Imperial College London
Department of Electrical and Electronic Engineering

Non-convex Optimization Problems with Multiple Variables: Practical Applications and Learning Approaches

Jingyuan Xia

August 2020

Supervised by Doctor Imad Jaimoukha

Submitted in part fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
and the Diploma of Imperial College London

Declaration of Originality

I declare that this thesis and the research it contains are the product of my own work under the guidance of my thesis supervisor Doctor Imad Jaimoukha. All material that is not my own work has been properly acknowledged.

Jingyuan Xia

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Abstract

Optimization is pivotal for a wide range of technologies in scientific and industrial areas. In recent years, there has been an increasing interest in non-convex optimization in modern domains such as machine learning, signal processing and computer science. This thesis provides a deep insight into non-convex multi-variable optimization problems in the aspects of practical applications and solution approaches. We firstly present two applications which use non-convex optimization techniques on practical problems, where a bi-linear inverse model is proposed to solve an origin-destination (OD) flow estimation problem and a non-linear model is improved for solving multiple kernel clustering (MKC). In the light of optimizing the variables in the non-convex optimization problem on a global scope, a meta-learning based global scope optimization (GSO) approach is introduced for solving non-convex multi-variable problems. It performs significant improvements on accuracy than alternating minimization(AM)-based methods and shows .

First, we study the OD flow estimation problem to present that a significant dimension reduction is achieved by reformulating the previous NP-hard OD flow estimation problem as a bi-linear inverse problem. Specifically, a new forward model is developed which does not involve OD flows directly but is built upon the flows characterized only by their origins, henceforth referred as O-flows. The new O-flow model preserves all the OD information and more importantly reduces the dimension of the inverse problem substantially. An AM-based Gauss-Seidel method is deployed to solve the inverse problem, and the necessary condition for the uniqueness of the solution is derived. Simulations demonstrate that blind estimation, where no prior information is available,

is possible. Some challenging network settings are identified and discussed, where a remedy based on temporal patterns of the O-flows is developed and numerically shown effective.

We then study a non-linear non-convex multi-variable problem that is MKC problem. In this work, we improve the existing non-convex optimization model by incorporating an extra regularization term to provide finer structural constraints, therefore rising the accuracy and efficiency on clustering performance. Specifically, we propose a Localized Incomplete Multiple Kernel k-means with Matrix-induced Regularization (LI-MKKM-MR) model to solve MKC problem with taking the correlation among incomplete base kernels into account. The proposed model effectively reduces the redundancy and increases the diversity of the obtained kernel matrices. This improves the imputation of incomplete kernel matrices, and thus offers better clustering performance. A three-step optimization algorithm is proposed for solving LI-MKKM-MR, and the convergence is proved. We also carry out the theoretical analysis on the generalization error bound of the proposed LI-MKKM-MR. Extensive experiments on multiple benchmark data sets for MKC have been proposed to evaluate LI-MKKM-MR.

At last, we propose a new meta-learning approach, GSO approach, for solving non-convex optimization problems. Typically, the AM-based approaches are widely applied to optimize non-convex optimization problem with multiple variables. It splits the overall problem into several sub-problems corresponding to each variable, then it finds the optimum by iterating between the sub-problems. However, due to the intrinsic non-convexity, the stationary solutions of overall problem are not universally guaranteed, even when each sub-problem can be globally optimized at each iteration. Instead of exhaustively minimizing each sub-problem as what the AM strategy does, our meta-learning model explores a less greedy but more global scope strategy that intends to optimize variables based on its corresponding sub-problem and overall problem integrally. This is achieved via replacing the variable updating functions by neural

networks, whose parameters are updated with respect to minimizing the accumulated global losses through backpropagation. Specifically, the proposed meta-learning model exploits Long Short-Term Memory (LSTM) networks as meta-learners, which are updated in two levels of meta-learning. We evaluate the proposed model based on extensive simulations. The results verify that our proposed approach outperforms the exiting AM-based methods in standard cases, and is able to find the optimum of the overall problem while the AM strategy typically fails in some challenging cases.

Acknowledgements

First and foremost, I would like to thank my doctoral thesis advisor Dr. Imad Jaimoukha for his guidance, supervision and support not only on research but on all aspects. He has given me maximum freedom and full support. I am sincerely grateful for his consistent belief, trust and countless wise advice during my PhD study, especially during the difficult times. The research works would not even be possible without his vision, help and encouragement.

I would like to express my gratitude to my PhD examiners Prof. Deniz Gunduz from Imperial College London and Prof. Hai Xiang Lin from University of Delft University of Technology and Leiden University for their constructive and valuable feedback and suggestions.

I would like to thank my friends Austin, Darren, Xiaoye, Yiche for their support and care across four years and all over the world. I would like to thank my friends in CSP Cheng, Hamdi, Hengyan, Mortaz, Roxana for their help and support. I would like to thank my friends Junjie, Mohammad, Qi, Shengxi, Shuaifeng, Tianrui for their support and edification in the academy and living. I would like to thank Cioran for offering inspirations and motivations at the beginning of my PhD works. I would like to thank Nan for his support in gym and in my PhD life, especially those cafe time around Holy Trinity Brompton church. I would like to thank Xibo for the three years memorable and enjoyable apartment-sharing experience, it is my best luck to meet him in this city. I would like to thank Ruochen for the indelible moving and heartfelt care since the first day I arrive London. All of you have made my PhD journey fabulous and unforgettable.

I would like to thank my parents for their constant and unconditional support and love. I would like to thank my master supervisor Pro.Xiang Li for his constant care and support to help me go through my whole PhD process. I would like to thank my co-author Prof. Xinwang Liu for his support and guidance on my research. I would like to thank my buddy Billy for persistent friendship and heartwarming strong support. Last but not least, I would like to thank my darling Hong for always being by my side during every ups and downs of my PhD study. I would never be able to reach that far without you.

Jingyuan Xia, *London, UK.*

Contents

1 Introduction **27**

1.1 Motivations 27

1.2 Outline of the Thesis 31

1.3 Publications 33

2 Background **35**

2.1 Non-convex Optimization 36

2.1.1 Multi-variable Non-convex Optimization 37

2.1.2 Non-convex Optimization Approaches 42

2.2 Practical Applications 46

2.2.1 Origin-Destination Flow Estimation 47

2.2.2 MKC 53

2.3 Meta-learning for Optimization 60

2.3.1 Fundamental Basics 61

2.3.2 Learning Principle 63

2.3.3 An Example of Meta-learning for Image Classification 65

2.3.4 Related Works and Expansion 66

3 Blind Estimation for Origin-Destination Flows **69**

3.1 Introduction 69

3.2 A New Forward Model for OD Estimation 72

3.2.1 O-flow Mapping Models 73

3.2.2 Connections and Differences of Models 75

3.3 Estimation of OD-flows 78

3.3.1 Blind Estimation Model 78

3.3.2 Uniqueness of Solution 84

3.3.3 Estimation with the Sparsity Assumption . . . 89
3.4 Simulations and Results . . . 90
3.4.1 Simulation Setting . . . 91
3.4.2 Blind Estimation is Feasible . . . 94
3.4.3 Sparsity Improves the Estimation Capacity . . . 97
3.5 Summary . . . 97

4 Localized Incomplete Multiple Kernel Clustering 100
4.1 Introduction . . . 100
4.2 The Proposed LI-MKKM-MR . . . 102
4.2.1 Formulations . . . 102
4.2.2 Three-step Alternative Algorithm for LI-MKKM-MR . . . 103
4.2.3 Theoretical Analysis . . . 107
4.3 Simulation Results . . . 110
4.3.1 Experimental Settings . . . 110
4.3.2 Experimental Results . . . 112
4.3.3 Parameter Sensitivity of LI-MKKM-MR . . . 121
4.4 Summary . . . 122

5 Learning to Optimize Non-convex Multi-variable problems 124
5.1 Introduction . . . 124
5.2 Overall Structure of the GSO Method . . . 128
5.3 GSO Method with Hierarchical LSTM-based Mata-learners . . . 131
5.4 Applications in Typical Problems . . . 137
5.4.1 Bi-linear Inverse Problem: Matrix Completion . . . 137
5.4.2 Non-linear Problem: Gaussian Mixture Model . . . 139
5.5 Experiments . . . 141
5.5.1 Numerical Results on Matrix Completion . . . 142
5.5.2 Numerical Results on Gaussian Mixture Model . . . 150
5.6 Summary . . . 153

6 Conclusion 154
6.1 Edification . . . 154
6.2 Future research . . . 157

Bibliography **158**

List of Tables

4.1 Ten applied datasets. 112

4.2 Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Flower17 and Flower102 datasets. 112

4.3 Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Caltech102 data sets. 115

4.4 Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on UCI-Digital data set. 119

4.5 Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Protein Fold Prediction dataset. 119

4.6 Running time of the compared algorithms on various benchmark datasets (in seconds) 120

5.1 RMSE of rank-5 matrix completion on different observation rates . . . 146

5.2 RMSE of matrix completion on different ranks 147

5.3 RMSE of rank-10 matrix completion on different choices of k_m 147

5.4 RMSEs of mixed matrix completion on different choice of k_m 149

5.5 High dimensional GMM simulation results 150

List of Figures

2.1 Computational graph of general meta-learner. 63

2.2 Example meta-learning set-up for few-shot image classification, visual adapted from Ravi *et al.* [1] 65

3.1 An example of unidirectional network with nine nodes and twelve links. 76

3.2 A three-node unidirectional network 86

3.3 The 3-by-3 unidirectional network 91

3.4 The 3-by-3 bidirectional network. 92

3.5 The 8-by-8 bidirectional network. 92

3.6 The GEANT network. 92

3.7 Histograms of relative error in OD flow estimation for multiple step model. 96

3.8 Estimation without using sparsity. 97

3.9 Estimation using sparsity 98

4.1 Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Flower17 and Flower102 datasets. For each given missing ratio, the ”incomplete patterns" are randomly generated for 10 times and their averaged results are reported. 113

4.2 Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Caltech102-20, Caltech102-25 and Caltech102-30. For each given missing ratio, the “incomplete patterns" are randomly generated for 10 times and their averaged results are reported. The results on Caltech102-5, Caltech102-10 and Caltech102-15 are provided in the supplemental material due to space limit. 114

4.3 Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Caltech102-5, Caltech102-10 and Caltech102-15. For each given missing ratio, the “incomplete patterns" are randomly generated for 10 times and their averaged results are reported. 117

4.4 Clustering accuracy, NMI and purity comparison with the variation of missing ratios on UCI-digital dataset. For each given missing ratio, the “incomplete patterns" are randomly generated for 10 times and their averaged results are reported. 118

4.5 Clustering accuracy, NMI and purity comparison with the variation of missing ratios on protein Fold Prediction dataset. For each given missing ratio, the “incomplete patterns" are randomly generated for 10 times and their averaged results are reported. 120

4.6 The sensitivity of LI-MKKM-MR with the variation of λ and τ. 122

5.1 There are two levels of meta-learning in our proposed approach. The first level of meta-learning is performed within each single task $F_i(\boldsymbol{W}, \boldsymbol{X})$ which consists of a sequence of sub-problems. The second level of meta-learning is performed across a set of training tasks $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^{N}$. At the first and the second level, our model learns to achieve a fast and high quality minimization for the current optimization problem and all optimization problems, respectively. 128

5.2 In the AM-based model, there are only information flows of variables inside the corresponding inner loops while the outer loops are isolated from inner loops optimizations. In contrast, our GSO model builds a new connection between inner and outer loops, which allows the information flows of variables circulate among these two level loops. We use solid edges to denote information flow along arrow, and dashed edges indicate there is no information flow. 136

5.3 Performance of different choices on t_i and t_o ranging from 1 to 20. . . . 145

5.4 Averaged log(RMSE) Curves of three methods from 100 simulations in 20% observation rate. 146

5.5 Averaged RMSE Curves of three methods from 100 simulations on different choices of k_m 148

5.6 EM and GSO performance on 50 samples. Averaged Non-negative log-likelihood of 50 trails through EM algorithm is 7.77, while the result through GSO model is 7.56. 151

5.7 Clustering results on one sample. 151

5.8 EM and GSO performance on 10 flower-shaped data. Black dash lines denote EM clustering results; Red solid lines denote GSO clustering results. 151

Nomenclature

List of Acronyms

ADAM	A Method for Stochastic Optimization
ALS	Alternating Least Square
AM	Alternating-Minimization
BPTT	backpropagation through time
DCT	Discrete Cosine Transform
DCT	Super-resolution
EM	Expectation Maximization
GAN	Generative Adversarial Networks
GMM	Gaussian Mixture Model
GSO	Global Scope Optimization
KNN	K-nearest-neighbor
LI-MKKM	Localized Incomplete Multiple Kernel K-means
LI-MKKM-MR	localized incomplete multiple kernel k-means with matrix-induced regularization
LSTM	Long-Short Term memory
ME	Maximum Entropy
MKC	Multiple Kernel Clustering
MKKM	Multiple Kernel K-means
MLL	Maximize the Log-likelihood
NMI	Normalized Mutual Information
OD	Origin-Destination

ONKC	Optimal Neighborhood Kernel Clustering
PCA	Principal Component Analysis
RL	Reinforcement Learning
RMSE	Relative Mean Square Error
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent

Notations

$\boldsymbol{X}$	Matrix: $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with $m, n > 1$
$\boldsymbol{x}$	Vector: $\boldsymbol{x} \in \mathbb{R}^{m}$ with $m > 1$
$\boldsymbol{X}^T$	Transpose of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{-1}$	Inverse of matrix of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{\perp}$	Orthogonal complement matrix $\boldsymbol{X}$
$\boldsymbol{X}^{\dagger}$	Pseudo-inverse of matrix of matrix $\boldsymbol{X}$
$\log(\cdot)$	Natural logarithm
$\|x\|$	Absolute value of a real number x
$\mathbb{R}$	The set of real numbers
$\mathbf{I}$	Identity matrix
$\mathbf{T}$	Toeplitz matrix
$\text{rank}(\cdot)$	Matrix rank
$\|\boldsymbol{x}\|_0$	l_0-norm, number of non-zero entries in $\boldsymbol{x}$
$\|\boldsymbol{x}\|_1$	l_1-norm: $\sum_i \|\boldsymbol{x}(i)\|$
$\|\boldsymbol{x}\|_2$	l_2-norm: $\sqrt{\sum_i \boldsymbol{x}(i)^2}$
$\|\boldsymbol{X}\|_F$	Frobenius norm: $\sqrt{\sum_i \boldsymbol{X}(i,j)^2}$
x_i	The i-th element of a vector $\boldsymbol{x}$
$x_{i,j}$	The element at the i-th row and the j-th column of a matrix $\boldsymbol{X}$

Chapter 1

Introduction

1.1 Motivations

Optimization is one of the most significant component in science and technology. Tremendous fields of practical applications such as transportation management in city, large dimensional data processing in finance, retail business, and artificial intelligence all rely on the developments of optimization mechanism. Optimization enables the real world problems to be modelled as mathematical formulations, thus variety of algorithms can be applied to make predictions or estimations in these problems.

Previous literature suggests that convex optimization problems have been focused largely in last decades, where variety of effective solutions and deep understanding of the geometric features on convex landscape are already well studied. However, not all realistic problems can be described in convex formulations, such as low-rank matrix recovery in signal processing, training non-linear models in machine learning and large dimensional data processing in deep learning. In these cases, non-convex optimization can often provide desirable benefits such as dimension reduction and benign landscape.

There are many applications that benefit from non-convex optimization techniques

such as recommendation systems, image classification, image recovery, and incomplete kernel clustering. A large amount of works have demonstrated in a heuristic way that even suffering from the intrinsic no closed form in general, non-convex optimization can achieve better performance on accuracy and efficiency than the convex counterpart. This might be explained by possessing specific constraints such as sparsity and low-rank and formulating the problem itself in non-convex formulation to capture additional structural features from data. Thus, the capability of modeling the practical problem as a non-convex optimization problem allows great flexibility to the algorithm designer to ensure more accurate and scalable solutions than convex optimization, especially in case of high-dimensional data processing such as deep neural networks and large dimensional data processing.

However, provably effective solutions for non-convex optimization are rare. Specifically, some non-convex optimization problems are known to be NP-hard, and thus the solutions are highly dependent on the structural constraints. Two main types of solutions have been intensively studied. The first kind is known as convex solution that converts the non-convex optimization into a convex relaxation. The relaxed convex problem shares common optimums with original problem, thus convex solutions are able to be implemented. However, these approaches generally have higher complexity, therefore are time-consuming and still challenging at large scales. In contrast, more recent works have proved that the non-convex solution that directly solves non-convex optimization problems in their original landscapes providing faster convergence and comparable accuracy even without closed form solutions. These popular heuristic methods, including projected gradient descent (PGD) and alternating minimization (AM), have achieved excellent performance on a wide range of applications such as dictionary learning, matrix completion, blind deconvolution and kernel clustering.

In this thesis, we focus on non-convex optimization problems with two or multiple variables. We classify these problems into two types: bi-linear inverse problems and

non-linear problems. The first kind of problems have a linear mapping between the variables and the observation data, such as matrix completion, blind deconvolution and sparse recovery. The non-linear counterpart usually models the problem itself in non-linear form or contains non-linear structural constraints, including Gaussian mixture model (GMM) problems and kernel clustering problems. Not only there are still inadequate applications of non-convex optimization with multiple variables in many practical scenarios, the understanding of intrinsic methodology in non-convex solutions are lacking; thus, the advance of effective solutions is limited as well. In this instance, we investigate two aspects of non-convex multi-variable optimization problems: beneficial applications in practical problems and new solution method. We summarize our major contributions in this thesis in the following:

- The first aim of this thesis considers practical problems by conducting non-convex optimization. Two applications that can benefit from adopting non-convex optimization techniques are presented. In the first application, motivated by the capacity of dimension reduction of non-convex optimization mechanism, we propose a non-convex and bi-linear model for solving the origin-destination (OD) flow estimation problem, which is essential to many transportation management and network analysis tasks. We prove that, for the first time, the original ill-posed OD flow estimation problem can be turned into a well-posed problem with the proposed non-convex model. Thus, the ground truth OD flows can be uniquely identified without any prior information. One of the most widely used non-convex optimization approaches, alternating minimization (AM), is applied to solve this non-convex optimization problem. The second application focuses on solving multiple kernel clustering (MKC) problems. In this part, we study the non-convex and non-linear models for MKC problems and present a improved localized incomplete multiple kernel k-means with matrix-induced regularization (LI-MKKM-MR) model. The proposed model provide better performance by incorporating extra variables and structural constraints on the non-convex model

for further utilizing intra-feature of the observation data. Consequently, we design a three-step iterative algorithm, and analyze its convergence with the proof of bounded generalization error. The simulation results show superior performance when compared with the state-of-the-art approaches.

- The second objective of this thesis is to acquire a deeper understanding of the methodologies used in existing non-convex optimization approaches, and to design a new solution method to address the problematic issues in previous approaches. Based on the analysis on the AM-approaches, we demonstrate that previous non-convex approaches are typically determined by a specific fixed updating rules on variables. These hand-designed algorithms with fixed updating rules generally optimize the variables locally, and the global loss information is seldom integrated into the updating functions. In order to address this limitation, we propose a learning-based method for non-convex optimization problems. In the light of the recent success of meta-learning in optimization, we propose a meta-learning based approach for solving non-convex multi-variable optimization problems. The proposed method is name as the global scope optimization (GSO) method. It adaptively optimizes each variable with respect to the variations of global loss over iterations. Therefore, the GSO method provides global scope information over different optimization iterations for different variables, while the previous methods typically perform local operations for variable updates. We evaluate our GSO method on a series of non-convex optimization problems including matrix completion and GMM problems. The proposed GSO method outperforms previous methods in intensive simulations, and is able to achieve effective optimization in challenging cases, while standard methods would typically fail.

1.2 Outline of the Thesis

The remainder of this thesis is organised as follows:

In Chapter 2, we will first give an introduction to non-convex optimization, especially the problems with multiple variables. Bi-linear inverse problems and non-linear problems are two main representative categories of non-convex multiple variables problems. We start with the problem formulation and then propose a brief review on the existing methods for these two types of problems, respectively. Then, OD flow estimation and MKC are introduced as the practical applications corresponding to bi-linear and non-linear problems. We will illustrate the existing approaches and inherent limitations in previous works of these two problems. Finally, the meta-learning based methods for optimization will be introduced. The working principles and the different meta-learning models for optimizations will be reviewed.

In Chapter 3, we propose a non-convex bi-linear model for OD estimation problem. The main contribution of this work is, for the first time, we show the possibility of accurately estimating OD flows entirely based on the observed link flows, without any prior information. This is achieved by reformulating the original ill-posed problem as a non-convex formulation that is turned into a well-posed problem. The new model does not directly involves OD flows but preserves all the OD information and more importantly reduces the dimension of the bi-linear inverse problem in orders of magnitude. For the case that the network topology does not support a unique solution, we assume that the OD flows exhibit some time domain patterns with unknown parameters. These structural constraints can lead to accurate estimation of the parameters to describe time-domain patterns in the OD flows and the OD flows. Inspired by the terminology blind deconvolution in signal processing community, we refer to the estimation without prior knowledge as blind estimation. Algorithms for blind estimation, a necessary condition for exact recovery, and a sparsity constraints for exact recovery

of particular traffic graph are provided. Simulations results demonstrate the capability of our proposed model, where accurate blind estimations for OD flows in several classic evaluation networks are successful.

In Chapter 4, an improved non-convex and non-linear model is proposed for solving MKC problems. The MKC problems are a kind of the representative non-convex optimization problem whose objective function generally is imposed with non-linear formulations. We propose a LI-MKKM-MR method to address the insufficient operations on diversity and the complementarity of incomplete base kernels in MFC problems. The proposed LI-MKKM-MR method improves the exiting models by incorporating a matrix-induced regularization term to handle the correlation among base kernels, which captures additional data structural to increase the capability of making the base kernels better utilized for clustering. Corresponding to the proposed model, we design a new three-step iterative algorithm. The derivations on generalization error bound and analysis on convergence of the new algorithm is also presented. Intensive simulations have been conducted on several multiple kernel learning benchmark datasets to evaluate the clustering performance of the proposed algorithm. The proposed LI-MKKM-MR method significantly and consistently outperforms the state-of-the-art approaches.

In Chapter 5, we propose a new meta-learning based method for solving non-convex multi-variable optimization problems. We will first show in this thesis that, minimize the global loss function, there exists a better descending step for each variable in a single iteration compared to the one obtained by exhaustively optimizing each variable locally and individually on each iteration. In other words, for a faster and better convergence in terms of the global loss function, the descending steps are not necessarily the optimal solution for each iteration step during the iterative optimization process. Our proposed method aims to achieve a global scope optimization. We thus name the proposed optimization strategy as GSO. Our GSO method automatically learns to appropriately optimize each variable under the guidance of the overall objective function. An adaptive

and global gradient descent rule is continuously learned through specific iterations and applied on all the down-stream optimization processes within each iteration. To achieve this, we build a two level meta-learning procedure in our model: The first level is to leverage previous optimization experiences across different iterations within one problem, and therefore this will enable the learned optimization strategy to extract the inter-information of different iterations; the second level meta-learning process leverages optimization experiences across different optimization problems in training sets, and hence it will enable to improve the adaptability and efficiency of learned algorithm for solving a class of problems instead of a specific one.

Finally in Chapter 6, we conclude the thesis by summarizing the achievements and providing possible future research directions.

1.3 Publications

The materials presented in this thesis have led to the following publications:

Peer-reviewed Journal

- Jingyuan Xia, Wei Dai, John Polak, and Michel Bierlaire, "Dimension Reduction for Origin-Destination Flows Estimation:Blind Estimation Made Possible," submitted to *Operations Research* and received a major revision, October 2019.
- Jingyuan Xia and Xinwang Liu, "Localized Incomplete Multiple Kernel k-means with Matrix-induced Regularization", submitted to *Transactions on Knowledge and Data Engineering* and received a major revision, July 2020.
- Jingyuan Xia, Shengxi Li, Jun-Jie Huang, Imad Jaimoukha, and Xinwang Liu, "Learning to Optimize Multi-variable Non-convex Optimization Problems:Iterating Non-optimums Makes Optimum Possible," submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*, June 2020.

The following publications are not directly related to the thesis but produced during my PhD:

- Liu, Xinwang, Miaomiao Li, Chang Tang, Jingyuan Xia, Jian Xiong, Li Liu, Marius Kloft, and En Zhu. "Efficient and effective regularized incomplete multi-view clustering." IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

- Tang, Chang, Xinwang Liu, Xinzhong Zhu, Jian Xiong, Miaomiao Li, Jingyuan Xia, Xiangke Wang, and Lizhe Wang. "Feature selective projection with low-rank embedding and dual Laplacian regularization." IEEE Transactions on Knowledge and Data Engineering (2019).

- Wang, Siwei, Xinwang Liu, En Zhu, Chang Tang, Jiyuan Liu, Jingtao Hu, Jingyuan Xia, and Jianping Yin. "Multi-view Clustering via Late Fusion Alignment Maximization." In IJCAI, pp. 3778-3784. 2019.

- Zhou, Sihang, En Zhu, Xinwang Liu, Tianming Zheng, Qiang Liu, Jingyuan Xia, and Jianping Yin. "Subspace segmentation-based robust multiple kernel clustering." Information Fusion 53 (2020): 145-154.

- Xia, J. Y., X. Li, and Y. X. Liu. "Application of a new restricted Boltzmann machine to radar target recognition." In 2016 Progress in Electromagnetic Research Symposium (PIERS), pp. 2195-2201. IEEE, 2016.

Chapter 2

Background

In this chapter, we first give an introduction to non-convex optimization problems focusing on bi-linear inverse problems and non-linear problems, respectively. The problem formulations and the representative applications are discussed and analyzed, and then the existing methods are presented. Two representative non-convex optimization problems, *matrix completion problem* and *Gaussian mixture model problem* (GMM), are briefly reviewed corresponding to bi-linear inverse problem and non-linear problem, respectively. In the second part, the background of two specific practical applications: the *OD estimation problem* and *MKC* will be reviewed, respectively. The problem descriptions, previous approaches and inherent drawbacks will be illustrated for each of them. Collectively, these existing investigations provide us a view of the necessities and benefits of adapting non-convex optimizations in real world scenarios. Finally, we will review meta-learning for optimization, including the basics of meta-learning, the learning principle and the existing works. Meta-learning has witnessed increasing importance in recent years. The key idea of meta-learning is to learn an algorithm that is capable of achieving fast adaption and good performances on a large number of new tasks. A number of techniques have been developed to utilize meta-learning for optimization.

2.1 Non-convex Optimization

An analytic optimization problem with single variable typically has the following form:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^p} f(\boldsymbol{x}) \\ &\text{s.t. } \boldsymbol{x} \in \mathcal{C}, \end{aligned} \tag{2.1.1}$$

where $\boldsymbol{x}$ denotes the *variable*, f denotes the *objective function*, and $\mathcal{C}$ denotes the *constraint set* of the optimization problem. An optimization problem is *convex* if both of the objective function f and the constraint set $\mathcal{C}$ are convex. On the other hand, either one of these terms is non-convex, the problem is a non-convex optimization problem. However, the progressively modern applications in domains such as machine learning, signal processing, and artificial intelligence are unable to be modelled in convex formulations. These applications either require the capability of large dimensional data processing, contain sparsity or low-rank non-convex constraints, or the natural settings on objectives are non-convex. Therefore, non-convex optimization has been increasingly significant in recent years due to the better performance on large dimensional data processing and the capacity of modelling more complicated practical problems.

The solution approaches for non-convex optimization can be classified into convex relaxation approaches and non-convex optimization approaches. The convex relaxation approaches typically reformulate the original non-convex problem as a convex problem with specific structural constraints. Nevertheless, these approaches are not only limited by the strict conditions, more important is that the relaxed convex models typically fail to perform efficiently at large scales. In contrast, non-convex approaches are more straightforward, which directly optimize the non-convex objective functions through heuristic algorithms. The existing non-convex methods such as PGD, AM-methods, the EM algorithm, stochastic optimization, and the variants have been widely used in many machine learning and signal processing tasks. However, the intrinsic non-

convexity and general NP-hardness in non-convex problems could lead to no guarantee on convergence and performance.

In this section, we focus on the non-convex problem with two or more variables and the corresponding non-convex optimization approaches. We will firstly present two representative non-convex multi-variable optimization problems, the matrix completion problem and the GMM problem, their practical applications and corresponding existing solution approaches. In this instance, we review the non-convex optimization approaches with respect to the AM-based strategy, where the methodology and limitations of AM-based methods are analyzed.

2.1.1 Multi-variable Non-convex Optimization

In machine learning, optimization problem that contains more than one variable with an underlying linear (e.g., product or convolution) or a non-linear (e.g., logarithmic operation, exponential kernel) transmission between variables, is of great importance but typically difficult to be well accommodated. A general formulation of non-convex optimization problem over an intersection of two variables can be expressed in the form of:

$$\min_{(\boldsymbol{W},\boldsymbol{X})\in\mathcal{W}\times\mathcal{X}} F(\boldsymbol{W},\boldsymbol{X}), \tag{2.1.2}$$

where $F : \mathcal{W}\times\mathcal{X} \to \mathbb{R}$ is a non-convex function with two variables $\boldsymbol{W} \in \mathcal{W}$ and $\boldsymbol{X} \in \mathcal{X}$. Depending on the different formulations of F, we sort the non-convex multi-variable problem as *bi-linear inverse problem* and *non-linear problem*. In the following two parts, we will illustrate bi-linear inverse problem with respect to matrix completion problem and non-linear problem with respect to Gaussian mixture model problem, respectively. For these multi-variable optimization problems, we define the original problem as the overall problem while the sub-problems refer to optimize one variable with the rest being fixed.

Bi-linear Inverse Problems

Bi-linear inverse problems have been widely applied in machine learning, signal processing and computer science, including low-rank matrix recovery [2–6], dictionary learning [7,8], and blind deconvolution [9–11]. Typically, a bi-linear inverse problem aims to estimate the variable pair $(\boldsymbol{W}, \boldsymbol{X})$ from the observed samples $\boldsymbol{Y}$ underlying a bi-linear mapping $\boldsymbol{F}(\boldsymbol{W}, \boldsymbol{X}) = \boldsymbol{Y}$. Most of these problems enjoy good landscapes where their second-order stationary points are all global optimums [2, 12]. The benign landscapes can be typically guaranteed by assuming certain constraints on variable sets, such as sparse constraint [13–15], semi-defined constraint [5, 15], and rank limitations [2, 12].

In this section, we take the matrix completion problem [4–6, 16, 17] as a representative case of bi-linear inverse problem to demonstrate the properties and corresponding non-convex optimization approaches. Matrix completion is a class of problems that aims to recover the missing entries in a target matrix [15–17] and has been widely applied to practical problems such as recommending system [6] and collaborative filtering [18]. Typically, it is cast as a low-rank matrix recovery problem in which the matrix to be completed is assumed to be low-rank with given rank information.

Matrix completion problem generally represents a low-rank matrix $\boldsymbol{R} \in \mathbb{R}^{p\times q}$, by a multiplication of two rectangular matrices as $\boldsymbol{U} \in \mathbb{R}^{p\times s}$ and $\boldsymbol{V} \in \mathbb{R}^{q\times s}$ [4, 18]. The optimization problem is then formulated as [19],

$$\min_{\boldsymbol{U},\boldsymbol{V}} F(\boldsymbol{U}, \boldsymbol{V}) := \frac{1}{2} \left\| \mathcal{P}_\Omega(\boldsymbol{R} - \boldsymbol{U}\boldsymbol{V}^T) \right\|_F^2 + \frac{\lambda}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2), \tag{2.1.3}$$

where the projection $\mathcal{P}_\Omega(\cdot)$ preserves the observed elements defined by Ω and replaces the missing entries with 0. The problem (2.1.3) is a typically bi-linear inverse problem where the objective is to estimate two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ in multiplications from the only observations $\boldsymbol{R}$.

It is straightforward that the overall problem $F(\boldsymbol{U}, \boldsymbol{V})$ is not convex in $\boldsymbol{U}$ and $\boldsymbol{V}$,

but the sub-problems are convex when fixing one variable and updating another one. Therefore, we split the problem (2.1.3) into two sub-problems in quadratic form with respect to fixing one variable in (2.1.3) and updating the other one, referring to $f_{\boldsymbol{U}}(\boldsymbol{V})$ and $f_{\boldsymbol{V}}(\boldsymbol{U})$. Typically, iterative solutions are widely applied to solve matrix completion problems by alternatively optimized $f_{\boldsymbol{U}}(\boldsymbol{V})$ and $f_{\boldsymbol{V}}(\boldsymbol{U})$ to their optimums through optimization courses.

The AM-based iterative methods often perform well on solving problem (2.1.3), such as alternating least square (ALS) [20] and stochastic gradient descent (SGD) [21], when variables are restricted by some constraints. Some constraints, e.g., rank of matrix $\boldsymbol{R}$, provide a good loss landscape geometry of problem (2.1.3), which makes convergence to a global optimum by AM-based methods become possible. At the same time, problem (2.1.3) can be converted into some convex relaxations used for low-rank matrix recovery problem [2, 3, 22]. In this way, it can be solved under a convex subspace. However, as we aforementioned, this is not universal. On the one hand, the good performance is dependent on variable initialization, the parameter choice of gradient descent algorithm, the sparsity level of low-rank matrix, and the choice of constraints. On the other hand, these methods assume some extra assumptions, such as sparsity constraints [14,15], while these are not common in practical scenarios. More analysis and illustrations on AM-based methods will be soon presented in Section 2.1.2.

In the above introduction on matrix completion problem, we infer that a bi-linear inverse problem is typically ill-posed and non-convex when there is no sufficient constraints. Considering the matrix completion problem in (2.1.3), it could be ill-posed when the low-rank constraints are no longer satisfied. In Chapter 3 and Chapter 5, we will further discuss the benefits of using non-convex techniques for solving ill-posed problems.

Non-linear Problems

Compare to the bi-linear inverse problems, a non-linear non-convex optimization problem is more difficult. These non-convex non-linear problems also play a significant role in statistical machine learning, including Bayesian models [23], graphical models [24, 25], and GMM [26]. In this section, we will go through GMM as a representative of non-convex non-linear multi-variable problems. The formulations and solution algorithms for GMM are reviewed.

GMM [27–29] is one of the most important probabilistic models in machine learning. It is usually constructed as a maximum likelihood problem over a set of Gaussian distributions in the form of weighted Gaussian component density [29]. The variables in GMM problem possess a non-linear mapping to the observations with respect to the Gaussian distributions. Specifically, given a set of N i.i.d samples $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$, each entry $\boldsymbol{x}_n$ is a D-dimensional data vector. Then, a typical optimization when using the GMM to model the samples is to maximize the log-likelihood (MLL) [29], which is equivalent to minimize the Kullback–Leibler divergence. The parameters of the GMM can then be optimized as follows,

$$\arg \max_{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \log p(\boldsymbol{X}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.1.4}$$

where

$$\mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{\exp\{-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)\}}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}}. \tag{2.1.5}$$

In (2.1.4), for the k-th Gaussian component, $\boldsymbol{\mu}_k$ is the mean vector defining the cluster centre, covariance $\boldsymbol{\Sigma}_k$ denotes the cluster scatter, and π_k represents mixing proportion, with $\sum_{k=1}^{K} \pi_k = 1$.

However, it is intractable to directly obtain a closed-form solution that maximizes $\log p(\boldsymbol{X})$ in (2.1.4). The key difficulty is that by differentiating $\log p(\boldsymbol{X})$ (summation

of logarithmic summation) and equalizing it to 0, each parameter is intertwined with each other. Gradient descent methods in an AM manner can alternatively solve problem (2.1.4) by iteratively finding first-order optimums. However, they typically perform inferior to the one called expectation-maximization (EM) algorithm [26, 27]. One possible reason is due to the non-convexity and non-linearity of the problem (2.1.4) that require a sophisticated strategy to find a good stationary point. On the other hand, EM algorithm omits the hyper-parameter related to the step size by converting the original ML estimation problem into a relaxed problem where a lower bound is maximized monotonically.

Let the parameters be compactly denoted by $\theta_k = \{\boldsymbol{\pi}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, and $\boldsymbol{\theta} = \{\theta_k\}_{k=1}^K$. Then, the EM algorithm solves the problem of (2.1.4) via relaxing it in terms of the lower bound as follows,

$$\arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} Q_{n,k} \log \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.1.6}$$

where

$$Q_{n,k} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.1.7}$$

is the posterior of the cluster index given $\boldsymbol{x}_n$ (or responsibility weight). This relaxation enables a closed-form solution towards each π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ when differentiating (2.1.6) and setting it to 0, respectively. Thus, EM algorithm iterates the following two steps until converge,

- Expectation step (E-step): For each $\boldsymbol{x}_n$, calculate $Q_{n,k}$ with the current parameters $\boldsymbol{\theta}$.
- Maximization step (M-step): Update parameters by differentiating (2.1.6) and setting it to 0, where the update for each parameter can be explicitly formulated.

It has been proved that by iterating E-step and M-step, the cost function increases

monotonously, thus reaching a stationary point eventually. For more details, we refer to [27, 30].

In summary, the non-convex multi-variable optimization problems, such as GMM, are typically solved by heuristic methods due to the uniqueness of solutions is not guaranteed. The most widely used solution strategy can be inferred as iterative methods which alternatively optimize each variable until the objective of the problem is satisfied. These methods, such as EM algorithm, perform well in wide range of applications even though a closed-form solution is not guaranteed. We should also point out that the EM algorithm still updates the GMM parameters in an AM manner, i.e., iterating to optimum over each parameter at which its gradient equals to 0 with the other parameters being fixed. Furthermore, the EM algorithm, together with other first-order methods, has been proved to converge to arbitrary bad local optimum almost surely [31]. As we have discussed before, the AM strategy can be improved by replacing the fixed updating rule that searches optimum in a local landscape, by a less-greedy rule that updates variables up to global scope knowledge on loss landscape of global objective function. We will further analyze AM-based methods in next section and discuss the less-greedy strategy in Chapter 5.

