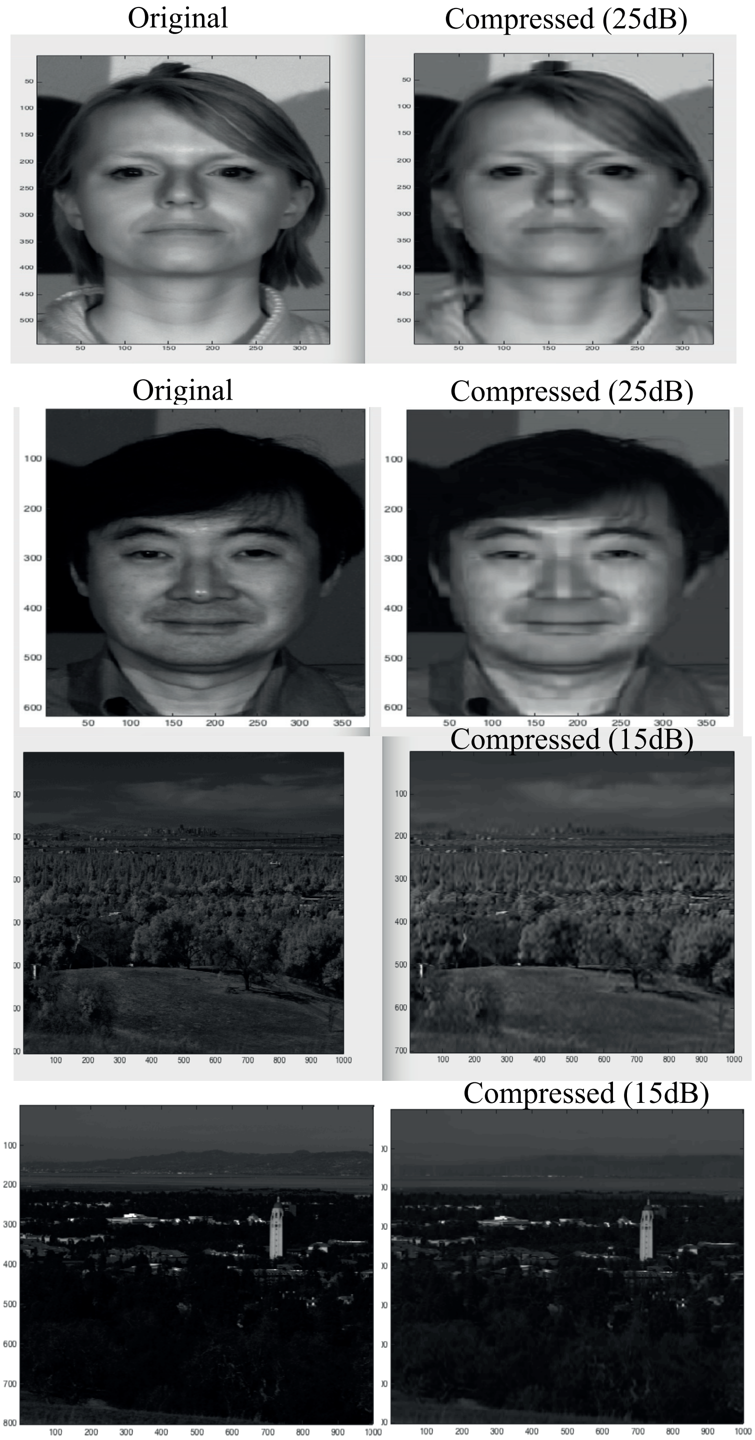
4.4 GMLSVD application: Tensor Compression

An obvious application of HOSVD or MLSVD is tensor compression, as one can store the factors and core tensor instead of the entire tensor. For a tensor $\mathcal{Y} \in \mathbb{R}^{n\times n\times n}$ with a multilinear rank (r, r, r), MLSVD results in a compression of the order $n^3/(3nr + r^3)$. For example, for $n = 100$ and $r = 10$, this results in a compression of approximately 250 times. As GMLSVD is an alternative to MLSVD, it can be used for tensor compression as well. In fact, unlike MLSVD, one only needs the graph eigenvectors $\boldsymbol{P}_k^{\mu}$ and the GCT $\mathcal{X}$ to compress. Thus, the ALS algorithm does not need to be run to do an additional MLSVD on $\mathcal{X}$. Similarly to MLSVD, our GMLSVD algorithm can result in a compression of $n^3/(3nk + k^3)$ times.

Fig. 4.6 shows the compression results for four hyperspectral tensors obtained from the Stanford database. The first row shows a slice of a face tensor of size $\mathcal{Y} \in \mathbb{R}^{542\times 333\times 148}$, which is compressed using a GCT of size $\mathcal{X} \in \mathbb{R}^{70\times 70\times 30}$. A compression of 150 times is obtained while maintaining an SNR of 25dB. Second row shows a slice from another face tensor $\mathcal{Y} \in \mathbb{R}^{615\times 376\times 148}$ for which a GCT of size $\mathcal{X} \in \mathbb{R}^{100\times 50\times 20}$ is used to attain a compression of 186 times while maintaining an SNR of 25dB. The third and fourth rows show results for landscape tensors $\mathcal{Y} \in \mathbb{R}^{702\times 1000\times 148}$ and $\mathcal{Y} \in \mathbb{R}^{801\times 1000\times 148}$ which were compressed using a GCT $\mathcal{X} \in \mathbb{R}^{200\times 200\times 50}$ while maintaining an SNR of 15dB.

4.5 Conclusion

We propose a Graph Multilinear SVD (GMLSVD) method for MLRTG which is an alternative to the standard PCA and MLSVD for low-rank tensors. The method provides us with a mechanism to extract a low-dimensional linear PCA based embedding from the non-linear graph embedding. This can be done by linearly combining the first few Laplacian eigenvectors of the modes of a tensor, where the linear combination is given by the singular vectors of the full SVD of the GCT. Although the method is computationally not advantageous over the standard methods, it can be used to perform cheap SVD updates in iterative algorithms, such as Robust matrix and tensor PCA algorithms. Such computationally tractable approximation problems will be defined in the next part of this thesis.



Original

Compressed (25dB)

Original

Compressed (25dB)

Compressed (15dB)

Compressed (15dB)

Figure 4.6 – Qualitative and quantitative results for the compression of various hyperspectral tensors from the Stanford database.

Part III

Proposed Approximation Methods & Applications

5 Fast Robust PCA on Graphs (FRPCAG)

Note: *Most of the parts of this chapter have been published in the following article:*

"N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast Robust PCA on Graphs," IEEE Journal of Selected Topics in Signal Processing, vol. 10, no. 4, pp. 740–756, 2016. "

Some parts have also appeared in the following technical report on arXiv:

"N. Shahid, N. Perraudin, P. Vandergheynst, "Low-Rank Matrices on Graphs: Generalized Recovery & Applications", arXiv: preprint arXiv:1605.05579. "

In the previous chapters we presented our novel low-rank signal model, Multilinear Low-Rank Tensors on Graphs (MLRTG), which states that a low-rank tensor or matrix can be encoded as a multilinear combination of graph eigenvectors. We also presented a formal method to recover PCA based *linear* row and column embedding for a dataset from the graph eigenvectors.

We recall, that our primary goal in this thesis is to recover the underlying non-linear low-rank structure in data space via highly scalable algorithms, using graphs. From Chapter 1, we know that scalabilty is a primary issue in the nuclear norm based iterative algorithms, such as matrix and tensor Robust PCA. Therefore, our goal is to recover low-rank structures from sparse or grossly corrupted data, i.e, low-rank and sparse decomposition. In this and the next chapters we propose a set of low-rank recovery problems for matrices and tensors. More generally, we present a set of approximation problems which can benefit algorithmically and computationally from MLRTG and the GMLSVD setup to improve the scalability of low-rank recovery problems. We present the optimization problems and algorithms to solve these problems as well as detailed theoretical analysis.

Throughout this part of the thesis, our exposition will be centered around the two standard properties of low-rank recovery methods, presented in Chapter 3, i.e, singular vector determination and singular value thresholding. We will somehow indirectly or directly bring into use the GMLSVD framework we proposed previously to provide us with an initial left and right singular estimate for the data matrix, in terms of the graph eigenvectors. This will lead to computational advantages. We will conclude our discussion via a detailed complexity comparison of each of the methods. We will present applications of the proposed approximation problems for low-rank and sparse decomposition and low-rank feature clustering.

Given a noisy matrix $\boldsymbol{Y}$, our goal is to recover the underlying low-rank matrix $\boldsymbol{X}^*$ from sparse component. We propose two problems for this purpose:

1. Fast Robust PCA on Graphs (FRPCAG)
2. Compressive PCA on Graphs (CPCA)

Each of the above problems assumes that the recovered tensor / matrix is MLRTG. Our analysis will show that in addition to recovering low-rank structures approximately, our methods also reveal the class structure of the datasets, if it exists. Thus, we will show experiments on low-rank and sparse decomposition and clustering for each of these methods.

We pointed out earlier in Chapter 1, that our proposed approximation problems fall into two categories, one which makes implicit use of the GMLSVD framework and the other which makes explicit use of it. We specifically highlight that FRPCAG and CPCA are meant for the cases when the number of samples n and the features p are very large and one can afford to lose the exactness of the representation for gain in speed. In this context, these methods *implicitly* bring into use the GMLSVD framework, i.e., they rely on the graph Dirichlet priors rather than the explicit use of the graph eigenvectors, as computing the graph eigenvectors and tuning the size of subspaces can be cumbersome.

5.1 Introduction to Fast Robust PCA on Graphs

FRPCAG is a fast, scalable, robust and convex method which recovers a low-rank matrix $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^{\mu})$ from a sparsely or grossly corrupted matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$.

1. We propose an approximate low-rank recovery method for corrupted data by utilizing only the graph smoothness assumptions both between the samples and between the features.
2. Our theoretical analysis proves that the proposed model is able to recover approximate low-rank representations with a bounded error for clusterable data. Thus, the method can be used for clustering of datasets.
3. Our model is convex and although non-smooth it can be solved efficiently, that is in linear time in the number of samples, with a few iterations of the well-known FISTA algorithm. The construction of the two graphs costs $\mathcal{O}(np\log n) + \mathcal{O}(np\log p)$ time, where n is the number of data samples and p the number of features.
4. The resulting algorithm is highly scalable for large datasets since it requires only the multiplication of two sparse matrices with full vectors and elementwise soft-thresholding operations.
5. Our extensive experimentation shows that the recovered close-to-low-rank matrix is a good approximation of the low-rank matrix obtained by solving the expensive state-of-the-art method [99] which uses the much more expensive nuclear norm. This is observed even in the presence of gross corruptions in the data.

Throughout this and the next chapter, for the ease of notation, we will refer to the row and column graphs, Laplacians and eigenvectors with ’r’ and ’c’ super-scripts.

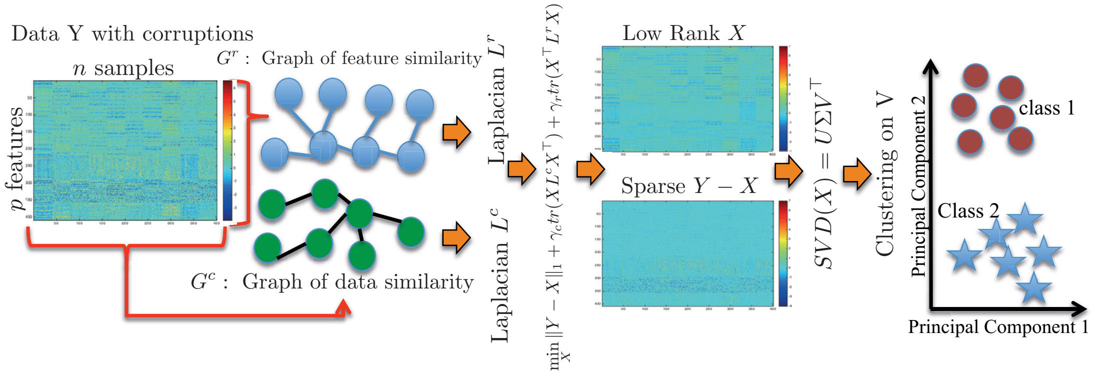


Figure 5.1 – Main idea of Fast Robust PCA on Graphs.

5.2 Optimization Problem

In order to formulate FRPCAG, we begin from the definition of approximate MLRTG which states the following:

$$\begin{aligned}
\boldsymbol{Y} &= \boldsymbol{X} + \boldsymbol{\eta} = \boldsymbol{X}^* + \bar{\boldsymbol{X}} + \boldsymbol{\eta} \\
&= \boldsymbol{P}_k^r \boldsymbol{Z} \boldsymbol{P}_k^{c\top} + \bar{\boldsymbol{P}}_k^r \bar{\boldsymbol{Z}} \boldsymbol{P}_k^{\bar{c}\top} + \boldsymbol{\eta} \quad \text{where, } \|\bar{\boldsymbol{Z}}\|_F \ll \|\boldsymbol{Z}\|_F \\
&= \boldsymbol{P}_k^r \boldsymbol{A} + \bar{\boldsymbol{P}}_k^r \bar{\boldsymbol{A}} + \boldsymbol{\eta} = \boldsymbol{B} \boldsymbol{P}_k^{c\top} + \bar{\boldsymbol{B}} \boldsymbol{P}_k^{\bar{c}\top} + \boldsymbol{\eta} \quad \text{where, } \|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F \quad \text{and} \quad \|\bar{\boldsymbol{B}}\|_F \ll \|\boldsymbol{B}\|_F
\end{aligned}$$

From Chapter 3 we know that this is equivalent to the following:

$$\boldsymbol{X}(i) \in span(\boldsymbol{P}_k^r), \forall\, i \quad \text{and} \quad \boldsymbol{X}^\top(i) \in span(\boldsymbol{P}_k^c), \forall\, i,$$

where $\boldsymbol{P}^c$ and $\boldsymbol{P}^r$ are the eigenvector matrices of the graph Laplacians $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ of the k_{nn}-graphs constructed between the columns and rows of $\boldsymbol{Y}$.

The simplest method to recover $\boldsymbol{X}$ would be to solve the following optimization problem:

$$\begin{aligned}
&\min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{X}\|_1 \\
&\quad \text{s.t } \boldsymbol{X}(i) \in span(\boldsymbol{P}_k^r) \quad \forall\, i \\
&\qquad \boldsymbol{X}^\top(i) \in span(\boldsymbol{P}_k^c) \quad \forall\, i,
\end{aligned}$$

where $\phi(\cdot)$ is a proper, positive, convex and lower semi-continuous loss function (possibly ℓ_p-norms $\|\cdot\|_1, \|\cdot\|_2^2, \ldots, \|\cdot\|_p^p$). The choice of $\phi(\cdot)$ depends on the noise, for example, ϕ is an ℓ_2 norm for Gaussian noise and ℓ_1 for sparse noise.

The above optimization problem is computationally prohibitive because the exact value of k is not known and one might need a full eigenvalue decomposition of the graph Laplacians $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$, which costs $\mathcal{O}(n^3)$ and $\mathcal{O}(p^3)$ respectively. Furthermore, the span constraints do not dictate singular value behavior of $\boldsymbol{X}$. In order to avoid this cost and ambiguity, we recall the notion of smoothness of a signal $\boldsymbol{X}$ lying on a graph $\mathbb{G}$, measured by the graph Dirichlet energy $\mathrm{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top)$. This measure of smoothness is equivalent to the notion of band-limitedness of a signal on the graph, i.e, the signal belongs to the span of the first few Laplacian eigenvectors [100]. Using this measure for both of the constraints, we write our proposed model as following:

$$\min_{\boldsymbol{X}} \|\boldsymbol{Y}-\boldsymbol{X}\|_1 + \gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X}), \tag{5.1}$$

where $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ are combinatorial graph Laplacians of the k_{nn}-graphs constructed from the rows and columns of $\boldsymbol{Y}$ and γ_c and γ_r are the graph regularization parameters. We call the above model "Fast Robust PCA on Graphs". The main idea is summarized in Fig. 5.1. As we will explain shortly, the trace terms model the joint smoothness of the rows and columns of $\boldsymbol{X}$ on their respective graphs and recover a low-rank matrix and as well as dictate the decreasing behavior of the singular values of $\boldsymbol{X}$.

Note that in eq. 5.1, we do not have direct access to the rank of matrix $\boldsymbol{X}$. Instead, the rank is controlled by regularization parameters γ_c and γ_r. The bigger these parameters, higher is the smoothness and lower the rank of $\boldsymbol{X}$. However, as explained in the later sections of this chapter, $\boldsymbol{X}$ is approximately low-rank, since the lower singular values are never exactly zero. If one desires to obtain an exact low-rank matrix $\boldsymbol{X}$, then we propose to perform one SVD of $\boldsymbol{X}$ at the end of the algorithm and then set the singular values below a certain threshold to zero. One can also use the "OptShrink" method for this purpose [101].

5.3 Connections with the state-of-the-art

We discuss some important implications of our work and its connections with the state-of-the-art that uses dual graph based filtering. Obviously, our dual graph filtering approach makes sense for a number of real world applications mentioned throughout this thesis. This is proved experimentally in the results section of this work as well. Apparently, the use of dual-graph filtering should help in a broad range of clustering applications. However, we highly discourage that this approach cannot be adopted as a rule-of-thumb in any graph based convex optimization problem, specially if one wants to target an exact low-rank recovery. Interestingly, we point out a few examples from the state-of-the-art which have close connections with our approach and justify why our model is different from those works.

The collaborative filtering approach is commonly used in the recommendation systems where one uses a notion of graph based similarity between the users and items to perform an effective recommendation. The authors of [102] use an NMF based approach with dual graph filtering in their proposed recommendation system. Similarly, the authors of [33] propose a hybrid recommendation with nuclear-norm based exact recovery and dual-graph filtering. While the authors of [33] show that when the number of measurements is very small then their proposed dual-graph filtering approach outperforms the state-of-the-art methods, their method does not improve upon the exact recovery of the nuclear-norm based approach [31]. Thus, it makes sense to use the dual-graph filtering to reduce the error of approximation in the low-rank but it does not ensure an exact recovery if a standard exact recovery mechanism (like nuclear norm) is not available.

Robust PCA on Graphs (RPCAG) [99], which incorporates a graph regularization to the RPCA framework [99], has also proved that the use of graph filtering does not improve upon the exact recovery conditions over the standard RPCA framework [1]. In fact, as shown in the rank-sparsity plots in [99], the graph based filtering reduces the error over RPCA [1] for various rank-sparsity pairs but it does not guarantee an exact recovery in the region where the nuclear norm based recovery fails.

The conclusions drawn from the above two state-of-the-art techniques are exactly in accordance with the motivation of this work. Our motivation is not to design a framework that returns an exact low-rank representation but to present a system that works well under a broad range of conditions in contrast to those required for exact recovery [1]. Thus, we target an approximate and fast recovery under a broad range of conditions and data types. This is exactly why our proposed framework, apparently very closely related to [102] & [33], is entirely different from these works. The recovery of our method totally depends on the quality of information provided by the graphs. Moreover, our framework relies on the assumption that the data matrices are dual band-limited on two graphs. Although, such an assumption is implicit in [102] & [33], the authors do not discuss it theoretically or experimentally in their works. Furthermore, RPCAG [99] uses only one graph. Thus, the assumption behind the RPCAG model is totally different from our proposed framework.

5.4 Algorithm

We use the Fast Iterative Soft Thresholding Algorithm (FISTA) [103] to solve problem (5.1). Let $g:\mathbb{R}^{\mathbb{N}} \to \mathbb{R}$ be a convex, differentiable function with a β-Lipschitz continuous gradient ∇g and $h:\mathbb{R}^{\mathbb{N}} \to \mathbb{R}$ a convex function with a proximity operator $\mathrm{prox}_h : \mathbb{R}^N \to \mathbb{R}^N$ defined as:

$$\mathrm{prox}_{\lambda h}(y) = \operatorname*{argmin}_x \frac{1}{2}\|x-y\|_2^2 + \lambda h(x).$$

Our goal is to minimize the sum $g(x)+h(x)$, which is done efficiently with proximal splitting methods. More information about proximal operators and splitting methods for non-smooth convex optimization can be found in [104]. For model 5.1, $g(\boldsymbol{X}) = \gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X})$ and $h(\boldsymbol{X}) = \|\boldsymbol{Y}-\boldsymbol{X}\|_1$. The gradient of g becomes

$$\nabla_g(\boldsymbol{X}) = 2(\gamma_c \boldsymbol{X}\boldsymbol{L}^c + \gamma_r \boldsymbol{L}^r \boldsymbol{X}). \tag{5.2}$$

We define an upper bound on the Lipschitz constant β as $\beta \leq \beta' = 2\gamma_c\|\boldsymbol{L}^c\|_2 + 2\gamma_r\|\boldsymbol{L}^r\|_2$ where $\|\boldsymbol{L}\|_2$ is the spectral norm (or maximum eigenvalue) of $\boldsymbol{L}$. Moreover, the proximal operator of the function h is the ℓ_1 soft-thresholding given by the elementwise operations (here $\circ$ is the Hadamard product)

$$\mathrm{prox}_{\lambda h}(\boldsymbol{X}) = \boldsymbol{Y} + \operatorname{sgn}(\boldsymbol{X}-\boldsymbol{Y}) \circ \max(|\boldsymbol{Y}-\boldsymbol{X}| - \lambda, 0). \tag{5.3}$$

The FISTA algorithm [103] can now be stated as Algorithm 1, where λ is the step size (we use $\lambda = \frac{1}{\beta'}$), ϵ the stopping tolerance and J the maximum number of iterations.

5.5 Computational Complexity

The overall computational complexity of FRPCAG consists of the one time complexity of graph construction between the rows and columns of $\boldsymbol{Y}$ and the per iteration complexity of the FISTA algorithm. We discuss each of the two in detail below.

Complexity of Graph Construction: We use the Fast Approximate nearest neighbor search based library (FLANN) [60] for construction of $\mathbb{G}^c$ and $\mathbb{G}^r$. For n p-dimensional vectors, the computational

Algorithm 1 FISTA for FRPCAG

```
INPUT: $\boldsymbol{T}_1 = \boldsymbol{Y}, \boldsymbol{X}_0 = \boldsymbol{Y}, t_1 = 1, \epsilon > 0$
for $j = 1, \dots J$ do
    $\boldsymbol{X}_j = \text{prox}_{\lambda_j h}(\boldsymbol{T}_j - \lambda_j \nabla g(\boldsymbol{T}_j))$
    $t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2}$
    $\boldsymbol{T}_{j+1} = \boldsymbol{X}_j + \frac{t_j - 1}{t_{j+1}}(\boldsymbol{X}_j - \boldsymbol{X}_{j-1})$
    if $\|\boldsymbol{T}_{j+1} - \boldsymbol{T}_j\|_F^2 < \epsilon \|\boldsymbol{T}_j\|_F^2$ then
        BREAK
    end if
end for
OUTPUT: $\boldsymbol{X}_{j+1}$
```

complexity of the FLANN algorithm is $\mathcal{O}(pnk_{nn}(\log(n)/\log(k_{nn})))$ for the graph $\mathbb{G}^c$ between the samples and $\mathcal{O}(pnk_{nn}(\log(p)/\log(k_{nn})))$ for the graph $\mathbb{G}^r$ between the features, where k_{nn} is the number of nearest neighbors. This has been shown in [61] & [60] . For a fixed k_{nn} the complexity of $\mathbb{G}^c$ is $\mathcal{O}(pn\log(n))$ and that of $\mathbb{G}^r$ is $\mathcal{O}(np\log(p))$. We use $k_{nn} = 10$ for all the experiments reported in this work. The effect of the variation of k_{nn} on the performance of our model is studied briefly in the experimental section of this chapter.

FISTA: Let I denote the number of iterations for the algorithm to converge, p is the data dimension, n is the number of samples and r is the rank of the low-dimensional space. The computational cost of our algorithm per iteration is linear in the number of data samples n, i.e. $\mathcal{O}(Ipn)$ for I iterations.

Final SVD: Our model, in order to preserve convexity, finds an approximately low-rank solution $\boldsymbol{X}$ without explicitly factorizing it. While this gives a great advantage, depending on the application we have in hand, we might need to provide explicitly the low dimensional representation in a factorized form. This can be done by computing an "economic" SVD of $\boldsymbol{X}$ after our algorithm has finished.

Most importantly, this computation can be done in time that scales linearly with the number of samples for a fixed number of features $p < n$. Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ the SVD of $\boldsymbol{X}$. The orthonormal basis $\boldsymbol{U}$ can be computed by the eigenvalue decomposition of the small $p \times p$ matrix $\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{U}\boldsymbol{E}\boldsymbol{U}^\top$ that also reveals the singular values $\boldsymbol{\Sigma} = \sqrt{\boldsymbol{E}}$ since $\boldsymbol{X}\boldsymbol{X}^\top$ is s.p.s.d. and therefore $\boldsymbol{E}$ is non-negative diagonal. Here we choose to keep only the r biggest singular values and corresponding vectors according to the application in hand (the procedure for determining r is explained in Section 5.8). Given $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ the sample projections are computed as $\boldsymbol{V} = \boldsymbol{\Sigma}^{-1}\boldsymbol{U}^\top\boldsymbol{X}$.

The complexity of this SVD is $\mathcal{O}(np^2)$ –due to the multiplication $\boldsymbol{X}\boldsymbol{X}^\top$– and does not change the asymptotic complexity of our algorithm. Note that the standard economic SVD implementation in numerical analysis software typically does not use this simple trick, in order to achieve better numerical error. However, in most machine learning applications like the ones of interest in this paper, the compromise in terms of numerical error is negligible compared to the gains in terms of scalability.

Overall Complexity: The complexity of FISTA is $\mathcal{O}(Ipn)$, the graph $\mathbb{G}^c$ is $\mathcal{O}(pn\log(n))$, $\mathbb{G}^r$ is $\mathcal{O}(pn\log(p))$ and the final SVD step is $\mathcal{O}(np^2)$. Given that $p < n$, the overall complexity of our algorithm is $\mathcal{O}(pn(\log(n) + I + p + \log(p)))$. Table 6.3 presents the computational complexities of all the models considered in this work (discussed in Section 5.8).

Scalability: The construction of graphs $\mathbb{G}^c$ and $\mathbb{G}^r$ is highly scalable using the approximate k_{nn}-nearest neighbors scheme (FLANN). Next, our proposed FISTA algorithm for FRPCAG requires two important computations at every iteration: 1) computation of proximal operator $\text{prox}_{\lambda h}(\boldsymbol{X})$ and 2) the gradient $\nabla g(\boldsymbol{X})$. The former computation is given by the element-wise soft-thresholding (eq. (5.3)) that can be performed in parallel for all the entries of a matrix. The gradient computation, as given by eq. (5.2), involves matrix-matrix multiplications that involve sparse matrices $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ and can be performed very efficiently in parallel as well. Thus the overall algorithm is highly scalable in every iteration of the algorithm.

5.6 Theoretical Analysis of FRPCAG

Before we start the theoretical analysis of FRPCAG, we first justify the following:

- FRPCAG satisfies the underlying signal model: $\boldsymbol{Y} = \boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k + \boldsymbol{\eta}$, where $\|\bar{\boldsymbol{Z}}\|_F \ll \|\boldsymbol{Z}\|_F$. As $\|\bar{\boldsymbol{Z}}\|_F \neq 0$, so it results in approximate rank k matrix.
- The underlying assumption of the existence of eigen-gap for FRPCAG.

In order to show that FRPCAG satisfies the signal model and results in an approximate rank minimization, we use $\boldsymbol{X} = \boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k, \boldsymbol{L}^c = \boldsymbol{P}^c_k \boldsymbol{\Lambda}^c_k \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^c_k \bar{\boldsymbol{\Lambda}}^c_k \bar{\boldsymbol{P}}^{c\top}_k, \boldsymbol{L}^r = \boldsymbol{P}^r_k \boldsymbol{\Lambda}^r_k \boldsymbol{P}^{r\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{\Lambda}}^r_k \bar{\boldsymbol{P}}^{r\top}_k$ in the equation. We examine each of the terms in the objective function:

$$\phi(\boldsymbol{Y} - \boldsymbol{X}) = \phi(\boldsymbol{Y} - \boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k)$$

$$\begin{aligned}
\text{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) &= \text{tr}\Big((\boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k)(\boldsymbol{P}^c_k \boldsymbol{\Lambda}^c_k \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^c_k \bar{\boldsymbol{\Lambda}}^c_k \bar{\boldsymbol{P}}^{c\top}_k)(\boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k)^\top\Big) \\
&= \text{tr}(\boldsymbol{\Lambda}^c_k \boldsymbol{Z}^\top \boldsymbol{Z} + \bar{\boldsymbol{\Lambda}}^c_k \bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}}) \\
&= \left\|\sqrt{\boldsymbol{\Lambda}^c_k}\boldsymbol{Z}\right\|_F^2 + \left\|\sqrt{\bar{\boldsymbol{\Lambda}}^c_k}\bar{\boldsymbol{Z}}\right\|_F^2,
\end{aligned}$$

where the 2nd step above follows from the fact that $\boldsymbol{P}^{r\top}_k \boldsymbol{P}^r_k = \boldsymbol{I}_k$, $\boldsymbol{P}^{c\top}_k \boldsymbol{P}^c_k = \boldsymbol{I}_k$, $\bar{\boldsymbol{P}}^{c\top}_k \boldsymbol{P}^c_k = \boldsymbol{0}$, $\bar{\boldsymbol{P}}^{r\top}_k \boldsymbol{P}^r_k = \boldsymbol{0}$ and invariance of trace under cyclic permutation. Similarly,

$$\text{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X}) = \left\|\sqrt{\boldsymbol{\Lambda}^r_k}\boldsymbol{Z}\right\|_F^2 + \left\|\sqrt{\bar{\boldsymbol{\Lambda}}^r_k}\bar{\boldsymbol{Z}}\right\|_F^2.$$

Then the transformed optimization problem reads:

$$\min_{\boldsymbol{Z},\bar{\boldsymbol{Z}}} \phi\Big(\boldsymbol{Y} - \boldsymbol{P}^r_k \boldsymbol{Z} \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}^{c\top}_k\Big) + \gamma_c\Big(\left\|\sqrt{\boldsymbol{\Lambda}^c_k}\boldsymbol{Z}\right\|_F^2 + \left\|\sqrt{\bar{\boldsymbol{\Lambda}}^c_k}\bar{\boldsymbol{Z}}\right\|_F^2\Big) + \gamma_r\Big(\left\|\sqrt{\boldsymbol{\Lambda}^r_k}\boldsymbol{Z}\right\|_F^2 + \left\|\sqrt{\bar{\boldsymbol{\Lambda}}^r_k}\bar{\boldsymbol{Z}}\right\|_F^2\Big),$$

which is a weighted Frobenius norm minimization problem. As the graph eigenvalues are sorted in the increasing order, the norm of $\bar{\boldsymbol{Z}}$ is penalized more as compared that of $\boldsymbol{Z}$, resulting in $\|\bar{\boldsymbol{Z}}\|_F \ll \|\boldsymbol{Z}\|_F$.

Next, we justify the underlying assumption behind MLRTG, i.e, the existence of an eigen-gap of the cartesian product graphs governs this model (Section 3.3). In order to reveal the circumstances under which this assumption is satisfied in the FRPCAG model, we study a simpler case. Using $\gamma_c = \gamma_r = \gamma$, we get

$$\min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{X}\|_1 + \gamma\left(\text{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \text{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})\right)$$

Let $\boldsymbol{L} = \boldsymbol{L}^c \times \boldsymbol{L}^r$, where × denotes the cartesian product of graph Laplacians as defined in Section 3.3, the above equation can we re-written as:

$$\min_{\text{vec}(\boldsymbol{X})} \|\text{vec}(\boldsymbol{Y}) - \text{vec}(\boldsymbol{X})\|_1 + \gamma \text{vec}(\boldsymbol{X})^\top \boldsymbol{L} \text{vec}(\boldsymbol{X}),$$

where $\boldsymbol{L} \in \mathbb{R}^{np\times np}$ and $\text{vec}(\boldsymbol{X}) \in \mathbb{R}^{np}$. The above optimization problem tends to recover the vectorized form of $\boldsymbol{X}$, such that it is smooth on the cartesian product graph $\boldsymbol{L}$. Thus, the underlying signal model behind this problem is:

$$\text{vec}(\boldsymbol{X}) \in span(\boldsymbol{P}^c_k \otimes \boldsymbol{P}^r_k),$$

where ⊗ denotes the Kronecker product of Laplacian eigenvectors. Note that unless the eigen-gap assumption for the cartesian product graph holds, the above model will lead to a worse approximation, since higher frequency eigenvectors in $\boldsymbol{P}^c$ and $\boldsymbol{P}^r$ might get involved in the approximation. This validates our claim in the beginning that FRPCAG is a method which implicitly brings into work the GMLSVD framework, in terms of providing a linear combination of graph basis as an intial estimate for the left and right singular vectors.

While the above optimization is a special and more specific case of FRPCAG eq. (5.1), the original model is more general as it allows $\gamma_c \neq \gamma_r$. This is intuitive, since, different low-rank matrices $\boldsymbol{X}$ might portray different levels of smoothness across the row and column graphs which calls for the use of different graph regularization parameters. Therefore, our theoretical analysis deals with this general case.

5.6.1 Main Theorem

Now we are ready to formalize our findings mathematically and prove that any solution of (5.1) yields an approximately low-rank matrix. In fact, we prove this for any proper, positive, convex and lower semi-continuous loss function ϕ (possibly ℓ_p-norms $\|\cdot\|_1, \|\cdot\|_2^2, ..., \|\cdot\|_p^p$).

Before presenting our mathematical analysis we gather a few facts which will be used later. Note that some of these facts were already stated in Chapter 3.

- We assume that the observed data matrix $\boldsymbol{Y}$ satisfies $\boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta}$ where $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ and η models noise/corruptions. Furthermore, for any $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ there exists a matrix $\boldsymbol{C}$ such that $\boldsymbol{X}^* = \boldsymbol{P}^r_k \boldsymbol{C} \boldsymbol{P}^{c\top}_k$.
- $\boldsymbol{L}^c = \boldsymbol{P}^c \boldsymbol{\Lambda}^c \boldsymbol{P}^{c\top} = \boldsymbol{P}^c_k \boldsymbol{\Lambda}^c_k \boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{P}}^c_k \bar{\boldsymbol{\Lambda}}^c_k \bar{\boldsymbol{P}}^{c\top}_k$, where $\boldsymbol{\Lambda}^c_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix of lower eigenvalues and $\bar{\boldsymbol{\Lambda}}^c_k \in \mathbb{R}^{(n-k)\times(n-k)}$ is also a diagonal matrix of higher graph eigenvalues. All values in $\boldsymbol{\Lambda}^c$ are sorted in increasing order, thus $0 = \lambda^c_0 \leq \lambda^c_1 \leq \cdots \leq \lambda^c_k \leq \cdots \leq \lambda^c_{n-1}$. The same holds for $\boldsymbol{L}^r$ as well.
- For a k_{nn}-nearest neighbors graph constructed from a k-clusterable data (along samples / columns) one can expect $\lambda^c_k / \lambda^c_{k+1} \approx 0$ as $\lambda^c_k \approx 0$ and $\lambda^c_k \ll \lambda^c_{k+1}$. The same holds for the graph of features / rows $\boldsymbol{L}^r$ as well.
- For the proof of the theorem, we will use the fact that for any $\boldsymbol{X} \in \Re^{p\times n}$, there exist $\boldsymbol{A} \in \mathbb{R}^{k\times n}$ and $\bar{\boldsymbol{A}} \in \mathbb{R}^{(p-k)\times n}$ such that $\boldsymbol{X} = \boldsymbol{P}^r_k \boldsymbol{A} + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{A}}$, and $\boldsymbol{B} \in \Re^{p\times k}$ and $\bar{\boldsymbol{B}} \in \mathbb{R}^{p\times(n-k)}$ such that $\boldsymbol{X} = \boldsymbol{B}\boldsymbol{P}^{c\top}_k + \bar{\boldsymbol{B}}\bar{\boldsymbol{P}}^{c\top}_k$.

Theorem 5.6.1. *Let $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^{\mu})$, $\gamma > 0$, and $\boldsymbol{\eta} \in \mathbb{R}^{p\times n}$. Any solution $\boldsymbol{X} \in \mathbb{R}^{p\times n}$ of eq.* (5.1) *with $\gamma_c = \gamma/\lambda_{k+1}^c$, $\gamma_r = \gamma/\lambda_{k+1}^r$ and $\boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta}$ satisfies*

$$\phi(\boldsymbol{Y}-\boldsymbol{X}) + \gamma_c \left\| \boldsymbol{X}\bar{\boldsymbol{P}}_k^c \right\|_F^2 + \gamma_r \left\| \bar{\boldsymbol{P}}_k^{r\top} \boldsymbol{X} \right\|_F^2 \leq \phi(\boldsymbol{\eta}) + \gamma \|\boldsymbol{X}^*\|_F^2 \left(\frac{\lambda_k^c}{\lambda_{k+1}^c} + \frac{\lambda_k^r}{\lambda_{k+1}^r} \right). \tag{5.4}$$

where $\lambda_k^c, \lambda_{k+1}^c$ denote the $k, k+1$ eigenvalues of $\boldsymbol{\Lambda}^c$, $\lambda_k^r, \lambda_{k+1}^r$ denote the $k, k+1$ eigenvalues of $\boldsymbol{\Lambda}^r$.

Proof. Please refer to Appendix A.3. ☐

Eq. 5.4 implies that

$$\left\| \boldsymbol{X}\bar{\boldsymbol{P}}_k^c \right\|_F^2 + \left\| \bar{\boldsymbol{P}}_k^{r\top} \boldsymbol{X} \right\|_F^2 \leq \frac{1}{\gamma}\phi(\boldsymbol{\eta}) + \|\boldsymbol{X}^*\|_F^2 \left(\frac{\lambda_k^c}{\lambda_{k+1}^c} + \frac{\lambda_k^r}{\lambda_{k+1}^r} \right).$$

The smaller $\|\boldsymbol{X}\bar{\boldsymbol{P}}_k^c\|_F^2 + \|\bar{\boldsymbol{P}}_k^{r\top}\boldsymbol{X}\|_F^2$ is, the closer $\boldsymbol{X}$ to $\mathbb{MLT}(\boldsymbol{P}_k^{\mu})$ is. The above bound shows that to recover a low-rank matrix one should have large eigen-gaps, i.e, $\lambda_k^c/\lambda_{k+1}^c \approx 0$ and $\lambda_k^r/\lambda_{k+1}^r \approx 0$. This occurs when the rows and columns of Y can be clustered into k clusters (Chapter 3). Furthermore, one should also try to choose a metric ϕ (or ℓ_p-norm) that minimizes $\phi(\boldsymbol{\eta})$.

The theoretical analysis of FRPCAG reveals that the method can recover low-rank representation for a dataset $\boldsymbol{Y}$ that is k clusterable across its rows / features and columns / samples. This is not strange, as we show in the next section that FRPCAG is able to successfully recover meaningful clusters not only across samples but features as well.

5.6.2 Why FRPCAG gives a low-rank solution? Implications of the Theoretical Analysis

This section constitutes of a more formal and theoretical discussion of FRPCAG. We base our explanations on two arguments:

1. FRPCAG penalizes the the singular values of the data matrix. This can be viewed in two ways: a) In the data domain via SVD analysis and b) In the graph fourier domain [1].
2. FRPCAG is a subspace rotation / alignment method, where the left and right singular vectors of the resultant low-rank matrix are being aligned with the singular vectors of the clean data.

FRPCAG is a dual graph filtering / singular value penalization method

In order to demonstrate how FRPCAG penalizes the singular values of the data we study another way to cater the graph regularization in the solution of the optimization problem which is contrary to the one presented in Section 5.4. In Section 5.4 we used a gradient for the graph regularization terms $\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$ and used this gradient as an argument of the proximal operator for the soft-thresholding. What we did not point out there was that the solution of the graph regularizations can also be computed by proximal operators. It is due to the reason that using proximal operators

---

[1] Here we assume that the data is stationary

for graph regularization (that we present here) is more computationally expensive. Assume that the proximal operator of $\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top)$ is computed first, and let $\boldsymbol{Z}$ be a temporary variable, then it can be written as:

$$\min_{\boldsymbol{Z}} \|\boldsymbol{Y}-\boldsymbol{Z}\|_F^2 + \gamma_c \operatorname{tr}(\boldsymbol{Z}\boldsymbol{L}^c\boldsymbol{Z}^\top)$$

The above equation has a closed form solution which is given as:

$$\boldsymbol{Z} = \boldsymbol{Y}(\boldsymbol{I}+\gamma_c\boldsymbol{L}^c)^{-1}$$

Now, compute the proximal operator for the term $\gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$

$$\min_{\boldsymbol{X}} \|\boldsymbol{Z}-\boldsymbol{X}\|_F^2 + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$$

The closed form solution of the above equation is given as:

$$\boldsymbol{X} = (\boldsymbol{I}+\gamma_r\boldsymbol{L}^r)^{-1}\boldsymbol{Z}$$

Thus, the low-rank matrix $\boldsymbol{X}$, in every iteration can be written as:

$$\boldsymbol{X} = (\boldsymbol{I}+\gamma_r\boldsymbol{L}^r)^{-1}\boldsymbol{Y}(\boldsymbol{I}+\gamma_c\boldsymbol{L}^c)^{-1}$$

after this the soft thresholding can be applied on $\boldsymbol{X}$.

Let the SVD of $\boldsymbol{Y}$, $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, $\boldsymbol{L}^c = \boldsymbol{P}^c\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top}$ and $\boldsymbol{L}^r = \boldsymbol{P}^r\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top}$, then we get:

$$\begin{aligned}\boldsymbol{X} &= (\boldsymbol{I}+\gamma_r\boldsymbol{P}^r\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top})^{-1}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top(\boldsymbol{I}+\gamma_c\boldsymbol{P}^c\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top})^{-1}\\ &= \boldsymbol{P}^r(\boldsymbol{I}+\gamma_r\boldsymbol{\Lambda}^r)^{-1}\boldsymbol{P}^{r\top}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c(\boldsymbol{I}+\gamma_c\boldsymbol{\Lambda}^c)^{-1}\boldsymbol{P}^{c\top}\end{aligned}$$

thus, the singular values $\boldsymbol{\Sigma}$ of $\boldsymbol{Y}$ are penalized proportional to $1/(\boldsymbol{I}+\gamma_c\boldsymbol{\Lambda}^c)(\boldsymbol{I}+\gamma_r\boldsymbol{\Lambda}^r)$ in every iteration of the algorithm. Clearly, the above solution requires the computation of two inverses which can be computationally intractable for big datasets.

The singular value penalization effect is also shown in Fig. 5.11, where the green vectors (scaled by their singular values) as learned via our model tend to shrink with the increasing regularization parameter.

FRPCAG is a penalization method in the spectral graph fourier domain

Another way to analyze FRPCAG is to look at the penalization $\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$ in the spectral graph Fourier domain. For a signal $\boldsymbol{x}$ lying on a graph $\mathbb{G}$ with a combinatorial Laplacian $\boldsymbol{L}$, its graph Fourier transform is given as $\boldsymbol{P}^\top x$, where $\boldsymbol{P}$ is the set of Laplacian eigenvectors [20]. Using this, it is possible to rewrite the graph penalization terms as a spectral penalization:

$$\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X}) = \gamma_c\|\sqrt{\boldsymbol{\Lambda}^c}\hat{\boldsymbol{X}}\|_F^2 + \gamma_r\|\sqrt{\boldsymbol{\Lambda}^r}\hat{\boldsymbol{X}}^\top\|_F^2. \tag{5.5}$$

The signal is thus penalized in the spectral graph domain with a weight proportional to the squared root of the graph eigenvalue. As a result, the signal is pushed towards the lowest frequencies of both graphs, enforcing its low-rank structure. This interpretation allows us to clearly identify the role of the regularization constants γ_c and γ_r. With big constants γ_c and γ_r, the resulting signal is close to low-rank but we also attenuate the content inside the band as illustrated in Fig. 5.11.

FRPCAG is a weighted subspace alignment method

Let $\boldsymbol{Y} = \boldsymbol{U\Sigma V}^\top$ be the SVD of $\boldsymbol{Y}$ and suppose that we use $\boldsymbol{Y}$ as an initialization of FRPCAG, then, we have:

$$\begin{aligned}
\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X}) =& \gamma_c \operatorname{tr}(\boldsymbol{U\Sigma V}^\top\boldsymbol{P}^c\boldsymbol{L}^c\boldsymbol{P}^{c\top}\boldsymbol{V\Sigma U}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{V\Sigma U}^\top\boldsymbol{P}^r\boldsymbol{L}^r\boldsymbol{P}^{r\top}\boldsymbol{U\Sigma V}^\top) \\
=& \gamma_c \operatorname{tr}(\boldsymbol{\Sigma V}^\top\boldsymbol{P}^c\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top}\boldsymbol{V\Sigma}) + \gamma_r \operatorname{tr}(\boldsymbol{\Sigma U}^\top\boldsymbol{P}^r\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top}\boldsymbol{U\Sigma}) \qquad (5.6)\\
=& \sum_{i,j=1}^{\min\{n,p\}} \sigma_i^2(\gamma_c\lambda_j^c(\boldsymbol{V}_i^\top\boldsymbol{P}_j^1)^2 + \gamma_r\lambda_j^r(\boldsymbol{U}_i^\top\boldsymbol{P}_j^2)^2) \\
=& \sum_{i=1}^{\min\{n,p\}} \sigma_i^2\left(\gamma_c\left(\sum_{j=1}^{n}\lambda_j^c(\boldsymbol{V}_i^\top\boldsymbol{P}_j^c)^2\right) + \gamma_r\left(\sum_{j=1}^{p}\lambda_j^r(\boldsymbol{U}_i^\top\boldsymbol{P}_j^r)^2\right)\right),
\end{aligned}$$

where λ_j^c and λ_j^r are the eigenvalues in the matrices $\boldsymbol{\Lambda}^c$ and $\boldsymbol{\Lambda}^r$ respectively. The second step follows from $\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}$ and $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$ and the cyclic permutation invariance of the trace. In the standard terminology $\boldsymbol{V}_i$ and $\boldsymbol{U}_i$ are the principal components and principal directions of of the low-rank matrix $\boldsymbol{X}$. From the above expression, the minimization is carried out with respect to the singular values σ_i and the singular vectors $\boldsymbol{V}_i, \boldsymbol{U}_i$. The minimization has the following effect:

1. Minimize σ_i by performing a penalization with the graph eigenvalues as explained earlier.

2. When σ_i is big, the principal components $\boldsymbol{V}_i$ are more well aligned with the graph eigenvectors $\boldsymbol{P}_j^c$ for small values of λ_j^c, i.e, the lower graph frequencies of $\boldsymbol{L}^c$ as compared to the $\boldsymbol{P}_j^c$ for higher λ_j^c. The principal directions $\boldsymbol{U}_i$ are also more aligned with the graph eigenvectors $\boldsymbol{P}_j^r$ for small values of λ_j^r, i.e, the lower graph frequencies of $\boldsymbol{L}^r$ as compared to higher frequencies. This alignment makes sense as the higher eigenvalues correspond to the higher graph frequencies which constitute the noise in data.

FRPCAG is a weighted GSC minimizer

Eq. (5.6) can be re-written as:

$$\begin{aligned}
&= \gamma_c \operatorname{tr}(\boldsymbol{P}^c\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top}\boldsymbol{V\Sigma}^2\boldsymbol{V}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{P}^r\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top}\boldsymbol{U\Sigma}^2\boldsymbol{U}^\top) \\
&= \gamma_c \operatorname{tr}(\boldsymbol{P}^c\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top}\boldsymbol{\Omega}^c) + \gamma_r \operatorname{tr}(\boldsymbol{P}^r\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top}\boldsymbol{\Omega}^r) \\
&= \gamma_c \operatorname{tr}(\boldsymbol{\Lambda}^c\boldsymbol{P}^{c\top}\boldsymbol{\Omega}^1\boldsymbol{P}^c) + \gamma_r \operatorname{tr}(\boldsymbol{\Lambda}^r\boldsymbol{P}^{r\top}\boldsymbol{\Omega}^2\boldsymbol{P}^r) \\
&= \gamma_c \operatorname{tr}(\boldsymbol{\Lambda}^c\boldsymbol{\Gamma}^c) + \gamma_r \operatorname{tr}(\boldsymbol{\Lambda}^r\boldsymbol{\Gamma}^r) \\
&= \gamma_c \sum_i \lambda_i^c\boldsymbol{\Gamma}_{i,i}^c + \gamma_r \sum_j \lambda_i^r\boldsymbol{\Gamma}_{j,j}^r,
\end{aligned}$$

where $\boldsymbol{\Omega}^c, \boldsymbol{\Omega}^r$ are the sample and feature covariance matrices and $\boldsymbol{\Gamma}^c, \boldsymbol{\Gamma}^r$ are the sample and feature Graph Spectral Covariance matrices as defined in eq. (3.3). Assuming the entries of $\boldsymbol{\Lambda}^c, \boldsymbol{\Lambda}^r$ are sorted in the increasing order, the last equation above corresponds to a weighted minimization of the diagonal entries of $\boldsymbol{\Gamma}^c$ and $\boldsymbol{\Gamma}^r$. Thus, the lower diagonal entries undergo more penalization than the top ones. Now assume that there is a spectral gap, i.e, $\lambda_k^c \ll \lambda_{k+1}^c$ and $\lambda_k^r \ll \lambda_{k+1}^r$, then the minimization above leads to a higher penalization of all the entries above k in $\boldsymbol{\Gamma}^c$ and $\boldsymbol{\Gamma}^r$. This automatically leads to a maximization of eq. (3.4).

FRPCAG is a bi-clustering method

In the biclustering problem, one seeks to simultaneously cluster samples and features. Biclustering has applications in a wide variety of domains, ranging from text mining to collaborative filtering [105]. Very briefly, a typical bi-clustering method reveals:

1. the clusters across the rows and columns of a data matrix.
2. the checkerboard pattern hidden in a noisy dataset.

Our theoretical analysis reveals that FRPCAG is able to recover a low-rank representation for a matrix that is simultaneously clusterable across the rows and columns, therefore, FRPCAG is a bi-clustering method.

5.6.3 Conclusion

From the above detailed analysis, we can conclude the following facts about FRPCAG.

- FRPCAG recovers a low-rank matrix via a non-linear penalization of singular values, instead of a “thresholding”. This non-linear penalization depends on the graph eigenvalues and has a scaling effect, as demonstrated in the experiments later.
- FRPCAG recovers low-rank matrix by assuming that the rows and columns of $\boldsymbol{X}$ belong to the complete set of graph eigenvectors, i.e,

  $$\boldsymbol{X}(i) \in span(\boldsymbol{P}^r) \quad \text{and} \quad \boldsymbol{X}^\top(i) \in span(\boldsymbol{P}^c) \quad \forall\, i.$$

  This is intuitive, as unlike the RPCA, FRPCAG does not involve a soft-thresholding of singular values. The non-linear singular value thresholding effect only leads to an approximate low-rank matrix, as the lowest singular values never reach zero. The rank is indirectly controlled by the regularization parameters γ_c and γ_r. More formally, FRPCAG relies on approximate MLRTG framework, i.e,

  $$\boldsymbol{X} = \boldsymbol{P}^r_k \boldsymbol{A} + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{A}},$$

  where $\|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F$.

5.7 Studying the model: Working Examples of FRPCAG

We presented several different implications of FRPCAG in the previous section. These implications are justified with some artificial and real world datasets in this section.

5.7.1 Low-rank recovery, singular value penalization and subspace alignment

We generate an approximate MLRTG $\boldsymbol{Y}_0$ of size 300 × 300 and rank $r = 10$ using the method 3 from Section 3.5.1. Then, we corrupt $\boldsymbol{Y}_0$ with Laplacian noise with standard deviation 0.1 to get $\boldsymbol{Y}$. Finally we perform FRPCAG on noisy $\boldsymbol{Y}$ to recover the low-rank matrix $\boldsymbol{X}$ for three different regularization parameter settings: $\gamma_c = \gamma_r = 0.01, 1, 100$ using 10-nearest neighbors graphs between the rows $\mathbb{G}^r$ and columns $\mathbb{G}^c$ of $\boldsymbol{Y}$. Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the SVD of low-rank $\boldsymbol{X}$ and $\boldsymbol{Y}_0 = \boldsymbol{U}_0\boldsymbol{\Sigma}_0\boldsymbol{V}_0^\top$ be the SVD of clean data. To study the error of approximation $\|\boldsymbol{Y}_0 - \boldsymbol{X}\|_F^2 / \|\boldsymbol{Y}_0\|_F^2$, we plot the low-rank $\boldsymbol{X}$, $\boldsymbol{\Gamma}^c, \boldsymbol{\Gamma}^r$, low-frequency power concentrations $\hat{s}(\boldsymbol{\Gamma}^c)$, $\hat{s}(\boldsymbol{\Gamma}^r)$, the singular values $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$, the coherency between $(\boldsymbol{U}, \boldsymbol{U}_0)$, $(\boldsymbol{V}, \boldsymbol{V}_0)$ in Fig. 5.2. Following conclusions can be drawn:

1. The lowest error occurs for $\gamma_c = \gamma_r = 4$ (it is possible to get a lower error with finer parameter tuning but the purpose here is to study the effect of regularization only).
2. For this parameter setting, the singular values $\boldsymbol{\Sigma}$ decay to 0 after 10.
3. An interesting observation is the coherence between $(\boldsymbol{U}, \boldsymbol{U}_0)$ and $(\boldsymbol{V}, \boldsymbol{V}_0)$ for increasing regularization. Clearly, for small parameters, these pairs are incoherent. The 1^{st} 9 vectors of $\boldsymbol{U}$ and $\boldsymbol{V}$ are aligned with the 1^{st} 9 vectors of $\boldsymbol{U}_0$ and $\boldsymbol{V}_0$ for the best parameter $\gamma_c = \gamma_r = 4$. The coherency of the pairs of singular vectors decreases again for higher parameters and the thresholding on the singular values is more than required. The *low-frequency power concentration* also shows a very high degree of alignment between the eigenvectors of graphs $\boldsymbol{P}^c, \boldsymbol{P}^r$ and those of the covariance matrices $\boldsymbol{C}^c, \boldsymbol{C}^r$.

For the best parameters $\gamma_c = \gamma_r = 4$, there is a strong alignment between the 1^{st} 9 singular vectors. Furthermore, thresholding on the 10^{th} singular value makes it slightly smaller than that of the clean data Y_0 which results in a miss-alignment of the 10^{th} singular vector.

5.7.2 Clustering, singular value penalization and subspace alignment

The purpose of this example is to show the following effects of FRPCAG:

1. The model recovers an approximate low-rank representation.
2. The principal components $\boldsymbol{V}$ and principal directions $\boldsymbol{U}$ of $\boldsymbol{X}$ (assuming $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$) align with the first few eigenvectors of their respective graphs, automatically revealing a low-rank and enhanced class structure, when the data belongs to multiple clusters.
3. The singular values of the low-rank matrix obtained using our model closely approximate those obtained by nuclear norm based models even in the presence of corruptions.

Our justification relies mostly on the quality of the singular values of the low-rank representation and the alignment of the singular vectors with their respective graph eigenvectors.

