Adaptation and learning over multitask networks and graphs

Fei Hua

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

THÈSE DE DOCTORAT

Adaptation et apprentissage sur des réseaux et des graphiques multitâches

Fei HUA

Laboratoire J.-L. LAGRANGE

Présentée en vue de l'obtention
du grade de docteur en Science pour l'ingénieur
d'Université Côte d'Azur
et de Northewestern Polytechnical University
Dirigée par :Cédric Richard
Haiyan Wang
Soutenue le :15/07/2020

Devant le jury, composé de :
David Brie, PU, Université de Lorraine
Paul Honeine, PU, Université de Rouen
Jie Chen, PU, Northwestern Polytechnical University
Jingdong Chen PU, Northwestern Polytechnical University
Yuantao Gu, PU, Tsinghua University
André Ferrari, PU, Université Côte d'Azur

Adaptation and Learning over Multitask Networks and Graphs

Jury :

Rapporteurs

David Brie, PU, Université de Lorraine, France

Paul Honeine, PU, Grenoble-INP, France

Examinateurs

Jie Chen, PU, Northwestern Polytechnical University, China

Jingdong Chen, PU, Northwestern Polytechnical University, China

Yuantao Gu, PU, Tsinghua University, China

André Ferrari, PU, Université côte d'Azur, France

Directeurs

Cédric Richard, PU, Université côte d'Azur, France

Haiyan Wang, PU, Shaanxi University of Science and Technology, China

Adaptation and Learning over Multitask Networks and Graphs

Abstract: Multitask learning has received considerable attention in signal processing and machine learning communities. It aims at simultaneously learning several related tasks other than the traditional single-task problems. There also have witnessed a wide spectrum of data processing problems that are network- or graph-structured and require adaptation ability to streaming data and time-varying dynamics. Distributed adaptive learning strategies over networks enable a collection of interconnected agents to accomplish a certain task, such as parameter estimation, collaboratively through local computation and cooperation among neighboring agents. Further, they endow the agents with continuous adaptation and learning ability to track possible drifts in the underlying model. Despite the heterogeneous nature and the fact that each agent may solve a different task in multitask network, it could still benefit from a collaboration between agents to improve the estimation accuracy by leveraging the relations and capitalizing on inductive transfer between them. The objective of this thesis is to devise and analyze multitask adaptive learning strategies over networks and graphs. First, we consider multitask estimation problems where each agent is interested in estimating its own parameter vector and where the parameter vectors at neighboring agents are related linearly according to a set of constraints. Based on the penalty method, an unconstrained optimization problem is reformulated and a distributed algorithm is derived. The behavior of the algorithm in the mean and in the mean-square-error sense is analyzed. Next, we relax the local constraints assumption and consider the multitask problem with non-local constraints. We devise the distributed algorithm by employing a multi-hop relay protocol across the agents. We prove that the algorithm will continue to converge and provide theoretical performance analysis. In the third part, we extend the distributed learning strategies to the emerging graph signal processing applications where the signal itself is network-structured. Several graph diffusion LMS strategies are proposed to cope with streaming graph signals. We also extend the multitask model to graph filters and propose an on-line clustering mechanism. Last, we consider the problem of modeling graph signals by using a combination of multiple graph filters. An efficient algorithm is proposed to simultaneously learn coefficients of multiple graph filters and perform model selection. Simulation and numerical results are provided to illustrate the effectiveness of all proposed algorithms and validate the theoretical analyses.

Keywords: Multitask networks, distributed estimation, adaptive algorithms, Diffusion LMS, constraints, performance analysis, graph signal processing, graph filter, clustering, parallel graph filters, model selection.