2.1.2 Non-convex Optimization Approaches

As aforementioned, existing literature optimizes over multiple variables by using an AM strategy which converts the original problem into a sequence of iterative optimizations on a set of sub-problems corresponding to each variable. In this section, we will illustrate the AM-based approaches in the perspectives of the solution strategy, basic algorithm and inherent limitations to elaborate their prominent properties and drawbacks in methodology.

AM based approaches [32–37] have been widely applied to solve optimization problems

with two or multiple variables. In fact, most of the approaches for solving the bi-linear and non-linear problems adopt an AM-based strategy, of which the basic idea is to sequentially optimize the sub-problems corresponding to one variable while keeping the other variable fixed to their optimums. That is, starting with an arbitrary initialization $\boldsymbol{W}_0 \in \mathcal{W}$, the AM-based algorithm sequentially solves the two sub-problems at the n-th iteration via:

$$\begin{aligned} \boldsymbol{X}_n &= \arg\min_{\boldsymbol{X}\in\mathcal{X}} f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X}), \\ \boldsymbol{W}_n &= \arg\min_{\boldsymbol{W}\in\mathcal{W}} f_{\boldsymbol{X}_n}(\boldsymbol{W}), \end{aligned} \tag{2.1.8}$$

where $f_{\boldsymbol{W}_{n-1}} = F(\boldsymbol{W}_{n-1}, \boldsymbol{X})$, and $f_{\boldsymbol{X}_n} = F(\boldsymbol{W}, \boldsymbol{X}_n)$ with respect to the problem 5.1.1.

Before proceeding further, we first clarify some important concepts that will be used throughout this thesis. We define the *overall problem* as the optimization problem with two variables over a linear or non-linear algebra space with objective function $F(\boldsymbol{W}, \boldsymbol{X})$, which is also called as global loss function, and define the *sub-problem* as the optimization problem over each single variable, i.e., with objective functions $f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_n}(\boldsymbol{W})$ for $n \geq 1$, which are also denoted as local loss functions.

We show the general structure of AM based methods in Algorithm 1. More specifically, there are two levels of loops when iterating. Let the variables update steps (step 3 to 6 and 9 to 12) be the inner loops, each outer loop contains two inner loops and the related steps for updating the fixed variables of sub-problems. At the inner loops, each variable is updated using an updating function ϕ while fixing the other variable for I and J times (I and J are the update steps after reaching the stopping criterion). The updating function ϕ defines updating rules for each variable, such as vanilla gradient descent, stochastic gradient descent. At the outer loop, all the variables are updated with respect to the inner loop updating results. The global loss function is then evaluated and new sub-problems are generated.

The AM-based algorithm attempts to achieve a satisfactory solution to the overall problem by iteratively minimizing the two sub-problems $f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_n}(\boldsymbol{W})$ se-

Algorithm 1: General structure of alternating optimization

Given: global loss function $F(\boldsymbol{W}, \boldsymbol{X})$, local loss functions $f_{\boldsymbol{W}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}}(\boldsymbol{W})$, a random initialization $\boldsymbol{W}_{n=0}$:
for $n \leftarrow 1,2, \ldots$ **do**
  for $i \leftarrow 1,2,\cdots,I$ **do**
    $\Delta \boldsymbol{X} = \phi\left(\{\boldsymbol{X}_k\}_{k=0}^{i-1}, \nabla f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X}_{i-1})\right)$
    $\boldsymbol{X}_i = \boldsymbol{X}_{i-1} + \Delta \boldsymbol{X}$
  end
  $\boldsymbol{X}_n = \boldsymbol{X}_I$
  $f_{\boldsymbol{X}_n}(\boldsymbol{W}) \leftarrow \boldsymbol{X}_n$
  for $j \leftarrow 1,2,\cdots,J$ **do**
    $\Delta \boldsymbol{W} = \phi\left(\{\boldsymbol{W}_k\}_{k=0}^{j-1}, \nabla f_{\boldsymbol{X}_n}(\boldsymbol{W}_{j-1})\right)$
    $\boldsymbol{W}_j = \boldsymbol{W}_{j-1} + \Delta \boldsymbol{W}$
  end
  $\boldsymbol{W}_n = \boldsymbol{W}_J$
  $f_{\boldsymbol{W}_n}(\boldsymbol{X}) \leftarrow \boldsymbol{W}_n$
end

quentially. When sub-problems have closed-form solutions, it is thus easy to implement and apply an AM-based algorithm.

However, the AM-based algorithm may not always be able to converge to a good solution. This could be due to two main issues in the AM-based methods: 1) AM-based method optimizes over the local loss functions without fully utilizing the information from the global loss function, and 2) AM-based methods usually solve the local loss function through exhaustive optimization using the first order information which may not necessarily lead to the best solution in terms of the global objective. We will elaborate these two issues in the sequel.

The first issue is straightforward that the AM algorithm is essentially implemented on optimizing each local loss function corresponding to one variable. Mathematically, the updating rule for $f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X})$ in the inner loop can be expressed as:

$$\boldsymbol{X}_i = \boldsymbol{X}_{i-1} + \phi\left(\{\boldsymbol{X}_k\}_{k=0}^{i-1}, \nabla f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X}_{i-1})\right), \tag{2.1.9}$$

where $\boldsymbol{X}_i$ is the i-th updating step of variable $\boldsymbol{X}$ inside the process of minimizing

$f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X})$, and ϕ is a function to solve a steep descent with respect to the local loss function. It is clear that ϕ essentially determines the updating rule for the variables, while it does not directly take global loss function $F(\boldsymbol{W}, \boldsymbol{X})$ into account.

We then explain the *exhaustive-optimization* issue. Within each iteration of the AM-based algorithm, the two variables $\boldsymbol{W}$ and $\boldsymbol{X}$, sequentially optimized while fixing the other variable, are usually updated towards the first order stationary point of $f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_n}(\boldsymbol{W})$. That is, the solutions shall eventually satisfy:

$$\begin{aligned} \nabla_{\boldsymbol{X}} f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X}_n) = 0, \\ \nabla_{\boldsymbol{W}} f_{\boldsymbol{X}_n}(\boldsymbol{W}_n) = 0. \end{aligned} \tag{2.1.10}$$

However, those first order stationary points are not necessarily the best solutions of (2.1.8) when our objective is to achieve a good convergence for solving (5.1.1). In other words, there exist $\boldsymbol{W}_n^*$ and $\boldsymbol{X}_n^*$ satisfying

$$\begin{aligned} \nabla_{\boldsymbol{X}} f_{\boldsymbol{W}_{n-1}}(\boldsymbol{X}_n^*) \neq 0, \\ \nabla_{\boldsymbol{W}} f_{\boldsymbol{X}_n}(\boldsymbol{W}_n^*) \neq 0, \end{aligned} \tag{2.1.11}$$

but achieving $F(\boldsymbol{W}_n^*, \boldsymbol{X}_n^*) \leq F(\boldsymbol{W}_n, \boldsymbol{X}_n)$. In contrast, the AM procedure essentially leads to an exhaustive-optimization on sub-problems since its updating strategy for variables is isolated from the global loss information.

Addressing the aforementioned two issues of the AM is non-trivial because it is intractable to manually design an algorithm to find all of those non-optimums $\boldsymbol{W}_n^*$ and $\boldsymbol{X}_n^*$ of each sub-problem in (2.1.11). The key obstacle is that the manually designed algorithms are essentially with fixed updating rule with respect to ϕ, while to find those non-optimums requires an algorithm with adaptive updating rules for different scenarios.

In conclusion, AM based methods are one of the most popular solution algorithms for

non-convex multi-variable optimization problems, including EM algorithm for GMM, ALS and SGD algorithms for matrix completion. It is a heuristic method that directly optimizes non-convex optimization problem. The freedom to vary with the different variables in the optimization problem gives AM-based methods immense flexibility to generate different variants for specific constraints or demands from the natural settings of the problems. Despite it has been proved successful in many applications, the intrinsic optimizing strategy of AM methods does not serve the objective well. Significant improvements are expected to be achieved when the exhaustive-optimization could be well sorted. In the Chapter 3 and 4, we will present two specific variants of AM-based approaches for non-convex multi-variable problems, respectively. Then, In Chapter 5, we propose a new approach to solve the exhaustive-optimization issue of AM-based methods.

2.2 Practical Applications

In Chapters 3 and 4, we illustrate bi-linear inverse problem and non-linear problem in two practical scenarios in details, respectively. For the bi-linear case, we study the OD flows estimation problem in Chapter 3 to show that by reformulating the original OD flows estimation problem as a bi-linear inverse problem, significant dimension reduction and better performance have been successfully achieved. For non-linear problem, the MKC problems are analyzed and an improved solution is presented in Chapter 4 to solve localized incomplete MKC. We essentially amend the previous non-convex optimization algorithm by adding an extra structural constraints regularization term. We will briefly demonstrate the basic definition and tools in OD flows estimation and MKC respectively. The previous works of these two practical problems will be reviewed to provide a fundamental background, and the properties and limitations will be analyzed.

2.2.1 Origin-Destination Flow Estimation

The OD flow estimation problem can be stated as follows. Consider a network specified by the sets of nodes (also known as vertices) and links (also called edges). Suppose that the quantities of link flows are given during the time period of interest involving multiple time intervals. The task is to estimate the volume of the OD flows.

OD flow estimation is essential to many network analysis tasks. This thesis uses transportation networks as an example though the developed approach can be readily extended to many other application scenarios. In the transportation community, it has been widely accepted that OD information reflects the travel demands and plays an essential role in long term infrastructure planning [38, Sec.1], traffic prediction under unexpected changes in the infrastructure [38, Sec.1] or the traffic status, and commercial applications linked to population migration [38, Sec.1;Sec.10], and serves as an important input to traffic simulation models [39, 40]. Despite its importance, accurate OD information is difficult, expensive, and sometimes impossible to be obtained [41]. The old fashioned household survey data are expensive and time-consuming [42] to collect and it is typically incomplete and biased [43]. Commercial transportation service data from taxies and Uber are highly biased towards commercial activities. Recently modern technologies, for instance GPS, mobile phone, automatic plate recognition systems, automatic vehicle identification systems, and combinations of these systems, provide new data sources and new opportunities for acquiring OD information [44–46]. However, the data are highly privacy-sensitive, biased, and error-prone [44–46].

Forward Mapping Model

Typically, OD flow estimation problems are modelled by a mapping model from link flow to OD flow. In this section, the basic forward model between flows, and various of conventional approaches for OD estimation problem are reviewed. We highlight that

the prominent deficiency of OD estimation is the high dimensionality of OD flows. As a consequence, it is typically ill-posed, and the solutions generally rely on prior information and historical data.

Before analyzing the types of different methods on seeking an excellent model with efficiency and accuracy to OD estimation problem, a fundamental fact underlies on all these approaches is that the OD flows are related to some observed sample data via a forward model. In modern traffic society, the link flow is the most widely applied sample data. Therefore, the most basic model is to assume that there is a map from OD flows to link flows, which is linear and hence represented by a matrix. Refer to the linear operator as the traffic assignment matrix [38, 42, 43, Sec.10] (also known as link choice proportions [47, 48], or path proportions [49, 50]). The entries denote the fractions of OD flows passing through specific links.

More formally, assume in a graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, a link from a node i to its adjacent node j is denoted by $i \to j$ and the quantity of flow across the link is denoted by $y_{i\to j}$. Similarly the volume of the flow from an origin node o to a destination node d is denoted by s_{od} where od denotes the OD pair from the node o to the node d. For adjacent nodes i and j, it holds that $s_{ij} \leq y_{i\to j}$, where the equality holds if and only if all the link flow $y_{i\to j}$ is generated by the OD flow s_{ij}. Assume that the network has n_n nodes and n_ℓ links. Group the link flows $y_{i\to j}$ into a vector $\boldsymbol{y}$. It is clear that $\boldsymbol{y} \in \bar{\mathbb{R}}^{n_\ell}$ where $\bar{\mathbb{R}} := \mathbb{R}^+ \bigcup \{0\}$ is the set of non-negative real numbers.[1] Let $\boldsymbol{s}$ denote the OD flow vector of which the entries are of the form $s_{ij} \in \bar{\mathbb{R}}$. Then the relation between OD flows and link flows can be described by the following linear form

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{s}, \tag{2.2.1}$$

where the matrix $\boldsymbol{A}$ is referred to as traffic assignment matrix, and its entries $a_{i\to j,od}$

[1] For modeling and computational simplicity, we relax the domain of traffic counts from $\mathbb{Z}^+ \bigcup \{0\}$ to $\mathbb{R}^+ \bigcup \{0\}$.

gives the fraction of OD flow s_{od} passing the link $i \to j$. Majority conventional methods adopt the linear equation (2.2.1) or analogue equations, referring to [43, 49, 51–55].

Finer time resolution can be added to the above model. Model (2.2.1) implicitly assumes that all the OD flows finish in a single time window, which is reasonable in real life when the time window is roughly of a day long. With finer time resolution, for example the time windows for link flow observations are of tens of minutes long, OD flows may require multiple time windows (steps) to finish. The relations between OD flows and link flows are then described by a convolution form:

$$\boldsymbol{y}^t = \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{A}^\tau \boldsymbol{s}^{t-\tau+1}, \text{ or equivalently } \boldsymbol{y}^t = \boldsymbol{A}^t * \boldsymbol{s}^t, \tag{2.2.2}$$

where $\boldsymbol{y}^t$, $\boldsymbol{A}^t$, and $\boldsymbol{s}^t$ are the link flow, traffic assignment matrix, and the OD flow at time instance t, respectively, $\tau_{\max} \in \mathbb{Z}^+$ is the maximum number of time steps involved to finish all OD flows, and the symbol $*$ denotes a convolution which is commonly used in signal processing community. Refer to this model as multiple step model. It provides finer information about the flows and fits the physical systems better. However, it involves much more unknown variables and requires much more prior information.

Remark 1 *It is noteworthy that the linear model (2.2.1) assumes a separable model where the traffic assignment matrix $\boldsymbol{A}$ is independent of the OD flows $\boldsymbol{s}$. In actual transportation networks, the assignment matrix $\boldsymbol{A}$ and the OD flows are correlated (non-separable) according to the fundamental diagram of traffic flow. A non-separable model results in a non-linear model [49, 51, 56–59], e.g., user equilibrium model [51], goes beyond the scope of this work.*

OD Estimation Approaches in the Literature

Solving the inverse problem of OD estimation is challenging. As the number of OD flows can be orders of magnitude larger than that of observed variables, e.g., the link

flows, the inverse problem from observations to OD flows is ill-posed, meaning that there are multiple (can be infinitely many) feasible solutions. Traffic models often involve many parameters, and prior information is needed to determine their values. For a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$ with n_n many nodes, the number of possible OD pairs is $n_n(n_n - 1) = O(n_n^2)$. The number of links in the network is typically $n_\ell = O(n_n)$. For a network with reasonably large n_n, the number of equations $O(n_n)$ in the linear system (2.2.1) is less than the number of unknowns $O(n_n^2)$. The OD estimation problem is therefore ill-posed.

In the literature, it is typically assumed that the traffic assignment matrix $\boldsymbol{A}$ is given. In this instance, to solve the ill-posed problem, the most widely applied strategy is incorporating the historical OD flows. Many classical models have the objective function which are composed of historical OD flows as a significant measure, and here we briefly pick out some representative ones to demonstrate their characteristics as follows.

The generalized least squares (GLS) estimator [43, 55, 56] is one of the most popular OD estimation approach with the following form:

$$\min_{\boldsymbol{s}}(\boldsymbol{A}\boldsymbol{s} - \boldsymbol{y})^\top \boldsymbol{W}^{-1}(\boldsymbol{A}\boldsymbol{s} - \boldsymbol{y}) \ + (\boldsymbol{s} - \boldsymbol{s}^H)^\top \boldsymbol{V}^{-1}(\boldsymbol{s} - \boldsymbol{s}^H), \tag{2.2.3}$$

where $\boldsymbol{s}^H$ are the historical OD flows, $\boldsymbol{W}$ and $\boldsymbol{V}$ are the covariance matrices of weighting, which are typically obtained through parameters tunning. The principle insight of the objective function (2.2.3) is to minimize the Euclidean distance between estimated OD flow and historical data (e.g., [43, 47, 49, 54–56, 59–65]). Considering the second term in Equation (2.2.3), it indicates that the GLS estimator estimates $\boldsymbol{s}$ highly based on the historical OD flows, when the first term is seen to solve the Equation (2.2.1).

Another classical method is based on the maximum entropy (ME) theory [42,51,54,56],

which has a different objective function as follows:

$$\min_{\boldsymbol{s}\geq 0} \sum \boldsymbol{s} \log \frac{\boldsymbol{s}}{\boldsymbol{s}^H} : \boldsymbol{y} = \boldsymbol{A}\boldsymbol{s}. \tag{2.2.4}$$

The ME approach also aims at enforcing the estimated OD flows to be close to the historical data, when the proportional traffic flows routing pattern is given.

Instead of directly adopting historical OD data, statistical models assume that the mean value of the estimated OD flows are determined by those from the historical data. The deviations from the mean values are assumed to be due to random effects and statistical models can be built for modeling. The commonly used statistical models such as maximize likelihood estimator and Bayesian approach tend to seek a solution to maximum the posterior estimation of $\mathbb{P}(\boldsymbol{s}|\boldsymbol{A}, \boldsymbol{y})$ under Gaussian distribution or Poisson distribution with given prior $\mathbb{P}(\boldsymbol{s})$ [47, 48, 53, 66, 67].

The work in [52] introduces another indirect way to use historical OD data: calibrating the deviations and serial correlations of the OD flows. The deviation of time interval t between OD flows and historical data is defined as $\partial s_t = s_t - s_t^H$. The transition equation based on auto-regressive process on this deviation equation is given by

$$\partial s_t = \sum_{p=t-q'}^{t-1} f_t^p \partial s_p + w_t, \tag{2.2.5}$$

where f_t^p denotes to the contribution of ∂s_p to ∂s_t, and w_t describes the errors. Then it has a similar formulation to Equation (2.2.1) to demonstrate the serial correlations between deviation of OD flows and link flows as follows:

$$\partial y_t = \sum_{p=t-q'}^{t} a_t^p \partial s_p + v_h, \tag{2.2.6}$$

where $\partial y_t = y_t - \sum_{p=t-q'}^{t} a_t^P s_p^H$. In [52], it conducts a least square form of objective

function

$$\min_{S} \sum_{h=1}^{k} \|\boldsymbol{\Omega}_h^{-1}\boldsymbol{C}_h^N\boldsymbol{S} - \boldsymbol{\Omega}_h^{-1}\boldsymbol{z}_h\|_2^2 + \sum_{h=k+1}^{N} \|\boldsymbol{\Omega}_h^{-1}\boldsymbol{C}_h^N\boldsymbol{S}\|_2^2,$$

where $\boldsymbol{S} = (\partial s_1, \partial s_2, \cdots, \partial s_N)$, $\boldsymbol{\Omega}_h^{-1}$, $\boldsymbol{C}_h^N$ and $\boldsymbol{z}_h$ are the matrices contain f_t^p, a_t^p and ∂y_t to reflect the Equations (2.2.5) and (2.2.6). The first item provides an estimation of OD flows from interval 1 to interval k , and the second one predicts the OD flows for $k+1$ to N, based on the estimated OD flows by a direct application of the auto-regressive process. More detailed information can be found in [52, Eq(4)].

There is a fact that obtaining the historical OD flows is time-consuming, expensive and sometime impossible to be collected. Replacing the historical data by the prior knowledge of OD flow structure is also an important way to mitigate the ill-posedness.

Using the structure of OD flows provides certain help on solving the ill-posed problem. For example, principal component analysis (PCA) is used in [68, 69] by assuming that the OD flow vectors from multiple time intervals form an approximately low-rank matrix. Clearly this approach requires the prior knowledge of the paramount nodes and OD flows, and runs the risk of over-simplifying the network and losing detailed and valuable information.

Another interesting approach is to use a sparsity assumption that among all possible OD flows, only few of them generate nearly all the link flows (e.g., [56] conducts sparsity to estimate OD flow, [70] adopts the sparsity of OD flows in frequency domain, and [64] enrolls speed and density to improve the estimation model. [56] promotes the aforementioned models by using ℓ_1 regularization term to mitigate the ill-posedness when the prior OD matrix is unavailable with the following form:

$$\min_{\boldsymbol{s}\geq 0}(\boldsymbol{A}\boldsymbol{s} - \boldsymbol{y})^\top \boldsymbol{W}^{-1}(\boldsymbol{A}\boldsymbol{s} - \boldsymbol{y}) \ + \lambda\|\boldsymbol{s}\|_1. \tag{2.2.7}$$

In that case, the sparsity regularization does help recover the OD flows but it requires strict sparse condition on OD flows and fixed traffic assignment matrix as a priori.

In spite of many approaches assuming that the traffic assignment matrix is given, in reality, it is generally unavailable. The OD estimation problem without given traffic assignment matrix mainly involves non-linear models, such as the works in [39, 71–73], which bypass the traffic assignment matrix and directly estimate OD flows by a deterministic function describing the correlations at different time intervals, and works in [49, 74, 75] that conduct bi-level models to update the traffic assignment matrix by a non-linear transition between OD flows and link flows. However, both of those solutions are based on sufficient prior knowledge or OD historical data.

Summary

We have discussed that usefulness of various of models are primarily determined by the prior information. The review of these conventional approaches is neither to criticize any model, nor to dispute their contributions. The main purpose is to emphasize that all of them use prior information or historical OD data to solve the ill-posed problem by increasing known variables instead of decreasing unknowns. In this instance, we intend to sort the ill-posedness by incorporating a new forward model to achieve dimension reduction on unknown variables in Chapter 3.

2.2.2 MKC

In this section, we will firstly introduce the previous works on MKC algorithms, and then give a brief introductions on the definition and formulations of MKC methods and the popular variants. Clustering is a significant unsupervised learning approach for splitting data into a set of disjoint groups based on the intra-cluster and inter-cluster similarities. Most clustering algorithms assume a single affinity matrix to record pairwise similarity between data. However, in applications, there could be multiple affinity matrices due to the fruitful potential features. To address these problems,

MKC methods have been widely adopted for MKC [76–82], which integrates a number of pre-calculated base kernel matrices to aggregate samples into clusters.

Previous Approaches and Limitations

In the last few decades, various applications and variants of MKC algorithms [83–91] have been developed and applied. One of the most widely used MKC algorithm is the multiple kernel k-means (MKKM) algorithm. MKKM has been studied for decades and performs well on many MKC problems. One major types of works focus on improving the performance of MKKM algorithm on different observation data sets. The work in [84] improves MKKM by incorporating the capacity of clustering analysis, where an optimal kernel is learned from multiple data. Similar work can be found in [86] that extends MKKM by designing a localized matrix to utilize the characteristics of each individual sample for adaptively updating the kernel combination weights across different samples. To enhance the robustness towards noise, the work in [87] proposes a robust MKKM algorithm which replaces the squared error in existing k-means by an $\ell_{2,1}$-norm penalty term and thus achieves a simultaneous optimization on the best clustering labels and the optimal combination of kernels. Meanwhile, another group of works focuses on increasing the diversity and decreasing the redundancy of the selected base kernels. The work in [88] extends existing MKKM algorithms by designing a matrix-induced regularization term to sufficiently explore the correlation among pre-specified base kernels. Moreover, an optimal neighborhood kernel clustering algorithm is proposed in [91], where the representative ability of the optimal kernel is significantly improved and the negotiation between kernel learning and clustering is enhanced. In conclusion, the MKKM algorithms have achieved promising performance in various applications, more details of the aforementioned methods can be found in [92–94].

Typically, MKC algorithms share a common assumption: *all the pre-specified base kernels are complete*, which means all the base kernels are fully observed. However, in real

world applications, it is not universal that all the views of a sample are observed [95,96]. In such cases, the corresponding rows/columns of related base kernel matrices are generally unknown, referring to the incomplete base kernel matrices. The presence of incomplete base kernel matrices leads to increase of the complexity and non-convexity of the clustering problems. The existing methods generally adopt a "two-step" algorithm that approximates the missing elements in base kernel matrices with imputation methods and then performs MKC algorithms on these imputed kernels. Several imputation strategies, including zero-filling, mean value filling, k-nearest-neighbor filling and expectation-maximization (EM) filling [97], have been widely used to fill the missing parts of base kernel matrices. Some advanced methods have recently been proposed to perform matrix imputation [98–101]. The work in [98] estimates the missing views of the kernels based on the algebra combinations of the complete views. Besides, the work in [99] proposes an incomplete multi-view learning by utilizing the correlations among different views, which assumes that the different views are generated from a common subspace. The work in [100] presents a novel multiple incomplete view clustering algorithm to optimize a latent feature matrix corresponding to each view which then generates a consensus matrix by minimizing the differences of each view between latent feature matrix and the generated consensus matrix. Meanwhile, the approach in [101] predicts the missing elements of base kernel matrices via utilizing within-view and between-view correlation among kernel matrices.

One major drawback in previous works is that the imputation is separated from the subsequent clustering. Incorporating suitable interactions between the imputation and clustering processes has been proved to be conductive to the performance [102–104]. Typically, many works tend to integrate the imputation and clustering processes into a comprehensive optimization framework. Specifically, these works adopt the clustering partition at the last iteration to impute the missing elements of base kernel matrices, and then the imputed kernels are assigned to the clustering procedure. Therefore, the two processes collaborate on achieving a better clustering performance. Although

demonstrating superior clustering results in specific applications, these works, no matter aligning the combined kernel with the clustering partition in a global or local manner, do not sufficiently utilize the redundancy and diversity among pre-specified kernel matrices. In a sense, the selected kernels might have high redundancy and low diversity [88], which causing less effectiveness of the utilization on two processes and decrements on clustering performance. In Chapter 4, we present an improved MKKM-based non-convex optimization model to address this issue.

In the following, we will briefly review MKKM, MKKM with incomplete kernels (MKKM-IK) and its localized incomplete variant (LI-MKKM) in the terms of formulations and algorithms.

Multiple kernel k-means (MKKM)

We start with the vanilla MKKM method. Assume the number of clusters is k Let $\{\boldsymbol{x}_i\}_{i=1}^n \subseteq \mathcal{X}$ denote a set of n training samples, and $\phi_p(\cdot) : \boldsymbol{x} \in \mathcal{X} \mapsto \mathcal{H}_p$ denotes the p-th feature mapping that projects $\boldsymbol{x}$ onto a reproducing kernel Hilbert space $\mathcal{H}_p$ ($1 \leq p \leq m$). Each sample in MKC is represented by $\phi_{\boldsymbol{\gamma}}(\boldsymbol{x}) = [\gamma_1\phi_1^\top(\boldsymbol{x}), \cdots, \gamma_m\phi_m^\top(\boldsymbol{x})]^\top$, where $\boldsymbol{\gamma} = [\gamma_1, \cdots, \gamma_m]^\top$ is a vector of weights for the m pre-specified base kernel functions $\{\kappa_p(\cdot,\cdot)\}_{p=1}^m$. These weights are adaptively adjusted during MKC process. Therefore, the corresponding kernel function can be formed as follows.

$$\kappa_{\boldsymbol{\gamma}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi_{\boldsymbol{\gamma}}^\top(\boldsymbol{x}_i)\phi_{\boldsymbol{\gamma}}(\boldsymbol{x}_j) = \sum_{p=1}^m \gamma_p^2 \kappa_p(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.2.8}$$

The kernel matrix $\boldsymbol{K}_{\boldsymbol{\gamma}}$ on training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ thus is obtained through the kernel function in Equation (2.2.8), and the objective of MKKM with $\boldsymbol{K}_{\boldsymbol{\gamma}}$ can be formulated as

$$\begin{aligned} &\min_{\boldsymbol{H},\boldsymbol{\gamma}} \ \mathrm{Tr}\left(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top)\right) \\ &s.t. \ \ \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k, \ \ \boldsymbol{\gamma}^\top\mathbf{1}_m = 1, \ \gamma_p \geq 0, \ \forall p, \end{aligned} \tag{2.2.9}$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times k}$ is a relaxation of the cluster assignment matrix, and $\boldsymbol{I}_k$ is a $k \times k$ identity matrix. More detailed introduction on $\boldsymbol{H}$ please check in [88]. Equation (2.2.9) can be optimized via AM algorithms over $\boldsymbol{H}$ and $\boldsymbol{\gamma}$ as follows.

- Optimizing $\boldsymbol{H}$ with fixed $\boldsymbol{\gamma}$. While $\boldsymbol{\gamma}$ is fixed, the optimization of (2.2.9) with respect to $\boldsymbol{H}$ is a traditional kernel k-means in the following form

$$\begin{aligned} &\max_{\boldsymbol{H}} \ \mathrm{Tr}\left(\boldsymbol{H}^\top \boldsymbol{K}_{\boldsymbol{\gamma}} \boldsymbol{H}\right) \\ &s.t. \ \ \boldsymbol{H} \in \mathbb{R}^{n\times k}, \boldsymbol{H}^\top \boldsymbol{H} = \boldsymbol{I}_k, \end{aligned} \tag{2.2.10}$$

  The optimal $\boldsymbol{H}$ in (2.2.10) is composed of the k eigenvectors corresponding to the top-k eigenvalues of $\boldsymbol{K}_{\boldsymbol{\gamma}}$ [105].

- Optimizing $\boldsymbol{\gamma}$ with fixed $\boldsymbol{H}$. When $\boldsymbol{H}$ is fixed, the optimization in (2.2.9) with respect to $\boldsymbol{\gamma}$ is equivalent to

$$\begin{aligned} &\min_{\boldsymbol{\gamma}} \ \sum\nolimits_{p=1}^{m} \gamma_p^2 \mathrm{Tr}\left(\boldsymbol{K}_p(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top)\right) \\ &s.t. \ \ \boldsymbol{\gamma}^\top \mathbf{1}_m = 1, \ \gamma_p \geq 0, \end{aligned} \tag{2.2.11}$$

  which has a closed-form solution.

MKKM with Incomplete Kernels (MKKM-IK)

MKKM-IK is proposed to handle incomplete MKC based on the MKKM [102,104]. Different from imputing the incomplete kernel matrices at the first step and then applying existing MKKM on the imputed kernel matrices, MKKM-IK combines the imputation and clustering procedure into one comprehensive optimization problem and solves it via alternative algorithms. In MKKM-IK, the incomplete (unobserved) parts of base kernel matrices are imputed at each iteration, integrating with clustering results. Thus, the imputed base kernel matrices are continuously updated during the whole optimization

course. The objective function of MKKM-IK can be formulated as follows.

$$\begin{aligned} \min_{\boldsymbol{H},\boldsymbol{\gamma},\{\boldsymbol{K}_p\}_{p=1}^m} \quad & \mathrm{Tr}\left(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top)\right) \\ s.t. \quad & \boldsymbol{H} \in \mathbb{R}^{n\times k}, \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k, \\ & \boldsymbol{\gamma}^\top \mathbf{1}_m = 1, \gamma_p \geq 0, \\ & \boldsymbol{K}_p(\boldsymbol{s}_p, \boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0,\ \forall p, \end{aligned} \tag{2.2.12}$$

where $\boldsymbol{s}_p\,(1 \leq p \leq m)$ denotes the observed entry indices in sample, and $\boldsymbol{K}_p^{(cc)}$ denotes the kernel sub-matrix calculated by the observed samples. Note that it imposes a constraint $\boldsymbol{K}_p(\boldsymbol{s}_p, \boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)}$ to ensure the observed entries of $\boldsymbol{K}_p$ keep fixed during the learning course.

A three-step optimization strategy is proposed in [102] to solve Equation (2.2.12) through an alternative approach, which repeats iteration until convergence of the following three steps:

- Optimizing $\boldsymbol{H}$ with $\boldsymbol{\gamma}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$ fixed. Given $\boldsymbol{\gamma}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$, the optimization of (2.2.12) with respect to $\boldsymbol{H}$ reduces to a kernel k-means problem, which can be solved by (2.2.10);

- Optimizing $\{\boldsymbol{K}_p\}_{p=1}^m$ with $\boldsymbol{\gamma}$ and $\boldsymbol{H}$ fixed. Given $\boldsymbol{\gamma}$ and $\boldsymbol{H}$, Problem (2.2.12) with respect to each $\boldsymbol{K}_p$ can be reformulated as the following optimization problem,

$$\begin{aligned} \min_{\boldsymbol{K}_p} \quad & \mathrm{Tr}\left(\boldsymbol{K}_p(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top)\right) \\ s.t. \quad & \boldsymbol{K}_p(\boldsymbol{s}_p, \boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0. \end{aligned} \tag{2.2.13}$$

  It is clear that (2.2.13) is a semi-definite programming (SDP) problem, which is typically computationally intensive. It has been proven in [102] that the optimal $\boldsymbol{K}_p$ in (2.2.13) has the closed-form formulation as shown in Equation (2.2.14), where $\boldsymbol{W} = \boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top$ and $\boldsymbol{W}^{(cm)}$ are constructed by the elements of $\boldsymbol{W}$ with

respect to the observed and unobserved sample indices. $\boldsymbol{W}^{(mm)}$ is the elements to be optimized in matrix $\boldsymbol{W}$. More details can be checked in [102].

$$\boldsymbol{K}_p = \begin{bmatrix} \boldsymbol{K}_p^{(cc)} & -\boldsymbol{K}_p^{(cc)}\boldsymbol{W}^{(cm)}(\boldsymbol{W}^{(mm)})^{-1} \\ -(\boldsymbol{W}^{(mm)})^{-1}\boldsymbol{W}^{(cm)\top}\boldsymbol{K}_p^{(cc)} & (\boldsymbol{W}^{(mm)})^{-1}\boldsymbol{W}^{(cm)\top}\boldsymbol{K}_p^{(cc)}\boldsymbol{W}^{(cm)}(\boldsymbol{W}^{(mm)})^{-1} \end{bmatrix} \tag{2.2.14}$$

- Optimizing $\boldsymbol{\gamma}$ with $\boldsymbol{H}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$ fixed. Given $\boldsymbol{H}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$, Problem (2.2.12) with respect to $\boldsymbol{\gamma}$ reduces to a quadratic programming with linear constraints, which can be directly solved by the gradient descent based algorithms.

Localized Incomplete MKKM (LI-MKKM)

Although unifying clustering and imputation into on comprehensive procedure brings significant improvements on performance, the unification of these two procedures in MKKM-IK is inadequate. There is a fact that this unification is achieved via globally maximizing the alignment between the optimal kernel matrix $\boldsymbol{K}_{\boldsymbol{\gamma}}$ and the ideal matrix $\boldsymbol{H}\boldsymbol{H}^\top$, as shown in Equation (2.2.9). This thus requires all pairwise samples to be arbitrarily aligned with the ideal similarity, while ignoring the distances factors between samples.

Instead of calculating the alignment between the optimal kernel and the ideal matrix in a global manner as shown in Equation (2.2.12), LI-MKKM [103] tends to utilize the local structure among data to decrease the computational complexity and increase the solution efficiency by only requiring each sample to align the similarity with its k-nearest neighbours. In this way, LI-MKKM is able to well handle the intra-cluster variations. Mathematically, it calculates the local alignment of the i-th sample

$$\langle (\boldsymbol{S}^{(i)})^\top \boldsymbol{K}_{\boldsymbol{\gamma}} \boldsymbol{S}^{(i)}, (\boldsymbol{S}^{(i)})^\top (\boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^\top)\boldsymbol{S}^{(i)} \rangle, \tag{2.2.15}$$

where $\boldsymbol{S}^{(i)} \in \{0,1\}^{n\times\tau}(1 \leq i \leq n)$ denotes the neighborhood indication matrices. For example, $S_{jv}^{(i)} = 1$ denotes $\boldsymbol{x}_j$ is the v-th nearest neighbor of $\boldsymbol{x}_i$, where $1 \leq v \leq \tau$, and τ is the number of nearest neighbors. The objective function of LI-MKKM is given by,

$$\begin{aligned} \min_{\boldsymbol{\gamma},\{\boldsymbol{K}_p\}_{p=1}^m,\boldsymbol{H}} \quad & \sum_{i=1}^{n} \mathrm{Tr}(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{B}^{(i)})) \\ s.t. \quad & \boldsymbol{H} \in \mathbb{R}^{n\times k},\ \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k,\ \boldsymbol{\gamma}^\top\mathbf{1}_m = 1,\ \gamma_p \geq 0, \\ & \boldsymbol{K}_p(\boldsymbol{s}_p,\boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0,\ \forall p, \end{aligned} \tag{2.2.16}$$

where $\boldsymbol{B}^{(i)}$ is the neighborhood index matrix of the i-th samples, capturing the localized k-nearest neighborhood of each kernel matrix.

Similar to [102], the work in [103] develops a three-step optimization algorithm to solve Equation (2.2.16) and theoretically proves its convergence. The detailed proofs can be found in [103].

Summary

The aforementioned methods for MKC problems are typically solved in an alternating strategy, and the trend of the improvements is lying on the integration of imputation procedures and clustering procedures. In this instance, we present a new approach for MKC that iteratively solves problem via a three-step algorithm with better performance and derivation on the generation error bound in Chapter 4.

2.3 Meta-learning for Optimization

In this section, we present a brief introduction to the meta-learning as a fundamental basics for the later analysis on our meta-learning based approach for solving non-convex multi-variable problems.