Experiment on MNIST dataset

We perform an experiment with 1000 samples of the MNIST dataset belonging to two different classes (digits 0 and 1). We vectorize all the digits and form a data matrix $\boldsymbol{Y}$ whose columns contain the digits. Then we compute a 10-nearest neighbor graph of samples between the columns of $\boldsymbol{Y}$ and a graph of features between the rows of $\boldsymbol{Y}$. We determine the low-rank matrix $\boldsymbol{X}$ by solving eq. (5.1) and perform one SVD at the end $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Finally, we do the clustering by performing k-means (k = 2) on the low-rank $\boldsymbol{X}$. As argued in [46], if the data is arranged according to the classes, the matrix $\boldsymbol{V}\boldsymbol{V}^\top$ (where $\boldsymbol{V}$ are the principal components of the data) reveals the subspace structure. The matrix $\boldsymbol{V}\boldsymbol{V}^\top$ is also known as the shape interaction matrix (SIM) [106]. If the subspaces are orthogonal then SIM

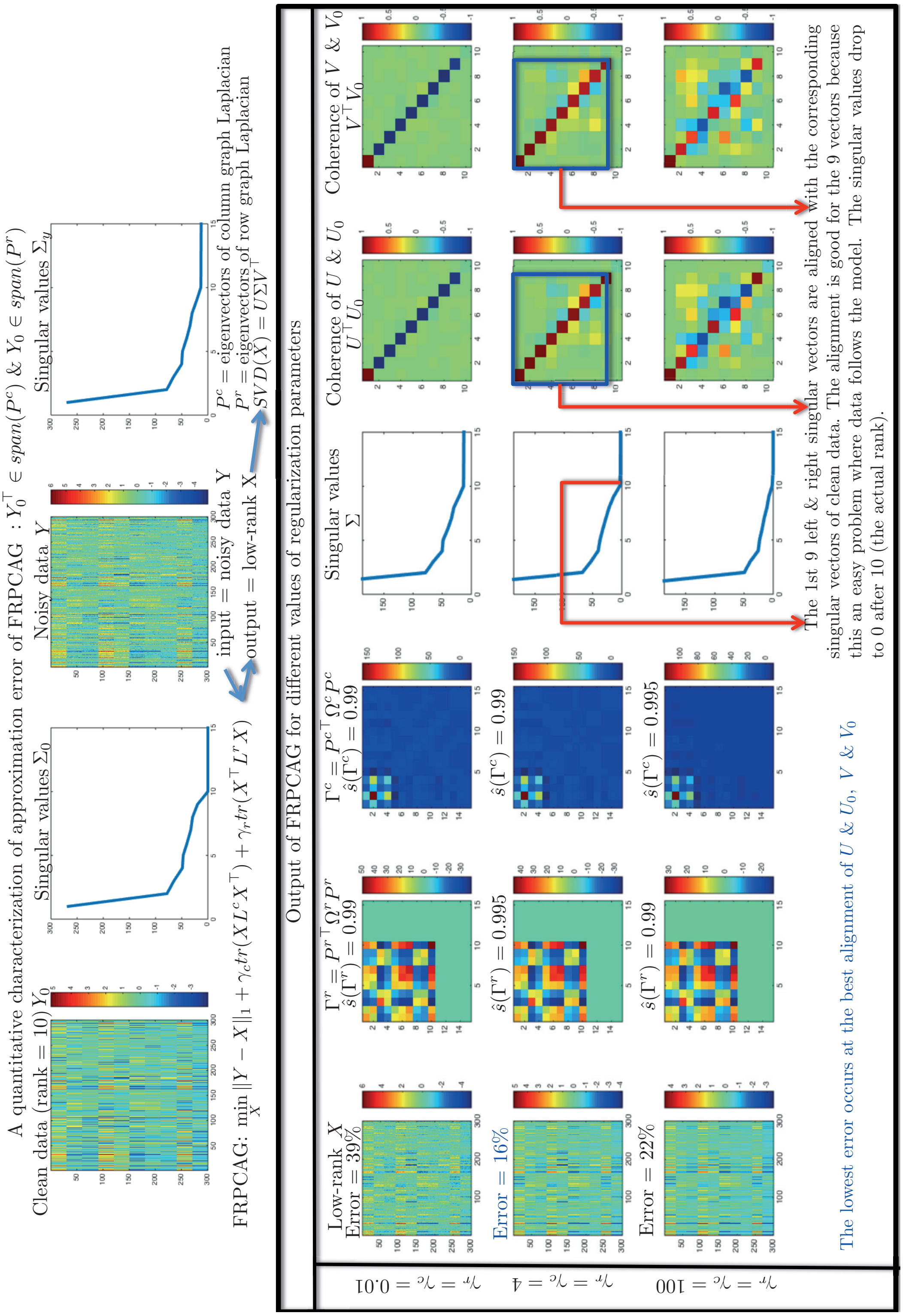


Figure 5.2 – A study of the error of approximation $\|\boldsymbol{Y}_0 - \boldsymbol{X}\|_F^2 / \|\boldsymbol{Y}_0\|_F^2$ for a data generated from graph eigenvectors. We plot the low-rank matrix $\boldsymbol{X}$, the alignment order $s(\boldsymbol{\Gamma}^r)$, $s(\boldsymbol{\Gamma}^c)$ and $\hat{s}(\boldsymbol{\Gamma}^c)$, $\hat{s}(\boldsymbol{\Gamma}^r)$ ($k = 10$), the singular values $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$, the coherency between $(\boldsymbol{U}, \boldsymbol{U}_0)$, $(\boldsymbol{V}, \boldsymbol{V}_0)$

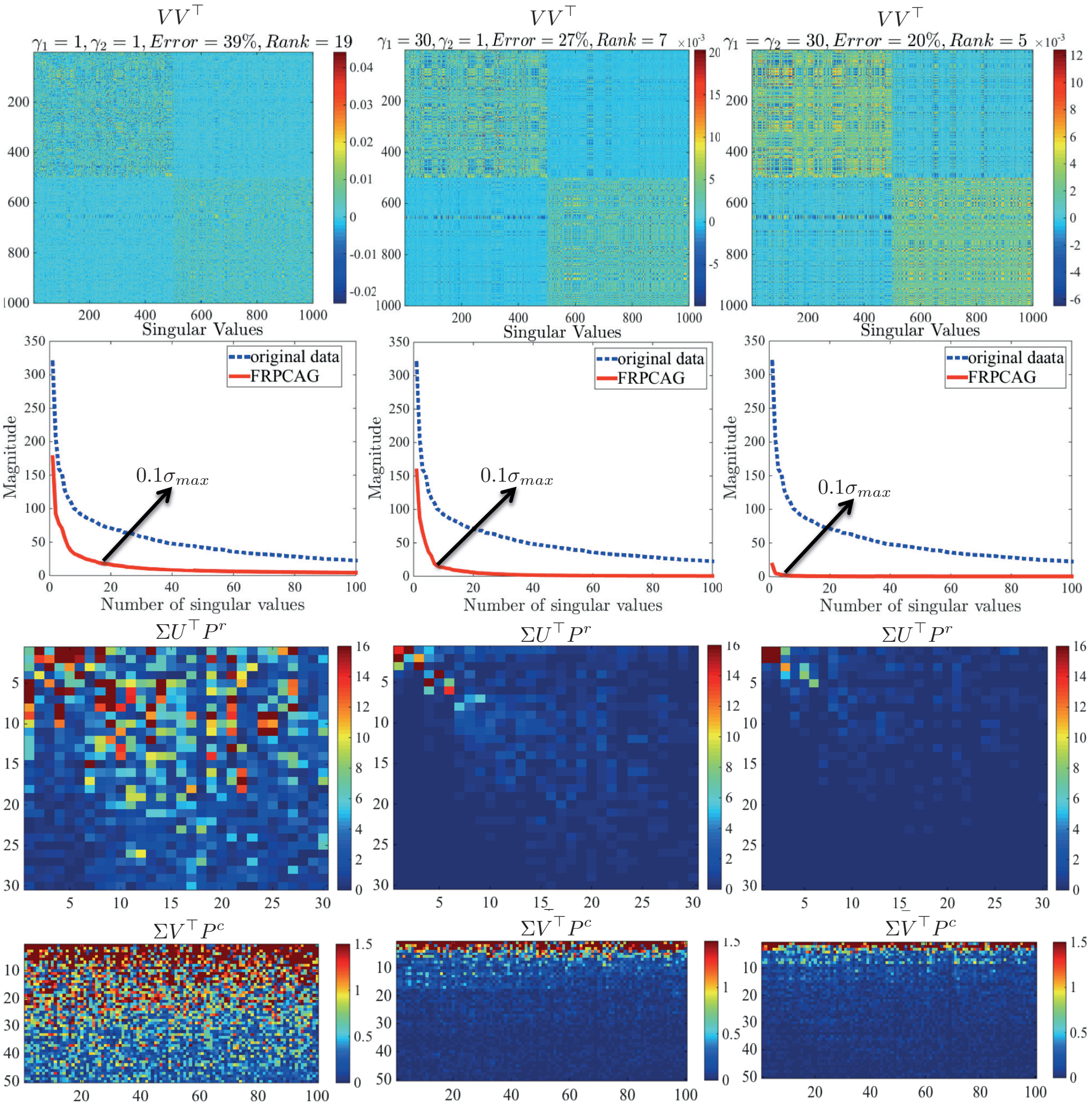


Figure 5.3 – The matrices $\boldsymbol{V}\boldsymbol{V}^\top$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$, $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ and the corresponding clustering errors obtained for different values of the weights on the two graph regularization terms for 1000 samples of MNIST dataset (digits 0 and 1). The block diagonal structure of $\boldsymbol{V}\boldsymbol{V}^\top$ becomes more clear by increasing γ_c and γ_r with a thresholding of the singular values in $\boldsymbol{\Sigma}$. Furthermore, the sparse structures of $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$ and $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ towards the rightmost corners show that the number of left and right singular vectors (weighted by singular values) which align with the eigenvectors of the Laplacians $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ go on decreasing with increasing γ_c and γ_r. This shows that the two graphs help in attaining a low-rank structure with a low clustering error.

should acquire a block diagonal structure. Furthermore, our model eq. (5.1) tends to align the first few principal components $\boldsymbol{V}_i$ and principal directions $\boldsymbol{U}_i$ of $\boldsymbol{X}$ to the first few eigenvectors $\boldsymbol{P}^c_j$ and $\boldsymbol{P}^r_j$ of $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ for larger σ_i respectively. Thus, it is interesting to observe the matrices $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$ and $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ scaled with the singular values $\boldsymbol{\Sigma}$ of the low-rank matrix $\boldsymbol{X}$, as justified by eq. (5.6). This scaling takes into account the importance of the eigenvectors that are associated to bigger singular values.

Fig. 5.3 plots the matrix $\boldsymbol{V}\boldsymbol{V}^\top$, the corresponding clustering error, the matrices $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{P}^c$, and $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{P}^r$ for different values of γ_c and γ_r from left to right. Increasing γ_c and γ_r from 1 to 30 leads to 1) the penalization of the singular values in $\boldsymbol{\Sigma}$ resulting in a lower rank matrix 2) alignment of the first few principal components $\boldsymbol{V}_i$ and principal directions $\boldsymbol{U}_i$ in the direction of the first few eigenvectors $\boldsymbol{P}^c_j$ and $\boldsymbol{P}^r_j$ of $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ respectively 3) an enhanced subspace structure in $\boldsymbol{V}\boldsymbol{V}^\top$ and 4) a lower clustering error. Together the two graphs help in acquiring a low-rank structure that is suitable for clustering applications as well.

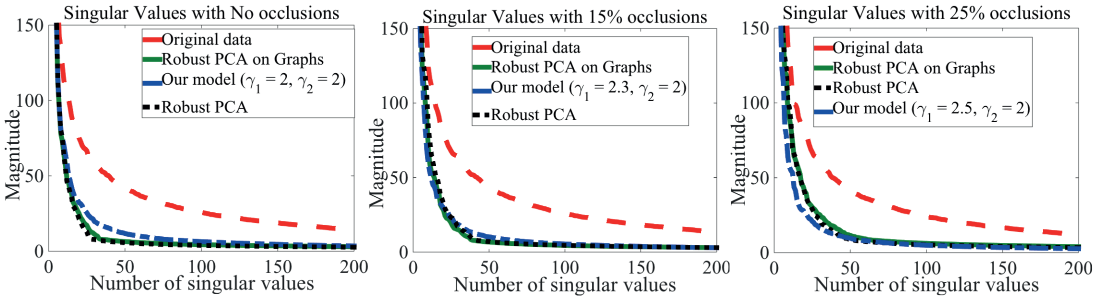


Figure 5.4 – A comparison of singular values of the low-rank matrix obtained via our model, RPCA and RPCAG. The experiments were performed on the ORL dataset with different levels of block occlusions. The parameters corresponding to the minimum validation clustering error for each of the model were used.

Experiment on ORL dataset with noise

Next we demonstrate that for data with or without corruptions, FRPCAG is able to acquire singular values as good as the nuclear norm based models, RPCA [1] and RPCAG [99]. We perform three clustering experiments on 30 classes of ORL dataset with no block occlusions, 15% block occlusions and 25% block occlusions. Fig. 5.4 presents a comparison of the singular values of the original data with the singular values of the low-rank matrix obtained by solving RPCA, RPCAG and our model. The parameters for all the models are selected corresponding to the lowest clustering error for each model. It is straightforward to conclude that the singular values of the low-rank representation using our fast method closely approximate those of the nuclear norm based models irrespective of the level of corruptions.

5.7.3 Bi-clustering Example on ORL dataset

We perform a small experiment with 50 faces from ORL dataset, corresponding to 5 different classes. We take the samples corresponding to 5 different faces and pre-process the dataset to zero mean across the features and run FRPCAG with $\gamma_c = 3 = \gamma_r = 3$. Then, we cluster the low-rank matrix $\boldsymbol{X}$ to 5 clusters (number of classes in ORL) across the columns and 5 clusters across the features. While, clustering across samples / columns has a straight-forward implication and will be discussed in detail in the next section of this chapter, we discuss the feature clustering in detail here. Fig. 5.5 shows the 5 clusters of the features of the ORL dataset for one face of each of the 5 different types of faces. It is interesting to note that clusters across the features correspond to very localized regions of the faces. For example, the first cluster (first row of Fig. 5.5) corresponds to the region of eyes and upper lips while the fifth cluster corresponds mostly to the smooth portion of the face (cheeks and chin). Of course, as FRPCAG is a low-rank recovery method which exploits the similarity information of the rows and columns, it also reveals the checkerboard pattern of the ORL dataset. This checkerboard pattern is visible in Fig. 5.6. This pattern will become more visible for larger regularization parameters γ_c and γ_r.

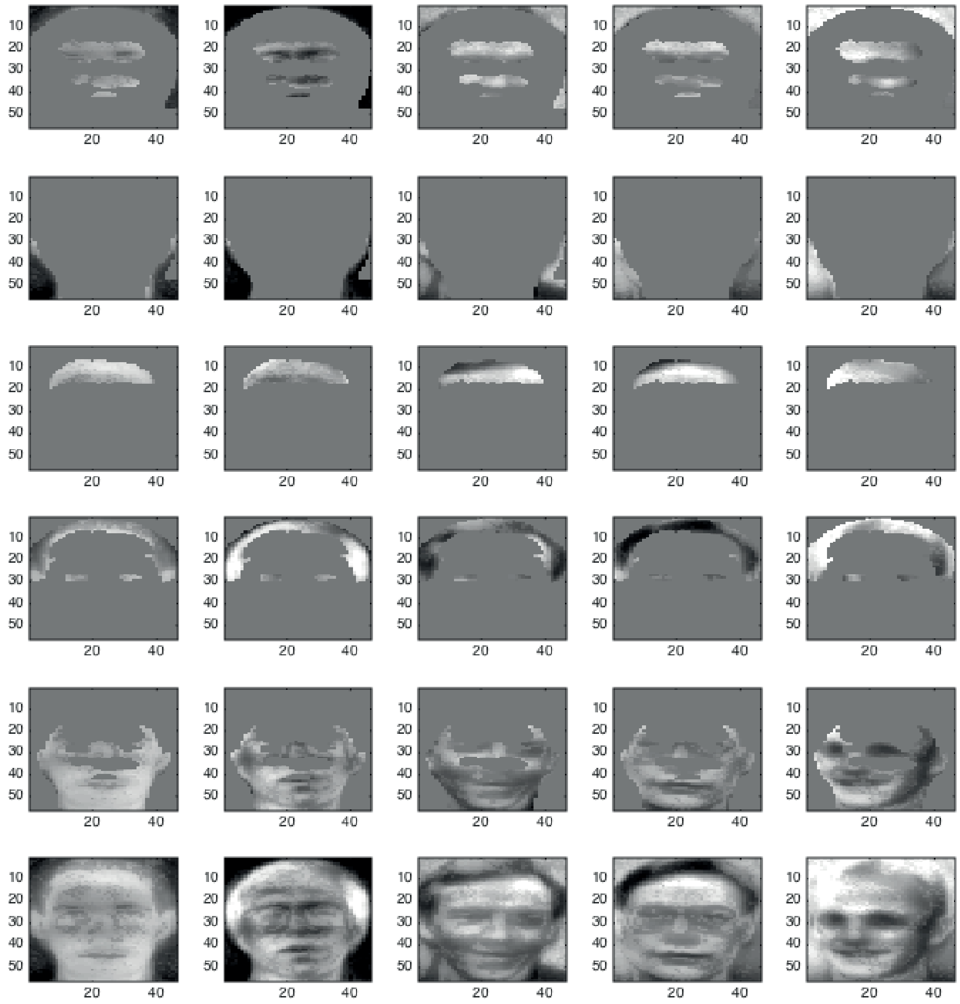


Figure 5.5 – The clustering of features for ORL dataset. The top five rows show the five different feature clusters across five different faces. The last row shows the actual face which is a sum of all the feature clusters.

5.8 Experiments

Experiments were done using two open-source toolboxes: the UNLocBoX [107] for the optimization part and the GSPBox [62] for the graph creation. We perform two types of experiments corresponding to two applications of PCA.

1. Data clustering in low-dimensional space.

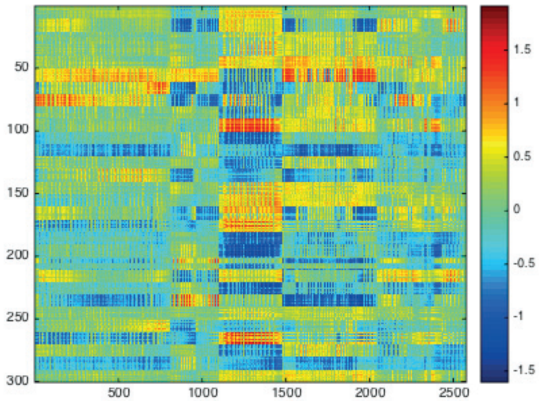


Figure 5.6 – The checkerboard pattern of the low-rank representation of ORL dataset, obtained by FRPCAG.

2. Low-rank and sparse decomposition: Static background separation from dynamic foreground.

We perform our clustering experiments on 7 benchmark databases: CMU PIE, ORL, YALE, COIL20, MNIST, USPS and MFEAT. CMU PIE, ORL and YALE are face databases with small pose variations. COIL20 is a dataset of objects with significant pose changes so we select images for each object with less than 45 degrees of pose change. USPS and MNIST contain images of handwritten digits and MFeat consists of features extracted from handwritten numerals. The details of all the datasets used are provided in Table 5.1.

Table 5.1 – Details of the datasets used for clustering experiments in this work.

Dataset	Samples	Dimension	Classes
CMU PIE	1200	32×32	30
ORL	400	56×46	40
COIL20	1400	32×32	20
YALE	165	32×32	11
MNIST	50000	28×28	10
USPS	3500	16×16	10
MFEAT	400	409	10

In order to evaluate the robustness of our model to gross corruptions we corrupt the datasets with two different types of errors 1) block occlusions and 2) random missing pixels. Block occlusions of three different sizes, i.e, 15%, 25% and 40% of the total size of the image are placed uniformly randomly in all the images of the datasets. Similarly, all the images of the datasets are also corrupted by removing 10%, 20%, 30% and 40% pixels uniformly randomly. Separate clustering experiments are performed for each of the different types of corruptions.

We compare the clustering performance of our model with 10 other models including the state-of-art: 1) k-means on original data 2) Normalized Cut (NCut) [108] 3) Laplacian Eigenmaps (LE) [2] 4) Standard PCA 5) Graph Laplacian PCA (GLPCA) [3] 6) Manifold Regularized Matrix Factorization (MMF) [6] 7) Non-negative Matrix Factorization (NMF) [4] 8) Graph Regularized Non-negative Matrix Factorization (GNMF) [5] 9) Robust PCA (RPCA) [1] and 10) Robust PCA on Graphs (RPCAG) [99].

For ORL, CMU PIE, COIL20, YALE and USPS datasets we compare three different versions of our model corresponding to the three types of graphs $\mathbb{G}^c$ and $\mathbb{G}^r$. FRPCAG(A) corresponds to our model using sample and feature graphs constructed using the default FLANN settings, FRPCAG(B) to the case using a sample graph constructed from the default FLANN setting and feature graph whose nearest neighbor search phase was made more approximate by making the algorithm highly parallel, and FRPCAG(C) to the case where the nearest neighbor search algorithm for both the sample and feature graphs was made highly parallel to compromise the approximation even further. The degree of parallelism of FLANN was changed by tuning certain input parameters in the FLANN library. All other models for these datasets are evaluated using a good quality sample graph $\mathbb{G}^c$ and good quality feature graph $\mathbb{G}^r$, i.e using the default FLANN settings.

Due to the large size of the MNIST dataset and to have a fair comparison with other algorithms, we construct both graphs by using high degree of parallelism in FLANN. We call such a graph as 'noisy' to differentiate it from the one constructed using default FLANN setting. Thus, the experiments on MNIST dataset are kept separate from the rest of the datasets to emphasize the difference in the graph construction strategy. We also perform a separate set of experiments on the ORL dataset and compare the performance of our model with state-of-the-art nuclear norm based models, RPCA [1] and RPCAG [99], both with a good quality and noisy graph. We perform this set of experiments only on the ORL dataset (due to its small size) as the nuclear norm based models are computationally expensive. Finally, the experiments on MFeat dataset are only performed with missing values because block occlusions in non-image datasets correspond to an unrealistic assumption. The computational complexities of all these models are presented in Table 6.3.

Pre-processing: All datasets are transformed to zero-mean and unit standard deviation along the features for the RPCA, RPCAG and FRPCAG. For MMF the samples are additionally normalized to unit-norm. For NMF and GNMF only the unit-norm normalization is applied to all the samples of the dataset.

Evaluation: We use *clustering error* as a metric to compare the clustering performance of various models. NCut, LE, PCA, GLPCA, MMF, NMF and GNMF are matrix factorization models that explicitly learn the principal components $\boldsymbol{V}$. The clustering error for these models is evaluated by performing k-means on the principal components. RPCA, RPCAG and FRPCAG learn the low-rank matrix $\boldsymbol{X}$. The clustering error for these models can be evaluated by performing k-means on 1) principal components $\boldsymbol{V}$ obtained by the SVD of the low-rank matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ or 2) the low-rank $\boldsymbol{X}$ directly. Note that RPCA and RPCAG determine the exact low-rank representation $\boldsymbol{X}$, whereas our model only shrinks singular values and therefore only recovers an approximate low-rank representation $\boldsymbol{X}$. Thus, if one desires to use the principal components $\boldsymbol{V}$ for clustering, the dimension of the subspace (number of columns of $\boldsymbol{V}$) can be decided by selecting the number of singular values greater than a particular threshold. However, this procedure requires an SVD and can be expensive for big datasets. Instead, it is more feasible to perform clustering on the low-rank $\boldsymbol{X}$ directly. We observed that similar clustering results are obtained by using either $\boldsymbol{V}$ or $\boldsymbol{X}$, however, for brevity these results are not reported. Due to the non-deterministic nature of k-means, it is run 10 times and the minimum error over all runs is reported.

Parameter selection for various models: Each model has several parameters which have to be selected in the validation stage of the experiment. To perform a fair validation for each of the models we use a range of parameter values as presented in Table 5.2. For a given dataset, each of the models is run for each of the parameter tuples in this table and the parameters corresponding to minimum clustering error are selected for testing purpose. Furthermore, GLPCA, NMF and GNMF are non-convex models so they are run 10 times for each of the parameter tuple. All the other models are convex so they are run only once.

Table 5.2 – Range of parameter values for each of the models considered in this work. r is the rank or dimension of subspace, λ is the weight associated with the sparse term for Robust PCA framework [1] and γ is the parameter associated with the graph regularization term.

Model	Parameters	Parameter Range
NCut [108]	r	$r \in \{2^1, 2^2, \cdots, \min(n,p)\}$
LE [2]		
PCA		
GLPCA [3]	r, γ, β	$r \in \{2^1, 2^2, \cdots, \min(n,p)\}$ $\gamma \implies \beta$ using [3] $\beta \in \{0.1, 0.2, \cdots, 0.9\}$
MMF [6]	r, γ	$r \in \{2^1, 2^2, \cdots, \min(n,p)\}$ $\gamma \in \{2^{-3}, 2^{-2}, \cdots, 2^{10}\}$
NMF [4]	r	
GNMF [5]	r, γ	
RPCA [1]	λ	$\lambda \in \{\frac{2^{-3}}{\sqrt{\max(n,p)}} : 0.1 : \frac{2^3}{\sqrt{\max(n,p)}}\}$ $\gamma \in \{2^{-3}, 2^{-2}, \cdots, 2^{10}\}$
RPCAG [99]	λ, γ	
FRPCAG	γ_r, γ_c	$\gamma_r, \gamma_c \in \{1, 2, \cdots, 100\}$
CPCA	γ_r, γ_c	$\gamma_r, \gamma_c \in (0, 30)$
	r (approximate decoder)	$\Sigma_{r,r} / \Sigma_{1,1} < 0.1$

Parameter selection for Graphs: For all the experiments reported in this paper we use the following parameters for graphs $\mathbb{G}^c$ and $\mathbb{G}^r$: $k_{nn} = 10$ and σ^2 is automatically set to the average of the nearest neighbors. It is important to point out here that different types of data might call for slightly different parameters for graphs. However, for a given dataset, the use of same graph parameters (same graph quality) for all the graph regularized models ensures a fair comparison.

5.8.1 Clustering

Comparison with Matrix Factorization Models

Fig. 5.7 presents the clustering error for various matrix factorization and our proposed models. NMF and GNMF are not evaluated for the USPS and MFeat datasets as they are not originally non-negative. It can be seen that our proposed model FRPCAG(A) with the two good quality graphs performs better than all the other models in most of the cases both in the presence and absence of data corruptions. Even FRPCAG(B) with a good sample graph $\mathbb{G}^c$ and a noisy feature graph $\mathbb{G}^r$ performs reasonably well. This shows that our model is quite robust to the quality of graph $\mathbb{G}^r$. However, as expected, FRPCAG(C) performs worse for ORL, CMU PIE, COIL20, YALE and USPS datasets as compared to other models evaluated with a good sample graph $\mathbb{G}^c$. Sometimes, as in the case of CMUPIE dataset, the clustering error with corruptions is lower than the uncorrupted dataset. This is due to the fact that random nature of the noise might make the clustering relatively easier for a certain noise case. However, it is not a general result. Finally, our model outperforms others in most of the cases for the interesting case of MNIST dataset where both graphs $\mathbb{G}^c$ and $\mathbb{G}^r$ are noisy for all models under consideration. It is worth mentioning here that even though the absolute errors are quite high for FRPCAG on the MNIST dataset, it performs relatively better than the other models. As PCA is mostly used as a feature extraction or a pre-processing step for a variety of machine learning algorithms, a better absolute classification performance can be obtained for these datasets by using FRPCAG as a pre-processing step for supervised algorithms as compared to other PCA models.

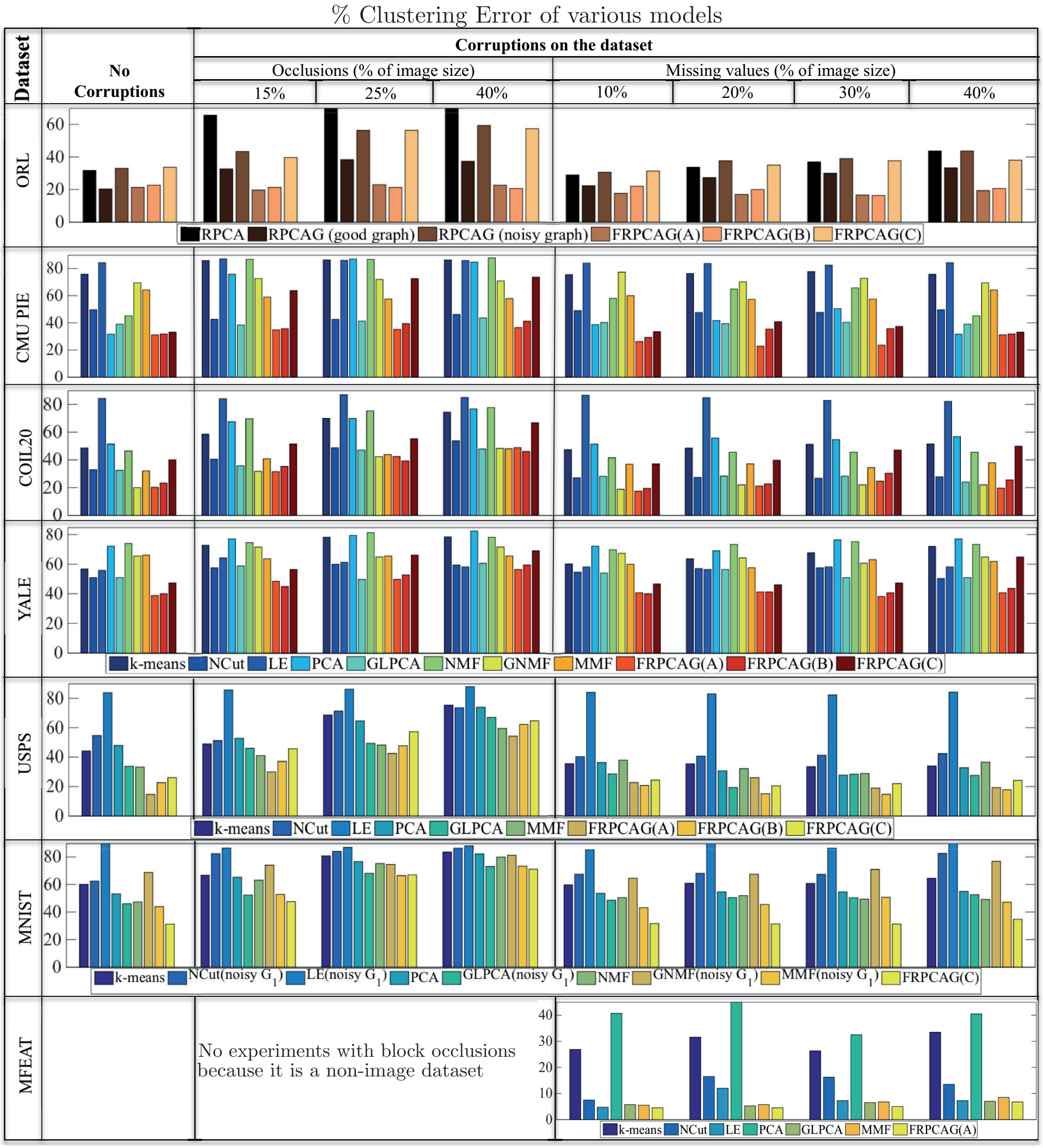


Figure 5.7 – A comparison of clustering error of our model with various dimensionality reduction models. The image data sets include: 1) ORL 2) CMU PIE 3) COIL20 and 4) YALE. The compared models are: k-means, NCuts, LE [2], PCA, GLPCA [3], NMF [4], GNMF [5], MMF [6], 9) RPCA [1], FRPCAG(A), FRPCAG(B) and FRPCAG(C). Two types of corruptions are introduced in the data: 1) Block occlusions and 2) Random missing values. NCut, LE, GLPCA, MMF and GNMF are evaluated with a good sample graph $\mathbb{G}^c$. FRPCAG(A) corresponds to our model evaluated with a good sample and a good feature graph, FRPCAG(B) to a good sample graph and a noisy feature graph and FRPCAG(C) to a noisy sample and feature graph. NMF and GNMF require non-negative data so they were not evaluated for the USPS and MFeat datasets because they are negative as well. MFeat is a non-image dataset so it is not evaluated with block occlusions. For MNIST both $\mathbb{G}^c$ and $\mathbb{G}^r$ are noisy.

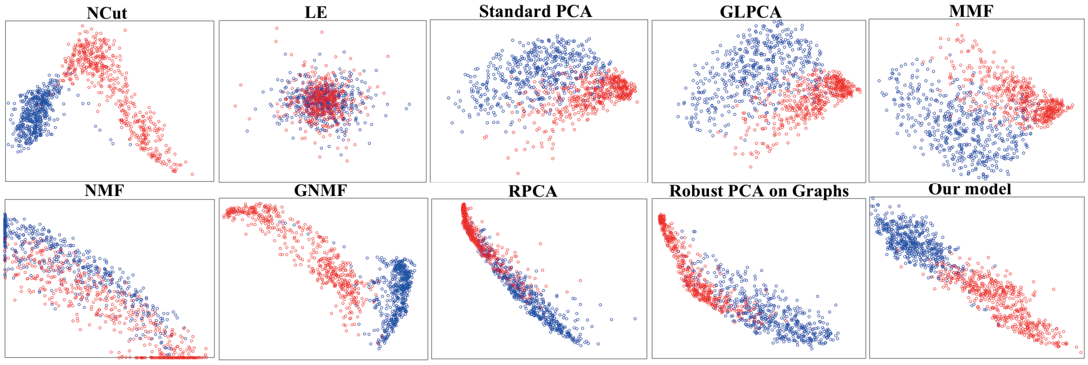


Figure 5.8 – Principal Components of 1000 samples of digits 0 and 1 of the MNIST dataset in 2D space. For this experiment all the digits were corrupted randomly with 15% missing pixels. Our proposed model (lower right) attains a good separation between the digits which is comparable and even better than other state-of-the-art dimensionality reduction models.

Comparison with Nuclear Norm based Models

Fig. 5.7 also presents a comparison of the clustering error of our model with nuclear norm based models, i.e, RPCA and RPCAG for ORL dataset. This comparison is of specific interest because of the convexity of all the algorithms under consideration. As these models require an expensive SVD step on the whole low-rank matrix at every iteration of the algorithm, these experiments are performed on small ORL dataset. Clearly, our proposed model FRPCAG(A) performs better than the nuclear norm based models even in the presence of large fraction of gross errors. Interestingly, even FRPCAG(B) with a noisy graph $\mathbb{G}^r$ performs better than RPCAG with a good graph $\mathbb{G}^c$. Furthermore, the performance of FRPCAG(C) with two noisy graphs is comparable to RPCAG with noisy graph, but still better than RPCA.

Principal Components

Fig. 5.8 shows the principal components of 1000 samples of MNIST dataset in two dimensional space obtained by various dimensionality reduction models. 500 samples of digit 0 and 1 each are chosen and randomly corrupted by 15% missing pixels for this experiment. Clearly, our proposed model attains a good separation between the digits 0 and 1 (represented by blue and red points respectively) comparable with other state-of-the-art dimensionality reduction models.

Effect of the number of nearest neighbors for graphs

In order to demonstrate the effect of number of nearest neighbors k_{nn} on the clustering performance of our model we perform a small experiment on the ORL dataset which has 400 images corresponding to 40 classes (10 images per class). We perform clustering for different values of $k_{nn} = 5, 10, 25, 40$. The clustering errors are $17.5\%, 17\%, 23\%$ and 31% respectively. Interestingly the minimum clustering error occurs for $k_{nn} = 5, 10$ which is less or equal to the number of images per class. Thus, when the number of nearest neighbors k_{nn} is approximately equal to or less than the number of images per class then the images of the same class are more well connected and those across the classes have weak connections. This results in a lower clustering error. A good way to set k_{nn} is to use some prior information about the average number of samples per class or the rank of the dataset. For our experiments we use $k_{nn} = 10$ for all the datasets and this value works quite well. The value of k_{nn} also depends on the number of data samples. For big datasets, sparser graphs (obtained with lower values of k_{nn}) tend to be more useful. For example, our experiments show that for the MNIST dataset (70,000 samples), $k_{nn} = 10$ is again a good value, even though the average number of samples per class is 7000.

5.8.2 Low-rank & Sparse Decomposition

In order to demonstrate the effectiveness of our model to recover low-rank static background from dynamic foreground, as an application of low-rank and sparse decomposition, we perform experiments on 1000 frames of 3 videos available online at https://sites.google.com/site/backgroundsubtraction/test-sequences. All the frames are vectorized and arranged in a matrix $\boldsymbol{Y}$ whose columns correspond to frames. The graph $\mathbb{G}^c$ is constructed between 1000 frames (columns of $\boldsymbol{Y}$) of the video and the graph $\mathbb{G}^r$ is constructed between the pixels of the frames (rows of $\boldsymbol{Y}$) using FLANN. Both graphs for all the videos are constructed without the prior knowledge of the mask of sparse errors (moving people). Fig. 5.9 shows the recovery of low-rank frames for one actual frame of each of the videos. The leftmost image in each row shows the actual frame, the other three show the recovered low-rank representations using RPCA, RPCAG and our proposed model (FRPCAG). The first row corresponds to a frame from the video of a restaurant food counter, the second row to the shopping mall lobby and the third row to an airport lobby. In each of the three plots it can be seen that our proposed model is able to separate the static backgrounds very accurately from the moving people which do not belong to the static ground truth. *Our model converged in less than 2 minutes for each of the three videos, whereas RPCA and RPCAG converged in more than 45 minutes.* Due to the unavailability of low-rank ground truth for these videos we do not present any quantitative results.

5.8.3 Computational Time

Table 5.3 presents the computational time and number of iterations for the convergence of FRPCAG, RPCAG and RPCA on different sizes and dimensions of the datasets. We also present the time needed for graph construction. The computation is done on a single core machine with a 3.3 GHz processor using Matlab without any distributed or parallel computing tricks. An ∞ in the table indicates that the algorithm did not converge in 4 hours. It is notable that our model requires a very small number of iterations to converge irrespective of the size of the dataset. Furthermore, the model is orders of magnitude faster than RPCA and RPCAG. This is clearly observed from the experiments on MNIST dataset where our proposed model is 100 times faster than RPCAG. Specially for MNIST dataset with 25000 samples, RPCAG and RPCA did not converge even in 4 hours, whereas, FRPCAG converged in less than a minute.

Table 5.3 – Computation times (in seconds) for graphs $\mathbb{G}^c$, $\mathbb{G}^r$, FRPCAG, RPCAG, RPCA and the number of iterations to converge for different datasets. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. ∞ indicates that the algorithm did not converge in 4 hours.

Dataset	**Samples**	**Features**	**Classes**	**Graphs**		**FRPCAG**		**RPCAG**		**RPCA**	
				$\mathbb{G}^c$	$\mathbb{G}^r$	**time**	**Iters**	**time**	**Iters**	**time**	**Iters**
MNIST	5000	784	10	10.8	4.3	13.7	27	1345	325	1090	378
MNIST	15000	784	10	32.5	13.3	35.4	23	3801	412	3400	323
MNIST	25000	784	10	40.7	22.2	58.6	24	∞	∞	∞	∞
ORL	300	10304	30	1.8	56.4	4.7	12	360	301	240	320
USPS	3500	256	10	5.8	10.8	1.76	16	900	410	790	350
US census	2.5 million	68	-	540	42.3	3900	200	∞	∞	∞	∞

To demonstrate the scalability of our model for big datasets, we perform an experiment on the US

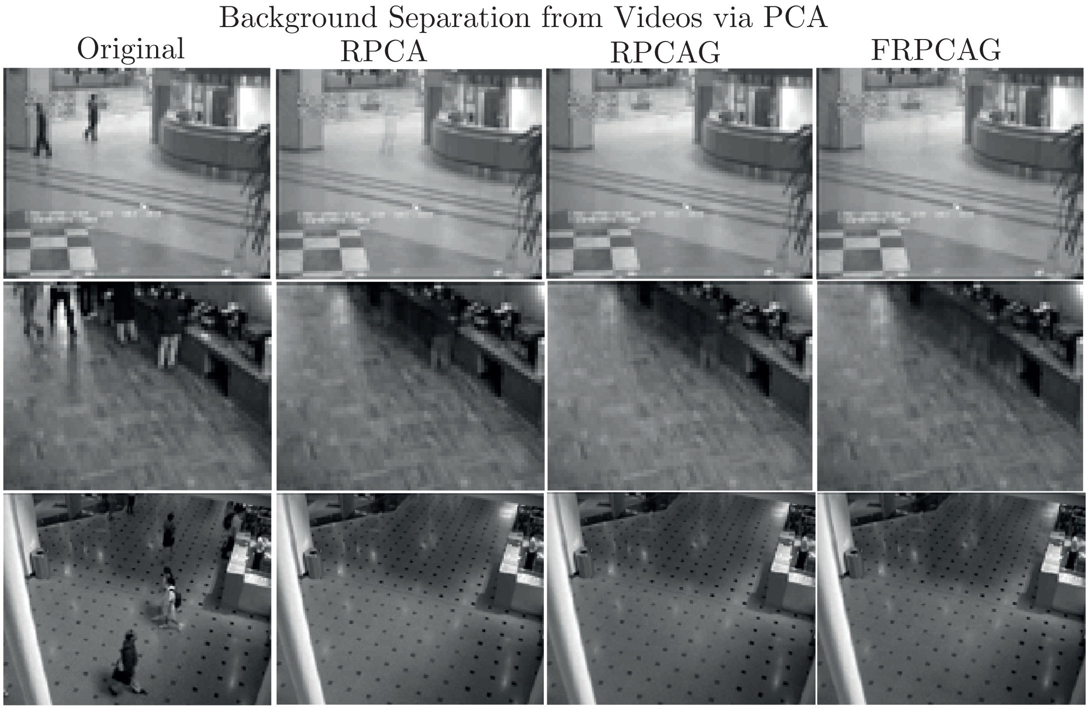


Figure 5.9 – Static background separation from three videos. Each row shows the actual frame (left), recovered static low-rank background using RPCA, RPCAG and our proposed model. The first row corresponds to the video of a restaurant food counter, the second row to the shopping mall lobby and the third to an airport lobby. In all the three videos the moving people belong to the sparse component. Thus, our model is able to accurately separate the static portion from the three frames as good as the RPCAG. Our model converged in less than 2 minutes for each of the three videos, whereas RPCA and RPCAG converged in more than 45 minutes.

census 1990 https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990) dataset available at the UCI machine learning repository. This dataset consists of approximately 2.5 million samples and 68 features. The approximate k_{nn}-nearest neighbors graph construction strategy using the FLANN algorithm took only 540 secs to construct $\mathbb{G}^c$ between 2.5 million samples and 42.3 secs to construct $\mathbb{G}^r$ between 68 features. We do not compare the performance of this model with other state-of-the-art models as the ground truth for this dataset is not available. However, we run our algorithm in order to see how long it takes to recover a low-rank representation for this dataset. It took 65 minutes and 200 iterations for the algorithm to converge on a single core machine with 3.3 GHz of CPU power.

5.9 Generalized Fast Robust PCA on Graphs (GFRPCAG)

In principle, model 5.1, has some connections with the state-of-the-art nuclear norm based low-rank recovery method RPCA [1]. RPCA determines a low-rank representation by: 1) Soft-thresholding the singular values by a model parameter and 2) Determining the clean left and right singular vectors iteratively from the data by performing an SVD in every iteration.

We discussed what makes our model an approximate recovery as compared to exact recovery and studied what the approximation error is composed of. Now we use these arguments to motivate a more general model. We say that our model (5.1) has connections with RPCA because it performs the same two steps but in a different way, as explained in the previous section. In our model: 1) The singular values are penalized by the graph eigenvalues. Therefore, one does not have a direct control on this operation as the thresholding depends on the graph spectrum. 2) The left and right singular vectors are constrained to be aligned with the low frequency eigenvectors of the row and column graph. Thus, we take away the freedom of the singular vectors. In fact this is the primary reason our method does not need an SVD and scales very well for big datasets.

5.9.1 Re-visiting the Approximation Error of FRPCAG

There are two main reasons of the approximation error, a glimpse of which has already been given above. We now discuss these two reasons in detail:

Uncontrolled attenuation of the higher singular values: The Manifold Shrinking Effect

Ideally one would like the singular values corresponding to high frequency of the data matrix (smaller in magnitude) to be attenuated more because they correspond to noise and errors in the data. Furthermore, the singular values corresponding to the lower frequencies, which constitute most of the data variance should be attenuated less. Such an operation would require the graph spectrum to be a step function where the position of the step depends on the rank of the matrix. From our theoretical analysis, one can see that the error of the optimal low-rank $\boldsymbol{X}^*$ depends on the two ratios: $\frac{\lambda_k^c}{\lambda_{k+1}^c}$ and $\frac{\lambda_k^r}{\lambda_{k+1}^r}$. These ratios are known as the eigen-gaps of the graphs $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$ respectively. In practice, the graphs constructed via the k_{nn}-nearest neighbors strategy result in a smoothly increasing spectrum. Therefore, these ratios are never small enough. As a consequence, the undesired singular values (smaller in magnitude) are not attenuated enough and they are not exactly zero. If one forces higher attenuation by increasing the constants γ_c and γ_r then the desired singular values are penalized more than required. This results in a shrinking effect of the manifold as shown in Fig. 5.10. In this Fig. the first row shows the recovery of manifold with increasing values of the graph regularization parameter in FRPCAG (with

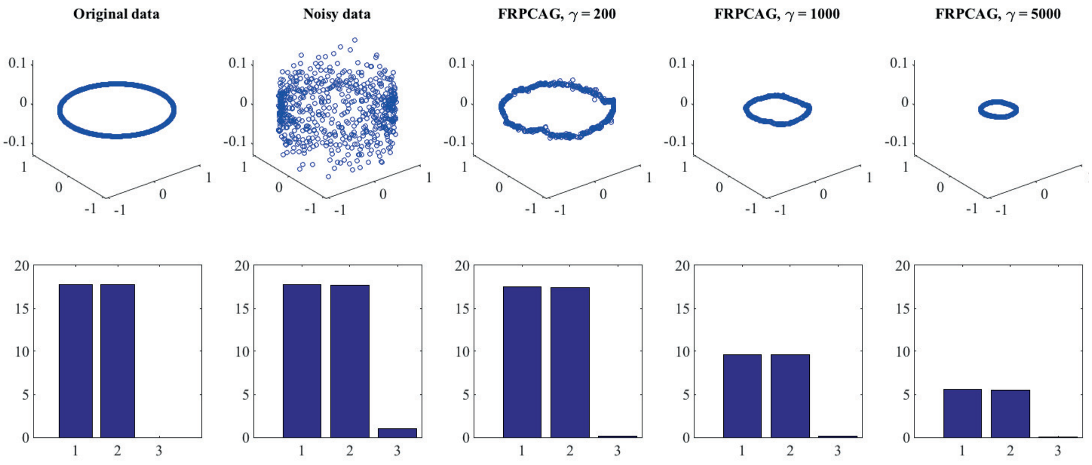


Figure 5.10 – The manifold shrinking effect of a 2D manifold corrupted with noise. The first row shows the recovery of manifold with increasing values of FRPCAG (with one graph only) and the second row shows the singular values. Clearly, the 3^{rd} singular value does not become zero with higher penalization which degrades the lower singular values and the manifold begins to shrink.

one graph only) and the second row shows the singular values. Clearly, the 3^{rd} singular value does not become zero with higher penalization which degrades the desired singular values and the manifold begins to shrink. This calls for a procedure to manually threshold the singular values beyond a certain penalization.

The subspace alignment gap

We do not allow the left and right singular vectors of the low-rank matrix to be determined freely via SVD. This results in a slight misalignment of the clean singular vectors of the resultant low-rank matrix as compared to the singular vectors of the original clean data. Note that the alignment occurs as a result of a force acting on the singular vectors which depends on the regularization parameters. While increasing this force results in more alignment, the downside is the shrinking effect of singular values. Thus, a good parameter setting might still leave some alignment gap in order not to threshold the singular values more than required.

These two effects with the increasing values of the graph regularization parameter for FRPCAG are illustrated in Fig. 5.11. Here we show this effect for only one subspace (left singular vectors). Similar phenomenon takes place for the right singular vectors as well. For very small values of the regularization parameter, the singular vectors learned via FRPCAG are still aligned with the singular vectors of the noisy data and penalization on the higher singular values is not enough to remove noise. With increasing regularization, the singular vectors start aligning and the higher singular values are thresholded more. Very high values of the regularization parameter might still provide a good alignment of the singular vectors but the singular values might be thresholded more than required, resulting in a shrinking effect. Thus, beyond a certain point, the increase of parameters contributes to higher errors due to the excessive shrinking of the singular values.

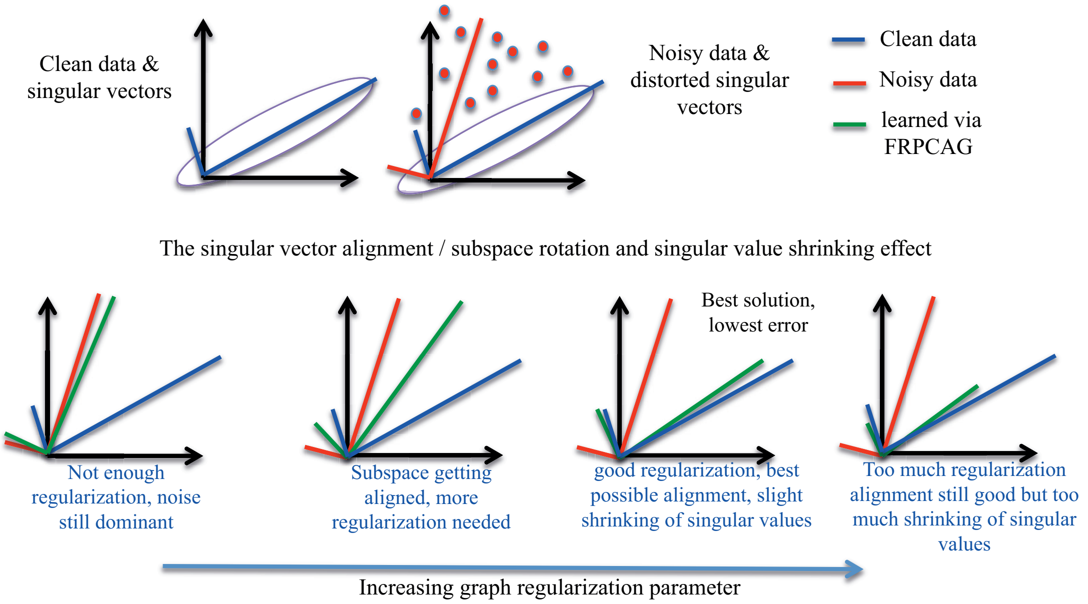

Clean data &
singular vectors
Noisy data &
distorted singular
vectors
Clean data
Noisy data
learned via
FRPCAG
The singular vector alignment / subspace rotation and singular value shrinking effect
Best solution,
lowest error
Not enough
regularization, noise
still dominant
Subspace getting
aligned, more
regularization needed
good regularization, best
possible alignment, slight
shrinking of singular values
Too much regularization
alignment still good but too
much shrinking of singular
values
Increasing graph regularization parameter

Figure 5.11 – The alignment of the singular vectors with the singular vectors of clean data and the corresponding shrinking of the singular values with the increasing graph regularization parameter. The first row shows the original and noisy data, where the blue vectors show the first two singular vectors of the clean data and the red ones for the noisy data. The second row shows how the singular vectors (green) learned with the FRPCAG eq. 5.1 are being aligned with the singular vectors of the clean data. All the singular vectors are scaled by their singular values. Too much regularization shrinks the singular vectors.

5.9.2 Design of deterministic graph filters ‘g’ to improve approximation error

The thresholding of the singular values is governed by the graph spectra as also pointed out in theorem 5.6.1. While the subspace rotation effect is an implicit attribute of our graph based low-rank recovery method and it cannot be improved trivially, we can counter the shrinking effect to improve the approximation. A straight-forward method to improve the approximation would be to have a better control on the singular values by manipulating the graph spectra. To do so, it would be wise to define functions of the graph spectrum which make the spectral gap as large as possible. In simpler words we need a family of functions $g(\boldsymbol{a})$ which are characterized by some parameter $\boldsymbol{a}$ that force the graph filtering operation to follow a step behavior. More specifically, we would like to solve the following generalized problem instead of eq. 5.1.

$$\min_{\boldsymbol{X}} \phi(\boldsymbol{Y}-\boldsymbol{X}) + \gamma_c \operatorname{tr}(\boldsymbol{X} g_c(\boldsymbol{L}^c)\boldsymbol{X}^\top) + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top g_r(\boldsymbol{L}^r)\boldsymbol{X}) \tag{5.7}$$

The term $\operatorname{tr}(\boldsymbol{X}^\top g(\boldsymbol{L})\boldsymbol{X}) = \|g(\boldsymbol{\Lambda})^{\frac{1}{2}}\hat{\boldsymbol{X}}\|$ allows for a better control of the graph spectral penalization.

For our specific problem, we would like to have the lowest frequencies untouched as well as the highest frequencies strongly penalized. In order to do so, we propose the following design.

Let us first define a function

$$h_b(\boldsymbol{x}) = \begin{cases} e^{-\frac{b}{x-\frac{b}{2}}} & \text{if } \boldsymbol{x} \geq \frac{b}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Please note that this function is differentiable at all points. We then define g_b as

$$g_b(\boldsymbol{x}) = \frac{h_b(\boldsymbol{x})}{h_b(2b-\boldsymbol{x})}$$

This new function has a few important properties. It is equal to 0 for all values smaller than $\frac{b}{2}$, and then grows. It is infinite for all values above $\frac{3}{2}b$. As a result, choosing the function $\sqrt{g_b}$ in our regularization term has the following effect: It preserves the frequency content for $\lambda_\ell << b$ and cuts the high frequencies $\lambda_\ell >> b$. The parameter b acts as a frequency limit for our function. See Fig. 5.12.

Inner mechanism of the proposed regularization

Since we solve our problem with proximal splitting algorithm, we also need to compute the proximal operator of $\operatorname{tr}(\boldsymbol{X}^\top g(\boldsymbol{L})\boldsymbol{X}) = \|g_b(\boldsymbol{\Lambda})^{\frac{1}{2}}\hat{\boldsymbol{X}}\|$. The computation is straightforward and explains the inner mechanism of such a regularization term. We first compute the gradient of

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \gamma\|g_b(\Lambda)^{\frac{1}{2}}\boldsymbol{P}^\top\boldsymbol{x}\|_2^2$$

and set it to 0.

$$2(\boldsymbol{x}-\boldsymbol{y}) + 2\gamma\boldsymbol{P}g_b(\Lambda)\boldsymbol{P}^\top\boldsymbol{x} = 0$$

We then rewrite the following expression in the spectral domain:

$$\hat{x}(\ell) - \hat{\boldsymbol{y}}(\ell) + \gamma g_b(\lambda_\ell)\hat{\boldsymbol{x}}(\ell) = 0, \quad \forall \ell \in \{0,\dots,n-1\}$$

Shuffling the equation leads to the solution.

$$\hat{\boldsymbol{x}}(\ell) = \frac{1}{1+\gamma g_b(\lambda_\ell)}\hat{\boldsymbol{y}}(\ell), \quad \forall \ell \in \{0,\dots,n-1\} \tag{5.8}$$

Eq. (5.8) shows that the proximal operator of $\operatorname{tr}(\boldsymbol{X}^\top g(\boldsymbol{L})\boldsymbol{X})$ is a graph spectral filtering with

$$\begin{aligned} f_b(\boldsymbol{x},\gamma) &= \frac{1}{1+\gamma g(\boldsymbol{x},b)} \\ &= \frac{h(2b-\boldsymbol{x})}{h(2b-\boldsymbol{x})+\gamma h(\boldsymbol{x})} \end{aligned}$$

In fact the filter f_b is infinitely many times differentiable and is smooth enough to be be easily approximated by a low order polynomial. We observe that $f_b(\cdot,1)$ is a low pass filter with bandwidth b. The computation cost of a filtering operation grows with the number edges e and the order of the polynomial o: $\mathcal{O}(oe)$ [7]. In figure 5.12, we show examples of the different aforementioned functions.

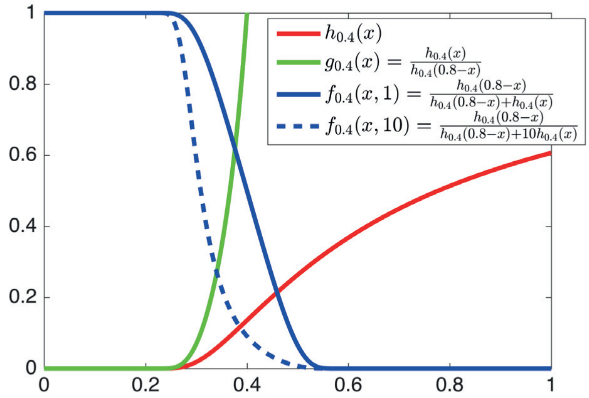


Figure 5.12 – Example of functions. Here $b = 0.4$.

5.10 Conclusions

We present Fast Robust PCA on Graphs (FRPCAG), a fast dimensionality reduction algorithm for mining low-rank structure from high dimensional and large datasets. The idea lies on the novel concept of multilinear low-rank matrices on graphs - matrices whose rows and columns belong to the span to the first few eigenvectors of the graphs constructed between them. FRPCAG implicitly brings into use the GMLSVD framework to determine the left and right singular vectors of low-rank matrix from fixed graph eigenvectors via computationally feasible operations. The power of the model lies in its ability to effectively exploit the hidden information about the intrinsic dimensionality of the smooth low-dimensional manifolds on which reside the clusterable signals and features of the data. Therefore, it targets an approximate recovery of low-rank signals by exploiting the local smoothness assumption of the samples and features of the data via graph structures only. In short, our method leverages 1) smoothness of the samples on a sample graph and 2) smoothness of the features on a feature graph. This dual or bi-graph filtering approach has been proved to implicitly follow the MLRTG and GMLSVD frameworks. The proposed method is convex, scalable and efficient and tends to outperform several other state-of-the-art exact low-rank recovery methods in clustering tasks that use the expensive nuclear norm. In an ordinary clustering task FRPCAG is approximately 100 times faster than nuclear norm based methods. The double graph structure also plays an important role towards the robustness of the model to gross corruptions.

6 Compressive PCA on Graphs

Note: *Most of the parts of this chapter have been published in the following article:*

"N. Shahid, N. Perraudin, G. Puy, and P. Vandergheynst, "Compressive PCA for Low-rank Matrices on Graphs," IEEE transactions on Signal and Information Processing over Networks, 2016. "

In the previous chapter we introduced Fast Robust PCA, which approximates a scalable recovery method for non-linear low-rank datasets $\boldsymbol{X}^* \in \mathbb{M}\mathbb{L}\mathbb{T}(\boldsymbol{P}_k^\mu)$. FRPCAG does not require an SVD and scales linearly with n. It relies on fast dual graph filtering operations which involve matrix vector multiplications and can be parallelized on a GPU in every iteration. However, the size of the problem is still an issue for big datasets because the problem cannot be broken down into small sub-problems and the solution merged at the end. Thus, for the non-GPU implementation, it still suffers from 1) memory requirements 2) cost of k-means for clustering 3) the cost of parameter tuning for large p and large n and 4) scalability for very big datasets. This said, sometimes one might not even have access to the full dataset $\boldsymbol{Y}$. This is typical, for instance for biomedical applications, such as MRI and tomography. In such applications the number of observations is limited by the data acquisition protocols. In MRI, the number of observations is proportional to the time and dose required for the procedure. In tomography one might have access to the projections only. Thus, FRPCAG is not usable if 1) the dataset is large and 2) only a subset of the dataset or measurements are available. Despite the above limitations of data acquisition, one might have access to some additional information about the unobserved samples.

In this chapter we answer the following questions:

- What would be an efficient and highly scalable recovery framework, involving compression, for datasets which are jointly low-rank on two manifolds?
- Alternatively, given a few randomly sampled observations and features from a data matrix $\boldsymbol{Y} \in \Re^{p \times n}$, is it possible to efficiently recover the complete non-linear low-rank representation?

We mostly limit ourselves to the case 1 above, where a graphical prior is available or can be conveniently constructed for the complete set of observations for the application under consideration. A brief initial treatment of the 2nd case constitutes the future applications in Chapter 8. We call our proposed framework, "Compressive PCA on Graphs (CPCA)".