Adaptation et apprentissage sur des réseaux et des graphiques multitâches

Résumé: L'apprentissage multitâche a reçu une attention considérable dans les communautés de traitement du signal et d'apprentissage automatique. Au contraire du traitement traditionnel des problèmes à tâche unique, il vise à apprendre d'une façon simultanée plusieurs tâches connexes. Il y a également eu un large éventail de problèmes de traitement de données qui sont structurés en réseau ou en graphiques et qui nécessitent une capacité d'adaptation à la transmission de données en continu et à des dynamiques variant dans le temps. Les stratégies d'apprentissage adaptatif réparties sur les réseaux permettent à une collection d'agents interconnectés d'accomplir une certaine tâche, telle que l'estimation des paramètres, en collaboration grâce au calcul local et à la coopération entre les agents voisins. De plus, ils confèrent aux agents une capacité d'adaptation et d'apprentissage continue pour suivre les dérives possibles dans le modèle sous-jacent. Malgré la nature hétérogène et le fait que chaque agent peut résoudre une tâche différente dans un réseau multitâche, il pourrait encore bénéficier d'une collaboration entre les agents pour améliorer la précision de l'estimation en tirant parti des relations et en capitalisant sur le transfert inductif entre eux. L'objectif de cette thèse est de concevoir et d'analyser des stratégies d'apprentissage adaptatif multitâche sur des réseaux et des graphiques. Premièrement, nous abordons les problèmes d'estimation multitâche où chaque agent est intéressé à estimer son propre vecteur de paramètres et où les vecteurs de paramètres aux agents voisins sont liés linéairement selon un ensemble de contraintes. Sur la base de la méthode des pénalités, un problème d'optimisation non contraint est reformulé et un algorithme distribué est dérivé. Le comportement de l'algorithme dans la moyenne et dans le sens de l'erreur quadratique moyenne est analysé. Ensuite, nous assouplissons l'hypothèse des contraintes locales et travaillons sur le problème multitâche avec des contraintes non locales. Nous concevons l'algorithme distribué en utilisant un protocole de relais à sauts multiples entre les agents. Nous prouvons que l'algorithme continuera de converger et fournira une analyse théorique des performances. Dans la troisième partie, nous étendons les stratégies d'apprentissage distribué aux applications émergentes de traitement du signal graphique où le signal lui-même est structuré en réseau. Plusieurs stratégies LMS de diffusion de graphe sont proposées pour faire face aux signaux de graphe en streaming. Nous étendons également le modèle multitâche aux filtres graphiques et proposons un mécanisme de clustering en ligne. Enfin, nous nous penchons sur le problème de la modélisation des signaux graphiques en utilisant une combinaison de plusieurs filtres graphiques. Un algorithme efficace est proposé pour apprendre simultanément les coefficients de plusieurs filtres graphiques et effectuer la sélection du modèle. Des résultats de simulation et numériques sont fournis pour illustrer l'efficacité de tous les algorithmes proposés et valider les analyses théoriques.

Mots clés: Réseaux multitâches, estimation distribuée, algorithmes adaptatifs, LMS de diffusion, contraintes, analyse des performances, traitement du signal graphique, filtre graphique, clustering, filtres graphiques parallèles, sélection de modèle.

Dedicated to my family

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Cédric Richard for providing me an opportunity to study at Nice. Despite his own active research schedule, he has given me sufficient daily guidance. His enthusiasm, knowledge, discipline and high standard have inspired me to work on the right track. I have also been encouraged, supported and tolerated by him. I am grateful to him for introducing me the fascinating network topics and helping me build a collaboration network. I am also indebted to my co-advisor Prof. Haiyan Wang for his continuous support and trust. I have learnt a lot from his wisdom and advice. I am grateful for his generous financial support when I was at Northwestern Polytechnical University (NPU). It is my great honor to work with them during this journey.

I would like to thank Prof. David Brie and Prof. Paul Honeine for kindly reviewing the manuscript and providing valuable comments and suggestions. I also appreciate Prof. André Ferrari, Prof. Jingdong Chen, Prof. Yuantao Gu, and Prof. Jie Chen accepting to be members of my thesis committee. Thanks for your time and efforts spending on my thesis.

During my PhD studies, I am fortunate to work with some great minds. I would like to thank Dr. Roula Nassif who was always there to lend me a hand. I appreciate her supports in my research work. I would like to express my gratitude to Prof. Ali H. Sayed from the Ecole Polytechnique Fédérale de Lausanne. His useful feedback, critical comments, and thorough reading have greatly improved the quality of my work. I am much grateful to Prof. Pierre Borgnat, Prof. Paulo Gonçalves from the Université de Lyon, Prof. Jie Chen, Prof. Xiaohong Shen from the NPU, and Dr. Yongsheng Yan from the Nanyang Technological University for their contributions in my research work. In particular, I would like to thank Prof. Xiaohong Shen for her mentorship during my Masters studies.

I would like to thank other members of the Lagrange Laboratory for their kindness: André Ferrari, Céline Theys, Rémi Flamary, Henri Lantéri, and Mamadou N'Diaye. I also would like to thank my lab-mates Roula, Rita, Khalil, Ikram, Mircea, Elisson, and Jun for your friendships. In addition, I am grateful to Ikram and Mircea for helping me to deal with the daily routine when my French level was not adequate. I am also lucky to have met some smart and interesting minds from other labs: Jiqiang, Zhiyan, Zhaoqiang, Shaoqin, Chenmin, Jiao, Jie, Jiaxing, Jing, Furong, Xi, Pengfei. My thanks also go to Delphine Saissi, Jocelyne Bettini, and Jérôme Mifsud for providing me with all possible convenience in administrative affairs.

I am grateful to Prof. Jianguo Huang, Prof. Chengbing He, and Dr. Wei Gao from the NPU who encouraged me to study abroad. Dr. Gao has also provided me many useful suggestions that helped me to smoothly adapt to French lives. I would like to thank my

lab-mates at NPU: Jingjie, Weigang, Zhichen, Xiaobo, Haiyang, Wei, Yong, Muhang, Lei, Haodi, and Tianyi. I also thank my friends Penghua, Jiangjian, Shuangquan, Bing, Hao, Wei, Xin, Hang, Tao, Heting, Chao, and Lei for their occasionally greetings and chats when I was at Nice.