2.3.1 Fundamental Basics

One of the most significant features of human intelligence is the capability of learning to solve new tasks by leveraging previous experiences as a guidance. Recently, the idea of learning to learn or meta-learning [106–108] enables an artificial intelligent agent to learn an adaptive strategy in solving problems from both existing data and historical information, therefore to be able to achieve generally good performances on a set of new tasks. Meta-learning [106, 109, 110], also known as learning to learn, aims to acquire specific refined knowledge from a set of different tasks, and thus improves the learning efficiency on new tasks. An example of the fundamental meta-learning approach is from the works of [109, 110] which builds an alternative gradient-based system with two linear networks. The weight updates of these two networks have an interactions: The first network outputs weights update terms for the second networks, whose weights can vary quickly up to the outputs of the first networks. This idea indicates a basic two-level strategy that allows the weight update of a network to be dependent on the information collected from another network. The key idea of meta-learning is to learn an meta-learner that is capable of achieving fast adaption and good performances on a large number of new tasks. This is achieved by extracting inductive bias across a set of tasks, which is the refined knowledge from previous experiences of solving different tasks and is exploited to provide efficient and effective solutions for solving new tasks. More specifically, the meta-learning establishes a learning strategy which collects domain-specific knowledge within each single task, and then extracts domain-general knowledge across different tasks as inductive bias. In this way, the learned algorithm is constantly updated by the inductive bias from leveraging prior experiences on solving different tasks to provide satisfactory performances on new tasks.

Mathematically, machine learning problem generally seeks a model to describe the mapping between observations $\mathbf{y}$, variables $\boldsymbol{x}$ and outputs losses $\mathcal{L}$. Then, it designs an algorithm to optimize the parametric model on the training dataset $\mathcal{D} = D_1, D_2, \cdots, D_T$

which are then evaluated by the testing dataset $\mathcal{D}^{'}$ thereof. Meta-learning typically contains a meta-learner that is trained on the training sets D and outputs an optimizer with parameters $\boldsymbol{\theta}$. The outputs of the optimizer are generally feedback to the meta-learner to update the parameters for better performance. The obtained meta-learner is expected to generate corresponding optimizer that is able to quickly adapt to new tasks in testing sets $\mathcal{D}^{'} = D_1^{'}, D_2^{'}, \cdots, D_T^{'}$. A computational graph is depicted in Figure 2.1. The red dashed line splits meta-learning into meta-training and meta-testing. In meta-training course, each input training sample $\mathbf{y}_i$ is the i-th sample from the training dataset, while $\mathbf{y}_{test}$ are the whole samples of the testing set. The black dashed arrows denote that the parameters $\boldsymbol{\theta}$ are updated up to the loss $\mathcal{L}_i$ and the corresponding gradient $\nabla_{\boldsymbol{\theta}_{i-1}}\mathcal{L}_i$ of the optimizer at step i in meta-training course. Referring the optimizer as m, the ultimate target of meta-learning is to obtain a set of parameters $\boldsymbol{\theta}_T$ after T (limited) numbers of training steps, which is expected to be well-performed on all the samples $\mathbf{y}_{\text{test}}$ in testing set. The performance is typically evaluated by a loss function $\mathcal{L}(m(\boldsymbol{\theta}_T, \boldsymbol{x}), \mathbf{y}_{\text{test}})$, where $m(\boldsymbol{\theta}_T, \boldsymbol{x})$ denotes the outputs of the optimizer that optimizes variable $\boldsymbol{x}$ with respect to the learned parameters $\boldsymbol{\theta}_T$ after meta-training.

Remark 2 *Typically, an usual machine learning training procedure continuously update parameters with respect to the performance of parameters on each training samples. In meta-learning, the parameters of meta-learners are updated up to the performance of the optimizer. Then the parameters of the optimizers are updated by the meta-learners.*

Remark 3 *In many meta-learning applications, e.g., few-shot learning [1], the input training sample $\mathbf{y}_i$ is generally composed of samples for training and samples for query. The query samples are labelled, therefore the output losses $\mathcal{L}_i$ can be calculated via computing the deviation between the prediction and label. In this thesis, we focus on scenarios where no labelled query sample is available. In a sense, $\mathcal{L}_i$ is directly obtained through the training data from $\mathbf{y}_i$ and the optimization results within each task.*

In summary, the meta-learning aims to learn meta-learner with limited samples for

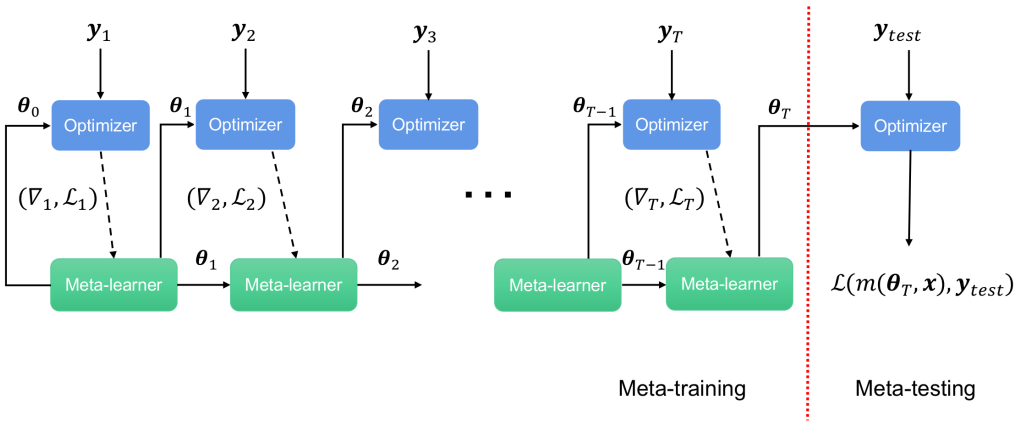


Figure 2.1: Computational graph of general meta-learner.

generating optimizer to solve a type of tasks which have different landscapes. Instead of directly designing or optimizing the optimizer for one specific task, meta-learning focuses on learning the strategy of generalizing optimizer for a set of tasks.

2.3.2 Learning Principle

In machine learning, the standard optimization algorithms generally train the model with specific gradient descent based methods, that adopt variable update function in the form

$$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha_i \nabla_{i-1} \mathcal{L}_i, \tag{2.3.1}$$

where $\boldsymbol{x}_{i-1}$ is the variable of the model after $i-1$ updates, α_i is the step size at the i-th update, $\nabla_{i-1}\mathcal{L}_i$ represents the loss optimized by the algorithm at the i-th update. We refer to $\alpha_i \nabla_{i-1}\mathcal{L}_i$ as the variable update function ϕ. Typically, different optimization algorithms are distinguished by their choices of ϕ. Once the ϕ is determined, the updating rule of the variables is fixed for the whole optimization course.

In meta-learning, it essentially converts the design of an algorithm into a learning task, where specific optimization strategy is learned through generalizing optimizers for several optimization problems in the training dataset. Compared to the standard optimization algorithm for a specific problem in machine learning, meta-learning tends to refine knowledge across a set of optimization problems. This is achieved by incorporating an extra module referring to the meta-learners, that are generally composed of neural networks. The parameters $\boldsymbol{\theta}$ of meta-learner determine the optimization algorithms with respect to the variable updating function ϕ. For different optimization problems, learning an optimization strategy can be considered as learning parameters $\boldsymbol{\theta}$ to generalize the variable updating function ϕ in the form of

$$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \phi(\{\boldsymbol{x}_k\}_{k=0}^{i-1}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_{i-1}), \boldsymbol{\theta}), \tag{2.3.2}$$

where $\boldsymbol{\theta}$ represents the parameters of the meta-learner, $\{\boldsymbol{x}_k\}_{k=0}^{i-1}$ represents the historical values of variables for $i-1$ optimization steps, and $\nabla_{\boldsymbol{x}} f$ is the gradient of objective function on $\boldsymbol{x}$.

It can be concluded that in meta-learning the tasks (optimization problems) are optimized by the algorithms that are determined by variable updating function ϕ, as well as ϕ is updated by the parameters $\boldsymbol{\theta}$ from meta-learners. Therefore, the optimization rule on parameters $\boldsymbol{\theta}$ essentially determines the learning principle of meta-learning on a set of optimization problems. Typically, the parameters $\boldsymbol{\theta}$ are updated by the backpropagation algorithm with respect to specific variation on losses or predictions across different problems instead of that of within single problem. In this way, the meta-learning leverages the experience of optimizing a set of optimization problems to refine task-across knowledge for constantly optimized on $\boldsymbol{\theta}$, while the experiences of solving up-streaming tasks are embedded. To the end, the learned algorithm is determined by the parameters of the optimized meta-learners. We will provide more detailed explanations with specific examples in Chapter 5

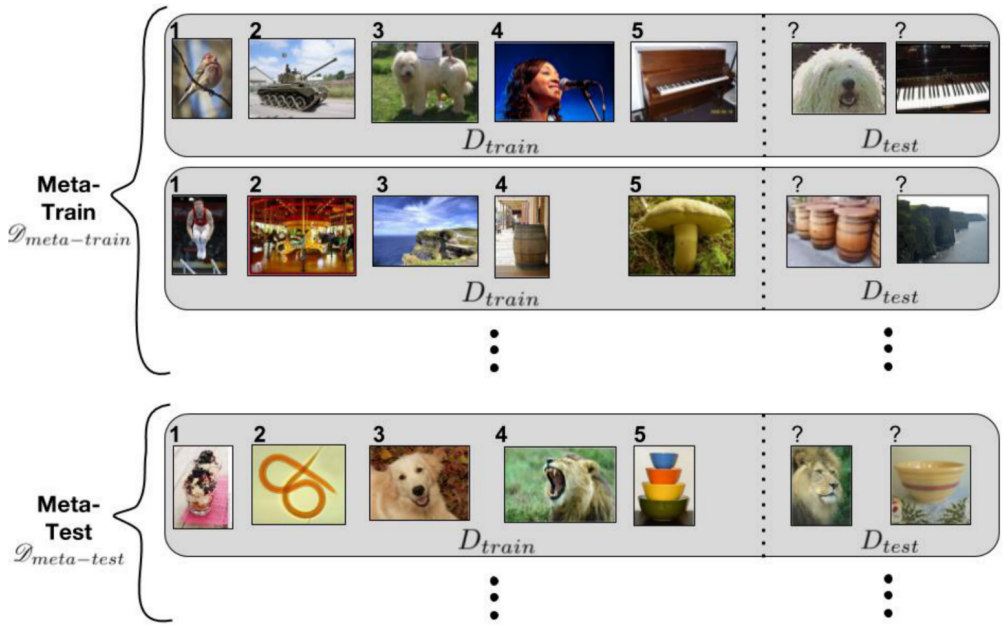


Figure 2.2: Example meta-learning set-up for few-shot image classification, visual adapted from Ravi *et al.* [1]

2.3.3 An Example of Meta-learning for Image Classification

Here we take the work in [1], which use meta-learning on image classification as an example of meta-learning setup.

In Figure (2.2), it presents a 5-way one shot image classification problem solved by meta-learning. Each gray box of meta-train is an independent meta-train dataset, which contains training dataset D_{train}(five images in five classes on the left) and testing dataset D_{test} two image in those five classes on the right. The setting of the meta-test dataset is similar, but we note that the testing dataset contains different classes of images that never appear in meta-train datasets.

In each grey box on top, the meta-train process with datasets $D_{meta-train}$ works in the following: A neural network $m_{learner}$ takes training dataset D_{train} as input. Then, $m_{learner}$ generates a classifier network $m_{optimizer}$ after learning each training dataset. The generated $m_{optimizer}$ is used to classify the two images from testing dataset D_{test}.

After that, parameters of $m_{learner}$ are updated with respect to the classification loss. The meta-test process works in a similar way. The $m_{learner}$ generates a new $m_{optimizer}$ for the input images from novel classes in data set $D_{meta-test}$. The new $m_{optimizer}$ classifies the two test images. We note that there is no update for $m_{learner}$ during the meta-test process.

The meta-train aims to learn a $m_{learner}$, which is able to generate a classifier with given different classes of images. The generated classifier can achieve high average classification performance on it's corresponding test set D_{test} in meta-test.

We conclude that in this meta-learning application, the training task is to learn an meta-learner, which is able to constantly generate classifier for different classes of images.

Here we note that in meta-train process, all the labels able ls of images are given. We infer that this meta-learning is a supervised learning. In this thesis, our meta-learning can be considered as a unsupervised learning, where during the meta-train, we don't have the D_{test} to generate performance score (i.e., classification loss).

2.3.4 Related Works and Expansion

Recently, meta-learning has witnessed increasing importance on a variety of machine learning tasks including few-shot learning [1], model-agnostic meta-learning (MAML) [111], training neural networks [108], and learning a gradient-descent based optimizations algorithm [106]. In this thesis, we mainly focus on optimization-based meta-learning, especially the gradient descent based meta-learning.

Using meta-learning to learn a strategy for solving optimization problems is not new. Andrychowicz *et al.* [106] proposed an LSTM meta-learner to learn a gradient-based strategy for updating the parameters of neural networks. The learned strategy outperforms standard approaches including Adam and RMSProp in terms of convergence

speed. This is achieved by using a meta-learner to train a neural network for generalizing the gradient descent update rule. The learned update rule is with the objective of minimizing partial accumulated losses for several gradient descent steps instead of minimizing loss at each step.

In [1], an Long-Short-Term-Memory (LSTM) networks based meta-learner is applied to learn an algorithm for training another neural network that is used as an image classifier as we present above. LSTM network is a widely used in sequential data processing, such as neural language processing. This work makes the utmost of LSTM structural features to learn an update rule for training a network. With respect to the capability of storing the valuable features of data and ignoring trivial information of the cell state in LSTM networks, it sets the cell state as the parameters of the classifier networks, thus the LSTM meta-learner can determine the significance of information about the gradient descent rule through the optimization processes.

Meta-learning has also been applied to find approximate algorithms for gradient based optimization problems [108, 112], and to obtain a superior initialization which can lead to fast convergence in gradient descent procedure by the popular Model-Agnostic Meta-Learning (MAML) algorithm, which is proposed by Finn *et al.* [107, 111]. The key idea of MAML can be concluded as learning a superior base model or initialization for the optimization problem to obtain quick convergence. The learned initialization is able to provide quickly convergence to optimal solution for a set of new tasks.

In summary, these meta-learning methods share the common framework, where the meta-learning is formed as a bi-level optimization model. Two levels of optimizations are constructed for the meta-learning, where the inner optimization generate optimizer with given data and meta-learner to solve each task, and the outer optimization optimizes the meta-learner with respect to the performance of inner optimization. In these works, meta-learning is mainly applied to solve optimization problems with benign landscape. Nevertheless, meta-learning for solving optimization problems which are

non-convex with multiple variables has not been investigated in previous works.

As we mentioned in Section 2.1.2, non-convex multi-variable optimization problem shares similar alternative framework with two levels of optimization as meta-learning does. In this thesis, our works seek to solve non-convex multi-variable optimization problems in meta-learning framework. In contrast to the previous works, we mainly focus on those non-convex optimization problems that are typically intractable in existing works. In Chapter 5, we present that the bi-level framework of meta-learning can be performed well to solve non-convex multi-variable optimization problem that are ill-posed in previous works.

Chapter 3

Blind Estimation for Origin-Destination Flows

3.1 Introduction

In this chapter, we present an application of non-convex optimization technique, where bi-linear model is applied to estimate vehicle flows in transportation networks. The problem of estimating origin-destination (OD) flows from link flows is well studied and a new forward model is proposed. In this new model, the OD flows estimation is cast as a bi-linear inverse problem and is solved by non-convex optimization algorithm. We verify that by reformulating as a non-convex optimization problem, a dramatic dimension reduction has been achieved to allow an estimation of OD flows without any prior information possible.

The problem of interest on this application is to infer OD flows from the link flows. Generally, the traffic assignment matrix describes this relation between OD flows and link flow. In the past decades, accurate estimation of OD flows based on link flows and traffic assignment matrix is a typically ill-posed problem. Then two types of

prior information are typically needed: the parameters (or their approximations) of the model, and some approximations of the OD flows. Our work focuses on the basic linear model for simplicity and for the purpose of proof-of-concept. We assume that the traffic assignment matrix is unknown but fixed during the time period of study. The OD flows are dynamic, meaning that the OD flows can vary across time intervals. We are particularly interested in the case that historical data is not available. As discussed in section 2.2.1, the number of link flows is typically much less than that of OD flows, thus the inverse problem of OD flows estimation is severely ill-posed and hence prior information is required to recover the ground truth.

The basic approach in the literature relies on the forward model (2.2.1), where the so called traffic assignment matrix maps OD flows to link flows. Typical prior information includes approximations of the assignment matrix and OD flows based on historical data, temporal correlations or spatial structures of OD flows, etc. However, we show that, the ill-posed problem can be turned into a well-posed problem by the proposed non-convex optimization model so that the ground truth OD flows can be uniquely identified without any prior information. We refer to the estimation as *blind estimation* as it does not require prior information of either assignment matrix or historical OD flows. The new forward model does not involve OD flows directly but is built upon the flows characterised only by their origins, henceforth referred as *O-flows*. The new model preserves all the OD information and more importantly reduces the dimension of the inverse problem by orders of magnitude. Algorithms for blind estimation, a necessary condition for exact recovery, and a sparsity assumption for exact recovery of particular traffic graph are provided. Simulations demonstrate the new capability of blind estimation, on both of small and large scale networks.

The main contribution of this chapter is, for the first time, to show the possibility to accurately estimate OD flows entirely based on the observed link flows, without prior information of either the assignment matrix or the historical OD flows. For the case

that the network topology does not support a unique solution, we assume that the OD flows exhibit some time domain patterns with unknown parameters. This structural assumption can lead to accurate estimation of the linear model, the parameters to describe time-domain patterns in the OD flows, and the OD flows.

The key technical novelty is a non-convex optimization model which does not involve OD flows directly. More specifically, we define a flow originating from a node as an O-flow. A new linear mapping model is constructed to map O-flows to link flows. With a slight abuse of terminology, we refer to the corresponding linear operator also as traffic assignment. (The two different assignment operators can be distinguished according to the context.) Our formulation based on O-flows has the following properties.

- With given traffic assignment from O-flows to link flows, the linear inverse problem to find O-flows is typically well posed. The number of O-flows is at most the number of nodes, which is typically less than the number of links and can be orders of magnitude smaller than the dimension of OD flows.

- The dimension reduction from our new forward model based on O-flows allows an estimation of both O-flows and the corresponding traffic assignment simultaneously without historical OD flows. In particular cases (in bidirectional networks), a blind estimation of OD flows is feasible. Mathematically, the inference problem is a bi-linear inverse problem.

- Uniqueness of solutions is discussed in Subsection 3.3.2. We observe that the uniqueness of solutions depends on the network topology. In Subsection 3.3.2 we illustrate the necessary condition for uniqueness of solution, and use this to show that a unidirectional network does not support a unique solution.

- Nevertheless, sparsity in O-flows can enforce the uniqueness. Different from the sparsity assumption in [56] where most OD flows are assumed to be zero or tiny, in our assumptions all O-flows count. They are sparse in the sense that

O-flows exhibit time-domain patterns, that is, the coefficients of the discrete cosine transform (DCT) of O-flows have only a few significant components. This assumption is aligned with our daily experience.

- The OD flow information can be obtained from the estimation of O-flows. That is, the model based on O-flows preserves all the OD flow information. See more detailed discussions on this in Section 3.2.2.

For demonstration convenience, multiple simplifications of the traffic model are made in this chapter. As mentioned before, only linear models are assumed to model the traffic flows while in reality traffic loading models are nonlinear according to the fundamental diagram of traffic flow[1]. The linear model is assumed to be static, i.e., the assignment matrix remains unchanged over time. The impacts of random noise and randomness in traffic flows are not studied in this chapter. Flows with a constant and identical speed are assumed. We also exclude any prior information of flows and traffic assignment which may exist in practice.

This chapter is organized as follows. Section 3.2 introduces our new formulation, and discusses the properties of the new formulation. Section 3.3 describes the computational procedure to jointly estimate both O-flows and traffic assignment, and discusses the uniqueness of solutions. Simulations are provided in Section 3.4 to demonstrate the capability of new our models. Conclusions are given in Section 3.5.

3.2 A New Forward Model for OD Estimation

This section presents a new forward model for OD-flow estimation. As discussed in Section 2.2.1, the main technical difficulty of using the standard model (2.2.1) for OD-flow estimation comes from its large dimensionality. Define the total number of nodes

[1]One way to handle non-linearity is to use local linear approximations derived from Taylor series. In this thesis, we focus on linear models.

in the traffic network is n_n. The new model reduces the dimension of the inquired flows from $O(n_n^2)$ to $O(n_n)$, and hence allows a blind estimation for both traffic assignment matrix and OD flows.

3.2.1 O-flow Mapping Models

We build a forward model not directly involving the inquired OD-flows. It is built on the traffic flows that are specified with their origins but not their destinations, henceforth referred to as *O-flow*. As defined in Section 2.2.1, $\boldsymbol{y}^t$, $\boldsymbol{A}^t$, and $\boldsymbol{s}^t$ are the link flow, traffic assignment matrix, and the OD flow at time instance t, respectively. Let x_o denote the flow originating from node o, i.e., $x_o = \sum_{d \neq o} s_{od}$. Define the O-flow vector as $\boldsymbol{x} = [x_1, \cdots, x_o, \cdots, x_{n_n}]^T$. Let $p_{i \to j,o}$ denote the proportion of the flow originating from node o that passes the link $i \to j$. Denote the traffic assignment matrix associated with O-flows $\boldsymbol{x}$ by $\boldsymbol{P}$ of which the entries are $p_{i \to j,o}$. Assume constant traffic assignment and dynamic flows. When the sampling time interval is longer than the trip time (the maximum trip time is $\tau_{\max}$), one has the following single step model

$$\boldsymbol{y}^t = \boldsymbol{P}\boldsymbol{x}^t, \quad t = 1, 2, \cdots, n_T, \tag{3.2.1}$$

where $n_T \in \mathbb{Z}^+$ denotes the number of sampling time intervals during the time period of interest. Otherwise, one has a multiple step model where

$$\boldsymbol{y}^t = \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{P}^\tau \boldsymbol{x}^{t-\tau+1}, \text{ or } \boldsymbol{y}^t = \boldsymbol{P}^t * \boldsymbol{x}^t, \quad t = 1, 2, \cdots, n_T. \tag{3.2.2}$$

Remark 4 *In the sequel, we will often use single step models for illustration due to their simplicity.*

It is clear that the dimension of the O-flow vector $\boldsymbol{x}^t$ is upper bounded by n_n. Assume the typical scenario where the number of links $n_l = cn_n$ with $c > 1$. When the traffic

assignment is given, the number of equations is more than the number of unknowns and hence the inverse problem is well-posed (assuming the network topology implies that the assignment matrices are of full column rank). In the blind estimation setting in Section 3.3, we shall assume that neither O-flows nor assignment matrices are known.

Though the new model does not involve OD-flows directly, such information is preserved. We prove it in the single step model (3.2.1) and multiple step model (3.2.2) as follows.

Proposition 1 *All the OD flows, can be inferred from the given O-flows and the corresponding assignment matrix.*

Proof *In the single step model (3.2.1), the OD flow s_{od}, $o \neq d$ originating from the node o and ending at the node d can be calculated from*

$$s_{od} = x_o \left(\sum_i p_{i \to d,o} - \sum_j p_{d \to j,o} \right), \tag{3.2.3}$$

where the term $x_o \sum_i p_{i \to d,o}$ calculates the inflow to node d that originating from node o, the term $x_o \sum_j p_{d \to j,o}$ gives the outflow from node d originating from node o, and thus the difference between them is clearly the flow ending at node d that originates from node o.

In the multiple step model (3.2.2). All the OD flows during the t-th time interval s^t_{od}, $o \neq d$,$t = 1, 2, \cdots, n_T$, can be inferred from given O-flows during the t-th time interval and the corresponding assignment matrix via

$$s^t_{od} = x^t_o \left(\sum_{\tau=1}^{\tau_{\max}} \sum_i p^\tau_{i \to d,o} - \sum_{\tau=1}^{\tau_{\max}} \sum_j p^\tau_{d \to j,o} \right). \tag{3.2.4}$$

Remark 5 *The concept of O-flows has been mentioned in the literature, e.g., [40]. However, when coming to OD-flow estimation, the inverse problem is built upon path*

flows in [40] rather than O-flows directly. Hence, the approach there does not enjoy the benefit of dimension reduction.

Remark 6 *A model similar to (3.2.1) can be constructed based on the D-flow as well (D stands for destination) which also allows for dimensional reduction and the preservation of OD flow information. More specifically, let x_d be the flows ending at the node d, and $p_{i\to j,d}$ be the proportion of x_d passing the link $i \to j$. Then the OD flow can be computed via*

$$s_{od} = x_d \left(\sum_j p_{o\to j,d} - \sum_i p_{i\to o,d} \right)$$

for single step model, and

$$s_{od}^t = x_o^t \left(\sum_{\tau=1}^{\tau_{\max}} \sum_j p_{o\to j,d}^\tau - \sum_{\tau=1}^{\tau_{\max}} \sum_i p_{i\to o,d}^\tau \right)$$

for multiple step model. The models based on O-flow and D-flow are duality of each other. This thesis focuses on the O-flow model only.

3.2.2 Connections and Differences of Models

On one hand, our model preserves the OD flow information via (3.2.3). On the other hand, the number of unknown variables (including both traffic assignment and traffic flows) in our new model (3.2.1) are substantially reduced from the standard model (2.2.1). An educated instinct is that some information must get lost by this dimension reduction. To characterise the exact information that is lost and to understand its importance, we need to study the relationship among three different models built upon path flows, OD-flows, and O-flows, respectively. For simplicity of discussion, we focus on a single snapshot of single step models.

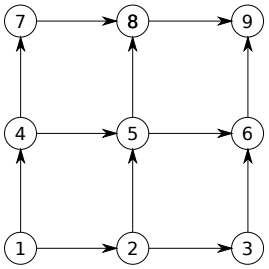


Figure 3.1: An example of unidirectional network with nine nodes and twelve links.

Many papers [40, 52, 113] in the literature use path flow models. For a given path $\boldsymbol{p}$ specified by $o \to i \to \cdots \to d$, denote the flow along this particular path by $s_{\boldsymbol{p}}^{\text{path}} = s_{o\to i\to\cdots\to d}^{\text{path}}$. Group all path flows into a vector to form path flow vector. Then the observed link flows are given by

$$\boldsymbol{y} = \boldsymbol{A}^{\text{path}} \boldsymbol{s}^{\text{path}},$$

where $\boldsymbol{A}^{\text{path}}$ is the path incidence matrix where

$$A_{i\to j,\boldsymbol{p}}^{\text{path}} = \begin{cases} 1 & \text{if path } \boldsymbol{p} \text{ involves } i \to j, \\ 0 & \text{otherwise.} \end{cases}$$

The path flow model preserves all the information of the OD-flow model, but not vice versa. Let $\mathcal{P}_{od}$ be the set of all paths with origin o and destination d. It is straightforward to verify that

$$\begin{aligned} s_{od} &= \sum_{\boldsymbol{p}\in\mathcal{P}_{od}} s_{\boldsymbol{p}}^{\text{path}}, \text{ and} \\ a_{i\to j,od} &= \frac{\sum_{\boldsymbol{p}\in\mathcal{P}_{od}} a_{i\to j,\boldsymbol{p}}^{\text{path}} s_{\boldsymbol{p}}^{\text{path}}}{\sum_{\boldsymbol{p}\in\mathcal{P}_{od}} s_{\boldsymbol{p}}^{\text{path}}}. \end{aligned} \tag{3.2.5}$$

However, complete information of the OD-flow model does not fully characterise the path flow model in general. Specifically, we use the following example to illustrate that

the lost information is the information of path selection when multiple paths share the same subset of links.

Example 1 *In the unidirectional network in Figure 3.1, consider the four different paths* $\boldsymbol{p}_1 = 1 \to 2 \to 5 \to 6 \to 9$, $\boldsymbol{p}_2 = 1 \to 2 \to 5 \to 8 \to 9$, $\boldsymbol{p}_3 = 1 \to 4 \to 5 \to 6 \to 9$, *and* $\boldsymbol{p}_4 = 1 \to 4 \to 5 \to 8 \to 9$ *which correspond the same OD pair* 19. *Suppose that the observed link flows of all involved links are* 100. *This link flow information uniquely identifies the OD-flow* s_{19} *and the corresponding assignment:* $s_{19} = 200$ *and* $a_{i\to j,19} = \frac{1}{2}$ *for all involved links* $i \to j$. *However, consistent path flow models are not unique: that* $\boldsymbol{p}_1 = \boldsymbol{p}_4 = 100$ *and* $\boldsymbol{p}_2 = \boldsymbol{p}_3 = 0$, *that* $\boldsymbol{p}_2 = \boldsymbol{p}_3 = 100$ *and* $\boldsymbol{p}_1 = \boldsymbol{p}_4 = 0$, *and any convex combination of these two cases can give the exactly same link flows.*

Similar relationship is true from the OD-flow model to the O-flow model. The O-flow model can be derived from the OD flow model via

$$\begin{aligned} x_o &= \sum_{d\neq o} s_{od}, \text{ and} \\ P_{i\to j,o} &= \frac{\sum_{d\neq o} a_{i\to j,od} s_{od}}{\sum_{d\neq o} s_{od}}, \end{aligned} \tag{3.2.6}$$

but not vice versa. An example is given below to show that it is the path selection information that is lost.

Example 2 *Consider the same unidirectional network in Figure 3.1. Consider the OD flows* s_{18} *and* s_{16}. *Further assume that* s_{18} *can be generated by only two possible paths* $\boldsymbol{p}_1 = 1 \to 2 \to 5 \to 8$ *and* $\boldsymbol{p}_2 = 1 \to 4 \to 5 \to 8$, *and that* s_{16} *can be generated by only two possible paths* $\boldsymbol{p}_3 = 1 \to 2 \to 5 \to 6$ *and* $\boldsymbol{p}_4 = 1 \to 4 \to 5 \to 6$. *Suppose that the observed link flows of all involved links are* 100. *This uniquely identifies the corresponding O-flow model:* $x_1 = 200$ *and* $p_{i\to j,1} = \frac{1}{2}$ *for all involved links. However, there is no unique OD-flow model. The OD-flows are uniquely given by* $s_{18} = s_{16} = 100$. *But not so for the assignment: the case that* $s_{18} = s^{\text{path}}_{\boldsymbol{p}_1}$ *and* $s_{16} = s^{\text{path}}_{\boldsymbol{p}_4}$, *the case that*

$s_{18} = s^{\text{path}}_{\boldsymbol{p}_2}$ and $s_{16} = s^{\text{path}}_{\boldsymbol{p}_3}$, and any convex combination of these two cases give the exactly same link flows.

It is clear that the path selection information gets partially lost in both cases. In traffic analysis applications, if the OD-flow model serves the purpose, then most likely the O-flow model also does. The fact that the OD-flow model is applicable already implies that the path selection information is not critical for the application scenario.

3.3 Estimation of OD-flows

As advertised before, the substantial reduction of dimension in our new mapping model allows a new estimation of OD-flows, that is, recover the ground truth OD-flows without prior information on either OD-flows or the corresponding assignment matrix. This is achieved by two step procedures. In the first step, we estimate both the O-flows and the corresponding assignment matrices. In the second step, we recover the OD-flows using the results from the first step. The focus of this section is the first step as the second step is straightforward by using Equation (3.2.3).

3.3.1 Blind Estimation Model

Let $\boldsymbol{y}^t$, $t = 1, 2, \cdots, n_T$, be the measured link flows. Assume that $\tau_{\max} < n_T$. Then the joint estimation problems can be formulated as

$$\min_{\boldsymbol{P},\, \boldsymbol{x}^t:\, 1 \le t \le n_T} f_{\text{cost}}\left(\boldsymbol{P}, \left\{\boldsymbol{x}^t\right\}\right)$$

and

$$\begin{aligned} &\min && f_{\text{cost}}\left(\left\{\boldsymbol{P}^t\right\}, \left\{\boldsymbol{x}^t\right\}\right) \\ &\boldsymbol{P}^t:\ 1 \le t \le \tau_{\max} \\ &\boldsymbol{x}^t:\ 2-\tau_{\max} \le t \le n_T \end{aligned}$$

for the single step and multiple step models, respectively. f_{cost} relies on the underlying assumption on noise and typically is chosen to be a convex function for computational convenience. In this thesis, we adopt the most commonly used least squared norm for f_{cost}, resulting in

$$\min_{\boldsymbol{P},\ \{\boldsymbol{x}^t\}} \quad \sum_{t=1}^{n_T} \left\|\boldsymbol{y}^t - \boldsymbol{P}\boldsymbol{x}^t\right\|_2^2 \tag{3.3.1}$$

for the single step model, and

$$\min_{\left\{\boldsymbol{P}^t\right\},\ \{\boldsymbol{x}^t\}} \quad \sum_{t=1}^{n_T} \left\| \boldsymbol{y}^t - \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{P}^\tau \boldsymbol{x}^{t-\tau+1} \right\|_2^2 \tag{3.3.2}$$

for the multiple step model.

The following constraints imposed by the physics and network topology should be also included.

C1: Non-negativity of flows. That is,

$$\boldsymbol{x}^t \ge \boldsymbol{0},\ \forall t.$$

C2: Probability constraint on assignment matrix. By the definition of traffic assignment,

$$0 \le p_{i\to j,o}^t \le 1$$

for multiple step models, and

$$0 \le p_{i\to j,o} \le 1$$

for single step models.

C3: Observability constraint. This means that all the traffic originated from a node is observed. That is, for multiple step models, one has

$$\sum_{\tau=1}^{\tau_{\max}} \sum_{i \neq o} p^{\tau}_{o \to i,o} = 1, \ \forall o,$$

and for single step models it holds that

$$\sum_{i \neq o} p_{o \to i,o} = 1, \ \forall o.$$

This constraint is consistent with the principles of flow conservation mentioned in [48] derived for the OD flows using principles of flow conservation.

C4: Reachability constraint.

For multiple step models, one has that $p^t_{i \to j,o} \geq 0$ if the link $i \to j$ is involved in the t-th step of the O-flow x_o, and $p^t_{i \to j,o} = 0$ otherwise. For example, under the rigid multiple step model stated in Remark 8, $p^t_{o \to i,o} \geq 0$ for $t = 1$, $p^t_{o \to i,o} = 0$ for $t > 1$, and $p^t_{i \to j,o} = 0$ for $t = 1, \ \forall o, \ \forall i \neq o$. The complete set of speed constraints depends on the network topology. Application of the speed constraint typically makes $\boldsymbol{P}^t$ sparse especially when the sampling time interval is small compared to maximum trip time.

For single step models, it holds that $p_{i \to j,o} = 0$ when a link $i \to j$ is not reachable within the maximum trip time $\tau_{\max}$.

C5: Flow constraint. Consider the flow originated from any node o. For any given node $i \neq o$, its inflow must be larger than or equal to its outflow. For multiple step models, one has

$$\sum_{\tau=1}^{t} \sum_{j} p^{\tau}_{j \to i,o} - \sum_{\tau=t}^{\tau_{\max}} \sum_{k} p^{\tau}_{i \to k,o} \geq 0, \ \forall t, \ \forall o, \text{ and } \forall i \neq o,$$

which can be simplified into

$$\sum_j p^{t-1}_{j\to i,o} - \sum_k p^t_{i\to k,o} \geq 0,\ \forall t,\ \forall o,\ \text{and}\ \forall i \neq o,$$

for rigid model, referring to Remark 8. For single step models, it holds that

$$\sum_j p_{j\to i,o} - \sum_k p_{i\to k,o} \geq 0,\ \forall o,\ \text{and}\ \forall i \neq o.$$

The problem (3.3.2) is known as a bi-linear inverse problem [114]. The name comes from the fact that when one fixes either $\{\boldsymbol{P}^t\}$ or $\{\boldsymbol{x}^t\}$ and solves for the other, the inverse problem becomes a linear inverse problem. Generally speaking, bi-linear inverse problems are non-convex and does not admit unique solutions. Nevertheless, discussions in Section 3.3.2 and simulations in Section 3.4 show the possibility to reach a unique optimal solution via non-convex optimization approach with above structural constraints C1-C5.

Remark 7 *Although the non-convex optimization does not guarantee unique solutions, the estimated OD flows through the proposed blind estimation approach have little variations to the ground truth. Therefore, we infer that the obtained solutions are approximately equal to the optimal solutions.*

We present the Gauss-Seidel method for OD estimation and the following description is based on multiple step model. It is an AM approach where in each iteration either $\{\boldsymbol{P}^t\}$ or $\{\boldsymbol{x}^t\}$ is fixed and the minimization is with respect to the other variable. See Algorithm 2 for a high level description.

Remark 8 *At this point, it is worth to distinguish two traffic models: rigid and elastic traffic models, respectively. This simulation part focuses on a rigid multiple step model, where all links are of the same length, and in one unit time all the unfinished traffic flows move forward across exactly one link. In other words, all the traffic flows have*

constant and identical speeds. In the elastic model, the constraints on both link length and traffic speed are removed. In one unit time some proportions of the traffic may cross multiple links and some proportions of the traffic may require more than one unit time to finish a single link; and the average travel time for different links can be different. The elastic model fits actual systems better. The elastic model will dramatically increase the complexity of traffic assignment matrix in our model. The unknown elements in traffic assignment matrix will be much more lager in elastic model. For computational simplicity, we focus on the rigid model in this simulation part.