We pointed out in Chapter 1, that our proposed approximation problems fall into two categories, one which makes implicit use of the GMLSVD framework and the other which makes explicit use of it. We specifically highlight that CPCA is a method which is effective when the number of samples n and the features p are very large and one can afford to lose the exactness of the representation for gain in speed. In this context, CPCA implicitly brings into use the GMLSVD framework, i.e., it relies on the graph Dirichlet priors rather than the explicit use of the graph eigenvectors, as computing the graph eigenvectors and tuning the size of subspace can be cumbersome.

6.1 Contributions

Below we describe our contributions in detail.

1. Sampling & RIP for MLRTG: To solve the per iteration scalability problem of FRPCAG we propose to perform a dual uniform sampling of the data matrices, along rows and columns. We present a restricted isometry property (RIP) for low-rank matrices on graphs and relate it to the cumulative coherence of the graph eigenvectors. FRPCAG is then used to recover the low-rank representation for the sampled data.

2. Decoders for low-rank recovery: We present two (ideal and alternate) convex and efficient decoders for recovering the full low-rank matrix from the corresponding low-rank matrix of the sampled data. However, our main contribution comprises the set of 3 additional parallel, low-cost and parameter-free *approximate* decoders, which significantly boost the speed of our framework by introducing a few approximations. Our rigorous theoretical analysis also proves that the recovery error of the above decoders depends on the eigen-gaps of the row and column graph Laplacians.

3. Low-Rank Clustering: For the clustering application of PCA, we propose a low-cost and parallel scheme based on CPCA. The key idea is to decode the labels of the complete dataset from the labels of a sampled low-rank dataset, without computing the complete low-rank matrix.

Low-rank recovery experiments on 3 real video datasets and clustering experiments on 5 benchmark datasets reveal that the performance of our model is comparable to 10 different state-of-the-art PCA and non-PCA based methods. We also study some cases where CPCA fails to perform as well as the state-of-the-art.

Our proposed framework is inspired by the recently introduced sampling of band-limited signals on graphs [100]. While we borrow several concepts from here, our framework is significantly different from [100] in many contexts. We target the *low-rank* recovery of matrices, whereas [100] targets the recovery of band-limited signals / vectors. For our framework it is important for the data matrix to be low-rank jointly on the row and column graphs. Thus, our sampling scheme and RIP are generalized for two graphs. The design of a sampling scheme is the major focus of [100], while we just focus on the case of *uniform sampling* and instead focus on *how much to sample jointly* given the two graphs. Of course, our method can be extended directly for other sampling schemes in [100]. A major difference lies in the application domain and hence the experiments. Unlike [100], we target two applications related to PCA: 1) low-rank recovery and 2) clustering. Thus, contrary to [100] our proposed decoders are designed for these applications. A major contribution of our work in contrast to [100] is the design of approximate decoders for low-rank recovery and clustering which significantly boost the speed of our framework for big datasets.

6.2 A glimpse of Compressive PCA on Graphs

Given a data matrix $\boldsymbol{Y} \in \mathbb{R}^{p\times n} = \boldsymbol{X}^* + \boldsymbol{\eta}$, where $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^{\mu})$ and $\boldsymbol{\eta}$ models the errors and noise, the goal is to develop a method to efficiently recover $\boldsymbol{X}^*$. We propose to

1. Construct graphs $\mathbb{G}^r$ and $\mathbb{G}^c$ with Laplacians $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$ between the rows and columns of $\boldsymbol{Y}$.
2. Sample the rows and columns of $\boldsymbol{Y}$ to get a subsampled matrix $\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{X}}^* + \boldsymbol{E}$ using the sampling scheme of Section 6.3.
3. Construct the compressed Laplacians $\tilde{\boldsymbol{L}}^c, \tilde{\boldsymbol{L}}^r$ from $\boldsymbol{L}^c, \boldsymbol{L}^r$ (Section 6.4.1).
4. Determine a low-rank matrix $\tilde{\boldsymbol{X}}$ for $\tilde{\boldsymbol{Y}}$ with $\tilde{\boldsymbol{L}}^c, \tilde{\boldsymbol{L}}^r$ in algorithm 1 of Chapter 5.

   $$\min_{\tilde{\boldsymbol{X}}} \phi(\tilde{\boldsymbol{Y}} - \tilde{\boldsymbol{X}}) + \gamma_c \operatorname{tr}(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{L}}^c\tilde{\boldsymbol{X}}^\top) + \gamma_r \operatorname{tr}(\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{L}}^r\tilde{\boldsymbol{X}}),$$

   where ϕ is a loss function (possibly l_p norm) and the sampled low-rank matrix follows the signal model below:

   $$\begin{aligned}\tilde{\boldsymbol{X}} &= \tilde{\boldsymbol{X}}^* + \tilde{\boldsymbol{E}}\\ &= \boldsymbol{M}^r\boldsymbol{X}^*\boldsymbol{M}^c + \tilde{\boldsymbol{E}},\end{aligned}$$

   where $\tilde{\boldsymbol{E}}$ models the errors in the recovery of the subsampled low-rank matrix $\tilde{\boldsymbol{X}}$ and $\boldsymbol{M}^c, \boldsymbol{M}^r$ are the column and row sampling matrices whose design is discussed in Section 6.3.
5. Use the decoders presented in Section 6.5 to decode the low-rank matrix $\bar{\boldsymbol{X}}^* = \underline{\boldsymbol{X}}^* + \boldsymbol{E}^*$ (where $\boldsymbol{E}^*$ denotes the error on the recovery of optimal $\underline{\boldsymbol{X}}^*$) on graphs $\boldsymbol{L}^c, \boldsymbol{L}^r$ if the task is low-rank recovery, or perform k-means on $\tilde{\boldsymbol{X}}$ to get cluster labels $\tilde{\boldsymbol{C}}$ and use the clustering algorithm (presented in Section 6.6) to get the cluster labels $\boldsymbol{C}$ for the full matrix $\boldsymbol{X}$.

Throughout this work we use the approximate nearest neighbor algorithm (FLANN [60]) for graph construction whose complexity is $\mathcal{O}(np\log(n))$ for $p < n$ [61] (and it can be performed in parallel). Unlike, the previous chapter, where we assumed for simplicity that the matrix $\boldsymbol{X}^*$ is MLRTG with the same number k of eigenvectors across the rows and columns, here we will assume MLRTG w.r.t (k_r, k_c) eigenvectors across the rows and columns.

6.3 RIP for MLRTG

Let $\boldsymbol{M}^r \in \Re^{\rho_r\times p}$ be the subsampling matrix for sampling the rows and $\boldsymbol{M}^c \in \Re^{n\times\rho_c}$ for sampling the columns of $\boldsymbol{Y}$. $\boldsymbol{M}^c$ and $\boldsymbol{M}^r$ are constructed by drawing ρ_c and ρ_r indices $\Omega_c = \{\omega_1\cdots\omega_{\rho_c}\}$ and $\Omega_r = \{\omega_1\cdots\omega_{\rho_r}\}$ **uniformly without replacement** from the sets $\{1,2,\cdots,n\}$ and $\{1,2,\cdots,p\}$ and satisfy:

$$\boldsymbol{M}^c_{ij} = \begin{cases} 1 & \text{if } i = \omega_j \\ 0 & \text{otherwise,}\end{cases} \qquad \boldsymbol{M}^r_{ij} = \begin{cases} 1 & \text{if } j = \omega_i \\ 0 & \text{otherwise.}\end{cases} \tag{6.1}$$

Now, the subsampled data matrix $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{\rho_r\times\rho_c}$ can be written as $\widetilde{\boldsymbol{Y}} = \boldsymbol{M}^r\boldsymbol{Y}\boldsymbol{M}^c$. CPCA requires $\boldsymbol{M}^r$ and $\boldsymbol{M}^c$ to be constructed such that the "low-rankness" property of the data $\boldsymbol{Y}$ is preserved under sampling. Before discussing this, we introduce a few basic definitions in the context of graphs $\mathbb{G}^c$ and $\mathbb{G}^r$.

Definition 6.3.1. *(**Graph cumulative coherence**). The cumulative coherence of order k_c, k_r of $\mathbb{G}^c$ and $\mathbb{G}^r$ is:*

$$\nu_{k_c} = \max_{1 \le i \le n} \sqrt{n} \| \boldsymbol{P}_{k_c}^{c\ \top} \boldsymbol{\Delta}_i^c \|_2 \ \ \& \ \nu_{k_r} = \max_{1 \le j \le p} \sqrt{p} \| \boldsymbol{P}_{k_r}^{r\ \top} \boldsymbol{\Delta}_j^r \|_2,$$

where $\boldsymbol{\Delta}^c \in \{0,1\}^n, \boldsymbol{\Delta}^r \in \{0,1\}^p$ are binary vectors and $\boldsymbol{\Delta}_i^c = 1$ if the i^{th} entry of $\boldsymbol{\Delta}^c$ is 1 and 0 otherwise. Thus, $\boldsymbol{\Delta}_i^c$ corresponds to a specific node of the graph.

In the above equations $\boldsymbol{P}_{k_c}^{c\ \top} \boldsymbol{\Delta}_i^c$ and $\boldsymbol{P}_{k_r}^{r\ \top} \boldsymbol{\Delta}_j^r$ characterize the first k_c and k_r fourier modes [20] of the nodes i and j on the graphs $\mathbb{G}^c$ and $\mathbb{G}^r$ respectively. Thus, the cumulative coherence is a measure of how well the energy of the (k_r, k_c) low-rank matrices spreads over the nodes of the graphs. These quantities exactly control the number of vertices ρ_c and ρ_r that need to be sampled from the graphs $\mathbb{G}^r$ and $\mathbb{G}^c$ such that the properties of the graphs are preserved [100].

Consider the example where a particular node i has a high coherence. Then, it implies that their exist some low-rank signals whose energy is highly concentrated on the node i. Removing this node would result in a loss of information in the data. If the coherence of this node is low then removing it in the sampling process would result in no loss of information. We already mentioned that we are interested in the case of uniform sampling. Therefore, in order to be able to sample a small number of nodes uniformly from the graphs, the cumulative coherence should be as low as possible.

We remind that for our application we desire to sample the data matrix $\boldsymbol{Y}$ such that its low-rank structure is preserved under this sampling. How can we ensure this via graph cumulative coherence? This follows directly from the fact that we are concerned about the data matrices which are also low-rank with respect to the two graphs under consideration $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$. In simple words, the columns of the clean data matrix $\boldsymbol{X}^*$ belong to the span of the eigenvectors $\boldsymbol{P}_{k_r}^r$ and the rows to the span of $\boldsymbol{P}_{k_c}^c$. Thus, the coherence conditions for the graph directly imply the coherence condition on the data matrix $\boldsymbol{X}^*$ itself. Therefore, using these quantities to sample the data matrix $\boldsymbol{X}^*$ will ensure the preservation of two properties under sampling: 1) the structure of the corresponding graphs and 2) the low-rankness of the data matrix $\boldsymbol{X}^*$. Given the above definitions, we are now ready to present the restricted-isometry theorem for MLRTG.

Theorem 6.3.1. *(**Restricted-isometry property (RIP) for MLRTG**) Let $\boldsymbol{M}^c$ and $\boldsymbol{M}^r$ be two random subsampling matrices as constructed in* (6.1). *For any $\delta, \epsilon \in (0,1)$, with probability at least $1-\epsilon$,*

$$(1-\delta)\|\boldsymbol{X}^*\|_F^2 \le \frac{np}{\rho_r \rho_c} \|\boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c\|_F^2 \le (1+\delta)\|\boldsymbol{X}^*\|_F^2 \tag{6.2}$$

for all $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ provided that

$$\rho_c \ge \frac{27}{\delta^2} \nu_{k_c}^2 \log\left(\frac{4k_c}{\epsilon}\right) \ \& \ \rho_r \ge \frac{27}{\delta^2} \nu_{k_r}^2 \log\left(\frac{4k_r}{\epsilon}\right), \tag{6.3}$$

where ν_{k_c}, ν_{k_r} characterize the graph cumulative coherence as in Definition 6.3.1 and $\frac{np}{\rho_c \rho_r}$ is just a normalization constant which quantifies the norm conservation in (6.2).

Proof. Please refer to Appendix A.4. □

Theorem 6.3.1 is a direct extension of the RIP for k-bandlimited signals on a graph [100]. It states that the information in $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ is preserved with overwhelming probability if the sampling matrices (6.1) are constructed with a uniform sampling strategy satisfying (6.3). Note that ρ_r and ρ_c

depend on the cumulative coherence of the graph eigenvectors. The better spread the eigenvectors are, the smaller is the number of vertices that need to be sampled.

It is proved in [100] that $\nu_{k_c} \geq \sqrt{k_c}$ and $\nu_{k_r} \geq \sqrt{k_r}$. Hence, when the lower bounds are attained, one only needs to sample an order of $O(k_c \log(k_c))$ columns and $O(k_r \log(k_r))$ rows to ensure that the RIP (eq. (6.3)) holds. This is the ideal scenario. However, one can also have $\nu_{k_c} = \sqrt{n}$ or $\nu_{k_r} = \sqrt{p}$ in some situations. Let us give some examples. The lower bound on ν_k is attained, e.g, when the graph is a regular lattice. In this case the graph Fourier transform is the "usual" Fourier transform and $\nu_k = \sqrt{k}$ for all k. Another example where the lower bound is attained is when the graph contains k disconnected components of identical size. In this case, one can prove that $\nu_k = \sqrt{k}$ [100]. Intuitively, the coherence remains close to this lower bound when these k components are weakly interconnected [100].

The upper bound on ν_k is attained when, for example, the graph has one of its nodes not connected to any other node. In this case, one must sample this node. Indeed, there is no way to guess the value of the signal on this node from any neighbour. As the sampling is random, one is sure to sample this node only when all the nodes are sampled. Uniform sampling is not the best strategy in this setting. Furthermore, note that such a case is only possible if the graph is noisy or the data has strong outliers. One should resort to a more distribution aware sampling in such a case, as presented in [100].

We choose in this work to present the results using a uniform distribution for simplicity. Note however that one can adapt the sampling distribution to the underlying structure of the graph to ensure optimal sampling results. A consequence of the result in [100] is that there always exist distributions that ensure that the RIP holds when sampling $O(k_r \log(k_r))$ rows and $O(k_c \log(k_c))$ columns only. The optimal sampling distribution for which this result holds is defined in [100] (see Section 2.2). Furthermore, a fast algorithm to compute this distribution also exists (Section 4 of [100]).

However, note that the above discussion is only valid for $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$. In reality, one never has access to $\boldsymbol{X}^*$, but its noisy version $\boldsymbol{Y} = \boldsymbol{X}^* + \boldsymbol{\eta}$. Therefore, the sampling is done on the matrix $\boldsymbol{Y}$ instead of $\boldsymbol{X}^*$. Furthermore, the coherence bounds are also derived from the matrix $\boldsymbol{Y}$. This results in an error in the sampling process, such that the sampled matrix $\tilde{\boldsymbol{Y}}$ can be represented as:

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{X}}^* + \boldsymbol{E} = \boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c + \boldsymbol{E},$$

where $\boldsymbol{E}$ models noise and errors.

6.4 Compressed Low-Rank Matrix

Once the compressed dataset $\tilde{\boldsymbol{Y}} \in \mathbb{R}^{\rho_r \times \rho_c}$ is obtained, the low-rank representation has to be extracted which takes into account the graph structures. Thus we propose the following two step strategy:

1. Construct graphs for compressed data.

2. Run Fast Robust PCA on Graphs (FRPCAG) on the compressed data.

These two steps are elaborated in the following subsections.

6.4.1 Graphs for Compressed data

To ensure the preservation of algebraic and spectral properties one can construct the compressed Laplacians $\tilde{\boldsymbol{L}}^r \in \mathbb{R}^{\rho_r \times \rho_r}$ and $\tilde{\boldsymbol{L}}^c \in \mathbb{R}^{\rho_c \times \rho_c}$ from the Kron reduction of $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$ [109]. Let Ω be the set of sampled nodes and $\bar{\Omega}$ the complement set and let $\boldsymbol{L}(A_r, A_c)$ denote the (row, column) sampling of $\boldsymbol{L}$ w.r.t sets A_r, A_c then the Laplacian $\tilde{\boldsymbol{L}}^c$ for the columns of compressed matrix $\tilde{\boldsymbol{Y}}$ is:

$$\tilde{\boldsymbol{L}}^c = \boldsymbol{L}^c(\Omega,\Omega) - \boldsymbol{L}^c(\Omega,\bar{\Omega})\boldsymbol{L}^{c-1}(\bar{\Omega},\bar{\Omega})\boldsymbol{L}^c(\bar{\Omega},\Omega).$$

Let $\boldsymbol{L}^c$ has k_c connected components or $\lambda^c_{k_c}/\lambda^c_{k_c+1} \approx 0$. Then, as argued in theorem $III.4$ of [109], two nodes α, β are not connected in $\tilde{\boldsymbol{L}}^c$ if there is no path between them in $\boldsymbol{L}^c$ via $\bar{\Omega}$. Assume that each of the connected components has the same number of nodes. Then, if the sampling is done uniformly within each of the connected components according to the sampling bounds described in eq.(6.3), one can expect $\tilde{\boldsymbol{L}}^c$ to have k_c connected components as well. This is an inherent property of the Kron reduction method. However, for the case of large variation of the number of nodes among the connected components one might want to resort to a more distribution aware sampling scheme. Such schemes have been discussed in [100] and have not been addressed in this work. Nevertheless, the Kron reduction strategy mentioned here is independent of the sampling strategy used. The same concepts hold for $\tilde{\boldsymbol{L}}^r$ as well.

Complexity: The Kron reduction method involves the multiplication of 3 sparse matrices. The only expensive operation above is the inverse of $\boldsymbol{L}(\bar{\Omega},\bar{\Omega})$ which can be performed with $\mathcal{O}(O_l k_{nn} n)$ cost using the Lancoz method [7], where O_l is the number of iterations for Lancoz approximation.

6.4.2 FRPCAG on the Compressed Data

Once the Laplacians $\tilde{\boldsymbol{L}}^r \in \mathbb{R}^{\rho_r \times \rho_r}, \tilde{\boldsymbol{L}}^c \in \mathbb{R}^{\rho_c \times \rho_c}$ are obtained, the next step is to recover the low-rank matrix $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r \times \rho_c}$.

Let $\tilde{\boldsymbol{L}}^c = \tilde{\boldsymbol{P}}^c \tilde{\boldsymbol{\Lambda}}^c \tilde{\boldsymbol{P}}^{c\top} = \tilde{\boldsymbol{P}}^c_{k_c} \tilde{\boldsymbol{\Lambda}}^c_{k_c} \tilde{\boldsymbol{P}}^{c\ \top}_{k_c} + \bar{\tilde{\boldsymbol{P}}}^c_{k_c} \bar{\tilde{\boldsymbol{\Lambda}}}^c_{k_c} \bar{\tilde{\boldsymbol{P}}}^{c\ \top}_{k_c}$,

where $\tilde{\boldsymbol{\Lambda}}^c_{k_c} \in \mathbb{R}^{k_c \times k_c}$ is a diagonal matrix of lower eigenvalues and $\bar{\tilde{\boldsymbol{\Lambda}}}^c_{k_c} \in \mathbb{R}^{(\rho_c - k_c)\times(\rho_c - k_c)}$ is a diagonal matrix of higher graph eigenvalues.

Similarly, let $\tilde{\boldsymbol{L}}^r = \tilde{\boldsymbol{P}}^r \tilde{\boldsymbol{\Lambda}}^r \tilde{\boldsymbol{P}}^{r\top} = \tilde{\boldsymbol{P}}^r_{k_r} \tilde{\boldsymbol{\Lambda}}^r_{k_r} \tilde{\boldsymbol{P}}^{r\ \top}_{k_r} + \bar{\tilde{\boldsymbol{P}}}^r_{k_r} \bar{\tilde{\boldsymbol{\Lambda}}}^r_{k_r} \bar{\tilde{\boldsymbol{P}}}^{r\ \top}_{k_r}$,

where all the values in $\tilde{\boldsymbol{\Lambda}}^r$ and $\tilde{\boldsymbol{\Lambda}}^c$ are sorted in increasing order.

Assume $\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{X}}^* + \boldsymbol{E}$, where $\boldsymbol{E}$ models the noise in the compressed data and $\tilde{\boldsymbol{X}}^* \in \mathbb{MLT}(\tilde{\boldsymbol{P}}^{\mu}_k)$. The low-rank matrix $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{X}}^* + \tilde{\boldsymbol{E}}$ can be recovered by solving the FRPCAG problem as proposed in [110] and re-written below:

$$\min_{\tilde{\boldsymbol{X}}} \phi(\tilde{\boldsymbol{Y}} - \tilde{\boldsymbol{X}}) + \gamma_c \operatorname{tr}(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{L}}^c \tilde{\boldsymbol{X}}^\top) + \gamma_r \operatorname{tr}(\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{L}}^r \tilde{\boldsymbol{X}}), \tag{6.4}$$

where ϕ is a proper, positive, convex and lower semi-continuous loss function (possibly l_p norm). From Theorem 1 in [110], the low-rank approximation error comprises the orthogonal projection of $\tilde{\boldsymbol{X}}^*$ on the complement graph eigenvectors $(\bar{\tilde{\boldsymbol{P}}}^c_{k_c}, \bar{\tilde{\boldsymbol{P}}}^r_{k_r})$ and depends on the eigen-gaps $\tilde{\lambda}^c_{k_c}/\tilde{\lambda}^c_{k_c+1}, \tilde{\lambda}^r_{k_r}/\tilde{\lambda}^r_{k_r+1}$ as following:

$$\|\tilde{\boldsymbol{X}}\bar{\tilde{\boldsymbol{P}}}^c_{k_c}\|_F^2 + \|\boldsymbol{P}^{\bar{\tilde{r}}}_{k_r}{}^{\top}\tilde{\boldsymbol{X}}\|_F^2 = \|\tilde{\boldsymbol{E}}\|_F^2 \le \frac{1}{\gamma}\phi(\boldsymbol{E}) + \|\tilde{\boldsymbol{X}}^*\|_F^2 \left(\frac{\tilde{\lambda}^c_{k_c}}{\tilde{\lambda}^c_{k_c+1}} + \frac{\tilde{\lambda}^r_{k_r}}{\tilde{\lambda}_{k_r+1}}\right), \tag{6.5}$$

where γ depends on the signal-to-noise ratio. Clearly, if $\lambda^c_{k_c}/\lambda^c_{k_c+1} \approx 0$ and $\lambda^r_{k_r}/\lambda^r_{k_r+1} \approx 0$ and the compressed Laplacians are constructed using the Kron reduction then $\tilde{\lambda}^c_{k_c}/\tilde{\lambda}^c_{k_c+1} \approx 0$ and $\tilde{\lambda}^r_{k_r}/\tilde{\lambda}^r_{k_r+1} \approx 0$. Thus, exact recovery is attained.

6.5 Decoders for low-rank recovery

Let $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r \times \rho_c}$ be the low-rank solution of (6.4) with the compressed graph Laplacians $\tilde{\boldsymbol{L}}^r, \tilde{\boldsymbol{L}}^c$ and sampled data $\tilde{\boldsymbol{Y}}$. The goal is to decode the low-rank matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ for the full $\boldsymbol{Y}$. We assume that,

$$\tilde{\boldsymbol{X}} = \boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c + \tilde{\boldsymbol{E}},$$

where $\tilde{\boldsymbol{E}} \in \mathbb{R}^{\rho_r \times \rho_c}$ models the noise incurred by (6.4) and characterized by eq. (6.5). The only information available for the decode stage is: 1) the Laplacians $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$ of the matrix $\boldsymbol{Y}$ and 2) the data matrix $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^{\mu}_k)$.

6.5.1 Ideal Decoder

A straight-forward way to decode $\boldsymbol{X}$ on the original graphs $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$, when one knows the basis $\boldsymbol{P}^r_{k_r}, \boldsymbol{P}^c_{k_c}$ involves solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{M}^r \boldsymbol{X} \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F^2 \\ &\text{s.t:} \quad \boldsymbol{X}(i) \in span(\boldsymbol{P}^r_{k_r}), \ \boldsymbol{X}^{\top}(i) \in span(\boldsymbol{P}^c_{k_c}). \end{aligned} \tag{6.6}$$

Theorem 6.5.1 below establishes an upper error bound for the recovery of the complete low-rank matrix using this ideal decoder.

Theorem 6.5.1. *Let $\boldsymbol{M}^r$ and $\boldsymbol{M}^c$ be such that* (6.2) *holds and $\boldsymbol{X}^*$ be the solution of* (6.6) *with $\tilde{\boldsymbol{X}} = \boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c + \tilde{\boldsymbol{E}}$, where $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^{\mu}_k)$ and $\tilde{\boldsymbol{E}} \in \mathbb{R}^{\rho_r \times \rho_c}$. We have:*

$$\|\bar{\boldsymbol{X}}^* - \boldsymbol{X}^*\|_F \le 2\sqrt{\frac{np}{\rho_c \rho_r (1-\delta)}}\|\tilde{\boldsymbol{E}}\|_F, \tag{6.7}$$

where $\sqrt{np/\rho_c\rho_r(1-\delta)}$ is a constant resulting from the norm preservation in (6.2) *and $\|\tilde{\boldsymbol{E}}\|_F^2$ is bounded by eq.* (6.5).

Proof. Please refer to Appendix A.5. □

Thus, the error of the ideal decoder is only bounded by the error $\tilde{\boldsymbol{E}}$ in the low-rank matrix $\tilde{\boldsymbol{X}}$ obtained by solving (6.4). In fact $\tilde{\boldsymbol{E}}$ depends on the spectral gaps of $\tilde{\boldsymbol{L}}^c, \tilde{\boldsymbol{L}}^r$, as given in eq. (6.5). So,

$$\|\bar{\boldsymbol{X}}^* - \boldsymbol{X}^*\|_F \leq 2\sqrt{\frac{np}{\rho_c\rho_r(1-\delta)}}\left[\frac{1}{\gamma}\phi(\boldsymbol{E}) + \|\tilde{\boldsymbol{X}}^*\|_F^2\left(\frac{\tilde{\lambda}^c_{k_c}}{\tilde{\lambda}^c_{k_c+1}} + \frac{\tilde{\lambda}^r_{k_r}}{\tilde{\lambda}_{k_r+1}}\right)\right].$$

Hence, the ideal decoder itself does not introduce any error in the decode stage.

Complexity: The solution for this decoder requires projecting over the eigenvectors $\boldsymbol{P}^r$ and $\boldsymbol{P}^c$ of $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$. As our motivation behind CPCA is to design a highly scalable low-rank recovery framework for the case of very large n, it is computationally prohibitive to determine the graph eigenvectors, even though the cost to determine these eigenvectors scales linearly with n, i.e., $\mathcal{O}(nk^2)$, where k is the number of eigenvectors. Moreover, even if one desires to solve the above optimization problem, the constants k_r, k_c are not known beforehand and require tuning. This poses even more computational challenges. Nevertheless, the problems of the form eq. (6.6) are interesting and a solution to such problems is connected with out GMLSVD framework, which was presented in Chapter 4. We will get back to such problems in detail in Chapter 7, where we will explicitly make use of the graph eigenvectors to recover low-rank tensors for which the number of samples n in every dimension is not large enough.

6.5.2 Alternate Decoder

As the ideal decoder is computationally costly, we propose to decode $\boldsymbol{X}$ from $\tilde{\boldsymbol{X}}$ by using a convex and computationally tractable problem which involves the minimization of graph dirichlet energies.

$$\min_{\boldsymbol{X}} \|\boldsymbol{M}^r\boldsymbol{X}\boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F^2 + \bar{\gamma}_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \bar{\gamma}_r \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X}). \tag{6.8}$$

Theorem 6.5.2 below establishes an upper error bound for the recovery of the complete low-rank matrix using this alternate decoder.

Theorem 6.5.2. *Let $\boldsymbol{M}^r$ and $\boldsymbol{M}^c$ be such that eq.* (6.2) *holds and $\gamma > 0$. Let also $\bar{\boldsymbol{X}}^* = \underline{\boldsymbol{X}}^* + \boldsymbol{E}^*$ be the solution of eq.* (6.8) *with $\bar{\gamma}_c = \gamma/\lambda^c_{k_c+1}$, $\bar{\gamma}_r = \gamma/\lambda^r_{k_r+1}$, and $\tilde{\boldsymbol{X}} = \boldsymbol{M}^r\boldsymbol{X}^*\boldsymbol{M}^c + \tilde{\boldsymbol{E}}$, where $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ and $\tilde{\boldsymbol{E}} \in \mathbb{R}^{\rho_r\times\rho_c}$. We have:*

$$\|\underline{\boldsymbol{X}}^* - \boldsymbol{X}^*\|_F \leq \sqrt{\frac{np}{\rho_c\rho_r(1-\delta)}}\left[\left(2 + \frac{1}{\sqrt{2\gamma}}\right)\|\tilde{\boldsymbol{E}}\|_F + (\frac{1}{\sqrt{2}} + \sqrt{\gamma})\sqrt{\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}} + \frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F\right], \quad \textit{and}$$

$$\|\boldsymbol{E}^*\|_F \leq \frac{\|\tilde{\boldsymbol{E}}\|_F}{\sqrt{2\gamma}} + \frac{1}{\sqrt{2}}\sqrt{\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}} + \frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F, \tag{6.9}$$

where $\underline{\boldsymbol{X}}^* = \operatorname{Proj}_{\mathbb{MLT}(\boldsymbol{P}^\mu_k)}(\boldsymbol{X})$ and $\boldsymbol{E}^* = \bar{\boldsymbol{X}}^* - \underline{\boldsymbol{X}}^*$. $\operatorname{Proj}_{\mathbb{MLT}(\boldsymbol{P}^\mu_k)}(.)$ denotes the orthogonal projection onto $\mathbb{MLT}(\boldsymbol{P}^\mu_k)$ and γ depends on the signal to noise ratio.

Proof. Please refer to Appendix A.6 ☐

Theorem 6.5.2 states that in addition to the error $\tilde{\boldsymbol{E}}$ in $\tilde{\boldsymbol{X}}$ incurred by (6.4) and characterized by the bound in eq. (6.5), the error of the alternate decoder (6.8) also depends on the eigen-gaps of the Laplacians $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$ respectively. This is the price that one has to pay in order to avoid the expensive ideal decoder. For a k_r, k_c clusterable data $\boldsymbol{Y}$ across the rows and columns, one can expect $\lambda^r_{k_r}/\lambda^r_{k_r+1} \approx 0$

and $\lambda^c_{k_c}/\lambda^c_{k_c+1} \approx 0$ and the solution is as good as the ideal decoder. Nevertheless, it is possible to reduce this error by using graph filters g such that the ratios $g(\lambda^c_{k_c})/g(\lambda^c_{k_c+1})$ and $g(\lambda^r_{k_r})/g(\lambda^r_{k_r+1})$ approach zero. However, we do not discuss this approach in our work.

Complexity: It is trivial to solve (6.8) using a conjugate gradient scheme that costs $\mathcal{O}(Inpk_{nn})$, where I is the number of iterations for the algorithm to converge.

6.5.3 Approximate Decoder

The alternate decoder proposed above has a few disadvantages. First, note that it is almost as computationally expensive as FRPCAG. Recall that our proposed method FRPCAG has a computational complexity that scales with $\mathcal{O}(np)$ per iteration. Since, the primary goal of CPCA is to come up with a method that scales better than FRPCAG, the alternate decoder above is not feasible. Second, note that It requires tuning two model parameters, which is computationally unfeasible as well.

In this section we describe the step-by-step construction of an approximate, parameter free and computationally feasible decoder. The main idea is to breakdown the decode phase of low-rank matrix $\boldsymbol{X}$ into its left and right singular vectors or subspaces. Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^\top$ be the SVD of $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$. We propose to recover $\boldsymbol{U}$ from $\tilde{\boldsymbol{U}}$ and $\boldsymbol{V}$ from $\tilde{\boldsymbol{V}}$ in 3 steps.

1. Split the alternate decoder to subspace learning problems.
2. Drop the orthonormality constraints on subspaces.
3. Run an efficient upsampling algorithm to solve the problem of step 2.

The goal of this step-by-step approach is to guide the reader throughout to observe the close relationship between the alternate and approximate decoder. Note that $\boldsymbol{U} = \boldsymbol{U}_r$ and $\boldsymbol{V} = \boldsymbol{V}_r$, i.e., both $\boldsymbol{U}$ and $\boldsymbol{V}$ have r columns, however, we use $\boldsymbol{U}$ and $\boldsymbol{V}$ to keep the notation simple throughout our discussion. Now we begin to describe these steps in detail.

Step 1: Splitting the alternate decoder

Using the SVD of $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ and the invariance property of trace under cyclic permutations, we can replace (6.8) by:

$$\begin{aligned}&\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{M}^r\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{M}^c - \tilde{\boldsymbol{U}}\tilde{\Sigma}\tilde{\boldsymbol{V}}^\top\|_F^2 + \bar{\gamma}_c\,\mathrm{tr}(\boldsymbol{\Sigma}^2\boldsymbol{V}^\top\boldsymbol{L}^c\boldsymbol{V}) + \bar{\gamma}_r\,\mathrm{tr}(\boldsymbol{U}^\top\boldsymbol{L}^r\boldsymbol{U}\boldsymbol{\Sigma}^2)\\ &\quad\text{s.t:}\ \ \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_r,\ \ \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}_r.\end{aligned} \tag{6.10}$$

The above eq. introduces two new variables based on the SVD of $\boldsymbol{X}$, i.e, $\boldsymbol{U} \in \mathbb{R}^{p\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$. Clearly, with the introduction of these new variables, one needs to specify r as the dimension of the subspaces $\boldsymbol{U}$ and $\boldsymbol{V}$. We propose the following strategy for this:

1. First, determine $\tilde{\boldsymbol{\Sigma}}$ by one inexpensive SVD of $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r\times\rho_c}$. This costs $\mathcal{O}(\rho_r^2\rho_c)$ for $\rho_r < \rho_c$.
2. Then set r equal to the number of entries in $\tilde{\boldsymbol{\Sigma}}$ which are above a threshold.

It is important to note that so far eq. (6.10) and the alternate decoder eq. (6.8) are equivalent. Also note that we did not introduce the singular values $\boldsymbol{\Sigma}$ as an additional variable in eq. (6.10) because they are related to the singular values $\tilde{\boldsymbol{\Sigma}}$ of $\tilde{\boldsymbol{X}}$. We argue this as following: If eq. (6.9) holds for the alternate decoder then $\|\tilde{\boldsymbol{\Sigma}}^* - \tilde{\boldsymbol{\Sigma}}\|_F$ (where $\tilde{\boldsymbol{\Sigma}}^*, \tilde{\boldsymbol{\Sigma}}$ are the singular values of $\tilde{\boldsymbol{X}}^*, \tilde{\boldsymbol{X}}$) is also bounded as argued below.

Let $\boldsymbol{M}^r \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top \boldsymbol{M}^c = \tilde{\boldsymbol{X}}^*$, then we observe that

$$\|\tilde{\boldsymbol{X}}^* - \tilde{\boldsymbol{X}}\|_F^2 = \|\tilde{\boldsymbol{U}}^* \tilde{\boldsymbol{\Sigma}}^* \tilde{\boldsymbol{V}}^{*\top} - \tilde{\boldsymbol{U}} \tilde{\boldsymbol{\Sigma}} \tilde{\boldsymbol{V}}^\top\|_F^2 = \|\tilde{\boldsymbol{U}}^\top \tilde{\boldsymbol{U}}^* \tilde{\boldsymbol{\Sigma}}^* \tilde{\boldsymbol{V}}^\top \tilde{\boldsymbol{V}}^* - \tilde{\boldsymbol{\Sigma}}\|_F^2$$

which implies that $\tilde{\boldsymbol{U}}^\top \tilde{\boldsymbol{U}}^* \tilde{\boldsymbol{\Sigma}}^* \tilde{\boldsymbol{V}}^\top \tilde{\boldsymbol{V}}^* \approx \tilde{\boldsymbol{\Sigma}}$. This is equivalent to saying that for the significant values of $\tilde{\boldsymbol{\Sigma}}$, the inner product of orthonormal matrices $\tilde{\boldsymbol{U}}^\top \tilde{\boldsymbol{U}}^*$ and $\tilde{\boldsymbol{V}}^\top \tilde{\boldsymbol{V}}^*$ have to be almost diagonal. As a result, for the significant values of $\tilde{\boldsymbol{\Sigma}}$, $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ have to be aligned with $\tilde{\boldsymbol{U}}^*$ and $\tilde{\boldsymbol{V}}^*$. The same reason also implies that $\tilde{\boldsymbol{\Sigma}}^* \approx \tilde{\boldsymbol{\Sigma}}$. Thus, the singular values $\boldsymbol{\Sigma}$ and $\tilde{\boldsymbol{\Sigma}}$ of $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ differ approximately by the normalization constant of theorem 6.3.1, i.e,

$$\boldsymbol{\Sigma} = \sqrt{\frac{np}{\rho_r \rho_c (1-\delta)}} \tilde{\boldsymbol{\Sigma}}$$

Note that with the above relationship, the subspaces $\boldsymbol{U}, \boldsymbol{V}$ can be determined independently of each other. Thus eq. (6.10) can be decoupled as following which separately solves the subspace ($\boldsymbol{U}$ and $\boldsymbol{V}$) learning problems.

$$\begin{aligned}
&\min_{\boldsymbol{U}} \|\boldsymbol{M}^r \boldsymbol{U} - \tilde{\boldsymbol{U}}\|_F^2 + \gamma'_r \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \\
&\qquad \text{s.t:} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r, \\
&\min_{\boldsymbol{V}} \|\boldsymbol{V}^\top \boldsymbol{M}^c - \tilde{\boldsymbol{V}}\|_F^2 + \gamma'_c \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V}) \\
&\qquad \text{s.t:} \quad \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}_r.
\end{aligned} \tag{6.11}$$

Step 2: Dropping Orthonormality Constraints

Solving eq. (6.11) is as expensive as eq. (6.8) due to the orthonormality constraints as explained below.

First, note that the problem can be reformulated as:

$$\begin{aligned}
&\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \\
&\qquad \text{s.t:} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r, \quad \|\boldsymbol{M}^r \boldsymbol{U} - \tilde{\boldsymbol{U}}\|_F^2 < \epsilon
\end{aligned}$$

Let $\boldsymbol{U}'$ is the zero appended matrix of $\tilde{\boldsymbol{U}}$, then we can re-write it as:

$$\begin{aligned}
&\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \\
&\qquad \text{s.t:} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r, \quad \|\boldsymbol{M}^r (\boldsymbol{U} - \boldsymbol{U}')\|_F^2 < \epsilon
\end{aligned}$$

The above problem is equivalent to eq. (6.11), as the term $\|\boldsymbol{M}^r (\boldsymbol{U} - \boldsymbol{U}')\|_F^2$ has been removed from the objective and introduced as a constraint. Note that the constant γ_r is not needed anymore. The new model parameter ϵ controls the radius of the ℓ_2 ball $\|\boldsymbol{M}^r (\boldsymbol{U} - \boldsymbol{U}')\|_F^2$. In simple words it controls how much noise is tolerated by the projection of $\boldsymbol{U}$ on the ball that is centered at $\boldsymbol{U}'$. To solve the above problem one needs to split it down into two sub-problems and solve iteratively between:

1. The optimization $\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \quad \text{s.t:} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r$. The solution to this problem is given by the lowest r eigenvectors of $\boldsymbol{L}^r$. Thus it requires a complexity of $\mathcal{O}((n+p)r^2)$ for solving both problems (6.11).

2. The projection on the ℓ_2 ball $\|\boldsymbol{M}^r(\boldsymbol{U}-\boldsymbol{U}^{'})\|_F^2$ whose complexity is $\mathcal{O}(\rho_c+\rho_r)$.

Thus the solution requires a double iteration with a complexity of $\mathcal{O}(Inr^2)$ and is almost as expensive as FRPCAG.

Therefore, we drop the constraints and get

$$\min_{\boldsymbol{U}} \|\boldsymbol{M}^r\boldsymbol{U}-\tilde{\boldsymbol{U}}\|_F^2 + {\gamma^{'}}_r \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}), \tag{6.12}$$

$$\min_{\boldsymbol{V}} \|\boldsymbol{V}^\top\boldsymbol{M}^c-\tilde{\boldsymbol{V}}\|_F^2 + {\gamma^{'}}_c \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V}). \tag{6.13}$$

The solutions to (6.12) & (6.13) are not orthonormal anymore. The deviation from the orthonormality depends on the constants $\gamma_r^{'}$ and $\gamma_c^{'}$, but $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is still a good enough (error characterized in Theorem 6.5.3) low-rank representation due to the intuitive explanation that we present here. We argue that the solutions of eqs.(6.12) &(6.13) are feasible solutions of the joint non-convex, factorized, and graph regularized low-rank optimization problem like the one presented in [34]. Let A and B be the subspaces that we want to recover then we can re-write the problem studied in [34] as following:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \|\boldsymbol{M}^r\boldsymbol{A}\boldsymbol{B}^\top\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F^2 + \gamma_r^{'} \operatorname{tr}(\boldsymbol{A}^\top \boldsymbol{L}^r \boldsymbol{A}) + \gamma_c^{'} \operatorname{tr}(\boldsymbol{B}^\top \boldsymbol{L}^c \boldsymbol{B})$$

The above non-convex problem does not require $\boldsymbol{A}$ and $\boldsymbol{B}$ to be orthonormal, but is still widely used for recovering a low-rank $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}^\top$. Our problem setting (eqs.(6.12) &(6.13)) is just equivalent except that it is convex as we decouple the learning of two subspaces due to the known $\boldsymbol{\Sigma}$ that relates $\boldsymbol{U}$ and $\boldsymbol{V}$. Thus, for any orthonormal $\boldsymbol{U}, \boldsymbol{V}$ and a scaling matrix $\boldsymbol{\Sigma}$, $\boldsymbol{A} = \boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}$ and $\boldsymbol{B} = \boldsymbol{V}\sqrt{\boldsymbol{\Sigma}}$ is a feasible solution.

Step 3: Subspace Upsampling

Eqs. (6.12) &(6.13) require the tuning of two parameters $\gamma_r^{'}$ and $\gamma_c^{'}$ which can be computationally cumbersome. Therefore, our final step in the construction of the approximate decoder is to get rid of the two parameters. But before we present the final construction step we study the problems eqs.(6.12) &(6.13) and their solutions more closely.

First, note that solving eqs.(6.12) &(6.13) is equivalent to making the following assumptions:

$$\tilde{\boldsymbol{U}} = \boldsymbol{M}^r\bar{\boldsymbol{U}} + \tilde{\boldsymbol{E}}^u \quad \text{and} \quad \tilde{\boldsymbol{V}} = \bar{\boldsymbol{V}}\boldsymbol{M}^c + \tilde{\boldsymbol{E}}^v,$$

where the columns of $\bar{\boldsymbol{U}}$, $\bar{\boldsymbol{U}(i)} \in span(\boldsymbol{P}^r_{k_r})$, $i = 1, \cdots, p$, and the columns of $\bar{\boldsymbol{V}}$ $\bar{\boldsymbol{V}(j)} \in span(\boldsymbol{P}^c_{k_c})$, $j = 1, \cdots, n$ and $\tilde{\boldsymbol{E}}^u \in \mathbb{R}^{\rho_r \times \rho_r}$, $\tilde{\boldsymbol{E}}^v \in \mathbb{R}^{\rho_c \times \rho_c}$ model the noise in the estimate of the subspaces.

Secondly, the closed form solutions of eqs.(6.12) &(6.13) are given as following:

$$\boldsymbol{U} = ({\boldsymbol{M}^r}^\top\boldsymbol{M}^r + \gamma_r^{'}\boldsymbol{L}^r)^{-1}{\boldsymbol{M}^r}^\top\tilde{\boldsymbol{U}}, \tag{6.14}$$

$$\boldsymbol{V} = (\boldsymbol{M}^c \boldsymbol{M}^{c\top} + \gamma'_c \boldsymbol{L}^c)^{-1} \boldsymbol{M}^c \tilde{\boldsymbol{V}}. \tag{6.15}$$

Thus, problems (6.12) & (6.13) decode the subspaces $\boldsymbol{U}$ and $\boldsymbol{V}$ such that they are smooth on their respective graphs $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$. This can also be referred to as:

1. simultaneous decoding and
2. subspace denoising stage.

We call it a 'subspace denoising' method because the operator $(\boldsymbol{M}^{r\top} \boldsymbol{M}^r + \gamma'_r \boldsymbol{L}^r)^{-1}$ can be viewed as low-pass filtering the subspace $\boldsymbol{U}$ in the graph fourier domain.

Note that we want to decode and denoise $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ which are in turn determined by the SVD of $\tilde{\boldsymbol{X}}$. Furthermore, $\tilde{\boldsymbol{X}}$ has been determined by solving the FRPCAG problem of eq.(6.4). FRPCAG is already robust to noise and outliers, therefore, it is safe to assume that the subspaces determined from it, i.e, $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ are also noise and outlier free. Thus, the extra denoising step (performed via graph filtering) of eqs.(6.12) &(6.13) is redundant.

Therefore, we can directly upsample $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ to determine $\boldsymbol{U}$ and $\boldsymbol{V}$ without needing a margin for noise. To do this, we reformulate eq.(6.12) as follows:

$$\min_{\boldsymbol{U}} \frac{1}{\gamma'_r} \|\boldsymbol{M}^r \boldsymbol{U} - \tilde{\boldsymbol{U}}\|_F^2 + \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}).$$

For $\gamma'_r \to 0$, $\frac{1}{\gamma'_r} \to \infty$, the emphasis on first term of the objective increases and it turns to an equality constraint $\boldsymbol{M}^r \boldsymbol{U} = \tilde{\boldsymbol{U}}$. The same holds for eq.(6.13) as well. Thus, the modified problems are:

$$\begin{aligned} &\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \qquad \text{and} \qquad && \min_{\boldsymbol{V}} \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V}) \\ &\text{s.t: } \boldsymbol{M}^r \boldsymbol{U} = \tilde{\boldsymbol{U}}, && \text{s.t: } \boldsymbol{M}^{c\top} \boldsymbol{V} = \tilde{\boldsymbol{V}}. \end{aligned} \tag{6.16}$$

Note that now we have a parameter-free decoding stage.

It is important now to study the theoretical guarantees on eqs. (6.16). To do this, as eqs. (6.16) are a specific case of eqs. (6.12) & (6.13), we first study the guarantees on eqs. (6.12) & (6.13) in Theorem 6.5.3. Then, based on this study we directly present the guarantees on the *final approximate decoder* of eqs. (6.16) in Theorem 6.5.4.

Theorem 6.5.3. *Let* $\boldsymbol{M}^r$ *and* $\boldsymbol{M}^c$ *be such that* (6.2) *holds and* $\gamma'_r, \gamma'_c > 0$. *Let also* $\boldsymbol{U}^*$ *and* $\boldsymbol{V}^*$ *be respectively the solutions of* (6.12) *and* (6.13) *with* $\tilde{\boldsymbol{U}} = \boldsymbol{M}^r \bar{\boldsymbol{U}} + \tilde{\boldsymbol{E}}^u$ *and* $\tilde{\boldsymbol{V}} = \boldsymbol{M}^c \bar{\boldsymbol{V}} + \tilde{\boldsymbol{E}}^v$, *where* $\bar{\boldsymbol{U}}(i) \in span(\boldsymbol{P}^r_{k_r})$, $i = 1, \cdots, p$, $\bar{\boldsymbol{V}}(j) \in span(\boldsymbol{P}^c_{k_c})$, $j = 1, \cdots, n$, $\tilde{\boldsymbol{E}}^u \in \mathbb{R}^{\rho_r \times \rho_r}$, $\tilde{\boldsymbol{E}}^v \in \mathbb{R}^{\rho_c \times \rho_c}$. *We have:*

$$\|\bar{\boldsymbol{U}}^* - \bar{\boldsymbol{U}}\|_F \leq \sqrt{\frac{2p}{\rho_r(1-\delta)}} \left[\left(2 + \frac{1}{\sqrt{\gamma'_r \lambda^r_{k_r+1}}} \right) \|\tilde{\boldsymbol{E}}^u\|_F + \left(\sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}} + \sqrt{\gamma'_r \lambda^r_{k_r}} \right) \|\bar{\boldsymbol{U}}\|_F \right], \text{ and}$$

$$\|\boldsymbol{E}^*\|_F \leq \sqrt{\frac{2}{\gamma'_r \lambda^r_{k_r+1}}}\|\tilde{\boldsymbol{E}}^u\|_F + \sqrt{2\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}}\|\bar{\boldsymbol{U}}\|_F.$$

where $\bar{\boldsymbol{U}}^* = \boldsymbol{P}^r_{k_r}\boldsymbol{P}^{r\ \top}_{k_r}\boldsymbol{U}^*$ *and* $\boldsymbol{E}^* = \boldsymbol{U}^* - \bar{\boldsymbol{U}}^*$. *The same inequalities with slight modification also hold for* $\boldsymbol{V}^*$*,*

$$\|\bar{\boldsymbol{V}}^* - \bar{\boldsymbol{V}}\|_F \leq \sqrt{\frac{2n}{\rho_c(1-\delta)}}\left[\left(2 + \frac{1}{\sqrt{\gamma'_c \lambda^c_{k_c+1}}}\right)\|\tilde{\boldsymbol{E}}^v\|_F + \left(\sqrt{\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}} + \sqrt{\gamma'_c \lambda^c_{k_c}}\right)\|\bar{\boldsymbol{V}}\|_F\right], \quad \textit{and}$$

$$\|\boldsymbol{E}^*\|_F \leq \sqrt{\frac{2}{\gamma'_c \lambda^c_{k_c+1}}}\|\tilde{\boldsymbol{E}}^v\|_F + \sqrt{2\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}}\|\bar{\boldsymbol{V}}\|_F,$$

where $\bar{\boldsymbol{V}}^* = \boldsymbol{V}^*\boldsymbol{P}^{c\ \top}_{k_c}\boldsymbol{P}^c_{k_c}$ *and* $\boldsymbol{E}^* = \boldsymbol{V}^* - \bar{\boldsymbol{V}}^*$.

Proof. Please refer to Appendix A.7. □

Theorem 6.5.4. *Let* $\boldsymbol{M}^r$ *and* $\boldsymbol{M}^c$ *be such that* (6.2) *holds. Let also* $\boldsymbol{U}^*$ *and* $\boldsymbol{V}^*$ *be the solutions of* (6.16) *with* $\tilde{\boldsymbol{U}} = \boldsymbol{M}^r\bar{\boldsymbol{U}}$ *and* $\tilde{\boldsymbol{V}} = \boldsymbol{M}^c\bar{\boldsymbol{V}}$, *where* $\bar{\boldsymbol{U}(i)} \in span(\boldsymbol{P}^r_{k_r})$, $i = 1, \cdots, p$, $\bar{\boldsymbol{V}(j)} \in span(\boldsymbol{P}^c_{k_c})$, $j = 1, \cdots, n$. *We have:*

$$\|\boldsymbol{U}^* - \bar{\boldsymbol{U}}\|_F \leq \sqrt{\frac{2p}{\rho_r(1-\delta)}}\sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}}\|\bar{\boldsymbol{U}}\|_F$$

where $\boldsymbol{U}^* = \boldsymbol{P}^r_{k_r}\boldsymbol{P}^{r\ \top}_{k_r}\boldsymbol{U}$. *The same inequalities with slight modification also hold for* $\boldsymbol{V}^*$*,*

$$\|\boldsymbol{V}^* - \bar{\boldsymbol{V}}\|_F \leq \sqrt{\frac{2n}{\rho_c(1-\delta)}}\sqrt{\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}}\|\bar{\boldsymbol{V}}\|_F$$

where $\boldsymbol{V}^* = \boldsymbol{V}\boldsymbol{P}^{c\ \top}_{k_c}\boldsymbol{P}^c_{k_c}$

Proof. The proof directly follows from the proof of Theorem 6.5.3 by using $\tilde{\boldsymbol{E}}^u = 0$ and $\gamma'_r = 0$. □

As $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^\top$, we can say that the error with eqs.(6.16) is upper bounded by the product of the errors of the individual subspace decoders. Also note that the error again depends on the eigen-gaps defined by the ratios $\lambda^c_{k_c}/\lambda^c_{k_c+1}$ and $\lambda^r_{k_r}/\lambda^r_{k_r+1}$.

The solution to the above problems (eqs. 6.16) is simply a graph upsampling operation as explained in Lemma 6.5.1.

Lemma 6.5.1. *Let* $\boldsymbol{S} \in \mathbb{R}^{c \times r}$ *and* $\boldsymbol{R} \in \mathbb{R}^{d \times r}$ *be the two matrices such that* $d < r$ *and* $d < c$. *Furthermore, let* $\boldsymbol{M} \in \Re^{d \times c}$ *be a sampling matrix as constructed in* (6.1) *and* $\boldsymbol{L} \in \mathbb{R}^{c \times c}$ *be a symmetric positive semi-definite matrix. We can write* $\boldsymbol{S} = [\boldsymbol{S}_a^\top | \boldsymbol{S}_b^\top]^\top$, *where* $\boldsymbol{S}_b \in \mathbb{R}^{d \times r}$ *and* $\boldsymbol{S}_a \in \Re^{(c-d) \times r}$ *are the known and unknown submatrices of* $\boldsymbol{S}$. *Then the exact and unique solution to the following problem:*

$$\min_{\boldsymbol{S}_a} \operatorname{tr}(\boldsymbol{S}^\top \boldsymbol{L}\boldsymbol{S}), \quad \textit{s.t:} \quad \boldsymbol{M}\boldsymbol{S} = \boldsymbol{R} \tag{6.17}$$

is given by $\boldsymbol{S}_a = -\boldsymbol{L}_{aa}^{-1}\boldsymbol{L}_{ab}\boldsymbol{R}$.

Proof. Please refer to Appendix A.8. □

Using Lemma 6.5.1 and the notation of Section 6.4.1 we can write:

$$\boldsymbol{U} = \begin{bmatrix} -\boldsymbol{L}^{r^{-1}}(\bar{\Omega}_r, \bar{\Omega}_r)\boldsymbol{L}^r(\bar{\Omega}_r, \Omega_r)\tilde{\boldsymbol{U}} \\ \tilde{\boldsymbol{U}} \end{bmatrix}$$
$$\boldsymbol{V} = \begin{bmatrix} -\boldsymbol{L}^{c^{-1}}(\bar{\Omega}_c, \bar{\Omega}_c)\boldsymbol{L}^c(\bar{\Omega}_c, \Omega_c)\tilde{\boldsymbol{V}} \\ \tilde{\boldsymbol{V}} \end{bmatrix}. \tag{6.18}$$

Eqs. (6.18) involves solving a sparse linear system. If each connected component of the graph has at least one labeled element, $\boldsymbol{L}^r(\bar{\Omega}_r, \bar{\Omega}_r)$ is full rank and invertible. If the linear system above is not large then one can directly use eq. (6.18). However, to avoid inverting the big matrix we can use the standard Preconditioned Conjugate Gradient (PCG) method to solve it. Note that the eqs. (6.18) and even PCG can be implemented in parallel for every column of $\boldsymbol{U}$ and $\boldsymbol{V}$. This gives a significant advantage over the alternate decoder in terms of computation time.

Complexity: The cost of this decoder is $\mathcal{O}(O_l k_{nn} kn)$ where O_l is the number of iterations for the PCG method.

The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are not normalized with the above solution, therefore, a unit norm normalization step is needed at the end. Once $\boldsymbol{U}, \boldsymbol{V}$ are determined, one can use $\boldsymbol{X} = \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^\top\sqrt{np/\rho_r\rho_c(1-\delta)}$ to determine the required low-rank matrix $\boldsymbol{X}$. The decoder for approximate recovery is presented in Algorithm 2.

Algorithm 2 Subspace Upsampling based Approximate Decoder for low-rank recovery

```
INPUT: $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r \times \rho_c}$, $\boldsymbol{L}^r \in \mathbb{R}^{p \times p}$, $\boldsymbol{L}^c \in \mathbb{R}^{n \times n}$
1. do $SVD(\tilde{\boldsymbol{X}}) = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^\top$
2. find $r$ such that $\tilde{\boldsymbol{\Sigma}}_{r,r}/\tilde{\boldsymbol{\Sigma}}_{1,1} < 0.1$
3. Solve eqs. (6.16) for every column of $\boldsymbol{U}, \boldsymbol{V}$ as following:
for $i = 1, \dots r$ do
    solve $\min_{\boldsymbol{U}(i)} \boldsymbol{U}(i)^\top \boldsymbol{L}^r \boldsymbol{U}(i)$   s.t $\boldsymbol{M}^r \boldsymbol{U}(i) = \tilde{\boldsymbol{U}}(i)$ using PCG
    solve $\min_{\boldsymbol{V}(i)} \boldsymbol{V}(i)^\top \boldsymbol{L}^c \boldsymbol{V}(i)$   s.t $\boldsymbol{M}^{c\top} \boldsymbol{V}(i) = \tilde{\boldsymbol{V}(i)}$ using PCG
end for
4. Set $\boldsymbol{U}(i) = \boldsymbol{U}(i)/\|\boldsymbol{U}(i)\|_F, \boldsymbol{V}(i) = \boldsymbol{V}(i)/\|\boldsymbol{V}(i)\|_F, \forall i = 1, \cdots, r$
5. Set $\boldsymbol{\Sigma} = \sqrt{\frac{np}{\rho_r\rho_c(1-\delta)}}\tilde{\boldsymbol{\Sigma}}$
6. Set $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
OUTPUT: The full low-rank $\boldsymbol{X} \in \mathbb{R}^{p \times n}$
```

6.5.4 Other Approximate Decoders

Alternatively, if the complete data matrix $\boldsymbol{Y}$ is available then we can reduce the complexity further by performing a graph-upsampling for only one of the two subspaces $\boldsymbol{U}$ or $\boldsymbol{V}$.