I also acknowledge the financial support from the China Scholarship Council, and all the support from the Lagrange Laboratory.

Last, I am deeply indebted to my parents for their endless love and support. I could not have completed this without your encouragement. I would like to thank my brother for his support.

Contents

List of figures

List of notations

General notation and symbols

$\mathbb{R}$	Field of real numbers.
$\mathbb{E}$	Expected value operator.
$\mathbf{0}$	Vectors or matrices containing all zero entries.
$\mathbf{1}_M$	$M \times 1$ column vectors with all its entries equal to one.
$\boldsymbol{e}_k$	Column vector with a unit entry at position k and zeros elsewhere.
$\boldsymbol{I}_M$	Identity matrix of size $M \times M$.
a	Normal font letters denote scalars.
$\boldsymbol{a}$	Boldface lowercase letters denote column vectors.
$\boldsymbol{A}$	Boldface uppercase letters denote matrices.
$\mathcal{A}$	Calligraphy normal font uppercase letters denote sets.
$\boldsymbol{\mathcal{A}}$	Calligraphy bold font uppercase letters denote block matrices.
$[\boldsymbol{a}]_k$ or a_k	k-th entry of vector $\boldsymbol{a}$.
$[\boldsymbol{A}]_{k,\ell}$ or $a_{k\ell}$	(k,ℓ)-th entry of matrix $\boldsymbol{A}$.
$[\boldsymbol{A}]_{k,\bullet}$	k-th row of matrix $\boldsymbol{A}$.
$[\boldsymbol{A}]_{\bullet,\ell}$	ℓ-th column of matrix $\boldsymbol{A}$.
$(\cdot)^\top$	Transpose of matrix or vector.
$\boldsymbol{A}^{-1}$	Inverse of matrix $\boldsymbol{A}$.
$\boldsymbol{A}^\dagger$	Pseudo-inverse of matrix $\boldsymbol{A}$
$\mathrm{Tr}(\boldsymbol{A})$	Trace of matrix $\boldsymbol{A}$.
$\mathrm{col}\{a,b\}$	Column vector with entries a and b.
$\mathrm{col}\{\boldsymbol{a},\boldsymbol{b}\}$	Block column vector with entries $\boldsymbol{a}$ and $\boldsymbol{b}$.
$\mathrm{vec}(\boldsymbol{A})$	Vector obtained by stacking the columns of matrix $\boldsymbol{A}$.
$\mathrm{bvec}(\boldsymbol{\mathcal{A}})$	Vector obtained by vectorizing and stacking blocks of $\boldsymbol{A}$.
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

$\boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{B}}$	Block Kronecker product of matrices $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$.
$\mathrm{diag}(\boldsymbol{a})$	Diagonal matrix containing the vector $\boldsymbol{a}$ along its main diagonal.
$\mathrm{diag}(\boldsymbol{A})$	Vector stores the diagonal entries of $\boldsymbol{A}$.
$\mathrm{bdiag}\{\boldsymbol{A}, \boldsymbol{B}\}$	Block diagonal matrix with block entry $\boldsymbol{B}$ placed immediately below and to the right of its predecessor $\boldsymbol{A}$.
$\lvert a \rvert$	Absolute value of scalar a.
$\lVert \boldsymbol{a} \rVert$ or $\lVert \boldsymbol{a} \rVert_2$	Euclidean norm of vector $\boldsymbol{a}$.
$\lVert \boldsymbol{a} \rVert_{\boldsymbol{\Sigma}}^2$ or $\lVert \boldsymbol{a} \rVert_{\boldsymbol{\sigma}}^2$	Weighted square value $\boldsymbol{a}^\top \boldsymbol{\Sigma} \boldsymbol{a}$.
$\lVert \boldsymbol{A} \rVert$ or $\lVert \boldsymbol{A} \rVert_2$	Spectral norm of matrix $\boldsymbol{A}$.
$\lVert \boldsymbol{A} \rVert_\infty$	Maximum absolute row sum of matrix $\boldsymbol{A}$.
$\lVert \boldsymbol{\mathcal{A}} \rVert_{b,\infty}$	Block maximum norm of block matrix $\boldsymbol{\mathcal{A}}$.
$\boldsymbol{A} \succ 0$	Matrix $\boldsymbol{A}$ is positive-definite.
$\rho(\boldsymbol{A})$	Spectral radius of matrix $\boldsymbol{A}$.
$\lambda(\boldsymbol{A})$	Eigenvalues of matrix $\boldsymbol{A}$.
$\lambda_{\min}(\boldsymbol{A})$	Minimum eigenvalue of matrix $\boldsymbol{A}$.
$\lambda_{\max}(\boldsymbol{A})$	Maximum eigenvalue of matrix $\boldsymbol{A}$.
$\lvert \mathcal{N} \rvert$	Cardinality of set $\mathcal{N}$.
δ_{ij}	Kronecker delta function: $\delta_{ij} = 1$ if $i = j$, and zero otherwise.
$\max(a, b)$	Larger value in a and b.