Remark 9 (Extensions to Accommodate Prior Information) *The formulation (3.3.2) can be extended to accommodate prior information by adding extra terms into the objective function. For example, when approximations of* $\{\boldsymbol{P}^t\}$ *and* $\{\boldsymbol{x}^t\}$*, denoted by* $\left\{\tilde{\boldsymbol{P}}^t\right\}$ *and* $\{\tilde{\boldsymbol{x}}^t\}$*, are available from historical data [56, 62, 67], recall to GLS model in Section equation (2.2.3), one may formulate the estimation problem as*

$$\min_{\{\boldsymbol{P}^t\},\,\{\boldsymbol{x}^t\}} \sum_{t=1}^{n_T} \left\| \boldsymbol{y}^t - \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{P}^{\tau} \boldsymbol{x}^{t-\tau+1} \right\|_2^2 + \sum_{t=1}^{\tau_{\max}} \operatorname{vect}\left(\boldsymbol{P}^t - \tilde{\boldsymbol{P}}^t\right)^T \boldsymbol{W}_P^{-1} \operatorname{vect}\left(\boldsymbol{P}^t - \tilde{\boldsymbol{P}}^t\right) + \sum_{t=2-\tau_{\max}}^{n_T} \left(\boldsymbol{x}^t - \tilde{\boldsymbol{x}}^t\right)^T \boldsymbol{W}_X^{-1} \left(\boldsymbol{x}^t - \tilde{\boldsymbol{x}}^t\right),$$

where $\operatorname{vect}(\cdot)$ *denotes the vector formed by stacking the columns of the input matrix, and* $\boldsymbol{W}_P$ *and* $\boldsymbol{W}_X$ *are the covariance matrices of the error terms. When fixing* $\{\boldsymbol{P}^t\}$ *and updating* $\{\boldsymbol{x}^t\}$*, the corresponding optimization problem becomes the generalized least squares estimator which has been widely used in the literature for decades [43].*

Remark 10 *The subtle point is the time intervals of the output* $\boldsymbol{x}^t$*. From the definition of the convolution,* $\boldsymbol{x}^t,\ 2-\tau_{\max} \le t \le n_T$*, is involved in producing* $\boldsymbol{y}^t,\ 1 \le t \le n_T$*. However, it is clear that with given information* $\boldsymbol{y}^t,\ 1 \le t \le n_T$*, it is impossible to uniquely recover* $\boldsymbol{x}^t$ *for* $2-\tau_{\max} \le t \le 0$ *in general. As a consequence, the outputs only involve* $\boldsymbol{x}^t$ *for* $1 \le t \le n_T$*.*

Algorithm 2: Alternative minimization to solve the bi-linear inverse problem (3.3.2).

1 **Input**: The graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$ and link flows $\boldsymbol{y}^t$, $t = 1, 2, \cdots, n_T$

2 **Output**: $\boldsymbol{P}^t$, $t = 1, 2, \cdots, \tau_{\max}$, and $\boldsymbol{x}^t$, $t = 1, 2, \cdots, n_T$

3 **Initialization**: Based on $\mathcal{G}(\mathcal{N}, \mathcal{L})$ randomly initialize $\{\boldsymbol{P}^t\}$ subject to Constraints C2-C5.

4 **Iteration** until the exit criteria are satisfied:

- With fixed $\{\boldsymbol{P}^t\}$, update $\{\boldsymbol{x}^t\}$ via minimizing f_{cost} with respect to $\{\boldsymbol{x}^t\}$ subject to Constraint C1.
- With fixed $\{\boldsymbol{x}^t\}$, update $\{\boldsymbol{P}^t\}$ via minimizing f_{cost} with respect to $\{\boldsymbol{P}^t\}$ subject to Constraints C2-C5.

The two least squares problems involved in each iteration are detailed as follows: For the update of $\{\boldsymbol{x}^t\}$, the objective function of the least square problem is given by

$$\left\| \begin{bmatrix} \boldsymbol{y}^1 \\ \boldsymbol{y}^2 \\ \vdots \\ \boldsymbol{y}^{n_T} \end{bmatrix} - \begin{bmatrix} \boldsymbol{P}^{\tau_{\max}} & \boldsymbol{P}^{\tau_{\max}-1} & \cdots & \boldsymbol{P}^1 & & \\ & \boldsymbol{P}^{\tau_{\max}} & \boldsymbol{P}^{\tau_{\max}-1} & \cdots & \boldsymbol{P}^1 & \\ & & \ddots & \ddots & & \ddots \\ & & & \boldsymbol{P}^{\tau_{\max}} & \boldsymbol{P}^{\tau_{\max}-1} & \cdots & \boldsymbol{P}^1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{2-\tau_{\max}} \\ \boldsymbol{x}^{3-\tau_{\max}} \\ \vdots \\ \boldsymbol{x}^{n_T} \end{bmatrix} \right\|_2^2 .$$

When updating $\{\boldsymbol{P}^t\}$, it is given as follows:

$$\left\| \begin{bmatrix} (\boldsymbol{y}^1)^T \\ (\boldsymbol{y}^2)^T \\ \vdots \\ (\boldsymbol{y}^{n_T})^T \end{bmatrix} - \begin{bmatrix} (\boldsymbol{x}^1)^T & (\boldsymbol{x}^0)^T & \cdots & (\boldsymbol{x}^{2-\tau_{\max}})^T \\ (\boldsymbol{x}^2)^T & (\boldsymbol{x}^1)^T & \cdots & (\boldsymbol{x}^{3-\tau_{\max}})^T \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{x}^{n_T})^T & (\boldsymbol{x}^{n_T-1})^T & \cdots & (\boldsymbol{x}^{n_T-\tau_{\max}+1})^T \end{bmatrix} \begin{bmatrix} (\boldsymbol{P}^1)^T \\ (\boldsymbol{P}^2)^T \\ \vdots \\ (\boldsymbol{P}^{\tau_{\max}})^T \end{bmatrix} \right\|_F^2 .$$

In this stage, we infer that despite the intrinsic non-convexity of the objective function, the above AM algorithm is expected to converge to a global optimum with respect

to the benign landscape constructed by structural constraints C1-C5. One significant factor is the dimension of the observed link flows data set $\boldsymbol{y}^{t}{}_{t=1}^{n_T}$. This data set should be sufficiently large enough to ensure the unique solutions. We will present the uniqueness of the solution next.

3.3.2 Uniqueness of Solution

In the discussion of the uniqueness, we assume the noise free case where $\boldsymbol{y}^t = \boldsymbol{P}^t * \boldsymbol{x}^t$. Denote the estimated assignment matrices and O-flows by $\left\{\hat{\boldsymbol{P}}^t\right\}$ and $\{\hat{\boldsymbol{x}}^t\}$, respectively. It holds that $\boldsymbol{y}^t = \hat{\boldsymbol{P}}^t * \hat{\boldsymbol{x}}^t$. The uniqueness of the solution implies that the estimated $\left\{\hat{\boldsymbol{P}}^t\right\}$ and $\{\hat{\boldsymbol{x}}^t\}$ are the same as the ground truth.

The solution of a general bi-linear problem is not unique: if $\left\{\hat{\boldsymbol{P}}^t\right\}$ and $\{\hat{\boldsymbol{x}}^t\}$ are a solution of $\boldsymbol{y}^t = \boldsymbol{P}^t * \boldsymbol{x}^t$, $1 \leq t \leq n_T$, then so are $\left\{\hat{\boldsymbol{P}}^t\boldsymbol{M}\right\}$ and $\{\boldsymbol{M}^{-1}\hat{\boldsymbol{x}}^t\}$ for arbitrary invertible matrix $\boldsymbol{M}$. In our problem, constraints C1-C5 help avoid such ambiguity. For example, constrain C1 avoids sign ambiguity, and Constraint C3 implies that each column of $\boldsymbol{P}^1$ has unit ℓ_1-norm for multiple step models. The solution of the estimation problem (3.3.2) can be unique.

Necessary Condition for the Uniqueness of Solution

The following theorem states the necessary condition for the uniqueness of the solution of Equations (3.2.1) and (3.2.2). Note that n_o is the number of origin nodes in the networks (in bidirectional network, generally $n_o = n_n$), which is equal to the number of equations referring to the observability constraint C3, and n_4 represents the number of equations, referring to the reachability constraint C4.

Theorem 1 *If the solution of Equation (3.2.1) with constraints C1-C5 is unique, the*

following necessary condition for uniqueness of solution

$$n_\ell n_T + n_o + n_4 \geq n_\ell n_o + n_o n_T$$

holds. If the solution of Equation (3.2.2) with constraints C1-C5 is unique, the following inequality

$$n_\ell n_T + n_o + n_4 \geq \tau_{\max} n_\ell n_o + n_o n_T$$

holds.

To estimate an n_T to allow the uniqueness of the solution, a relaxed but less accurate version can be derived as follows. Firstly, the above two inequalities are equivalent to

$$n_T \geq \frac{(n_l - 1)\, n_o - n_4}{n_\ell - n_o}, \tag{3.3.3}$$

and

$$n_T \geq \frac{(\tau_{\max} n_l - 1)\, n_o - n_4}{n_\ell - n_o}. \tag{3.3.4}$$

Define $c := n_\ell / n_o$ which is the average number of links per node. For typical transportation networks $1 \leq c \leq 4$. We relax the right hand side of the inequalities (3.3.3) and (3.3.4) by ignoring the terms $-n_o$, $-n_4$ in the numerators. Then a useful rule of thumb for practice can be obtained where

$$n_T \gtrapprox \frac{c}{c-1} n_o \tag{3.3.5}$$

for single step models, and

$$n_T \gtrapprox \frac{\tau_{\max} c}{c-1} n_o \tag{3.3.6}$$

for multiple step models.

Non-uniqueness of Solutions for Unidirectional Networks

In the following, we shall use the necessary condition to show that in general unidirectional networks do not admit a unique solution. We start the analysis by studying a three-node unidirectional network. For simplicity, we focus on the rigid model mentioned in Remark 8.

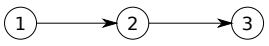


Figure 3.2: A three-node unidirectional network

Lemma 1 *Consider the three-node unidirectional network depicted in Figure 3.2. It does not admit a unique solution in general due to the limited data set* $\boldsymbol{y}^{t}{}_{t=1}^{n_T}$. *(The case that admits a unique solution is detailed in the proof.)*

Proof *We start with the single step model. For the unidirectional network in Figure 3.2, Constraints C3 and C4 imply that* $p_{1\to 2,1}=1$, $p_{1\to 2,2}=0$, *and* $p_{2\to 3,2}=1$. *The system becomes*

$$\begin{bmatrix} \cdots & y_{1\to 2}^t & \cdots \\ & y_{2\to 3}^t & \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ p_{2\to 3,1} & 1 \end{bmatrix} \begin{bmatrix} \cdots & x_1^t & \cdots \\ & x_2^t & \end{bmatrix}, \ 1 \le t \le n_T.$$

The number of equations is $2n_T$ *while the number of unknowns is* $1+2n_T$. *Therefore, this network does not admit a unique solution in general.*

A more careful study reveals that $x_1^t = y_{1\to 2}^t$, $\forall t$, *and*

$$p_{2\to 3,1} \in \left[0, \min\left(1, \min_{1\le t\le n_T} \frac{y_{2\to 3}^t}{x_1^t}\right)\right] = \left[0, \min\left(1, \min_{1\le t\le n_T} \frac{y_{2\to 3}^t}{y_{1\to 2}^t}\right)\right]. \tag{3.3.7}$$

A unique solution exists if and only if $y_{2\to 3}^t = 0$ *and* $y_{1\to 2}^t \neq 0$ *for some* $t \in \{1, \cdots, n_T\}$, *which corresponds the solution that* $p_{2\to 3,1}=0$ *and* $y_{2\to 3}^t = x_2^t$.

The proof for the multiple step model is similar and hence omitted here. The solution is

not unique except that $y_{2\to3}^{t+1}=0$ *and* $y_{1\to2}^{t}\neq 0$ *for some* $t\in\{1,\cdots,n_T-1\}$*, in which case* $p_{2\to3,1}^{2}=0$.

The following corollary is a direct application of Lemma 1.

Corollary 1 *Unidirectional networks that contain Figure 3.2 as a subgraph do not admit a unique solution in general.*

The above negative result for unidirectional networks can be addressed by assuming that the O-flows are sparse. In particular, define the vector $\boldsymbol{x}_o = [x_o^1,\cdots,x_o^{n_T}]^T$, $\forall o\in\{1,\cdots,n_n\}$. This vector is called *sparse* if the number of non-zero entries in $\boldsymbol{x}_0$ is much less than the zero entries.

Go back to the three-node unidirectional network in Figure 3.2 and recall the range of $p_{2\to3,1}$ given in (3.3.7) in the proof for Lemma 1. Suppose that

$$\min_{1\le t\le n_T}\frac{y_{2\to3}^t}{y_{1\to2}^t}\le 1.$$

Then the solution

$$p_{2\to3,1}=\min_{1\le t\le n_T}\frac{y_{2\to3}^t}{y_{1\to2}^t}$$

gives the sparsest solution $\boldsymbol{X}$ where $x_2^t=0$ for all

$$t\in\left\{\tilde{t}:\ \frac{y_{2\to3}^{\tilde{t}}}{y_{1\to2}^{\tilde{t}}}=\min_{1\le t'\le n_T}\frac{y_{2\to3}^{t'}}{y_{1\to2}^{t'}}\right\}.$$

All other solutions

$$p_{2\to3,1}\in\left[0,\min_{1\le t\le n_T}\frac{y_{2\to3}^t}{x_1^t}\right)$$

result in less sparse solutions, i.e., $x_2^t>0$ for all t.

For general unidirectional networks, we infer that certain sparsity patterns in $\boldsymbol{X}$ support a unique solution. In particular, let $\mathcal{T}_{o,\mathrm{nz}}=\{t:\ x_o^t\neq 0\}$. We consider that if

$\mathcal{T}_{o,\mathrm{nz}} \nsubseteq \bigcup_{o'} \mathcal{T}_{o',\mathrm{nz}}$ for all $o \in \{1, \cdots, n_n\}$, then the single step model $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{X}$ admits a unique solution of $\boldsymbol{P}$ and $\boldsymbol{X}$ with sufficiently large n_T and diverse $\boldsymbol{X}$.

Here we claim that the sparsity constraint is prior information. In contrast to historic data, it is much more cheap and easy to be obtained. Nevertheless, in the following description and simulations, we will distinguish estimation with sparsity assumption from blind estimation.

It has been assumed that OD flows are sparse, e.g., in [56], to mitigate the ill-posedness. However, the way that the sparsity assumption is applied in this thesis is significantly different. In [56], it has been assumed that among the majority of OD flows, only a small fraction of them are significant and the rest can be safely approximated zero. Then the authors use historical data to identify which OD flows are significant. However, in the concern of O-flows, a zero O-flow means that there is no traffic originate from the corresponding node, which may not be true especially when large city communities are analyzed. Furthermore, to show the potentials of our model for blind estimation, it is not approximate to assume the prior knowledge of which O-flows are nearly zero.

A more reasonable assumption is that O-flows are sparse in a transform domain. For example, we may assume that they have temporal patterns. Consider the time series represented in a vector form $\boldsymbol{x}_o = [x_o^1, \cdots, x_o^{n_T}]^T$. Assume that for all o, $\boldsymbol{x}_o$ is sparse under an invertible transform $\boldsymbol{D}$, i.e., $\boldsymbol{D}^{-1}$ exists and most entries of $\boldsymbol{c}_o := \boldsymbol{D}^{-1}\boldsymbol{x}_o$ are zeros. Then the single step model can be written as $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{X} = \boldsymbol{P}\boldsymbol{C}\boldsymbol{D}^T$, where the rows of the matrix $\boldsymbol{C}$ are $\boldsymbol{c}_o^T$, $o \in \{1, \cdots, n_n\}$. Define $\tilde{\boldsymbol{Y}} = \boldsymbol{Y}\left(\boldsymbol{D}^T\right)^{-1} = \boldsymbol{Y}\left(\boldsymbol{D}^{-1}\right)^T$. Then $\tilde{\boldsymbol{Y}} = \boldsymbol{P}\boldsymbol{C}$. The previous analysis for single step model is still valid. In this thesis, the transform is chosen to be discrete cosine transform (DCT) where the transformation matrix $\boldsymbol{D}$ is orthonormal.

In practice, the origin flows in a traffic network usually follow specific distributions over time sequence. For example, the origin flows from residential areas in the morning typically much more lager than those in the evening. In the light of this, we consider

a sparsity assumption when optimizing O-flows.

3.3.3 Estimation with the Sparsity Assumption

As shown in Section 3.3.2, the sparsity assumption is essential for unidirectional networks to admit a unique solution. Following the notations at the end of Section 3.3.2, we assume that $\boldsymbol{x}_o$ is sparse under an orthonormal transformation matrix $\boldsymbol{D}$, that is, $\boldsymbol{x}_o = \boldsymbol{D}\boldsymbol{c}_o$ and the number of non-zeros in $\boldsymbol{c}_o$ is much less than n_T the dimension of $\boldsymbol{c}_i$. This transformation matrix $\boldsymbol{D}$ is given. To enforce this sparsity constraint, the estimation problem can be written in the following form:

$$\begin{aligned}\min_{\{\boldsymbol{P}^t\},\,\{\boldsymbol{x}^t\}} \; & \sum_{t=1}^{n_T} \frac{1}{2} \left\| \boldsymbol{y}^t - \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{P}^\tau \boldsymbol{x}^{t-\tau+1} \right\|_2^2 + \sum_{o=1}^{n_n} \lambda_o \left\| \boldsymbol{c}_o \right\|_0 \\ & \text{s. t. } \boldsymbol{x}_i = \boldsymbol{D}\boldsymbol{c}_i,\end{aligned}$$

where $\lambda_o \geq 0$, $o = 1, \cdots, n_n$, are appropriately chosen parameters, and $\|\cdot\|_0$ denotes the ℓ_0 pseudo-norm which counts the number of nonzero elements. The ℓ_0 pseudo-norm is not convex, resulting in a combinational optimization problem. The common practice is to replace it with its convex envelope ℓ_1-norm where $\|\boldsymbol{c}\|_1 := \sum_j |c_j|$. One has

$$\min_{\{\boldsymbol{P}^t\},\,\{\boldsymbol{x}^t\}} \; \sum_{t} \frac{1}{2} \left\| \boldsymbol{y}^t - \sum_{\tau=1}^{\tau_{\max}} \boldsymbol{P}^\tau \boldsymbol{x}^{t-\tau+1} \right\|_2^2 + \sum_{o=1}^{n_n} \lambda_o \left\| \boldsymbol{D}\boldsymbol{x}_o \right\|_1 . \tag{3.3.8}$$

This formulation looks very similar to the famous LASSO form [115, 116]. The major difference is that the problem in (3.3.8) is non-convex due to the bi-linear formulation.

The regularization parameters λ_o in (3.3.8) have to be carefully chosen. According to the authors' knowledge, there is no recipe to choose the optimal values of the regularization parameters λ_o. Typically, their values are set by trial and error.

To minimize possible efforts of parameter tuning, we assume that the level of uncer-

tainty in link flow observations is known a priori (which is acceptable for real applications) and design the following constrained optimization approach. Suppose that the relative level of noise is upper bounded by $\epsilon > 0$, i.e.,

$$\frac{\sum_t \|\boldsymbol{y}^t - \boldsymbol{P}^t * \boldsymbol{x}^t\|_2^2}{\sum_t \|\boldsymbol{y}^t\|_2^2} \leq \epsilon.$$

We have the following formulation:

$$\begin{aligned} \min_{\{\boldsymbol{P}^t\},\,\{\boldsymbol{x}^t\}} \; & \sum_{o=1}^{n_n} \|\boldsymbol{D}\boldsymbol{x}_o\|_1 \\ \text{s. t.} \; & \sum_{t=1}^{n_T} \left\|\boldsymbol{y}^t - \boldsymbol{P}^t * \boldsymbol{x}^t\right\|_2^2 \leq \epsilon \sum_{t=1}^{n_T} \left\|\boldsymbol{y}^t\right\|_2^2. \end{aligned} \tag{3.3.9}$$

The Gauss-Seidel method can be applied to solve the optimization problem iteratively.

However, there is a technical issue for early iterations when the Gauss-Seidel method is applied. In the early stage of the iterations, the estimated $\left\{\hat{\boldsymbol{P}}^t\right\}$ and $\{\hat{\boldsymbol{x}}^t\}$ may not satisfy the constraint specified in (3.3.9). To address this issue, one may first run the Gauss-Seidel method for the objective function (3.3.2) until the estimated $\left\{\hat{\boldsymbol{P}}^t\right\}$ and $\{\hat{\boldsymbol{x}}^t\}$ satisfy the constraint in (3.3.9), and then apply the Gauss-Seidel method for the problem (3.3.9). In the case that the convergence rate is too slow, one may first relax $\epsilon > 0$ to a larger value $\epsilon' > \epsilon$, and then swap back to ϵ after a convergence for ϵ'.

3.4 Simulations and Results

With the benefit of dimension reduction via incorporating the new forward model based on O-flows, we show that a blind estimation for OD flows is feasible in particular cases. The numerical tests are proceeded in the scenarios that neither historical OD data nor traffic assignment matrix is given. The tests in bidirectional networks with multiple step models obtains some exciting results, which perfectly demonstrates the feasibility

of blind estimation. Single step models are also workable in bidirectional networks, though the results are not as good as multiple step models. As discussed in Subsections 3.3.2, an unique solution may not be guaranteed for unidirectional networks. Our tests confirm this claim and further verify that by incorporating sparsity assumption of O-flow in transform domain referring to the algorithm in 3.3.3, the performance in unidirectional networks can been highly improved.

3.4.1 Simulation Setting

For the proof of concept, four networks are involved in the numerical test. Basic information about these four networks are listed below.

The 3-by-3 unidirectional network in Figure 3.3: 12 links, 27 OD pairs, and 8 origin nodes.

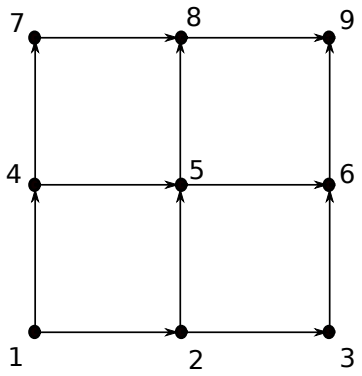


Figure 3.3: The 3-by-3 unidirectional network

The 3-by-3 bidirectional network in Figure 3.4: 24 links, 72 OD pairs, and 9 origin nodes.

The 8-by-8 bidirectional network in Figure 3.5: 224 links, 1660 OD pairs, and 64 origin nodes.

The GEANT network [117] in Figure 3.6: 74 links, 490 OD pairs, and 23 origin nodes.

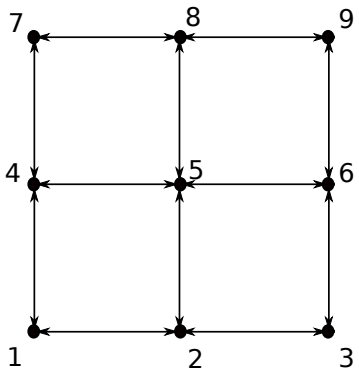


Figure 3.4: The 3-by-3 bidirectional network.

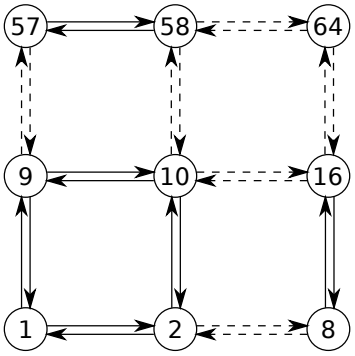


Figure 3.5: The 8-by-8 bidirectional network.

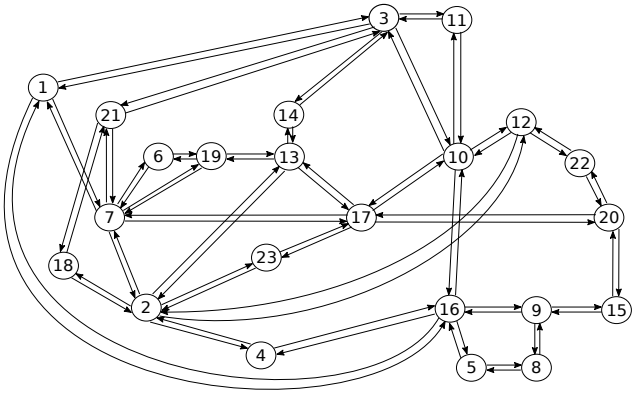


Figure 3.6: The GEANT network.

The ground truth data are generated as follows.

- Firstly, we find out all paths and OD pairs in the study network: For a given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$, we find all the loop-free paths that involve at most $\tau_{\max}$ links and denote the corresponding list by $\mathcal{L}_{\text{path}}$. In generating ground truth data, we only consider paths involving at most $\tau_{\max} = 4$ links. Using these paths, a list of OD pairs is generated and denoted by $\mathcal{L}_{\text{OD}}$. Based on $\mathcal{L}_{\text{OD}}$, a list of origin nodes denoted by $\mathcal{L}_{\text{O}}$ is constructed.

- Then, the distributions based on OD pairs and O-flows are generated:

  - For each given origin node $o \in \mathcal{L}_{\text{O}}$, all the OD pairs in $\mathcal{L}_{\text{OD}}$ originating from o are identified and a probability distribution defined on these OD pairs is generated. This distribution decomposes a given O-flow x_o into OD flows s_{od}.

  - For each given OD pair $od \in \mathcal{L}_{\text{OD}}$, all the paths in $\mathcal{L}_{\text{path}}$ starting from the node o and ending at the node d are found and a probability distribution defined on these paths is produced. This distribution assigns a given OD flow s_{od} into path flows $s_{\boldsymbol{p}}^{\text{path}}$. From the above two probability distributions, both the OD flow assignment matrix and the O-flow assignment matrix can be determined (they are just distribution ratios/weights which sum up to 1).

- Finally, we generate all traffic flows: Let n_T be the total number of sampling time intervals. We generate O-flows x_o^t, $t = 2 - \tau_{\max}, 3 - \tau_{\max}, \cdots, n_T$, $\forall o \in \mathcal{L}_{\text{O}}$, according to the sparse model described in Section 3.3.2 where DCT transform is used. Based on the generated O-flows, both the OD flows s_{od}^t, $t = 2 - \tau_{\max}, 3 - \tau_{\max}, \cdots, n_T$, $\forall od \in \mathcal{L}_{\text{OD}}$, and the link flows $y_{i\to j}^t$, $t = 1, 2, \cdots, n_T$, can be found.

After generating the ground truth data, the simulation proceeds as follows. Given the

link flows $\boldsymbol{y}^t$, the AM procedure described in Section 3.3 is used to estimate the O-flows $\{\hat{\boldsymbol{x}}^t\}$, $t = 1, 2, \cdots, n_T$, and the assignment matrices $\{\hat{\boldsymbol{P}}^t\}$, $t = 1, 2, \cdots, \tau_{\max}$.

The stopping criteria to quit the iterations includes the following:

1. The maximum number of iterations for Algorithm 2 is 5000.

2. Define the normalized mean square error by

$$\text{NMSE}_y = \frac{\sum_t \|\hat{\boldsymbol{y}}^t - \boldsymbol{y}^t\|_2^2}{\sum_t \|\boldsymbol{y}^t\|_2^2} = \frac{\sum_t \left\|\hat{\boldsymbol{P}}^t * \hat{\boldsymbol{x}}^t - \boldsymbol{y}^t\right\|_2^2}{\sum_t \|\boldsymbol{y}^t\|_2^2}.$$

   The iteration stops when $\text{NMSE}_y < 10^{-5}$.

3. When the sparsity constraint is included, i.e., the formulation (3.3.9) is concerned, another stopping criteria is the speed of convergence. Specially, we define $\Delta = (\sum_{o=1}^{n_n} \|\boldsymbol{D}\boldsymbol{x}_o\|_1)_{k-1} - (\sum_{o=1}^{n_n} \|\boldsymbol{D}\boldsymbol{x}_o\|_1)_k$, which indicates the decrease of the objective function at the kth iteration. The iteration stops when $\Delta < 10^{-5}$.

After obtaining the estimated $\{\hat{\boldsymbol{x}}^t\}$ and $\{\hat{\boldsymbol{P}}^t\}$, we compute the estimated $\{\hat{\boldsymbol{s}}^t\}$, $t = 1, 2, \cdots, n_T$, via Equations (3.2.3) and (3.2.4) for single and multiple step models, respectively. The estimation performance using the relative error in the estimated OD flows is defined by

$$E_s^R = \frac{\hat{s}_{od}^t - s_{od}^t}{s_{od}^t}, \tag{3.4.1}$$

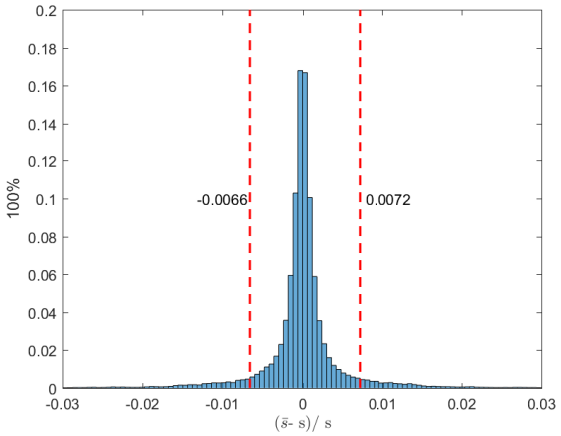
where $od \in \mathcal{L}_{\text{OD}}$ and $t \in \{1, \cdots, n_T\}$. The performance measure is for all valid OD pairs od and time t. All the simulation results are based on 100 trails.

3.4.2 Blind Estimation is Feasible

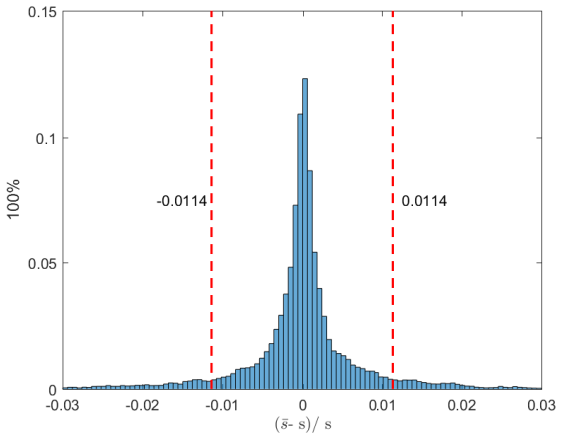
The blind estimation for OD flows has been successfully achieved on three bidirectional networks with multiple step models. The histograms of relative errors of OD flows based

on multiple step models in three bidirectional networks 3.4, 3.5 and 3.6 are given in Figure 3.7 successively. We set the time scale $n_T = 60$, i.e., $t = 1, \cdots, 60$, for the 3-by-3 bidirectional network, and then increase it to 150 for the 8-by-8 and GEANT networks referring to the necessary condition discussed in section 3.3.2.

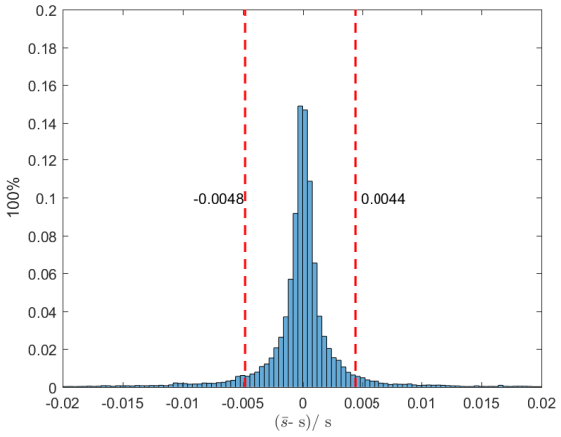
In the 3-by-3 bidirectional network , we can clearly see in Figure 3.7(a) that, majority of the estimated OD flows ($\geq 95\%$) have $E_s^R < 1\%$. When considering a typically large-scale network as Figure 3.7(b) shows, it has up to 1660 OD pairs which is a difficult scenario. The results shows that more than 95% OD flows are estimated with $E_s^R < 1.1\%$. The large-scale of OD flows does not limit the performance of blind estimation. The simulation results in Figure 3.7(c) reflects the performance of blind estimation on a realitic network, the GEANT network. In this mid-scale network, 95% OD flows have $E_s^R < 0.5\%$. The mean E_s^R in these three cases are all less than 10^{-3}. Note that, in these particular cases there are absolutely no prior information available for either the assignment $\boldsymbol{P}^t$ or the flows $\boldsymbol{s}^t$ and that the blind estimation process does not use sparsity information.



(a) Result of 3-by-3 bidirectional network



(b) Result of 8-by-8 bidirectional network



(c) Result of GEANT network

Figure 3.7: Histograms of relative error in OD flow estimation for multiple step model.

3.4.3 Sparsity Improves the Estimation Capacity

In this subsection, we show that incorporating sparsity assumption can improve the performance of blind estimation for unidirectional networks. As predicted in Example 1, the solutions of the 3-by-3 unidirectional network in Figure 3.3 are not unique. The results in Figure 3.8 verifies this claim: there is a non-negligible fraction of estimated OD flows of which the relative errors are about 100% or larger (between 90% and 150%), and the mean error is around 53%. In Section 3.3.2, we argue that sparsity assumption can help in enforcing the uniqueness of the solution. For the purpose of demonstration, we use the iterative procedure described in Section 3.3.3. With the ℓ_1-term involved to promote sparse solutions, the results in Figure 3.9 are largely improved: the majority of the OD flows($\geq 95\%$) now have $E_s^R < 10\%$ with average error 3%.

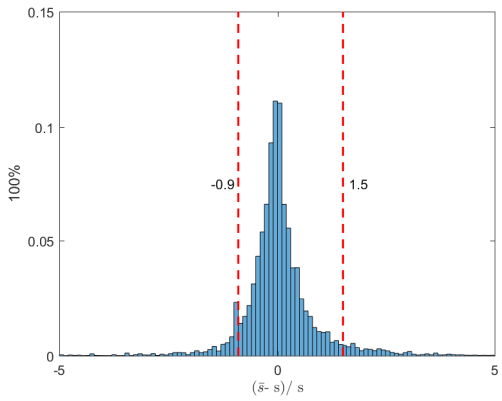


Figure 3.8: Estimation without using sparsity.

3.5 Summary

In this chapter, we demonstrated that OD flow estimation is possible even without any prior information. The technical crux is a new forward model based on O-flows and the associated substantial reduction in dimensionality. For the sake of simplicity, we restricted ourselves to some ideal assumptions as mentioned at the end of the introduction section. The proposed approach can be largely enriched if extensions and

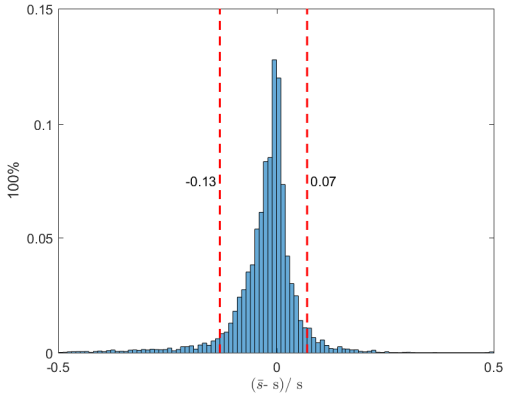


Figure 3.9: Estimation using sparsity

evaluations are made for more realistic scenarios in the future. We have discussed the uniqueness of the solutions in this chapter. A more detailed and deep understanding on this can directly impact on the process of data collection and modeling. From algorithmic development point of view, some guidelines for choosing a good formula for the objective function can be very useful in avoiding complicated parameter tuning, and speedups of the algorithm will allow the application of the proposed approach to large networks.

At the same time, the dramatic dimension reduction on OD estimation problem is essentially achieved via reformulating the original problem as a bi-linear inverse problem. Though the new model is non-convex, it allows a set of extra structural constraints to be composed with the objective function, which lead to a benign landscape thus uniqueness of solution is possible. The heuristic AM algorithm, referring to Gauss-Seidel method, is applied and performs well. Meanwhile, the performance of Gauss-Seidel method get worse when the structural constraints are not sufficient (in uni-directional networks). In this instance, sparsity constraints are incorporated to the objective functions to guarantee the solution accuracy, which alleviates the non-convexity of the objective function and improves the performance dramatically.

In summary, we infer that our proposed new model for OD estimation problem is an application of taking non-convex optimization in solving practical problem. Modeling a

practical problem in non-convex formulation is able to bring specific benefits including dimension reduction for large dimensional problem, flexibility and freedom to append structural and sparsity constraints for better landscapes, and less requirements on prior information. Although the convergence is not guaranteed, the proposed non-convex optimization is solved by AM method with great performance.

Chapter 4

Localized Incomplete Multiple Kernel Clustering

4.1 Introduction

In this chapter, we propose a LI-MKKM-MR to improve the MKC problems discussed in Section 2.2.2. LI-MKKM-MR inherits the advantage of localized incomplete multiple kernel k-means (LI-MKKM), which only requires the similarity of each sample to its top k-nearest neighbours. Besides, LI-MKKM-MR achieves a better utilization of imputation and clustering procedures by allowing the capacity of selecting two kernel matrices with large diversity within the imputation, and thus improves the initialization states of matrices in clustering procedure.

The main technical crux of this work is a matrix-induced regularization term which integrates the redundancy and diversity of kernel matrix into the optimization objective. At the same time, an efficient algorithm with proved convergence is developed to solve the new non-convex non-linear optimization problems over three variables. Specifically, the generalization error bound of the proposed clustering algorithm is derived,

which theoretically guarantees its effectiveness. Extensive experiments are conducted on ten multiple kernel learning (MKL) benchmark datasets to evaluate the clustering performance of the proposed algorithm. The experimental results well validate the effectiveness of incorporating the matrix-induced regularization.

The main contributions of this chapter are summarized as follows.

- We propose a new model to improve LI-MKKM by incorporating a matrix-induced regularization to select low-redundant and high-diverse kernel matrices, and thus design an three-steps algorithm to iteratively solve the optimization problem.

- We present a theoretical connection between global and local kernel alignment criteria. Based on this connection, we derive the generalization error bound of the proposed LI-MKKM-MR method.

- We conduct comprehensive experimental studies on ten benchmark datasets, and the proposed LI-MKKM-MR significantly and consistently outperforms the state-of-the-art methods.

The rest of this chapter is organized as follows. Section 4.2 demonstrates the formulation and optimization algorithm of our proposed LI-MKKM-MR method. In Section 4.2.3, we analyze theoretical connection between existing MKKM-IK and LI-MKKM-MR, thus then derive the generalization error bound of the proposed LI-MKKM-MR based on the theoretical results. Section 4.3 presents simulation results on ten widely used MKL benchmark data sets and Section 4.4 draws conclusions.