Approximate decoder 2

Suppose we do the upsampling only for $\boldsymbol{U}$, then the approximate decoder 2 can be written as:

$$\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U}) \quad \text{s.t:} \quad \boldsymbol{M}^r \boldsymbol{U} = \tilde{\boldsymbol{U}}.$$

The solution for $\boldsymbol{U}$ is given by eq. 6.18. Then, we can write $\boldsymbol{V}$ as:

$$\boldsymbol{V} = \boldsymbol{Y}^\top \boldsymbol{U} \tilde{\boldsymbol{\Sigma}}^{-1} \sqrt{\frac{\rho_c \rho_r (1-\delta)}{np}}$$

However, we do not need to explicitly determine $\boldsymbol{V}$ here. Instead the low-rank $\boldsymbol{X}$ can be determined directly from $\boldsymbol{U}$ with the projection given below:

$$\boldsymbol{X} = \boldsymbol{U} \tilde{\boldsymbol{\Sigma}} \sqrt{\frac{np}{\rho_c \rho_r (1-\delta)}} \boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{Y}.$$

Approximate decoder 3

Similar to the approximate decoder 2, we can propose another approximate decoder which performs a graph upsampling on $\boldsymbol{V}$ and then determines $\boldsymbol{U}$ via matrix multiplication operation.

$$\min_{\boldsymbol{V}} \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V}) \quad \text{s.t:} \quad \boldsymbol{M}^{c\top} \boldsymbol{V} = \tilde{\boldsymbol{V}}$$

The solution for $\boldsymbol{V}$ is given by eq. 6.18. Using the similar trick as for the approximate decoder 2, we can compute $\boldsymbol{X}$ without computing $\boldsymbol{U}$. Therefore, $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{V}\boldsymbol{V}^\top$.

Complexity: For the proposed approximate decoders, we would need to do one SVD to determine the singular values $\tilde{(\boldsymbol{\Sigma})}$. However, note that this SVD is on the compressed matrix $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r \times \rho_c}$. Thus, it is inexpensive $\mathcal{O}(\rho_r^2 \rho_c)$ assuming that $\rho_r < \rho_c$. The cost of these decoders is the same as the approximate decoder, i.e., $\mathcal{O}(O_l k_{nn} kn)$ where O_l is the number of iterations for the PCG method.

6.6 Decoder for clustering

As already mentioned earlier, PCA has been widely used for two types of applications: 1) low-rank recovery and 2) clustering. Therefore, in this section, we present a method to perform clustering using our framework.

For the clustering application we do not need the full low-rank matrix $\boldsymbol{X}$. Thus, we propose to do k-means on the low-rank representation of the sampled data $\tilde{\boldsymbol{X}}$ obtained using (6.4), extract the cluster labels $\tilde{\boldsymbol{C}}$ and then decode the cluster labels $\boldsymbol{C}$ for $\boldsymbol{X}$ on the graphs $\boldsymbol{L}^r$ and $\boldsymbol{L}^c$.

Let $\tilde{\boldsymbol{C}} \in \{0,1\}^{\rho_c \times k}$ be the cluster labels of $\tilde{\boldsymbol{X}}$ (for k clusters) which are obtained by performing k-means. Then,

$$\tilde{\boldsymbol{C}}_{ij} = \begin{cases} 1 & \text{if } \tilde{\boldsymbol{x}}_i \in j^{th} \text{ cluster} \\ 0 & \text{otherwise.} \end{cases}$$

Note that each of the columns $\tilde{\boldsymbol{C}(i)}$ of $\tilde{\boldsymbol{C}}$ is the cluster indicator for one of the k clusters. The goal now is to decode the cluster indicator matrix $\boldsymbol{C} \in \{0,1\}^{n\times k}$. We refer to the Compressive Spectral Clustering (CSC) framework [111], where the authors solve a similar problem by arguing that each of the columns of $\boldsymbol{C}$ can be obtained by assuming that it lies close to the $span(\boldsymbol{P}^c_{k_c})$, where $\boldsymbol{P}^c_{k_c}$ are the first k_c Laplacian eigenvectors of the graph $\mathbb{G}^c$. This requires solving the following convex minimization problem:

$$\min_{\boldsymbol{C}} \|\boldsymbol{M}^{c\top}\boldsymbol{C} - \tilde{\boldsymbol{C}}\|_F^2 + \gamma \operatorname{tr}(\boldsymbol{C}^\top \boldsymbol{L}^c \boldsymbol{C}) \tag{6.19}$$

The above problem can be solved independently for each of the columns of $\boldsymbol{C}$, thus,

$$\min_{\boldsymbol{C}(i)} \|\boldsymbol{M}^{c\top}\boldsymbol{C}(i) - \tilde{\boldsymbol{C}(i)}\|_2^2 + \gamma \boldsymbol{C}(i)^\top \boldsymbol{L}^c \boldsymbol{C}(i) \tag{6.20}$$

Furthermore, note that the graph $\mathbb{G}^r$ is not required for this process. Eq.(6.20) gives a faithful solution for $\boldsymbol{C}(i)$ if the sampling operator $\boldsymbol{M}^c$ satisfies the restricted isometry property RIP. Thus, for any $\delta_c, \epsilon_c \in (0,1)$, with probability at least $1-\epsilon_c$,

$$(1-\delta_c)\|\boldsymbol{w}\|_2^2 \le \frac{n}{\rho_c}\|\boldsymbol{w}^\top \boldsymbol{M}^c\|_2^2 \le (1+\delta_c)\|\boldsymbol{w}\|_2^2 \tag{6.21}$$

for all $\boldsymbol{w} \in span(\boldsymbol{P}^c_{k_c})$ provided that

$$\rho_c \ge \frac{3}{\delta_c^2}\nu_{k_c}^2 \log\left(\frac{2k_c}{\epsilon_c}\right). \tag{6.22}$$

This holds true as a consequence of Theorem 6.3.1 (eq.(A.7) in the proof of Theorem 6.3.1 and Theorem 5 in [100]).

Eq.(6.20) requires the tuning of a model parameter γ which we want to avoid. Therefore, we use the same strategy as for the approximate low-rank decoder in Section 6.5.3. The cluster labels $\tilde{\boldsymbol{C}}(i)$ are not noisy because they are obtained by running k-means on the result of FRPCAG eq.(6.4), which is robust to outliers. Thus, we set $\gamma = 0$ in eq.(6.20) and propose to solve the following problem:

$$\min_{\boldsymbol{C}(i)} \boldsymbol{C}(i)^\top \boldsymbol{L}^c \boldsymbol{C}(i) \quad \text{s.t:} \quad \boldsymbol{M}^{c\top}\boldsymbol{C}(i) = \tilde{\boldsymbol{C}}(i). \tag{6.23}$$

According to Lemma 6.5.1, the solution is given by:

$$\boldsymbol{C}(i) = \begin{bmatrix} -\boldsymbol{L}^{c-1}(\bar{\Omega}_c, \bar{\Omega}_c)\boldsymbol{L}^c(\bar{\Omega}_c, \Omega_c)\tilde{\boldsymbol{C}(i)} \\ \tilde{\boldsymbol{C}(i)} \end{bmatrix}. \tag{6.24}$$

Ideally, every row of the matrix $\boldsymbol{C}$ should have 1 in exactly one of the k columns, indicating the cluster membership of that data sample. However, the solution $\boldsymbol{C} \in \mathbb{R}^{n\times k}$ obtained by solving the above problem is not binary. Thus, to finalize the cluster membership (one of the k columns), we perform a maximum pooling for each of the rows of $\boldsymbol{C}$, i.e,

$$\boldsymbol{C}_{ij} \leftarrow \begin{cases} 1 & \text{if } \boldsymbol{C}_{ij} = \max\{\boldsymbol{C}_{ij}\ \forall\ j = 1\cdots k\} \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 3 summarizes this procedure. The complete CPCA algorithm is stated in Table 6.1.

Algorithm 3 Approximate Decoder for clustering

```
INPUT: $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_r \times \rho_c}$, $\boldsymbol{L}^c \in \mathbb{R}^{n\times n}$
1. do k-means on $\tilde{\boldsymbol{X}}$ to get the labels $\tilde{\boldsymbol{C}} \in \{0,1\}^{\rho_c \times k}$
2. Solve eqs. (6.23) for every column of $\boldsymbol{C}$ as following:
for $i = 1, \dots k$ do
    solve $\min_{\boldsymbol{C}(i)} \boldsymbol{C}(i)^\top \boldsymbol{L}^c \boldsymbol{C}(i)$    s.t $\boldsymbol{M}^{c\top} \boldsymbol{C}(i) = \tilde{\boldsymbol{C}(i)}$ using PCG
end for
3. Set $\boldsymbol{C}_{ij} = 1$ if $\max\{\boldsymbol{C}_{ij}\ \forall\ j = 1\cdots k\}$ and 0 otherwise.
OUTPUT: cluster indicators for $\boldsymbol{X}$: $\boldsymbol{C} \in \{0,1\}^{n\times k}$
```

Theorem 6.6.1. *Let* $\boldsymbol{M}^c$ *be such that eq.* (6.21) *holds. Let also* $\boldsymbol{C}(i)^*$ *be the solution of* (6.23) *with* $\tilde{\boldsymbol{C}(i)} = \boldsymbol{M}^{c\top}\bar{\boldsymbol{C}(i)}$, *where* $\bar{\boldsymbol{C}(i)} \in span(\boldsymbol{P}^c_{k_c})$, $i = 1, \cdots, n$. *We have:*

$$\|\boldsymbol{C}(i)^* - \bar{\boldsymbol{C}(i)}\|_2 \leq \sqrt{\frac{n}{\rho_c(1-\delta_c)}}\sqrt{\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}}\|\bar{\boldsymbol{C}(i)}\|_2$$

where $\boldsymbol{C}(i)^* = \boldsymbol{P}^c_{k_c}\boldsymbol{P}^{c\ \top}_{k_c}\boldsymbol{C}(i)$.

Proof. The proof directly follows from the proof of Theorem 3.2 in [100]. These steps have been repeated in the proof of Theorem 6.5.3 in Appendix A.7 as well. Using $\boldsymbol{C}(i)^* = \bar{\boldsymbol{U}(i)}^*$, $\bar{\boldsymbol{C}(i)} = \bar{\boldsymbol{U}(i)}$, $n = p, \rho_c = \rho_r$ in eq.(A.18) one can get theoretical guarantees for eq. (6.20). Then, by using $\tilde{\boldsymbol{E}(i)}^u = 0$ and $\gamma = 0$ we get the result of above theorem. □

Table 6.1 – Summary of CPCA and its computational complexity for a dataset $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$. Throughout we assume that $k_{nn}, r, k, \rho_r, \rho_c, p \ll n$.

The Complete CPCA Algorithm	**Complexity**
1. Construct graph Laplacians between the rows $\boldsymbol{L}^r$ and columns $\boldsymbol{L}^c$ of $\boldsymbol{Y}$ using FLANN.	$\mathcal{O}(np\log(n))$
2. Construct row and column sampling matrices $\boldsymbol{M}^r \in \mathbb{R}^{\rho_r\times p}$ and $\boldsymbol{M}^c \in \mathbb{R}^{n\times \rho_c}$ satisfying (6.1) and theorem 6.3.1	–
3. Sample the data matrix $\boldsymbol{Y}$ as $\tilde{\boldsymbol{Y}} = \boldsymbol{M}^r\boldsymbol{Y}\boldsymbol{M}^c$	–
4. Construct the new graph Laplacians between the rows $\tilde{\boldsymbol{L}}^r$ and columns $\tilde{\boldsymbol{L}}^c$ of $\tilde{\boldsymbol{Y}}$ using Section 6.4.1.	$\mathcal{O}(O_l k_{nn} n)$
5. Solve FRPCAG (6.4) using Algorithm 1 to get the low-rank $\tilde{\boldsymbol{X}}$	$\mathcal{O}(I\rho_r\rho_c k_{nn})$
6. **For low-rank recovery**: Decode $\boldsymbol{X}$ from $\tilde{\boldsymbol{X}}$ using the approximate decoder Algorithm 2	$\mathcal{O}(O_l n r k_{nn})$
7. **For clustering**: Decode the cluster labels $\boldsymbol{C}$ for $\boldsymbol{X}$ using Algorithm 3	$\mathcal{O}(O_l n k)$

6.7 Complexity, Memory & Convergence

Computational Complexity: Now that we have presented our proposed matrix based low-rank recovery methods, we are ready to comment on the computational complexities of ours and other methods in detail. First, we present a set of rules which we follow for our calculations.

For a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$, let I denote the number of iterations for the algorithms to converge, p is the number of features, n is the number of samples, ρ_r, ρ_c are the number of features and samples for the compressed data $\tilde{\boldsymbol{Y}}$ and satisfy eq. (6.1) and theorem 6.3.1, r is the rank of the low-dimensional space, k is the number of clusters, k_{nn} is the number of nearest neighbors for graph construction, O_l, O_c correspond to the number of iterations in the Lanczos and Chebyshev approximation methods. Furthermore,

1. We assume that $k_{nn}, k, k_r, k_c, r, \rho_r, \rho_c << n$.
2. The complexity of $\phi(\boldsymbol{Y} - \boldsymbol{X})$ is $\mathcal{O}(np)$ per iteration and that of $\phi(\tilde{\boldsymbol{Y}} - \tilde{\boldsymbol{X}})$ is $\mathcal{O}(\rho_c \rho_r)$.
3. The complexity of the computations corresponding to the graph regularization $\text{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \text{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X}) = \mathcal{O}(p|\mathbb{E}^c| + n|\mathbb{E}^r|) = \mathcal{O}(pnk_{nn} + npk_{nn})$, where $\mathbb{E}_r, \mathbb{E}_c$ denote the number of non-zero edges in $\boldsymbol{L}^r, \boldsymbol{L}^c$. Note that we use the k_{nn}-nearest neighbors graphs so $\mathbb{E}^r \approx k_{nn}p$ and $\mathbb{E}^c \approx k_{nn}n$.
4. The complexity for the construction of $\tilde{\boldsymbol{L}}^c$ and $\tilde{\boldsymbol{L}}^r$ for compressed data $\tilde{\boldsymbol{Y}}$ is negligible if FLANN is used, i.e, $\mathcal{O}(\rho_c\rho_r \log(\rho_c))$ and $\mathcal{O}(\rho_c\rho_r \log(\rho_r))$. However, if the kron reduction strategy of Section 6.4.1 is used then the cost is $\mathcal{O}(k_{nn}O_l(n+p)) \approx \mathcal{O}(k_{nn}O_l n)$.
5. We use the complexity $\mathcal{O}(np^2)$ for all the SVD computations on the matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ and $\mathcal{O}(\rho_c \rho_r^2)$ for $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\rho_c \times \rho_r}$.
6. The complexity of $\phi(\boldsymbol{M}^r\boldsymbol{X}\boldsymbol{M}^c - \tilde{\boldsymbol{X}})$ is negligible as compared to the graph regularization terms $\text{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top) + \text{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$.
7. We use the approximate decoders for low-rank recovery in the complexity calculations (eq. (6.18) in Section 6.5.3). All the decoders for low-rank recovery are summarized in Table 6.2.
8. The complexity of k-means [112] is $\mathcal{O}(Inkp)$ for a matrix $\boldsymbol{X} \in \Re^{p \times n}$ and $\mathcal{O}(I\rho_r\rho_c k)$ for a matrix $\tilde{\boldsymbol{X}} \in \Re^{\rho_r \times \rho_c}$.

A summary of all the decoders and their computational complexities is presented in Table 6.2. The complete CPCA algorithm and the computational complexities of different steps are presented in Table 6.1. All the models which use the graph $\mathbb{G}^c$ are marked by '+'. The construction of graph $\mathbb{G}^r$ is included only in FRPCAG and CPCA. For $k_{nn}, r, k, \rho_r, \rho_c \ll n$ CPCA algorithm scales as $O(nkk_{nn})$ per iteration. Thus, assuming that the row and column graphs are available from external source, a speed-up of p/r per iteration is obtained over FRPCAG and p^2/r over RPCA.

Memory Requirements: We compare the memory requirements of CPCA with FRPCAG. For a matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, FRPCAG and CPCA require the construction of two graphs $\mathbb{G}^r, \mathbb{G}^c$ whose Laplacians $\boldsymbol{L}^r \in \mathbb{R}^{p \times p}, \boldsymbol{L}^c \in \mathbb{R}^{n \times n}$ are used in the core algorithm. However, these Laplacians are sparse, therefore the memory requirement for $\boldsymbol{L}^r, \boldsymbol{L}^c$ is $\mathcal{O}(k_{nn}(|\mathbb{E}_r| + |\mathbb{E}_c|))$ respectively. The core algorithm of FRPCAG requires operation on the full matrix $\boldsymbol{Y}$ and the graph Laplacians $\boldsymbol{L}^r, \boldsymbol{L}^c$. As $|\mathbb{E}_r| \approx k_{nn}p$ and $|\mathbb{E}_c| \approx k_{nn}n$ therefore, the memory requirement for the regularization terms $\text{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top)$ and $\text{tr}(\boldsymbol{X}^\top\boldsymbol{L}^r\boldsymbol{X})$ is $\mathcal{O}(k_{nn}np)$. For the CPCA algorithm, assuming $n > p$ and letting $\rho_r = p/b$ and $\rho_c = n/a$, the complexity of FRPCAG on the sampled data is $\mathcal{O}(k_{nn}np/(ab))$ and the approximate decode stage for subspaces of dimension r is $\mathcal{O}(k_{nn}nr)$. Thus the overall memory requirement of CPCA is $\mathcal{O}(k_{nn}n(p/(ab) + r))$. As compared to FRPCAG, an improvement of $pab/(p + kab)$ is obtained. For example for $n = 1000, p = 200, a = 10, b = 1, r = 10$, a reduction of approximately 6.6 times is obtained.

Table 6.2 – A summary and computational complexities of all the decoders proposed in this work. The Lancoz method used here is presented in [7].

Type	Low-rank			
	model	**complexity**	**Algorithm**	**parallel implementation**
ideal	$\min_{\boldsymbol{X}} \lVert \boldsymbol{M}^r \boldsymbol{X} \boldsymbol{M}^c - \tilde{\boldsymbol{X}} \rVert_F^2$ s.t: $\boldsymbol{X}(i)^\top \in span(\boldsymbol{P}^c_{k_c})$ $\boldsymbol{X}(i) \in span(\boldsymbol{P}^r_{k_r})$	$\mathcal{O}(n^3)$	–	–
alternate	$\min_{\boldsymbol{X}} \lVert \boldsymbol{M}^r \boldsymbol{X} \boldsymbol{M}^c - \tilde{\boldsymbol{X}} \rVert_F^2$ $+\gamma_c \operatorname{tr}(\boldsymbol{X} \boldsymbol{L}^c \boldsymbol{X}^\top)$ $+\gamma_r \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X})$	$\mathcal{O}(Inpk_{nn})$	gradient descent	no
approximate	$\min_{\boldsymbol{U}} \lVert \boldsymbol{M}^r \boldsymbol{U} - \tilde{\boldsymbol{U}} \rVert_F^2$ $+\gamma'_r \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U})$ $\min_{\boldsymbol{V}} \lVert \boldsymbol{M}^{c\top} \boldsymbol{V} - \tilde{\boldsymbol{V}} \rVert_F^2$ $+\gamma'_c \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V})$ $\boldsymbol{X} = \boldsymbol{U} \tilde{\boldsymbol{\Sigma}} \boldsymbol{V}^\top$	$\mathcal{O}(Ipk_{nn})$ $\mathcal{O}(Ink_{nn})$ $\mathcal{O}(\rho_r^2 \rho_c)$	gradient descent gradient descent SVD	yes yes
Subspace Upsampling	$\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U})$ s.t:$\boldsymbol{M}^r \boldsymbol{U} = \tilde{\boldsymbol{U}}$ $\min_{\boldsymbol{V}} \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V})$ s.t:$\boldsymbol{M}^{c\top} \boldsymbol{V} = \tilde{\boldsymbol{V}}$ $\boldsymbol{X} = \boldsymbol{U} \tilde{\boldsymbol{\Sigma}} \boldsymbol{V}^\top$	$\mathcal{O}(pkO_l k_{nn})$ $\mathcal{O}(nkO_l k_{nn})$ $\mathcal{O}(\rho_r^2 \rho_c)$	PCG PCG SVD	yes
approximate 2	$\min_{\boldsymbol{U}} \operatorname{tr}(\boldsymbol{U}^\top \boldsymbol{L}^r \boldsymbol{U})$ s.t:$\boldsymbol{M}^r \boldsymbol{U} = \tilde{\boldsymbol{U}}$ $\boldsymbol{X} = \boldsymbol{U} \boldsymbol{U}^\top \boldsymbol{Y}$	$\mathcal{O}(pkO_l k_{nn})$	PCG	yes
approximate 3	$\min_{\boldsymbol{V}} \operatorname{tr}(\boldsymbol{V}^\top \boldsymbol{L}^c \boldsymbol{V})$ s.t:$\boldsymbol{M}_c^\top \boldsymbol{V} = \tilde{\boldsymbol{V}}$ $\boldsymbol{X} = \boldsymbol{Y} \boldsymbol{V} \boldsymbol{V}^\top$	$\mathcal{O}(nkO_l k_{nn})$	PCG	yes

Table 6.3 – Computational complexity of all the models considered in this work

Model	**Complexity** $\mathbb{G}^c$ $\mathcal{O}(np\log(n))$	**Complexity** $\mathbb{G}^r$ $\mathcal{O}(np\log(p))$	**Complexity Algorithm** for $p \ll n$	**Overall Complexity (low-rank)**			**Overall Complexity (clustering)**		
				Fast SVD for $p \ll n$	**Decoder**	**Total**	**k-means** [112]	**Decoder**	**Total**
LE [2]	+	–	$\mathcal{O}(nr^2)$	–	–	–	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(n^3)$
LLE [24]	–	–	$\mathcal{O}((p+r)n^2)$	–	–	–	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(pn^2)$
PCA	–	–	$\mathcal{O}(nr^2)$	–	–	$\mathcal{O}(n(p^2\log(n)+p^2))$	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(np^2\log(n)+np^2)$
GLPCA [3]	+	–	$\mathcal{O}(n^3)$	–	–	$\mathcal{O}(n(p\log(n)+n^2))$	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(n(p\log(n)+n^2))$
NMF [4]	–	–	$\mathcal{O}(Inpr)$	–	–	$\mathcal{O}(Inpk)$	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(Inp(r+k))$
GNMF [5]	+	–	$\mathcal{O}(Inpr)$	–	–	$\mathcal{O}(np\log(n))$	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(np\log(n))$
MMF [6]	+	–	$\mathcal{O}(((p+r)r^2+pr)I)$	–	–	$\mathcal{O}(np\log(n))$	$\mathcal{O}(Inkr)$	–	$\mathcal{O}(np\log(n))$
RPCA [1]	–	–	$\mathcal{O}(Inp^2)$	–	–	$\mathcal{O}(np^2I+n\log(n))$	$\mathcal{O}(Inpk)$	–	$\mathcal{O}(np^2I+n\log(n))$
RPCAG [99]	+	–	$\mathcal{O}(Inp^2)$	–	–	$\mathcal{O}(np^2I+n\log(n))$	$\mathcal{O}(Inpk)$	–	$\mathcal{O}(np^2I+n\log(n))$
FRPCAG [110]	+	+	$\mathcal{O}(Inpk_{nn})$	–	–	$\mathcal{O}(np\log(n))$	$\mathcal{O}(Inpk)$	–	$\mathcal{O}(np\log(n))$
CPCA	+	+	$\mathcal{O}(I\rho_c\rho_r k_{nn})$	$\mathcal{O}(\rho_c{\rho^2}_r)$	$\mathcal{O}(nkO_l k_{nn})$	$\mathcal{O}(np\log(n))$	$\mathcal{O}(I\rho_c k\rho_r)$	$\mathcal{O}(nkO_l)$	$\mathcal{O}(np\log(n))$

Convergence of CPCA: The CPCA based algorithm (Table 6.1) has two main steps: 1) FRPCAG on the compressed data matrix and 2) low-rank matrix or cluster label decoding. FRPCAG is solved by the FISTA (Algorithm 1) and the decode step is solved using the PCG method. Both of these methods have been well studied in terms of their convergence guarantees. More specifically, one can refer to [103] for a detailed study on FISTA and [113] for PCG. Let $\tilde{\boldsymbol{X}}^j$ represent the iterates of the FRPCAG problem 6.4, $\tilde{X}^*$ the optimal solution, $F(\tilde{\boldsymbol{X}})$ the objective function and $2\gamma_r\|\tilde{\boldsymbol{L}}^r\|_2 + 2\gamma_c\|\tilde{\boldsymbol{L}}^c\|_2$ the Lipschitz constant, then the Algorithm 1 (FISTA) converges as following:

$$F(\tilde{\boldsymbol{X}}^j) - F(\tilde{\boldsymbol{X}}^*) \leq \frac{4\alpha(\gamma_r\|\tilde{\boldsymbol{L}}^r\|_2 + \gamma_c\|\tilde{\boldsymbol{L}}^c\|_2)\|\tilde{\boldsymbol{X}}^j - \tilde{\boldsymbol{X}}^*\|_F^2}{(j+1)^2},$$

where α is the step size and j denotes the iteration number. The PCG method on the other hand can theoretically be viewed as a direct method, as it produces the exact solution after a finite number of iterations [113].

Tables 6.9 & 6.12 show detailed computational time and iterations for our CPCA based algorithms to converge, for a wide variety and sizes of datasets.

6.8 Experimental Results

Similarly to FRPCAG (Chapter 5), we perform two types of experiments corresponding to two applications of PCA 1) Data clustering and 2) Low-rank recovery using two open-source toolboxes: the UNLocBoX [107] and the GSPBox [62].

6.8.1 Clustering

Experimental Setup

Datasets: We perform our clustering experiments on 5 benchmark databases (as in [99, 110]): CMU PIE, ORL, YALE, MNIST and USPS. For the USPS and ORL datasets, we further run two types of experiments 1) on subset of datasets and 2) on full datasets. The experiments on the subsets of the datasets take less time so they are used to show the efficiency of our model for a wide variety of noise types. The details of all datasets used are provided in Table 6.4.

Table 6.4 – Details of the datasets used for clustering experiments in this work.

Dataset	Samples	Dimension	Classes
ORL large	400	56 × 46	40
ORL small	300	28 × 23	30
CMU PIE	1200	32 × 32	30
YALE	165	32 × 32	11
MNIST	70000	28 × 28	10
MNIST small	1000	28 × 28	10
USPS large	10000	16 × 16	10
USPS small	3500	16 × 16	10

Noise & Errors: CPCA is a memory and computationally efficient alternative for FRPCAG. An important property of FRPCAG is its robustness to noise and outliers, just like RPCA. Therefore, it is important to study the performance of CPCA under noise and corruptions similar to those for FRPCAG and RPCA.

To do so we add 3 different types of noise in all the samples of datasets in different experiments: 1) Gaussian noise and 2) Laplacian noise with standard deviation ranging from 5% to 20% of the original data 3) Sparse noise (randomly corrupted pixels) occupying 5% to 20% of each data sample. Note that this is different from the experiments carried out for FRPCAG. For the experiments with Gaussian noise we use $\phi = \ell_2$ and for those with sparse we use $\phi = \ell_1$ norm.

Similarly to FRPCAG, we compare with many different state-of-the-art methods, using similar validation procedure and parameters for graph construction, pre-process all the datasets in a similar manner and use clustering error as an evaluation metric. Although our CPCA approach is convex, it involves a bit of randomness due to the sampling step. Due to the extensive nature of the experimental setup, most of the clustering experiments are performed under one sampling condition. However, it is interesting to study the variation of error under different sampling scenarios. For this purpose we perform some additional experiments on the USPS dataset. For our proposed CPCA method, we use a convention $CPCA(a, b)$, where a and b denote the downsampling factors on the columns and rows respectively. A uniform sampling strategy is always used for CPCA. The clustering error for RPCA, RPCAG and FRPCAG is determined by performing k-means directly on the low-rank $\boldsymbol{X}$. For our CPCA method, k-means is performed on the small low-rank $\tilde{\boldsymbol{X}}$ and then the labels for full $\boldsymbol{X}$ are decoded using Algorithm 3.

Discussion on clustering performance

We point out here that the purpose of our clustering experiments is three-fold:

- To show the efficiency of CPCA for a wide variety of noise and errors and downsampling.
- To study the conditions under which CPCA performs worse than the other models.
- To study the variation of performance under different sampling scenarios.

For this purpose, we test CPCA under a variety of downsampling rates for different datasets. Cases with $p \ll n$ and $n \ll p$ carry special interest. Therefore, we present our discussion below in the light of the above goals.

Tables 6.5, 6.6, 6.7 & 6.8 present the clustering results for USPS small, USPS large, MNIST large, MNIST small, ORL small, ORL large, CMU PIE and YALE datasets. Note that not all the models are run for all the datasets due to computational constraints. The best results are highlighted in bold and the second best in blue. From Table 6.5 for the USPS dataset, it is clear that our proposed CPCA model attains comparable clustering results to the state-of-the-art RPCAG and FRPCAG models and better than the others in most of the cases. Similar observation can be made about the MNIST large dataset from Table 6.8 in comparison to FRPCAG.

It is important to note that for the USPS and MNIST datasets $p \ll n$. Thus, for the USPS dataset, the compression is only applied along the columns (n) of the dataset. This compression results in clustering error which is comparable to the other state-of-the-art algorithms. As $p = 256$ for the USPS dataset, it was observed that even a 2 times downsampling on p results in a loss of information and the clustering quality deteriorates. The same observation can be made about ORL small, ORL large and YALE datasets from Tables 6.6, 6.7 for CPCA (2,2). i.e, two times downsampling on both rows and columns. On the other hand the performance of CPCA (1,2) is reasonable for the ORL small dataset. Recall that CPCA (a,b) means a downsampling by a and b across columns and rows (samples and features). Also note that for ORL dataset $n \ll p$.

Table 6.5 – Clustering error of USPS datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue. NMF and GNMF require non-negative data so they are not evaluated for USPS because the features of the USPS dataset are also negative.

Dataset	**Model**	**no noise**	**Gaussian noise**				**Laplacian noise**				**Sparse noise**			
			5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
USPS small ($n = 3500$ $p = 256$)	k-means	0.31	0.31	0.31	0.33	0.32	0.32	0.30	0.36	0.37	0.40	0.45	0.53	0.73
	LLE	0.40	0.34	0.32	0.35	0.24	0.40	0.40	0.33	0.36	0.23	0.30	0.33	0.37
	LE	0.38	0.38	0.38	0.36	0.35	0.38	0.38	0.38	0.38	0.32	0.33	0.36	0.48
	PCA	0.27	0.29	0.25	0.28	0.26	0.29	0.29	0.28	0.24	0.29	0.26	0.26	0.28
	MMF	0.21	**0.20**	0.21	0.22	0.21	0.21	0.21	0.22	0.21	0.27	0.23	0.25	0.27
	GLPCA	**0.20**	**0.20**	0.21	0.23	0.23	0.21	0.21	0.22	0.21	0.26	0.24	0.24	0.28
	RPCA	0.26	0.25	0.23	0.24	0.22	0.26	0.26	0.25	0.24	0.26	0.24	0.23	0.30
	RPCAG	**0.20**	**0.20**	0.21	0.20	0.21	**0.20**	0.21	0.21	0.21	**0.21**	**0.22**	0.23	0.25
	FRPCAG	**0.20**	**0.20**	**0.20**	**0.19**	**0.20**	**0.20**	**0.19**	**0.17**	**0.17**	**0.21**	**0.22**	**0.22**	**0.23**
	CPCA (2,1)	**0.20**	**0.20**	0.21	0.20	0.22	**0.20**	0.22	0.22	0.21	0.23	0.23	0.25	0.28
USPS large ($n = 10000$ $p = 256$)	k-means	0.26	0.26	0.26	0.26	0.28	0.27	0.26	0.26	0.26	0.26	0.25	0.34	0.30
	LLE	0.51	0.29	0.22	0.21	0.22	0.22	0.22	0.22	0.21	0.22	**0.19**	0.26	0.31
	LE	0.33	0.32	0.32	0.27	0.27	0.32	0.34	0.31	0.34	0.35	0.44	0.49	0.53
	PCA	0.21	0.21	0.21	0.22	0.21	0.21	0.21	0.22	0.21	0.22	0.23	0.23	0.23
	MMF	0.24	0.23	0.23	0.24	0.24	0.19	0.23	0.22	0.23	0.24	0.25	0.26	0.26
	GLPCA	0.16	0.16	0.17	0.17	0.16	0.17	0.15	0.15	0.17	**0.18**	**0.19**	**0.21**	0.23
	FRPCAG	**0.15**	0.16	0.17	**0.15**	**0.14**	0.16	0.16	0.16	0.17	**0.18**	0.21	**0.21**	**0.21**
	CPCA (10,1)	**0.15**	**0.14**	**0.14**	**0.15**	**0.14**	**0.14**	**0.14**	**0.13**	**0.14**	**0.18**	**0.19**	0.22	0.24
MNIST small ($n = 1000$ $p = 784$)	K-means	0.51	0.51	0.52	0.51	0.58	0.52	0.51	0.52	0.52	0.80	0.88	0.88	0.88
	PCA	0.43	0.43	0.41	0.43	0.42	0.43	0.42	0.38	0.43	0.42	0.42	0.42	0.44
	MMF	0.33	0.34	0.34	0.34	**0.33**	0.34	**0.33**	0.37	0.34	0.40	0.38	0.38	0.42
	GLPCA	0.38	0.36	0.37	0.35	0.38	0.39	0.36	0.36	0.36	0.37	0.39	0.38	**0.39**
	PCAG-(1,0)	0.40	0.40	0.41	0.40	0.40	0.40	0.40	0.41	0.40	0.37	0.37	0.39	0.44
	FRPCAG	**0.32**	**0.33**	**0.32**	**0.33**	**0.33**	**0.32**	**0.33**	**0.32**	**0.32**	**0.33**	**0.36**	**0.35**	**0.39**
	CPCA (5,1)	0.39	0.38	0.37	0.38	0.38	0.39	0.39	0.36	0.37	0.39	0.40	0.41	0.50

Table 6.6 – Clustering error of ORL datasets for different models. The best results per column are highlighted in bold and the 2nd best in blue.

Data set	Model	no noise	Gaussian noise			
			5%	10%	15%	20%
	k-means	0.40	0.43	0.43	0.45	0.44
	LLE	0.26	0.26	0.26	0.26	0.19
O	LE	0.21	0.18	0.19	0.19	0.19
R	PCA	0.30	0.30	0.32	0.34	0.32
L	MMF	0.21	0.20	0.18	0.18	0.17
R	GLPCA	**0.14**	**0.13**	**0.13**	**0.13**	**0.14**
s	NMF	0.31	0.34	0.29	0.31	0.34
m	GNMF	0.29	0.29	0.29	0.31	0.29
a	RPCA	0.36	0.34	0.33	0.35	0.36
l	RPCAG	0.17	0.17	0.17	0.16	0.16
l	FRPCAG	0.15	0.15	0.15	0.14	0.16
	CPCA (2,2)	0.23	0.23	0.23	0.24	0.25
	CPCA (1,2)	0.17	0.17	0.17	0.14	0.17
O	k-means	0.49	0.50	0.51	0.51	0.51
R	LLE	0.28	0.27	0.27	0.24	0.25
L	LE	0.24	0.25	0.25	0.24	0.25
	PCA	0.35	0.34	0.35	0.36	0.36
l	MMF	0.23	0.23	0.23	0.24	0.24
a	GLPCA	0.18	0.18	0.18	0.19	0.19
r	NMF	0.36	0.33	0.36	0.32	0.36
g	GNMF	0.34	0.37	0.36	0.39	0.39
e	FRPCAG	**0.17**	**0.17**	**0.17**	**0.17**	**0.17**
	CPCA (2,2)	0.21	0.21	0.21	0.23	0.22

Finally, we comment about the results on MNIST small dataset ($p = 784, n = 1000$) from Table 6.5. It is clear that FRPCAG (no compression) results in the best performance. CPCA (5,1) results in a highly undersampled dataset which does not capture enough variations in the MNIST small dataset to deliver a good clustering performance. This particular case supports the fact that compression does not always yield a good performance at the advantage of reduced complexity. Therefore, we study this phenomena below.

The above findings for the MNIST dataset are intuitive as it only makes sense to compress both rows and columns in our CPCA based framework if a reasonable speed-up can be obtained without compromising the performance, i.e, *if both n and p are large.* If either n or p is small then one might only apply compression along the larger dimension, as compression on the smaller dimension would not speed up the computations significantly. For example, for the USPS dataset, a speed-up of $p/k = 256/10 \approx 25$ times would be obtained over FRPCAG by compressing along the samples (columns n) only without a loss of clustering quality. Our experiments showed that this speed up increased upto 30 by compressing along the features but with a loss of performance (The results are not presented for brevity).

Tables 6.5, 6.6, 6.7 & 6.8 also show that CPCA is quite robust to a variety of noise and errors in the dataset. Even in the presence of higher levels of Gaussian and Laplacian noise, CPCA performs comparable to other methods for the USPS dataset. Thus, CPCA tends to preserve the robustness property of FRPCAG. This will also be clear from the low-rank recovery experiments in the next section.

Computation Time vs Performance & Sampling

It is interesting to compare 1) the time needed for FRPCAG and CPCA to perform clustering 2) the corresponding clustering error and 3) the sub-sampling rates in CPCA. Table 6.8 shows such a comparison

Table 6.7 – Clustering error of CMU PIE and YALE datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue.

Data set	Model	no noise	Gaussian noise			
			5%	**10%**	**15%**	**20%**
CMU PIE	k-means	0.76	0.76	0.76	0.75	0.76
	LLE	0.47	0.50	0.52	0.50	0.55
	LE	0.60	0.58	0.59	0.58	0.60
	PCA	0.27	0.27	0.27	0.27	0.29
	MMF	0.67	0.67	0.67	0.66	0.67
	GLPCA	0.37	0.39	0.37	0.38	0.38
	NMF	0.24	0.27	**0.24**	0.25	0.27
	GNMF	0.58	0.59	0.56	0.58	0.59
	RPCA	0.39	0.38	0.41	0.41	0.38
	RPCAG	0.24	**0.24**	**0.24**	**0.24**	0.25
	FRPCAG	**0.23**	**0.24**	**0.24**	**0.24**	**0.24**
	CPCA (2,1)	0.26	0.26	0.26	0.28	0.27
	CPCA (2,2)	0.28	0.29	0.29	0.30	0.30
YALE	k-means	0.76	0.76	0.76	0.76	0.77
	LLE	0.51	0.47	0.46	0.49	0.50
	LE	0.52	0.56	0.55	0.54	0.54
	PCA	0.53	0.52	0.53	0.56	0.55
	MMF	0.58	0.59	0.56	0.57	0.58
	GLPCA	0.47	0.46	0.47	0.45	0.48
	NMF	0.57	0.58	0.59	0.57	0.59
	GNMF	0.59	0.59	0.61	0.59	0.60
	RPCA	0.45	0.48	0.46	0.46	0.48
	RPCAG	**0.40**	0.40	0.41	0.41	0.41
	FRPCAG	**0.40**	**0.37**	**0.40**	**0.40**	**0.40**
	CPCA (2,2)	0.43	0.45	0.42	0.46	0.43

for 70,000 digits of MNIST with (10, 2) times downsampling on the (columns, rows) respectively for CPCA. The time needed by CPCA is an order of magnitude lower than FRPCAG. Note that the time reported here does not include the construction of graphs $\mathbb{G}^r, \mathbb{G}^c$ as both methods use the same graphs. Furthermore, these graphs can be constructed in the order of a few seconds if parallel processing is used. The time for CPCA includes steps 2 to 5 and 7 of Table 6.1. For the information about the graph construction time, please refer to Table 6.9 and the discussion thereof.

Table 6.9 presents the computational time and number of iterations for the convergence of CPCA, FRPCAG, RPCAG & RPCA on different sizes and dimensions of the datasets. We also present the time needed for the graph construction. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. An ∞ in the table indicates that the algorithm did not converge in 4 hours. It is notable that our model requires a very small number of iterations to converge irrespective of the size of the dataset. Furthermore, the model is orders of magnitude faster than RPCA and RPCAG. This is clearly observed from the experiments on MNIST dataset where our proposed model is 100 times faster than RPCAG. Specially for MNIST dataset with 25000 samples, RPCAG and RPCA did not converge even in 4 hours whereas CPCA converged in less than a minute.

It is interesting to study the reduction in the total time attained by using CPCA as compared to FRPCAG. Table 6.9 can be used to perform this comparison as well. For example, for the MNIST dataset with 5000 samples, the total time (including graph construction) for FRPCAG is 28.2 secs and that for CPCA is 20.1 secs. Thus, a speed-up of 1.4 times is obtained over FRPCAG. The time required for the construction of the graph between the samples or features is often more than that required for the CPCA to converge. This is a small computational bottleneck of the graph construction algorithm. While graph learning or graph construction is an active and interesting field of research, it is not a part of this work. The

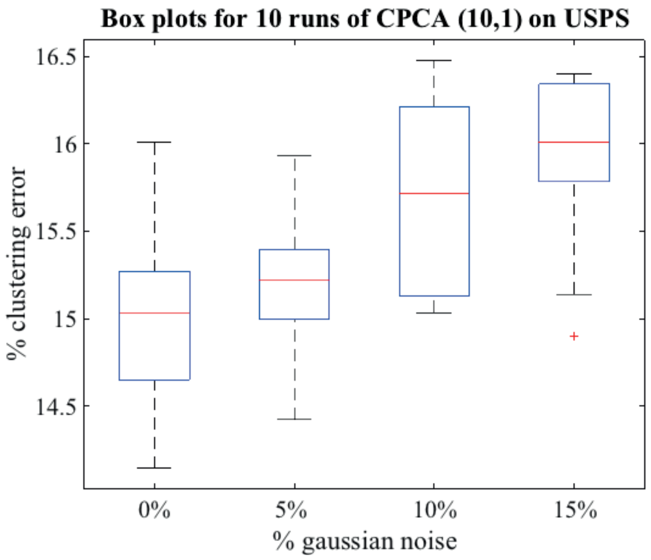


Figure 6.1 – Box plots for clustering error over 10 random sampling runs of CPCA (10,1) for the full USPS dataset (256 × 10000) with increasing levels of Gaussian noise. For each run CPCA is evaluated for the full parameter grid $\gamma'_r, \gamma'_c \in (0, 30)$ and the minimum clustering error is considered. A slight increase in the average clustering error with Gaussian noise shows that CPCA is quite robust to sampling and noise.

state-of-the-art results [92], [93], [114] do not provide any scalable solutions yet.

Table 6.8 – Clustering error and computational times of FRPCAG and CPCA on MNIST large dataset (784 × 70,000).

Model	**FRPCAG**	**CPCA (10,2)**
Error	0.25	0.24
time (secs)	350	58

Table 6.9 – Computation times (in seconds) for graphs G_r, G_c, FRPCAG, CPCA, RPCAG, RPCA and the number of iterations to converge for different datasets. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. An ∞ indicates that the algorithm did not converge in 4 hours.

Dataset	**Samples**	**Features**	**Classes**	**Graphs**		**FRPCAG**		**CPCA**			**RPCAG**		**RPCA**	
				G_r	G_c	**time**	**Iters**	**(a,b)**	**time**	**Iters**	**time**	**Iters**	**time**	**Iters**
MNIST	5000	784	10	10.8	4.3	13.7	27	(5,1)	5	30	1345	325	1090	378
MNIST	15000	784	10	32.5	13.3	35.4	23	(5,1)	13	25	3801	412	3400	323
MNIST	25000	784	10	40.7	22.2	58.6	24	(10,1)	20	37	∞	∞	∞	∞
ORL	300	10304	30	1.8	56.4	24.7	12	(2,1)	14	15	360	301	240	320
USPS	3500	256	10	5.8	10.8	21.7	16	(10,1)	12	31	900	410	790	350

Effect of random sampling

An interesting observation can be made from Table 6.8 for the MNIST dataset: the error of CPCA is also lower than FRPCAG. Such cases can also be observed in USPS dataset (Table 6.5). As the downsampling step is random, it might remove some spurious samples sometimes and the clustering scheme (Section 6.6) becomes robust to these samples. For the clustering application, the spurious samples mostly lie on the cluster borders and deteriorate the clustering decision.

To incorporate the extensive nature of the experiments presented in this work, Tables 6.5 & 6.8 correspond to one run of the CPCA for one specific sampling case. However, it is imperative to study

Table 6.10 – Preservation of the rank of the datasets in the compressed low-rank $\tilde{\boldsymbol{X}}$ determined by solving FRPCAG (6.4).

Dataset	**downsampling factor**	**actual rank**	**Rank after FRPCAG on sampled matrix**					
	$(columns, rows)$		**Gaussian noise**			**Laplacian noise**		
			5%	**10%**	**15%**	**5%**	**10%**	**15%**
ORL large	(2,1)	40	41	41	41	41	41	42
USPS large	(10,2)	10	10	11	11	11	11	11
MNIST	(10,2)	10	11	11	11	11	11	11
CMU PIE	(2,1)	30	31	31	31	31	31	32
YALE	(2,1)	11	11	11	11	11	11	11

the effect of random sampling on the performance of CPCA. Therefore, in order to study the effect of random sampling on the clustering performance, we perform an experiment on the full USPS dataset (256×10000) with different levels of artificial noise. The results in Fig. 6.1 correspond to 10 runs of the CPCA under different uniform sampling scenarios. For this experiment, we downsample 10 times along the columns (digits), whereas no downsampling is used across the features and then add different levels of Gaussian noise from 0 to 15% in the dataset. Thus, we downsample the dataset, run FRPCAG to get a low-rank $\tilde{X} \in \mathbb{R}^{256\times 1000}$, perform k-means ($k = 10$) and then use the approximate clustering decoder (Algorithm 3) to decode the labels for the full dataset. This process is repeated over the whole parameter grid $\gamma_r', \gamma_c' \in (0,30)$ and the minimum error over the grid is considered. Each of the boxplots in this figure summarizes the clustering error over 10 runs of CPCA(10,1) for different levels of Gaussian noise. The mean clustering error is 15.05% for the case of no noise and 15.2%, 15.6% and 16% for 5%, 10% and 15% Gaussian noise respectively. Furthermore, the standard deviation for each of the boxplots varies between 0.4% to 0.55%. This result clearly shows that the CPCA performance is quite robust to random sampling. Similar results were observed for other datasets and are not reported here for the purpose of brevity.

Rank preservation under downsampling

An interesting question about CPCA is if it preserves the underlying rank of the dataset under the proposed sampling scheme. Table 6.10 shows that the rank is preserved even in the presence of noise. For this experiment, we take different datasets and corrupt them with different types of noise and perform cross-validation for clustering using the parameter range for CPCA mentioned in Table 5.2. Then, we report the rank of $\tilde{\boldsymbol{X}}$ for the parameter corresponding to the minimum clustering error. As $\tilde{\boldsymbol{X}}$ is approximately low-rank so we use the following strategy to determine the rank r: $\tilde{\Sigma}_{r,r}/\tilde{\Sigma}_{1,1} < 0.1$. FRPCAG assumes that the number of clusters $\approx$ rank of the dataset. Our findings show that this assumption is almost satisfied for the sampled matrices even in the presence of various types of noise. Thus, the rank is preserved under the proposed sampling strategy. For clustering experiments, the lowest error with CPCA occurs when the rank $\approx$ number of clusters.

Table 6.11 – Variation of clustering error of CPCA with different uniform downsampling schemes / factors across rows and columns of the USPS dataset ($256 \times 10{,}000$).

downsampling (rows / cols)	**1**	**5**	**10**	**15**	**20**
1	0.16	0.16	0.21	0.21	0.21
2	0.21	0.23	0.23	0.24	0.25
4	0.26	0.30	0.31	0.31	0.31

Clustering error vs downsampling rate

Table 6.11 shows the variation of clustering error of CPCA with different downsampling factors across rows and columns of the USPS dataset ($256 \times 10,000$). Obviously, higher downsampling results in an increase in the clustering error. However, note that we can downsample the samples (columns) by a factor of 5 without observing an error increase. The downsampling of features results in an error increase because the number of features for this dataset is only 256 and downsampling results in a loss of information. Similar behavior can also be observed for the ORL small and ORL large datasets in Table 6.6 where the performance of CPCA is slightly worse than FRPCAG because the number of samples n for ORL is only 400.

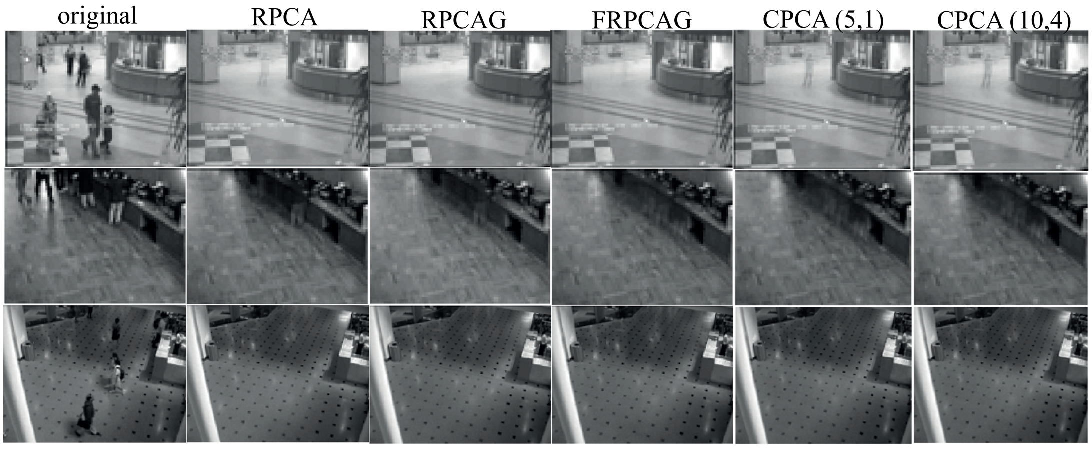


Figure 6.2 – Static background separation from videos using different PCA based models. The first row corresponds to a frame from the video of a shopping mall lobby, the second row to a restaurant food counter and the third row to an airport lobby. The leftmost plot in each row shows the actual frame, the other 5 show the recovered low-rank using RPCA, RPCAG, FRPCAG and CPCA with two different uniform downsampling schemes.

6.8.2 Low-rank and sparse decomposition

In order to demonstrate the effectiveness of our model to recover low-rank static background from videos we perform experiments on 1000 frames of 3 videos available online [7]. All the frames are vectorized and arranged in a data matrix $\boldsymbol{Y}$ whose columns correspond to frames. The graph $\mathbb{G}^c$ is constructed between the columns of $\boldsymbol{Y}$ and the graph $\mathbb{G}^r$ is constructed between the rows of $\boldsymbol{Y}$ using FLANN with $k_{nn} = 10$. Fig. 6.2 shows the recovery of low-rank frames for one actual frame of each of

---

[7] https://sites.google.com/site/backgroundsubtraction/test-sequences

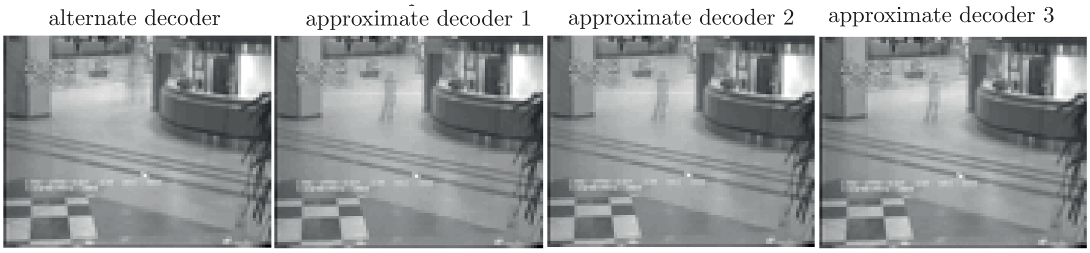


Figure 6.3 – A quality comparison of various low-rank decoders discussed in this work.

the videos. The first row corresponds to a frame from the video of a shopping mall lobby, the second row to a restaurant food counter and the third row to an airport lobby. The leftmost plot in each row shows the actual frame, the other 5 show the recovered low-rank representations using RPCA, RPCAG, FRPCAG and CPCA with two different uniform downsampling rates. For CPCA Algorithm 2 is used with $\phi = \ell_1$ norm and the rank r for the approximate decoder is set such that $\tilde{\boldsymbol{\Sigma}}_{r,r}/\tilde{\boldsymbol{\Sigma}}_{1,1} < 0.1$, where $\tilde{\boldsymbol{\Sigma}}$ are the singular values of $\tilde{\boldsymbol{X}}$.

For the 2nd and 3rd rows of Fig. 6.2 it can be seen that our proposed model is able to separate the static backgrounds very accurately from the moving people which do not belong to the static ground truth. However, the quality is slightly compromised in the 1st row where the shadow of the person appears in the low-rank frames recovered with CPCA. In fact, this person remains static for a long time in the video and the uniform sampling compromises the quality slightly.

Table 6.12 – Computational time in seconds of RPCA, RPCAG, FRPCAG and CPCA for low-rank recovery of different videos in Fig. 6.2.

Videos	RPCA	RPCAG	FRPCAG	CPCA (5,1)	CPCA (10,4)
1	2700	3550	120	21	8
2	1650	2130	85	15	6
3	3650	4100	152	32	11

Table 6.12 presents the computational time in seconds of RPCA, RPCAG, FRPCAG and CPCA for low-rank recovery of different videos in Fig. 6.2. The time reported here corresponds to steps 2 to 6 of Table 6.1, Algorithm 1 of [110] for FRPCAG, [1] for RPCA and [99] for RPCAG, excluding the construction of graphs $\mathbb{G}^r, \mathbb{G}^c$. The speed-up observed for these experiments from Table 6.12 is 10 times over FRPCAG and 100 times over RPCA and RPCAG.

Fig. 6.3 presents a comparison of the quality of the low-rank static background extracted using the alternate (6.8) and approximate decoders discussed in (6.18) for a video (1st row) of Fig. 6.2. Clearly, the alternate decoder performs slightly better than the approximate decoders but at the price of tuning of two model parameters.

Fig. 6.4 presents a comparison of the quality of low-rankness for the same video extracted using the approximate decoder (6.18) using different downsampling factors on the pixels and frames. It is obvious that the quality of low-rankness remains intact even with higher downsampling factors.

6.9 Conclusion

We presented Compressive PCA on Graphs (CPCA) which approximates a recovery of multilinear low-rank tensors on graphs from their sampled measurements. CPCA is supported by the proposed restricted isometry property (RIP) which is related to the coherence of the eigenvectors of graphs between the rows and columns of the data matrix. Accompanied with several efficient, parallel, parameter free and low-cost decoders for low-rank recovery and clustering, the presented framework gains a several orders of magnitude speed-up over the low-rank recovery methods like Robust PCA. The method is specially effective when the number of samples n and the features p are very large and one can afford to lose the exactness of the representation for gain in speed. CPCA implicitly brings into use the GMLSVD framework, i.e., it relies on the graph dirichlet priors rather than the explicit use of the graph eigenvectors. Our theoretical analysis reveals that CPCA targets exact recovery for low-rank matrices which are clusterable across the rows and columns. Thus, the error depends on the spectral

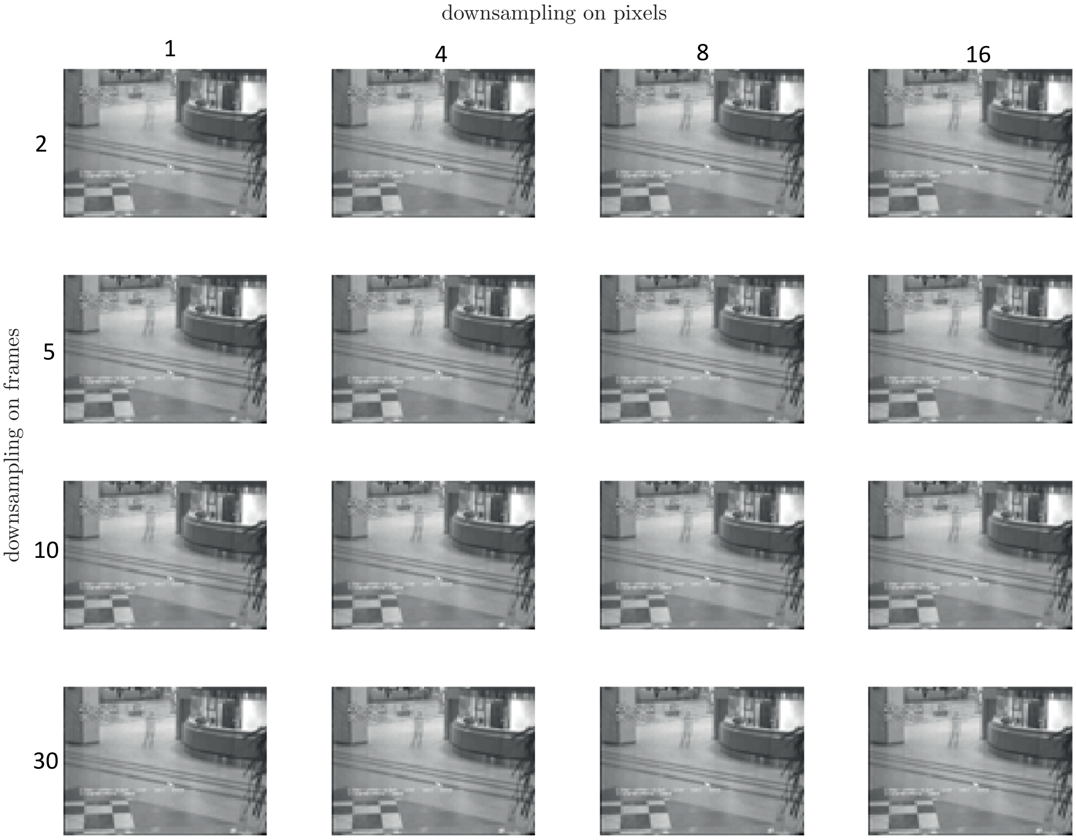


Figure 6.4 – A comparison of the quality of low-rank frames for the shopping mall video (1st row of Fig. 6.2) extracted using the approximate decoder (6.18) for different downsampling factors on the pixels and frames. It is obvious that the quality of low-rank remains intact even with higher downsampling factors.

gaps of the graph Laplacians. Extensive clustering experiments on 5 datasets with various types of noise and comparison with 11 state-of-the-art methods reveal the efficiency of our model. CPCA also achieves state-of-the-art results for background separation from videos.