List of abbreviations

Abbreviations

i.i.d.	independent and identically distributed
s.t.	subject to
w.r.t.	with respect to
ADMM	Alternating Direction Method of Multipliers
AR	Autoregressive
ARMA	Autoregressive Moving Average
APA	Affine Projection Algorithm
ATC	Adapt-then-Combine
BIBO	Bounded Input Bounded Output
CLMS	Constrained Least-Mean-Squares
CTA	Combine-then-Adapt
EMSE	Excess Mean Squared Error
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
GFT	Graph Fourier Transform
GSP	Graph Signal Processing
GSO	Graph Shift Operator
KKF	Kernel Kalman Filter
KRR	Kernel Ridge Regression
LCMV	Linearly-Constrained-Minimum-Variance
LMS	Least-Mean-Square
MSD	Mean Squared Deviation
MSE	Mean Squared Error
NLMS	Normalized Least-Mean-Square
NMSE	Normalized Mean Square Error

QP	Quadratic Programming
RHS	Right-Hand Side
RLS	Recursive-Least-Squares
RNMSE	Root Normalized Mean Square Error
ULA	Uniform Linear Array
VAR	Vector Autoregressive
VARMA	Vector Autoregressive Moving Average

Chapter 1

Introduction

With the advent of "Internet of Things", pervasive sensors are deployed and can be connected into networks to accomplish certain tasks collaboratively. There are a wide range of applications are network oriented, such as sensor networks, smart grids, transportation networks, communication networks, to name a few. Recently, the area of cooperative and graph signal processing has received a lot of attention to deal with such complex network systems described by interconnected agents [Djurić 2018]. One important direction is studying and developing strategies that endow networks with adaptation and learning ability. In this Chapter, we first review techniques for adaptation and learning over networks. Then we give a brief introduction of the concepts of graph signal processing. Last, we provide an overview of the contents in this thesis.

1.1 Adaptation and learning over networks

We start with the single agent learning problem. Many learning problems can be interpreted as *stochastic* optimization problems where the objective is to learn a parameter vector $\boldsymbol{w}$ that minimizes a cost function $J(\boldsymbol{w}) : \mathbb{R}^M \to \mathbb{R}$ [Sayed 2014a]:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w} \in \mathbb{R}^M} J(\boldsymbol{w}) = \arg \min_{\boldsymbol{w} \in \mathbb{R}^M} \mathbb{E}\, Q(\boldsymbol{w}; \boldsymbol{x}) \tag{1.1}$$

where the cost function is constructed as the expectation of some loss function, i.e. $J(\boldsymbol{w}) = \mathbb{E}\, Q(\boldsymbol{w}; \boldsymbol{x})$, which is evaluated over the distribution of $\boldsymbol{x}$. The *gradient descent* algorithm can be used to solve the above problem

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mu \nabla J(\boldsymbol{w}_i), \quad i \geq 0 \tag{1.2}$$

where i is an iteration index, $\mu > 0$ is a small step-size, and $\nabla J(\boldsymbol{w}_i)$ is the gradient vector evaluated at $\boldsymbol{w}_i$. However, the distribution of $\boldsymbol{x}$ is usually unknown beforehand, the expectation of $\mathbb{E}\, Q(\boldsymbol{w}; \boldsymbol{x})$ can not be computed and therefore the true gradient is generally not available. Alternatively, if we replace the true gradient by some instantaneous approximation $\widehat{\nabla J}(\boldsymbol{w}_i)$, we arrive at the *stochastic gradient descent* algorithm:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mu \widehat{\nabla J}(\boldsymbol{w}_i), \quad i \geq 0. \tag{1.3}$$

Note that, random perturbations are introduced due to approximation of gradient vectors. Despite that, it can be established that (1.3) will converge to a small region around the minimizer $\boldsymbol{w}^*$ under reasonable conditions on the cost functions and gradient noises [Sayed 2014a].

Let d_i denote a zero-mean real-value random variable realization at i, $\boldsymbol{x}_i \in \mathbb{R}^M$ denote a regression vector with positive-definite covariance matrix $\boldsymbol{R}_x = \mathbb{E}\{\boldsymbol{x}_i \boldsymbol{x}_i^\top\} \succ 0$. Let $\boldsymbol{r}_{dx} = \mathbb{E}\{d_i \boldsymbol{x}_i\}$ denote the cross-covariance vector. The data $\{d_i, \boldsymbol{x}_i\}$ are assumed to be related via the linear regression model:

$$d_i = \boldsymbol{x}_i^\top \boldsymbol{w}^* + v_i \tag{1.4}$$

where $\boldsymbol{w}^*$ is some unknown vector to be estimated and v_i is a zero-mean white noise. The problem of learning $\boldsymbol{w}^*$ can be formulated as minimizing the mean-square error cost:

$$J(\boldsymbol{w}) = \mathbb{E}(d_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2. \tag{1.5}$$