4.2 The Proposed LI-MKKM-MR

4.2.1 Formulations

According to the analysis in [88], we have inferred that the maximization of aligning the optimal kernel with the average similarity would lead to high redundancy and low diversity among the selected kernels. To address this issue, we incorporate a matrix-induced regularization $\boldsymbol{\gamma}^\top \boldsymbol{M}\boldsymbol{\gamma}$ to reduce the redundancy and enforce the diversity of the selected base kernels on LI-MKKM. We adopt $M_{pq} = \mathrm{Tr}(\boldsymbol{K}_p\boldsymbol{K}_q)$ to measure the correlation between $\boldsymbol{K}_p$ and $\boldsymbol{K}_q$, where M_{pq} is the entry of matrix $\boldsymbol{M}$. With respect to the notations and analysis in Section 2.2.2 for LI-MKKM problem (2.2.16), we present our new model in the following form

$$\begin{aligned} \min_{\boldsymbol{\gamma},\{\boldsymbol{K}_p\}_{p=1}^m,\boldsymbol{H}} & \sum_{i=1}^{n} \mathrm{Tr}\left(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{B}^{(i)})\right) + \frac{\lambda}{2}\boldsymbol{\gamma}^\top\boldsymbol{M}\boldsymbol{\gamma} \\ s.t.\ & \boldsymbol{H} \in \mathbb{R}^{n\times k},\ \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k, \\ & \boldsymbol{\gamma}^\top\mathbf{1}_m = 1,\ \gamma_p \geq 0, \\ & \boldsymbol{K}_p(\boldsymbol{s}_p,\boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0,\ \forall p, \end{aligned} \tag{4.2.1}$$

where λ is a hyper-parameter to trade-off the loss of local kernel k-means and the regularization on kernel weights, $\mathbf{B}^{(i)}$ is the neighborhood index matrix of the i-th samples. The incorporation of $\boldsymbol{\gamma}^\top\boldsymbol{M}\boldsymbol{\gamma}$ makes the resultant optimization become more challenging since the optimization on each $\boldsymbol{K}_p$ is a quadratic semi-definite programming, whose computational cost is intensive. The new regularization term leads to significant increase in computation complexity, therefore being applied to large scale applications will be computational resource consuming. Therefore, we further optimize the objective function with fixed M_{pq} over the learning courses and let it be estimated through

$$\tilde{M}_{pq} = \mathrm{Tr}(\boldsymbol{K}_p^{(0)}\boldsymbol{K}_q^{(0)}), \tag{4.2.2}$$

where $\boldsymbol{K}_p^{(0)}$ and $\boldsymbol{K}_q^{(0)}$ are initial imputation of $\boldsymbol{K}_p$ and $\boldsymbol{K}_q$ respectively. By substituting $\boldsymbol{M}$ with $\tilde{\boldsymbol{M}}$, the objective function of the proposed LI-MKKM-MR can be reformulated as follows,

$$\begin{aligned} \min_{\boldsymbol{\gamma},\{\boldsymbol{K}_p\}_{p=1}^m,\boldsymbol{H}} & \sum_{i=1}^{n} \mathrm{Tr}(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{B}^{(i)})) + \frac{\lambda}{2}\boldsymbol{\gamma}^\top\tilde{\boldsymbol{M}}\boldsymbol{\gamma} \\ s.t. \ & \boldsymbol{H} \in \mathbb{R}^{n\times k},\ \boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k, \\ & \boldsymbol{\gamma}^\top\mathbf{1}_m = 1,\ \gamma_p \geq 0, \\ & \boldsymbol{K}_p(\boldsymbol{s}_p,\boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0,\ \forall p. \end{aligned} \tag{4.2.3}$$

In this instance, the proposed method aims to evaluate the correlation of pairwise kernels via observed similarity. The approximation $\tilde{\boldsymbol{M}}$ can be regarded as a prior of $\boldsymbol{M}$. This approximation provides significant benefits in two aspects: Firstly, it satisfies the demands of kernel coefficients to reduce the redundancy and enhance the diversity of kernel matrices. Secondly, a closed-form solution of $\{\boldsymbol{K}_p\}_{p=1}^m$ is guaranteed and significant computation reductions is achieved thereof. More detailed demonstrations will be soon presented in Algorithm 3. By incorporating the matrix-induced regularization, the proposed model essentially forms as a non-convex non-linear optimization problem over three variables. Despite the number of unknowns and the non-convexity of the objective are significantly increased, the new model allows more structural constraints to be applied to improve the optimization landscape and more feasibility of utilization of the incomplete kernels (e.x., the approximation $\tilde{\boldsymbol{M}}$).

4.2.2 Three-step Alternative Algorithm for LI-MKKM-MR

We propose a three-step iterative optimization algorithm in Algorithm 3 to solve the proposed LI-MKKM-MR. The absent elements of $\{\boldsymbol{K}_p^{(0)}\}_{p=1}^m$ are initially imputed with zeros. $\mathrm{obj}^{(t)}$ denotes the objective value at the t-th iteration. It can be seen that the neighborhood matrix $\boldsymbol{S}_i$ of the i-th sample keeps fixed and meatured by the $\boldsymbol{K}_{\boldsymbol{\gamma}^{(0)}}$

during the optimization in LI-MKKM-MR. Specifically, the τ-nearest neighbors of each sample are measured by $\boldsymbol{K}_{\boldsymbol{\gamma}^{(0)}}$. The proposed algorithm is lower-bounded by zero and the derivation will be presented soon. The computational complexity of LI-MKKM-MR is $\mathcal{O}(n^3 + \sum_{p=1}^{m} n_p^3 + m^3)$ per iteration, where $n_p\,(n_p \leq n)$ denotes the number of observed samples of $\boldsymbol{K}_p$ and m denotes the number of observed samples of base kernel, respectively. Despite the proposed model increases the number of unknown variables, the computation complexity keeps the same with the existing methods MKKM-IK [102] and LI-MKKM [103].

Algorithm 3: The Proposed LI-MKKM-MR

Input: $\{\mathbf{K}_p^{(cc)}\}_{p=1}^m$, $\{\mathbf{s}_p\}_{p=1}^m, k, \tau, \lambda$ and ϵ_0.
Output: $\mathbf{H}$, $\boldsymbol{\gamma}$ and $\{\mathbf{K}_p\}_{p=1}^m$.
Initialize: $\boldsymbol{\gamma}^{(0)} = \mathbf{1}_m/m$, $\{\mathbf{K}_p^{(0)}\}_{p=1}^m$ and $t = 1$.
Generating: $\mathbf{S}^{(i)}$ for i-th samples $(1 \leq i \leq n)$ by $\mathbf{K}_{\boldsymbol{\gamma}^{(0)}}$.
while $\left(obj^{(t-1)} - obj^{(t)}\right)/obj^{(t)} \leq \epsilon_0$ **do**
  $\mathbf{K}_{\boldsymbol{\gamma}^{(t)}} = \sum_{p=1}^{m} \left(\gamma_p^{(t-1)}\right)^2 \mathbf{K}_p^{(t-1)}$.
  Update $\mathbf{H}^{(t)}$ by solving Eq. (4.2.4) with $\mathbf{K}_{\boldsymbol{\gamma}^{(t)}}$.
  Update $\{\mathbf{K}_p^{(t)}\}_{p=1}^m$ with $\mathbf{H}^{(t)}$ by Eq. (4.2.6).
  Update $\boldsymbol{\gamma}^{(t)}$ by solving Eq.(4.2.11) with $\mathbf{H}^{(t)}$ and $\{\mathbf{K}_p^{(t)}\}_{p=1}^m$.
  $t = t + 1$.
end

In our three-step algorithm, we optimize each variable with respect to the rest are fixed. We describe every step of the proposed algorithm as follows.

- Optimizing $\boldsymbol{H}$ with $\boldsymbol{\gamma}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$ fixed. Given $\boldsymbol{\gamma}$ and $\{\boldsymbol{K}_p\}_{p=1}^m$, the optimization w.r.t $\boldsymbol{H}$ in problem (4.2.3) reduces to

$$\begin{aligned} \max_{\boldsymbol{H}} \ & \operatorname{Tr}\left(\boldsymbol{H}^\top \sum_{i=1}^{n} (\boldsymbol{B}^{(i)} \boldsymbol{K}_{\boldsymbol{\gamma}} \boldsymbol{B}^{(i)}) \boldsymbol{H}\right) \\ s.t. \ & \boldsymbol{H} \in \mathbb{R}^{n \times k}, \ \boldsymbol{H}^\top \boldsymbol{H} = \boldsymbol{I}_k, \end{aligned} \tag{4.2.4}$$

  which is a conventional kernel k-means optimization problem and can be readily solved by existing off-the-shelf packages directly.

- Optimizing $\{\boldsymbol{K}_p\}_{p=1}^m$ with $\boldsymbol{\gamma}$ and $\boldsymbol{H}$ fixed. Given $\boldsymbol{\gamma}$ and $\boldsymbol{H}$, the optimization w.r.t $\{\boldsymbol{K}_p\}_{p=1}^m$ in problem (4.2.3) can be expressed as

$$\begin{aligned} \min_{\{\boldsymbol{K}_p\}_{p=1}^m} \quad & \sum_{p=1}^{m} \gamma_p^2 \mathrm{Tr}\left(\boldsymbol{K}_p \sum_{i=1}^{n} \mathrm{Tr}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top \boldsymbol{B}^{(i)})\right) \\ s.t. \quad & \boldsymbol{K}_p(\boldsymbol{s}_p, \boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0,\ \forall p. \end{aligned} \tag{4.2.5}$$

Problem (4.2.5) is intractable in direct solutions due to the multiple unknowns up to multiple kernel matrices. In this stage, we infer that: i) each kernel matrix $\boldsymbol{K}_p$ has its own separate constraint; ii) the objective function in problem (4.2.5) is a sum over $\boldsymbol{K}_p$. Therefore, it can be rewritten as $m = 1, 2, \ldots, p$ number of independent sub-problems as follows,

$$\begin{aligned} \min_{\boldsymbol{K}_p} \quad & \mathrm{Tr}(\boldsymbol{K}_p\boldsymbol{Q}) \\ s.t. \quad & \boldsymbol{K}_p(\boldsymbol{s}_p, \boldsymbol{s}_p) = \boldsymbol{K}_p^{(cc)},\ \boldsymbol{K}_p \succeq 0, \end{aligned} \tag{4.2.6}$$

where $\boldsymbol{Q} = \sum_{i=1}^{n}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{B}^{(i)})$. Referring to the derivation in [103], we parameterize each $\boldsymbol{K}_p$ with respect to their equality and PSD constraints as follows,

$$\boldsymbol{K}_p = \begin{bmatrix} \boldsymbol{K}_p^{(cc)} & \boldsymbol{K}_p^{(cc)}\boldsymbol{W}_p \\ \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)} & \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)}\boldsymbol{W}_p \end{bmatrix}, \tag{4.2.7}$$

where $\boldsymbol{W}_p \in \mathbb{R}^{c\times m}$, c and m denote the number of observed and unobserved samples, respectively. In this way, the missing kernel entries are recomposed by the observed elements. Referring to the works in [103], $\boldsymbol{K}_p$ in Equation (4.2.7) still satisfies all the constraints after this parameterization.

Theorem 1 *$\boldsymbol{K}_p$ in Equation (4.2.7) is a feasible set of the optimization problem (4.2.6).*

The proof is given as follows.

Proof 1 *For any vector* $\boldsymbol{x} \in \mathbb{R}^n$*, it has* $\boldsymbol{x} = [\boldsymbol{x}_c^\top, \boldsymbol{x}_m^\top]^\top$*. We have*

$$\begin{aligned}\boldsymbol{x}^\top \boldsymbol{K}_p \boldsymbol{x} =& [\boldsymbol{x}_c^\top, \boldsymbol{x}_m^\top] \begin{bmatrix} \boldsymbol{K}_p^{(cc)} & \boldsymbol{K}_p^{(cc)} \boldsymbol{W}_p \\ \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)} & \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)} \boldsymbol{W}_p \end{bmatrix} [\boldsymbol{x}_c^\top, \boldsymbol{x}_m^\top]^\top \\ =& (\boldsymbol{x}_c + \boldsymbol{W}_p \boldsymbol{x}_m)^\top \boldsymbol{K}_p^{(cc)} (\boldsymbol{x}_c + \boldsymbol{W}_p \boldsymbol{x}_m) \geq 0.\end{aligned} \tag{4.2.8}$$

This verifies that the parameterization of $\boldsymbol{K}_p$ *is PSD and* $\boldsymbol{K}_p$ *satisfies all the constraint. Therefore,* $\boldsymbol{K}_p$ *in Equation (4.2.7) is feasible and this completes the proof.*

Based on the parameterization in Equation (4.2.7) , the constrained optimization in problem (4.2.6) can be converted into the following non-constrained problem,

$$\min_{\boldsymbol{W}_p} \operatorname{Tr} \left(\begin{bmatrix} \boldsymbol{K}_p^{(cc)} & \boldsymbol{K}_p^{(cc)} \boldsymbol{W}_p \\ \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)} & \boldsymbol{W}_p^\top \boldsymbol{K}_p^{(cc)} \boldsymbol{W}_p \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}^{(cc)} & \boldsymbol{Q}^{(cm)} \\ \boldsymbol{Q}^{(cm)\top} & \boldsymbol{Q}^{(mm)} \end{bmatrix} \right), \tag{4.2.9}$$

where $\boldsymbol{Q}$ is decomposed into the sub-matrices $\begin{bmatrix} \boldsymbol{Q}^{(cc)} & \boldsymbol{Q}^{(cm)} \\ \boldsymbol{Q}^{(cm)\top} & \boldsymbol{Q}^{(mm)} \end{bmatrix}$. In this stage, the objective is to minimize Equation (4.2.9). The first-order stationary point with respect to $\boldsymbol{W}_p$ is thus given by

$$\boldsymbol{W}_p = -\boldsymbol{Q}^{(cm)} (\boldsymbol{Q}^{(mm)})^{-1}. \tag{4.2.10}$$

Consequently, we replace the analytical solution for the optimal $\boldsymbol{K}_p$ in Equation (4.2.7) with the optimal solution of $\boldsymbol{W}_p$ in Equation (4.2.10). We infer that problem (4.2.6) exploits extra local structure of data to guide the imputation process of each base kernel. Specifically, it computes the alignment between the similarity of each sample and its τ-nearest neighbors with corresponding ideal matrix in a local manner. This enables the proposed algorithm to well utilize the intra-cluster variations among samples. In this way, the incomplete kernels could be more effectively imputed with better clustering performance.

- Optimizing $\boldsymbol{\gamma}$ with $\{\boldsymbol{K}_p\}_{p=1}^m$ and $\boldsymbol{H}$ fixed. Given $\{\boldsymbol{K}_p\}_{p=1}^m$ and $\boldsymbol{H}$, it is straight-

forward to show that problem (4.2.3) w.r.t. $\boldsymbol{\gamma}$ can be reformulated as follows,

$$\begin{aligned} \min_{\boldsymbol{\gamma}} \ & \frac{1}{2}\boldsymbol{\gamma}^\top \left(2\boldsymbol{Z} + \lambda\tilde{\boldsymbol{M}}\right)\boldsymbol{\gamma} \\ s.t. \ & \boldsymbol{\gamma}^\top \mathbf{1}_m = 1, \gamma_p \geq 0, \end{aligned} \tag{4.2.11}$$

where $\boldsymbol{Z} = \mathrm{diag}([\mathrm{Tr}(\boldsymbol{K}_1\boldsymbol{Q}), \cdots, \mathrm{Tr}(\boldsymbol{K}_m\boldsymbol{Q})])$. The following Proof 2 illustrates that $\boldsymbol{Z}$ and $2\boldsymbol{Z} + \lambda\tilde{\boldsymbol{M}}$ are symmetric PSD matrices.

Proof 2 *By denoting* $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_k]$, *we can see that* $\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{h}_c = \boldsymbol{h}_c(1 \leq c \leq k)$ *since* $\boldsymbol{H}^\top\boldsymbol{H} = \boldsymbol{I}_k$. *This indicates that* $\boldsymbol{H}\boldsymbol{H}^\top$ *has* k *eigenvalue with* 1. *Also, its rank is no greater than* k. *This means that its has* $n-k$ *eigenvalue with* 0. *Correspondingly,* $\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top$ *has* $n-k$ *and* k *eigenvalue with* 1 *and* 0. *As a result,* $\boldsymbol{B}^{(i)}(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top)\boldsymbol{B}^{(i)}$ *is PSD, and this guarantees that* $\boldsymbol{Q} = \sum_{i=1}^{n}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{B}^{(i)})$ *is PSD. Therefore, we have* $z_p = \mathrm{Tr}(\boldsymbol{K}_p\boldsymbol{Q}) \geq 0, \forall p$, *guaranteeing the positiveness of* $\boldsymbol{Z}$. *Meanwhile,* $\boldsymbol{Z}$ *is also a symmetric PSD matrix according to [118]. Consequently,* $2\boldsymbol{Z} + \lambda\tilde{\boldsymbol{M}}$ *is a symmetric PSD matrix. This completes the proof.*

According to the above proof, it is guaranteed that the optimization in problem (4.2.11) w.r.t $\boldsymbol{\gamma}$ is a quadratic programming (QP) with linear constraints. Thus it can be directly solve by mathematical toolboxes for QP problem.

4.2.3 Theoretical Analysis

In this section, we first illustrate the connections between the existing MKKM-IK [119] method with the proposed LI-MKKM-MR. Then, with respect to the analysis on MKKM-IK, we derive the generalization error bound of the proposed LI-MKKM-MR. The generalization error of k-means clustering algorithm measures the clustering performance of the learned clusters in training procedure on unseen samples [119–121].

We start with the following Theorem 2 which states that the local kernel alignment adopted in our LI-MKKM-MR. It can be fulfilled by normalizing each base kernel matrix.

Theorem 2 *The local kernel alignment criterion in the LI-MKKM problem (2.2.16) keeps equivalent to the global kernel alignment that normalizes each base kernel matrix.*

Proof 3 *The objective function of problem (2.2.16) can be written as*

$$\begin{aligned}
&\sum_{i=1}^{n} \mathrm{Tr}\left(\boldsymbol{K}_{\boldsymbol{\gamma}}(\boldsymbol{B}^{(i)} - \boldsymbol{B}^{(i)}\boldsymbol{H}\boldsymbol{H}^{\top}\boldsymbol{B}^{(i)})\right) \\
&= \sum_{i=1}^{n} \langle \boldsymbol{B}^{(i)} \circ \boldsymbol{K}_{\boldsymbol{\gamma}}, \boldsymbol{B}^{(i)} \circ (\boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top}) \rangle_{\mathrm{F}} \\
&= \sum_{i=1}^{n} \langle \boldsymbol{B}^{(i)} \circ \boldsymbol{K}_{\boldsymbol{\gamma}}, \boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top} \rangle_{\mathrm{F}} \\
&= \langle \left(\sum_{i=1}^{n} \boldsymbol{B}^{(i)} \right) \circ \boldsymbol{K}_{\boldsymbol{\gamma}}, \boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top} \rangle_{\mathrm{F}} \\
&= \sum_{p=1}^{m} \gamma_p^2 \langle \left(\sum_{i=1}^{n} \boldsymbol{B}^{(i)} \right) \circ \boldsymbol{K}_p, \boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top} \rangle_{\mathrm{F}} \\
&= \sum_{p=1}^{m} \gamma_p^2 \langle \tilde{\boldsymbol{K}}_p, \boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top} \rangle_{\mathrm{F}} \\
&= \mathrm{Tr}\left(\tilde{\boldsymbol{K}}_{\boldsymbol{\gamma}} (\boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}^{\top}) \right),
\end{aligned} \tag{4.2.12}$$

where $\circ$ denotes Hadamard product (also known as element-wise product), $\tilde{\boldsymbol{K}}_{\boldsymbol{\gamma}} = \sum_{p=1}^{m} \gamma_p^2 \tilde{\boldsymbol{K}}_p$, and $\tilde{\boldsymbol{K}}_p = \left(\sum_{i=1}^{n} \boldsymbol{B}^{(i)}\right) \circ \boldsymbol{K}_p$ is the normalization of $\boldsymbol{K}_p$. Consequently, by applying above normalization on each base kernel, it can be concluded that the local kernel alignment criterion in problem (2.2.16) equals the global kernel alignment. This completes the proof.

Let $t(\boldsymbol{x}^{(p)}) = 1$ when the p-th view of $\boldsymbol{x}$ is available, and $\hat{\boldsymbol{C}} = [\hat{\boldsymbol{C}}_1, \cdots, \hat{\boldsymbol{C}}_k]$ denotes the k centroids, and $\hat{\boldsymbol{\beta}}$ denotes the vector of kernel weights learned by LI-MKKM-MR. Note that $t(\boldsymbol{x}^{(p)})$ is a random variable which depends on $\boldsymbol{x}$. Then, the k-means clustering

aims to minimize the following reconstruction error

$$\mathbb{E}\left[\min_{\boldsymbol{y}\in\{\boldsymbol{e}_1,\cdots,\boldsymbol{e}_k\}}\left\|\phi_{\hat{\boldsymbol{\gamma}}}(\boldsymbol{x})-\hat{\boldsymbol{C}}\boldsymbol{y}\right\|_{\mathcal{H}}^2\right], \tag{4.2.13}$$

where $\phi_{\hat{\boldsymbol{\gamma}}}(\boldsymbol{x})=[\hat{\gamma}_1 t(\boldsymbol{x}^{(1)})\phi_1^\top(\boldsymbol{x}^{(1)}),\cdots,\hat{\gamma}_m t(\boldsymbol{x}^{(m)})\phi_m^\top(\boldsymbol{x}^{(m)})]^\top$, and $\boldsymbol{e}_1,\cdots,\boldsymbol{e}_k$ denote the orthogonal bases of $\mathbb{R}^k$.

We firstly define a function class:

$$\begin{aligned}\mathcal{F}=\Big\{f:\boldsymbol{x}\mapsto \min_{\boldsymbol{y}\in\{\boldsymbol{e}_1,\ldots,\boldsymbol{e}_k\}}\|\phi_{\boldsymbol{\gamma}}(\boldsymbol{x})-\boldsymbol{C}\boldsymbol{y}\|_{\mathcal{H}}^2\,\Big|\,\boldsymbol{\gamma}^\top\mathbf{1}_m=1,\,\gamma_p\geq 0,\\ \boldsymbol{C}\in\mathcal{H}^k,\,t(\boldsymbol{x}_i^{(p)})t(\boldsymbol{x}_j^{(p)})\tilde{\kappa}_p^\top(\boldsymbol{x}_i^{(p)},\boldsymbol{x}_j^{(p)})\leq b,\,\forall p,\forall\boldsymbol{x}_i\in\mathcal{X}\Big\},\end{aligned} \tag{4.2.14}$$

where $\mathcal{H}^k$ represents the multiple kernel Hilbert space and $\tilde{\kappa}(\cdot,\cdot)$ is a kernel function corresponding to $\tilde{\boldsymbol{K}}_p$. Then, we derive the generalization error bound of the proposed LI-MKKM-MR as follows.

Theorem 3 *For any $\delta>0$, with probability at least $1-\delta$, the following inequality holds for all $f\in\mathcal{F}$:*

$$\mathbb{E}[f(\boldsymbol{x})]\leq\frac{1}{n}\sum_{i=1}^n f(\boldsymbol{x}_i)+\frac{4\sqrt{\pi}mb\mathcal{G}_{1n}(\boldsymbol{\gamma},t)}{n}+\frac{4\sqrt{\pi m}b\mathcal{G}_{2n}(\boldsymbol{\gamma},t)}{n}+\frac{\sqrt{8\pi}bk^2}{\sqrt{n}}+2b\sqrt{\frac{\log 1/\delta}{2n}}, \tag{4.2.15}$$

where

$$\mathcal{G}_{1n}(\boldsymbol{\gamma},t)\triangleq\mathbb{E}_\beta\left[\sup_{\boldsymbol{\gamma},t}\sum_{i=1}^n\sum_{p,q=1}^m\beta_{ipq}t(\boldsymbol{x}_i^{(p)})t(\boldsymbol{x}_i^{(q)})\gamma_p\gamma_q\right], \tag{4.2.16}$$

$$\mathcal{G}_{2n}(\boldsymbol{\gamma},t)=\mathbb{E}_\beta\left[\sup_{\boldsymbol{\gamma},t}\sum_{i=1}^n\sum_{c=1}^k\sum_{p=1}^m\beta_{icp}\gamma_p t(\boldsymbol{x}_i^{(p)})\right], \tag{4.2.17}$$

and $\beta_{ipq},\beta_{icp},i\in\{1,\ldots,n\},p,q\in\{1,\ldots,m\},c\in\{1,\ldots,k\}$ are i.i.d. Gaussian random variables with zero mean and unit standard deviation.

Referring to the derivation in [119], our local kernel alignment criterion $\mathrm{Tr}(\tilde{\boldsymbol{K}}_{\boldsymbol{\beta}}(\boldsymbol{I}_n-\boldsymbol{H}\boldsymbol{H}^\top))$ in problem (2.2.16) is an upper bound of $\frac{1}{n}\sum_{i=1}^n f(\boldsymbol{x}_i)$. Therefore, better gen-

eralization can be guaranteed by minimizing $\mathrm{Tr}(\tilde{\boldsymbol{K}}_{\boldsymbol{\beta}}(\boldsymbol{I}_n - \boldsymbol{H}\boldsymbol{H}^\top))$. This thus justifies the good generalization ability of the proposed algorithm.

4.3 Simulation Results

4.3.1 Experimental Settings

In this part, ten widely used MKL benchmark datasets have been applied to evaluate the proposed algorithms, including Oxford Flower17 and Flower102[1], Caltech102[2], Digital[3] and Protein Fold Prediction[4] The details of these datasets are shown in Table 4.1. The kernel matrices of these datasets are pre-computed. All of them can be directly downloaded from the links in footnotes. The public access Matlab codes for kernel k-means and MKKM can be found in the link below [5].

Several well-known and widely used imputation methods [98], including zero filling (ZF), mean filling (MF), k-nearest-neighbor filling (KNN), alignment-maximization filling (AF) are applied to approximate kernel matrices at the first step. After that, these imputed kernel matrices are then adopted as the input of the existing MKKM variations respectively, including MKKM+ZF, MKKM+MF, MKKM+KNN and MKKM+AF. Also, the newly proposed MKKM-IK [102] and LI-MKKM [103], are also incorporated as a strong baseline. The algorithms in [99, 100, 122] are not incorporated into our experimental comparison since these algorithms only consider the absence of input features, instead of the rows/columns of base kernel matrices in our case.

In the experiment, we use ε to denote the percentage of incomplete samples. Intuitively, the clustering performance will become less accurate when the value of ε is increasing.

[1] http://www.robots.ox.ac.uk/~+vgg/data/flowers/,
[2] http://files.is.tue.mpg.de/pgehler/projects/iccv09/
[3] http://ss.sysu.edu.cn/~+py/
[4] http://mkl.ucsd.edu/dataset/protein-fold-prediction/
[5] https://github.com/mehmetgonen/lmkkmeans/

In our simulation, ε is set as $[0.1 : 0.1 : 0.9]$ on all the 10 data sets. The performance criterion in this simulation is composed of the clustering accuracy (ACC), normalized mutual information (NMI) and purity. We define these three criteria of clustering quality as follows:

$$\text{purity} = (\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j|, \tag{4.3.1}$$

where $\Omega = \omega_1, \omega_2, \cdots, \omega_K$ is the clustering results set, $\mathbb{C} = c_1, c_2, \cdots, c_J$ is the original clustering for the dataset $x_{ii} = 1^N$,

$$\text{NMI} = \frac{2 \sum_k \sum_j \frac{|\omega_k \cap c_k|}{N} \log \frac{N|\omega_k \cap c_k|}{|\omega_k||c_j|}}{-\sum_k \frac{|\omega_k|}{N} \log \frac{\omega_k}{N} - \sum_k \frac{|c_j|}{N} \log \frac{c_j}{N}}, \tag{4.3.2}$$

and

$$\text{ACC} = \frac{\sum_{i=1}^{N} \sigma(\omega_i, c_i)}{N}, \tag{4.3.3}$$

where $\sigma(x, y) = 1$ if $x = y$, otherwise, $\sigma(x, y) = 0$.

For each method, we reported the best result among 50 trials, where each trial started from a random initialization state. Consequently, the effect of randomness caused by k-means could be alleviated. The "incomplete" patterns are randomly generated for 10 times and the statistical results are reported.

For all datasets, the number of clusters is given and set as the ground truth of classes. The generation of the missing vectors $\{\boldsymbol{s}_p\}_{p=1}^{m}$ follows the approach in [102]: (1) Randomly select round$(\varepsilon * n)$ samples with the rounding function round$(\cdot)$. (2) Generate a random vector $\boldsymbol{v} = (v_1, \cdots, v_k, \cdots, v_m),\ v_k \in [0, 1]$ and a scalar $v_0,\ v_0 \in [0, 1]$ for each selected sample. (3) If $v_p \geq v_0$, present the p-th view for this sample. (4) If there is no $v_p \geq v_0$, generate a new $\boldsymbol{v}$. Note that there is no requirement on complete view across all the samples. In this instance, the index vector $\boldsymbol{s}_p$ is obtained to list the samples with the presentation on the p-th view.

Table 4.1: Ten applied datasets.

Dataset	#Samples	#Views	#Classes
Flower17	1360	7	17
Flower102	8189	4	102
Caltech102-5	510	48	102
Caltech102-10	1020	48	102
Caltech102-15	1530	48	102
Caltech102-20	2040	48	102
Caltech102-25	2550	48	102
Caltech102-30	3060	48	102
Digital	2000	3	10
ProteinFold	694	12	27

4.3.2 Experimental Results

Table 4.2: Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Flower17 and Flower102 datasets.

Datasets	MKKM+ZF	MKKM+MF	MKKM+KNN	MKKM+AF [98]	MKKM-IK [104]	LI-MKKM [103]	LI-MKKM-MR Proposed
ACC							
Flower17	36.90 ± 0.77	36.75 ± 0.57	37.78 ± 0.61	40.48 ± 0.73	44.63 ± 0.57	47.98 ± 0.36	$\mathbf{55.05 \pm 0.38}$
Flower102	17.98 ± 0.15	18.01 ± 0.17	18.23 ± 0.13	19.22 ± 0.14	21.09 ± 0.16	23.13 ± 0.10	$\mathbf{30.45 \pm 0.28}$
NMI							
Flower17	37.31 ± 0.41	37.28 ± 0.45	38.22 ± 0.48	40.07 ± 0.44	43.67 ± 0.29	46.43 ± 0.17	$\mathbf{52.50 \pm 0.26}$
Flower102	37.38 ± 0.13	37.39 ± 0.14	37.80 ± 0.09	38.38 ± 0.13	39.56 ± 0.12	41.78 ± 0.08	$\mathbf{47.52 \pm 0.13}$
Purity							
Flower17	38.36 ± 0.63	38.30 ± 0.60	39.30 ± 0.57	41.99 ± 0.60	45.89 ± 0.48	48.90 ± 0.36	$\mathbf{56.36 \pm 0.42}$
Flower102	22.45 ± 0.11	22.43 ± 0.12	22.80 ± 0.12	23.73 ± 0.18	25.76 ± 0.18	28.13 ± 0.14	$\mathbf{35.80 \pm 0.27}$

Experiments on Flower17 and Flower102.

Three performance criteria, including the ACC, NMI and purity, of the testing algorithms with the variation of missing ratios in $[0.1, \cdots, 0.9]$ on the Flower17 and Flower102 datasets have been demonstrated in Figure 4.1. We have the following observations.

- The proposed MKKM-IK [104] (in green) has shown dramatic performance improvements on the ACC, NMI and purity comparing to the previous two-step

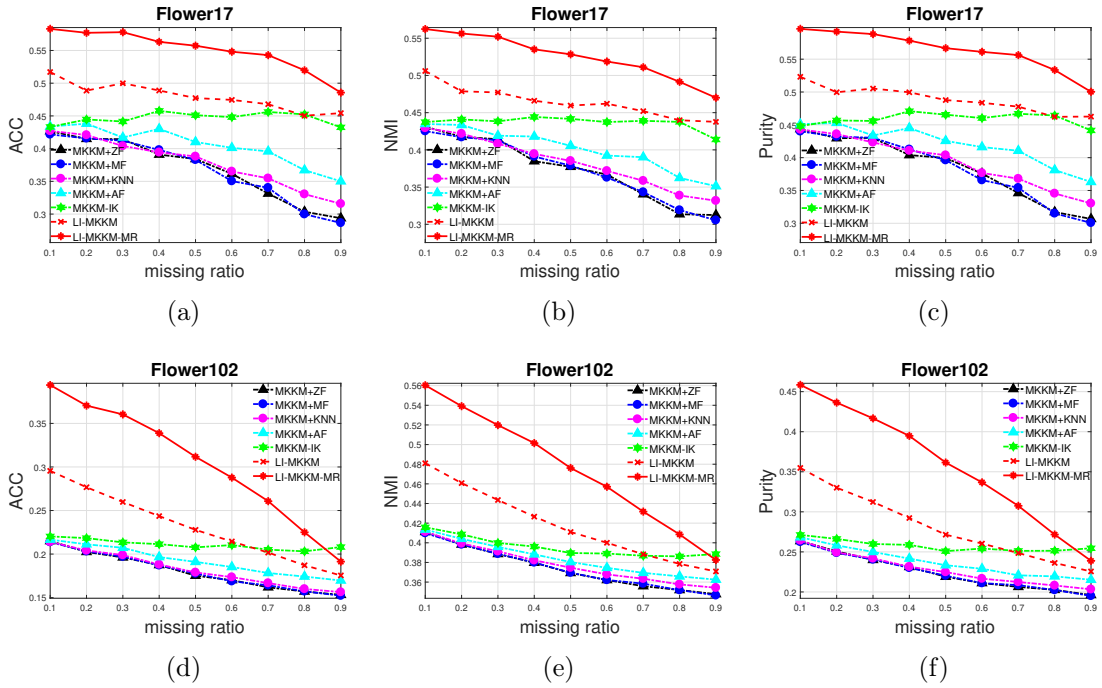


Figure 4.1: Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Flower17 and Flower102 datasets. For each given missing ratio, the ”incomplete patterns" are randomly generated for 10 times and their averaged results are reported.

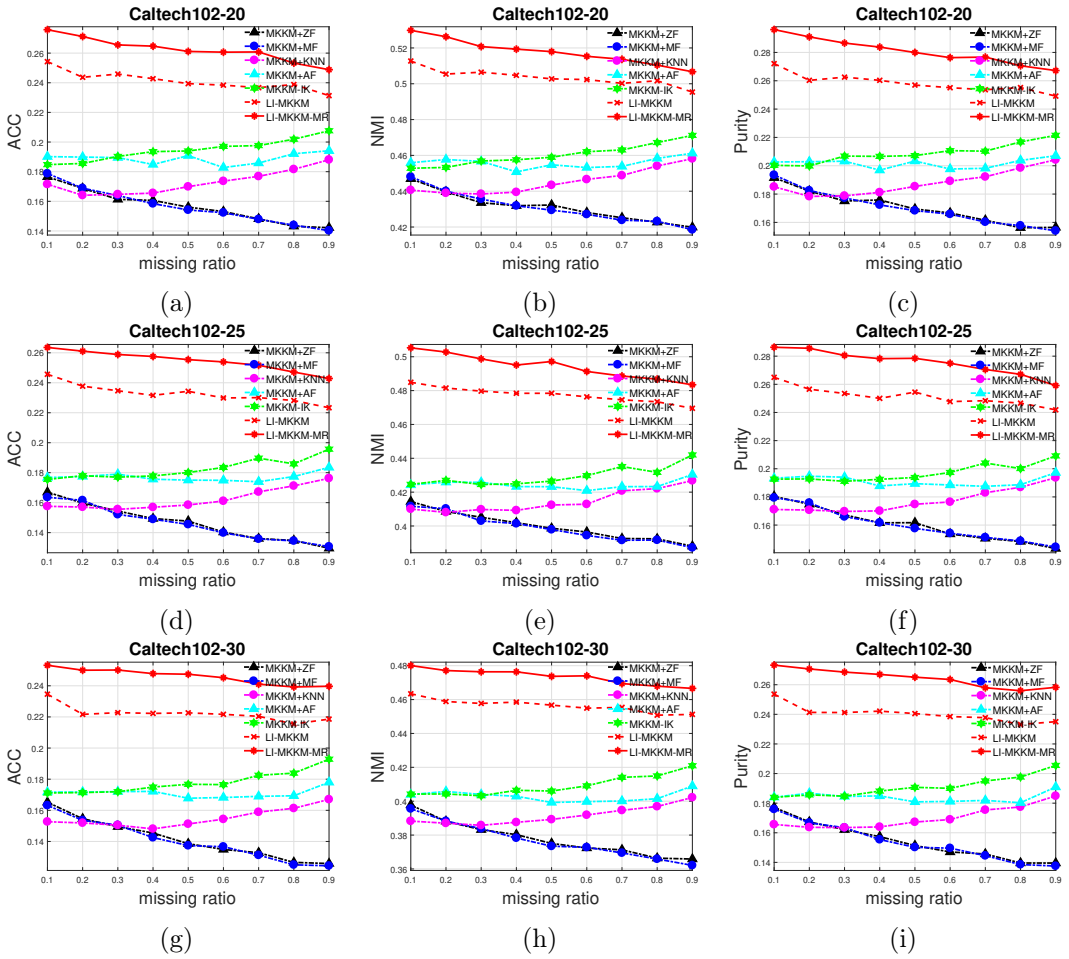


Figure 4.2: Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Caltech102-20, Caltech102-25 and Caltech102-30. For each given missing ratio, the "incomplete patterns" are randomly generated for 10 times and their averaged results are reported. The results on Caltech102-5, Caltech102-10 and Caltech102-15 are provided in the supplemental material due to space limit.

imputation methods. For example, the MKKM+AF outperforms MKKM-IK by $0.1\%, 0.6\%, 2.5\%, 2.8\%, 4.1\%, 4.7\%, 6.0\%, 8.5\%, 8.2\%$ in terms of clustering accuracy on Flower17. These results verify the advantage of the joint optimization on imputation and clustering.