7 Tensor Low-rank Decomposition on Graphs

Note: *Most of the parts of this chapter have appeared in the following arXiv submission:*

"N. Shahid, F. Grassi, P. Vandergheynst, "Multilinear Low-rank Tensors on Graphs & Applications", arXiv: preprint arXiv:1611.04835."

In chapters 5 and 6 we presented our matrix based low-rank decomposition methods, FRPCAG and its extension CPCA. The proposed methods are computationally much more feasible as compared to those which use nuclear norm, such as RPCA and RPCAG. However, this scalability comes at the cost of approximation which is quantified by the eigen-gaps of the graph Laplacians. We recall that the underlying low-rank signal model for FRPCAG and CPCA based methods is MLRTG (chapter 3), which states that a low-rank tensor can be encoded as a multilinear combination of the first few Laplacian eigenvectors, constructed from the flattened modes of a tensor or between the rows and columns of a matrix. FRPCAG and CPCA methods implicitly make use of the GMLSVD framework via graph Dirichlet priors and are suitable for the matrix recovery when both n and p are large.

Recall, from our discussion in Chapter 1 that similarly to the matrices, tensors can also be low-rank. In Chapters 1 and 3, we presented several applications of low-rank tensors. Typical examples of low-rank tensors include 3D hyper-spectral images, 3D EEG data (time series, electrodes, frequency), 4D FMRI data (x-dimension, y-dimension, slices, frequency) and 4D videos (x-dimension, y-dimension, z-dimension, time). One of the many possible ways to perform low-rank decomposition on tensors is to re-arrange them in the form of a matrix and apply the standard matrix methods that have been presented in this thesis. However, such high dimensional tensors would lose the interpretation of their distinct features / modes if arranged in the form of a matrix to perform low-rank decomposition. Simply put, it is not possible to perform a multi-modal low-rank tensor analysis via matrix based methods.

To enable multi-modal analysis, Tensor Robust PCA (TRPCA) based methods have been proposed which perform low-rank and sparse decomposition on the tensor as a whole [16], [17], [18]. Like the RPCA algorithm on a matrix, such methods need the notion of nuclear norm on a tensor. Tensor nuclear norm [19], which minimizes the sum of nuclear norms on each of the matrices $\boldsymbol{Y}^{\mu}$ formed by flattening the various modes of a tensor, is used for this purpose. For a 3D tensor $\mathcal{Y} \in \mathbb{R}^{n \times n \times n}$, *this costs* $\mathcal{O}(n^4)$ *per iteration, which is computationally prohibitive, even for n as small as 100.*

Tensors often possess some additional intrinsic structure which should be exploited during the analysis. For example, for the case of 3D EEG tensor (time series, electrodes, frequency), the electrodes belong to the spatial locations of the brain. A small subset of the electrodes in a given neighborhood might possess similar signal characteristics. In the sequel we call such types of correlations along a specific mode of a tensor as *intra-mode correlations*. Furthermore, there might exist some correlations among the features from different modes. For example, for the case of EEG tensor, some common patterns might repeat in the joint time-frequency domain. We call such correlations as *inter-mode correlations*.

Graph regularized low-rank tensors: Some attempts to incorporate structure in the tensors via graph regularization have appeared in the past [115], [72], [73], [116]. Similarly to the factorized PCA model [3], these methods incorporate a graph regularization on the factors of the low-rank tensor. This strategy enables a structured recovery of the factors of the low-rank tensor, but also makes these models non-convex and compromises the orthonormality of the factors.

Fast tensor decomposition: Several methods have been presented in the past which propose fast tensor based decompositions [117], [118], [119], [120], [121], [122], [123]. These methods can be divided into three types based on the technique involved: 1) fast alternating least squares (ALS) algorithms which mostly rely on speeding up the convergence of various methods, 2) randomized sampling of the tensor in every dimension. Such methods iteratively sample the tensor along all the modes except the one under consideration and then perform ALS updates, 3) the sketching of empirical moments of the tensor. Such methods have recently started appearing in the literature and require the computation of higher order moments of the datasets. However, none of these methods can be used for low-rank and sparse decomposition and they do not use graphs to exploit intra-mode and inter-mode correlations.

Thus, our goal here is to propose a scalable tensor low-rank decomposition framework that can exploit the intra and inter-mode correlations. A straight-froward extension would be to extend our FRPCAG framework for tensors as well. For simplicity we work with 3D tensors of same size n and rank r in each dimension. Let $\mathscr{Y} \in \mathbb{R}^{n\times n\times n}$ be a noisy tensor such that $\mathscr{Y} = \mathscr{X}^* + \boldsymbol{\eta}$, where $\mathscr{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ and $\boldsymbol{\eta}$ models noise and errors. Furthermore, let $\boldsymbol{Y}^\mu$ and $\boldsymbol{X}*^\mu$ be the μ^{th} matricization of $\mathscr{Y}$ and $\mathscr{X}^*$. Then, a straight-forward extension of FRPCAG for recovering $\mathscr{X}$ would be to solve the following optimization problem:

$$\min_{\mathscr{X}} \phi(\mathscr{Y} - \mathscr{X}) + \sum_{\mu} \gamma_\mu \operatorname{tr}(\boldsymbol{X}^\mu \boldsymbol{L}^\mu \boldsymbol{X}^{\mu\top}), \tag{7.1}$$

where $\boldsymbol{L}^\mu$ is the combinatorial graph Laplacian constructed between the rows of $\boldsymbol{Y}^\mu$ and γ_μ are the regularization parameters.

Let us analyze the above optimization problem in detail.

- First, note that the the term $\operatorname{tr}(\boldsymbol{X}^\mu \boldsymbol{L}^\mu \boldsymbol{X}^{\mu\top})$ is differentiable $\forall\ \mu$ and can be minimized by a gradient step, i.e, $\nabla(\cdot) = \boldsymbol{L}^\mu \boldsymbol{X}^\mu$, which involves the multiplication of a sparse matrix $\boldsymbol{L}^\mu \in \mathbb{R}^{n\times n}$ with $\boldsymbol{X} \in \mathbb{R}^{n\times n^2}$. The computational complexity of this step is $\mathscr{O}(n^2|\mathbb{E}^\mu|) \approx \mathscr{O}(n^3 k_{nn})$. Even though this is less than $\mathscr{O}(n^4)$ for TRPCA, it is still infeasible for large n. Nevertheless, this cost can be minimized using a GPU implementation of matrix multiplication.

- Secondly, we point out the manifold shrinking effect, as discussed in Chapter 5, which is a drawback of FRPCAG, due to non-linear penalization of the singular values of $\boldsymbol{X}^\mu$, which indirectly depends on the spectrum of Laplacians. This effect is even more prominent in the tensor based optimization problem above, because the singular values of the tensor $\mathscr{X}$ are being penalized in all the modes. Although, we proposed the design of deterministic graph filters 'g' in Chapter 5 to improve the singular value penalization property, the non-linear thresholding effect still exists and penalization occurs in an undesired manner.

- Finally, note that FRPCAG relies on approximate MLRTG framework, i.e, $\boldsymbol{X} = \boldsymbol{P}^r_k \boldsymbol{A} + \bar{\boldsymbol{P}}^r_k \bar{\boldsymbol{A}}$, where $\|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F$ due to indirect control on the singular values of $\boldsymbol{X}$. This implies that $\boldsymbol{X}(i) \in span(\boldsymbol{P}^r)$, the complete set of eigenvectors. Thus, one cannot come up with a simple modification of FRPCAG to reduce the computational complexity.

To conclude, optimization problem (7.1) is not a feasible extension of FRPCAG for tensors due to the above mentioned reasons. Thus, we need to come up with a novel generalized approximation problem for tensor based applications.

In this chapter we propose a novel tensor low-rank decomposition method which overcomes the limitations of the state-of-the-art. In particular, our method is convex, scalable, performs a multimodal analysis by analyzing the tensor in its actual form, exploits the intrinsic intra-mode correlations or structures that might exist and also exploits inter-mode correlations which arise among the features of various modes. The proposed methodology is general in the sense that it may also be used for the recovery of low-rank matrices as well. A specialty of our proposed framework is that it makes explicit use of the GMLSVD framework to determine the singular vectors of a low-rank tensor from the graph eigenvectors. As the complexity of computing graph eigenvectors scales linearly with the n, the framework is reasonable for tensors for which the number of samples n, in each mode is not large but n^d, where d is the number of dimensions of the tensor, is large.

7.1 The General Framework

In order to derive our proposed tensor based framework, we start by studying a simpler matrix based approximation problem which is a state-of-the-art alternative to the FRPCAG framework. Throughout, we will perform the derivation for a general ℓ_p loss function and discuss applications for different cases of ℓ_p norm at the end. A number of such problems have been proposed under the framework of graph regularized matrix factorization.

$$\min_{\boldsymbol{U}_r, \boldsymbol{V}_r} \phi(\boldsymbol{Y} - \boldsymbol{U}_r \boldsymbol{V}_r^\top) + \gamma_r \operatorname{tr}(\boldsymbol{U}_r^\top \boldsymbol{L}^r \boldsymbol{U}_r) + \gamma_c \operatorname{tr}(\boldsymbol{V}_r^\top \boldsymbol{L}^c \boldsymbol{V}_r). \tag{7.2}$$

In fact we study a more general optimization problem involving the use of deterministic graph filters 'g' for eq. (7.2), as following:

$$\min_{\boldsymbol{U}_r, \boldsymbol{V}_r} \phi(\boldsymbol{Y} - \boldsymbol{U}_r \boldsymbol{V}_r^\top) + \gamma_r \operatorname{tr}(\boldsymbol{U}_r^\top g(\boldsymbol{L}^r) \boldsymbol{U}_r) + \gamma_c \operatorname{tr}(\boldsymbol{V}_r^\top g(\boldsymbol{L}^c) \boldsymbol{V}_r). \tag{7.3}$$

where $\boldsymbol{U}_r \in \mathbb{R}^{p \times r}$ and $\boldsymbol{V}_r \in \mathbb{R}^{n \times r}$ are the singular vectors or factors of the low-rank approximation of $\boldsymbol{Y}$ and $\phi(\cdot)$ is ℓ_p loss function. For our purpose, we define the graph filter 'g' to be a monotonically increasing polynomial function as following:

$$g(\boldsymbol{x}) = \boldsymbol{x}^q \quad \text{where} \quad q \geq 2.$$

The above problem is computationally much less expensive as compared to FRPCAG, because it requires the multiplication of thin matrices with sparse Laplacians. For $\boldsymbol{U} \in \mathbb{R}^{p \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$, the computational complexity scales with $\mathcal{O}((n+p)rk_{nn})$, which is much less as compared to that of FRPCAG ($\mathcal{O}(npk_{nn})$) as $r \ll p < n$. However, the problem is non-convex and hence suffers from the problem of local minima.

We follow a 3-step approach to transform this problem into a convex and computationally tractable optimization problem:

1. Transformation into a convex, weighted nuclear norm problem by analyzing the trace terms.
2. Transformation of the weighted nuclear norm problem into computationally tractable problem using our MLRTG framework.
3. Extension to tensors.

7.1.1 Step 1: Weighted Nuclear Norm Problem

First note that the trace terms above can be re-written as:

$$\begin{aligned}\gamma_r \operatorname{tr}\left(\boldsymbol{U}_r^\top g(\boldsymbol{L}^r)\boldsymbol{U}_r\right) + \gamma_c \operatorname{tr}\left(\boldsymbol{V}_r^\top g(\boldsymbol{L}^c)\boldsymbol{V}_r\right) &= \left\| \sqrt{\gamma_r}\sqrt{g(\boldsymbol{L}^{r\top})}\boldsymbol{U}_r \right\|_F^2 + \left\| \sqrt{\gamma_c}\sqrt{g(\boldsymbol{L}^{c\top})}\boldsymbol{V}_r \right\|_F^2 \\ &= \left\| \sqrt{\gamma_r}\sqrt{g(\boldsymbol{\Lambda}^r)}\boldsymbol{P}^{r\top}\boldsymbol{U}_r \right\|_F^2 + \left\| \sqrt{\gamma_c}\sqrt{g(\boldsymbol{\Lambda}^c)}\boldsymbol{P}^{c\top}\boldsymbol{V}_r \right\|_F^2,\end{aligned}$$

where the last step above follows from the fact that $\sqrt{g(\boldsymbol{L}^c)} = \boldsymbol{P}^c\sqrt{g(\boldsymbol{\Lambda}^c)}\boldsymbol{P}^{c\top}$, $\sqrt{g(\boldsymbol{L}^r)} = \boldsymbol{P}_k^r\sqrt{g(\boldsymbol{\Lambda}^r)}\boldsymbol{P}^{r\top}$

Using the results from [34], let $\boldsymbol{A} = \boldsymbol{P}^r g(\boldsymbol{\Lambda}^r)^{-1/2}$ and $\boldsymbol{B} = \boldsymbol{P}^c g(\boldsymbol{\Lambda}^c)^{-1/2}$ and define a preferential atomic set:

$$\mathbb{A} = \{\alpha_i = \boldsymbol{U}_r(i)\boldsymbol{V}_r^\top(i) \ : \ \boldsymbol{U}_r(i) = \boldsymbol{A}\boldsymbol{w}_i, \boldsymbol{V}_r(i) = \boldsymbol{B}\boldsymbol{h}_i, \|\boldsymbol{w}_i\| = \|\boldsymbol{h}_i\| = 1\}$$

and an atomic norm as following:

$$\|\boldsymbol{\beta}\|_{\mathbb{A}} = \inf \sum_{\alpha_i \in \mathbb{A}} |\boldsymbol{c}_i| \quad \text{s.t. } \boldsymbol{\beta} = \sum_{\alpha_i \in \mathbb{A}} \boldsymbol{c}_i \alpha_i.$$

Then, from Theorem 1 in [34],

$$\|\boldsymbol{\beta}\|_{\mathbb{A}} = \inf_{\boldsymbol{U}_r, \boldsymbol{V}_r} \frac{1}{2}\{\|\boldsymbol{A}^{-1}\boldsymbol{U}_r\|_F^2 + \|\boldsymbol{B}^{-1}\boldsymbol{V}_r\|_F^2\} \quad \text{s.t. } \boldsymbol{\beta} = \boldsymbol{U}_r\boldsymbol{V}_r^\top,$$

which is equivalent to

$$\|\boldsymbol{\beta}\|_{\mathbb{A}} = \|\boldsymbol{A}^{-1}\boldsymbol{\beta}\boldsymbol{B}\|_* = \left\| \sqrt{\gamma_r}\sqrt{g(\boldsymbol{\Lambda}^r)}\boldsymbol{P}^{r\top}\boldsymbol{\beta}\boldsymbol{P}^c\sqrt{g(\boldsymbol{\Lambda}^c)}\sqrt{\gamma_c} \right\|_*,$$

which corresponds to the equivalence of the factorized graph regularized problem to that of a weighted nuclear norm, where the weights are given by

$$\bar{\boldsymbol{W}} = \sqrt{\gamma_r\gamma_c g(\boldsymbol{\Lambda}^c \circ \boldsymbol{\Lambda}^r)},$$

where $\circ$ denotes the Hadamard product. Now, as

$$\lambda_0^r \leq \lambda_1^r \leq \cdots \leq \lambda_{p-1}^r \quad \text{and} \quad \lambda_0^c \leq \lambda_1^c \leq \cdots \leq \lambda_{n-1}^c, \quad \text{this implies}$$

$$\lambda_0^r\lambda_0^c \leq \lambda_1^r\lambda_1^c \leq \cdots \leq \lambda_p^r\lambda_p^c,$$

Finally, for our definition of $g(\cdot)$, this implies

$$\sqrt{g(\lambda_0^r \lambda_0^c)} \leq \sqrt{g(\lambda_1^r \lambda_1^c)} \leq \cdots \leq \sqrt{g(\lambda_p^r \lambda_p^c)},$$

assuming $p < n$. Thus, the weights are in the non-decreasing order. As nuclear norm provides a model for singular value soft thresholding and singular vector determination, the weighted nuclear norm above results in the following effect on the singular values and singular vectors during every iteration of the algorithm:

$$\hat{\boldsymbol{\Sigma}} = \operatorname{sgn}(\boldsymbol{\Sigma}) \max\Big(|\boldsymbol{\Sigma}| - \sqrt{\gamma_c \gamma_r g(\boldsymbol{\Lambda}^c \circ \boldsymbol{\Lambda}^r)}, 0\Big), \quad \boldsymbol{U}_r(i) \in span(\boldsymbol{P}^r) \quad \text{and} \quad \boldsymbol{V}_r(i) \in span(\boldsymbol{P}^c).$$

Thus, more preference is given to the lower atoms of the eigenvalue matrices $\boldsymbol{P}^c$ and $\boldsymbol{P}^r$ in the low-rank extraction framework, which is similar to the motivation behind our MLRTG framework and the approximation problems proposed so far in this thesis.

Thus, the optimization problem (7.3), after the above transformations and using $\gamma = \sqrt{\gamma_r \gamma_c}$ can be written as:

$$\min_{\boldsymbol{X}} \phi(\boldsymbol{Y} - \boldsymbol{X}) + \gamma \left\| \sqrt{g(\boldsymbol{\Lambda}^r)} \boldsymbol{P}^{r\top} \boldsymbol{X} \boldsymbol{P}^c \sqrt{g(\boldsymbol{\Lambda}^c)} \right\|_* . \tag{7.4}$$

Note that this problem is convex as it involves a weighted nuclear norm operator. The weights of the nucelar norm are in the non-decreasing order, which according to Corollary 2 in [124], results in the convexity of the above optimization problem.

7.1.2 Step 2: Computationally feasible weighted nuclear norm problem

Although eq. (7.4) is a convex equivalent of eq. (7.3), the weighted nuclear norm above requires the full knowledge of the eigenvectors $\boldsymbol{P}^c$ and $\boldsymbol{P}^r$ of the Laplacians $\boldsymbol{L}^c$ and $\boldsymbol{L}^r$. This requires a computational complexity of $\mathcal{O}(n^3)$ and $\mathcal{O}(p^3)$. Furthermore, the nuclear norm above is on the full matrix of size $p \times n$ and scales as $\mathcal{O}(np^2 I)$, where I is the number of iterations. The nuclear norm operator starts from a full rank matrix and gradually reduces the rank in every iteration of the algorithm which results in a high complexity. So overall, including the cost of graph construction using FLANN, the solution of eq. (7.4) scales as $\mathcal{O}(Ip^2 n + n^3 + p^3 + np\log(n) + np\log(p))$, which is computationally prohibitive.

It is possible to re-write eq. (7.4) using our MLRTG framework, in a form which could reduce the computational complexity significantly. Assuming that the eigen-gap of the cartesian product graph $\boldsymbol{L} = \boldsymbol{L}^c \times \boldsymbol{L}^r$ exists, i.e,

$$\max\{\lambda_k^\mu\} \ll \min\{\lambda_{k+1}^\mu\} \; \forall \; \mu,$$

then for our choice of 'g'

$$\max\left\{\sqrt{g(\lambda_k^\mu)}\right\} \ll \min\left\{\sqrt{g(\lambda_{k+1}^\mu)}\right\} \; \forall \; \mu.$$

Now, let $\boldsymbol{X} = \boldsymbol{P}_k^r \boldsymbol{Z} \boldsymbol{P}_k^{c\top}$, $\boldsymbol{P}^r = [\boldsymbol{P}_k^r | \bar{\boldsymbol{P}}_k^r]$ and $\boldsymbol{P}^c = [\boldsymbol{P}_k^c | \bar{\boldsymbol{P}}_k^c]$, we get:

$$\left\| \sqrt{g(\boldsymbol{\Lambda}^r)} \boldsymbol{P}^{r\top} \boldsymbol{X} \boldsymbol{P}^c \sqrt{g(\boldsymbol{\Lambda}^c)} \right\|_* = \left\| \sqrt{g(\boldsymbol{\Lambda}^r)} [\boldsymbol{P}_k^r | \bar{\boldsymbol{P}}_k^r]^\top \boldsymbol{P}_k^r \boldsymbol{Z} \boldsymbol{P}_k^{c\top} [\boldsymbol{P}_k^c | \bar{\boldsymbol{P}}_k^c] \sqrt{g(\boldsymbol{\Lambda}^c)} \right\|_* \tag{7.5}$$

Using $\boldsymbol{P}_k^{r\top}\boldsymbol{P}_k^r = \boldsymbol{I}_k$, $\boldsymbol{P}_k^{c\top}\boldsymbol{P}_k^c = \boldsymbol{I}_k$, $\bar{\boldsymbol{P}}_k^{c\top}\boldsymbol{P}_k^c = \boldsymbol{0}$ and $\bar{\boldsymbol{P}}_k^{r\top}\boldsymbol{P}_k^r = \boldsymbol{0}$, we get:

$$\left\|\sqrt{g(\boldsymbol{\Lambda}^r)}\boldsymbol{P}^{r\top}\boldsymbol{X}\boldsymbol{P}^c\sqrt{g(\boldsymbol{\Lambda}^c)}\right\|_* = \left\|\sqrt{g(\boldsymbol{\Lambda}^r)}\boldsymbol{Z}\sqrt{g(\boldsymbol{\Lambda}^c)}\right\|_*,$$

where $\boldsymbol{Z} \in \mathbb{R}^{k\times k}$ and $k \ll p < n$. Now, the optimization problem (7.4) reduces to:

$$\min_{\boldsymbol{Z}} \phi\Big(\boldsymbol{Y} - \boldsymbol{P}_k^r\boldsymbol{Z}\boldsymbol{P}_k^{c\top}\Big) + \gamma\left\|f(\boldsymbol{\Lambda}_k^r)\boldsymbol{Z}f(\boldsymbol{\Lambda}_k^c)\right\|_*, \tag{7.6}$$

where we use $f(\boldsymbol{x}) = \sqrt{g(\boldsymbol{x})} = \boldsymbol{x}^{q/2}$ and $q \geq 2$. As compared to eq. (7.4), the complexity of the above problem scales with $\mathcal{O}(k^3 I)$, where $k \ll p < n$. Similarly to eq. (7.4), it is a weighted nuclear norm problem, where the weights are given by the Hadamard product of diagonal matrices $f(\boldsymbol{\Lambda}_k^r) \circ f(\boldsymbol{\Lambda}_k^c)$.

Finally, we show that is possible to re-write eq. (7.6) in terms of $\boldsymbol{X}$ which reveals the intuition of this optimization problem. Using $\boldsymbol{Z} = \boldsymbol{P}_k^{r\top}\boldsymbol{X}\boldsymbol{P}_k^c$ back in eq. (7.6), we get,

$$\min_{\boldsymbol{X}} \phi\Big(\boldsymbol{Y} - \boldsymbol{X}\Big) + \gamma\left\|f(\boldsymbol{\Lambda}_k^r)\boldsymbol{P}_k^{r\top}\boldsymbol{X}\boldsymbol{P}_k^c f(\boldsymbol{\Lambda}_k^c)\right\|_*, \tag{7.7}$$

which can also be written as:

$$\min_{\boldsymbol{X}} \phi\Big(\boldsymbol{Y} - \boldsymbol{X}\Big) + \gamma\left\|\boldsymbol{P}_k^{r\top}\boldsymbol{X}\boldsymbol{P}_k^c\right\|_{*f(\boldsymbol{\Lambda}_k^r)\circ f(\boldsymbol{\Lambda}_k^c)}.$$

Note that multiplying the term inside nuclear norm by $\boldsymbol{P}_k^r$ from left and $\boldsymbol{P}_k^{c\top}$ from right does not change the problem. Thus,

$$\min_{\boldsymbol{X}} \phi\Big(\boldsymbol{Y} - \boldsymbol{X}\Big) + \gamma\left\|\boldsymbol{P}_k^r\boldsymbol{P}_k^{r\top}\boldsymbol{X}\boldsymbol{P}_k^c\boldsymbol{P}_k^{c\top}\right\|_{*f(\boldsymbol{\Lambda}_k^r)\circ f(\boldsymbol{\Lambda}_k^c)}.$$

Now, let

$$h(\boldsymbol{L}^r) = \boldsymbol{P}_k^r h(\boldsymbol{\Lambda}_k^r)\boldsymbol{P}_k^{r\top} + \bar{\boldsymbol{P}}_k^r h(\bar{\boldsymbol{\Lambda}}_k^r)\bar{\boldsymbol{P}}_k^{r\top} \quad \text{s.t.} \quad h(x) = 1, \text{ if } x \leq \lambda_k^r \;\&\; h(x) = 0 \text{ if } x > \lambda_k^r.$$

$$h(\boldsymbol{L}^c) = \boldsymbol{P}_k^c h(\boldsymbol{\Lambda}_k^c)\boldsymbol{P}_k^{c\top} + \bar{\boldsymbol{P}}_k^c h(\bar{\boldsymbol{\Lambda}}_k^c)\bar{\boldsymbol{P}}_k^{c\top} \quad \text{s.t.} \quad h(x) = 1, \text{ if } x \leq \lambda_k^c \;\&\; h(x) = 0 \text{ if } x > \lambda_k^c.$$

It is easy to note that for the above definition of $h(\cdot)$, we have:

$$h(\boldsymbol{L}^r) = \underline{\boldsymbol{L}}^r = \boldsymbol{P}_k^r\boldsymbol{P}_k^{r\top} \quad \text{and} \quad h(\boldsymbol{L}^c) = \underline{\boldsymbol{L}}^c = \boldsymbol{P}_k^c\boldsymbol{P}_k^{c\top}.$$

Using this, the final problem can be written as:

$$\min_{\boldsymbol{X}} \phi\Big(\boldsymbol{Y} - \boldsymbol{X}\Big) + \gamma\left\|\underline{\boldsymbol{L}}^r\boldsymbol{X}\underline{\boldsymbol{L}}^c\right\|_{*f(\boldsymbol{\Lambda}_k^r)\circ f(\boldsymbol{\Lambda}_k^c)}.$$

The above problem states that our framework is equivalent to low-rank filtering the matrix $\boldsymbol{X}$, jointly with the rank k Laplacians $\underline{\boldsymbol{L}}^r$ and $\underline{\boldsymbol{L}}^c$ and then soft-thresholding the singular values with the graph eigenvalues. It is possible to perform fast filtering of $\boldsymbol{X}$ by using Chebyshev polynomials [20]. However, the nuclear norm on the full matrix $\boldsymbol{X}$ is still expensive to compute. Therefore, we solve eq. (7.6) directly.

7.1.3 Step 3: Extension for tensors

It is straight-forward to extend eq. (7.6) for tensors. For the case of tenors, one uses the notion of tensor nuclear norm [19], which is a sum of nuclear norms of the matrices formed by flattening out the various modes of tensor. Consider a 3D-tensor $\mathcal{Y} \in \mathbb{R}^{n \times n \times n}$ and a GCT $\mathcal{Z} \in \mathbb{R}^{k \times k \times k}$ and let

$$f_\mu(\boldsymbol{\Lambda}_k) = f(\boldsymbol{\Lambda}_k^\mu) \circ \left[\Big(\times_{i \neq \mu} f(\boldsymbol{\Lambda}_k^i) \Big) \right]_{k_\mu} ,$$

where $\times_i$ denotes the Cartesian product of the matrices and $[\cdot]_{k_\mu}$ denotes the selection of the first k_μ entries of the resultant matrix. E.g.,

$$f_1(\boldsymbol{\Lambda}_k) = f(\boldsymbol{\Lambda}_k^1) \circ \left[\Big(f(\boldsymbol{\Lambda}_k^2) \times f(\boldsymbol{\Lambda}_k^3) \Big) \right]_k = f(\boldsymbol{\Lambda}_k^1) \circ \left[\Big(f(\boldsymbol{\Lambda}_k^2) \otimes \boldsymbol{I}_k + \boldsymbol{I}_k \otimes f(\boldsymbol{\Lambda}_k^3) \Big) \right]_k ,$$

then we can re-write eq. (7.6) as following:

$$\min_{\mathcal{Z}} \phi\Big(\mathrm{vec}(\mathcal{Y}) - (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3) \mathrm{vec}(\mathcal{Z}) \Big) + \gamma \sum_\mu \left\| f_\mu(\boldsymbol{\Lambda}_k) \boldsymbol{Z}^\mu \right\|_* , \tag{7.8}$$

where $\sum_\mu \left\| f_\mu(\boldsymbol{\Lambda}_k) \boldsymbol{Z}^\mu \right\|_*$ denotes a weighted tensor nuclear norm.

Comments

For the case of a matrix, as compared to FRPCAG, where $\boldsymbol{U}_r = \boldsymbol{P}_k^r \boldsymbol{A} + \boldsymbol{P}_k^r \bar{\boldsymbol{A}}$, such that $\|\bar{\boldsymbol{A}}\|_F \ll \|\boldsymbol{A}\|_F$, this framework explicitly assumes the following:

$$\boldsymbol{U}_r(i) \in span(\boldsymbol{P}_k^r) \quad \text{and} \quad \boldsymbol{V}_r(i) \in span(\boldsymbol{P}_k^c), \ \forall \ i.$$

Thus, unlike FRPCAG where one always gets an approximate low-rank representation, the rank for this framework in every mode can be at-most k. The choice of an appropriate value of k for every mode of the tensor was discussed in Chapter 4 and will be discussed again in the experimental section of this chapter. A detailed theoretical analysis on the approximation error under the above framework and its comparison to FRPCAG will be done in the up-coming section of this chapter.

7.2 Proposed Approximation Problems

Eq. (7.8) is a general setup for tensor based approximation problems which holds for an ℓ_p loss function $\phi(\cdot)$. Depending on the type of noise in $\mathcal{Y}$, eq. (7.8), with a proper choice of $\phi(\cdot)$ can be used for a variety of applications. We point out two specific applications corresponding to the following cases: 1) when noise is Gaussian and 2) when the noise is sparse or Laplacian. For the first case, one can use $\phi = \ell_2$ norm and for the case of sparse noise one can use $\phi = \ell_1$ norm. In this section we derive and study in detail the two approximation problems for these two cases. More specifically, we propose two approximation problems for the two cases:

1. Graph Core Tensor Pursuit (GCTP): when the noise is Gaussian. Our GMLSVD method, proposed in Chapter 4 is a special case of GCTP.

2. Tensor Robust PCA on Graphs (TRPCAG): when the noise is sparse. This method can be used for tensor low-rank and sparse decomposition.

7.2.1 Graph Core Tensor Pursuit

When the noise is Gaussian, one can use $\phi = \ell_2$ norm and eq. (7.8) becomes:

$$\min_{\mathcal{Z}} \left\| \operatorname{vec}(\mathcal{Y}) - (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3) \operatorname{vec}(\mathcal{Z}) \right\|_2^2 + \gamma \sum_{\mu} \left\| f_{\mu}(\boldsymbol{\Lambda}_k) \boldsymbol{Z}^{\mu} \right\|_* .$$

Using $\operatorname{vec}(\mathcal{Y}) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3) \operatorname{vec}(\hat{\mathcal{Z}})$, the above problem can be transformed as following:

$$\min_{\mathcal{Z}} \left\| \operatorname{vec}(\hat{\mathcal{Z}}) - \operatorname{vec}(\mathcal{Z}) \right\|_2^2 + \gamma \sum_{\mu} \left\| f_{\mu}(\boldsymbol{\Lambda}_k) \boldsymbol{Z}^{\mu} \right\|_* , \tag{7.9}$$

where $\operatorname{vec}(\hat{\mathcal{Z}}) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)^\top \operatorname{vec}(\mathcal{Y})$ is the GCT determined from the noisy tensor $\mathcal{Y}$. We call the above problem 'Graph Core Tensor Pursuit' (GCTP). Starting from the GCT $\hat{\mathcal{Z}}$ of a noisy tensor $\mathcal{Y}$, we would like to determine the clean GCT $\mathcal{Z}$ such that the weighted tensor nuclear norm of $\mathcal{Z}$ is minimized. Thus, one is solely concerned about recovering the GCT only.

Algorithm & Complexity

To solve the GCTP problem above, it is important to note that due to the ℓ_2 norm, eq. (7.9) is in the form that directly defines a proximal operator of the weighted nuclear norm operator $\sum_{\mu} \left\| f(\boldsymbol{\Lambda}_k) \boldsymbol{Z}^{\mu} \right\|_*$. Thus, one just needs to solve for the proximal operator corresponding to a matrix nuclear norm (for each of the matricized modes of $\mathcal{Z}$) and iterate. We will defer defining an iterative algorithm for solving GCTP until the next section, where we discuss Tensor Robust PCA on Graphs. The steps highlighted in Algorithm 1, except the $\operatorname{prox}_{s_0}(\cdot)$ correspond to the GCTP algorithm. The tensor nuclear norm operator scales with a complexity of $\mathcal{O}(Ik^4)$ for a GCT of size k^3.

GMLSVD: A special case of GCTP

Recall that the role of nuclear norm is to decompose a data matrix into its singular vectors and singular values, such that their product can best approximate the data matrix. Similarly to the case of a matrix, the tensor nuclear norm plays a similar role, except that SVD operation is performed on the flattened modes of the tensor. Thus, our GCTP problem above is implicitly performing an SVD of flattened tensors in every iteration which is equivalent to doing an MLSVD of the tensor. This said, when $\hat{\mathcal{Z}}$ or $\mathcal{Y}$ has no noise, then one does not need a thresholding of the singular values. This amounts to the case where $\gamma \to 0$ and $f(\boldsymbol{\Lambda}_k) = \boldsymbol{I}_k$ i.e., the optimization problem in eq. (7.9) can be re-written as following:

$$\begin{aligned} &\min_{\boldsymbol{A}_k^{\mu}, \mathcal{R}} \left\| \operatorname{vec}(\hat{\mathcal{Z}}) - (\boldsymbol{A}_k^1 \otimes \boldsymbol{A}_k^2 \otimes \boldsymbol{A}_k^3) \operatorname{vec}(\mathcal{R}) \right\|_2^2 \\ &\qquad \text{s.t. } \boldsymbol{A}^{\mu\top} \boldsymbol{A}^{\mu} = \boldsymbol{I}, \end{aligned} \tag{7.10}$$

i.e., in terms of the singular vectors and the core tensor of the GCT $\mathcal{Z}$. This is equivalent to the GMLSVD framework which was proposed in Chapter 4. Thus GMLSVD is a special case of GCTP.

Why a weighted nuclear norm?

Our framework results in a rank k approximation of the tensor in each of the modes and the noise in the lower singular values is automatically removed. This thresholding can also be achieved by using the GMLSVD framework only. However, for many real tensors, noise effects the higher singular values as well but GMLSVD does not cater for this. In order to remove noise in the higher part of the spectrum, a soft-thresholding algorithm, like the one used for nuclear norm minimization is more suitable. Our proposed framework, however, uses a weighted nuclear norm and is even more efficient than the standard nuclear norm operator to remove large amount of noise. While we show this in detail in the experimental section, a simple argument to support our claim is as follows: The higher the noise in tensor $\mathcal{Y}$, the more erroneous are the singular values of lower magnitude. As the weights of nuclear norm in our framework are monotonically increasing function of the non-decreasing graph eigenvalues, each of the singular values is thresholded smartly according to the spectrum of the graph. In simple words, the high magnitude singular values are thresholded less than the lower magnitude and this thresholding effect varies according to the spectrum of graph Laplacians.

7.2.2 Tensor Robust PCA on Graphs

For the case when the noise is sparse, one can use $\phi = \ell_1$ norm in eq. (7.8).

$$\min_{\mathcal{Z}} \left\| \mathrm{vec}(\mathcal{Y}) - (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\,\mathrm{vec}(\mathcal{Z}) \right\|_1 + \gamma \sum_{\mu} \left\| f_\mu(\boldsymbol{\Lambda}_k)\boldsymbol{Z}^\mu \right\|_*, \tag{7.11}$$

which we call as 'Tensor Robust PCA on Graphs' (TRPCAG). A typical application of this optimization problem is low-rank and sparse decomposition, several examples of which we will see in the experiments.

Algorithm for TRPCAG

We use Parallel Proximal Splitting method [104] to solve this problem. First, we re-write TRPCAG (eq. (7.11)) as:

$$\min_{\mathcal{Z}} \|\boldsymbol{P}_k^1 \boldsymbol{Z}^1 \boldsymbol{P}_{k^2}^{-\mu\top} - \boldsymbol{Y}^1\|_1 + \gamma \sum_{\mu} \left\| f_\mu(\boldsymbol{\Lambda}_k)\boldsymbol{Z}^\mu \right\|_*,$$

where $\boldsymbol{Y}^1$ is the flattened matrix across mode 1 and $\boldsymbol{P}_{k^2}^{-\mu} = \boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2$. The objective function above can be split as $s(\boldsymbol{x}) = s_0(\boldsymbol{x}) + \sum_\mu s_\mu(\boldsymbol{x})$, where $s_0(\cdot) = \|\boldsymbol{P}_k^1 \boldsymbol{Z}^1 \boldsymbol{P}_{k^2}^{-\mu\top} - \boldsymbol{Y}^1\|_1$ and $s_\mu(\cdot) = \left\| f_\mu(\boldsymbol{\Lambda}_k)\boldsymbol{Z}^\mu \right\|_*$. To use the parallel proximal splitting method for the problem above, we need to define the proximal operators of each of the functions $s_\mu(\cdot)$.

Let $\boldsymbol{\Omega}(\boldsymbol{Z},\tau)$ denote the element-wise soft-thresholding matrix operator:

$$\boldsymbol{\Omega}(\boldsymbol{Z},\tau) = \mathrm{sgn}(\boldsymbol{Z})\max(|\boldsymbol{Z}| - \tau, 0),$$

then we can define

$$D_\mu(\boldsymbol{Z}^\mu, \tau) = \boldsymbol{A}_k^1 \boldsymbol{\Omega}(\boldsymbol{R}^\mu, \tau) \boldsymbol{A}_{k^2}^{-\mu\top},$$

as the singular value thresholding operator for matrix $\boldsymbol{Z}^\mu$, where $\boldsymbol{Z}^\mu = \boldsymbol{A}_k^1 \boldsymbol{R}^\mu \boldsymbol{A}_{k^2}^{-\mu\top}$ is any singular value decomposition of $\boldsymbol{Z}^{\mu}$. Clearly,

$$\operatorname{prox}_{\gamma s_\mu}(\boldsymbol{Z}^{\mu}) = D_\mu\big(\boldsymbol{Z}^{\mu}, \gamma f_\mu(\boldsymbol{\Lambda}_k)\big).$$

In order to define the proximal operator of s_0 we use the properties of proximal operators from [104]. Using the fact that $\boldsymbol{P}_k^{1\top}\boldsymbol{P}_k^1 = \boldsymbol{I}_k$ and $\boldsymbol{P}_{k^2}^{-\mu\top}\boldsymbol{P}_{k^2}^{-\mu} = \boldsymbol{I}_{k^2}$, we get

$$\operatorname{prox}_{s_0}(\boldsymbol{Z}^1) = \boldsymbol{Z}^1 + \boldsymbol{P}_k^{1\top}\Big(\boldsymbol{\Omega}\big(\boldsymbol{P}_k^1\boldsymbol{Z}^1\boldsymbol{P}_{k^2}^{-\mu\top} - \boldsymbol{Y}^1, \alpha\gamma\big) - \boldsymbol{P}_k^1\boldsymbol{Z}^1\boldsymbol{P}_{k^2}^{-\mu\top} - \boldsymbol{Y}^1\Big)\boldsymbol{P}_{k^2}^{-\mu},$$

where α is the step size. We represent a matrix $\boldsymbol{U}_{a,b}^{\mu}$ to show the a^{th} iteration, b^{th} function index and matricization along μ^{th} mode, $\boldsymbol{U}_a^{\mu}$ to show the μ^{th} matricization along the a^{th} iteration and 'reshape' to reshape the matrix along the proper mode. Algorithm 1 illustrates the steps needed for the proposed method.

Algorithm 4 Parallel Proximal Algorithm for TRPCAG

```
INPUT: $\boldsymbol{Y}^1$, $f_\mu(\boldsymbol{\Lambda}_k)\ \forall\ \mu$, $\gamma, \epsilon \in (0,1), \alpha \geq 0$.
$\boldsymbol{Z}_{0,0}^1, \cdots, \boldsymbol{Z}_{0,3}^1 \in \mathbb{R}^{k \times k^2}$, all matricized along mode 1.
Set $\boldsymbol{H}_0^1 = \sum_{i=0}^{3}\boldsymbol{Z}_{0,i}^1$
for $j = 1, \cdots, J$ do
    $\boldsymbol{F}_{j,0}^1 = \operatorname{prox}_{s_0}(\boldsymbol{H}_j^1, \alpha)$
    for $\mu = 1, \cdots, 3$ do
        $\boldsymbol{F}_{j,\mu}^1 = reshape(\operatorname{prox}_{(\gamma s_\mu)}(\boldsymbol{H}_j^{\mu}, \gamma(f_\mu(\boldsymbol{\Lambda}_k)))$
    end for
    $\epsilon \leq \beta_j \leq 2 - \epsilon$
    $\boldsymbol{F}_j^1 = \sum_\mu \boldsymbol{F}_{j,\mu}^1$
    for $\mu = 0, \cdots, 3$ do
        $\boldsymbol{Z}_{j+1,\mu}^1 = \boldsymbol{Z}_{j,\mu}^1 + \beta_j(2\boldsymbol{F}_j^1 - \boldsymbol{H}_j^1 - \boldsymbol{F}_{j,\mu}^1)$
    end for
    $\boldsymbol{H}_{j+1}^1 = \boldsymbol{H}_j^1 + \beta_j(\boldsymbol{F}_j^3 - \boldsymbol{H}_j^1)$
end for
OUTPUT: $\boldsymbol{H}_{j+1}^1$
```

Computational Complexity

The computational complexity of graph construction between the rows of a matrix of size $\boldsymbol{Y}^{\mu} \in \mathbb{R}^{n \times n^2}$, using FLANN scales as $\mathcal{O}(n^3\log(n))$ and the eigenvector computation scales as $\mathcal{O}(nk^2)$. Computing the $\operatorname{prox}_{\gamma s_\mu}$ involves the SVD of $\boldsymbol{Z}^{\mu} \in \mathbb{R}^{k \times k^2}$, so its computational complexity scales as $\mathcal{O}(Ik^4)$. The computation of $\operatorname{prox}_{s_0}$ involves matrix multiplications and scales as $\mathcal{O}(In^3k + In^2k^2)$. Thus, the overall complexity of TRPCAG scales as $\mathcal{O}(n^3\log(n) + nk^2 + I(n^3k + n^2k^2))$. However, note that the graph construction and eigenvector decomposition are performed only once and the biggest computational cost is incurred by the matrix multiplication $\mathcal{O}(I(n^3k + n^2k^2))$, which can be parallelized on the GPU. This is a significant reduction as compared to TRPCA, which scales as $\mathcal{O}(In^4)$ and camnnot be parallelized on a GPU because it involves SVD computations in every iteration.

7.3 Properties of our framework

Our proposed method overcomes many limitations of the state-of-the-art methods and possesses the following properties:

1. Convexity: Assuming the graph eigenvalues are sorted in the non-decreasing order, the weights of WNN are a non-decreasing polynomial function of the graph eigenvalues, which, according to Corollary 2 in [124], results in eq. (7.8) being a convex optimization problem.

2. Factorized model: Our model is a combination of factorized tensor based models [115], [72], [73], [116] and the convex tensor robust PCA based models [16]. Thus, it enjoys the speed up of the factorized and the convexity of the robust PCA based models. However, it comes at the cost of explicitly specifying the rank k for every mode.

3. Tensor compression: Note that tensor compression / decomposition in the form of graph eigenvectors and graph core tensor is an implicit step of our model which is missing in the tensor Robust PCA based models.

4. Scalability: We propose to use the fast nearest neighbors search algorithm based library FLANN [60] for the construction of $\mathbb{G}^{\mu}$. For a graph $\mathbb{G}^{\mu}$ constructed from $\boldsymbol{Y}^{\mu} \in \mathbb{R}^{n \times n^2}$, this costs $\mathscr{O}(n^3 \log(n))$ and is parallelizable. We also assume that a fast and parallelizable framework, such as the one proposed in [125] is available for the computation of $\boldsymbol{P}_k^{\mu}$ which costs $\mathscr{O}(n\frac{k^2}{c})$, where c is the number of processors.

Furthermore, as compared to the standard TRPCA problem, which solves for $\boldsymbol{X}^{\mu} \in \mathbb{R}^{n \times n^2}$, eq. 7.11 solves for $\boldsymbol{Z}^{\mu} \in \mathbb{R}^{k \times k^2}$, where $k \ll n$. This results in a significant reduction in the computational complexity of the nuclear norm optimization from $\mathscr{O}(In^4)$ to $\mathscr{O}(Ik^4)$, where I is the number of iterations. Finally, note that the graphs $\mathbb{G}^{\mu}$ and the eigenvectors $\boldsymbol{P}_k^{\mu}$ are fixed throughout the optimization. Thus the overall complexity of TRPCAG is $\mathscr{O}(I(k^4 + n^3 k + n^2 k^2) n^3 \log(n) + nk^2)$, while that for TRPCA is $\mathscr{O}(In^4)$.

5. Structure & Approximation error: Our model implicitly exploits structures in the form of graph regularization of factors along the various modes. However, it also introduces some approximation error. We characterize this in Section 7.4.

7.4 Theoretical Analysis

We started this chapter by arguing the dire need of a scalable framework for low-rank decomposition of tensors, which can exploit intra and inter-mode correlations as well. Recall, in Chapter 4, we presented our GMLSVD framework which describes a method to linearly combine the first few graph eigenvectors to obtain the first few singular vectors for a low-rank matrix or a tensor. Since then, we have presented various approximation problems such as FRPCAG (Chapter 5), CPCA (Chapter 6) and GCTP and TRPCAG (in this chapter), which make use of the graph priors to improve the scalability of low-rank decomposition problems. In this section we present a summary of the entire MLRTG framework in terms of a theoretical analysis which explains the following:

1. How GCTP and TRPCAG make explicit use of the graph eigenvectors to exploit structure along the modes of a tensor.
2. Our intuition behind Graph Tucker Decomposition (GTD), which can be obtained by the GMLSVD method.
3. The effect of the choice of the graph multilinear rank k.
4. Why tensor based approximation problems above result in a better approximation error as compared to FRPCAG.

5. The approximation error of GMLSVD algorithm which was presented in Chapter 4.

We pointed out several state-of-the-art methods in the beginning of this chapter for fast tensor decomposition. Therefore, an obvious question here is: *Besides the speed-up benefit obtained by compressing the tensor by projecting it on graph eigenvectors, why would one use graphs for modeling the low-rankness of the tensor?* The answer to this question is directly linked with the motivation to exploit structure along various modes of a tensor. More specifically, we justify our choice of using graphs for scalability and structure in this framework by linking points 1 and 2 above and quantifying the approximation error of GCTP and TRPCAG. Then, we compare this error to that incurred by FRPCAG. Our theoretical analysis and line of reasoning is similar to that used for FRPCAG in Chapter 5.

We perform our analysis for a simple case of 2D tensors, i.e, matrices of the form $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ and a general linear operator $\boldsymbol{M}$. The results can be extended for high order tensors in a straight-forward manner. For the case of 2D tensor, eq. 7.8, using $\|f_\mu(\boldsymbol{\Lambda}^k)\boldsymbol{Z}\|_* = \|f(\boldsymbol{\Lambda}^1_k)\boldsymbol{Z} f(\boldsymbol{\Lambda}^2_k)\|_*$ can be written as:

$$\min_{\boldsymbol{Z}} \phi\Big(\boldsymbol{M}(\boldsymbol{P}^1_k \boldsymbol{Z} \boldsymbol{P}^{2\top}_k) - \boldsymbol{Y}\Big) + \gamma \Big\| f(\boldsymbol{\Lambda}^1_k)\boldsymbol{Z} f(\boldsymbol{\Lambda}^2_k)\Big\|_* \tag{7.12}$$

Theorem 7.4.1. *For any $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$,*

1. *Let $\boldsymbol{X}^* = \boldsymbol{U}^1 \boldsymbol{\Sigma} \boldsymbol{U}^{2\top}$ be the SVD of $\boldsymbol{X}^*$ and $\boldsymbol{Z}^* = \boldsymbol{A}^1_{k^*} \boldsymbol{R} \boldsymbol{A}^{2\top}_{k^*}$ be the SVD of the GCT $\boldsymbol{Z}^*$. Now, let $\boldsymbol{H}^\mu = \boldsymbol{P}^\mu_{k^*} \boldsymbol{A}^\mu_{k^*}, \forall \mu$, where $\boldsymbol{P}^\mu_{k^*}$ are the Laplacian eigenvectors of $\boldsymbol{X}^*_\mu$, then, $\boldsymbol{H}^\mu = \boldsymbol{U}^\mu$ upto a sign permutation and $\boldsymbol{\Sigma} = \boldsymbol{R}$.*

2. *Solving eq.* (7.12)*, with a $k > k^*$ is equivalent to solving the following factorized graph regularized problem:*

$$\min_{\boldsymbol{H}^1, \boldsymbol{H}^2} \phi(\boldsymbol{M}(\boldsymbol{H}^1 \boldsymbol{H}^{2\top}) - \boldsymbol{Y}) + \gamma_1 \operatorname{tr}(\boldsymbol{H}^{1\top} f(\hat{\boldsymbol{L}}^1)\boldsymbol{H}^1) + \gamma_2 \operatorname{tr}(\boldsymbol{H}^{2\top} f(\hat{\boldsymbol{L}}^2)\boldsymbol{H}^2), \tag{7.13}$$

   where $\boldsymbol{H}^\mu = \boldsymbol{P}^\mu_k \boldsymbol{A}^\mu_k$, $\boldsymbol{X} = \boldsymbol{H}^1 \boldsymbol{H}^{2\top}$ and $f(\hat{\boldsymbol{L}}^\mu) = \boldsymbol{P}^\mu_k f(\boldsymbol{\Lambda}^\mu_k) \boldsymbol{P}^{\mu\top}_k$.

3. *Any solution $\boldsymbol{F}^* = \boldsymbol{H}^{*1} \boldsymbol{H}^{*2\top} \in \mathbb{R}^{n\times n}$ of* (7.13)*, where $\boldsymbol{H}^{*\mu} \in \mathbb{R}^{n\times k}$ and $k > k^*$ with $\gamma_\mu = \gamma / \lambda^\mu_{k^*+1}$ and $\boldsymbol{Y} = \boldsymbol{J}^{*1} \boldsymbol{J}^{*2\top} + \boldsymbol{\eta}$, where $\boldsymbol{\eta} \in \mathbb{R}^{n\times n}$ and $\gamma > 0$ satisfies*

$$\begin{aligned} &\phi(\boldsymbol{M}(\boldsymbol{F}^*) - \boldsymbol{Y}) + \gamma \Big\| \bar{\boldsymbol{P}}^{1\top}_{k^*} \boldsymbol{H}^{*1}\Big\|_F^2 + \gamma \Big\| \bar{\boldsymbol{P}}^{2\top}_{k^*} \boldsymbol{H}^{*2}\Big\|_F^2 \leq \\ &\phi(\boldsymbol{\eta}) + \gamma \left(\|\boldsymbol{J}^{*1}\|_F^2 \frac{f(\lambda^1_{k^*})}{f(\lambda^1_{k^*+1})} + \|\boldsymbol{J}^{*2}\|_F^2 \frac{f(\lambda^2_{k^*})}{f(\lambda^2_{k^*+1})} \right), \end{aligned} \tag{7.14}$$

   where $\lambda^\mu_{k^*}, \lambda^\mu_{k^*+1}$ denote the k^{*th}, k^*+1^{st} eigenvalues of $\boldsymbol{L}^\mu$ and $\bar{\boldsymbol{P}}^{\mu\top}_{k^*} \boldsymbol{H}^{*\mu}$, where $\bar{\boldsymbol{P}}^\mu_{k^*} \in \mathbb{R}^{n\times(k-k^*)}$ denote the projection of the factors on the $(k-k^*)$ complement eigenvectors.

Proof. Please see Appendix A.8.1. □

In the first part, we investigate how accurate are graph eigenvectors in modeling the low-rankness of a tensor. This would involve studying the relationship between the eigenvectors of the graphs and those of the covariance of various modes of the tensor. This provides another jutification of our GMLSVD method, which was proposed in Chapter 4. Our detailed analysis shows that the one can linearly combine the first k graph eigenvectors to approximate the low-rank covariance eigenvectors, where k is such that the eigen-gap $\lambda^\mu_k / \lambda^\mu_{k+1} \approx 0$ exists. Thus, our GCTP and TRPCAG methods can be seen as explicitly specifying a basis $\boldsymbol{P}^\mu_k$ for each of the modes of the tensor, which is further refined while solving these problems. Since we solve a smaller problem of size $k \ll n$ along each mode, our method enjoys scalability and exploits structure via graph regularization, but at the cost of an approximation.

The second part of the above theorem shows that the optimization problem in eq. (7.12) is equivalent to a graph regularization problem. *This verifies our claim since the beginning of the thesis that it is possible to explicitly use the GMLSVD framework and the graph eigenvectors to formulate a structured learning problem.* The graph regularization on the factors $\boldsymbol{H}^\mu_r$ implies that they are smooth w.r.t. graphs $\mathbb{G}^\mu$. Note that this is different from other factorized graph regularized problems [115], [72], [73], [116] as our case involves a low-rank and filtered approximation of the Laplacians $\boldsymbol{L}^\mu$. In simple words, we restrict $\boldsymbol{H}^\mu_r \in span(\boldsymbol{P}^\mu_k)$ rather than the full set of eigenvectors $\boldsymbol{P}^\mu$.

The third part characterizes the approximation error of GCTP and TRPCAG based method in terms of the eigen-gap of Laplacians. The bound eq. 7.14 implies:

$$\|\boldsymbol{E}^*\|_F = \sqrt{\left\|\boldsymbol{P}^{\bar{1}}_{k^*}{}^\top \boldsymbol{H}^{*1}\right\|_F^2 + \left\|\boldsymbol{P}^{\bar{2}}_{k^*}{}^\top \boldsymbol{H}^{*2}\right\|_F^2} \leq \frac{1}{\gamma}\phi(\boldsymbol{\eta}) + \left(\|\boldsymbol{J}^{*1}\|_F^2 \frac{f(\lambda^1_{k^*})^2}{f(\lambda^1_{k^*+1})^2} + \|\boldsymbol{J}^{*2}\|_F^2 \frac{f(\lambda^2_{k^*})^2}{f(\lambda^2_{k^*+1})^2}\right), \tag{7.15}$$

that to recover an optimal low-rank tensor one should have large eigen gaps $\lambda^\mu_{k^*} / \lambda^\mu_{k^*+1}$. This occurs when the rows of the matricized $\mathscr{Y}$ can be clustered into k^* clusters. The smaller $\left\|\boldsymbol{P}^{\bar{1}}_{k^*}{}^\top \boldsymbol{H}^{*1}\right\|_F^2 + \left\|\boldsymbol{P}^{\bar{2}}_{k^*}{}^\top \boldsymbol{H}^{*2}\right\|_F^2$ is, the closer $\boldsymbol{F}^*$ is to $\mathbb{MLT}(\boldsymbol{P}^\mu_k)$. In case one selects a $k > k^*$, the error is characterized by the projection of singular vectors $\boldsymbol{H}^{*\mu}$ on $(k - k^*)$ complement graph eigenvectors $\boldsymbol{P}^{\bar{1}}_{k^*}$. Our experiments show that selecting a $k > k^*$ always leads to a better recovery when the exact value of k^* is not known.

Next, we compare the approximation error of our GCTP and TRPCAG problems with that of FRPCAG and CPCA. To compare, we recall the approximation error of FRPCAG from Chapter 5 as below:

$$\|\boldsymbol{E}^*\|_F = \sqrt{\|\boldsymbol{X}\bar{\boldsymbol{P}}^1_{k^*}\|_F^2 + \|\boldsymbol{P}^{\bar{2}}_{k^*}{}^\top \boldsymbol{X}\|_F^2} \leq \sqrt{\frac{1}{\gamma}\phi(\boldsymbol{\eta}) + \|\boldsymbol{X}^*\|_F^2 \left(\frac{\lambda^1_{k^*}}{\lambda^1_{k^*+1}} + \frac{\lambda^2_{k^*}}{\lambda^2_{k^*+1}}\right)}, \tag{7.16}$$

and the approximation error of CPCA using the alternate decoder below:

$$\|\boldsymbol{E}^*\|_F \leq \frac{1}{\sqrt{2\gamma}} \sqrt{\left(\frac{1}{\gamma}\phi(\boldsymbol{\eta}) + \|\tilde{\boldsymbol{X}}^*\|_F^2 \left(\frac{\tilde{\lambda}^c_{k_c}}{\tilde{\lambda}^c_{k_c+1}} + \frac{\tilde{\lambda}^r_{k_r}}{\tilde{\lambda}_{k_r+1}}\right)\right)} + \frac{1}{\sqrt{2}} \sqrt{\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}} + \frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)} \|\boldsymbol{X}^*\|_F, \tag{7.17}$$

where all the variables with a $\tilde{(\cdot)}$ represent the errors incurred by FRPCAG on the compressed matrix.

Comparing the bounds in eqs. (7.15) and (7.16), it is easy to observe the following:

1. $\phi(\boldsymbol{\eta})$ for eq. (7.15) is less than $\phi(\boldsymbol{\eta})$ in eq. (7.16). This can be explained in terms of the fact

that $\boldsymbol{H}^{*1}\boldsymbol{H}^{*2^\top}$ gives an exact rank k approximation of $\boldsymbol{Y}$. Whereas $\boldsymbol{X}$ in eq. (7.16) gives an approximate rank k matrix. Thus, the noise still exists in the smaller singular values of $\boldsymbol{X}$.