Applying the gradient descent algorithm (1.2), we obtain

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mu \nabla J(\boldsymbol{w}_i) = \boldsymbol{w}_i - 2\mu(\boldsymbol{R}_x \boldsymbol{w}_i - \boldsymbol{r}_{dx}). \tag{1.6}$$

The challenge here is that the statistical moments $\{\boldsymbol{R}_x, \boldsymbol{r}_{dx}\}$ may not be available beforehand. Replacing them by instantaneous approximations at every time instant:

$$\boldsymbol{R}_x \approx \boldsymbol{x}_i \boldsymbol{x}_i^\top, \quad \boldsymbol{r}_{dx} \approx d_i \boldsymbol{x}_i \tag{1.7}$$

leads to the well-known least-mean-squares (LMS) algorithm [Widrow 1985, Haykin 2002, Sayed 2008]:

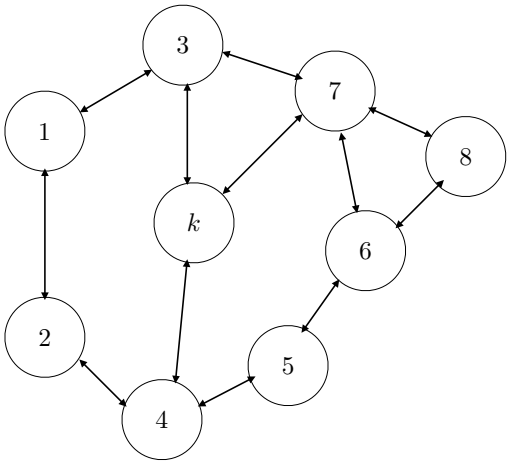
$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + 2\mu \boldsymbol{x}_i (d_i - \boldsymbol{x}_i^\top \boldsymbol{w}_i). \tag{1.8}$$

The LMS belongs to the class of stochastic gradient algorithm.

As noted before, recent years have witnessed a wide spectrum of data processing problems are network-structured [Newman 2010, Lewis 2011] and require adaptation to time-varying dynamics [Sayed 2013, Sayed 2014b]. Sensor networks, social networks, vehicular networks, communication networks, and power grids are some typical examples. These networked systems consist of a collection of autonomous agents (sensors, processors, actuators) linked together through a connection topology. Extending the single agent learning methods to the networked problem, we associate with each node k with an individual cost function $J_k(\boldsymbol{w}) : \mathbb{R}^M \to \mathbb{R}$, the network learning problem can be casted as minimizing the aggregate sum of cost functions:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w} \in \mathbb{R}^M} \sum_{k=1}^{N} J_k(\boldsymbol{w}). \tag{1.9}$$

There are basically two ways to solve the network optimization problem (1.9) collaboratively, namely, centralized and distributed solutions as shown in Figure 1.1. In the centralized mode of operation, there is a fusion center in the network that collects all the data (raw data or processed data) from agents. The fusion center is assumed to have powerful communication and computation ability to process the data centrally. Then, the fusion center sent back the result to each agent when it is needed. Although the centralized solution can be benefit more by collecting all the data, it still suffers from some limitations [Sayed 2014a]: (1) The centralized solution is vulnerable since it highly depends on the fusion center, any failure in the fusion center could lead collapses of network; (2) Exchanging information back and forth between agents and remote fusion center costs a lot of communication resources which would be demanding in very large networks or in some resource stringent applications; (3) In some sensitive applications, considering privacy and secrecy issues, remote agents may be reluctant to sent their data to the fusion center. In

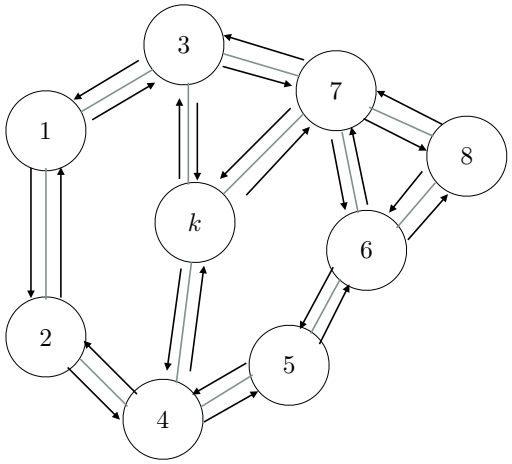


(b) Distributed network

Figure 1.1: Centralized and distributed networks.