- Similarly, LI-MKKM outperforms MKKM-IK by $8.4\%, 4.4\%, 5.8\%, 3.1\%, 2.6\%, 2.6\%, 1.2\%, 0.2\%, 2.2\%$ on Flower17. This verifies that the utilization of the local structure of data further boosts the clustering performance.

- Nevertheless, our proposed LI-MKKM-MR (in red) significantly outperforms the

Table 4.3: Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Caltech102 data sets.

Datasets	MKKM+ZF	MKKM+MF	MKKM+KNN	MKKM+AF [98]	MKKM-IK [104]	LI-MKKM [103]	LI-MKKM-MR Proposed
ACC							
Caltech102-5	26.07 ± 0.34	25.65 ± 0.26	27.30 ± 0.28	29.00 ± 0.29	28.88 ± 0.32	31.42 ± 0.33	$\mathbf{34.01 \pm 0.28}$
Caltech102-10	19.73 ± 0.19	19.68 ± 0.24	21.51 ± 0.20	22.56 ± 0.20	22.74 ± 0.17	27.30 ± 0.23	$\mathbf{28.94 \pm 0.27}$
Caltech102-15	17.12 ± 0.23	17.08 ± 0.17	18.89 ± 0.13	20.32 ± 0.19	20.79 ± 0.24	25.05 ± 0.23	$\mathbf{27.04 \pm 0.35}$
Caltech102-20	15.67 ± 0.12	15.65 ± 0.22	17.29 ± 0.16	18.89 ± 0.20	19.47 ± 0.14	24.12 ± 0.16	$\mathbf{26.24 \pm 0.18}$
Caltech102-25	14.65 ± 0.18	14.58 ± 0.13	16.24 ± 0.13	17.71 ± 0.20	18.26 ± 0.24	23.28 ± 0.15	$\mathbf{25.47 \pm 0.24}$
Caltech102-30	14.15 ± 0.12	14.05 ± 0.14	15.51 ± 0.16	17.11 ± 0.18	17.80 ± 0.22	22.22 ± 0.14	$\mathbf{24.60 \pm 0.14}$
NMI							
Caltech102-5	64.33 ± 0.18	63.93 ± 0.14	65.87 ± 0.19	66.55 ± 0.11	66.48 ± 0.16	67.09 ± 0.15	$\mathbf{68.57 \pm 0.19}$
Caltech102-10	53.62 ± 0.12	53.65 ± 0.09	55.24 ± 0.11	55.72 ± 0.19	55.83 ± 0.14	58.74 ± 0.12	$\mathbf{59.15 \pm 0.25}$
Caltech102-15	47.40 ± 0.13	47.39 ± 0.11	48.82 ± 0.11	49.69 ± 0.13	50.06 ± 0.11	53.56 ± 0.12	$\mathbf{54.55 \pm 0.20}$
Caltech102-20	43.11 ± 0.10	43.08 ± 0.17	44.54 ± 0.12	45.58 ± 0.15	46.03 ± 0.07	50.35 ± 0.09	$\mathbf{51.78 \pm 0.11}$
Caltech102-25	39.98 ± 0.10	39.88 ± 0.11	41.47 ± 0.09	42.45 ± 0.15	42.96 ± 0.18	47.74 ± 0.15	$\mathbf{49.44 \pm 0.10}$
Caltech102-30	37.78 ± 0.08	37.66 ± 0.12	39.15 ± 0.13	40.29 ± 0.12	40.92 ± 0.14	45.64 ± 0.11	$\mathbf{47.35 \pm 0.14}$
Purity							
Caltech102-5	26.73 ± 0.37	26.37 ± 0.31	27.90 ± 0.29	29.75 ± 0.34	29.59 ± 0.34	32.60 ± 0.29	$\mathbf{35.49 \pm 0.27}$
Caltech102-10	20.99 ± 0.15	20.97 ± 0.22	22.90 ± 0.22	23.96 ± 0.26	24.16 ± 0.23	29.03 ± 0.23	$\mathbf{30.82 \pm 0.28}$
Caltech102-15	18.52 ± 0.20	18.45 ± 0.16	20.39 ± 0.17	21.62 ± 0.18	22.16 ± 0.23	26.67 ± 0.18	$\mathbf{28.78 \pm 0.31}$
Caltech102-20	17.05 ± 0.11	17.02 ± 0.19	18.81 ± 0.22	20.17 ± 0.19	20.89 ± 0.14	25.84 ± 0.19	$\mathbf{28.09 \pm 0.19}$
Caltech102-25	16.02 ± 0.22	15.99 ± 0.15	17.74 ± 0.15	19.12 ± 0.15	19.71 ± 0.14	25.16 ± 0.17	$\mathbf{27.57 \pm 0.16}$
Caltech102-30	15.41 ± 0.10	15.36 ± 0.12	17.01 ± 0.14	18.39 ± 0.20	19.13 ± 0.22	24.04 ± 0.11	$\mathbf{26.45 \pm 0.14}$

LI-MKKM on clustering performance in all scenes from. For example, LI-MKKM-MR further improves LI-MKKM by 6.6%,8.8%,7.8%,7.4%,8.0%,7.4%,7.5%,6.9%,3.1%. This result indicates the effectiveness of incorporating the matrix-induced regularization.

- Moreover, our proposed method demonstrates stronger advantage when compared to previous ones, especially under low missing ratio. It is noticeably that in Fig.4.1 (a), when the missing ratio is extremely low (ε=0.1), LI-MKKM-MR improves the second best algorithm (LI-MKKM) by 6.6% in terms of clustering accuracy on Flower17.

In Table 4.2, the aggregated ACC, NMI, purity, and the standard deviation are reported. We highlight the best performance results in bold. Similarly, the results also show the fact that the proposed algorithm outperforms MKKM+ZF, MKKM+MF,

MKKM+KNN, MKKM+AF and MKKM-IK. Specifically, the proposed LI-MKKM-MR exceeds the second best one (LI-MKKM) by 7%.

Experiments on Caltech102 Dataset.

Figure 4.3 presents ACC, NMI and purity of all the testing algorithms over variational missing ratios on Caltech102 datasets. We observed that the newly proposed MKKM-IK [104] (in green) achieves a comparable clustering performance with a representative two-stage imputation method MKKM+AF, while the proposed LI-MKKM outperforms MKKM-IK with significant improvements on all the performance criterion. More precisely, LI-MKKM obtains 6.4%,5.0%,5.1%,4.7%,4.6%,4.5%,3.8%, 3.2%,2.6% higher clustering accuracy than MKKM-IK when the missing ratios vary from 0.1 to 0.9 on Caltech102-30. This also illustrates that the well utilization of the local structure of data assures performance improvement. Furthermore, by taking into account the correlation among base kernels, LI-MKKM-MR is capable of further improving the clustering performance on the base of LI-MKKM.

The aggregated ACC, NMI and purity, and the standard deviation on Caltech 102 datasets are reported in Table 4.3. Similarly, compare to MKKM+ZF, MKKM+MF, MKKM+KNN, MKKM+AF and MKKM-IK, our method still achieves much better clustering performance. For instance, the proposed LI-MKKM-MR obtains 2.1%,2.1%, 2.8%,2.4%,2.7%,2.4% higher clustering accuracy than LI-MKKM on each , which has the second-best performance on Caltech102 datasets. The results on Caltech102-5, Caltech102-10 and Caltech102-15 are also presented as follows, whose results demonstrate the same conclusion as well.

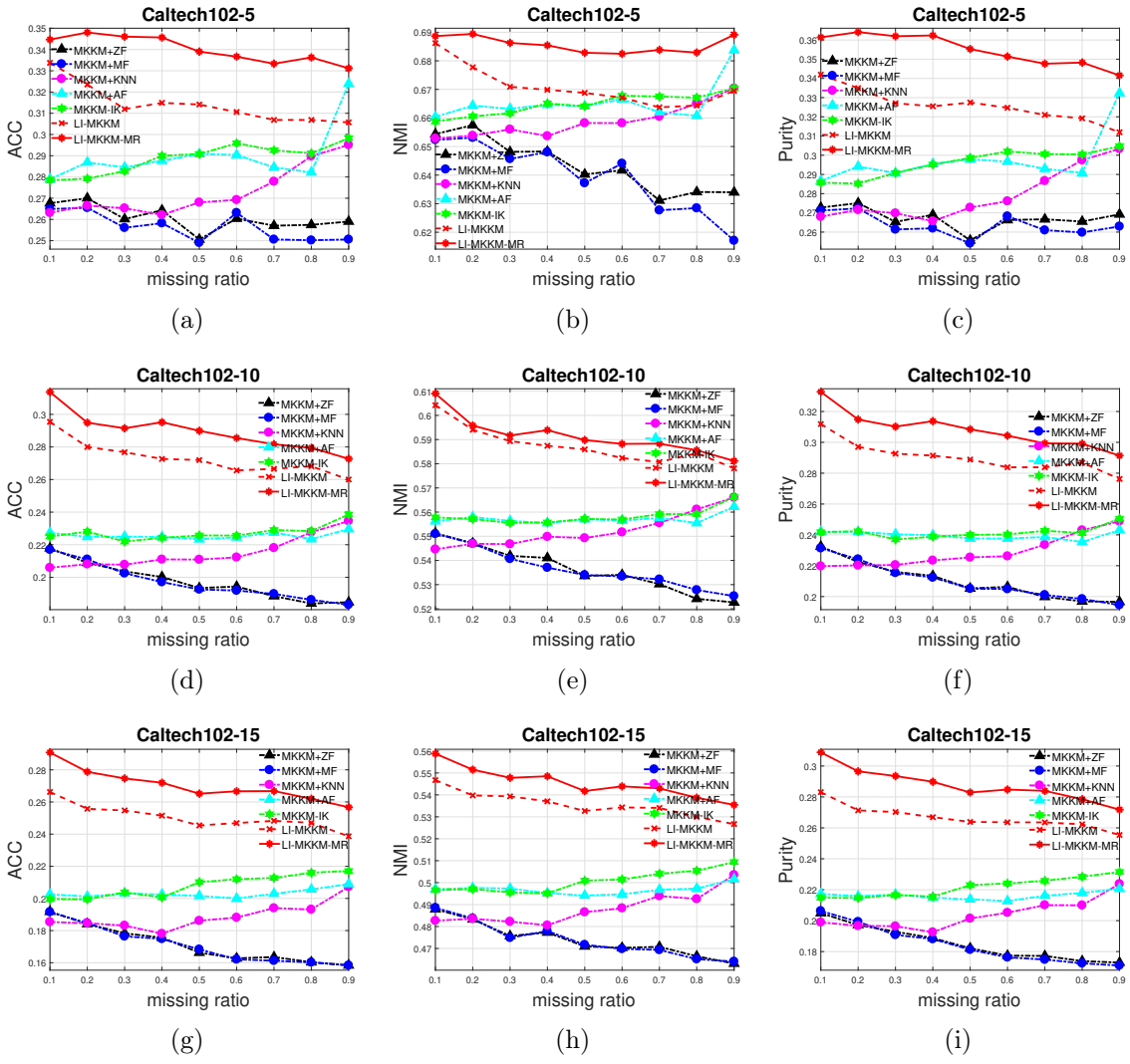
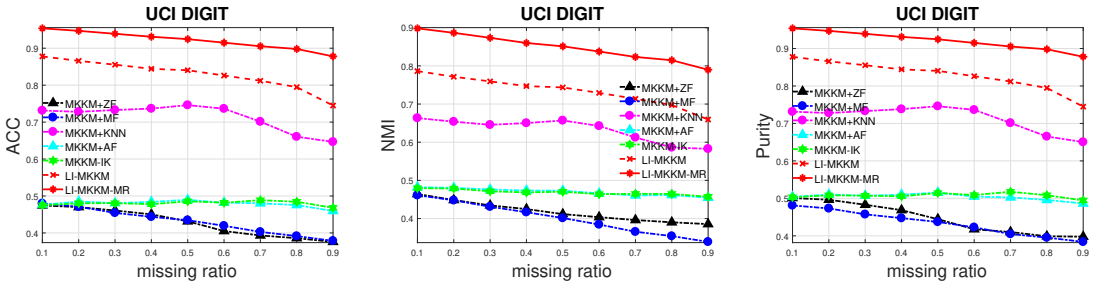


Figure 4.3: Clustering accuracy, NMI and purity comparison with the variation of missing ratios on Caltech102-5, Caltech102-10 and Caltech102-15. For each given missing ratio, the "incomplete patterns" are randomly generated for 10 times and their averaged results are reported.

Experiments on UCI-Digital Dataset.

In this simulation, we applied all the testing methods on UCI-Digital dataset, which is widely utilized in in MKC as a benchmark. For each missing ratio, the "incomplete patterns" are randomly generated for 10 times and their averaged results are reported.

The ACC, NMI and purity of all the testing algorithms over variational missing ratios are presented in Figure 4.4. It is clear that the latest proposed MKKM-IK provides unsatisfying performance on this dataset, which is even inferior to the MKKM+KNN. However, LI-MKKM significantly outperforms the second best one (MKKM+KNN) by 22.2%,21.9%,20.6%,19.5%,17.9%,17.9%,20.4%,23.8% and 23.2% on accuracy. In addition, the proposed LI-MKKM-MR further consistently improves the clustering performance of LI-MKKM. The aggregated clustering results in Table 4.4 also denote the same performance.



(a) (b) (c)

Figure 4.4: Clustering accuracy, NMI and purity comparison with the variation of missing ratios on UCI-digital dataset. For each given missing ratio, the "incomplete patterns" are randomly generated for 10 times and their averaged results are reported.

Experiments on Protein Fold Prediction Dataset.

In this experiment, the protein fold dataset is applied to evaluate the testing methods, and we report all results in Figure 4.5 and Table 4.5. Once again, we observe that the proposed LI-MKKM-MR still achieves much more better results (the red line) than the rest algorithms in terms of ACC, NMI and purity on this dataset.

Table 4.4: Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on UCI-Digital data set.

Datasets	MKKM+ZF	MKKM+MF	MKKM+KNN	MKKM+AF [98]	MKKM-IK [104]	LI-MKKM [103]	LI-MKKM-MR Proposed
ACC							
Digital	42.74 ± 0.42	43.06 ± 0.29	71.32 ± 0.97	47.91 ± 0.46	48.02 ± 0.43	82.90 ± 0.27	$\mathbf{92.13 \pm 0.32}$
NMI							
Digital	41.77 ± 0.19	40.01 ± 0.21	63.27 ± 0.52	46.98 ± 0.23	46.87 ± 0.24	73.42 ± 0.26	$\mathbf{84.84 \pm 0.42}$
Purity							
Digital	44.64 ± 0.46	43.36 ± 0.26	71.44 ± 0.70	50.39 ± 0.33	50.75 ± 0.38	82.90 ± 0.27	$\mathbf{92.13 \pm 0.32}$

Table 4.5: Aggregated ACC, NMI and purity comparison (mean±std) of different clustering algorithms on Protein Fold Prediction dataset.

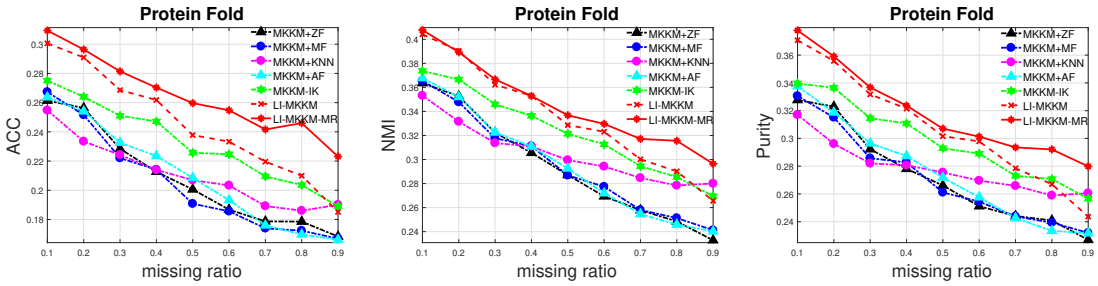
Datasets	MKKM+ZF	MKKM+MF	MKKM+KNN	MKKM+AF [98]	MKKM-IK [104]	LI-MKKM [103]	LI-MKKM-MR Proposed
ACC							
proteinFold	20.80 ± 0.20	20.49 ± 0.32	21.13 ± 0.51	20.97 ± 0.21	23.22 ± 0.56	24.53 ± 0.47	$\mathbf{26.47 \pm 0.21}$
NMI							
proteinFold	29.33 ± 0.35	29.52 ± 0.45	30.51 ± 0.40	29.54 ± 0.27	32.29 ± 0.56	33.53 ± 0.33	$\mathbf{34.58 \pm 0.18}$
Purity							
proteinFold	27.22 ± 0.39	27.16 ± 0.35	27.85 ± 0.51	27.52 ± 0.44	29.82 ± 0.67	30.76 ± 0.40	$\mathbf{31.90 \pm 0.30}$

Experiments on running time

In this section, we conduct experiments to record the running time of the compared algorithms, as reported in Table 4.6. The experiments are conducted on a PC with Intel(R) Core(TM)-i7-5820 3.3 GHz CPU and 32G RAM in MATLAB environment. It can be seen that the running time of LI-MKKM-MR is comparable with the previous methods. From Table 4.6, we observe that the proposed algorithm does not increase the running time of existing MKKM algorithms. The running time of MVEC [123] on Flower102, Caltech102-15, Caltech102-20, Caltech102-25, and Caltech102-30 are not reported due to out of memory.

Conclusion

In summary, we attribute the superiority of our algorithm as three aspects:



(a) (b) (c)

Figure 4.5: Clustering accuracy, NMI and purity comparison with the variation of missing ratios on protein Fold Prediction dataset. For each given missing ratio, the "incomplete patterns" are randomly generated for 10 times and their averaged results are reported.

Table 4.6: Running time of the compared algorithms on various benchmark datasets (in seconds)

Datasets	MKKM				MKKM-IK	LI-MKKM	MVEC	CG-IMVC	LI-MKKM-MR
	+ZF	+MF	+KNN	+AF [98]	[104]	[103]	[123]	[124]	Proposed
Flower17	2.0	2.0	2.9	2.4	3.7	9.0	456.7	5.6	8.5
Flower102	131.4	133.8	162.8	146.6	139.4	1259.4	-	331.7	1231.1
ProteinFold	1.5	1.3	1.8	1.4	6.9	9.2	110.4	3.0	3.4
UCI-Digit	2.3	2.3	2.4	2.6	3.1	17.5	1337.0	9.0	15.7
Caltech102-5	4.3	4.4	6.2	4.7	20.0	12.5	157.9	11.2	10.9
Caltech102-10	13.7	13.7	22.9	14.8	35.3	100.1	678.3	24.6	41.0
Caltech102-15	34.3	34.7	56.8	35.2	86.8	267.5	-	55.2	109.4
Caltech102-20	66.6	66.7	109.6	66.7	142.2	515.6	-	98.6	209.7
Caltech102-25	106.8	108.2	184.2	114.9	207.0	872.5	-	167.4	372.9
Caltech102-30	183.3	184.7	298.4	181.4	314.6	1505.3	-	278.1	603.7

- *The joint optimization on imputation and clustering.* The imputation and clustering processes are integrated comprehensively in our proposed methods. These two processes interact with each other dynamically in our model, therefore providing better performance. Meanwhile, MKKM+ZF, MKKM+MF, MKKM+KNN and MKKM+AF algorithms fail to fully take advantage of the connection between the imputation and clustering procedures. One major potential issue could be of a bad initialization that produced by imputation process for the clustering module.

- *Considerably utilizing the local structure of data.* Our local kernel alignment

criterion is flexible and allows the pre-specified kernels to be aligned for better clustering. This significantly improve the performance of the clustering results by providing a benign landscape for the optimization problem;

- *Well considering the correlation of base kernels.* The incorporated matrix-induced regularization reduces the high redundancy of kernel matrices and enhances low-diversity over the selected kernels. Thus the pre-specified kernels could be well utilized.

4.3.3 Parameter Sensitivity of LI-MKKM-MR

In this section, we analyze the relationship between the clustering performance and matrix-induced regularization. Referring to the Equation(4.2.3), the optimization on LI-MKKM-MR model induces the number of neighbors τ and regularization parameter λ. We show the variation of performance with respect to the parameters tuning on τ and λ based on simulations on Flower17 dataset.

Figure 4.6 (a) and 4.6 (b) shows the ACC and NMI of our algorithm by varying τ in a large range $[0.02 : 0.02 : 0.2] \times n$ with $\lambda = 2^{-6}$. It can be seen that: i) The ACC curves first increase and then decrease after a certain value of τ. That also yields for . ii) The start points of the ACC curves are typically higher than the end points, which induces that when the matrix-induced regularization term is dominated at ending points while the local kernel alignment maximization is dominated at starting points. These observations indicate the successful joint preservation of local structure of data and matrix-induced regularization term in our algorithm. Similarly, 4.6 (c) and 4.6 (d) presents the ACC and NMI of our algorithms by tuning λ from 2^{-9} to 2 with $\tau = 0.1 \times n$. In this scenario, our algorithm also shows stable performance over variational λ. As aforementioned, we conclude that comparing to only preserving global kernel alignment, such as MKKM-IK in [104], our proposed method preserves the local

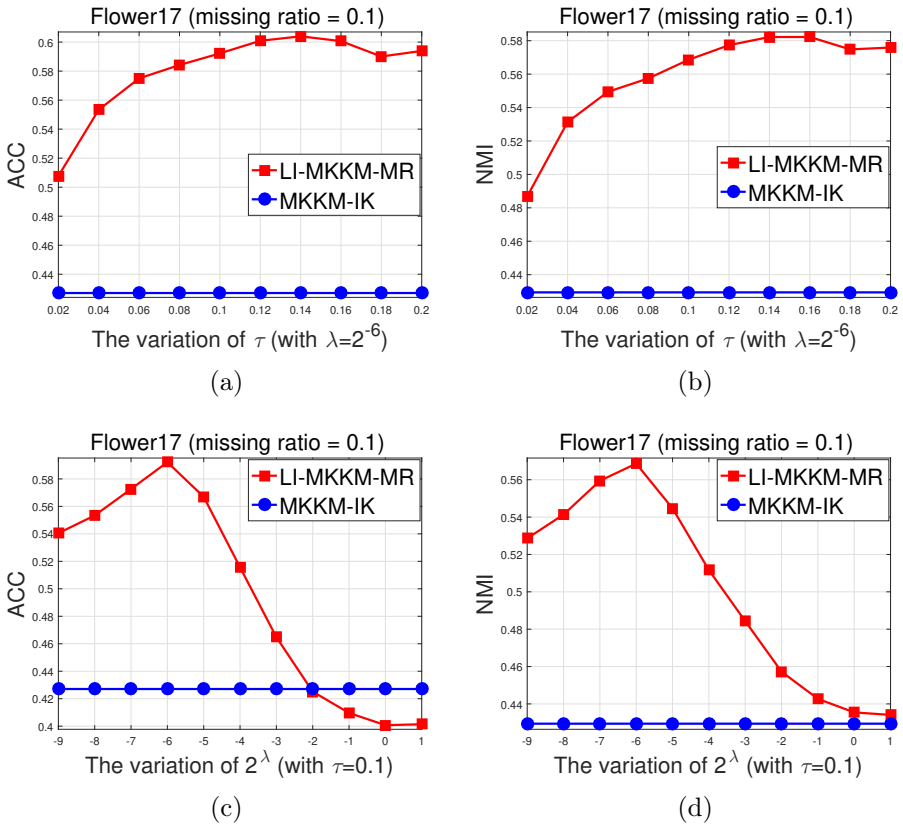


Figure 4.6: The sensitivity of LI-MKKM-MR with the variation of λ and τ.

structure of data, which is more essential to the clustering performance. Meanwhile, the clustering performance could be further improved by incorporating the correlation among base kernels. By appropriately integrating these two factors, it is possible to obtain the best clustering performance. Practically, there exists a trade-off between the preservation of the local geometric structure and the correlation of base kernels to ensure the best clustering.

4.4 Summary

In this chapter, we propose an improved non-convex non-linear LI-MKKM-MR model for solving incomplete MKC problems. The proposed model introduce a matrix-induced regularization to address the lack of sufficient utilization on correlation among base kernels in previous methods. Though the new model involves three variables that are

underlying non-linear mappings, the advanced freedom and feasibility of taking fully account on incomplete kernel through structural constraints and regularization terms allow a better landscape of the geometry of this non-convex non-linear optimization problem. Besides, we propose a three-step algorithm with the derivation of the generalization error bound. Extensive simulations on specific clustering datasets have proved the effectiveness of our LI-MKKM-MR model, that outperforms previous algorithms on performance.

In this stage, it is straightforward that aforementioned MKC problem is impossible to be modelled in convex optimization form. The excellent feasibility of mapping complicated natural settings and structural constraints in non-convex optimization is verified in this application as well. Although we still adopt a heuristic iterative optimization algorithm to solve the proposed model, the convergence and the generalization error bound of the algorithm can be derived with sufficient constraints to provide benign landscape. Therefore, we infer that deep understanding on the non-convex optimization mechanism offers significant advance on non-linear problem in the terms of model design and solution algorithms.

Chapter 5

Learning to Optimize Non-convex Multi-variable problems

5.1 Introduction

In this chapter, a meta-learning based approach for non-convex multi-variable problems are proposed. The non-convex multi-variable optimization problems play crucial roles in various machine learning tasks and applications, such as Gaussian mixture models (GMMs), dictionary learning, and matrix completion problems. However, it is challenging to optimize multiple variables stably and accurately. For instance, the multi-variable setting in GMMs makes the optimization very sensitive to initialization; dictionary recovery suffers from instability, and matrix completion requires specific structures on the matrix to be recovered and needs sufficient observations as well.

Recalling the definition in Section 2.1.2, a general formulation of non-convex optimization problem over an intersection of two variables is defiened as follows:

$$\min_{(\boldsymbol{W},\boldsymbol{X})\in\mathcal{W}\times\mathcal{X}} F(\boldsymbol{W},\boldsymbol{X}), \tag{5.1.1}$$

where $F : \mathcal{W} \times \mathcal{X} \rightarrow \mathbb{R}$ is a non-convex function with two variables $\boldsymbol{W} \in \mathcal{W}$ and $\boldsymbol{X} \in \mathcal{X}$. Considering there are N non-convex optimization problems $\{F_i(\boldsymbol{W}, \boldsymbol{X})_i\}_{i=1}^{N}$.

Referring to the analysis in Section 2.1.2 and two applications in Chapters 3 and 4, it can be noted that existing approaches generally optimize over multiple variables by using a heuristic method. One of the most widely adopted approaches is the AM approach, which converts the original problem into a sequence of iterative optimizations on a set of sub-problems corresponding to each variable. Due to the non-convexity of the objective function over multiple variables, the obtained solutions might be local optimum in the original problem, even when all sub-problems reach global optimums individually through the AM procedure at each iteration. So far, however, there has been little discussion about taking the global loss information to direct the optimization procedures in sub-problems. Therefore, instead of optimizing each sub-problem individually, it is essential to design an efficient and adaptive optimization algorithm to optimize each sub-problem within a bird eye on the overall problem. Recalling the aforementioned learning strategy of meta-learning, we observe that each sub-problem can be considered as a single task, and each overall problem is composed by a set of sub-problems. Defining the knowledge inside each sub-problem as intra-information, and the mutual knowledge across different sub-problems as inter-information. The meta-learning strategy aims to collect intra-information within each sub-problem, and then to extract inter-information across different sub-problems to update the optimization algorithm for variables within the down-streaming sub-problems.

Therefore, we propose a new meta-learning based method for solving the non-convex multi-variable optimization problem. We will first show in this thesis that to the end of minimizing the global loss function, there exists a better descending step for each variable in a single iteration compared to the one obtained by exhaustively optimizing each variable locally and individually on each sub-problem. In other words, to achieve a faster and better convergence in terms of the global loss function, the descending steps

are not necessarily leading to the optimal solution for each sub-problem. The proposed model exploits meta-learners to generate optimizers, which perform as variable update functions to update variables at inner loops. The meta-learners are updated at outer loops with respect to the global losses over multiple outer loop courses. Our proposed method aims to achieve a global scope optimization. We thus name the proposed optimization strategy as the global scope optimization (GSO). Our GSO method automatically learns to appropriately optimize each sub-problem under the guidance of the overall objective function. In a sense, the updates of meta-learners build an interaction between inner loop updates for variables and the global loss variation at outer loops. Therefore, an adaptive and global gradient descent rule is continuously generated by meta-learners through iterations and applied on the sub-problems within each iteration. To achieve this, we build two levels of meta-learning process in our model: the first level is to leverage previous optimization experiences across different sub-problems, and therefore this will enable the learned optimization strategy to extract the inter-information of sub-problems as inductive bias for generating adaptive algorithm across sub-problems within an overall problem; the second level meta-learning process leverages optimization experiences across different overall problems in training sets, and hence it will enable to improve the adaptability and efficiency of learned algorithm for solving a class of overall problems instead of overfitting on a specific one.

Specifically, the GSO model builds this connection via generating the variable updating functions by meta-learner networks. The parameters of these meta-learner networks are updated through backpropagation with the objective to minimize the accumulated global losses, and are fixed while generating the variable update functions. Consequently, an interaction between variable update rule and global losses is built to allow the variations on global loss directly impact update rule.

In this thesis, we exploit a LSTM network based meta-learner model [106] to construct our GSO model. The LSTM meta-learner continuously generates dynamic gradient

descent strategies for each variable within each sub-problem. It stores the information of loss trajectories and variable status within each previous sub-problem by the memory cell in the LSTM network, and learns a less-greedy algorithm for the overall problem via updating the network parameters by the backpropagation of global loss variation over iterations. Consequently, the learned algorithms result in more "optimally" descending points for the global loss function. In this way, a global scope optimization, that provides information of global geometry for each sub-problem on "local" geometry, can be achieved.

The contribution of this chapter is mainly three-fold:

- We propose a meta-learning based GSO model for solving non-convex optimization problems over the intersection of multiple variable sets. Our GSO model establishes an interaction between variable updates within sub-problems and global loss function which allows the variables to be directly and constantly updated with the objective of minimizing global loss.
- We apply a hierarchical LSTM-based meta-learner model to implement our GSO model in this thesis. The LSTM networks generate variable update functions with fixed parameters, and update parameters with respect to minimizing global losses in two levels of meta-learning process.
- We have successfully implemented the proposed GSO method for those non-convex multi-variable problems where traditional methods would typically fail. We evaluate our GSO method on matrix completion problems and GMM problems in some challenging scenarios, such as completion of matrices without rank knowledge, and matrices with different ranks completion.

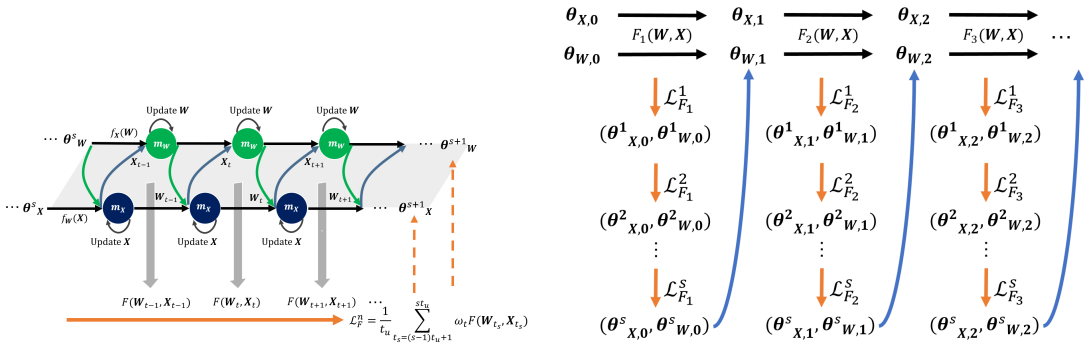
The rest of this chapter is organized as follows. Section 5.2 gives an overview of our proposed global scope optimization approach. Section 5.3 presents an LSTM-based meta-learner model, and Section 5.4 illustrates two practical cases of our GSO method..

Section 5.5 presents the results of our extensive experimental works and Section 5.6 draws conclusions.

5.2 Overall Structure of the GSO Method

In this section, we will demonstrate our GSO method in the aspects of basic motivations, microscopical structure, and structural properties. The notations follow the definitions in Section 2.1.2.

As mentioned in Section 2.1.2, the inherent limitation of the AM-based approaches is that the updating functions optimize over the sub-problems rather than the overall problem. At each iteration, a variable can only be optimized with respect to its corresponding sub-problem and updated to the first order stationary points of the sub-problem. However, this approach may not be the best choice in terms of optimizing the overall problem. There is a potential mismatch between the true and the AM-based optimization objectives. As a result, the AM-based methods may not be able to achieve satisfactory convergence speed and performances.



(a) First-level meta-learning within $F_i(\boldsymbol{W}, \boldsymbol{X})$. (b) Second-level meta-learning on training set $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^N$.

Figure 5.1: There are two levels of meta-learning in our proposed approach. The first level of meta-learning is performed within each single task $F_i(\boldsymbol{W}, \boldsymbol{X})$ which consists of a sequence of sub-problems. The second level of meta-learning is performed across a set of training tasks $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^N$. At the first and the second level, our model learns to achieve a fast and high quality minimization for the current optimization problem and all optimization problems, respectively.

Considering the limitation of the AM-based methods, our motivation is to optimize each variable with respect to its corresponding sub-problem as well as the overall problem. To achieve this, our GSO method casts the design of an optimization algorithm as a learning problem. The learning objective is to obtain an updating rule for each variable with the aim to optimize the overall problem. We propose to incorporate meta-learning mechanism in our GSO model. Thus the updating functions are modelled by meta-learner networks that are able to generate adaptive and dynamic updating rules across the sub-problems for each variable and can be tailored to a specific sub-problem at hand. In other words, the learned strategy is no longer designed for solving a single optimization problem, but a set of optimization problems. Each non-convex optimization problem is considered as a single task, and the whole learning process is accommodated on a set of tasks with the same variable dimension but different optimization landscapes.

Specifically, there are also nested loops in our GSO model. At the inner loops, the meta-learner networks provide the updating steps for variables with fixed network parameters. At the outer loops, the meta-learner networks update their parameters with respect to minimizing the global losses from the overall problem. The global loss information has therefore been used to update parameters, and then to determine the variable update function at following inner loops. As a result, our GSO model establishes a relationship between the variable update functions and global losses, which allows the learned updating functions may not necessarily exhaustively optimize the variables on each sub-problem, but is able to better accommodate the overall problem compared to the AM.

Remark 11 *In this thesis, our meta-learning approach aims to address the aforementioned issues in AM method. Therefore, we essentially use meta-learning in a smart way to achieve better performance on non-convex multi-variable optimization problems. Directly applying meta-learning on the joint optimization problem is different and will be considered as a potential future work.*

Our GSO model contains two levels of meta-learning processes. Corresponding to the inner and outer loops in the general AM algorithms, the first level of meta-learning considers each inner loop of solving a sub-problem as one training sample, and the second level meta-learning takes each outer loop of solving an overall problem (task) as one training sample. Detailed illustrations are given as follows:

- The first level of meta-learning corresponds to the inner loops in Algorithm 1. The general structure of the first-level meta-learning is shown in Figure 5.1(a), and it is performed within a single overall problem $F_i(\boldsymbol{W}, \boldsymbol{X})$. Within each inner loop, the updating function aims to provide a good solution for the sub-problem. As the optimization landscapes for different sub-problems can be significantly different, the learned updating function should be able to achieve a good and adaptive estimation for different sub-problems. That is, the mutual knowledge on updating rules for variables across different sub-problems should be learned through the meta-learning process. In other words, each inner loop within a sub-problem can be considered as a solving process for a single task in this meta-learning. In this way, two networks learn to work jointly and iteratively as updating functions to achieve a superior adaption at all inner loops for solving $F_i(\boldsymbol{W}, \boldsymbol{X})$.

- The second level of meta-learning corresponds to the outer loop in Algorithm 1. As depicted in Figure 5.1(b), the second level of meta-learning aims to extract the general knowledge across different non-convex optimziation problems (tasks). Even for a same type of non-convex optimization problem, the optimization trajectory of each problem might vary significantly due to the different parameters setting, i.e., for matrix completion problem, if the ranks of matrices to be estimated are different, the optimization process will provide different experience of convergence. Hence the training tasks $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^{N}$ have the same variable dimensionality while having significantly different optimization landscapes. Therefore, our second level meta-learning is with the objective to extract the

general knowledge across a given set of optimization problems. As a result, the learned GSO model is able to achieve a fast adaptation on new optimization problems and possess a good generalization ability at all the outer loops for each task.

Although the proposed GSO model follows the AM structure that solves overall problem under the framework of inner and outer loops, we establish a new bridge between the inner and outer loops by replacing previous updating functions with meta-learning networks. This allows variable updates at the inner loops being guided by global losses information from the outer loops, thus achieving a global scope optimization.

5.3 GSO Method with Hierarchical LSTM-based Mata-learners

In this section, we will introduce the details of using recurrent neural networks (RNNs) as the meta-learners in our GSO model. Our GSO model is composed by Long Short-Term Memory (LSTM)-based meta-learners. In the following, we will briefly introduce the properties of LSTM networks, followed by the LSTM-based GSO model in detail.