2. For fixed $\boldsymbol{\Lambda}_k^\mu$, $f(\lambda_k^\mu)^2/f(\lambda_{k+1}^\mu)^2 < \lambda_k^\mu/\lambda_{k+1}^\mu$.
3. For a given $k^* < k \ll n$,

$$\left\|\boldsymbol{P}_{k^*}^{\bar{1}}{}^\top \boldsymbol{H}^{*1}\right\|_F^2 + \left\|\boldsymbol{P}_{k^*}^{\bar{2}}{}^\top \boldsymbol{H}^{*2}\right\|_F^2 \leq \|\boldsymbol{X}\bar{\boldsymbol{P}}_{k^*}^1\|_F^2 + \|\boldsymbol{P}_{k^*}^{\bar{2}}{}^\top \boldsymbol{X}\|_F^2.$$

This is intuitive as the matrix $\boldsymbol{P}_{k^*}^{\bar{\mu}} \in \mathbb{R}^{n\times(k-k^*)}$ for eq. (7.15) and $\boldsymbol{P}_{k^*}^{\bar{\mu}} \in \mathbb{R}^{n\times k}$ for eq. (7.16). Thus, clearly $\left\|\boldsymbol{P}_{k^*}^{\bar{1}}{}^\top \boldsymbol{H}^{*1}\right\|_F^2 \leq \|\boldsymbol{X}\bar{\boldsymbol{P}}_{k^*}^1\|_F^2$ and $\left\|\boldsymbol{P}_{k^*}^{\bar{2}}{}^\top \boldsymbol{H}^{*2}\right\|_F^2 \leq \|\boldsymbol{P}_{k^*}^{\bar{2}}{}^\top \boldsymbol{X}\|_F^2$.

In Chapter 5, we pointed out the poor approximation of FRPCAG and attributed it to the manifold shrinking effect. It is important to point out here that TRPCAG and GCTP based problems counter this effect to a large extent as these methods involve a ‘thresholding’ of singular values instead of a penalization. Thus, a larger regularization parameter does not induce an over-shrinking effect on the manifold.

Finally, the error of CPCA (bound in eq. 7.17) is clearly greater than that of FRPCAG due to an added term which quantifies the error of FRPCAG on the compressed matrix.

Thus, to summarize, GCTP and TRPCAG provide a lower approximation error as compared to FRPCAG because of a better control over the singular value penalization, an exact rank k recovery and a smaller set of the graph eigenvectors being used for modeling the singular vectors of the low-rank matrix / tensor. We will study this phenomena in detail in the experimental section.

Finally, as our GMLSVD / GSVD is a special case of GCTP without noise, we can characterize the approximation error of GMLSVD / GSVD (Chapter 4) from the bound (7.14).

Theorem 7.4.2. *For any $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$, any solution $\boldsymbol{F}^* = \boldsymbol{H}^{*1}\boldsymbol{H}^{*2^\top} \in \mathbb{R}^{n\times n}$ of the GMLSVD algorithm, where $\boldsymbol{H}^{*\mu} \in \mathbb{R}^{n\times k}$ and $k > k^*$ with $\gamma_\mu = \gamma/\lambda_{k^*+1}^\mu$ and $\boldsymbol{X}^* = \boldsymbol{J}^{*1}\boldsymbol{J}^{*2^\top}$ and $\gamma > 0$ satisfies*

$$\left\|\boldsymbol{P}_{k^*}^{\bar{1}}{}^\top \boldsymbol{H}^{*1}\right\|_F^2 + \left\|\boldsymbol{P}_{k^*}^{\bar{2}}{}^\top \boldsymbol{H}^{*2}\right\|_F^2 \leq \left(\|\boldsymbol{J}^{*1}\|_F^2 \frac{f(\lambda_{k^*}^1)}{f(\lambda_{k^*+1}^1)} + \|\boldsymbol{J}^{*2}\|_F^2 \frac{f(\lambda_{k^*}^2)}{f(\lambda_{k^*+1}^2)}\right), \tag{7.18}$$

where $\lambda_{k^*}^\mu, \lambda_{k^*+1}^\mu$ denote the k^{*th}, k^*+1^{st} eigenvalues of $\boldsymbol{L}^\mu$ and $\boldsymbol{P}_{k^*}^{\bar{\mu}}{}^\top \boldsymbol{H}^{*\mu}$, where $\boldsymbol{P}_{k^*}^{\bar{\mu}} \in \mathbb{R}^{n\times(k-k^*)}$ denote the projection of the factors on the $(k-k^*)$ complement eigenvectors.

Proof. The proof follows simply by using $\phi(\boldsymbol{\eta}) = 0$ in eq. (7.14), as GMLSVD is a case of GCTP for $\boldsymbol{\eta} = 0$. □

7.5 Experimental Results

We perform extensive experimentation to evaluate GCTP for the recovery of low-rank tensor from Gaussian noise and TRPCAG in the following contexts:

- low-rank and sparse decomposition in terms of static background separation from dynamic sparse foreground

- robust subspace recovery, to study the quality of singular vectors and singular values obtained by TRPCAG.

Datasets: To study the performance of GCTP and TRPCAG, we perform experimentation on 4 artificial and 12 real 2D-4D tensors. All types of low-rank artificial datasets are generated by filtering a randomly generated tensor with the k_{nn} combinatorial Laplacians constructed from its flattened modes (method 3 in Chapter 3). Then, different levels of Gaussian and sparse noise are added to the tensor. The real datasets include Hyperspectral images, EEG, BCI, FMRI, 2D and 3D image and video tensors and 3 point cloud datasets. We report the source, size and dimension of each of the datasets used:

2D video datasets: Three video datasets (900 frames each) are collected from the following source: https://sites.google.com/site/backgroundsubtraction/test-sequences. Each of the frames is vectorized and stacked in the columns of a matrix. The datasets include: 1) Airport lobby: 25344 × 900, 2) Shopping Mall: 81920 × 900, 3) Escalator: 20800 × 900

3D datasets:

1. Functional Magnetic Resonance Imaging (FMRI): 129 × 88 × 1232 (frequency bins, ROIs, time samples), size 1.5GB. This dataset, originally $\boldsymbol{Y} \in \mathbb{R}^{88\times 1232}$ (ROIs, time samples), was obtained from [96] and a Short Time Fourier Transform with a Hamming window of size 32 and 99% overlap was applied along the time domain to transform it into a 3D tensor.
2. Brain Computer Interface (BCI): 513 × 256 × 64 (time, frequency, channels), size 200MB.
3. Hyperspectral face dataset: 542 × 333 × 148 (y-dimension, x-dimension, spectrum), size 250MB, source: https://scien.stanford.edu/index.php/faces-1-meter-viewing-distance/.
4. Hyperspectral landscape dataset: 702 × 1000 × 148 (y-dimension, x-dimension, spectrum), size 650MB, source: https://scien.stanford.edu/index.php/landscapes/.
5. Snowfall video: 1920 × 1080 × 500 (y-dimension, x-dimension, number of frames), size 1.5GB. This video is self made.

3D point cloud datasets: Three 3D datasets ($points \times time \times coordinates$) are collected from the following source: http://research.microsoft.com/en-us/um/redmond/events/geometrycompression/data/default.html. For the purpose of our experiments, we used one of the three coordinates of the tensors only. The datasets include: 1) dancing man: 1502 × 573 × 3, 2) walking dog: 2502 × 59 × 3, and 3) dancing girl 2502 × 720 × 3

4D dataset: EEG dataset: 513 × 128 × 30 × 200, (time, frequency, channels, subjects*trials), size 3GB

Methods: GCTP and TRPCAG are programmed using GSPBox [62], UNLocBox [107] and Tensorlab [126] toolboxes. As GCTP is just an extension of GMLSVD for noisy case, therefore we call it GMLSVD throughout the experimental section. GMLSVD is robust to Gaussian noise, so its performance on 2D and 3D artificial tensors is tested under varying levels of Gaussian noise in the tensor. The 3D version of GMLSVD is compared with the standard MLSVD, while the 2D version - GSVD is compared with the standard SVD.

TRPCAG is robust to sparse noise, therefore, its performance on artificial 2D and 3D tensors is tested under varying levels of sparse noise. For 3D tensors with sparse noise, we compare TRPCAG with

Tensor Robust PCA (TRPCA) [19] and Graph MLSVD (GMLSVD). For the 2D matrix with sparse noise we compare TRPCAG with Robust PCA (RPCA) [1], Robust PCA on Graphs (RPCAG) [99], Fast Robust PCA on Graphs (FRPCAG) [110] and Compressive PCA (CPCA) [127].

Parameters: For all the experiments involving TRPCAG and GMLSVD, the k_{nn} graphs are constructed from the rows of each of the flattened modes of the tensor, using $k_{nn} = 10$, a Gaussian kernel for weighting the edges and the FLANN library [60]. For all the other methods the graphs are constructed as required, using the same parameters as above. Each method has several hyper-parameters which require tuning. For a fair comparison, all the methods are properly tuned for their hyper-parameters (Table 5.2) and best results are reported. Since TRPCAG requires explicit specification of the rank k and for every mode of the tensor, or the size of GCT $\mathscr{X} \in \mathbb{R}^{k\times k\times k}$, we use $k > k^*$, where k^* is the actual rank of a given mode. Furthermore, we tune the regularization parameter $\gamma \in (0, 30)$.

Evaluation Metrics: The metrics used for the evaluation can be divided into two types: 1) quantitative and 2) qualitative. Three different types of quantitative measures are used: 1) normalized ℓ_2 reconstruction error of the tensor $\|\operatorname{vec}(\mathscr{X}) - \operatorname{vec}(\mathscr{X}^*)\|_2 / \|\operatorname{vec}(\mathscr{X}^*)\|_2$, 2) the normalized ℓ_2 reconstruction error of the first k (typically $k = 30$) singular values along mode 1: $\|\sigma^1_{1:k} - \sigma^{1*}_{1:k}\|_2 / \|\sigma^{1*}_{1:k}\|_2$, 3) the subspace angle (in radian) of mode 1 between the 1st five subspace vectors determined by the proposed method and those of the clean tensor: $\arccos|{\boldsymbol{H}^{1}}^{\top}_{5}\boldsymbol{U}^{1}_{5}|$ and 4) the alignment of the singular vectors $|{\boldsymbol{H}^{1}}^{\top}_{5}\boldsymbol{U}^{1}_{5}|$, where $\boldsymbol{H}^1$ and $\boldsymbol{U}^1$ denote the mode 1 singular vectors determined by the proposed method and clean tensor. The closer this matrix is to a diagonal, the better is the alignment. The qualitative measure involves the visual quality of the low-rank components of tensors.

7.5.1 Experiments on GMLSVD

Performance study on artificial datasets: The first two rows of Fig. 7.1 show the performance of GSVD (for 2D) and GMLSVD (for 3D) on artificial tensors of the size 100 and rank 10 in each mode, for varying levels of Gaussian noise ranging from 15dB to 1dB. The three plots show the ℓ_2 reconstruction error of the recovered tensor, the first $k^* = 30$ singular values and the subspace angle of the 1st mode subspace (top 5 vectors), w.r.t to those of the clean tensor. These results are compared with the standard SVD for 2D tensor and standard MLSVD for the 3D tensor. The approximation baseline corresponds to the reconstruction error of GMLSVD for the case of a non-noisy tensor. It is interesting to note from the leftmost plot that the ℓ_2 reconstruction error for GSVD tends to get lower as compared to SVD at higher noise levels (SNR less than 5dB). The middle plot explains this phenomena where one can see that the ℓ_2 error for singular values is significantly lower for GSVD than SVD at higher noise levels. This observation is logical, as for higher levels of noise the high magnitude singular values are also affected. SVD is a simple singular value thresholding method which does not eliminate the effect of noise on high magnitude singular values, whereas GSVD is a smart weighted nuclear norm method which thresholds all the singular values via a function of the graph eigenvalues. This effect is shown in detail in Fig. 7.2 for a 2D matrix corrupted with Gaussian noise (SNR 5dB). On the contrary, the subspace angle (for first 5 vectors) for GSVD is higher than SVD for all the levels of noise. This means that the subspaces of the GSVD are not well aligned with the ones of the clean tensor. However, as shown in the right plot in Fig. 7.2, strong first 7 elements of the diagonal $\operatorname{diag}|{\boldsymbol{H}^{1}}^{\top}_{5}\boldsymbol{U}^{1}_{5}|$ show that the individual subspace vectors of GSVD are very well aligned with those of the clean tensor. This is the primary reason why the ℓ_2 reconstruction error of GSVD is less as compared to SVD at low SNR. At higher SNR, the approximation error of GSVD dominates the error due to noise. Thus, GSVD reveals its true power at low SNR scenarios. Similar observations can be made about GMLSVD from the 2nd row of Fig. 7.1.

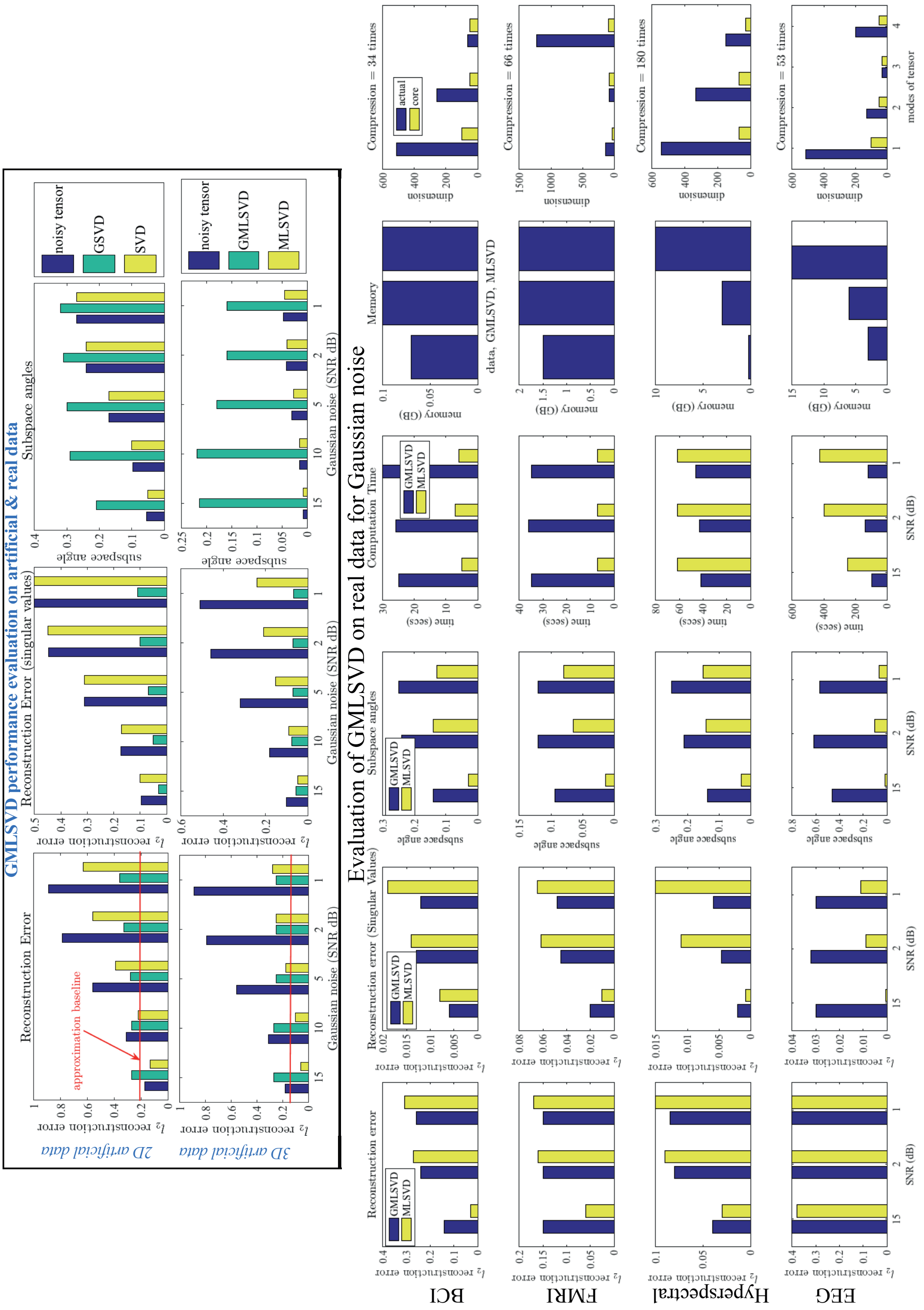


Figure 7.1 – Performance comparison of GSVD and GMLSVD with SVD and MLSVD for 2D artificial (1st row), 3D artificial (2nd row) and 4D real BCI, FMRI, Hyperspectral and EEG tensors (3rd to 6th rows) under different SNR scenarios. Clearly GSVD / GMLSVD outperforms SVD / MLSVD in terms of computation time and error for big and noisy datasets.

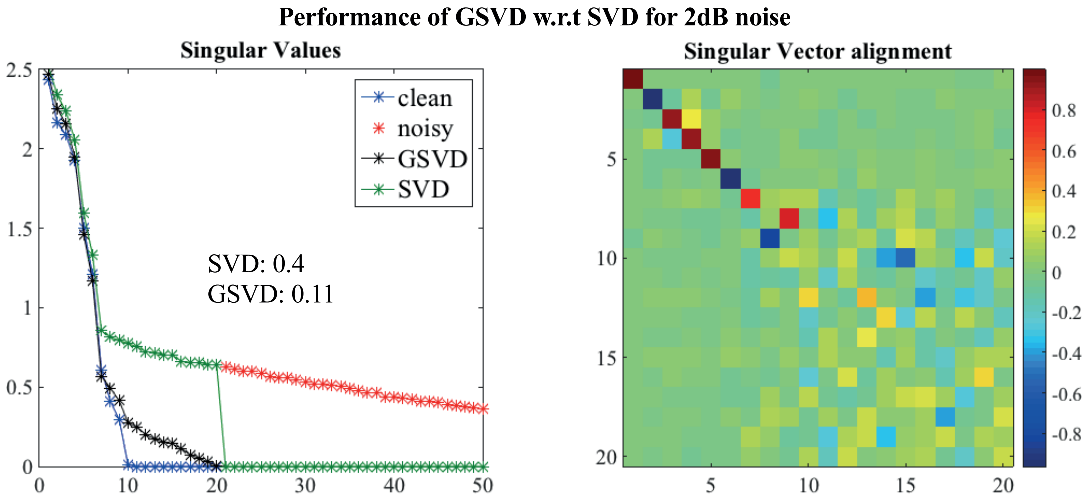


Figure 7.2 – Singular values (left plot) of GSVD, SVD and clean data and singular vector alignment of GSVD and clean data (right plot) for a 2D matrix corrupted with Gaussian noise (SNR 5dB). Clearly GSVD eliminates the effect of noise from lower singular values and aligns the first few singular vectors appropriately.

Time, memory & performance on real datasets: The 3rd to 6th rows of Fig. 7.1 show the results of GMLSVD compared to MLSVD for 4D real BCI, FMRI, Hyperspectral and EEG datasets with varying levels of Gaussian noise and reveal how GMLSVD performs better as compared to MLSVD while requiring less computation time and memory for big datasets in the low SNR regime. The leftmost plots show the ℓ_2 reconstruction error of the recovered tensor with the clean tensor, the 2nd plots show the reconstruction error for 30 singular values, 3rd plots show the subspace angle between the first 5 singular vectors, 4th plot shows the computation time, 5th plot shows the memory requirement (size of original tensor, memory for GMLSVD, memory for MLSVD) and the rightmost plots show the dimension of the tensor and the core along each mode and hence the amount of compression obtained. Another interesting observation is that the computation time and memory requirement for GMLSVD is less as compared to MLSVD for big datasets, such as EEG and hyperspectral images.

Let us consider the example of EEG dataset, which is 3GB in size with dimensions $513 \times 128 \times 30 \times 200$. It is interesting to note that for low SNR scenarios the ℓ_2 reconstruction error of both MLSVD and GMLSVD methods is approximately same. Both methods show a significantly lower error for the singular values (note that the scale is small), whereas GMLSVD's subspaces are less aligned as compared to those of MLSVD. Using a core tensor of size $100 \times 50 \times 30 \times 50$ for this experiment, GMLSVD requires 6GB of memory whereas MLSVD requires 15GB. The rightmost plot of this figure compares the computation time of both methods. *Clearly, GMLSVD wins on the computation time (120 secs) significantly as compared to MLSVD (400 secs).* Note that the BCI and FMRI datasets are smaller in size. Thus, even though the time required for GMLSVD is more than MLSVD for these datasets, GMLSVD scales far better than MLSVD.

For a visualization of the clean, noisy and GMLSVD recovered tensors, their singular values and the alignments of the subspace vectors (mode 1) for FMRI and hyperspectral tensors, please refer to Fig. 7.3. The first row shows the BCI tensor and second row shows the singular values recovered for this tensor for various modes. Clearly, the singular values of the recovered tensor (black) are well aligned with those of the clean tensor (blue). Note, how GMLSVD eliminates the effect of noise from the lower singular values. The rightmost plot shows the alignment between the singular vectors of the recovered and clean tensors. This alignment is calculated as $|\boldsymbol{H}^{1\top}\boldsymbol{U}^1|$, where $\boldsymbol{H}^1$ are the first mode singular vectors obtained via GMLSVD and $\boldsymbol{U}^1$ are the first mode singular vectors of the clean tensor. Again, note that the first few singular vectors are very well aligned. Third and fourth rows show the same type of results for hyperspectral face tensor.

7.5.2 Low-rank & sparse decomposition

Performance study on artificial datasets: The first two rows of Fig. 7.4 show experiments on the 2D and 3D artificial tensors of size 100 and rank 10 along each mode with varying levels of sparse noise.

The 2D version of TRPCAG is compared with state-of-the-art Robust PCA based methods like RPCA, FRPCAG and also with SVD based methods like MLSVD and GSVD. We use $k = 30$ as the initial rank for all the tensor modes for TRPCAG. The three plots in Fig. 7.4 show the ℓ_2 reconstruction error of the recovered tensor, the first $k = 30$ singular values and the subspace angle of the 1st mode subspace (top 5 vectors), w.r.t. those of the clean tensor. It is interesting to note from the leftmost plot that the ℓ_2 reconstruction error for TRPCAG is less as compared to RPCA and significantly less as compared to FRPCAG for higher levels of sparse noise. Similarly to the GMLSVD results, the middle plot explains this phenomena where one can see that the ℓ_2 error for singular values is significantly lower for TRPCAG than RPCA and FRPCAG at higher noise levels, as TRPCAG is an intelligent weighted nuclear norm minimization method. This effect is shown in detail in Fig. 7.5 for a 2D matrix which is corrupted uniformly (10%) with sparse noise of standard deviation 0.1. On the contrary, the subspace angle (for first 5 vectors) for TRPCAG is slightly higher than RPCA and MLSVD for all the levels of noise. This means that the subspaces of the TRPCAG are not well aligned with the ones of the clean tensor. Even though the subspaces are not well aligned as a whole, we will show in Section 7.5.3 that each of the individual subspace vectors of TRPCAG are very well aligned with those of the clean tensor, independently of the other vectors. This is the primary reason why the ℓ_2 reconstruction error of TRPCAG is less as compared to SVD at low SNR. At higher SNR, the approximation error of TRPCAG dominates the error due to noise. Thus, TRPCAG also reveals its true power at low SNR scenarios.

For the 3D tensor (2nd row), TRPCAG is compared with GMLSVD and TRPCA [19]. Interestingly, the performance of TRPCA is always better for this case, even in the presence of high levels of noise. While TRPCA produces the best results, its computation time is orders of magnitude more than TRPCAG (discussed next).

Time & performance on 2D real datasets: The 3rd and 4th rows of Fig. 7.4 present experiments on the 2D real video datasets obtained from an airport lobby (25344 × 900) and shopping mall (81920 × 900), where every frame is vectorized and stacked as the columns of a matrix. The goal is to separate the static low-rank component from the sparse part (moving people) in the video. The results of TRPCAG are compared with RPCA, RPCAG, FRPCAG and CPCA (5,1) with a downsampling factor of 5 along the frames. Clearly, TRPCAG recovers a low-rank which is qualitatively equivalent to the other methods in a time which is 100 times less than RPCA and RPCAG and an order of magnitude less as compared to FRPCAG. Furthermore, TRPCAG requires the same time as sampling-based CPCA but it recovers a better quality low-rank structure as shown by the 3rd and 4th rows.

The performance of TRPCAG is also evident from the point cloud experiments on 3 3D point cloud datasets ($points \times time \times coordinates$). The datasets include: dancing man (1502 × 573 × 3), walking dog: (2502 × 59 × 3) and dancing girl: (2502 × 720 × 3). For the purpose of our experiments, we used one of the three coordinates of the point cloud tensors only. Fig. 7.6 shows the actual point clouds on the

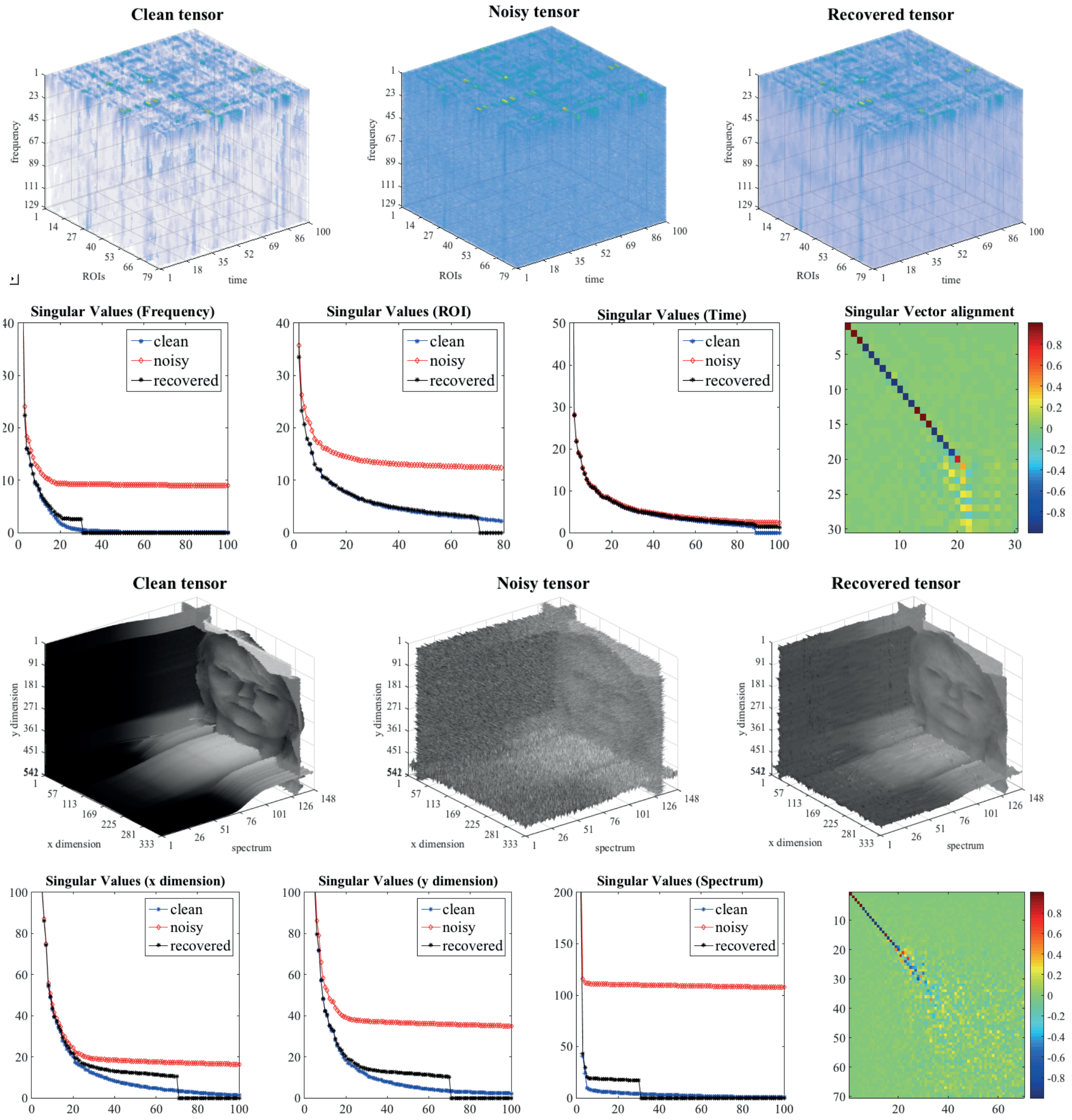


Figure 7.3 – A visualization of the clean, noisy and GMLSVD recovered BCI and hyperspectral face tensors along with a study of the singular values and subspace vectors of various modes. First row shows the BCI tensor and second row shows the singular values recovered for this tensor for various modes. Clearly, the singular values of the recovered tensor (black) are well aligned with those of the clean tensor (blue). Note, how GMLSVD eliminates the effect of noise from the lower singular values. The rightmost plot shows the alignment between the singular vectors of the recovered and clean tensors. This alignment is calculated as $|V_1^\top U_1|$, where V_1 are the first mode singular vectors obtained via GMLSVD and U_1 are the first mode singular vectors of the clean tensor. Again, note that the first few singular vectors are very well aligned. Third and fourth rows show the same type of results for hyperspectral face tensor.

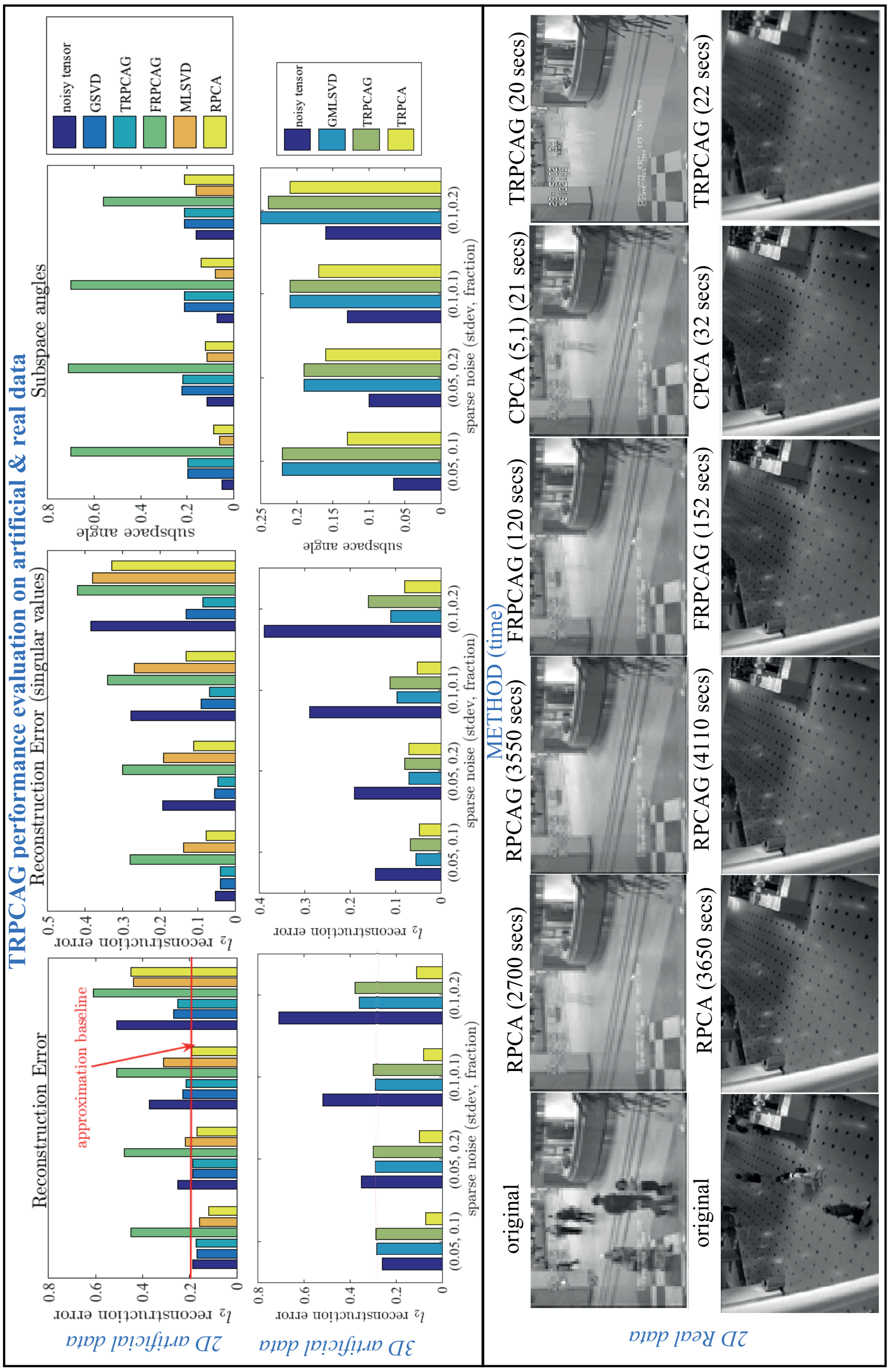


Figure 7.4 – Performance analysis of TRPCAG for 2D artificial (1st row) and 3D artificial (2nd row) with varying levels of sparse noise, 2D airport lobby (3rd row) and 2D shopping mall (4th row) video tensor. The artificial tensors have a size 100 and rank 10 along each mode. A core tensor of size 30 along each mode is used for the artificial tensors and 100 × 50 for the video tensors. The leftmost plots in 1st two rows show the ℓ_2 reconstruction errors, the middle plots show the ℓ_2 reconstruction error for 30 singular values and the right plots show the subspace angles for the 1st five subspace vectors. For the 2D real video datasets, the rightmost frame shows the result obtained via TRPCAG. The computation time for different methods is written on the top of frames. Clearly TRPCAG performs quite well while significantly reducing the computation time.

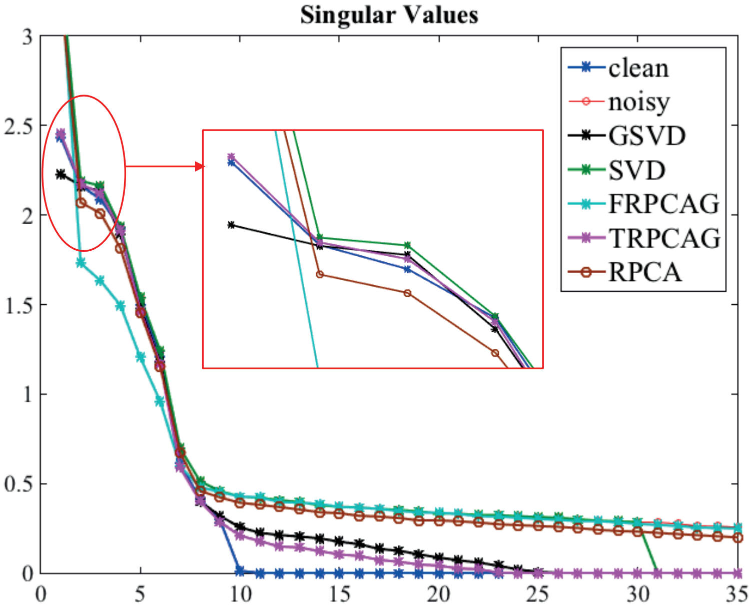


Figure 7.5 – Singular values of GSVD, SVD, FRPCAG, RPCA, TRPCAG and clean data for a 2D matrix which is corrupted uniformly (10%) with sparse noise of standard deviation 0.1. Clearly TRPCAG eliminates the effect of noise from lower singular values. In addition note that under sparse noise, the 1st singular value deviates significantly. This effect is also eliminated via TRPCAG.

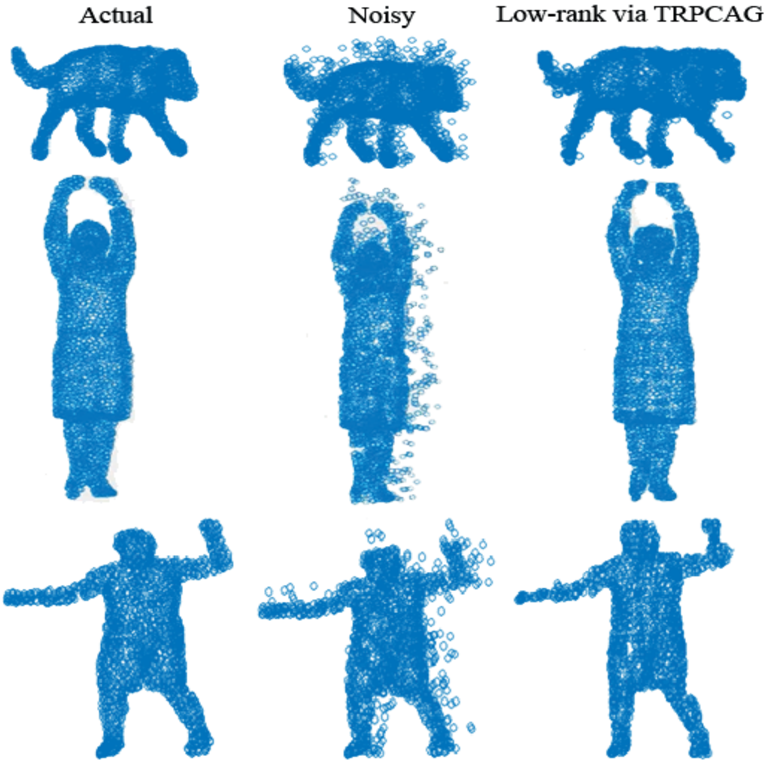


Figure 7.6 – TRPCAG performance for recovering the low-rank point clouds from the sparse noise. Actual point cloud (left), noisy point cloud (middle), recovered via TRPCAG (right).

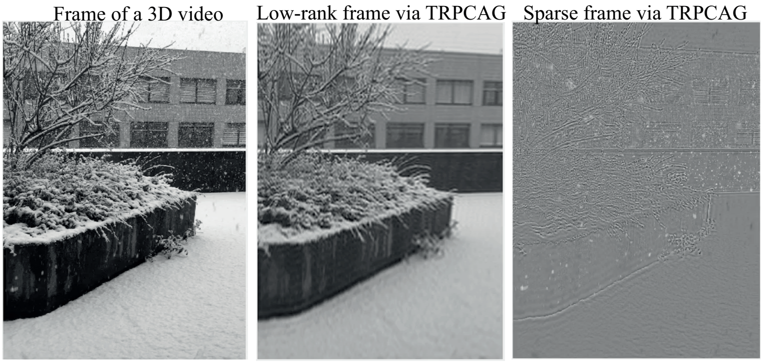


Figure 7.7 – Low-rank recovery for a 3D video of dimension 1920 × 1080 × 500 and size 1.5GB via TRPCAG. Using a core size of 100 × 100 × 50, TRPCAG converged in less than 3 minutes.

left, the noisy ones in the middle and those recovered by TRPCAG on the right. It is clear that TRPCAG performs quite well in recovering the clean low-rank tensor.

Scalability of TRPCAG on 3D video: To show the scalability of TRPCAG as compared to TRPCA, we made a video of snowfall at the campus and tried to separate the snow-fall from the low-rank background via both methods. For this 1.5GB video of dimension 1920 × 1080 × 500, TRPCAG (with core tensor size 100 × 100 × 50) took less than 3 minutes, whereas TRPCA did not converge even in 4 hours. The result obtained via TRPCAG is visualized in Fig. 7.7.

7.5.3 Robust Subspace Recovery

It is imperative to point out that eq. 7.11 implicitly determines the subspace structures (singular vectors) from grossly corrupted tensors. In this section we show that the singular vectors recovered by TRPCAG from the sparse and grossly corrupted tensors align closely with those of the clean tensor. Fig. 7.8 shows the examples of images from 2D ORL, CMUPIE and COIL20 datasets. The leftmost plots show a clean and a sparsely corrupted sample image. Other plots in the 1st row show the 1st singular vector recovered by various low-rank recovery methods, SVD, FRPCAG, RPCA, RPCAG and TRPCAG and the 2nd row shows the alignment of the first 20 singular vectors recovered by these methods $\boldsymbol{H}^1$ with those of the clean tensor $\boldsymbol{U}^1$ in terms of the matrix $|{\boldsymbol{H}^1}^\top_{20}\boldsymbol{U}^1_{20}|$. The closer this matrix is to a diagonal, the better is the alignment. The rightmost plots in each row clearly show that the recovered subspace with TRPCAG is robust to sparse noise.

7.5.4 The effect of parameters on GMLSVD

The GCTP and TRPCAG have three parameters each: 1) the regularization parameter γ, 2) the power q in $f_\mu(\boldsymbol{\Lambda}_k) = \boldsymbol{\Lambda}^{q/2}$ and 3) the graph multilinear rank k. Fig.7.9 (left plot) demonstrates the effect of varying the multilinear rank k on the singular values and the reconstruction error on the performance of GMLSVD. For this experiment, we fix $\gamma = 1$ and use a 100 × 100 matrix of rank 10 and add Gaussian noise to attain a SNR of 5dB. The best error occurs at $k = 35 (> 10)$ as the eigen-gap condition does not hold for the graphs and the dataset is noisy. Note also that the error decreases when k is increased from 10 to 35 and then starts increasing again. This observation motivates the use of a $k > k^*$ ($k^* = 10$ here) in the absence of an eigen-gap, that supports our theoretical finding in Section 7.4. Next, we fix $k = 35$ and repeat the experiments for γ. The best result (middle plot) occurs at $\gamma = 10$. Finally, we fix

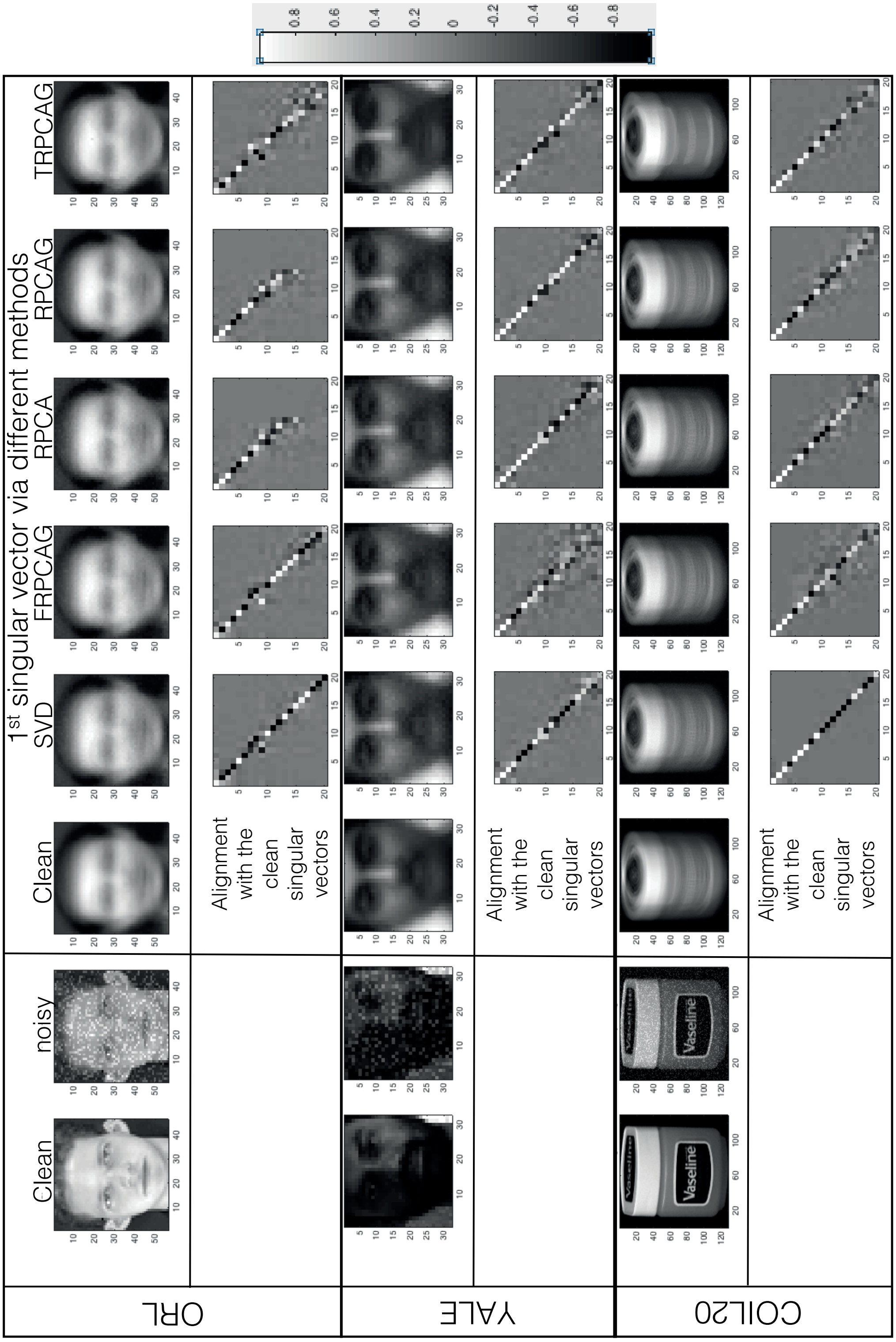


Figure 7.8 – Robust recovery of subspace structures via TRPCAG. The leftmost plots show a clean and sparsely corrupted sample image from various datasets. For each of the datasets, other plots in the 1st row show the 1st singular vector of the clean data and those recovered by various low-rank recovery methods, SVD, FRPCAG, RPCA, RPCAG and TRPCAG and the 2nd row shows the alignment of the 1st 20 singular vecors recovered by these methods with those of the clean tensor.

$k = 35, \gamma = 10$ and run experiments for different values of q. No change in the error and singular values was observed for different values of q. This shows that if the rank k and γ are tuned, the optimization problem is not sensitive to q. Although the graph k_{nn} parameter is not a model parameter of GCTP, it still effects the quality of results to some extent, as shown in the rightmost plot of Fig.7.9. For this experiment we fix $\gamma = 10, k = 35, q = 2$ and use the same dataset with 5dB SNR. The best result occurs for $k_{nn} = 15$ which is not surprising, as this value results in a sparse graph which keeps the most important connections only. Furthermore, note that the singular values of the recovered tensor for $k_{nn} = 15$ are closest to those of the clean tensor.

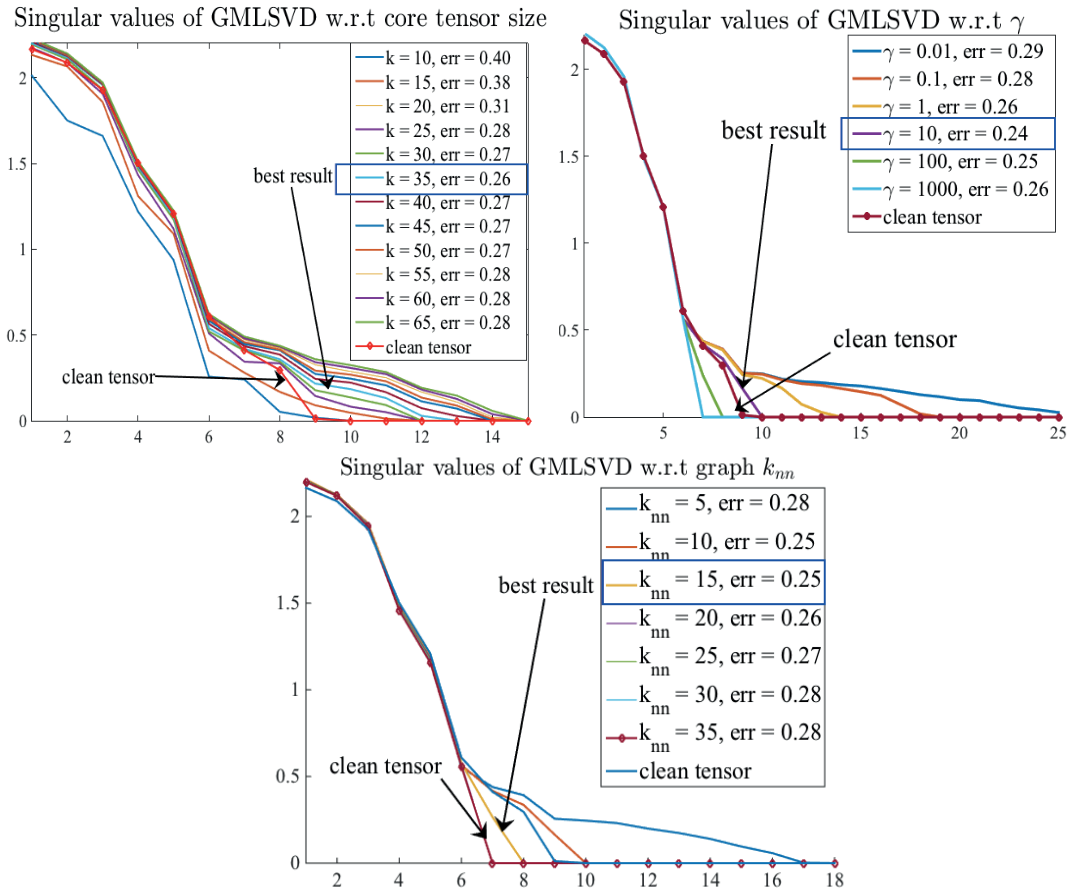


Figure 7.9 – The effect of regularization parameter γ, number of nearest neighbors k_{nn} and the power q in $f_\mu(\boldsymbol{\Lambda}_k)$ (the weights of nuclear norm) on the performance of GMLSVD. The experiments are performed a 100 × 100 matrix of rank 10 with Gaussian noise to attain a SNR of 5dB.

7.6 Conclusion

We present a generalized set of approximation problems for low-rank tensors which makes use for our MLRTG and GMLSVD frameworks. Two of the proposed methods include 'Graph Core Tensor Pursuit' (GCTP) and 'Tensor Robust PCA on Graphs' (TRPCAG), which can recover MLRTG from Gaussian and sparse noise. The GMLSVD algorithm is a special case of GCTP, when the tensor is noiseless. Our theoretical analysis shows that TRPCAG and GCTP perform better as compared to FRPCAG based methods due to their *factorized nature and the thresholding effect of singular values* (rather than a scaling effect like FRPCAG). Our experiments on the recovery of tensor from Gaussian noise and low-rank and sparse decomposition of tensors show the efficiency of our model as compared to the state-of-the-art methods.

8 Discussion & Conclusions

Low-rank matrix and tensor approximation span a wide variety of applications, such as matrix and tensor completion [30], [31], [32], [33], [34], [35], [36] , [37], [38], [39], [40], [41], convolutional neural networks and sparse coding [42], [43], [44], [45], learning the data structure via multiple subspaces [46], latent low-rank representation [47], [48], transfer learning [49], subspace clustering [50], dynamical system modeling [51], such as human behavior modeling [52], dynamic Magnetic Resonance Imaging Reconstructions [53], [54] and many others. However, the state-of-the-art methods suffer from a few important limitations:

- Linearity of the low-rank recovery: As argued throughout the thesis, for many datasets, a non-linear smooth low-dimensional manifold is a better fit as compared to a linear subspace. This is equivalent to exploiting the intrinsic structure in the dataset.
- Susceptibility to sparse noise: An important application in this context is low-rank and sparse decomposition. Examples include static background separation from videos, dynamic activity separation from MRI, FMRI and EEG scans.
- Computational complexity: State-of-the-art low-rank and sparse decomposition methods suffer from high computational complexity due to the use of nuclear norm. For a matrix of size $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$, these methods scale as $\mathcal{O}(Ip^2n)$ and for a tensor of size $\mathscr{Y} \in \mathbb{R}^{n\times n\times n}$, the tensor nuclear norm based methods scale as $\mathcal{O}(In^4)$.

8.1 This thesis in a nutshell

This thesis provides solutions to the above problems by proposing a new low-rank matrix and tensor decomposition model, known as 'Multilinear Low-rank Tensors on Graphs' (MLRTG). MLRTG states that a low-rank tensor can be encoded as a multilinear combination of first k combinatorial Laplacian eigenvectors, which come from k_{nn}-nearest neighbor graphs constructed between the rows of the matricized tensor (Chapter 3).

First, MLRTG is used to define an alternative to the expensive SVD and MLSVD operations for a matrix and tensor. These alternatives are known as Graph SVD and Graph MLSVD (Chapter 4). In simple words, GSVD and GMLSVD are methods which can be used to recover the first k singular vectors of a low-rank tensor by linearly combining the first k graph eigenvectors. This linear combination can be computed in a complexity which is much less as compared to the standard SVD. This framework can be used for the compression of tensors.

Second, GMLSVD framework is used to define a series of convex, scalable and generic approximation problems for recovering low-rank matrices and tensors from noisy measurements. More specifically, we define the following three approximation problems:

1. FRPCAG: Fast Robust PCA on Graphs (Chapter 5)
2. CPCA: Compressive PCA on Graphs (Chapter 6)
3. GCTP: Graph Core Tensor Pursuit (GCTP) and TRPCAG: Tensor Robust PCA on Graphs (Chapter 7)

The first two methods are meant for matrices, whereas GCTP and TRPCAG are meant for tensors. Furthermore, CPCA is just a highly scalable extension of FRPCAG which involves the use of sampling techniques for matrices.

Recall that a nuclear norm prior determines 1) singular vectors and 2) thresholds the singular values iteratively by performing an SVD in every iteration. The main idea behind these approximation problems is given below:

Given a set of fixed, first k graph eigenvectors, one can mimic the behavior of nuclear norm prior by defining graph priors, which:

- *determine singular vectors by linearly combining graph eigenvectors in a complexity much less as compared to the standard SVD, thus avoiding SVD altogether.*
- *threshold or penalize singular values as a function of graph eigenvalues, directly or indirectly.*

More specifically, FRPCAG and CPCA indirectly ‘penalize’ the singular values as a non-linear function of graph eigenvalues, whereas, GCTP and TRPCAG directly ‘threshold’ the singular values by a function of graph eigenvalues. We will compare the two types of methods in detail in the next section.

The entire framework relies on the notion of existence of eigen-gap and approximate stationarity of signals on graphs. This has been explained in detail in Chapter 3. As the eigen-gap assumption does not exist in general for a k_{nn}-nearest neighbor graph, the problems FRPCAG, CPCA, GCTP and TRPCAG result in an approximate low-rank recovery. The approximation error has been characterized in terms of the eigen-gap of the graph Laplacians.

The proposed methods should not only be considered as scalable alternatives to the state-of-the-art methods but also as propositions which exploit the intrinsic structure along various modes of matrix or tensor via graph regularization priors. Our theoretical characterization of this equivalence is included in the various chapters of this thesis.

Finally, to evaluate the performance of each of the proposed models, we performed experiments corresponding to two applications: 1) low-rank and sparse decomposition and 2) clustering of the data in low-dimensional space. The contributions of this thesis are summarized in Fig. 8.1.

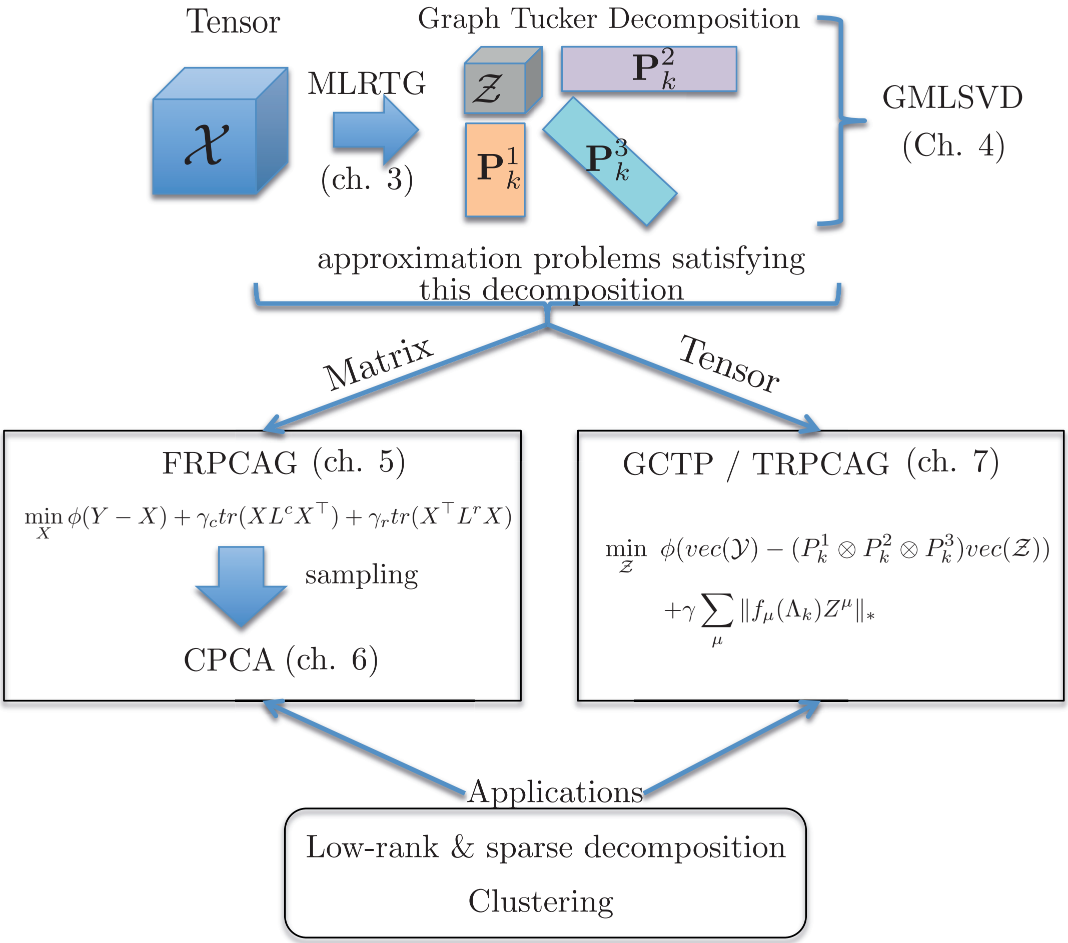

Tensor
Graph Tucker Decomposition
$\mathcal{X}$
MLRTG
(ch. 3)
$\mathcal{Z}$
$\mathbf{P}_k^2$
$\mathbf{P}_k^1$
$\mathbf{P}_k^3$
GMLSVD
(Ch. 4)
approximation problems satisfying this decomposition
Matrix
Tensor
FRPCAG (ch. 5)
$\min_X \phi(Y-X)+\gamma_c tr(XL^cX^\top)+\gamma_r tr(X^\top L^r X)$
sampling
CPCA (ch. 6)
GCTP / TRPCAG (ch. 7)
$\min_{\mathcal{Z}} \phi(vec(\mathcal{Y})-(P_k^1\otimes P_k^2\otimes P_k^3)vec(\mathcal{Z}))$
$+\gamma\sum_\mu \|f_\mu(\Lambda_k)Z^\mu\|_*$
Applications
Low-rank & sparse decomposition
Clustering

Figure 8.1 – This thesis in a nutshell.