Since distributed solution are more favorable for in-network processing, there are several useful distributed strategies that have been proposed in the literature, such as incremental strategies [Bertsekas 1997, Nedic 2001, Rabbat 2005, Lopes 2007, Blatt 2007], consensus strategies [Xiao 2005, Olfati-Saber 2007, Braca 2008, Nedic 2009, Dimakis 2010, Kar 2011, Kar 2012], and diffusion strategies [Lopes 2008, Cattivelli 2010a, Chen 2012]. As shown

in Figure 1.2(a), in the incremental strategy, an agent only communicate to one of its neighbor and data are processed in a cyclic manner through the network until optimization is achieved. However, it is known that finding a cyclic path that covers all nodes is an NP-hard problem. On the other hand, this cyclic path is sensitive to the failures of nodes or links. While in the consensus and diffusion strategies, agents are allowed to exchange information with all their immediate neighbors which are more robust to node and link failures. In the context of adaptation and learning ability in time-varying environments, it has been shown in [Tu 2012] that diffusion strategy outperform consensus strategy where the latter suffers from a stability issue with constant step-size. It is worth noting that, the aforementioned strategies are primal methods, there are also primal-dual methods, such as alternating direction method of multipliers (ADMM) [Boyd 2011, Barbarossa 2013], that can be applied to distributed consensus optimization problems [Schizas 2009, Mateos 2009]. In [Towfic 2015], the authors reveal that the advantages of primal-dual methods in deter-



(b) Consensus and Diffusion strategies

Figure 1.2: Illustration of different distributed strategies.

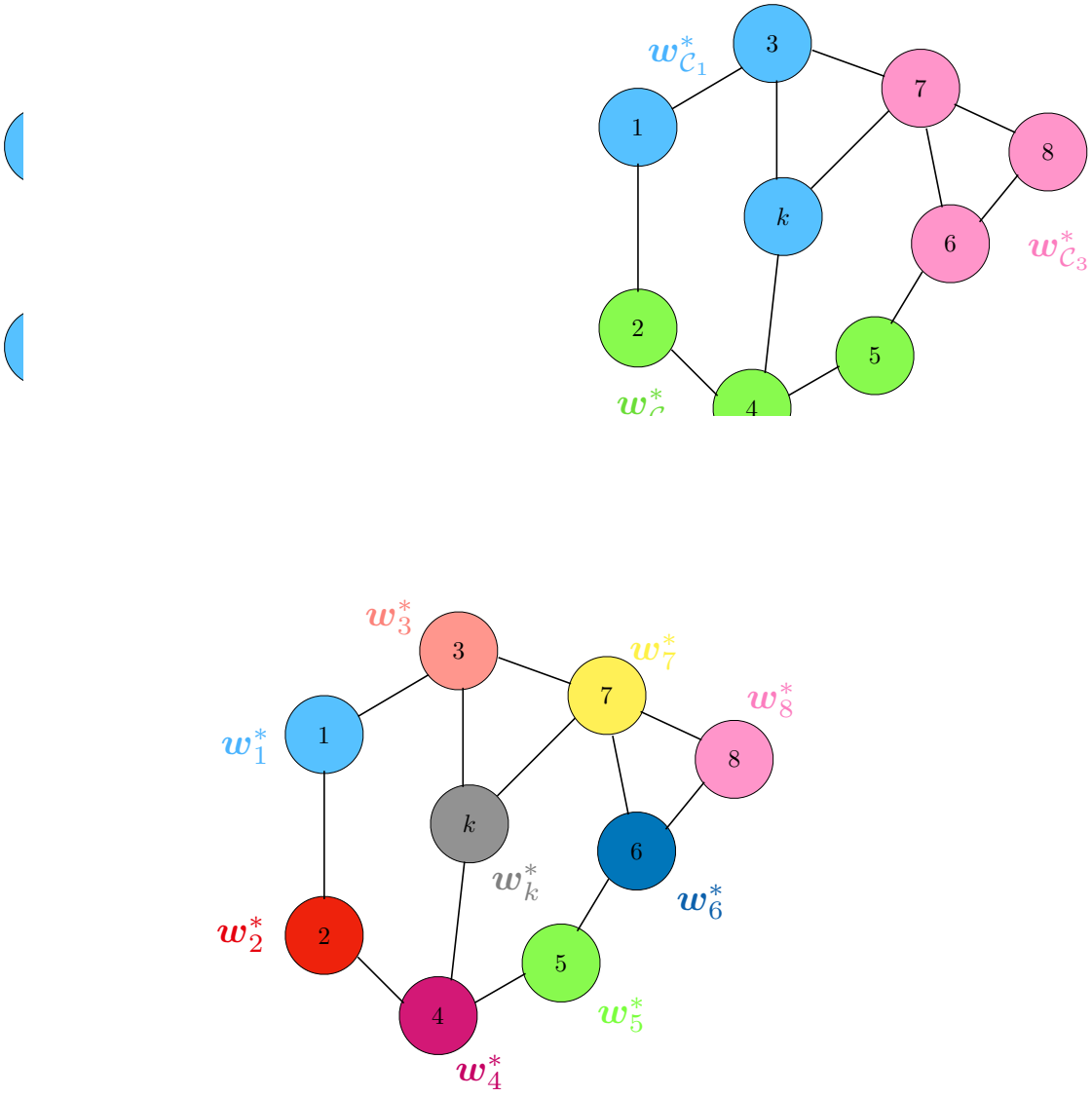
With the attractive features of adaptation and learning ability, diffusion strategies have been widely studied and developed in the literature. Diffusion recursive-least-squares (RLS) [Cattivelli 2008], diffusion affine projection algorithm (APA) [Li 2009], diffusion Kalman filter and smoother [Cattivelli 2010b], diffusion normalized LMS (NLMS) [Xie 2018] have been developed. A diffusion Gauss-Newton method has been proposed to solve the non-