RNN has a sequentially processing chain structure to achieve the capacity of "memory"on sequential data. LSTM is one of the most well-known RNNs, which is able to memorize and ignore sequential data. Memory is the most important feature of LSTM (RNN). It stores the status information of previous iterations and allows the information to flow along the entire chain process. In this way, the LSTM is able to integrate previous information with the current step input [125]. Specifically, the output of LSTM at the i-th iteration $\boldsymbol{H}^{(i)}$ is determined by the current gradient $\nabla f(\boldsymbol{x}_i)$

and the last cell state $\boldsymbol{C}^{(i)}$ in the following forms:

$$\begin{aligned} \boldsymbol{H}^{(i)} &= m(\nabla f(\boldsymbol{x}_i), \boldsymbol{C}^{(i)}, \boldsymbol{\theta}), \\ \boldsymbol{C}^{(i+1)} &= \boldsymbol{z}_{forget} \odot \boldsymbol{C}^{(i)} + \boldsymbol{z}_{infor} \odot \tilde{\boldsymbol{C}}^{(i-1)}, \end{aligned} \tag{5.3.1}$$

where $m(\cdot, \cdot, \boldsymbol{\theta})$ denotes the LSTM network with parameters $\boldsymbol{\theta}$, $\odot$ denotes Hadamard Product, $\boldsymbol{z}_{forget}$ and $\boldsymbol{z}_{infor}$ are the vectors of intermediate conditions inside of LSTM, and $\tilde{\boldsymbol{C}}^{(i)} = \nabla_{\boldsymbol{\theta}^{(i-1)}} \mathcal{L}^{(i)}$ is the candidate cell state, referring to the gradient of current loss $\mathcal{L}^{(i)}$ over the last parameters $\boldsymbol{\theta}^{(i-1)}$ in our problem.

Another key property of LSTM is that it can record long-term memory data while avoiding gradient vanishing or explosion. This is achieved by the forget gate $\boldsymbol{z}_{forget}$ that determines what information in previous states should be ignored, and the information gate $\boldsymbol{z}_{infor}$ that determines what information in the current new task should be stored. For more detailed introduction about LSTMs, we refer to [110, 125].

We adopt two LSTM networks $m_{\boldsymbol{X}}$ and $m_{\boldsymbol{W}}$ as meta-learners to generate updating step functions $\phi_{\boldsymbol{X}}()$ and $\phi_{\boldsymbol{W}}()$ for the sub-problems corresponding to variables $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively. We correspondingly denote $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ as the parameters of $m_{\boldsymbol{X}}$ and $m_{\boldsymbol{W}}$, and denote $\boldsymbol{C}_{\boldsymbol{X}}$ and $\boldsymbol{C}_{\boldsymbol{W}}$ as their cell states. The inputs of the LSTM networks are the gradients of sub-problems, and previous information of variables which is represented by cell state $\boldsymbol{C}$. The outputs of the LSTM networks include updating steps for variables, along with the update on cell state for the next step. Considering the inner loops update steps $i-1$ and $j-1$ at superscripts, and outer loops steps $t-1$ and t at subscripts for each sub-problem, the variables are updated in the following forms:

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{X}}^{(i-1)} &= m_{\boldsymbol{X}} \left(\nabla f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}^{(i-1)}), \boldsymbol{C}_{\boldsymbol{X}}^{(i-1)}, \boldsymbol{\theta}_{\boldsymbol{X}} \right), \\ \boldsymbol{X}^{(i)} &= \boldsymbol{X}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{X}}^{(i-1)}, \end{aligned} \tag{5.3.2}$$

and

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{W}}^{(j-1)} &= m_{\boldsymbol{W}}\left(\nabla f_{\boldsymbol{X}_t}(\boldsymbol{W}^{(j-1)}), \boldsymbol{C}_{\boldsymbol{W}}^{(j-1)}, \boldsymbol{\theta}_{\boldsymbol{W}}\right), \\ \boldsymbol{W}^{(j)} &= \boldsymbol{W}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{W}}^{(j-1)}. \end{aligned} \tag{5.3.3}$$

The cell states $\boldsymbol{C}_{\boldsymbol{W}}^{(j)}$ and $\boldsymbol{C}_{\boldsymbol{X}}^{(i)}$ contain the information of previous variable values, and can be adaptively adjusted by the inner control gates $\boldsymbol{z}_{forget}$ and $\boldsymbol{z}_{infor}$ inside of LSTM (recalling (5.3.1)). Thus, the outputs of two LSTM networks are essentially embedded with previous variables positions $\{\boldsymbol{X}^{(k)}\}_{k=0}^{i-1}$ and $\{\boldsymbol{W}^{(k)}\}_{k=0}^{j-1}$ at each inner loops step.

At the inner loops, the parameters $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ are fixed, and generate the updating steps $\boldsymbol{H}_{\boldsymbol{X}}$ and $\boldsymbol{H}_{\boldsymbol{W}}$ for variables. At the outer loops, we leverage the accumulated global losses to guide the parameter update for $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ through backpropagation. The accumulated global loss is given by

$$\mathcal{L}_F^s = \frac{1}{t_o} \sum_{t_s=(s-1)t_o+1}^{st_o} \omega_{t_s} F(\boldsymbol{W}_{t_s}, \boldsymbol{X}_{t_s}), \tag{5.3.4}$$

where $\omega_t \in \mathbb{R}_{\geq 0}$ denotes the weights associated with each outer step, and $s = 1, 2, \ldots, S$, $S = T/t_o$, T is the maximum outer step and S is the maximum update times for meta-learner networks. For every t_o outer loop iterations, the accumulated global losses $\mathcal{L}_F^s$ is computed and is used to update $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ according to equation (5.3.5). As t_o essentially indicates the update interval of parameters at outer loop, we define t_o as the outer loops update times. The updates of parameters are given by

$$\begin{aligned} \boldsymbol{\theta}_{\boldsymbol{X}}^{s+1} &= \boldsymbol{\theta}_{\boldsymbol{X}}^{s} + \alpha_{\boldsymbol{X}} \cdot \mathrm{Adam}(\boldsymbol{\theta}_{\boldsymbol{X}}^{s}, \nabla_{\boldsymbol{\theta}_{\boldsymbol{X}}^{s}} \mathcal{L}_F^s), \\ \boldsymbol{\theta}_{\boldsymbol{W}}^{s+1} &= \boldsymbol{\theta}_{\boldsymbol{W}}^{s} + \alpha_{\boldsymbol{W}} \cdot \mathrm{Adam}(\boldsymbol{\theta}_{\boldsymbol{W}}^{s}, \nabla_{\boldsymbol{\theta}_{\boldsymbol{W}}^{s}} \mathcal{L}_F^s), \end{aligned} \tag{5.3.5}$$

where $\alpha_{\boldsymbol{X}}$ and $\alpha_{\boldsymbol{W}}$ denote the learning rate for the LSTM networks $m_{\boldsymbol{X}}$ and $m_{\boldsymbol{W}}$, respectively. The parameters of LSTMs are updated using ADAM optimizer [126]. Hence, $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ successfully build a connection between the variable update functions and the global losses. At inner loops, $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ provide an extra global loss

Algorithm 4: General Structure of GSO algorithm

Given: global loss function $F(\boldsymbol{W}, \boldsymbol{X})$, local loss functions $f_{\boldsymbol{W}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}}(\boldsymbol{W})$:
for $t \leftarrow 1,2, \ldots, T$ **do**
  $i, j, s = 1$
  while $i \leq I$ **do**
    $\Delta \boldsymbol{X} = m_{\boldsymbol{X}}(\nabla f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}^{(i-1)}), \boldsymbol{C}_{\boldsymbol{X}}^{(i-1)}, \boldsymbol{\theta}_{\boldsymbol{X}}^{s})$;
    $\boldsymbol{X}^{(i)} \leftarrow \boldsymbol{X}^{(i-1)} + \Delta \boldsymbol{X}$;
    $i = i + 1$
  end
  $\boldsymbol{X}_t \leftarrow \boldsymbol{X}^{(i)}$, then $f_{\boldsymbol{X}_t}(\boldsymbol{W}) \leftarrow \boldsymbol{X}_t$
  while $j \leq J$ **do**
    $\Delta \boldsymbol{W} = m_{\boldsymbol{W}}(\nabla f_{\boldsymbol{X}_t}(\boldsymbol{W}^{(j-1)}), \boldsymbol{C}_{\boldsymbol{W}}^{(j-1)}, \boldsymbol{\theta}_{\boldsymbol{W}}^{s})$;
    $\boldsymbol{W}^{(j)} \leftarrow \boldsymbol{W}^{(j-1)} + \Delta \boldsymbol{W}$;
    $j = j + 1$
  end
  $\boldsymbol{W}_t \leftarrow \boldsymbol{W}^{(j)}$, then $f_{\boldsymbol{W}_t}(\boldsymbol{X}) \leftarrow \boldsymbol{W}_t$
  $F(\boldsymbol{W}_t, \boldsymbol{X}_t) \leftarrow \boldsymbol{W}_t, \boldsymbol{X}_t$
  while $s \leq t/t_o$ **do**
    $\mathcal{L}_F^s = \frac{1}{t_o} \sum_{t_s=(s-1)t_o+1}^{st_o} \omega_{t_s} F(\boldsymbol{W}_{t_s}, \boldsymbol{X}_{t_s})$
    $\boldsymbol{\theta}_{\boldsymbol{X}}^{s+1} = \boldsymbol{\theta}_{\boldsymbol{X}}^{s} - \alpha_{\boldsymbol{X}} \nabla_{\boldsymbol{\theta}_{\boldsymbol{X}}^{s}} \mathcal{L}_F^s$
    $\boldsymbol{\theta}_{\boldsymbol{W}}^{s+1} = \boldsymbol{\theta}_{\boldsymbol{W}}^{s} - \alpha_{\boldsymbol{W}} \nabla_{\boldsymbol{\theta}_{\boldsymbol{W}}^{s}} \mathcal{L}_F^s$
    $s = s + 1$
  end
end

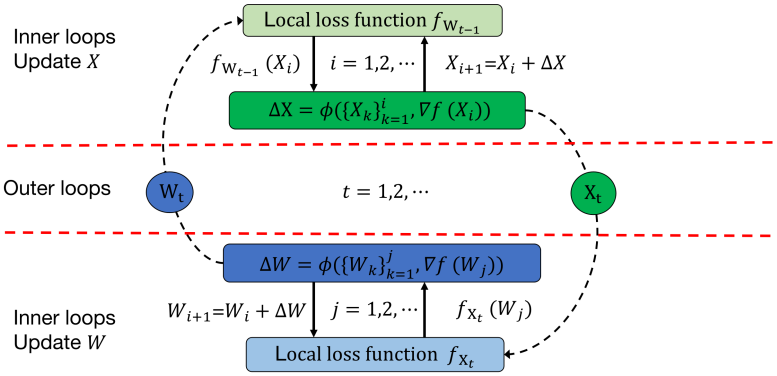
knowledge from the outer loops for the update functions compared to the standard one in Algorithm 1.

At this stage, a general structure of our proposed GSO algorithm is summarized in Algorithm 4. The algorithm starts from a random initialization $\boldsymbol{W}_0$ at the beginning of outer loop. At the t-th outer loop, it contains two inner loops for updating $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively. Each inner loop starts from a random initialization $\boldsymbol{X}^{(0)}$ and $\boldsymbol{W}^{(0)}$ and updates variables based on equations (5.3.2) and (5.3.3) and repeats for I and J times, respectively. In this thesis, we set $I = J = t_i$ in which t_i indicates the maximum steps for meta-learners to generate variable update terms and thus update variables at inner loops. At the end of each inner loop, the output of this inner loops will be regarded as $\boldsymbol{X}_t$ or $\boldsymbol{W}_t$ at the t-th outer loop, and are then assigned to generate

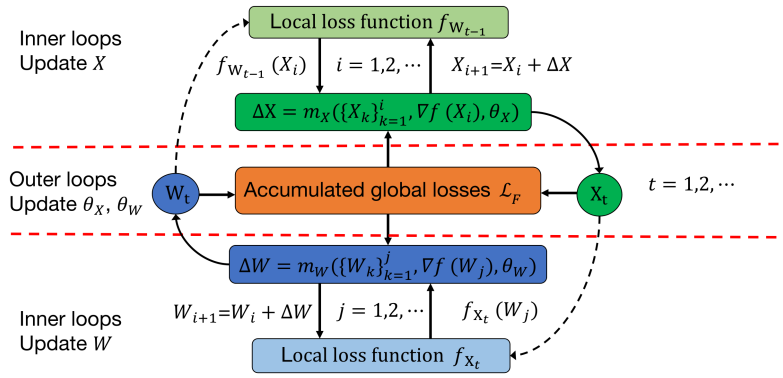
sub-problem $f_{\boldsymbol{X}_t}$ and $f_{\boldsymbol{W}_t}$, respectively. As aforementioned, for every t_o steps at outer loops, the parameters $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ are updated with respect to equations (5.3.5). The choices strategy for t_o, t_i are discussed in section 5.5. Finally, GSO algorithm will stop when t reaches the maximum setting number T.

Remark 12 *In this thesis, we set $I = J = t_i$ for demonstration convenience. In practical scenarios, it is reasonable to set different value for I and J according to the demand of the objective in practical problems.*

Our GSO algorithm follows the inner and outer loops framework, and improves the standard AM-based method by incorporating LSTM based meta-learners as our variable updating functions. In contrast, it can be seen in Figure **??**, the main difference between our GSO model (in Figure 5.2(b)) and AM-based model (in Figure 5.2(a)) is that there is an extra updating cue for the parameters of LSTM networks $m_{\boldsymbol{X}}$ and $m_{\boldsymbol{W}}$ at the outer loops. It minimizes $\mathcal{L}_F^n$ using gradient descent on $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$, respectively. In Fig 5.2(a), the information flows only circulate inside each inner loop, indicating that only the gradients of $f_{\boldsymbol{W}}$ and $f_{\boldsymbol{X}}$ are taken into account along with the optimization. However, in Figure 5.2(b), the gradients of $f_{\boldsymbol{W}}$ and $f_{\boldsymbol{X}}$ on $\boldsymbol{X}$ and $\boldsymbol{W}$ and the gradient of $\mathcal{L}_F^n$ on parameters $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ are integrated into one circulating system. At this stage, we can infer that a hierarchical LSTM-based meta-learner model is employed to achieve the global scope optimization. It contains two (or more for multiple variables) LSTM networks which process data at inner loops with fixed parameters and update parameters at outer loops with respect to minimizing accumulated global losses. The information flows are streaming across inner and outer loops, which allows global scope information to be well accommodated for the objective of updating multiple variables simultaneously.



(a) Information flow in AM-based model.



(b) Information flow in GSO model.

Figure 5.2: In the AM-based model, there are only information flows of variables inside the corresponding inner loops while the outer loops are isolated from inner loops optimizations. In contrast, our GSO model builds a new connection between inner and outer loops, which allows the information flows of variables circulate among these two level loops. We use solid edges to denote information flow along arrow, and dashed edges indicate there is no information flow.

5.4 Applications in Typical Problems

In this section, we show potential implementations of our GSO model on solving typical optimization problems with the intersection of two variable sets. More specifically, we address both a bi-linear inverse problem and a non-linear problem, via their representative tasks: the matrix completion and the GMM problems, respectively.

5.4.1 Bi-linear Inverse Problem: Matrix Completion

One of the most well known optimization problems over the intersection of two variable sets is the bi-linear inverse problem, which is a challenging and representative task of optimizing two variables, with a variety of important practical examples including blind deconvolution [11], blind source separation [127], dictionary learning [128], and matrix completion [129].

We take the matrix completion problem as a representative case of bi-linear inverse problems and we should point out that our proposed method can be applied to other bi-linear inverse problems in a similar way. The matrix completion problem is typically solved as a low-rank matrix recovery problem, which represents a low-rank matrix $\boldsymbol{R} \in \mathbb{R}^{p\times q}$, by a multiplication of two rectangular matrices as $\boldsymbol{U} \in \mathbb{R}^{p\times s}$ and $\boldsymbol{V} \in \mathbb{R}^{q\times s}$ [4, 18]. The matrix factorization problem is then formulated as [19],

$$\min_{\boldsymbol{U},\boldsymbol{V}} F(\boldsymbol{U},\boldsymbol{V}) := \frac{1}{2}\left\|\mathcal{P}_\Omega(\boldsymbol{R}-\boldsymbol{U}\boldsymbol{V}^T)\right\|_F^2 + \frac{\lambda}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2), \tag{5.4.1}$$

where the projection $\mathcal{P}_\Omega(\cdot)$ preserves the observed elements defined by Ω and replaces the missing entries with 0.

Here we define problem (5.4.1) is the overall problem, and $F(\boldsymbol{U},\boldsymbol{V})$ is the global loss function. Problem (5.4.1) is typically NP-hard and non-convex, and thus unique solution is not always guaranteed.

It is straightforward that the overall problem $F(\boldsymbol{U}, \boldsymbol{V})$ is not convex in $\boldsymbol{U}$ and $\boldsymbol{V}$, but the sub-problems are convex when fixing one variable and updating the other. Therefore, we split the problem (5.4.1) into two sub-problems in quadratic form with respect to fixing one variable in (5.4.1) and updating the other one, referring to $f_{\boldsymbol{U}}(\boldsymbol{V})$ and $f_{\boldsymbol{V}}(\boldsymbol{U})$. As mentioned in Section 2.1.1, Alternating Least Square (ALS) and Stochastic Gradient Descent (SGD) can be applied to solve overall problem $F(\boldsymbol{U}, \boldsymbol{V})$ by iteratively minimizing the two sub-problems.

At the same time, according to Algorithm 4 our GSO method can be directly applied to solve problem (5.4.1) by establishing two LSTM networks $m_{\boldsymbol{U}}$ and $m_{\boldsymbol{V}}$ with parameters $\boldsymbol{\theta_U}$ and $\boldsymbol{\theta_V}$, to optimize matrix $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. The variable update equations are given by

$$\begin{aligned} \boldsymbol{U}^{(i)} &= \boldsymbol{U}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{U}}^{(i-1)}, \\ \boldsymbol{V}^{(j)} &= \boldsymbol{V}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{V}}^{(j-1)}. \end{aligned} \tag{5.4.2}$$

Two LSTM networks are updated for every t_o outer loop steps, via back-propagating leveraged global losses $\mathcal{L}_F^s$ according to Equation (5.3.4). In this way the updating rule of the learned algorithm could be adjusted to find gradient descent steps $\boldsymbol{H_U}$ and $\boldsymbol{H_V}$ for minimizing $F(\boldsymbol{U}, \boldsymbol{V})$ adaptively.

The advantages of our GSO model for the matrix completion problem may be explained by the following aspects. One possible explanation lies in the replacement of updating step function ϕ by the neural networks $m_{\boldsymbol{U}}$ and $m_{\boldsymbol{V}}$. In the AM methods, when a local optimum $(\boldsymbol{U}', \boldsymbol{V}')$ is reached, the gradient of the variables is equal to zero, i.e., $\partial f_{\boldsymbol{V}'}(\boldsymbol{U}')/\partial \boldsymbol{U} = 0$ and $f_{\boldsymbol{U}'}(\boldsymbol{V}')/\partial \boldsymbol{V} = 0$. At this stage, the updating-step function, which is determined by the gradients of variables, sticks at local optimum points. However, things are different when it comes to our GSO model: our update functions are determined by their parameters, $\boldsymbol{theta_U}$ and $\boldsymbol{theta_V}$, which are further determined by leveraging of partial global loss trajectories across outer loops iteration steps. This major difference brings two potential benefits to the GSO method. One

advantage is that even at local optimum points, our neural network can still provide a certain step update on variables. This can be understood that even one of the input $\partial f_{\boldsymbol{V}'}(\boldsymbol{U}')/\partial \boldsymbol{U} = 0$, network $m_{\boldsymbol{U}}$ may still output a non-zero value respect to the Equations (5.3.2) and (5.3.3). Another advantage is that the leveraged global loss leads to a smooth optimization on a global loss landscape, which allows the learned algorithm to be essentially guided by the inductive bias from a smoother transform of global loss landscape.

5.4.2 Non-linear Problem: Gaussian Mixture Model

In contrast to the bi-linear inverse problem, the non-linear problem is more challenging due to the fact that there is no analytic landscape geometry for this type of problems and sub-problems. One of the most classical and significant problems is the GMM, which is a probabilistic model for representing observed data samples as weighted Gaussian components [29]. Detailed descriptions of GMM problems can be found in [130]. Given a set of G i.i.d samples $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^{G}$, each entry $\boldsymbol{x}_g$ is a D-dimensional data vector. Then, a typical optimization when using the GMM to model the samples is to maximize the log-likelihood (MLL) [29], which is equivalent to minimize the Kullback–Leibler divergence. The parameters of the GMM can then be optimized as follows,

$$\arg \max_{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \log p(\boldsymbol{X}) = \sum_{g=1}^{G} \log \sum_{k=1}^{K} \pi_k \mathcal{G}(\boldsymbol{x}_g | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5.4.3}$$

where

$$\mathcal{N}(\boldsymbol{x}_g | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{\exp\{-\frac{1}{2}(\boldsymbol{x}_g - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_g - \boldsymbol{\mu}_k)\}}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}}. \tag{5.4.4}$$

In (5.4.3), for the k-th Gaussian component, $\boldsymbol{\mu}_k$ is the mean vector defining the cluster centre, covariance $\boldsymbol{\Sigma}_k$ denotes the cluster scatter, and π_k represents mixing proportion, with $\sum_{k=1}^{K} \pi_k = 1$.

However, it is intractable to directly obtain a closed-form solution that maximizes

$\log p(\boldsymbol{X})$ in (5.4.3). The key difficulty is that by differentiating $\log p(\boldsymbol{X})$ (summation of logarithmic summation) and equalizing it to 0, each parameter is intertwined with each other. Gradient descent methods in an AM manner can alternatively solve (5.4.3), but they typically perform inferior to the one called expectation-maximization (EM) algorithm [26, 27].

We should also point out that the EM algorithm still updates the GMM parameters in an AM manner, i.e., iterating to optimum over each parameter at which its gradient equals to 0 with the other parameters being fixed. Furthermore, the EM algorithm, together with other first-order methods, has been proved to converge to arbitrary bad local optimum almost surely [31]. As we have discussed before, this AM strategy can be improved by replacing the fixed updating rule that searches optimum in a local landscape, by a less-greedy rule that updates variables up to global scope knowledge on loss landscape of global objective function.

Therefore, we propose to solve GMMs by adopting our GSO method, which directly applies a learning based gradient descent algorithm to the original MLL problem (5.4.3) without any extra constraints. In this scenario, we consider the GMM problem with covariance $\boldsymbol{\Sigma}$ being given; hence the MLL estimation of GMM is presented in terms of negative log-likelihood as follows

$$\arg\min_{\boldsymbol{\pi},\boldsymbol{\mu}} F_{ne}(\boldsymbol{\pi},\boldsymbol{\mu}) = -\sum_{g=1}^{G}\log\sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}_g|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k). \tag{5.4.5}$$

In this case, we treat the problem (5.4.5) as our overall problem, and split it into two sub-problems $f_{\boldsymbol{\pi}}(\boldsymbol{\mu})$ and $f_{\boldsymbol{\mu}}(\boldsymbol{\pi})$.

In our GSO framework, we build two LSTM neural networks $m_{\boldsymbol{\pi}}$ and $m_{\boldsymbol{\mu}}$ with param-

eters $\boldsymbol{\theta}_{\boldsymbol{\pi}}$ and $\boldsymbol{\theta}_{\boldsymbol{\mu}}$ to update $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ as follows

$$\begin{aligned} \boldsymbol{\pi}^{(i)} &= \boldsymbol{\pi}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{\pi}}^{(i-1)}, \\ \boldsymbol{\mu}^{(j)} &= \boldsymbol{\mu}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{\mu}}^{(j-1)}, \end{aligned} \tag{5.4.6}$$

where $\boldsymbol{H}_{\boldsymbol{\pi}}$ and $\boldsymbol{H}_{\boldsymbol{\mu}}$ are the outputs of the two LSTM neural networks respectively.

Similar to the applications in matrix completion, we start from a random initialization $\boldsymbol{\mu}_0$ and $\boldsymbol{\pi}_0$. During our GSO procedure given in Algorithm 4, $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ are updated based on equations (5.4.6) for I and J steps in inner loops optimizations respectively. Meanwhile, we back-forward the leveraged global losses with respect to the negative log-likelihood of GMM, referring to $\mathcal{L}_{GMM} = \frac{1}{t_o}\sum_{t_s} \omega_{t_s} F(\boldsymbol{\pi}_{t_s}, \boldsymbol{\mu}_{t_s})$, for every u steps on the outer loops. In this way, the algorithm is updated based on the global scope knowledge about global losses across outer loops iterations.

Considering that the existing numerical gradient-based solutions typically perform less effectively and accurately than the EM algorithm [27], we focus on comparing our GSO method and the EM algorithm. EM algorithm converts maximization of the log-likelihood into maximization on its lower bound; hence it has closed-form formulations to implement an AM strategy on its M-step for variable updating. Nevertheless, our GSO method directly optimizes the cost function, i.e., the log-likelihood, and the variables are updated constantly up to the global scope knowledge extracted by LSTM networks. The main advantages of applying our GSO model on GMM are similar to the analysis in Section 5.4.1, which allow the optimizations to perform better when the overall landscapes are not good.

5.5 Experiments

In this section, we will present simulations on the aforementioned matrix completion and GMM problems to validate the effectiveness and efficiency of our GSO method.

In all the experimental, our meta-learners employ two-layer LSTM networks with 500 hidden units in each layer. For training processes, each network is trained by minimizing the accumulated loss functions according to (5.3.4) via truncating backpropagation through time (BPTT) [131] to update weights in LSTMs. As mentioned in section 5.2, we consider each non-convex optimization problem to be solved as on training sample, and the whole training process contains 100 samples. When training each sample, the ground truth is not given. The parameters are updated with respect to the loss of the objective functions in each simulation case. The ground truth is only used when testing the performance of learned GSO algorithm. The weights of LSTMs are updated by the ADAM [126]. In all simulations, the weights of LSTM networks are randomly initialized and are updated during the training process. Specifically, these weight are updated at outer loop and keep fixed at inner loop. The variable to be optimized in inner loop is also randomly initialized at the beginning of each inner loop. For evaluation, we fix the parameters of our GSO model and evaluate the performance on the testing datasets.

5.5.1 Numerical Results on Matrix Completion

In this subsection, we consider learning to optimize synthetic D-dimensional matrix completion problems. We take $D = 10$ and $D = 100$ to evaluate algorithms in both small and large scale cases. For each matrix completion problem, the ground truth matrix $\boldsymbol{R} \in \mathbb{R}^{D\times D}$ is randomly and synthetically generated with a certain rank. Meanwhile, the observation $\boldsymbol{R}_S = \mathcal{P}_\Omega(\boldsymbol{R})$ is generated based on matrix $\boldsymbol{R}$ by randomly setting certain percentage of entries in $\boldsymbol{R}$ to be zeros, and the non-zero fraction of entries is the observation rate. Matrix completion for $\boldsymbol{R}$ is then achieved by doing matrix factorization on $\boldsymbol{R}_S$ in the form of Equation (5.4.1) in Section 5.4.1. The two factorized low-dimensional matrices $\boldsymbol{U} \in \mathbb{R}^{D\times k_m}$ and $\boldsymbol{V}^{D\times k_m}$ are then used to generate reconstruction of the ground truth matrix, denoted by $\hat{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{V}^T$. Thus the evaluation

criterion is given by Relative Mean Square Error (RMSE) of

$$\text{RMSE} = \frac{\left\| \boldsymbol{R} - \hat{\boldsymbol{R}} \right\|_F}{\|\boldsymbol{R}\|_F}. \tag{5.5.1}$$

ALS and SGD methods have been adopted for comparisons. For them, the learning rates for their gradient descent algorithms are set to be 10^{-3}, and the weight of regularizers λ in Equation (5.4.1) is set to be 10^{-3}.

Different simulation scenarios on matrix completion are evaluated comprehensively, and we emphasise at first the following settings employed in all the scenarios: i) Each of the scenarios contains a set of 200 matrix completion problems, half of which are employed as training samples for parameter update on LSTMs, while the other 100 are used to evaluate the performance as testing samples. ii) Set $T = 100$ as total alternating steps for one problem. iii) The presented results are the averaged RMSE over 100 testing samples. iv) In all the training and testing processes, ground truth matrix $\boldsymbol{R}$ is not given, which is only used to evaluate the performance after the optimization processes.

Parameters setting for inner and outer loops

The numbers of variable update steps on inner loops and LSTM's parameters update steps on outer loops, (denoted by t_i and t_o respectively,) are the two most important hyper-parameters in all simulations. Different settings on t_i and t_o are thus tested at first to provide a brief guidance on the choices of t_i and t_o in the following evaluations.

Empirically, there is a trade-off between the performance and efficiency. Here we set t_i and t_o to vary from 1 to 20, respectively, with 20% observation rate, and there are thus $20 \times 20 = 400$ scenarios of rank-5 matrix completion problems in total. Each scenario has a set that contains 200 independent matrix completion problems, half of which is used to train the networks while the others are used for testing. The performance of

these 400 simulations are shown in Figure 5.3. It can be seen that for all the choices of t_o, the increase in t_i leads to significant improvements in the performance when $t_i \leq 10$. However, further increase in t_i provides little gain in the performances when $t_i \geq 10$. At the same time, the variations of t_o have less impact on RMSE, while larger choices ($t_o \geq 10$) bring better stability (there are less fluctuations when $t_i \geq 10$, $t_o \geq 10$). The performance variation in Figure 5.3 indicates that sufficient numbers of update steps for inner and outer loops play a significant role in these optimization processes. It can be seen that when either t_i or t_o is small (less than 5), the optimization does not perform well enough. A larger choice of t_i, however, typically brings higher computational cost. Thus in the rest of this chapter, we set $t_i = 10$ and $t_o = 10$ on default.

It is understandable that t_i directly determines the variable update times within each inner loop. When t_i is small, the learned updating rule for variables is assuming that variables have to be optimized in a few steps, which could be intractable in general. Meanwhile, the accumulated global loss are dependent on the value of t_o at outer loops, which indicates the length of the trajectory of global losses at outer loops. Therefore, a large enough t_o could provide a longer enough global losses trajectory for parameter update. According to the two-level meta-learning in our GSO model, enough update steps at inner and outer loops can ensure each level of meta-learning works well. We infer that small t_i may limit the first level meta-learning corresponding to inner loops unable to extract effective sub-problems across knowledge with merely few update steps, and small t_o could cause the second level meta-learning corresponding to outer loops become less stable due to the lack of updates on parameters.

Standard matrix completion

In this part, we compare the performance of our GSO method with two conventional methods, i.e., ALS and SGD, on matrix completion problems. We first test them on low-rank matrix completion and then apply them to high rank matrix completion tasks.

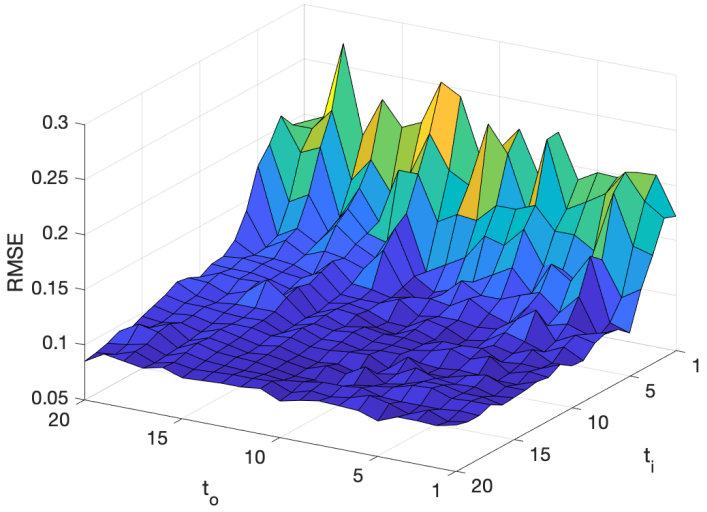


Figure 5.3: Performance of different choices on t_i and t_o ranging from 1 to 20.

In both simulations the rank knowledge is given, hence k_m is set to be equal to the rank of the matrices in the training and testing data set. We note that k_m is the lower dimension of two factorized low-dimensional matrices $\boldsymbol{U} \in \mathbb{R}^{D\times k_m}$ and $\boldsymbol{V}^{D\times k_m}$. We also emphasise that we run ALS and SGD more iterations than GSO does to ensure ALS and SGD are converged.

In Table 5.1, average RMSE of GSO methods and standard methods for four sets of 100-dimensional rank-5 matrix completion problems have been reported. Different sets have different observation rates, including 20%, 40%, 60% and 80%. From Table 5.1, it is clear that our GSO algorithm significantly outperforms the standard methods especially on high observation rate scenarios. It is also notable that when the observation rate is 20%, ALS cannot work well with 0.86 RMSE and SGD has 0.29 RMSE, while our GSO method achieves good reconstruction with RMSE $\leq$ 10%.

In Figure 5.4, we present the RMSE and loss curves of our GSO methods and standard methods averaged over 100 tested samples for 20% observation rate. Compared to the ALS and SGD methods, our GSO converges with the fastest speed and to the lowest RMSE as well.

Table 5.1: RMSE of rank-5 matrix completion on different observation rates

Methods \ Observed Rate	0.2	0.4	0.6	0.8
GSO	0.089	0.057	0.045	0.003
SGD	0.29	0.21	0.13	0.07
ALS	0.86	0.71	0.42	0.22

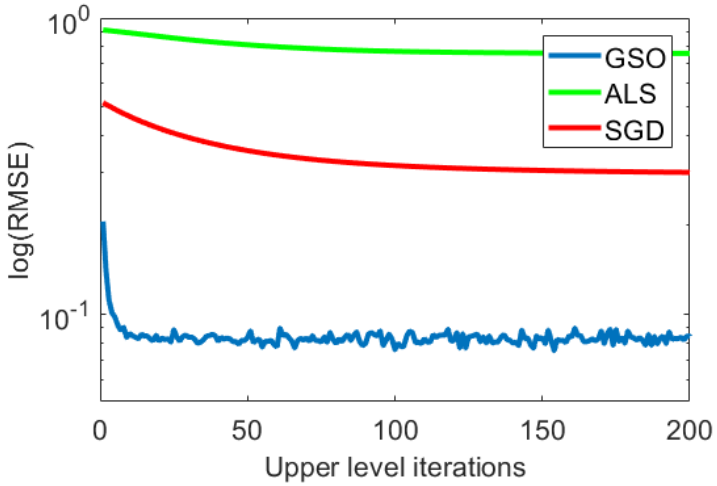


Figure 5.4: Averaged log(RMSE) Curves of three methods from 100 simulations in 20% observation rate.

Thus, we can conclude that in the classic low-rank matrix completion problem, our method has shown better numerical performances than standard methods. Especially in tough conditions where the dimension is high and no sufficient observations are known, our method achieves dramatic improvements over the standard methods, which typically fail.

Then, high-rank and full-rank matrix completion problems are taken into account. Generally, the matrix completion problem is assumed to be solved as a low-rank matrix completion problem. Here we are looking to solving high-rank or even full-rank matrix completion problems without adding any further assumptions. In this case, we test on the 20% observation rate, and apply our GSO method and two standard methods on six 100-dimensional sets of matrix completion problems whose ranks range from 5 to 100.

The averaged RMSE results are listed in Table 5.2. There is a clear trend of decreasing

Table 5.2: RMSE of matrix completion on different ranks

Methods \ Rank	5	10	20	40	80	100
GSO	0.089	0.085	0.078	0.072	0.053	0.052
SGD	0.29	0.40	0.68	1.33	NaN	NaN
ALS	0.86	0.94	1.57	5.56	NaN	NaN

Table 5.3: RMSE of rank-10 matrix completion on different choices of k_m

Methods \ k_m	10	20	40	80	100
GSO	0.092	0.12	0.19	0.30	0.36
SGD	0.31	0.78	1.35	1.65	NaN
ALS	0.89	1.04	1.22	5.21	NaN

performances of standard methods when the rank of matrices $\boldsymbol{R}$ increases. Noticeably, ALS is no longer workable when the rank is larger than 10, and SGD fails after the rank reaches 40. Meanwhile, our GSO method shows a significant difference on this relationship between the performance and rank: the higher the rank, the smaller the RMSEs.

Different from the traditional matrix completion methods, we can conclude that our algorithm is capable of solving the matrix completion problem with high-rank, even full-rank, without any extra constraints, while standard methods typically fail.

Blind matrix completion

In this part, we consider more challenging cases where the rank information is not known, as a result, the choice of k_m is not provided. In this case, the two factorized low-dimensional matrices $\boldsymbol{U} \in \mathbb{R}^{D\times k_m}$ and $\boldsymbol{V}^{D\times k_m}$ might have different lower dimension to the rank of the matrix $\boldsymbol{R}$. Therefore, the reconstruction will be much more difficult. We consider these are blind matrix completion problems which are typically intractable using previous methods. We firstly test our GSO model and standard methods in the case where a set of matrix completion problems have the same rank but not given. Then we expand our GSO model to a more difficult case which has a set of matrix completion problems with different unknown ranks.

In the first case, we test our GSO model and two standard methods on a set of rank-10 matrix completion problems with arbitrary k_m. In Table 5.3 we list the results of applying different k_m for reconstructing a rank-10 matrix through our GSO method and standard methods. It is clear that our GSO method shows robustness on a certain deviation from the true rank. When we take a large gap such as $k_m = 80$ or $k_m = 100$ to reconstruct a rank-10 matrix, our GSO method still has acceptable performances. Meanwhile, standard methods quickly degrade with the increase of the deviation between k_m and the rank value. As demonstrated in Figure 5.5, the RMSEs of ALS and SGD quickly rise to more than 100% when $k_m = 40$ and then sharply increase with larger k_m. In contrast, our GSO method shows a good tolerance on the increase of the variance between the real rank and k_m.