8.2 Comparison of approximation problems

In this section we compare the approximation problems FRPCAG / CPCA with GCTP / TRPCAG in detail in terms of a number of metrics. The corresponding problems are stated in Fig. 8.1.

Matrix / tensor: FRPCAG and CPCA are matrix based methods, whereas GCTP and TRPCAG can be used for tensors as well as matrices.

Signal Model: FRPCAG / CPCA determine a low-rank matrix which satisfies the following signal model:

$$\boldsymbol{Y} = \boldsymbol{P}_k^r \boldsymbol{Z}^* \boldsymbol{P}_k^{c\top} + \bar{\boldsymbol{P}}_k^r \bar{\boldsymbol{Z}} \bar{\boldsymbol{P}}_k^{c\top} + \boldsymbol{\eta} \quad \text{where} \quad \|\bar{\boldsymbol{Z}}\|_F \ll \|\boldsymbol{Z}^*\|_F,$$

whereas GCTP / TRPCAG determine a low-rank matrix / matricized tensor which satisfies the following signal model:

$$\boldsymbol{Y}^\mu = \boldsymbol{P}_k^\mu \boldsymbol{Z}^{*\mu} \boldsymbol{P}_{k^2}^{-\mu\top} + \boldsymbol{\eta} \quad \text{where} \quad \|\bar{\boldsymbol{Z}}^\mu\|_F = 0$$

Utilization of GMLSVD: As already explained earlier, our proposed approximation problems fall into two categories: 1) Those which make implicit use of the GMLSVD framework via graph Dirichlet / smoothness priors (explained later). 2) Those which make explicit use of the GMLSVD framework, i.e., the graph eigenvectors of each of the modes of the tensor. The matrix-based approximation problems FRPCAG and CPCA, are meant for the cases when the number of samples n and the features p are very large and one can afford to lose the exactness of the representation for gain in speed. In this context, these methods fall under the first category, as computing the graph eigenvectors and tuning the size of subspaces can be cumbersome. The tensor based method, TRPCAG is used when n is not very large but n^d, where d is the number of modes of the tensor is large. Such methods can afford to explicitly use the eigenvectors as their computation is not expensive and scales linearly with the size of the tensor mode (n).

Singular Value thresholding / penalization: FRPCAG and CPCA result in a low-rank matrix whose singular values are penalized as a non-linear function of the inverse of graph eigenvalues, i.e.,

$$\hat{\sigma}_i \propto \sigma_i \frac{1}{(1+\gamma_r \lambda_i^r)(1+\gamma_c \lambda_i^c)}.$$

Furthermore, this control is not explicit, as the trace terms implicitly perform this penalization. However, in GCTP and TRPCAG, the singular values are 'thresholded' as a non-linear function of the graph eigenvalues as follows:

$$\hat{\boldsymbol{\Sigma}} = \operatorname{sgn}(\boldsymbol{\Sigma}) \max\{|\Sigma| - \gamma f(\Lambda_k), 0\}.$$

This thresholding is explicit and hence provides a better control of singular values in these methods as compared to FRPCAG / CPCA.

Rank of the resultant matrix / tensor: Due to indirect non-linear penalization of the singular values in FRPCAG and CPCA, the resultant matrix is only approximately low-rank. As the lower singular value are scaled (not thresholded), they never reduce to zero. Thus, one needs to do a final SVD step to manually threshold the singular values of the resultant low-rank matrix $\boldsymbol{X}$. On the other hand, the matrix or tensor obtained from GCTP / TRPCAG can only be at most rank k along every mode. This results in an exact rank k matrix or tensor.

Computational Complexity: Excluding the cost of graph construction, FRPCAG method scales linearly with the number of elements in the matrix. Thus, for a matrix $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$, FRPCAG scales as $\mathcal{O}(Inpk_{nn})$.

As CPCA is an accelerated version of FRPCAG, it scales as $\mathcal{O}(nkO_l k_{nn} + \rho_c \rho_r^2 + I\rho_r \rho_c k_{nn})$, where $\rho_r < p, \rho_c < n$, O_l is the order of the Lancoz approximation, k_{nn} the number of nearest neighbors for graph and k the rank of matrix $\boldsymbol{X}$. GCTP, for a matrix $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$ scale as $\mathcal{O}(Ik^3)$ and TRPCAG as $\mathcal{O}(I(pnk + nk^2 + k^3) + nk^2)$.

Parallel Implementation on GPU: FRPCAG involves only matrix multiplications, subtractions and additions in every iteration. Thus, the computations in every iteration of the algorithm can be easily parallelized with a GPU implementation. GCTP and TRPCAG, on the other hand, involve SVD of the GCT in every iteration which cannot benefit from a GPU implementation.

Approximation Error: Due to an indirect control over the singular values and the scaling effect of penalization, FRPCAG results in a worse approximation error as compared to GCTP / TRPCAG which offer a ‘thresholding’ effect instead. It is intuitive, since FRPCAG results in an approximate low-rank matrix, whereas, GCTP and TRPCAG result in an exact rank k matrix. Finally, the approximation error incurred by CPCA is even worse than FRPCAG, as the sampling of low-rank matrix across the rows and columns may result in a loss of information. Moreover, the approximation error of CPCA is the sum of errors incurred by solving the FRPCAG problem on the sampled data matrix and the error incurred by the approximate decoder.

Model Parameters: FRPCAG has only two model parameters γ_r, γ_c which can be tuned via cross-validation. CPCA also has only two parameters for solving FRPCAG on the sub-sampled matrix, whereas the approximate decode stage of CPCA is parameter free. However, GCTP and TRPCAG have several parameters, such as the regularization parameter γ, the power q of the non-linear function $f(\cdot)$ and the graph multilinear rank k. Even though our experiments in Chapter 7 show that q does not effect the performance of these methods once the other parameters are properly tuned, choosing an appropriate k can be a cumbersome task.

Problem Ingredients: Both types of methods require the computation of graphs. GCTP and TRPCAG, however, also require the computation of graph eigenvectors and eigenvalues.

8.3 Discussion on the choice of appropriate method

We point out that our motivation, in this thesis, was not to design a framework that recovers an exact low-rank representation. In fact the goal was to present a system that works well under a broad range of conditions in contrast to those required for exact recovery [1]. Thus, we target an approximate and fast recovery under a broad range of conditions and data types. Nevertheless, if the quantitative error of the low-rank recovery is under concern then one might resort to Robust PCA based methods or their fast alternatives which involve randomly projecting the dataset on a low-dimensional space. However, if one is concerned about recovering a good enough low-rank representation for big datasets, then it is possible to choose from one of the techniques presented in this thesis.

Given a low-rank and sparse decomposition task, one can choose from FRPCAG, CPCA or TRPCAG, depending on if the dataset is a matrix or tensor, its size, approximation error tolerance and the number of model parameters which can be tuned easily. *Our general recommendation is that TRPCAG is used for tensors and FRPCAG and CPCA for matrices.* However several exceptions to this recommendation exist depending on the type of dataset. We present some general guidelines for the choice of an appropriate method.

FRPCAG: This method is suitable if the dataset is a matrix of size $p \times n$, where $p < n$ and one wants to make use of a GPU implementation. Furthermore, the method is more suitable when one is concerned only about the visual quality of the low-rank recovery and the dataset is not highly corrupted by noise, as the method only gives an approximate low-rank solution. Typical examples include background separation from dynamic foreground in videos, where every frame of the video can be vectorized and stacked in the columns of a matrix. The method incurs a higher approximation error in the presence of large amounts of noise, therefore, care should be taken to properly tune the parameters and avoid over-regularization (manifold shrinking effect) in such scenarios. Finally, in terms of computational complexity, the method is more suitable when both n and p are very large and an approximate low-rank recovery is sufficient. The method involves a final SVD step if an exact low-rank representation is desired, which can be expensive $\mathcal{O}(n^3)$ if both n and p are large and $p \rightarrow n$.

CPCA: CPCA is an accelerated version of FRPCAG, where the low-rank recovery problem is solved on a sub-sampled data matrix and the full low-rank matrix decoded at the end via an approximate but parameter free decoder. Therefore, it is more suitable when speed is of greater concern than the approximation error of the method, i.e., for the cases, when both n and p are large and $p \rightarrow n$. In fact CPCA is less suitable for the cases where n and p are small because the sampling on smaller matrices can result in a loss of information across the rows and columns of a matrix. This was also evident from our clustering experiments in Chapter 6.

TRPCAG: This method can be used for tensors as well as matrices. However, in terms of computational complexity, it is more useful when the dataset is in the form of a tensor and the number of samples in each of the modes of the tensor, i.e., n is not very large and the rank is very low, i.e., $r \ll n$ (for computational complexity considerations). Furthermore, the method is more useful when the modes of a tensor are such that it is meaningful to construct graphs between the rows of each of the modes. For example, it is intuitive to construct graphs for the modes of an FMRI tensor (time, voxels, subjects) but it is less intuitive to construct graphs for the modes of a video tensor (x-dimension, y-dimension of the frame, time). In the video example, a graph among the rows and columns of a video frame is less intuitive as compared to a graph between the patches of a frame or a graph among the different frames of the video in time. For the video example, therefore, it is preferable to vectorize the frames and stack them in the columns of a matrix and then either use FRPCAG or the matrix version of TRPCAG. A caution should be taken when using TRPCAG as it requires tuning more parameters than FRPCAG and CPCA.

8.4 Short-comings & Future Directions

The proposed methods provide scalable alternatives to the standard matrix and tensor low-rank decomposition methods which exists in the literature. However, the methods suffer from a number of short-comings whose remedy will be the focus of our future work.

Manifold shrinking effect of FRPCAG and CPCA: In Chapter 5 we observed the manifold shrinking effect of FRPCAG which is a cause of the undesired and excessive penalization of singular values. Generalized graph filters were proposed to overcome this effect but our experiments, which we do not present in this thesis, show that this effect is not significantly reduced. Although our TRPCAG based method provides an alternative to reduce this effect but TRPCAG has its own short-comings which will be discussed next.

Parameters of TRPCAG: TRPCAG improves over the manifold shrinking effect of FRPCAG method, however, it requires tuning many parameters, which involve the graph multilinear rank, regularization parameter and the power of the polynomial function. Therefore, care needs to be taken to appropriately tune the parameters. Our future work will focus on defining automated tuning procedures for some of these parameters.

The approximation error and its dependence on graph quality: The approximation error of all the methods proposed in this thesis depend on the eigen-gap of the Laplacians. We pointed out in Chapter 3 that it is possible to construct a graph with a large eigen-gap at the k^{th} eigenvalue if the data is clusterable into k-clusters and these clusters are disconnected but possess strong intra-connections. However, when a k_{nn}-nearest neighbor graph is constructed using FLANN, this property does not hold, which results in a poor approximation of these methods. While graph learning is not a focus of this thesis, methods which construct graphs possessing such properties are a requirement of our framework. Therefore, the future work should focus on graph learning methods in this respect. State-of-the-art graph learning frameworks, such as the one proposed by Kalofolias et. al. [92] do not provide us with a solution to counter the above mentioned problem. Ideally, one would like to construct a graph by inverting the covariance matrix, however, such a method would require an expensive inversion process. Moreover, the covariance of a low-rank matrix can be ill-conditioned and the inversion might not be possible. Another limitation of such a method is that the graph would loose its local smoothness property which results from the sparsity of the graph edges, as inverting a covariance results in a dense graph. Finally, a plethora of methods exist which propose to learn sparse precision matrices from a covariance matrix [128], [129], [130]. However, such methods do not learn a Laplacian which renders them useless for the graph signal processing framework. Recently, Pavez et. al. [93] proposed to learn a generalized Laplacian from a covariance matrix, however, the graph learned by this method does not possess the eigen-gap property. Therefore, there is a dire need to develop a graph learning framework which possesses the desired properties.

Error Metrics: Our proposed framework is meant to recover non-linear low-rank manifolds, as explained in Chapter 1. However, throughout the thesis, we compare our methods to the linear ones. This is potentially due to a lack of metrics for evaluating the performance of non-linear low-rank recovery and non-linear low-rank datasets. Thus, new benchmark non-linear low-rank datasets need to be defined to evaluate the performance of all the methods in a fair manner. Finally, we compare the alignment of the subspaces recovered by our methods with those recovered by the linear ones. Our experiments in Chapter 7 show that the angle between the subspaces is big even when the individual subspace atoms are quite well aligned. Therefore, we need new metrics to measure the alignment between subspaces. Such a metric would also help obtain tighter bounds on the approximation error for all the proposed methods, as compared to the current characterization in terms of eigen-gaps.

Online Methods: A primary limitation of the proposed framework is that it is offline. Thus, it does not propose how new data samples can be incorporated into the dimensionality reduction and low-rank feature extraction framework without re-training the model on the entire dataset. A bottleneck for such methods is the online update of the graphs based on the arrival of new data samples. One needs efficient graph update methods which can re-run the nearest neighbor algorithm on a subset of the dataset based on the new data sample. Furthermore, mechanisms are needed to update the graph eigenvectors for the tensor framework and assign low-rank features to ensure smooth learning of the entire low-rank matrix $\boldsymbol{X}$ on the row and column graphs, as guided by the graph regularization terms in the matrix based framework.

8.5 Further Applications

The proposed approximation problems can be extended for a number of applications. Here, we discuss a few of them for which we have done some initial work. Extensive evaluation of these methods constitutes the future work.

8.5.1 Matrix Completion

Matrix completion plays an important role in recommendation systems, where the goal is to complete a matrix of movie recommendation for a set of users [33], [131]. Our FRPCAG and TRPCAG based problems can be extended for this purpose in a straight-forward manner, given that the graphs of users and movies are available using some external information. Similar users would like similar movies and similar movies are liked by similar users, therefore, it is possible to extend our graph based methods. Let $\boldsymbol{M}$ be a mask binary mask operator for the user-movie matrix, where $\boldsymbol{M}(i,j)=1$ if the recommendation exists and 0 otherwise, then one can extend FRPCAG as follows:

$$\min_{\boldsymbol{X}} \|\boldsymbol{M}\circ(\boldsymbol{Y}-\boldsymbol{X})\|_F^2+\gamma_r \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X})+\gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c\boldsymbol{X}^\top),$$

where $\circ$ denotes the hadamard product. Similarly, the tensor based inverse problems can be proposed for this purpose:

$$\min_{\boldsymbol{Z}} \|\boldsymbol{M}\circ(\boldsymbol{Y}-\boldsymbol{P}_k^r\boldsymbol{Z}\boldsymbol{P}_k^{c\top})\|_F^2+\gamma\|f(\boldsymbol{\Lambda}^k)\boldsymbol{Z}\|_*. \tag{8.1}$$

We use the tensor based method above to present our initial experiments on the completion of a small artificial matrix of size $\boldsymbol{Y}\in\mathbb{R}^{100\times 200}$, possessing a community structure with noise. This dataset has been used in the work of Kalofolias et. al. [33]. We randomly remove 60% of the entries from the matrix and construct k_{nn}-graphs across the rows and columns using $k_{nn}=10$ and the following rule for the edge weights:

$$\boldsymbol{W}(i,j)=\exp\Big(-\frac{B_{ij}\circ(\boldsymbol{Y}(i)-\boldsymbol{Y}(j))}{\sigma^2}\Big),$$

where B_{ij} is a binary operator which has 1 for the entries which are known both in $\boldsymbol{Y}(i)$ and $\boldsymbol{Y}(j)$. We use the method of eq. 8.1 with $\gamma=1$ and $q=2$ (the power in function f) and $k=20$. Finally, we measure the root mean squared error (RMSE) which turns out to be 0.87 for this experiment. The original matrix, the incomplete and the one recovered using our method are presented in Fig. 8.2.

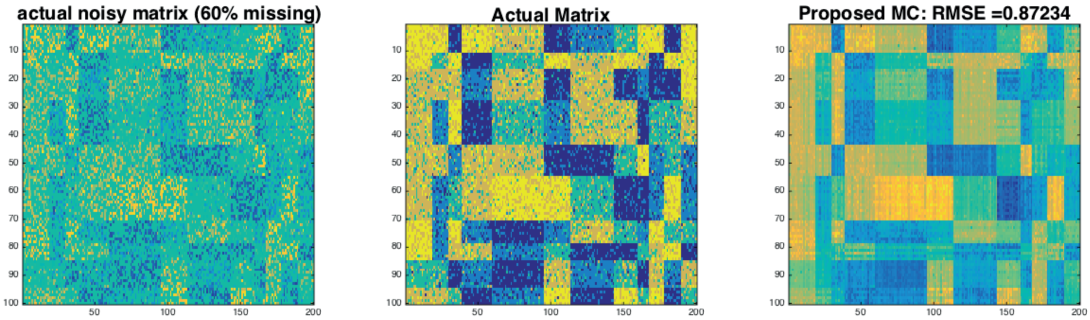


Figure 8.2 – Initial results for our matrix completion approach of eq. 8.1.

While the above results just correspond to an initial experiment, there are several issues which have to be dealt before such a method can be used for real data. The first issue is that the above matrix has only 60% missing entries and the graph construction using the above mentioned weighting rule led to a connected graph. However, for real applications, the fraction of missing entries can be much more than 60%. Thus, the graphs constructed using this strategy could result in disconnected nodes. One might need external features for graph construction as used in the experiments of [34]. Further experimentation should be done on highly sparse matrices for which the graphs are constructed using external information and the community structure does not hold.

The proposed method can also be directly extended for tensor completion. Tensor completion, is another widely studied problem. We refer the readers to the review paper by Evgeny et al. [41] for a detailed survey of exisiting techniques while we briefly shed light on a few of them which resemble in their anatomy with the tensor robust PCA. One of the first works in this field were done by Liu et al. [37] and Gandy et al. [35] using the tensor nuclear norm minimization. Several variants for solving the problem have been proposed since then. For example, the authors of [18] propose an iterative hard-thresholding based method to fit the factors of a Tucker Decomposition and those of [38] propose to do so in the hierarchical Tucker format. In contrast to the Tucker Decomposition, [36] proposes an optimization scheme to fit the factors of CP decomposition for incomplete data. Kressner et al. [132] propose a technique which exploits the manifold structure of the tensors of fixed multilinear rank [133]. The authors of [40] propose a Gauss-Seidel based method to minimize the multilinear rank in an attempt to reduce the computational cost of the standard approaches. Another method which performs the factorization of all the matricized modes of a tensor in parallel has been proposed in [39]. A detailed theoretical analysis for this type of problems has been carried out in [134].

8.5.2 Tensor non-linear embedding & clustering

Another important application aspect corresponds to the tensor counterparts of non-linear embedding methods, such as Laplacian Eigenmaps (LE) [2], Locality Preserving Projections (LPP) [23] and Locally Linear Embeddings (LLE) [24]. One of the first extensions for tensors appeared in [25] as Tensor Subspace Analysis. Such methods can then be used for applications such as tensor clustering in the low-dimensional space.

Throughout the thesis, we show several examples of clustering for our matrix based model, i.e., FRPCAG, however, we do not present any tensor clustering methods. Tensor clustering is one of the potential applications of our proposed tensor based framework. Since, our GCT has the interpretation of a low-dimensional tensor embedding in a product of low-dimensional manifolds, the proposed framework can be extended for this application. In order to explain this we consider the example of clustering along the time dimension of an EEG tensor (time, frequency, channels). Let us assume that we want to cluster chunks of activities along the time axis, i.e., we have access to sub-tensors along the time dimension, where, each 3D sub-tensor consists of a small portion of time axis and full frequency and channel axes. Let us further assume that each of the sub-tensors can be clustered into one of the possible c activity types (clusters). Since each of the subtensors can have large number of samples along each of the modes, it will be computationally unfeasible and counter-intuitive to cluster by using a distance metric between the sub-tensors. Therefore, we recommend to extract low-dimensional features from each of these sub-tensors and use these features for clustering. Our features here correspond to the GCT from each of the sub-tensors, which is rich in information and has nice interpretation of a low-dimensional embedding on a product of manifolds and joint k-GFT. In this respect, we propose the following algorithm.

- Define the clusterable mode of the tensor, i.e., a mode along which the clustering needs to be performed, e.g., time for our EEG example. Declare all the other modes as 'global modes', i.e., the modes which are not clusterable.
- Construct k_{nn}-graphs along the global modes by matricizing along them and determine the first k eigenvectors.
- Divide the tensor into sub-tensors along the clusterable mode.
- Project each of the sub-tensors on the eigenvectors.
- Use these low-dimensional features in a k-means or spectral clustering algorithm.

Fig. 8.3 shows the result of our clustering algorithm on the 4 classes of COIL20 dataset. Each of the images of the dataset are arranged slice by slice to form a tensor. We declare the x and y dimensions of the tensor as the global modes and construct k_{nn}-graphs with $k_{nn} = 10$ along these modes and then determine the first $k = 10$ eigenvectors. Each of the slices needs to be clustered into one of the 4 classes, therefore, we project each slice on the k-eigenvectors of x and y dimensions of the images and then perform spectral clustering on these low-dimensional features. Fig. 8.3 shows the adjacency matrices computed from the low-dimensional features (left) and the original images (right). We obtain a clustering error of 10% as compared to the error of 28% with the original images. The proposed method should be extended, improved and tested on many other tensor datasets.

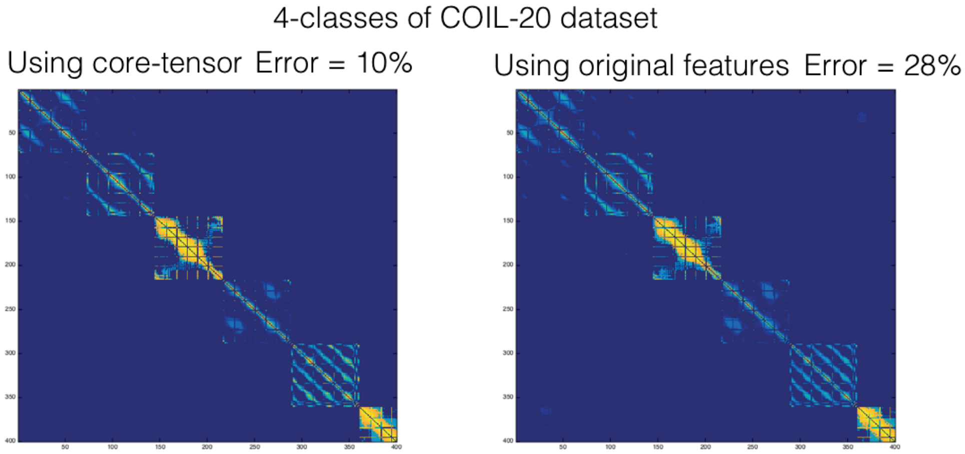
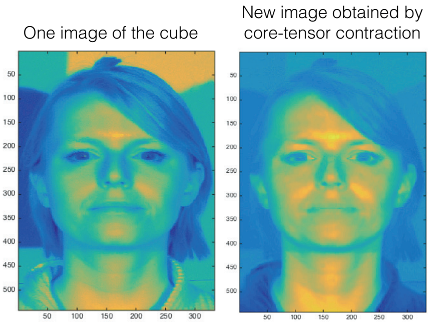


Figure 8.3 – A result of tensor clustering algorithm on the COIL20 dataset.

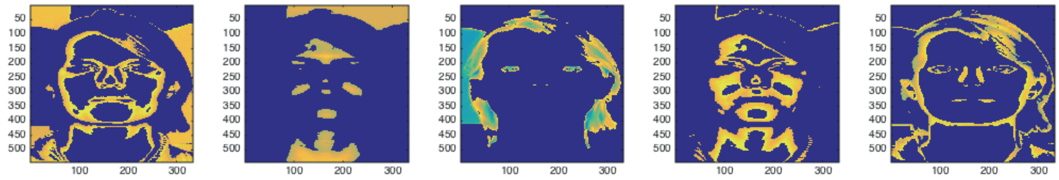
8.5.3 Tensor Segmentation

Tensor segmentation is another important application for hyperspectral tensors. For such cases it is important to merge all the slices of the tensor and perform segmentation on the resultant slice or perform segmentation on each of the slices separately. We propose a simple core tensor contraction based method for clustering the slices of a hyperspectral image tensor and present our results on two examples.

The hyperspectral image tensor segmentation involves clustering on the joint $x - y$ image dimension, therefore, information along all the frequency slices needs to be merged. We propose to first decompose the tensor via our GTD method using $k_{nn} = 10$ nearest neighbor graphs and $k = 20$ graph eigenvectors along each of the modes. Then, we propose to perform a tensor contraction (summing all the slices) along the frequency mode of the GCT. This results in the contracted GCT of size $k \times k$. Finally we recover a single slice by taking the outer-product of the contracted GCT with the eigen-vectors of the x and y dimensions and perform k-means on the pixels of this slice.



Segmentation on the contraction of actual tensor



Segmentation on the new image: contraction of core tensor

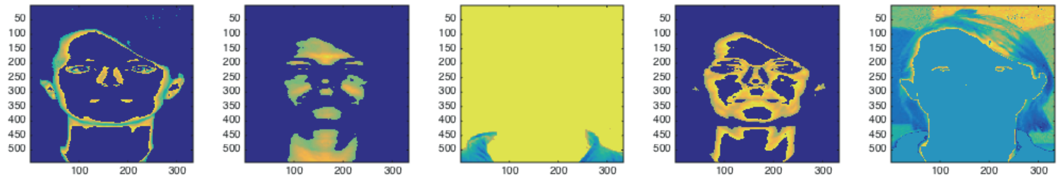


Figure 8.4 – Segmentation result on the hyperspectral face tensor from Stanford database.

Fig. 8.4 shows the results of segmentation of a Hyperspectral face tensor into 5 clusters using our method and a simple contraction of all the tensor slices. The first row shows a single slice of the tensor (left) and the slice obtained by our method (via GCT contraction). The 2nd and 3rd rows compare the results of segmentation using the two approaches. It is quite clear to see that our method results in more meaningful clusters as compared to the other method. For example, note that one of the clusters using our method (3rd figure in the middle) consists of the collar portion of the tensor only, whereas this part is mixed with other clusters for the other method. Fig. 8.5 shows results of segmentation on the hyperspectral tensor of the Geneva city into 4 clusters.

8.5.4 Low-Rank Recovery from Random Projections

Throughout the CPCA work (Chapter 6) we assumed that the graphs $\mathbb{G}^c$ and $\mathbb{G}^r$ for the complete data matrix $\boldsymbol{Y}$ are either available or can be constructed directly from $\boldsymbol{Y}$ itself using the standard graph construction algorithms. This is a reasonable assumption if one wants to reduce the computational burden by downsampling on the datasets. However, often the complete data matrix $\boldsymbol{Y}$ is not available

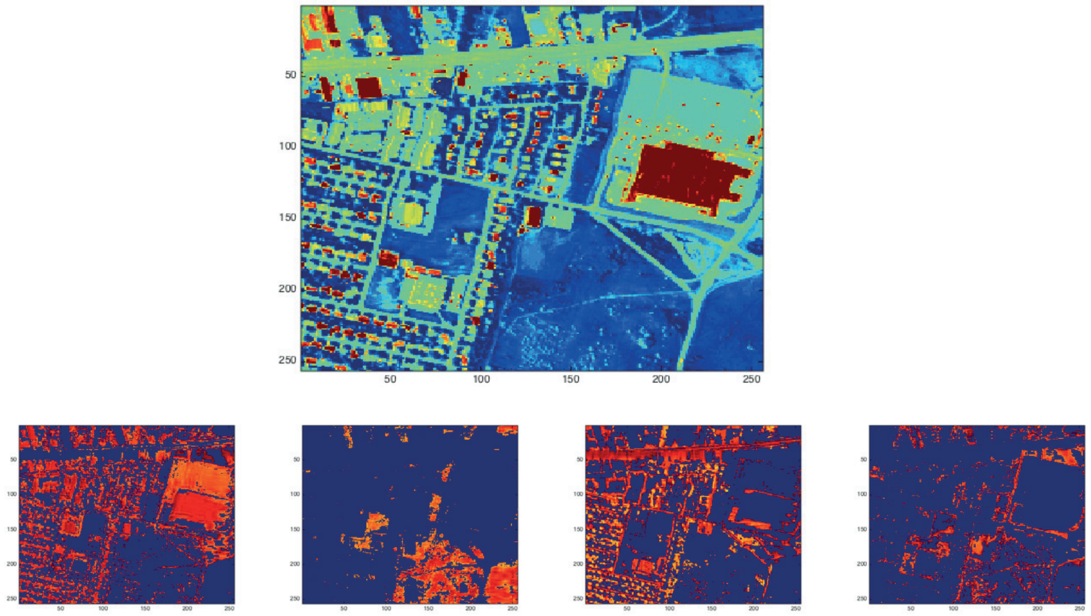


Figure 8.5 – Segmentation result on the Geneva hyperspectral tensor.

and the goal is to obtain an estimate of $\boldsymbol{Y}$ from some side information. Typical examples include Magnetic Resonance Imaging (MRI), Computational Tomography (CT) and Electron Tomography (ET) where one only has access to the projections b of $\boldsymbol{Y}$ acquired through a known projection operator $\boldsymbol{A}$. The purpose here is not to reduce the computational burden but to acquire a good enough estimate of $\boldsymbol{Y}$ from $\boldsymbol{b}$. Furthermore, for such applications, there is no notion of row or column projection / sampling operators. Nevertheless, one might want to exploit the row and column smoothness assumption for the purpose of reconstruction. While, this is not a significant part of our current work, it is still an obvious open question and the answer comes from an extension of this work. Therefore, to be complete, we propose a framework for such problems which might require a low-rank reconstruction from a few projections. It is important to emphasize though that the goal is not to compare and evaluate the performance rigorously with the state-of-the-art. In fact we mention this here just to give a flavour of how the current framework can be extended for such problems.

Assume that a CT sample, for example, a Shepp-Logan phantom of the size $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ needs to be reconstructed from its projections $\boldsymbol{b} \in \mathbb{R}^m$, obtained via a line projection matrix $\boldsymbol{A} \in \mathbb{R}^{m \times np}$. Thus, $\boldsymbol{b} = \boldsymbol{A}\text{vec}(\boldsymbol{X}) + \boldsymbol{e}$, where $\boldsymbol{e} \in \mathbb{R}^m$ models the noise in the projections. We propose to reconstruct the sample $\boldsymbol{X}$ by solving the following optimization problem:

$$\min_{\boldsymbol{X}} \|\boldsymbol{A}\text{vec}(\boldsymbol{X}) - \boldsymbol{b}\|_2^2 + \gamma_r \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{L}^r \boldsymbol{X}) + \gamma_c \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^c \boldsymbol{X}^\top), \tag{8.2}$$

where $\boldsymbol{L}^r, \boldsymbol{L}^c$ are the row and column graph Laplacians between the rows and columns of $\boldsymbol{X}$. Since, these graphs are not available in the beginning, one can obtain an initial estimate of $\boldsymbol{X}$ by running a standard compressed sensing problem for a few iterations and then construct these graphs from this estimate. The estimated graphs can also be improved after every few iterations from the more refined $\boldsymbol{X}$.

Fig. 8.6 shows the reconstruction of a 64×64 Modified Shepp-Logan phantom from 20% projections using eq.(8.2). The initial estimate of the graphs $\mathbb{G}^r, \mathbb{G}^c$ between the rows and columns of the phantom

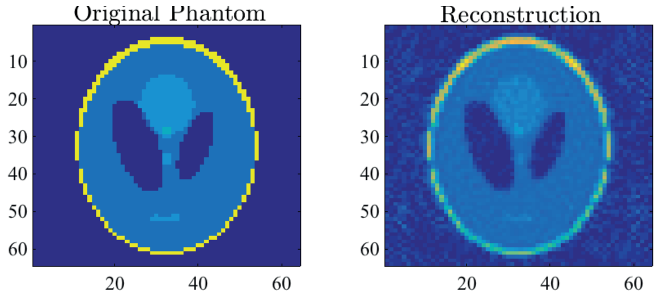


Figure 8.6 – The reconstruction of a 64 × 64 Modified Shepp-Logan phantom from 20% projections using eq.(8.2).

is obtained from the first 3 iterations of the compressed sensing based recovery problem and then these graphs are updated every 5 iterations. Our future work will focus on a detailed study of this method.

8.5.5 Dynamic MRI Reconstructions

In MRI, the goal is to efficiently reconstruct the 3D specimen under examination from a few frequency samples acquired over the scan time. In static MRI (sMRI) [135], [136], [137], where the specimen does not move or change in time, the goal is to sample sparsely to reduce the scan time while maintaining a good reconstruction quality of the image [54]. However, dynamic MRI (dMRI) [138], which involves applications such as dynamic cardiac imaging [139], spectroscopic imaging [140], diffusion imaging [141] and multiparameter mapping [142], is different in the sense that it is inherently undersampled because the specimen is changing as the data is being collected. In addition to being inherently under-sampled, dMRI is also high dimensional, which results in an exponential increase in the number of samples needed for imaging [143]. This poses a higher computational burden on the imaging algorithms, which need to cope simultaneously with the speed while maintaining a good reconstruction quality from highly undersampled data. The data collected from dMRI is a sequence of 2D slices or 3D volumes, corresponding to an image of a certain part of the body, varying over time, resulting in a 3D, 4D or even a higher dimensional tensor. It is hence desirable to use tensor models for reconstructing this data from the raw measurements.

An important application in dMRI is to recover the low-rank and sparse tensors corresponding to the specimen being imaged. More formally, the goal is to recover the low-rank $\mathscr{X}_l \in \Re^{n\times n\times n}$ and sparse $\mathscr{X}_s \in \Re^{n\times n\times n}$ components of the tensor $\mathscr{X} \in \Re^{n\times n\times n}$ from measurements y in an efficient and accelerated manner. We propose the following general class of convex optimization problems for this purpose:

$$\begin{aligned}&\min_{\mathcal{Z},\mathscr{X}_s,\mathscr{X}_l}\left\|\operatorname{vec}(\boldsymbol{M})\Big(\operatorname{vec}(\mathscr{X}_s)+(\boldsymbol{P}_k^1\otimes\boldsymbol{P}_k^2\otimes\boldsymbol{P}_k^3)\operatorname{vec}(\mathcal{Z})\Big)-\boldsymbol{y}\right\|_2^2+\gamma\sum_{\mu}\|\boldsymbol{Z}^{\mu}\|_*+\lambda R(\mathscr{X}_s),\\ &\quad\text{s.t. }\operatorname{vec}(\mathscr{X}_l)=(\boldsymbol{P}_k^1\otimes\boldsymbol{P}_k^2\otimes\boldsymbol{P}_k^3)\operatorname{vec}(\mathcal{Z}),\end{aligned}\tag{8.3}$$

where $R(\cdot)$ is a general sparsity regularizer and $\boldsymbol{P}_k^{\mu}$ are the Laplacian eigenvectors of graphs $\mathbb{G}^{\mu}$.

Note that in constrast to the MLRTG framework [144], one needs to recover $\mathscr{X}$ from the measurements y in eq. 8.3. Thus, the graphs $\mathbb{G}^{\mu}$ cannot be estimated directly from $\mathscr{X}$, because $\mathscr{X}$ itself is the optimization variable and is updated in every iteration of the algorithm. We propose to determine an initial estimate of $\mathcal{X}$ from y by a simple CS based algorithm to recover each of the spatial frames of the tensor $\mathcal{X}$ independently of the other frames in time.

Finally, several regularizers $R(\cdot)$ can be tested. The most common choice involves the use of a simple l_1 norm on $\mathcal{X}_s$ or an l_1 norm in a suitable transform basis on each of the frames in the tensor $\mathcal{X}_s$. Other choices of interest include a total variation (TV) norm or a graph total variation norm (GTV) in a spirit similar to the works [145], [146] and [147].

Instead of fixing the transform basis upfront, one can modify eq. 8.3 to incorporate data-adaptive dictionary learning for the sparse tensor $\mathcal{X}_s$, in the spirit similar to the works of [148] and [54]. Problem (8.3) can be reformulated as follows:

$$\begin{aligned}
\min_{\mathcal{Z},\mathcal{X}_s,\mathcal{X}_l,\boldsymbol{D},\boldsymbol{S}} \; & \frac{1}{2}\left\| \mathrm{vec}(\boldsymbol{M})\Big(\mathrm{vec}(\mathcal{X}_s) + (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\,\mathrm{vec}(\mathcal{Z})\Big) - y \right\|_2^2 \\
& + \lambda_l \sum_i \|\boldsymbol{Z}_i\|_* + \lambda_s \sum_{j=1}^{N} \left\| \boldsymbol{A}_j\,\mathrm{vec}(\mathcal{X}_s) - \boldsymbol{D}\boldsymbol{s}_j \right\|_2^2 + \lambda_s \|\boldsymbol{S}\|_1, \\
\text{s.t} \quad & \mathrm{vec}(\mathcal{X}_l) = (\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\,\mathrm{vec}(\mathcal{Z}), \\
& \|\boldsymbol{S}\|_\infty \le a, \quad \mathrm{rank}(R_2(\boldsymbol{d}_i)) \le r, \quad \|\boldsymbol{d}_i\|_2 = 1 \;\forall\; i,
\end{aligned} \tag{8.4}$$

where $\boldsymbol{A}_j$ is a patch extraction vector that extracts a spatio-temporal patch of size $m_1 \times m_2 \times m_3$ from the tensor $\mathcal{X}_s$, N is the total number of 3D patches and $\boldsymbol{D} \in \mathbb{R}^{m \times K}$, where $m = m_1 m_2 m_3$ is the dictionary to be learned and $\boldsymbol{s}_j \in \Re^K$ is the sparse code for the j^{th} patch. We arrange all the sparse codes $\boldsymbol{s}_j$ in the form of a matrix $\boldsymbol{S} \in \mathbb{R}^{K \times N}$. The l_∞ norm on $\boldsymbol{S}$ prevents pathologies that may arise due to non-coercive nature of problem (8.4) and the unit norm constraint on the atoms of the dictionary $\boldsymbol{D}$ prevents the scaling ambiguity between $\boldsymbol{D}$ and $\boldsymbol{S}$ [54]. $R_2(\cdot)$ is an operator that reshapes $\boldsymbol{d}_i$ into a matrix of size $m_1 m_2 \times m_3$. The rank constraint on $\boldsymbol{d}_i$ comes from the empirical observations made in [54].

8.5.6 Discriminative Feature Learning for Tensors

Several variants of the tensor non-linear embedding methods have been proposed for supervised and discriminative feature learning. For example, the authors of [26] proposed a discriminant LLE method for tensors which uses two graphs, one to ensure the same behavior as LLE and the other to enforce class strucure between the emebddings. The authors of [27] propose a multilinear graph embedding method and its kernel variant which uses the same two types of graphs as [26]. The authors of [28] propose a tensor based patch alignment method for hyperspectral data and those of [29] propose a semi-supervised framework for tensor based graph embedding.

Our proposed framework can also be extended for learning discriminative features for classification of low-rank tensors. Let $(\mathcal{X}_i, l_i)$ be the tensor-label pairs for a certain application. The goal is to learn a discriminant function and discrimnative graph core tensor based features $\mathcal{Z}_i$ for each of the training pairs by training a model. We propose the following class of optimization problems for this purpose:

$$\min_{\mathcal{Z}_i,\boldsymbol{w}} \sum_i \left(\phi\Big(\mathrm{vec}(\boldsymbol{P}_k^1 \otimes \boldsymbol{P}_k^2 \otimes \boldsymbol{P}_k^3)\,\mathrm{vec}(\mathcal{Z}_i) - \mathrm{vec}(\mathcal{X}_i)\Big) + \gamma \sum_\mu \|f(\boldsymbol{\Lambda}^k)\boldsymbol{Z}_i^\mu\|_* + \alpha h(\mathcal{Z}_i, l_i, \boldsymbol{w}) \right) + \beta\Omega(\boldsymbol{w}),$$

where $h(\cdot)$ is a certain loss function for classification, $\boldsymbol{w}$ the discriminant parameters and $\Omega(\cdot)$ an optional regularizer on the parameters $\boldsymbol{w}$.

8.5.7 Low-Rank Tensors for Convolutional Neural Networks

Low-rank tensors have also been shown to be of profound importance for convolutional neural networks (CNNs). These deep networks learn a high level non-linear representation of the input images for the purpose of classification and consist of several layers with millions of parameters. Interestingly, the parameters at every layer of a CNN, also called the weights, can be represented in the form of tensors. Since these weight tensors are large in number and possess the physical interpretation of filters, they are extremely redundant. One can exploit this redundancy to speed up the convolution operations by using tensor based low-rank decompositions. Such ideas started appearing very recently with the introduction of low-rank expansions for CNNs in [42] and [43]. Other techniques based on further optimizations for a better speed up have also been proposed, for example in [44] and [149].

Our proposed method can be used to decompose the low-rank weight tensors in a manner similar to the one used in the state-of-the-art techniques, specially in the first few layers of the network, where the weight tensor has large size along each of its modes. Note that our proposed framework is equivalent to a factorized graph regularized problem, therefore, using our framework would also exploit the intrinsic structure in the different modes of the weight tensor. Thus, enabling structured learning of weight tensors.

A Appendices

A.1 Proof of Lemma 3.3.1

Proof. The adjacency matrix of the Cartesian product graph $\mathbb{G}$ is given by the matrix Cartesian product:

$$\boldsymbol{W} = \boldsymbol{W}^1 \times \boldsymbol{W}^2 = \boldsymbol{W}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{W}^2$$

With this definition of the adjacency matrix, it is possible to write the degree matrix of the Cartesian product graph as cartesian product of the factor degree matrices:

$$\begin{aligned}
\boldsymbol{d} &= \boldsymbol{W}(\mathbf{1}^1 \otimes \mathbf{1}^2) = (\boldsymbol{W}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes W^2)(\mathbf{1}^1 \otimes \mathbf{1}^2) \\
&= (\boldsymbol{W}^1 \otimes \boldsymbol{I}^2)(\mathbf{1}^1 \otimes \mathbf{1}^2) + (\boldsymbol{I}^1 \otimes \boldsymbol{W}^2)(\mathbf{1}^1 \otimes \mathbf{1}^2) \\
&= (\boldsymbol{W}^1 \mathbf{1}^1) \otimes (\boldsymbol{I}^2 \mathbf{1}^2) + (\boldsymbol{I}^1 \mathbf{1}^1) \otimes (\boldsymbol{W}^2 \mathbf{1}^2) \\
&= \boldsymbol{d}^1 \otimes \mathbf{1}^1 + \mathbf{1}^2 \otimes \boldsymbol{d}^2
\end{aligned}$$

where we have used the following property

$$(\boldsymbol{A}^1 \otimes \boldsymbol{B}^1)(\boldsymbol{A}^2 \otimes \boldsymbol{B}^2) = (\boldsymbol{A}^1 \boldsymbol{A}^2) \otimes (\boldsymbol{B}^1 \boldsymbol{B}^2).$$

This implies the following matrix equality:

$$\boldsymbol{D} = \boldsymbol{D}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{D}^2 = \boldsymbol{D}^1 \times \boldsymbol{D}^2$$

The **combinatorial** Laplacian of the cartesian product is:

$$\begin{aligned}
\boldsymbol{L} &= \boldsymbol{D} - \boldsymbol{W} = \boldsymbol{D}^1 \times \boldsymbol{D}^2 - \boldsymbol{W}^1 \times \boldsymbol{W}^2 \\
&= (\boldsymbol{D}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{D}^2) - (\boldsymbol{W}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{W}^2) \\
&= (\boldsymbol{D}^1 - \boldsymbol{W}^1) \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes (\boldsymbol{D}^2 - \boldsymbol{W}^2) \\
&= \boldsymbol{L}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{L}^2 = \boldsymbol{L}^1 \times \boldsymbol{L}^2
\end{aligned}$$

□

A.2 Proof of Lemma 3.3.2

Proof. The eigenvector matrix of the cartesian product graph can be derived as:

$$\begin{aligned}
\boldsymbol{L} = \boldsymbol{L}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{L}^2 &= (\boldsymbol{P}^1 \boldsymbol{\Lambda}^1 \boldsymbol{P}^{1^\top}) \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes (\boldsymbol{P}^2 \boldsymbol{\Lambda}^2 \boldsymbol{P}^{2^\top}) \\
&= (\boldsymbol{P}^1 \boldsymbol{\Lambda}^1 \boldsymbol{P}^{1^\top}) \otimes (\boldsymbol{P}^2 \boldsymbol{I}^2 \boldsymbol{P}^{2^\top}) + (\boldsymbol{P}^1 \boldsymbol{I}^1 \boldsymbol{P}^{1^\top}) \otimes (\boldsymbol{P}^2 \boldsymbol{\Lambda}^2 \boldsymbol{P}^{2^\top}) \\
&= (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{\Lambda}^1 \otimes \boldsymbol{I}^2)(\boldsymbol{P}^{1^\top} \otimes \boldsymbol{P}^{2^\top}) + (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{I}^1 \otimes \boldsymbol{\Lambda}^2)(\boldsymbol{P}^{1^\top} \otimes \boldsymbol{P}^{2^\top}) \\
&= (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{\Lambda}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{\Lambda}^2)(\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)^\top \\
&= (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{\Lambda}^1 \times \boldsymbol{\Lambda}^2)(\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)^\top = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top
\end{aligned}$$

So the eigenvector matrix is given by the Kronecker product between the eigenvector matrices of the factor graphs and the eigenvalues are the element-wise summation between all the possible pairs of factors eigenvalues, i.e. the cartesian product between the eigenvalue matrices.

$$\begin{aligned}
\boldsymbol{L}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{L}^2 &= (\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)(\boldsymbol{\Lambda}^1 \times \boldsymbol{\Lambda}^2)(\boldsymbol{P}^1 \otimes \boldsymbol{P}^2)^\top \\
&= \big((\boldsymbol{P}^1_{k_1} + \bar{\boldsymbol{P}}^1_{k_1}) \otimes (\boldsymbol{P}^2_{k_2} + \bar{\boldsymbol{P}}^2_{k_2})\big)\big((\boldsymbol{\Lambda}^1_{k_1} + \bar{\boldsymbol{\Lambda}}^1_{k_1}) \times (\boldsymbol{\Lambda}^2_{k_2} + \bar{\boldsymbol{\Lambda}}^2_{k_2})\big) \\
&\quad \big((\boldsymbol{P}^1_{k_1} + \bar{\boldsymbol{P}}^1_{k_1}) \otimes (\boldsymbol{P}^2_{k_2} + \bar{\boldsymbol{P}}^2_{k_2})\big)^\top,
\end{aligned} \tag{A.1}$$

where we assume that $\boldsymbol{P}_{k_\mu} \in \mathbb{R}^{n_\mu \times n_\mu}$ with the first k_μ columns in $\boldsymbol{P}^1$ and 0 appended for others and $\bar{\boldsymbol{P}}_{k_\mu} \in \mathbb{R}^{n_\mu \times n_\mu}$ with the first k_μ columns equal to 0 and others copied from $\boldsymbol{P}^1$. The same holds for $\boldsymbol{\Lambda}^\mu, \bar{\boldsymbol{\Lambda}}^\mu$ as well. Now

$$\begin{aligned}
(\boldsymbol{P}^1_{k_1} + \bar{\boldsymbol{P}}^1_{k_1}) \otimes (\boldsymbol{P}^2_{k_2} + \bar{\boldsymbol{P}}^2_{k_2}) &= \big(\boldsymbol{P}^1_{k_1} \otimes \boldsymbol{P}^2_{k_2} + \boldsymbol{P}^1_{k_1} \otimes \bar{\boldsymbol{P}}^2_{k_2} + \bar{\boldsymbol{P}}^1_{k_1} \otimes \boldsymbol{P}^2_{k_2} + \bar{\boldsymbol{P}}^1_{k_1} \otimes \bar{\boldsymbol{P}}^2_{k_2}\big) \\
&= \boldsymbol{P}_{k_1 k_2} + \bar{\boldsymbol{P}}_{k_1 k_2},
\end{aligned}$$

where we use $\boldsymbol{P}_{k_1 k_2} = \boldsymbol{P}_{1 k_1} \otimes \boldsymbol{P}_{2 k_2}$. Now removing the zero appended columns we get $\boldsymbol{P} = (\boldsymbol{P}_{k_1 k_2}, \bar{\boldsymbol{P}}_{k_1 k_2})$. Now, let

$$(\boldsymbol{\Lambda}^1_{k_1} + \bar{\boldsymbol{\Lambda}}^1_{k_1}) \times (\boldsymbol{\Lambda}^2_{k_2} + \bar{\boldsymbol{\Lambda}}^2_{k_2}) = \boldsymbol{\Lambda}_{k_1 k_2} + \bar{\boldsymbol{\Lambda}}_{k_1 k_2}$$

removing the zero appended entries we get $\boldsymbol{\Lambda} = (\boldsymbol{\Lambda}_{k_1 k_2}, \bar{\boldsymbol{\Lambda}}_{k_1 k_2})$. For a k_{nn}-nearest neighbors graph constructed from a k_1-clusterable data (along rows) one can expect $\lambda_{k_1}/\lambda_{k_1+1} \approx 0$ as $\lambda_{k_1} \approx 0$ and $\lambda_{k_1} \ll \lambda_{k_1+1}$. Furthermore $\max_\mu(\lambda_{k_\mu}) \ll \min_\mu(\lambda_{k_\mu+1})$ (definitions 3.3.1 & 3.3.3). Thus eq. (A.1) can be written as:

$$\boldsymbol{L}^1 \otimes \boldsymbol{I}^2 + \boldsymbol{I}^1 \otimes \boldsymbol{L}^2 = [\boldsymbol{P}_{k_1 k_2} | \bar{\boldsymbol{P}}_{k_1 k_2}][\boldsymbol{\Lambda}_{k_1 k_2} | \bar{\boldsymbol{\Lambda}}_{k_1 k_2}][\boldsymbol{P}_{k_1 k_2} | \bar{\boldsymbol{P}}_{k_1 k_2}]^\top$$

□

A.3 Proof of Theorem 5.6.1

Proof. As $\boldsymbol{X}$ is a solution of eq. 5.1, we have

$$\phi(\boldsymbol{Y}-\boldsymbol{X})+\gamma_1 \operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^1\boldsymbol{X}^\top)+\gamma_2 \operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^2\boldsymbol{X}) \leq \phi(\boldsymbol{\eta})+\gamma_1 \operatorname{tr}(\boldsymbol{X}^*\boldsymbol{L}^1\boldsymbol{X}^{*\top})+\gamma_2 \operatorname{tr}(\boldsymbol{X}^{*\top}\boldsymbol{L}^2\boldsymbol{X}^*). \tag{A.2}$$

Using the facts that $\boldsymbol{L}^1 = \boldsymbol{P}_k^1\boldsymbol{L}_k^1\boldsymbol{P}_k^{1\top} + \bar{\boldsymbol{P}}_k^1\bar{\boldsymbol{L}}_k^1\bar{\boldsymbol{P}}_k^{1\top}$ and that there exists $\boldsymbol{B} \in \mathbb{R}^{p\times k}$ and $\bar{\boldsymbol{B}} \in \mathbb{R}^{p\times(n-k)}$ such that $\boldsymbol{X} = \boldsymbol{B}\boldsymbol{P}_k^{1\top} + \bar{\boldsymbol{B}}\bar{\boldsymbol{P}}_k^{1\top}$, we obtain

$$\begin{aligned}
\operatorname{tr}(\boldsymbol{X}\boldsymbol{L}^1\boldsymbol{X}^\top) &= \operatorname{tr}(\boldsymbol{B}\boldsymbol{\Lambda}_k^1\boldsymbol{B}^\top) + \operatorname{tr}(\bar{\boldsymbol{B}}\bar{\boldsymbol{\Lambda}}_k^1\bar{\boldsymbol{B}}^\top)\\
&\geq \operatorname{tr}(\bar{\boldsymbol{\Lambda}}_k^1\bar{\boldsymbol{B}}^\top\bar{\boldsymbol{B}})\\
&\geq \lambda_{k+1}^1\|\bar{\boldsymbol{B}}\|_F^2\\
&= \lambda_{k+1}^1\|\boldsymbol{X}\bar{\boldsymbol{P}}_k^1\|_F^2.
\end{aligned}$$

Then, using the fact that there exists $\boldsymbol{C} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{X}^* = \boldsymbol{P}_k^1\boldsymbol{C}\boldsymbol{P}_k^{2\top}$, we obtain

$$\operatorname{tr}(\boldsymbol{X}^*\boldsymbol{L}^1\boldsymbol{X}^{*\top}) = \operatorname{tr}(\boldsymbol{C}\boldsymbol{\Lambda}_k^1\boldsymbol{C}^\top) \leq \lambda_k^1\|\boldsymbol{C}\|_F^2 = \lambda_k^1\|\boldsymbol{X}^*\|_F^2.$$

Similarly, we have

$$\operatorname{tr}(\boldsymbol{X}^\top\boldsymbol{L}^2\boldsymbol{X}) \geq \lambda_{k+1}^2\|\bar{\boldsymbol{P}}_k^{2\top}\boldsymbol{X}\|_F^2,$$

$$\operatorname{tr}(\boldsymbol{X}^{*\top}\boldsymbol{L}^2\boldsymbol{X}^*) \leq \lambda_k^2\|\boldsymbol{X}^*\|_F^2.$$

Using the four last bounds in eq. A.2 yields

$$\phi(\boldsymbol{X}-\boldsymbol{Y})+\gamma_1\lambda_{k+1}^1\|\boldsymbol{X}\bar{\boldsymbol{P}}_k\|_F^2+\gamma_2\lambda_{k+1}^2\|\bar{\boldsymbol{P}}_k^{2\top}\boldsymbol{X}\|_F^2 \leq \phi(\boldsymbol{\eta})+\gamma_1\lambda_k^1\|\boldsymbol{X}^*\|_F^2+\gamma_2\lambda_k^2\|\boldsymbol{X}^*\|_F^2,$$

which becomes

$$\phi(\boldsymbol{Y}-\boldsymbol{X})+\gamma\|\boldsymbol{X}\bar{\boldsymbol{P}}_k^1\|_F^2+\gamma\|\bar{\boldsymbol{P}}_k^{2\top}\boldsymbol{X}\|_F^2 \leq \phi(\boldsymbol{\eta})+\gamma\|\boldsymbol{X}^*\|_F^2\left(\frac{\lambda_k^1}{\lambda_{k+1}^1}+\frac{\lambda_k^2}{\lambda_{k+1}^2}\right)$$

for our choice of γ_1 and γ_2. This terminates the proof. □

A.4 Proof of theorem 6.3.1

Proof. We start with the sampling of the rows. Theorem 5 in [100] shows that for any $\delta_r, \epsilon_r \in (0,1)$, with probability at least $1-\epsilon_r$,

$$(1-\delta_r)\|\boldsymbol{z}\|_2^2 \leq \frac{p}{\rho_r}\|\boldsymbol{M}^r\boldsymbol{z}\|_2^2 \leq (1+\delta_r)\|\boldsymbol{z}\|_2^2$$

for all $\boldsymbol{z} \in span(\boldsymbol{P}^r_{k_r})$ provided that

$$\rho_r \geq \frac{3}{\delta_r^2} \nu_{k_r}^2 \log\left(\frac{2k_r}{\epsilon_r}\right). \tag{A.3}$$

Notice that Theorem 5 in [100] is a uniform result. As a consequence, with probability at least $1-\epsilon_r$,

$$(1-\delta_r)\|\boldsymbol{X}^*(i)\|_2^2 \leq \frac{p}{\rho_r}\|\boldsymbol{M}^r\boldsymbol{X}^*(i)\|_2^2 \leq (1+\delta_r)\|\boldsymbol{X}^*(i)\|_2^2, \quad i = 1,\ldots,n, \tag{A.4}$$

for all $\boldsymbol{X}^*(1),\ldots,\boldsymbol{X}^*(n) \in span(\boldsymbol{P}^r_{k_r})$ provided that (A.3) holds. Summing the previous inequalities over all i shows that, with probability at least $1-\epsilon_r$,

$$(1-\delta_r)\|\boldsymbol{X}^*\|_F^2 \leq \frac{p}{\rho_r}\|\boldsymbol{M}^r\boldsymbol{X}^*\|_F^2 \leq (1+\delta_r)\|\boldsymbol{X}^*\|_F^2, \tag{A.5}$$

for all $\boldsymbol{X}^* \in \Re^{p\times n}$ with column-vectors in $span(\boldsymbol{P}^r_{k_r})$.

Let us continue with the sampling of the columns. Again, Theorem 5 in [100] shows that for any $\delta_c, \epsilon_c \in (0,1)$, with probability at least $1-\epsilon_c$,

$$(1-\delta_c)\|\boldsymbol{w}\|_2^2 \leq \frac{n}{\rho_c}\|\boldsymbol{w}^\top\boldsymbol{M}^c\|_2^2 \leq (1+\delta_c)\|\boldsymbol{w}\|_2^2$$

for all $\boldsymbol{w} \in span(\boldsymbol{P}^c_{k_c})$ provided that

$$\rho_c \geq \frac{3}{\delta_c^2} \nu_{k_c}^2 \log\left(\frac{2k_c}{\epsilon_c}\right). \tag{A.6}$$

As a consequence, with probability at least $1-\epsilon_c$,

$$(1-\delta_c)\|\boldsymbol{z}_i\|_2^2 \leq \frac{n}{\rho_c}\|\boldsymbol{Z}(i)^\top\boldsymbol{M}^c\|_2^2 \leq (1+\delta_c)\|\boldsymbol{Z}(i)\|_2^2, \quad i = 1,\ldots,\rho_r, \tag{A.7}$$

for all $\boldsymbol{Z}(1),\ldots,\boldsymbol{Z}(\rho_r) \in span(\boldsymbol{P}^c_{k_c})$ provided that (A.6) holds. Summing the previous inequalities over all i shows that, with probability at least $1-\epsilon_c$,

$$(1-\delta_c)\|\boldsymbol{Z}\|_F^2 \leq \frac{n}{\rho_c}\|\boldsymbol{Z}\boldsymbol{M}^c\|_F^2 \leq (1+\delta_c)\|\boldsymbol{Z}\|_F^2 \tag{A.8}$$

for all $\boldsymbol{Z} \in \mathbb{R}^{\rho_r\times n}$ with row-vectors in $span(\boldsymbol{P}^c_{k_c})$. In particular, this property holds, with at least the same probability, for all matrices $\boldsymbol{Z}$ of the form $\boldsymbol{M}^r\boldsymbol{X}^*$ where $\boldsymbol{X}^* \in \mathbb{R}^{p\times n}$ is a matrix with row-vectors in $span(\boldsymbol{P}^c_{k_c})$.