linear least-squares problem [Xiong 2019]. Distributed adaptive detection based on diffusion strategy has been proposed in [Cattivelli 2011]. Diffusion strategies with noisy links, link impairments, asynchronous events, and communication delays have been examined in [Khalili 2011, Abdolee 2016, Zhao 2012, Zhao 2015a, Zhao 2015b, Hua 2020a]. Algorithms for reducing communication costs of diffusion strategies have been proposed in [Arablouei 2014, Harrane 2019]. Taking account the sparsity in the underlying system model [Chen 2009], sparse diffusion LMS strategy has been proposed in [Di Lorenzo 2012]. Stability and performance analysis for diffusion LMS with correlated regressors are provided in [Piggott 2016, Piggott 2017]. Affine combination of diffusion strategies are devised and analyzed in [Jin 2020]. An exact diffusion strategy has been proposed in [Yuan 2019a] for deterministic optimization problem which can achieve exact solutions, while its performance in adaptive networks is analyzed in [Yuan 2019b].

Most of prior works on diffusion estimation assumed that all the agents in the network seek to estimate the same parameter vector, as shown in Figure 1.3(a). Besides single-task networks, there are also applications where it is desirable to estimate multiple parameter vectors at the same time [Plata-Chaves 2017, Nassif 2020a]. The multitask optimization problem can be formulated as:

$$\min_{\{\boldsymbol{w}_k\}_{k=1}^N} J^{\text{glob}}(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N) \triangleq \sum_{k=1}^N J_k(\boldsymbol{w}_k), \tag{1.10}$$

where each agent has its own optimum $\boldsymbol{w}_k^*$ to seek. This problem boils down to the single agent problem (1.1) if all the agents run a stand-alone procedure and do not exchange information with each other. It is shown in [Chen 2013a] that the diffusion strategy converges to a Pareto solution corresponding to a multi-objective optimization problem. Despite the heterogenous nature in multitask networks and the fact that each agent may solve a different task, the agents could still benefit from a collaboration between them. It is shown in [Chen 2015a] that, when the tasks are sufficiently similar to each other, the single-task diffusion LMS algorithm can still perform better than non-cooperative strategies. The similarities between tasks, if are known in prior, can be leveraged to devise multitask strategies to improve the estimation performance. Distributed adaptive node-specific signal estimation methods are proposed over fully connected network and tree networks in [Bertrand 2010] and [Bertrand 2011], respectively. The node-specific signals were assumed to share a common latent signal subspace. In [Bogdanovic 2014, Plata-Chaves 2015, Plata-Chaves 2016], distributed algorithms are derived to estimate node-specific parameter vectors where agents are interested in estimating parameters of local interest and parameters of global interest. Grouping strategy is proposed in [Chen 2016] where some group of entries are same for ad-



(c) Multitask network

Figure 1.3: Illustration of single task and multitask estimation networks. (a) In a single task network, all agents are seek to estimate the same parameter $\boldsymbol{w}^*$; (b) In a clustered multitask network, agents are divided into different clusters, agents in the same cluster (illustrated in the same color) estimate the same task; (c) In a multitask network, each agent estimates distinct but related parameters.

Another useful way to model relationships among tasks is to formulate optimiza-

tion problems with appropriate regularizer between agents [Chen 2014c, Nassif 2016a, Nassif 2016b, Wang 2017]. In this scenario shown in Figure 1.3 (b), the network are assumed to be divided into several clusters and the clusters are known by agents, the agents in the same cluster are assumed to estimate the same task while different clusters have different but similar tasks. The ℓ_1-norm and weighted ℓ_1-norm are employed as co-regularizer to devise multitask strategies for solving problems where the optimal models of adjacent clusters have a large number of similar entries [Nassif 2016b]. While a squared ℓ_2-norm is used in [Chen 2014c, Nassif 2016a, Wang 2017] to promoted smoothness between parameters. Graph spectral regularization was used to improve the multitask network performance by leveraging the graph spectral information [Nassif 2019].