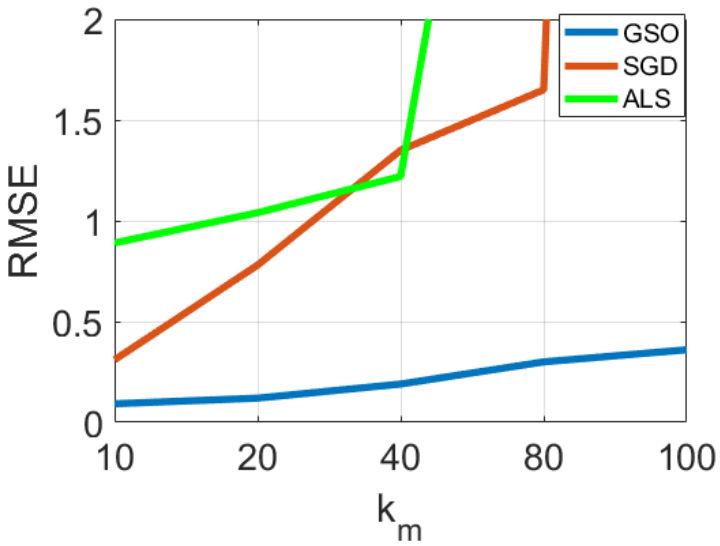


Figure 5.5: Averaged RMSE Curves of three methods from 100 simulations on different choices of k_m

Thus, we consider that our GSO method actually does not depend on the rank knowledge to achieve successful matrix completion. One possible reason is that the rank knowledge could be learned during the training process by GSO approach, consequently it is able to performs well without given rank information. It can be seen that GSO method can still perform well with a neighboring choice of k_m. However, in this case, standard methods can easily fail to converge.

Table 5.4: RMSEs of mixed matrix completion on different choice of k_m

k_m \ Rank	10	20	30	40	50	60	70	80	90	100
10	0.120	0.081	0.065	0.055	0.049	0.046	0.043	0.042	0.042	0.039
50	0.250	0.130	0.095	0.075	0.067	0.055	0.052	0.048	0.046	0.045
100	0.450	0.210	0.140	0.100	0.081	0.068	0.061	0.056	0.052	0.048

Then, we test our GSO method on matrix completion problems with different rank for the matrix $\boldsymbol{R}$ in the set. This means that in the training and testing sample sets, there are mixed matrices with different ranks. Thus the learned algorithm is required to adapt on matrix completion problems across different ranks.

In Table 5.4, RMSE results of three sets of matrix completion problems are presented, whose training and testing samples range from rank 10 to rank 100. For each set, 100 training samples and 100 testing samples are generated satisfying uniform distribution on rank 10 to 100, with a gap of 10. We take three different choices of k_m for each set, and present the mean RMSE on the sample of each rank respectively. The columns present the results of different choices on k_m and the rows show the results of different matrix completion problems whose matrices have different ranks.

On average, our GSO method in the three cases performs generally well on 10 types of samples. The comparison of the three cases further reveals that our method of choosing $k_m = 10$ achieves the best performance in all samples, especially on low-rank scenarios where it provides significantly better results than the others. For the other two choices of k_m, our method also has comparable performance on high-rank samples, and sufficiently good accuracy on the majority of the samples.

Therefore, we can see that our GSO method is essentially capable of solving a series of matrix completion problems with different ranks by even arbitrarily choosing k_m. This also verifies the meta-learning mechanism in our methods, which allows the learned strategy to generally perform well across different problems.

In summary, it has been shown that our GSO method is capable of solving the ma-

Table 5.5: High dimensional GMM simulation results

Method \ Dimension	4	8	16	32	64
EM	6.52	12.64	23.83	47.10	96.88
GSO	6.44	12.42	23.05	46.93	95.72

trix completion problem without any prior information. Typically, these non-convex optimization problems either have very bad landscape to get converge, or contains lots of local optimums. Therefore, it is hard for standard gradient descent based methods to perform well. Our GSO model makes it possible to find good solutions by incorporating two level meta-learning which essentially learns those landscapes from different problems to obtain an appropriate updating rule for variables.

5.5.2 Numerical Results on Gaussian Mixture Model

In this section, we apply our GSO method to the GMM problems. Given the data set $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^{G}$ (N denotes the number of observation samples), we optimize the mean cluster centre $\boldsymbol{\mu}_k$ and mixing proportion π_k whilst keeping the covariance $\boldsymbol{\Sigma}_k$ known, which is similar to the optimization of *k-means* problem. We consider one GMM problem (a set of $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k) as one sample for training or testing. For each training and testing sample, we set the maximum alternating steps as 100, and record the negative log-likelihood F_{ne} trajectory according to Equation (5.4.5).

EM algorithm has been adopted for the comparison. The stopping criterion for EM algorithm is $|F_{ne}(t-1) - F_{ne}(t)| < 10^{-4}$. EM algorithm and our GSO method all start from a same random initialization of $\boldsymbol{\mu}_k$ and π_k.

Similar to the settings in the matrix completion simulation, we also choose $t_i = 10$ and $t_o = 10$ for all the simulation scenarios in the GMM problems. Moreover, both of GSO method and EM algorithm are starting from the same random initialization points. The presented negative log-likelihoods are the averaged of 100 samples. In this chapter, we consider GMM problems with four clusters in different dimensions.

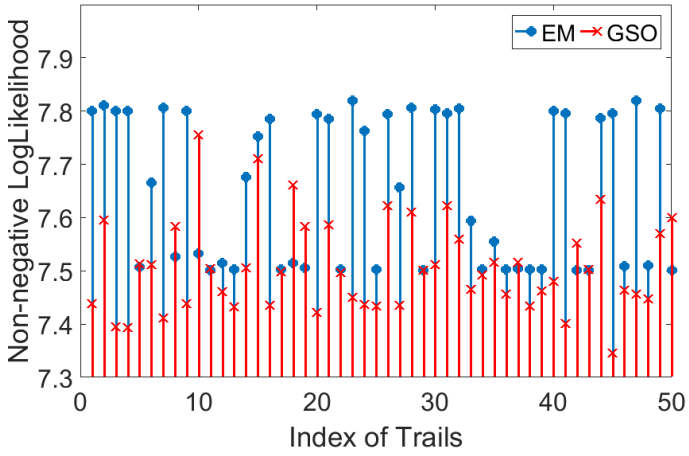
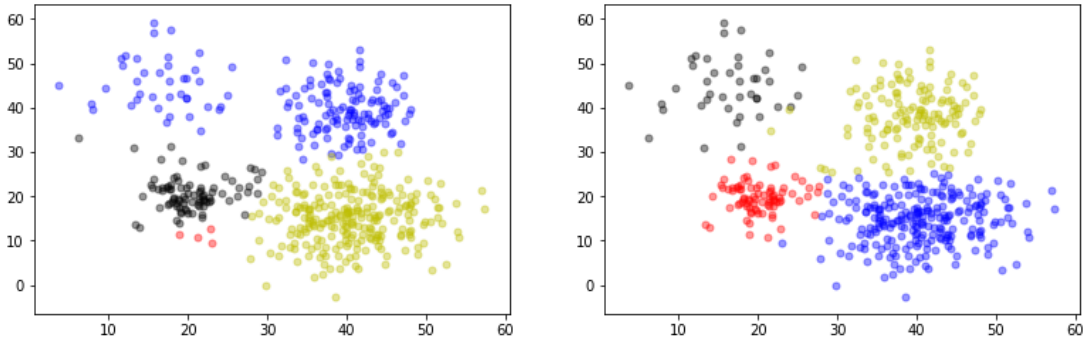


Figure 5.6: EM and GSO performance on 50 samples. Averaged Non-negative log-likelihood of 50 trails through EM algorithm is 7.77, while the result through GSO model is 7.56.



(a) EM clustering result.

(b) GSO clustering result.

Figure 5.7: Clustering results on one sample.

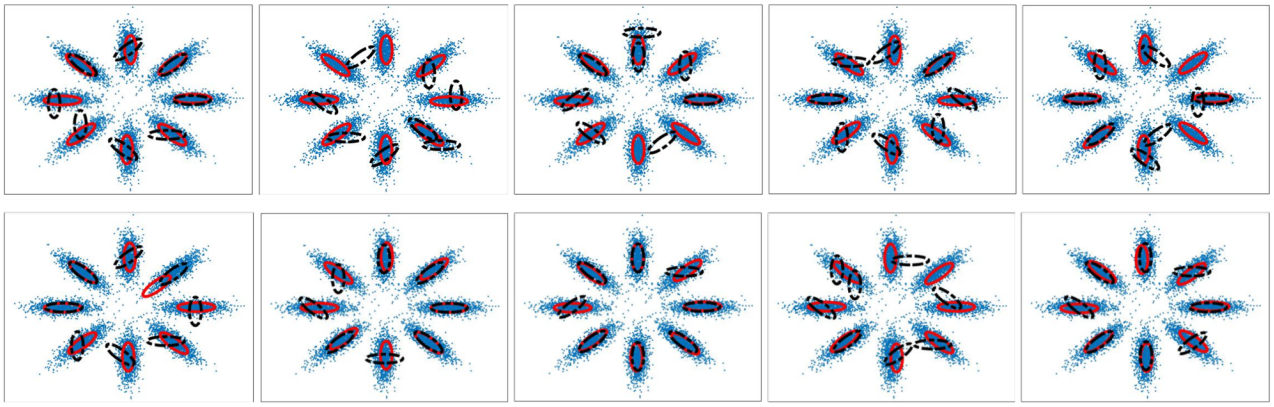


Figure 5.8: EM and GSO performance on 10 flower-shaped data. Black dash lines denote EM clustering results; Red solid lines denote GSO clustering results.

We first start from the 2-dimensional GMM problems, whose data vector $\boldsymbol{x}_g \in \mathbb{R}^{1\times 2}$. Given four clusters ($K = 4$), $G = 500$ data points in $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^{G}$, we show 50 tested results with random initialization in Figure 5.6, and also illustrate one of the clustering results in Figure 5.7. We only provide 500 data points here to increase the difficulty of optimizing these GMM problems, in which there are not sufficient observations. From Figure 5.6, it can be seen that on these 50 samples, the GSO method outperforms EM algorithm in most cases. The mean F_{ne} of GSO method on these 50 samples is 7.52 while that of the EM algorithm is 7.75. More specifically, we randomly select one clustering results among these 50 trails in Figure 5.7, which has clearly shown that the GSO method obtains much better clustering result than the EM algorithm on this sample with the same initialization points.

Besides, we further conduct the simulations on a flower-shaped synthetic data (N = 10,000) with random initializations. Each cluster in the flower-shaped data is composed by Gaussian-distributed samples. This is a typically hard problem as the anisotropic clusters lead to extensive local optimum. Ten representative examples can be found in Figure 5.8, in which our algorithm consistently achieves the optimal clustering but the results from EM are highly biased.

Therefore, our GSO method outperforms the EM algorithm for 2-Dimensional GMM problems with significantly improvements on accuracy, even for some challenging scenarios such as insufficient observation data samples and anisotropic clusters in these illustrative evaluations.

Then, we further evaluate our method in estimating high dimensional GMM problems. Several sets of high dimensional synthetic data ($G = 500$, $K = 4$) are also randomly generated for the evaluation, in terms of dimensions $4, 8, 16, 32$ and 64. The averaged negative log-likelihoods of 100 samples for each testing dimension are reported in Table 5.5.

Again, it is clear from Table 5.5 that our GSO method outperforms the EM algorithm

on all the high dimensional sample sets. We thus are able to conclude that, the variation on dimensions does not affect the performance of our method, while this typically deceases the performance of EM algorithm in general.

We conclude that our GSO model performs well on GMM problem and outperforms EM algorithm in both low and high dimensional cases. In this stage, it has been verified that our proposed model is able to solve non-convex multi-variable problem with non-linear mapping between variables and observation data. Even without closed-from landscapes in these problems, our GSO method still successfully find good solutions whilst EM algorithm is failed.

5.6 Summary

We have proposed a GSO model that aims to learn an algorithm for solving non-convex optimization problems on the intersection of multiple variable. The learned algorithm has been verified to have faster convergence and better performances than previous AM-based methods. To achieve that, our GSO model has employed LSTM-based meta-learners to build an interaction between variable updates and global loss. In this way, the variables are updated by the LSTM networks with fixed parameters at inner loops, which are then updated by minimizing accumulated global losses at outer loops. Two levels of meta-learning is processed in our GSO model: meta-learning across sub-problems within one training sample, and meta-learning across different training samples. Therefore, the learned algorithm is able to work on a set of optimization problems with different landscapes. The experimental results show that the proposed GSO model performs well for both matrix completion problem and GMM problem. It outperforms standard methods in both cases, and achieves good performance in some challenging cases where standard methods typically fail.

Chapter 6

Conclusion

6.1 Edification

In this thesis, two practical applications, several associated non-convex optimization problems, and the corresponding solutions are studied. The pain and gain during the four-year PhD studies have taught me that problems we face in real life are often non-smooth and non-convex, with very "harsh" constraints and difficult settings, yet it is still possible to converge to an optimal point. The key is never to give up at a local optimum even the gradient equals to zero. One must always search for new directions to update the variables including oneself, instead of getting stuck at the previous trajectory. In the following, the edifications from solving each part of the works are summarised briefly.

The OD flow estimation problem has been studied for decades. The traditional methods seek to directly optimize the OD flows and typically are formulated as convex optimization problems, even majority of which are ill-posed and rely on historical traffic flows data to ensure the convergence. We proposed a blind estimation model in non-convex form. However, for the first time, it makes an accuracy estimation for OD

flows without any prior information and historical data possible. It is understandable that non-convex problem is generally considered to be intractable and no convergence guarantee. But we infer that the non-convex model essentially provides more flexibility to discover structural constraints to allow the landscapes to become good enough for unique solution. Take the proposed OD flow estimation problem as an example, the proposed bi-linear inverse model increases the number of variables to be optimized, but the number of unknowns is reduced. Moreover, the incorporated O-flow and new traffic assignment matrix provide more related constraints, which dramatically improve the landscape of solving this bi-linear inverse problem and make the unique solution possible. At this stage, we consider that the non-convex optimization technique could provide immense flexibility to the model designer. Therefore, more constraints on variables and formulations of modelling are available, which might bring a better landscape to make the unique solution become possible. We consider that many practice problems might also receive great improvements by reformulating the convex model into a non-convex form.

In the studying of MKC problem, we dig into the correlation between different procedures of the alternating optimization process. The proposed LI-MKKM-MR approach essentially follows a three-step AM method, and it seeks to integrates the imputation procedure and clustering procedure. The LI-MKKM-MR algorithm reduces the redundancy and increases the diversity of the optimized kernel matrices by adding an extra penalty term during the imputation procedure, which improves the initialization states of matrices in clustering procedure as well. Utilize the imputation and clustering procedures better is essential to the MKC problem. In light of this, we infer that better performance could be achieved by a better utilization of different procedures in one AM optimization process. In both of OD estimation problem and MKC problem, we adopt an AM method to solve the non-convex problem. During the simulations, we observe that if we fix the maximum iteration steps of each procedure in an AM method, it could perform better in some cases. This phenomenon motivates us to dig

into the correlations between each procedure of an AM method as we discussed in section 2.1.2, and we prove that it is not always necessary to optimize each variable to the stationary point in the corresponding sub-problem. We consider that although the AM methods essentially follow a strategy which focuses on "local scope" optimization of each sub-problem. The correlations between the sub-problems and the overall problem is never effectively used in the optimization, which we consider as "global scope" information. In light of this, we proposed the GSO approach to build a "bridge" between the solution algorithm in sub-problems and the objective of the overall problem.

The works on GSO approach focus on solutions of the difficult non-convex optimization problems. In this thesis, we use meta-learning technique in a smart way (by alternatively optimizing each variable) to achieve the GSO strategy. The key concept is that the optimization algorithm in each sub-problem should be guided by the global scope information. We consider that this concept might also benefit the multi-task problem, such as the training of GAN. Moreover, the GSO model is not limited to the meta-learning technique. The fundamental mechanism is to build an interaction between the algorithms carried on the "local space" and the variation from the "global space", i.e., the optimization on sub-problem and the changes of global loss. It is possible to form a GSO model in an more explicit way, instead of using neural networks. Moreover, in this thesis, our GSO algorithm fixes the steps in inner loop and learns the variable update term. It will be interesting to fix the variable update term, i.e., vanilla gradient descent, and to learn the steps in inner loop.

This thesis proposes several interesting directions in applications and solutions of non-convex optimization problems. We would like to emphasise that non-convex optimization is widely used and quite effective in practice, although it gets less attentions than convex optimization in the last decades. We infer that a deep understanding of non-convex optimization will provide us immense freedom to design models and the corresponding solution algorithms. Especially with the development of machine

learning, the non-convex optimization will become more significant in modern society.

6.2 Future research

We conclude this thesis with some future research directions.

1. **Dynamic blind OD flows estimation** – In Chapter 3 we present a blind OD estimation approach for static transportation model, where the traffic assignment matrix is assumed to be fixed during the study time. In practical scenarios, real-time OD flows estimations are often not satisfied with a static model, and need to be formed as a dynamic model. The proposed forward model is also able to describe dynamic OD estimation model, however the increments of variables and sequential interactions between time incidents inevitably lead to NP-hardness of the optimization. For that purpose, it is essential to design a new algorithm for the bi-linear inverse problems of dynamic OD estimations that are non-convex and ill-posed. Therefore, as we presented in Chapter 5, it is a natural plan to apply GSO approach for solving dynamic OD flows estimation problems in the future works.

2. **MKC with global scope optimization** – We have improved localized incomplete MKC approach by adding matrix regularization to provide structural constraints to the non-convex non-linear objective function in Chapter 4. As discussed in Chapter 5, many structural constraints may not be universal in practice, thus the previous iterative algorithms would be no longer effective and accurate. Meanwhile, it would be very interesting to optimize multiple variables in MKC problems by our GSO approach proposed in Chapter 5, with respect to better interaction of imputation and clustering courses. This would enable us to further improve the performance of the LI-MKKM and apply it on other applications where the structural constraints are not given.

3. **Theoretical analysis and expanding applications on GSO method** – We introduce a new strategy for solving non-convex multi-variable optimization problems. Instead of exhaustive optimization, being cautious and adaptive to the intermediate process is more like human intelligence in learning and solving. Thus the intrinsic principle of our proposed approach is to provide a global scope optimization on separate variables where they can be optimized with extra prior knowledge of global loss variation to achieve an algorithm with adaptiveness and self-adjustment. It deserves to deeply dig into the principle analysis of GSO approach in terms of the convergence and other properties. This would allow us to deeply understand the proposed GSO approach and non-convex multi-variable optimization problems to improve the current works. In this thesis, we essentially take meta-learning in a smart way: alternating optimization. Directly using meta-learning on non-convex optimization is an interesting direction. Similar work has been proposed by [106], which directly use meta-learning for convex optimization problem. However, in that case, their contributions mainly focus on the speed of convergence of the meta-learning method. The performance of the method is not considered. Besides, this concept could be further expanded to more lager and challenging problems in iterative framework such as training Generative Adversarial Networks (GAN). It would be very interesting to generalize GAN model with composed meta-learning techniques in the form of our GSO approach to improve the efficiency and accuracy of GAN.

Bibliography

[1] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[2] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013.

[3] D. Gross, "Recovering low-rank matrices from few coefficients in any basis," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1548–1566, 2011.

[4] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[5] M. Hardt, "Understanding alternating minimization for matrix completion," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014, pp. 651–660.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, pp. 30–37, 2009.

[7] I. Tosic and P. Frossard, "Dictionary learning: What is the right representation for my signal?" *IEEE Signal Processing Magazine*, vol. 28, no. ARTICLE, pp. 27–38, 2011.

[8] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[9] Y. Zhang, Y. Lau, H.-w. Kuo, S. Cheung, A. Pasupathy, and J. Wright, "On the global geometry of sphere-constrained sparse blind deconvolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4894–4902.

[10] T. G. Stockham, T. M. Cannon, and R. B. Ingebretsen, "Blind deconvolution through digital signal processing," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 678–692, 1975.

[11] J. R. Hopgood and P. J. Rayner, "Blind single channel deconvolution using non-stationary signal processing," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 476–488, 2003.

[12] P. Jain, P. Kar *et al.*, "Non-convex optimization for machine learning," *Foundations and Trends® in Machine Learning*, vol. 10, no. 3-4, pp. 142–336, 2017.

[13] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[14] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries," in *International Conference on Machine Learning*, 2015, pp. 2463–2472.

[15] J. Fan and T. W. Chow, "Sparse subspace clustering for data with missing entries and high-rank matrix completion," *Neural Networks*, vol. 93, pp. 36–44, 2017.

[16] X. Lu, T. Gong, P. Yan, Y. Yuan, and X. Li, "Robust alternative minimization for matrix completion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 939–949, 2012.

[17] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[18] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International conference on algorithmic applications in management*. Springer, 2008, pp. 337–348.

[19] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, 2005.

[20] P. M. Kroonenberg and J. De Leeuw, "Principal component analysis of three-mode data by means of alternating least squares algorithms," *Psychometrika*, vol. 45, no. 1, pp. 69–97, 1980.

[21] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[22] S. Choudhary and U. Mitra, "On identifiability in bilinear inverse problems," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 4325–4329.

[23] S. Li, W. Ying, L. Jie, and X. Gao, "Multispectral image classification using a new bayesian approach with weighted markov random fields," in *CCF Chinese Conference on Computer Vision*, 2015.

[24] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Rejoinder: Latent variable graphical model selection via convex optimization," 2011.

[25] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, S. Li, and A. G. Constantinides, "Graph signal processing–part iii: Machine learning on graphs, from graph topology to applications," *arXiv preprint arXiv:2001.00426*, 2020.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[27] L. Xu and M. I. Jordan, "On convergence properties of the em algorithm for gaussian mixtures," *Neural computation*, vol. 8, no. 1, pp. 129–151, 1996.

[28] R. Hosseini and S. Sra, "Matrix manifold optimization for gaussian mixtures," in *Advances in Neural Information Processing Systems*, 2015, pp. 910–918.

[29] D. A. Reynolds, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, 2009.

[30] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the em algorithm," *SIAM review*, vol. 26, no. 2, pp. 195–239, 1984.

[31] C. Jin, Y. Zhang, S. Balakrishnan, M. J. Wainwright, and M. I. Jordan, "Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences," in *Advances in neural information processing systems*, 2016, pp. 4116–4124.

[32] C. L. Byrne, "Alternating minimization and alternating projection algorithms: A tutorial," *Sciences New York*, 2011.

[33] W. Ha and R. F. Barber, "Alternating minimization and alternating descent over nonconvex sets," *arXiv preprint arXiv:1709.04451*, 2017.

[34] U. Niesen, D. Shah, and G. Wornell, "Adaptive alternating minimization algorithms," in *2007 IEEE International Symposium on Information Theory*. IEEE, 2007, pp. 1641–1645.

[35] T. Sun, D. Li, H. Jiang, and Z. Quan, "Iteratively reweighted penalty alternating minimization methods with continuation for image deblurring," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3757–3761.

[36] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Mathematical Programming*, 2013.

[37] S. Guminov, P. Dvurechensky, and A. Gasnikov, "Accelerated alternating minimization," *arXiv preprint arXiv:1906.03622*, 2019.

[38] J. de Dios OrtÃozar and L. G. Willumsen, *Modelling transport*. John Wiley & Sons, 2011.

[39] T. Toledo, H. Koutsopoulos, A. Davol, M. Ben-Akiva, W. Burghout, I. Andréasson, T. Johansson, and C. Lundin, "Calibration and validation of microscopic traffic simulation tools: Stockholm case study," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1831, pp. 65–75, 2003.

[40] D. Bauer, G. Richter, J. Asamer, B. Heilmann, G. Lenz, and R. Kölbl, "Quasi-dynamic estimation of OD flows from traffic counts without prior OD matrix," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[41] F. Calabrese, M. Diao, G. Di Lorenzo, J. Ferreira, and C. Ratti, "Understanding individual mobility patterns from urban sensing data: A mobile phone trace example," *Transportation research part C: emerging technologies*, vol. 26, pp. 301–313, 2013.

[42] L. G. Willumsen, "Estimation of an od matrix from traffic counts–A review." 1978.

[43] E. Cascetta, "Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator," *Transportation Research Part B: Methodological*, vol. 18, no. 4, pp. 289–299, 1984.

[44] C. Antoniou, R. Balakrishna, and H. N. Koutsopoulos, "A synthesis of emerging data collection technologies and their impact on traffic management applications," *European Transport Research Review*, vol. 3, no. 3, pp. 139–148, 2011.

[45] L. Alexander, S. Jiang, M. Murga, and M. C. González, "Origin–destination trips by purpose and time of day inferred from mobile phone data," *Transportation research part c: emerging technologies*, vol. 58, pp. 240–250, 2015.

[46] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Time-evolving OD matrix estimation using high-speed gps data streams," *Expert systems with Applications*, vol. 44, pp. 275–288, 2016.

[47] H. Lo, N. Zhang, and W. Lam, "Decomposition algorithm for statistical estimation of OD matrix with random link choice proportions from traffic counts," *Transportation Research Part B: Methodological*, vol. 33, no. 5, pp. 369–385, 1999.

[48] H. Lo, N. Zhang, and W. H. Lam, "Estimation of an origin-destination matrix with random link choice proportions: A statistical approach," *Transportation Research Part B: Methodological*, vol. 30, no. 4, pp. 309–324, 1996.

[49] R. Frederix, F. Viti, and C. M. Tampere, "Dynamic origin–destination estimation in congested networks: Theoretical findings and implications in practice," *Transportmetrica A: Transport Science*, vol. 9, no. 6, pp. 494–513, 2013.

[50] J. Cao, S. V. Wiel, B. Yu, and Z. Zhu, "A scalable method for estimating network traffic matrices from link counts," *Preprint. Available at http://stat-www. berkeley. edu/binyu/publications. html*, 2000.

[51] L. Willumsen, "Simplified transport models based on traffic counts," *Transportation*, vol. 10, no. 3, pp. 257–278, 1981.

[52] M. Bierlaire and F. Crittin, "An efficient algorithm for real-time estimation and prediction of dynamic OD tables," *Operations Research*, vol. 52, no. 1, pp. 116–127, 2004.

[53] M. L. Hazelton, "Inference for origin–destination matrices: Estimation, prediction and reconstruction," *Transportation Research Part B: Methodological*, vol. 35, no. 7, pp. 667–676, 2001.

[54] C. Xie, K. Kockelman, and S. Waller, "Maximum entropy method for subnetwork origin-destination trip matrix estimation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2196, pp. 111–119, 2010.

[55] M. G. Bell, "The estimation of origin-destination matrices by constrained generalised least squares," *Transportation Research Part B: Methodological*, vol. 25, no. 1, pp. 13–22, 1991.

[56] A. K. Menon, C. Cai, W. Wang, T. Wen, and F. Chen, "Fine-grained OD estimation with automated zoning and sparsity regularisation," *Transportation Research Part B: Methodological*, vol. 80, pp. 150–172, 2015.

[57] H. Bar-Gera, "Traffic assignment by paired alternative segments," *Transportation Research Part B: Methodological*, vol. 44, no. 8-9, pp. 1022–1046, 2010.

[58] C.-C. Lu, X. Zhou, and K. Zhang, "Dynamic origin–destination demand flow estimation under congested traffic conditions," *Transportation Research Part C: Emerging Technologies*, vol. 34, pp. 16–37, 2013.

[59] A. Tympakianaki, H. N. Koutsopoulos, and E. Jenelius, "Robust spsa algorithms for dynamic OD matrix estimation," in *9th International Conference on Ambient Systems, Networks and Technologies May 8-11, 2018, Porto, Portugal*, 2018.

[60] X. Yang, Y. Lu, and W. Hao, "Origin-destination estimation using probe vehicle trajectory and link counts," *Journal of Advanced Transportation*, vol. 2017, 2017.

[61] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1. ACM, 2003, pp. 206–217.

[62] T. Toledo and T. Kolechkina, "Estimation of dynamic origin–destination matrices using linear assignment matrix approximations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 618–626, 2013.

[63] G. Cantelmo, E. Cipriani, A. Gemma, and M. Nigro, "An adaptive bi-level gradient procedure for the estimation of dynamic traffic demand," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1348–1361, 2014.

[64] E. Cipriani, M. Florian, M. Mahut, and M. Nigro, "A gradient approximation approach for adjusting temporal origin–destination matrices," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 2, pp. 270–282, 2011.

[65] P. Jin, M. Cebelak, F. Yang, J. Zhang, C. Walton, and B. Ran, "Location-based social networking data: Exploration into use of doubly constrained gravity model for origin-destination estimation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2430, pp. 72–82, 2014.

[66] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 557–573, 1998.

[67] K. Parry and M. L. Hazelton, "Estimation of origin–destination matrices from link counts and sporadic routing data," *Transportation Research Part B: Methodological*, vol. 46, no. 1, pp. 175–188, 2012.

[68] A. A. Prakash, R. Seshadri, C. Antoniou, F. C. Pereira, and M. E. Ben-Akiva, "Reducing the dimension of online calibration in dynamic traffic assignment systems," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2667, pp. 96–107, 2017.

[69] T. Djukic, G. Flötteröd, H. Van Lint, and S. Hoogendoorn, "Efficient real time od matrix estimation based on principal component analysis," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 115–121.

[70] D. Jiang, Z. Xu, Z. Chen, Y. Han, and H. Xu, "Joint time–frequency sparse estimation of large-scale network traffic," *Computer Networks*, vol. 55, no. 15, pp. 3533–3547, 2011.

[71] R. Balakrishna and H. Koutsopoulos, "Incorporating within-day transitions in simultaneous offline estimation of dynamic origin-destination flows without assignment matrices," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2085, pp. 31–38, 2008.

[72] K. Ashok and M. E. Ben-Akiva, "Alternative approaches for real-time estimation and prediction of time-dependent origin–destination flows," *Transportation Science*, vol. 34, no. 1, pp. 21–36, 2000.

[73] H. Zhang, R. Seshadri, A. A. Prakash, F. C. Pereira, C. Antoniou, and M. E. Ben-Akiva, "Improved calibration method for dynamic traffic assignment models: Constrained extended kalman filter," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2667, pp. 142–153, 2017.

[74] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, 2017.

[75] W. Shen and L. Wynter, "A new one-level convex optimization approach for estimating origin–destination demand," *Transportation Research Part B: Methodological*, vol. 46, no. 10, pp. 1535–1555, 2012.

[76] K. Zhan, X. Chang, J. Guan, L. Chen, Z. Ma, and Y. Yang, "Adaptive structure discovery for multimedia analysis using multiple features," *IEEE Trans. Cybernetics*, vol. 49, no. 5, pp. 1826–1834, 2019.

[77] K. Zhan, F. Nie, J. Wang, and Y. Yang, "Multiview consensus graph clustering," *IEEE Trans. Image Processing*, vol. 28, no. 3, pp. 1261–1270, 2019.

[78] D. Huang, J. Lai, and C. Wang, "Robust ensemble clustering using probability trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1312–1326, 2016.

[79] C. Wang, J. Lai, and P. S. Yu, "Multi-view clustering based on belief propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1007–1021, 2016.

[80] M. Yin, J. Gao, S. Xie, and Y. Guo, "Multiview subspace clustering via tensorial t-product representation," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 30, no. 3, pp. 851–864, 2019.

[81] M. Yin, S. Xie, Z. Wu, Y. Zhang, and J. Gao, "Subspace clustering via learning an adaptive low-rank graph," *IEEE Trans. Image Processing*, vol. 27, no. 8, pp. 3716–3728, 2018.

[82] K. Zhan, C. Niu, C. Chen, F. Nie, C. Zhang, and Y. Yang, "Graph structure fusion for multiview clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 10, pp. 1984–1993, 2018.

[83] B. Zhao, J. T. Kwok, and C. Zhang, "Multiple kernel clustering," in *SDM*, 2009, pp. 638–649.

[84] S. Yu, L.-C. Tranchevent, X. Liu, W. Glänzel, J. A. K. Suykens, B. D. Moor, and Y. Moreau, "Optimized data fusion for kernel k-means clustering," *IEEE TPAMI*, vol. 34, no. 5, pp. 1031–1039, 2012.

[85] S. Li, Y. Jiang, and Z. Zhou, "Partial multi-view clustering," in *AAAI*, 2014, pp. 1968–1974.

[86] M. Gönen and A. A. Margolin, "Localized data fusion for kernel k-means clustering with application to cancer biology," in *NIPS*, 2014, pp. 1305–1313.

[87] L. Du, P. Zhou, L. Shi, H. Wang, M. Fan, W. Wang, and Y.-D. Shen, "Robust multiple kernel k-means clustering using ℓ_{21}-norm." in *IJCAI*, 2015, pp. 3476–3482.

[88] X. Liu, Y. Dou, J. Yin, L. Wang, and E. Zhu, "Multiple kernel k-means clustering with matrix-induced regularization," in *AAAI*, 2016, pp. 1888–1894.

[89] D. Huang, C. Wang, and J. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybernetics*, vol. 48, no. 5, pp. 1460–1473, 2018.

[90] M. Li, X. Liu, L. Wang, Y. Dou, J. Yin, and E. Zhu, "Multiple kernel clustering with local kernel alignment maximization," in *IJCAI*, 2016, pp. 1704–1710.

[91] X. Liu, S. Zhou, Y. Wang, M. Li, Y. Dou, E. Zhu, and J. Yin, "Optimal neighborhood kernel clustering with multiple kernels," in *AAAI*, 2017, pp. 2266–2272.

[92] M. Yin, J. Gao, S. Xie, and Y. Guo, "Multiview subspace clustering via tensorial t-product representation," *IEEE TNNLS*, no. 99, pp. 1–14, 2018.

[93] Q. Wang, Z. Qin, F. Nie, and X. Li, "Spectral embedded adaptive neighbors clustering," *IEEE TNNLS*, no. 99, pp. 1–7, 2018.

[94] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE TNNLS*, no. 99, pp. 1–10, 2018.

[95] S. Xiang, L. Yuan, W. Fan, Y. Wang, P. M. Thompson, and J. Ye, "Multi-source learning with block-wise missing data for alzheimer's disease prediction," in *ACM SIGKDD*, 2013, pp. 185–193.

[96] R. Kumar, T. Chen, M. Hardt, D. Beymer, K. Brannon, and T. F. Syeda-Mahmood, "Multiple kernel completion and its application to cardiac disease discrimination," in *ISBI*, 2013, pp. 764–767.

[97] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *NIPS*, 1993, pp. 120–127.

[98] A. Trivedi, P. Rai, H. Daumé III, and S. L. DuVall, "Multiview clustering with incomplete views," in *NIPS 2010: Machine Learning for Social Computing Workshop ,Whistler, Canada*, 2010.

[99] C. Xu, D. Tao, and C. Xu, "Multi-view learning with incomplete views," *IEEE Trans. Image Processing*, vol. 24, no. 12, pp. 5812–5825, 2015.

[100] W. Shao, L. He, and P. S. Yu, "Multiple incomplete views clustering via weighted nonnegative matrix factorization with $\ell_{2,1}$ regularization," in *ECML PKDD*, 2015, pp. 318–334.

[101] S. Bhadra, S. Kaski, and J. Rousu, "Multi-view kernel completion," in *arXiv:1602.02518*, 2016.

[102] X. Liu, M. Li, L. Wang, Y. Dou, J. Yin, and E. Zhu, "Multiple kernel k-means with incomplete kernels," in *AAAI*, 2017, pp. 2259–2265.

[103] X. Zhu, X. Liu, M. Li, E. Zhu, L. Liu, Z. Cai, J. Yin, and W. Gao, "Localized incomplete multiple kernel k-means," in *IJCAI*, 2018, pp. 3271–3277.

[104] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao, "Multiple kernel k-means with incomplete kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2019.

[105] S. Jegelka, A. Gretton, B. Schölkopf, B. K. Sriperumbudur, and U. von Luxburg, "Generalized clustering via kernel embeddings," in *KI 2009: Advances in Artificial Intelligence, 32nd Annual German Conference on AI*, 2009, pp. 144–152.

[106] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in neural information processing systems*, 2016, pp. 3981–3989.

[107] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," *arXiv preprint arXiv:1710.11622*, 2017.

[108] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[109] J. Schmidhuber, "Evolutionary principles in self-referential learning," *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, vol. 1, p. 2, 1987.

[110] ——, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Computation*, vol. 4, no. 1, pp. 131–139, 1992.

[111] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[112] K. Li and J. Malik, "Learning to optimize neural nets," *arXiv preprint arXiv:1703.00441*, 2017.

[113] M. Bierlaire, "The total demand scale: a new measure of quality for static and dynamic origin-destination trip tables," *Transportation Research Part B: Methodological*, vol. 36, no. 9, pp. 837 – 850, 2002.

[114] M. A. Davenport and J. Romberg, "An overview of low-rank matrix recovery from incomplete observations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 608–622, 2016.

[115] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[116] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 6, pp. 797–829, 2006.

[117] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[118] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *JMLR*, vol. 13, pp. 795–828, 2012.

[119] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao, "Multiple kernel k-means with incomplete kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[120] A. Maurer and M. Pontil, "k-dimensional coding schemes in Hilbert spaces," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5839–5846, 2010.

[121] T. Liu, D. Tao, and D. Xu, "Dimensionality-dependent generalization bounds for k-dimensional coding schemes," *Neural computation*, vol. 28, no. 10, pp. 2213–2249, 2016.

[122] H. Zhao, H. Liu, and Y. Fu, "Incomplete multimodal visual data grouping," in *IJCAI*, 2016, pp. 2392–2398.

[123] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "From ensemble clustering to multi-view clustering," in *IJCAI*, 2017, pp. 2843–2849.

[124] W. Zhou, H. Wang, and Y. Yang, "Consensus graph learning for incomplete multi-view clustering," in *Advances in Knowledge Discovery and Data Mining*, 2019, pp. 529–540.

[125] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[126] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[127] P. D. O'grady, B. A. Pearlmutter, and S. T. Rickard, "Survey of sparse and non-sparse methods in source separation," *International Journal of Imaging Systems and Technology*, vol. 15, no. 1, pp. 18–33, 2005.

[128] Q. Yu, W. Dai, Z. Cvetkovic, and J. Zhu, "Bilinear dictionary update via linear least squares," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7923–7927.

[129] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *Advances in neural information processing systems*, 2004, pp. 1141–1148.

[130] G. J. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[131] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.