We now continue by combining (A.5) and (A.8). We obtain that

$$(1-\delta_c)(1-\delta_r)\|\boldsymbol{X}^*\|_F^2 \leq \frac{np}{\rho_c\rho_r}\|\boldsymbol{M}^r\boldsymbol{X}^*\boldsymbol{M}^c\|_F^2 \leq (1+\delta_c)(1+\delta_r)\|\boldsymbol{X}^*\|_F^2 \tag{A.9}$$

for all $\boldsymbol{X}^* \in \mathbb{R}^{p\times n}$ with column-vectors in $span(\boldsymbol{P}^r_{k_r})$ and row-vectors in $span(\boldsymbol{P}^c_{k_c})$, provided that (A.3) and (A.6) hold. It remains to compute the probability that (A.9) holds. Property (A.9) does not hold if (A.5) or (A.8) do not hold. Using the union bound, (A.9) does not hold with probability at most $\epsilon_r + \epsilon_c$. To finish the proof, one just need to choose $\epsilon_r = \epsilon_c = \epsilon/2$ and $\delta_r = \delta_c = \delta/3$, and notice that $(1+\delta/3)^2 \leq 1+\delta$ and $(1-\delta/3)^2 \geq 1-\delta$ for $\delta \in (0,1)$. □

A.5 Proof of Theorem 6.5.1

Proof. Using the optimality condition we have, for any $\boldsymbol{Z} \in \mathbb{R}^{p\times n}$,

$$\|\boldsymbol{M}^r \bar{\boldsymbol{X}}^* \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F \leq \|\boldsymbol{M}^r \boldsymbol{Z} \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F.$$

For $\boldsymbol{Z} = \boldsymbol{X}^*$, we have

$$\|\boldsymbol{M}^r \bar{\boldsymbol{X}}^* \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F \leq \|\boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F,$$

which gives

$$\|\boldsymbol{M}^r \bar{\boldsymbol{X}}^* \boldsymbol{M}^c - \boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c - \tilde{\boldsymbol{E}}\|_F \leq \|\tilde{\boldsymbol{E}}\|_F.$$

As (6.2) holds, we have

$$\begin{aligned}\|\boldsymbol{M}^r \bar{\boldsymbol{X}}^* \boldsymbol{M}^c - \boldsymbol{M}^r \boldsymbol{X}^* \boldsymbol{M}^c - \tilde{\boldsymbol{E}}\|_F &\geq \|\boldsymbol{M}^r (\bar{\boldsymbol{X}}^* - \boldsymbol{X}^*) \boldsymbol{M}^c\|_F - \|\tilde{\boldsymbol{E}}\|_F \\ &\geq \sqrt{\frac{\rho_r \rho_c (1-\delta)}{np}} \|\bar{\boldsymbol{X}}^* - \boldsymbol{X}^*\|_F - \|\tilde{\boldsymbol{E}}\|_F.\end{aligned}$$

Therefore, by combining the above equations we get

$$\|\bar{\boldsymbol{X}}^* - \boldsymbol{X}^*\|_F \leq 2\sqrt{\frac{np}{\rho_r \rho_c (1-\delta)}} \|\tilde{\boldsymbol{E}}\|_F.$$

□

A.6 Proof of Theorem 6.5.2

Proof. Using the optimality condition we have for any $\boldsymbol{Z} \in \mathbb{R}^{p\times n}$ and optimal solution $\bar{\boldsymbol{X}}^* = \underline{\boldsymbol{X}}^* + \boldsymbol{E}^*$:

$$\|\boldsymbol{M}^r \bar{\boldsymbol{X}}^* \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F^2 + \bar{\gamma}_c \operatorname{tr}(\bar{\boldsymbol{X}}^* \boldsymbol{L}^c \bar{\boldsymbol{X}}^{*\top}) + \bar{\gamma}_r \operatorname{tr}(\bar{\boldsymbol{X}}^{*\top} \boldsymbol{L}^r \bar{\boldsymbol{X}}^*) \leq \|\boldsymbol{M}^r \boldsymbol{Z} \boldsymbol{M}^c - \tilde{\boldsymbol{X}}\|_F^2 + \bar{\gamma}_c \operatorname{tr}(\boldsymbol{Z} \boldsymbol{L}^c \boldsymbol{Z}^\top) + \bar{\gamma}_r \operatorname{tr}(\boldsymbol{Z}^\top \boldsymbol{L}^r \boldsymbol{Z}) \tag{A.10}$$

using $\boldsymbol{Z} = \boldsymbol{X}^* = \boldsymbol{P}^r_{k_r} \boldsymbol{Y}_b \boldsymbol{P}^{c\,\top}_{k_c}$ as in the proof of theorem 2 in [110], where $\boldsymbol{Y}_b \in \mathbb{R}^{k_r \times k_c}$ and it is not necessarily diagonal. Note that $\|\boldsymbol{Y}_b\|_F = \|\boldsymbol{X}^*\|_F$, $\boldsymbol{P}^{c\,\top}_{k_c} \boldsymbol{P}^c_{k_c} = \boldsymbol{I}_{k_c}$, $\boldsymbol{P}^{r\,\top}_{k_r} \boldsymbol{P}^r_{k_r} = \boldsymbol{I}_{k_r}$, $\bar{\boldsymbol{P}}^{c\,\top}_{k_c} \boldsymbol{P}^c_{k_c} = \boldsymbol{0}$, $\bar{\boldsymbol{P}}^{r\,\top}_{k_r} \boldsymbol{P}^r_{k_r} = \boldsymbol{0}$. From the proof of theorem 2 in [110] we also know that:

$$\begin{gathered}\operatorname{tr}(\boldsymbol{X}^* \boldsymbol{L}^c \boldsymbol{X}^{*\top}) \leq \lambda^c_{k_c} \|\boldsymbol{X}^*\|_F^2 \\ \operatorname{tr}(\boldsymbol{X}^{*\top} \boldsymbol{L}^r \boldsymbol{X}^*) \leq \lambda^r_{k_r} \|\boldsymbol{X}^*\|_F^2 \\ \operatorname{tr}(\bar{\boldsymbol{X}}^* \boldsymbol{L}^c \bar{\boldsymbol{X}}^{*\top}) \geq \lambda^c_{k_c+1} \|\bar{\boldsymbol{X}}^* \bar{\boldsymbol{P}}^c_{k_c}\|_F^2 \\ \operatorname{tr}(\bar{\boldsymbol{X}}^{*\top} \boldsymbol{L}^r \bar{\boldsymbol{X}}^*) \geq \lambda^r_{k_r+1} \|\bar{\boldsymbol{P}}^{r\,\top}_{k_r} \bar{\boldsymbol{X}}^*\|_F^2\end{gathered}$$

Now using all this information in (A.10) we get

$$\|\boldsymbol{M}^r\bar{\boldsymbol{X}}^*\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F^2+\bar{\gamma}_c\lambda^c_{k_c+1}\|\bar{\boldsymbol{X}}^*\bar{\boldsymbol{P}}^c_{k_c}\|_F^2+\bar{\gamma}_r\lambda^r_{k_r+1}\|\bar{\boldsymbol{P}}^r_{k_r}{}^\top\bar{\boldsymbol{X}}^*\|_F^2\leq\|\tilde{\boldsymbol{E}}\|_F^2+(\bar{\gamma}_c\lambda^c_{k_c}+\bar{\gamma}_r\lambda^r_{k_r})\|\boldsymbol{X}^*\|_F^2$$

From above we have:

$$\|\boldsymbol{M}^r\bar{\boldsymbol{X}}^*\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F\leq\|\tilde{\boldsymbol{E}}\|_F+\sqrt{(\bar{\gamma}_c\lambda^c_{k_c}+\bar{\gamma}_r\lambda^r_{k_r})}\|\boldsymbol{X}^*\|_F \tag{A.11}$$

and

$$\sqrt{(\bar{\gamma}_c\lambda^c_{k_c+1}\|\bar{\boldsymbol{X}}^*\bar{\boldsymbol{P}}^c_{k_c}\|_F^2+\bar{\gamma}_r\lambda^r_{k_r+1}\|\bar{\boldsymbol{P}}^r_{k_r}{}^\top\bar{\boldsymbol{X}}^*\|_F^2)}\leq\|\tilde{\boldsymbol{E}}\|_F+\sqrt{(\bar{\gamma}_c\lambda^c_{k_c}+\bar{\gamma}_r\lambda^r_{k_r})}\|\boldsymbol{X}^*\|_F \tag{A.12}$$

using

$$\bar{\gamma}_c=\gamma\frac{1}{\lambda^c_{k_c+1}}\quad\text{and}\quad\bar{\gamma}_r=\gamma\frac{1}{\lambda^r_{k_r+1}},$$

and

$$\|\boldsymbol{E}^*\|_F^2=\|\bar{\boldsymbol{X}}^*\bar{\boldsymbol{P}}^c_{k_c}\|_F^2=\|\bar{\boldsymbol{P}}^r_{k_r}{}^\top\bar{\boldsymbol{X}}^*\|_F^2$$

we get:

$$\|\boldsymbol{M}^r\bar{\boldsymbol{X}}^*\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F\leq\|\tilde{\boldsymbol{E}}\|_F+\sqrt{\gamma\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}+\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F \tag{A.13}$$

and

$$\sqrt{2\gamma}\|\boldsymbol{E}^*\|_F\leq\|\tilde{\boldsymbol{E}}\|_F+\sqrt{\gamma\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}+\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F \tag{A.14}$$

which implies

$$\|\boldsymbol{E}^*\|_F\leq\frac{\|\tilde{\boldsymbol{E}}\|_F}{\sqrt{2\gamma}}+\frac{1}{\sqrt{2}}\sqrt{\gamma\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}+\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F \tag{A.15}$$

Now focus on $\|\boldsymbol{M}^r\bar{\boldsymbol{X}}^*\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F$. As $\boldsymbol{M}^r,\boldsymbol{M}^c$ are constructed with a sampling without replacement, we have $\|\boldsymbol{M}^r\boldsymbol{E}^*\boldsymbol{M}^c\|_F\leq\|\boldsymbol{E}^*\|_F$. Now using the above facts and the RIP we get:

$$\begin{aligned}\|\boldsymbol{M}^r\bar{\boldsymbol{X}}^*\boldsymbol{M}^c-\tilde{\boldsymbol{X}}\|_F&=\|\boldsymbol{M}^r(\underline{X}^*+\boldsymbol{E}^*)\boldsymbol{M}^c-\boldsymbol{M}^r\boldsymbol{X}^*\boldsymbol{M}^c-\tilde{\boldsymbol{E}}\|_F\\&\geq\sqrt{\frac{\rho_r\rho_c(1-\delta)}{np}}\|\underline{\boldsymbol{X}}^*-\boldsymbol{X}^*\|_F-\|\tilde{\boldsymbol{E}}\|_F-\|\boldsymbol{E}^*\|_F\end{aligned}$$

this implies

$$\|\underline{\boldsymbol{X}}^*-\boldsymbol{X}^*\|_F\leq\sqrt{\frac{np}{\rho_c\rho_r(1-\delta)}}\left[\left(2+\frac{1}{\sqrt{2\gamma}}\right)\|\tilde{\boldsymbol{E}}\|_F+(\frac{1}{\sqrt{2}}+\sqrt{\gamma})\sqrt{\left(\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}+\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}\right)}\|\boldsymbol{X}^*\|_F\right]$$

□

A.7 Proof of Theorem 6.5.3

Proof. We can write (6.12) and (6.13) as following:

$$\min_{\boldsymbol{U}(1)\cdots\boldsymbol{U}(p)} \sum_{i=1}^{p} \left[\|\boldsymbol{M}^r \boldsymbol{U}(i) - \tilde{\boldsymbol{U}}(i)\|_2^2 + {\gamma'}_r \boldsymbol{U}(i)^\top \boldsymbol{L}^r \boldsymbol{U}(i) \right] \tag{A.16}$$

$$\min_{\boldsymbol{V}(1)\cdots\boldsymbol{V}(n)} \sum_{i=1}^{n} \left[\|\boldsymbol{M}^c \boldsymbol{V}(i) - \tilde{\boldsymbol{V}}(i)\|_2^2 + {\gamma'}_c \boldsymbol{V}(i)^\top \boldsymbol{L}^c \boldsymbol{V}(i) \right] \tag{A.17}$$

In this proof, we only treat Problem (A.16) and the recovery of $\bar{\boldsymbol{U}}$. The proof for Problem (6.13) and the recovery of $\bar{\boldsymbol{V}}$ is identical. The above two problems can be solved independently for every i. From theorem 3.2 of [100] we obtain:

$$\|\bar{\boldsymbol{U}}^*(i) - \bar{\boldsymbol{U}}(i)\|_2 \leq \sqrt{\frac{p}{\rho_r(1-\delta)}} \left[\left(2 + \frac{1}{\sqrt{\gamma'_r \lambda^r_{k_r+1}}} \right) \|\tilde{\boldsymbol{E}}(i)^u\|_2 + \left(\sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}} + \sqrt{\gamma'_r \lambda^r_{k_r}} \right) \|\bar{\boldsymbol{U}}(i)\|_2 \right], \tag{A.18}$$

and

$$\|\boldsymbol{E}^*(i)\|_2 \leq \frac{1}{\sqrt{\gamma'_r \lambda^r_{k_r+1}}} \|\tilde{\boldsymbol{E}}(i)^u\|_2 + \sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}} \|\bar{\boldsymbol{U}}(i)\|_2,$$

which implies

$$\|\bar{\boldsymbol{U}}^*(i) - \bar{\boldsymbol{U}}(i)\|_2^2 \leq 2\frac{p}{\rho_r(1-\delta)} \left[\left(2 + \frac{1}{\sqrt{\gamma'_r \lambda^r_{k_r+1}}} \right)^2 \|\tilde{\boldsymbol{E}}(i)^u\|_2^2 + \left(\sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}} + \sqrt{\gamma'_r \lambda^r_{k_r}} \right)^2 \|\bar{\boldsymbol{U}}(i)\|_2^2 \right],$$

and

$$\|\boldsymbol{E}^*(i)\|_2^2 \leq \frac{2}{\gamma'_r \lambda^r_{k_r+1}} \|\tilde{\boldsymbol{E}}(i)^u\|_2^2 + 2\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}} \|\bar{\boldsymbol{U}}(i)\|_2^2.$$

Summing the previous inequalities over all i's yields

$$\|\bar{\boldsymbol{U}}^* - \bar{\boldsymbol{U}}\|_F^2 \leq 2\frac{p}{\rho_r(1-\delta)} \left[\left(2 + \frac{1}{\sqrt{\gamma'_r \lambda^r_{k_r+1}}} \right)^2 \|\tilde{\boldsymbol{E}}^u\|_F^2 + \left(\sqrt{\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}}} + \sqrt{\gamma'_r \lambda^r_{k_r}} \right)^2 \|\bar{\boldsymbol{U}}\|_F^2 \right],$$

and

$$\|\boldsymbol{E}^*\|_F^2 \leq \frac{2}{\gamma'_r \lambda^r_{k_r+1}} \|\tilde{\boldsymbol{E}}^u\|_F^2 + 2\frac{\lambda^r_{k_r}}{\lambda^r_{k_r+1}} \|\bar{\boldsymbol{U}}\|_F^2.$$

Taking the square root of both inequalities terminates the proof. Similarly, the expressions for $\bar{\boldsymbol{V}}$ can be derived:

$$\|\bar{\boldsymbol{V}}^* - \bar{\boldsymbol{V}}\|_F \leq \sqrt{\frac{2n}{\rho_c(1-\delta)}}\left[\left(2+\frac{1}{\sqrt{\gamma_c'\lambda^c_{k_c+1}}}\right)\|\tilde{\boldsymbol{E}}^v\|_F + \left(\sqrt{\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}} + \sqrt{\gamma_c'\lambda^c_{k_c}}\right)\|\bar{\boldsymbol{V}}\|_F\right]$$

and

$$\|\boldsymbol{E}^*\|_F \leq \sqrt{\frac{2}{\gamma_c'\lambda^c_{k_c+1}}}\|\tilde{\boldsymbol{E}}^v\|_F + \sqrt{2\frac{\lambda^c_{k_c}}{\lambda^c_{k_c+1}}}\|\bar{\boldsymbol{V}}\|_F.$$

□

A.8 Proof of Lemma 6.5.1

Proof. Let $\boldsymbol{S} = [\boldsymbol{S}_a^\top | \boldsymbol{S}_b^\top]^\top$. Further we split $\boldsymbol{L}$ into submatrices as follows:

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_{aa} & \boldsymbol{L}_{ab} \\ \boldsymbol{L}_{ba} & \boldsymbol{L}_{bb} \end{bmatrix}$$

Now (6.17) can be written as:

$$\min_{\boldsymbol{S}_a} \begin{bmatrix} \boldsymbol{S}_a \\ \boldsymbol{S}_b \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{L}_{aa} & \boldsymbol{L}_{ab} \\ \boldsymbol{L}_{ba} & \boldsymbol{L}_{bb} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_a \\ \boldsymbol{S}_b \end{bmatrix}$$
$$\text{s.t:} \quad \boldsymbol{S}_b = \boldsymbol{R}$$

further expanding we get:

$$\min_{\boldsymbol{S}_a} \boldsymbol{S}_a^\top \boldsymbol{L}_{aa}\boldsymbol{S}_a + \boldsymbol{S}_a^\top \boldsymbol{L}_{ab}\boldsymbol{R} + \boldsymbol{R}^\top \boldsymbol{L}_{ba}\boldsymbol{S}_a + \boldsymbol{R}\boldsymbol{L}_{bb}\boldsymbol{R}$$

using $\nabla \boldsymbol{S}_a = 0$, we get:

$$2\boldsymbol{L}_{ab}\boldsymbol{R} + 2\boldsymbol{L}_{aa}\boldsymbol{S}_a = 0$$

$$\boldsymbol{S}_a = -\boldsymbol{L}_{aa}^{-1}\boldsymbol{L}_{ab}\boldsymbol{R}$$

□

A.8.1 Proof of Theorem 7.4.1

Proof. We prove the 3 parts of this Theorem as following:

1. The existence of an eigen-gap and the fact that $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}_k^\mu)$ imply that one can obtain a loss-less compression of the tensor $\boldsymbol{X}^*$ as:

$$\boldsymbol{Z}^* = {\boldsymbol{P}^1_{k^*}}^\top \boldsymbol{X}^* \boldsymbol{P}^2_{k^*}.$$

Note that this compression is just a projection of the rows and columns of $\boldsymbol{X}^*$ onto the basis vectors which exactly encode the tensor, hence the compression is loss-less. Now, as $\boldsymbol{Z}^*$ is loss-less, the singular values of $\boldsymbol{X}^*$ should be an exact representation of the singular values of $\boldsymbol{Z}^*$. Thus, the SVD of $\boldsymbol{Z}^* = \boldsymbol{A}^1_{k^*}\boldsymbol{R}{\boldsymbol{A}^2_{k^*}}^\top$ implies that $\boldsymbol{R} = \boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ are the singular values of $\boldsymbol{X}^*$. Obviously, $\boldsymbol{H}^\mu = \boldsymbol{P}^\mu_{k^*}\boldsymbol{A}^\mu_{k^*}$ upto a sign permutation because the SVD of $\boldsymbol{Z}^*$ is also unique upto a sign permutation (a standard property of SVD).

2. The proof of second part follows directly from that of part 1 (above) and Theorem 1 in [34]. First, note that for any matrix (2D tensor) $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ eq. (7.12) can be written as following:

$$\min_{\boldsymbol{Z}} \phi\Big(\boldsymbol{M}(\boldsymbol{P}^1_k\boldsymbol{Z}{\boldsymbol{P}^2_k}^\top) - \boldsymbol{Y}\Big) + \gamma \Big\| f(\boldsymbol{\Lambda}^1_k)\boldsymbol{Z} f(\boldsymbol{\Lambda}^2_k)\Big\|_*, \tag{A.19}$$

where $f(\boldsymbol{\Lambda}^1_k)$ and $f(\boldsymbol{\Lambda}^2_k)$ are diagonal matrices which indicate the weights for the nuclear norm minimization.

Let $\boldsymbol{W}^1 = f(\boldsymbol{\Lambda}^1_k){\boldsymbol{P}^1_k}^\top$, $\boldsymbol{W}^2 = \boldsymbol{P}^2_k f(\boldsymbol{\Lambda}^2_k)$ and $\hat{\boldsymbol{X}} = \boldsymbol{P}^1_k\boldsymbol{Z}{\boldsymbol{P}^2_k}^\top$, then we can re-write eq. (A.19) as following:

$$\min_{\hat{\boldsymbol{X}}} \phi\Big(\boldsymbol{M}(\hat{\boldsymbol{X}}) - \boldsymbol{Y}\Big) + \gamma \Big\| \boldsymbol{W}^1\hat{\boldsymbol{X}}\boldsymbol{W}^2\Big\|_* \tag{A.20}$$

Eq. (A.20) is equivalent to the weighted nuclear norm (eq. 11) in [34]. From the proof of the first part we know that $\boldsymbol{H}^1 = \boldsymbol{P}^1_k\boldsymbol{A}^1_k$ and $\boldsymbol{H}^2 = \boldsymbol{P}^2_k\boldsymbol{A}^2_k$, thus we can write $\hat{\boldsymbol{X}} = \boldsymbol{H}^1{\boldsymbol{H}^2}^\top$. From Theorem 1 in [34]

$$\min_{\hat{\boldsymbol{X}}} \Big\| \boldsymbol{W}^1\hat{\boldsymbol{X}}\boldsymbol{W}^2\Big\|_* = \min_{\boldsymbol{H}^1,\boldsymbol{H}^2} \frac{1}{2}(\|\boldsymbol{W}^1\boldsymbol{H}^1\|^2_F + \|\boldsymbol{W}^2\boldsymbol{H}^2\|^2_F), \quad \text{s.t.} \quad \hat{\boldsymbol{X}} = \boldsymbol{H}^1{\boldsymbol{H}^2}^\top \tag{A.21}$$

using $f(\hat{\boldsymbol{L}}^\mu) = \boldsymbol{P}^\mu_k f(\boldsymbol{\Lambda}^\mu_k){\boldsymbol{P}^\mu_k}^\top$ this is equivalent to the following graph regularized problem:

$$\min_{\boldsymbol{H}^1,\boldsymbol{H}^2} \phi\Big(\boldsymbol{M}(\boldsymbol{H}^1{\boldsymbol{H}^2}^\top) - \boldsymbol{Y}\Big) + \gamma_1 \operatorname{tr}({\boldsymbol{H}^1}^\top f(\hat{\boldsymbol{L}}^1)\boldsymbol{H}^1) + \gamma_2 \operatorname{tr}({\boldsymbol{H}^2}^\top f(\hat{\boldsymbol{L}}^2)\boldsymbol{H}^2), \tag{A.22}$$

3. To prove the third point we directly work with eq.(7.13) and follow the steps of the proof of Theorem 1 in [110]. We assume the following:
   (a) We assume that the observed data matrix $\boldsymbol{Y}$ satisfies $\boldsymbol{Y} = \boldsymbol{M}(\boldsymbol{X}^*) + \boldsymbol{\eta}$ where $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ and $\boldsymbol{\eta}$ models noise/corruptions. Furthermore, for any $\boldsymbol{X}^* \in \mathbb{MLT}(\boldsymbol{P}^\mu_k)$ there exists a matrix $\boldsymbol{C}$ such that $\boldsymbol{X}^* = \boldsymbol{P}^1_{k^*}\boldsymbol{C}{\boldsymbol{P}^2_{k^*}}^\top$ and $\boldsymbol{C} = \boldsymbol{B}^1_{k^*}{\boldsymbol{B}^\top_{k^*}}^\top$, so $\boldsymbol{X}^* = \boldsymbol{J}^{*1}{\boldsymbol{J}^{*2}}^\top$.
   (b) For the proof of the theorem, we will use the fact that

$$\boldsymbol{H}^{*\mu} = \boldsymbol{P}^\mu_{k^*}\boldsymbol{A}^\mu_{k^*} + \boldsymbol{P}^{\bar{\mu}}_{k^*}\bar{\boldsymbol{A}}^\mu_{k^*} \in \mathbb{R}^{n\times k},$$

   where $\boldsymbol{P}^\mu_{k^*} \in \mathbb{R}^{n\times k^*}$, $\boldsymbol{P}^{\bar{\mu}}_{k^*} \in \mathbb{R}^{n\times (k-k^*)}$ and $\boldsymbol{A}^\mu_{k^*} \in \mathbb{R}^{k^*\times k^*}$, $\bar{\boldsymbol{A}}^\mu_{k^*} \in \mathbb{R}^{(k-k^*)\times k^*}$.

As $\boldsymbol{F}^* = \boldsymbol{H}^{*1}{\boldsymbol{H}^{*2}}^\top$ is the solution of eq.(7.13), we have

$$\begin{aligned}&\phi\Big(\boldsymbol{M}(\boldsymbol{H}^{*1}{\boldsymbol{H}^{*2}}^\top) - \boldsymbol{Y}\Big) + \gamma_1 \operatorname{tr}({\boldsymbol{H}^{*1}}^\top f(\hat{\boldsymbol{L}}^1)\boldsymbol{H}^{*1}) + \gamma_2 \operatorname{tr}({\boldsymbol{H}^{*2}}^\top f(\hat{\boldsymbol{L}}^2)\boldsymbol{H}^{*2}) \leq \\ &\phi(\boldsymbol{\eta}) + \gamma_1 \operatorname{tr}({\boldsymbol{J}^{*1}}^\top f(\hat{\boldsymbol{L}}^1)\boldsymbol{J}^{*1}) + \gamma_2 \operatorname{tr}({\boldsymbol{J}^{*2}}^\top f(\hat{\boldsymbol{L}}^2)\boldsymbol{J}^{*2})\end{aligned} \tag{A.23}$$

Now using the facts 3b and the eigen gap condition we obtain the following two:

$$\begin{aligned}
&\mathrm{tr}(\boldsymbol{H}^{*1^\top} f(\hat{\boldsymbol{L}}^1)\boldsymbol{H}^{*1}) = \mathrm{tr}(\boldsymbol{A}^1_{k^*} f(\boldsymbol{\Lambda}^1_{k^*})\boldsymbol{A}^{1\ \top}_{k^*}) + \mathrm{tr}(\bar{\boldsymbol{A}}^1_{k^*} f(\bar{\boldsymbol{\Lambda}}^1_{k^*})\boldsymbol{A}^{\bar{1}\ \top}_{k^*}) \\
&\geq f(\lambda^1_{k^*+1})\|\bar{\boldsymbol{A}}^1_{k^*}\|^2_F = f(\lambda^1_{k^*+1})\|\boldsymbol{P}^{\bar{1}\ \top}_{k^*}\boldsymbol{H}^{*1}\|^2_F
\end{aligned} \tag{A.24}$$

and similarly,

$$\mathrm{tr}(\boldsymbol{H}^{*2^\top} f(\hat{\boldsymbol{L}}^2)\boldsymbol{H}^{*2}) = f(\lambda^2_{k^*+1})\|\boldsymbol{P}^{\bar{2}\ \top}_{k^*}\boldsymbol{H}^{*2}\|^2_F \tag{A.25}$$

Now, using the fact 3a we get

$$\mathrm{tr}(\boldsymbol{J}^{*1^\top} f(\hat{\boldsymbol{L}}^1)\boldsymbol{J}^{*1}) \leq f(\lambda^1_{k^*})\|\boldsymbol{J}^{*1}\|^2_F \tag{A.26}$$

and

$$\mathrm{tr}(\boldsymbol{J}^{*2^\top} f(\hat{\boldsymbol{L}}^2)\boldsymbol{J}^{*2}) \leq f(\lambda^2_{k^*})\|\boldsymbol{J}^{*2}\|^2_F \tag{A.27}$$

using all the above bounds in eq.(A.23) yields

$$\begin{aligned}
&\phi(\boldsymbol{M}(\boldsymbol{F}^*) - \boldsymbol{Y}) + \gamma_1 f(\lambda^1_{k^*+1})\left\|\boldsymbol{P}^{\bar{1}\ \top}_{k^*}\boldsymbol{H}^{*1}\right\|^2_F + \gamma_2 f(\lambda^2_{k^*+1})\left\|\boldsymbol{P}^{\bar{2}\ \top}_{k^*}\boldsymbol{H}^{*2}\right\|^2_F \leq \\
&\phi(\boldsymbol{\eta}) + \gamma_1 f(\lambda^1_{k^*})\|\boldsymbol{J}^{*1}\|^2_F + \gamma_2 f(\lambda^2_{k^*})\|\boldsymbol{J}^{*2}\|^2_F,
\end{aligned} \tag{A.28}$$

for our choice of γ_1 and γ_2 this yields eq.(7.14).

□

Bibliography

[1] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[2] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[3] B. Jiang, C. Ding, and J. Tang, “Graph-laplacian PCA: Closed-form solution and robustness,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3492–3498.

[4] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[5] D. Cai, X. He, J. Han, and T. S. Huang, “Graph regularized nonnegative matrix factorization for data representation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1548–1560, 2011.

[6] Z. Zhang and K. Zhao, “Low-rank matrix approximation with manifold regularization,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 7, pp. 1717–1729, 2013.

[7] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst, “Accelerated filtering on graphs using lanczos method,” *arXiv preprint arXiv:1509.04537*, 2015.

[8] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, “Tensor decompositions for signal processing applications: From two-way to multiway component analysis,” *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 145–163, 2015.

[9] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle, “A multilinear singular value decomposition,” *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[11] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[12] L. R. Tucker, “Implications of factor analysis of three-way matrices for measurement of change,” *Problems in measuring change*, vol. 122137, 1963.

[13] ——, “The extension of factor analysis to three-dimensional matrices,” *Contributions to mathematical psychology*, vol. 110119, 1964.

[14] ——, “Some mathematical notes on three-mode factor analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[15] J. Levin, “Three-mode factor analysis.” *Psychological Bulletin*, vol. 64, no. 6, p. 442, 1965.

[16] D. Goldfarb and Z. Qin, “Robust low-rank tensor recovery: Models and algorithms,” *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 1, pp. 225–253, 2014.

[17] P. Shah, N. Rao, and G. Tang, “Sparse and low-rank tensor decomposition,” in *Advances in Neural Information Processing Systems*, 2015, pp. 2548–2556.

[18] H. Rauhut, R. Schneider, and Z. Stojanac, “Low rank tensor recovery via iterative hard thresholding,” *arXiv preprint arXiv:1602.05217*, 2016.

[19] R. Tomioka, K. Hayashi, and H. Kashima, “Estimation of low-rank tensors via convex optimization,” *arXiv preprint arXiv:1010.0789*, 2010.

[20] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, 2013.

[21] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[22] L. J. Grady and J. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science and; Business Media, 2010.

[23] X. Niyogi, “Locality preserving projections,” in *Neural information processing systems*, vol. 16. MIT, 2004, p. 153.

[24] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[25] X. He, D. Cai, and P. Niyogi, “Tensor subspace analysis,” in *Advances in neural information processing systems*, 2005, pp. 499–506.

[26] X. Li, S. Lin, S. Yan, and D. Xu, “Discriminant locally linear embedding with high-order tensor data,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 342–352, 2008.

[27] Y.-L. Chen and C.-T. Hsu, “Multilinear graph embedding: Representation and regularization for images,” *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 741–754, 2014.

[28] Y. Gao, X. Wang, Y. Cheng, and Z. J. Wang, “Dimensionality reduction for hyperspectral data based on class-aware tensor neighborhood graph and patch alignment,” *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1582–1593, 2015.

[29] W. Hu, J. Gao, J. Xing, C. Zhang, and S. Maybank, “Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking,” 2016.

[30] E. Candes and B. Recht, “Exact matrix completion via convex optimization,” *Communications of the ACM*, vol. 55, no. 6, pp. 111–119, 2012.

[31] E. J. Candes and Y. Plan, “Matrix completion with noise,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[32] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, “Recommender systems with social regularization,” in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 287–296.

[33] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, “Matrix completion on graphs,” *arXiv preprint arXiv:1408.1717*, 2014.

[34] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, “Collaborative filtering with graph information: Consistency and scalable methods,” in *Advances in Neural Information Processing Systems*, 2015, pp. 2107–2115.

[35] S. Gandy, B. Recht, and I. Yamada, “Tensor completion and low-n-rank tensor recovery via convex optimization,” *Inverse Problems*, vol. 27, no. 2, p. 025010, 2011.

[36] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, “Scalable tensor factorizations for incomplete data,” *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[37] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[38] C. Da Silva and F. J. Herrmann, “Hierarchical tucker tensor optimization-applications to tensor completion,” in *Proc. 10th International Conference on Sampling Theory and Applications*, 2013.

[39] Y. Xu, R. Hao, W. Yin, and Z. Su, “Parallel matrix factorization for low-rank tensor completion,” *arXiv preprint arXiv:1312.1254*, 2013.

[40] H. Tan, B. Cheng, W. Wang, Y.-J. Zhang, and B. Ran, “Tensor completion via a multi-linear low-n-rank factorization model,” *Neurocomputing*, vol. 133, pp. 161–169, 2014.

[41] E. Frolov and I. Oseledets, “Tensor methods and recommender systems,” *arXiv preprint arXiv:1603.06038*, 2016.

[42] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” *arXiv preprint arXiv:1405.3866*, 2014.

[43] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up convolutional neural networks using fine-tuned cp-decomposition,” *arXiv preprint arXiv:1412.6553*, 2014.

[44] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *Advances in Neural Information Processing Systems*, 2014, pp. 1269–1277.

[45] F. Huang and A. Anandkumar, “Convolutional dictionary learning through tensor factorization,” *arXiv preprint arXiv:1506.03509*, 2015.

[46] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, “Robust recovery of subspace structures by low-rank representation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 171–184, 2013.

[47] G. Liu and S. Yan, “Latent low-rank representation for subspace segmentation and feature extraction,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1615–1622.

[48] C. Jia, Y. Kong, Z. Ding, and Y. R. Fu, “Latent tensor transfer learning for rgb-d action recognition,” in *Proceedings of the 22nd ACM international conference on Multimedia.* ACM, 2014, pp. 87–96.

[49] M. Shao, C. Castillo, Z. Gu, and Y. Fu, “Low-rank transfer subspace learning,” in *Data Mining (ICDM), 2012 IEEE 12th International Conference on.* IEEE, 2012, pp. 1104–1109.

[50] R. Vidal and P. Favaro, “Low rank subspace clustering (lrsc),” *Pattern Recognition Letters*, vol. 43, pp. 47–61, 2014.

[51] M. Fazel, T. K. Pong, D. Sun, and P. Tseng, “Hankel matrix rank minimization with applications to system identification and realization,” *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 946–977, 2013.

[52] C. Georgakis, Y. Panagakis, and M. Pantic, “Dynamic behavior analysis via structured rank minimization,” *International Journal of Computer Vision*, pp. 1–25, 2017.

[53] R. Otazo, E. Candès, and D. K. Sodickson, “Low-rank plus sparse matrix decomposition for accelerated dynamic mri with separation of background and dynamic components,” *Magnetic Resonance in Medicine*, vol. 73, no. 3, pp. 1125–1136, 2015.

[54] S. Ravishankar, B. E. Moore, R. R. Nadakuditi, and J. A. Fessler, “Low-rank and adaptive sparse signal (lassi) models for highly accelerated dynamic imaging,” *arXiv preprint arXiv:1611.04069*, 2016.

[55] S. Gao, I.-H. Tsang, and L.-T. Chia, “Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 92–104, 2013.

[56] L. Tao, H. H. Ip, Y. Wang, and X. Shu, “Low rank approximation with sparse integration of multiple manifolds for data representation,” *Applied Intelligence*, pp. 1–17, 2014.

[57] T. Jin, J. Yu, J. You, K. Zeng, C. Li, and Z. Yu, “Low-rank matrix factorization with multiple hypergraph regularizers,” *Pattern Recognition*, 2014.

[58] F. Shang, L. Jiao, and F. Wang, “Graph dual regularization non-negative matrix factorization for co-clustering,” *Pattern Recognition*, vol. 45, no. 6, pp. 2237–2250, 2012.

[59] L. Grasedyck, D. Kressner, and C. Tobler, “A literature survey of low-rank tensor approximation techniques,” *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[60] M. Muja and D. Lowe, “Scalable nearest neighbour algorithms for high dimensional data,” 2014.

[61] J. Sankaranarayanan, H. Samet, and A. Varshney, “A fast all nearest neighbor algorithm for applications involving large point-clouds,” *Computers Graphics*, vol. 31, no. 2, pp. 157–174, 2007.

[62] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, “GSPBOX: A toolbox for signal processing on graphs,” *ArXiv e-prints*, #aug# 2014.

[63] B. N. Parlett, *The symmetric eigenvalue problem.* SIAM, 1980, vol. 7.

[64] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users’ guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.* Siam, 1998, vol. 6.

[65] R. M. Larsen, “Lanczos bidiagonalization with partial reorthogonalization,” *DAIMI Report Series*, vol. 27, no. 537, 1998.

[66] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[67] S. Si, D. Shin, I. S. Dhillon, and B. N. Parlett, “Multi-scale spectral decomposition of massive graphs,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2798–2806.

[68] D. Zhou, J. Huang, and B. Schölkopf, “Learning with hypergraphs: Clustering, classification, and embedding,” in *NIPS*, vol. 19, 2006, pp. 1633–1640.

[69] X. Lu, Y. Wang, and Y. Yuan, “Graph-regularized low-rank representation for destriping of hyperspectral images,” *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 7, pp. 4009–4018, 2013.

[70] Y. Zheng, X. Zhang, S. Yang, and L. Jiao, “Low-rank representation with local constraint for graph construction,” *Neurocomputing*, vol. 122, pp. 398–405, 2013.

[71] H. Du, X. Zhang, Q. Hu, and Y. Hou, “Sparse representation-based robust face recognition by graph regularized low-rank sparse representation recovery,” *Neurocomputing*, vol. 164, pp. 220–229, 2015.

[72] Y.-L. Chen, C.-T. Hsu, and H.-Y. M. Liao, “Simultaneous tensor decomposition and completion using factor priors,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 3, pp. 577–591, 2014.

[73] C. Wang, X. He, J. Bu, Z. Chen, C. Chen, and Z. Guan, “Image representation using laplacian regularized nonnegative tensor factorization,” *Pattern Recognition*, vol. 44, no. 10, pp. 2516–2526, 2011.

[74] Y. M. Lui, J. R. Beveridge, and M. Kirby, “Action classification on product manifolds,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 833–839.

[75] Y. M. Lui, “Human gesture recognition on product manifolds,” *Journal of Machine Learning Research*, vol. 13, no. Nov, pp. 3297–3321, 2012.

[76] C. Jia, G. Zhong, and Y. R. Fu, “Low-rank tensor learning with discriminant analysis for action classification and image recovery.” in *AAAI*, 2014, pp. 1228–1234.

[77] G. Zhong and M. Cheriet, “Large margin low rank tensor analysis,” *Neural computation*, vol. 26, no. 4, pp. 761–780, 2014.

[78] X. Li, M. K. Ng, G. Cong, Y. Ye, and Q. Wu, “Mr-ntd: Manifold regularization nonnegative tucker decomposition for tensor data dimension reduction and representation.”

[79] J. A. Tropp, “On the conditioning of random subdictionaries,” *Applied and Computational Harmonic Analysis*, vol. 25, no. 1, pp. 1–24, 2008.

[80] C. Boutsidis, M. W. Mahoney, and P. Drineas, “An improved approximation algorithm for the column subset selection problem,” in *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2009, pp. 968–977.

[81] R. Witten and E. Candes, “Randomized algorithms for low-rank matrix factorizations: sharp performance bounds,” *Algorithmica*, pp. 1–18, 2013.

[82] X. Li and J. Haupt, “Identifying outliers in large matrices via randomized adaptive compressive sampling,” *Signal Processing, IEEE Transactions on*, vol. 63, no. 7, pp. 1792–1807, 2015.

[83] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, “Fast randomized singular value thresholding for nuclear norm minimization,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4484–4493.

[84] M. Rahmani and G. Atia, “High dimensional low rank plus sparse matrix decomposition,” *arXiv preprint arXiv:1502.00182*, 2015.

[85] M. Rahmani and G. K. Atia, “Randomized robust subspace recovery for big data,” in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[86] W. Ha and R. F. Barber, “Robust PCA with compressed data,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1927–1935.

[87] M. Davenport, P. T. Boufounos, M. B. Wakin, R. G. Baraniuk *et al.*, “Signal processing with compressive measurements,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 2, pp. 445–460, 2010.

[88] A. Talwalkar and A. Rostamizadeh, “Matrix coherence and the nystrom method,” *arXiv preprint arXiv:1004.2008*, 2010.

[89] C. You, D. Robinson, and R. Vidal, “Scalable sparse subspace clustering by orthogonal matching pursuit,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2016.

[90] A. Aravkin, S. Becker, V. Cevher, and P. Olsen, “A variational approach to stable principal component pursuit,” *arXiv preprint arXiv:1406.1089*, 2014.

[91] N. Perraudin and P. Vandergheynst, “Stationary signal processing on graphs,” *ArXiv e-prints*, #jan# 2016.

[92] V. Kalofolias, “How to learn a graph from smooth signals,” in *th International Conference on Artificial Intelligence and Statistics AISTATS, Cadiz, Spain*, 2016.

[93] E. Pavez and A. Ortega, “Generalized Laplacian precision matrix estimation for graph signal processing,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6350–6354.

[94] T. Skauli and J. Farrell, “A collection of hyperspectral images for imaging systems research,” in *IS&amp;T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013, pp. 86 600C–86 600C.

[95] H. Eryilmaz, D. Van De Ville, S. Schwartz, and P. Vuilleumier, “Impact of transient emotions on functional connectivity during subsequent resting state: a wavelet correlation approach,” *Neuroimage*, vol. 54, no. 3, pp. 2481–2491, 2011.

[96] J. Richiardi, H. Eryilmaz, S. Schwartz, P. Vuilleumier, and D. Van De Ville, “Decoding brain states from fmri connectivity graphs,” *Neuroimage*, vol. 56, no. 2, pp. 616–626, 2011.

[97] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, “Face recognition using laplacianfaces,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 3, pp. 328–340, 2005.

[98] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[99] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Robust principal component analysis on graphs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2812–2820.

[100] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Applied and Computational Harmonic Analysis*, 2016.

[101] R. R. Nadakuditi, "Optshrink: An algorithm for improved low-rank signal matrix denoising by optimal, data-driven singular value shrinkage," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 3002–3018, 2014.

[102] Q. Gu, J. Zhou, and C. H. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs." in *SDM*. SIAM, 2010, pp. 199–210.

[103] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[104] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[105] E. C. Chi, G. I. Allen, and R. G. Baraniuk, "Convex biclustering," *arXiv preprint arXiv:1408.0856*, 2014.

[106] P. Ji, M. Salzmann, and H. Li, "Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4687–4695.

[107] N. Perraudin, D. Shuman, G. Puy, and P. Vandergheynst, "Unlocbox a matlab convex optimization toolbox using proximal splitting methods," *arXiv preprint arXiv:1402.0779*, 2014.

[108] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[109] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 60, no. 1, pp. 150–163, 2013.

[110] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast robust PCA on graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 740–756, 2016.

[111] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, "Compressive spectral clustering," in *Machine Learning, Proceedings of the Thirty-third International Conference (ICML 2016), June*, 2016, pp. 20–22.

[112] C. Elkan, "Using the triangle inequality to accelerate k-means," in *ICML*, vol. 3, 2003, pp. 147–153.

[113] O. Axelsson and G. Lindskog, "On the rate of convergence of the preconditioned conjugate gradient method," *Numerische Mathematik*, vol. 48, no. 5, pp. 499–523, 1986.

[114] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology identification from spectral templates," *arXiv preprint arXiv:1604.02610*, 2016.

[115] A. Narita, K. Hayashi, R. Tomioka, and H. Kashima, “Tensor factorization using auxiliary information,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 501–516.

[116] Y.-X. Wang, L.-Y. Gui, and Y.-J. Zhang, “Neighborhood preserving non-negative tensor factorization for image representation,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3389–3392.

[117] C. E. Tsourakakis, “Mach: Fast randomized tensor decompositions.” in *SDM*. SIAM, 2010, pp. 689–700.

[118] A.-H. Phan, P. Tichavskỳ, and A. Cichocki, “Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations,” *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4834–4846, 2013.

[119] S. Bhojanapalli and S. Sanghavi, “A new sampling technique for tensors,” *arXiv preprint arXiv:1502.05023*, 2015.

[120] J. H. Choi and S. Vishwanathan, “Dfacto: Distributed factorization of tensors,” in *Advances in Neural Information Processing Systems*, 2014, pp. 1296–1304.

[121] F. Huang, U. Niranjan, M. U. Hakeem, and A. Anandkumar, “Fast detection of overlapping communities via online tensor methods,” *arXiv preprint arXiv:1309.0787*, 2013.

[122] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, “Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 316–324.

[123] Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar, “Fast and guaranteed tensor decomposition via sketching,” in *Advances in Neural Information Processing Systems*, 2015, pp. 991–999.

[124] S. Gu, L. Zhang, W. Zuo, and X. Feng, “Weighted nuclear norm minimization with application to image denoising,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2862–2869.

[125] J. Paratte and L. Martin, “Fast eigenspace approximation using random signals,” *arXiv preprint arXiv:1611.00938*, 2016.

[126] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016, March) Tensorlab 3.0. [Online]. Available: http://www.tensorlab.net

[127] N. Shahid, N. Perraudin, G. Puy, and P. Vandergheynst, “Compressive pca for low-rank matrices on graphs,” *IEEE transactions on Signal and Information Processing over Networks*, 2016.

[128] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[129] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack, “Big and quic: Sparse inverse covariance estimation for a million variables,” in *Advances in Neural Information Processing Systems*, 2013, pp. 3165–3173.

[130] J. Ballani and D. Kressner, “Sparse inverse covariance estimation with hierarchical matrices,” tech. rep., EPFL Technical Report, Tech. Rep., 2014.

[131] K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst, “Song recommendation with non-negative matrix factorization and graph total variation,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2439–2443.

[132] D. Kressner, M. Steinlechner, and B. Vandereycken, “Low-rank tensor completion by riemannian optimization,” *BIT Numerical Mathematics*, vol. 54, no. 2, pp. 447–468, 2014.

[133] O. Koch and C. Lubich, “Dynamical tensor approximation,” *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 5, pp. 2360–2375, 2010.

[134] B. Huang, C. Mu, D. Goldfarb, and J. Wright, “Provable models for robust low-rank tensor completion,” *Pacific Journal of Optimization*, vol. 11, no. 2, pp. 339–364, 2015.

[135] M. Lustig, D. Donoho, and J. M. Pauly, “Sparse mri: The application of compressed sensing for rapid mr imaging,” *Magnetic resonance in medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[136] J. Trzasko and A. Manduca, “Highly undersampled magnetic resonance image reconstruction via homotopic l0 minimization,” *IEEE Transactions on Medical imaging*, vol. 28, no. 1, pp. 106–121, 2009.

[137] Y. Kim, M. Nadar, and A. Bilgin, “Wavelet-based compressed sensing using gaussian scale mixtures,” in *Proc. ISMRM*, 2010, p. 4856.

[138] M. Lustig, J. M. Santos, D. L. Donoho, and J. M. Pauly, “kt sparse: High frame rate dynamic mri exploiting spatio-temporal sparsity,” in *Proceedings of the 13th Annual Meeting of ISMRM, Seattle*, vol. 2420, 2006.

[139] J. P. Finn, K. Nael, V. Deshpande, O. Ratib, and G. Laub, “Cardiac mr imaging: State of the technology 1,” *Radiology*, vol. 241, no. 2, pp. 338–354, 2006.

[140] S. Posse, R. Otazo, S. R. Dager, and J. Alger, “Mr spectroscopic imaging: principles and recent advances,” *Journal of Magnetic Resonance Imaging*, vol. 37, no. 6, pp. 1301–1325, 2013.

[141] M. Descoteaux, R. Deriche, D. Le Bihan, J.-F. Mangin, and C. Poupon, “Multiple q-shell diffusion propagator imaging,” *Medical image analysis*, vol. 15, no. 4, pp. 603–621, 2011.

[142] D. Ma, V. Gulani, N. Seiberlich, K. Liu, J. L. Sunshine, J. L. Duerk, and M. A. Griswold, “Magnetic resonance fingerprinting,” *Nature*, vol. 495, no. 7440, pp. 187–192, 2013.

[143] J. He, Q. Liu, A. G. Christodoulou, C. Ma, F. Lam, and Z.-P. Liang, “Accelerated high-dimensional mr imaging with sparse sampling using low-rank tensors,” *IEEE transactions on medical imaging*, vol. 35, no. 9, pp. 2119–2129, 2016.

[144] N. Shahid, F. Grassi, and P. Vandergheynst, “Multilinear low-rank tensors on graphs and applications,” *arXiv preprint arXiv:1611.04835*, 2016.

[145] F. Knoll, K. Bredies, T. Pock, and R. Stollberger, “Second order total generalized variation (tgv) for mri,” *Magnetic resonance in medicine*, vol. 65, no. 2, pp. 480–491, 2011.

[146] F. Mahmood, N. Shahid, P. Vandergheynst, and U. Skoglund, “Graph-based sinogram denoising for tomographic reconstructions,” in *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*. IEEE, 2016, pp. 3961–3664.

[147] F. Mahmood, N. Shahid, U. Skoglund, and P. Vandergheynst, “Compressed sensing and adaptive graph total variation for tomographic reconstructions,” *arXiv preprint arXiv:1610.00893*, 2016.

[148] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[149] C. Tai, T. Xiao, X. Wang *et al.*, “Convolutional neural networks with low-rank regularization,” *arXiv preprint arXiv:1511.06067*, 2015.

NAUMAN SHAHID

PERSONAL INFORMATION

EPFL STI IEL LTS2,
ELE 227, Station 11,
CH-1015, Lausanne,
VD, Switzerland

Mobile: +41-788-99-66-59
Email: *naumanshahid6@gmail.com*
Personal Website: *https://lts2research.epfl.ch/blog/nauman/*
Google Scholar link:
https://scholar.google.ch/citations?user=dLM-zoUAAAAJ&hl=en

RESEARCH INTERESTS

Signal Processing on Graphs, Sparse Coding and Dictionary Learning for Graphs & Networks, Dimensionality Reduction, Low-rank Representation, Compressed Sensing, Convex Optimization, Biomedical Imaging, Data Mining, Machine Learning, Kernel Methods, Deep Learning (beginner)

EDUCATION

EPFL Doctoral School of Information & Communication Sciences, Switzerland

PhD student in Signal Processing Laboratory (LTS2) **September 2013 to present**

Doctoral Research

- **Thesis (in progress)**: "Principal Component Analysis on Graphs"
- **Advisor**: Prof. Dr. Pierre Vandergheynst (Vice President EPFL)
- **Exam date:** 15th July 2017.
- **Description**: The primary focus of my thesis is to develop scalable algorithms for recovering low-rank matrices & tensors lying on graphs.

Machine Learning & Signal Processing Based Collaborations & Projects

- Machine Learning Techniques for Dysphagia Detection
  In collaboration with Nestle R & D, Switzerland & Embedded Systems Laboratory, EPFL.
- Graph Based Techniques for Early Detection of Alzheimer's Disease
  In collaboration with University of Edinburgh, Dementia Research Centre
- Graph Based Techniques for Tomographic Reconstructions
  In collaboration with Skoglund Unit at Okinawa Institute of Science & Technology, Japan

Courses Taken

- Pattern Classification & Machine Learning, Big Data & Convex Optimization

Courses Taught

- Assisted in Advanced Signal Processing (theory + lab) involving signal processing based inverse problems, Continuous and Discrete Signals & Systems.

MS thesis supervision

- MS thesis of Ms. Xinrui Lyu on "Robust Local PCA on graphs" in May 2016.

Lahore University Of Management Sciences (LUMS), Lahore, Pakistan

M.Sc. Computer Engineering, CGPA: 3.68/4.0 **September 2010 to June 2012**

- Thesis: "Outlier Detection Techniques for Wireless Sensor Networks in Harsh Environments". Development of new One-class Support Vector Machine (SVM) based technique for outlier and event detection in Wireless Sensor Networks.
- Advisor: Dr. Ijaz Haider Naqvi

University Of Engineering & Technology (UET), Lahore, Pakistan

B.Sc. Electrical Engineering (Electronics & Communication)
CGPA: 3.834/4.0 (Honours), Rank: 10/232 **September 2006 to June 2010**

- Senior Year Thesis: "Frequency Synchronization algorithms for 802.11n based OFDM-MIMO systems"
- Advisor: Dr. Asim Loan

WORK EXPERIENCE

Team Lead (Inference & Protocol Design Team) **April 2012 to August 2013**

Department of Electrical Engineering, LUMS School of Science & Engineering
Project: Safety assurance in high stress environments: Machine Learning based solution for anomaly & event detection using Wireless Sensor Networks in underground coal mines.

Lecturer **March 2012 to August 2013**

Department of Electrical Engineering, UMT(School of Engineering): Involved in teaching of undergraduate courses: Computer Organization & Architecture, Modern Microprocessor Systems & Digital System Design, Communications Systems, Signals & Systems

Research Assistant **September 2010 to March 2012**

ADCOM Lab, Department of Electrical Engineering, LUMS School of Science & Engineering

- Advisor: Dr. Ijaz Haider Naqvi
- Research on one-class machine learning algorithms for wireless sensor networks.

Google Scholar

http://scholar.google.com/citations?user=dLM-zoUAAAAJ

Journal Publications

- **N.Shahid**, N.Perraudin, G.Puy, P.Vandergheynst, Compressive PCA for Low-rank Matrices on Graphs, in: IEEE Transaction on Signal and Information Processing on Networks, Issue 99, **April 2017**.
- K.Smith, B.Ricaud, **N.Shahid**, S.Rhodes, J.M.Starr, A.Ibanez, M.A.Parra, J.Escudero, P. Vandergheynst, Locating Temporal Functional Dynamics of Visual Short-Term Memory Binding using Graph Modular Dirichlet Energy, In: Nature Scientific Reports, **January 2017**.
- **N.Shahid**, N.Perraudin, V.Kalofolias, G.Puy, P.Vandergheynst, Fast Robust PCA on Graphs, In: IEEE Journal of Selected Topics in Signal Processing, Volume: 10, Issue: 4, **June 2016**.

Conference Publications

- **N.Shahid**, F. Mahmood, P. Vandergheynst U. Skoglund, Graph-based Sinogram denoising for Tomographic Reconstructions. In 38th IEEE Engineering in Medical & Biology Conference (EMBC), **August 16-20, 2016**, Orlando, FL.
- **N.Shahid**, N.Perraudin, V.Kalofolias, B.Ricaud, P.Vandergheynst, PCA using Graph Total Variation, In: IEEE International Conference on Acoustics, Speech and Signal Processing, **March 20-25, 2016**, Shanghai, China.
- **N.Shahid**, V.Kalofolias, X.Bresson, M.Bronstein, P.Vandergheynst, Robust Principal Component Analysis on Graphs, In: IEEE International Conference on Computer Vision (ICCV 2015). **December 11-17, 2015**, Santiago, Chile.

Submitted Papers

- **N.Shahid**, F.Grassi, P.Vandergheynst, Multilinear Low-rank Tensors on Graphs & Applications, **submitted to: IEEE ICCV 2017**.
- **N.Shahid**, F.Mahmood, U.Skoglund, P.Vandergheynst, Compressed Sensing and Adaptive Graph Total Variation for Tomographic Reconstructions, **submitted to: IEEE Transaction on Medical Imaging, 2017**.

Conferences, Workshops & Summer Schools

- Attended / presented at, BASP workshop 2017 (Villars-sur-Ollon, Switzerland), Graph Signal Processing Workshop in Philadelphia (May 2016), ICCV 2015 in Santiago (Chile), Signal Processing with Adaptive Sparse Structured Representations (SPARS 2015) Workshop in Cambridge UK, Machine Learning Summer School in Kyoto (2015), Graph Signal Processing Summer School in Leukerbad (Switzerland), Swiss Machine Learning Day 2015 in Lausanne, UCL-Duke Workshop on Sensing & Analysis of High-Dimensional Data (SAHD 2014), held in University College London, IEEE International Conference on Communication Systems 2012 (Singapore) and IEEE International Conference on Emerging Technologies 2011 (Islamabad, Pakistan).

Reviewer

- IEEE TSIPN, IEEE TIP, IEEE ICASSP, IEEE PIMRC, IEEE ICET, IEEE GLOBECOM, IEEE WCNC, Artificial Intelligence Review (Springer) and Journal of Electronic Imaging (JEI).

SCHOLARSHIPS

Received following scholarships for my PhD studies at University of Sydney and University of New South Wales, Australia.

- International Postgraduate Research Scholarship (IPRS)
- Australian Postgraduate Award (APA)
- Norman I Price Scholarship (NIP)
- 2013 Engineering International Tuition Fee & Stipend Support Scholarship (EITFS)

PROGRAMMING SKILLS

- MATLAB, C, Python (learning)

REFERENCES

Prof. Pierre Vandergheynst

- Full Professor at EPFL School of Electrical Engineering, Vice President EPFL (2017 onwards)
- Email: pierre.vandergheynst@epfl.ch

Dr. Benjamin Ricaud

- Owner of Evia Cybernetics
- Email: benjamin.ricaud@gmail.com

Dr. David Atienza Alonso

- Associate Professor at EPFL School of Electrical Engineering
- Email: david.atienza@epfl.ch