In some applications, it happens that each agent has its own parameter vector to estimate, and these vectors are belong to a low-dimensional subspace [Nassif 2017b, Hua 2017b, Hua 2017a, Alghunaim 2020, Nassif 2020c, Nassif 2020b, Di Lorenzo 2020]. Consider the case where the vectors are coupled together through a set of linear constraints, examples include the network flow control problem [Bertsekas 1998], the interference management problem in communication networks [Shen 2012], and the basis pursuit problem [Mota 2012]. In Chapter 2 and 3, we consider multitask estimation problems where the parameter vectors to be estimated at neighboring agents are related according to a set of linear equality constraints. Therefore, the objective of the network is to optimize the aggregate cost across all nodes subject to all constraints:

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N} \quad J^{\text{glob}}(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N) \triangleq \sum_{k=1}^{N} J_k(\boldsymbol{w}_k), \tag{1.11a}$$

$$\text{s. t.} \quad \sum_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{w}_\ell + \boldsymbol{b}_p = 0, \quad p = 1,\ldots,P \tag{1.11b}$$

with N the number of agents in the network. Each agent k seeks to estimate its own parameter vector $\boldsymbol{w}_k \in \mathbb{R}^{M_k\times 1}$, and has knowledge of its local cost $J_k(\cdot)$ and the set of linear equality constraints that it is involved in. Each constraint is indexed by p, and defined by the $L_p \times M_\ell$ matrix $\boldsymbol{D}_{p\ell}$, the $L_p \times 1$ vector $\boldsymbol{b}_p$, and the set $\mathcal{I}_p$ of agent indices involved in the p-th constraint. Total number of constraints in the network is denoted by P.

There is another scenario in multitask network where the cluster structures are unknown, the only available information is that clusters may exist in the network. Several unsupervised clustering schemes have been proposed in [Zhao 2015c, Chen 2015a, Plata-Chaves 2016, Khawatmi 2017] to extend the diffusion strategy. These methods enable agents to identify which neighbors are sharing a common learning task and which

neighbors should be ignored during cooperation step. We will examine this scenario in Chapter 4.

1.2 Graph signal processing

In many network-structured applications, the data generated by agents are naturally distributed and often exhibit non-Euclidean structures. The emergence of this kind of data and applications has attracted much attention in signal processing community [Shuman 2013, Sandryhaila 2014a, Bronstein 2017, Ortega 2018] as well as machine learning community [Zhang 2020, Wu 2020]. The relationships underlying these data can be captured by a graph. Unlike the traditional signals, such as 1-dimensional time series signal or 2-dimensional image signal, this irregular structured signal do not fall into the scopes of traditional Discrete Signal Processing (DSP) techniques. Graph signal processing (GSP) [Shuman 2013, Ortega 2018] allows us to develop signal processing techniques for analyzing and processing signals that reside on networks, such as graph frequency analysis [Shuman 2013, Sandryhaila 2014b], sampling [Chen 2015b, Wang 2015, Anis 2016, Tsitsvero 2016], and filtering [Shi 2015, Segarra 2017, Isufi 2017b, Coutino 2019, Liu 2019].

Let $\boldsymbol{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}\}$ denote a graph, where $\mathcal{V}$ is a set of N vertices and $\mathcal{E}$ is a set of edges such that $\{k, \ell\} \in \mathcal{E}$ if there is an edge from vertex k to vertex ℓ. The edges may have directions or not. The former type of graph is called a directed graph while the latter is called an undirected graph. Graph $\boldsymbol{\mathcal{G}}$ can be represented by its $N \times N$ adjacency matrix $\boldsymbol{W}$ whose (k, ℓ)-th entry $w_{k\ell}$ assigns a weight to the relation between vertices k and ℓ such that:

$$w_{k\ell} > 0, \text{ if } \{k, \ell\} \in \mathcal{E}, \quad \text{and} \quad w_{k\ell} = 0, \text{ otherwise.} \tag{1.12}$$

For an undirected graph, matrix $\boldsymbol{W}$ is a symmetric matrix. Let $\boldsymbol{D} = \text{diag}\{d_1, \ldots, d_N\}$ denote the diagonal degree matrix whose k-th diagonal entry d_k is the sum of the k-th row entries of $\boldsymbol{W}$:

$$[\boldsymbol{D}]_{k,k} = d_k = \sum_{\ell=1}^{N} w_{k\ell}. \tag{1.13}$$

The combinatorial graph Laplacian matrix is defined as [Chung 1997]:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}. \tag{1.14}$$

Matrix $\boldsymbol{L}$ is a symmetric positive semi-definite matrix for an undirected graph. The normalized Laplacian matrix is defined as follows:

$$\boldsymbol{L}_{\text{norm}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{L} \boldsymbol{D}^{-\frac{1}{2}}. \tag{1.15}$$

For a directed graph, the random walk operator is a probability transition matrix defined as [Lovász 1993]:

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}. \tag{1.16}$$

If the random walk is irreducible, it has a unique stationary distribution $\boldsymbol{\pi}$ that satisfies $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ [Horn 2012]. Its time reversed ergodic random walk is given by [Aldous 1995]:

$$\boldsymbol{P}^* = \boldsymbol{\Pi}^{-1}\boldsymbol{P}^\top\boldsymbol{\Pi}, \tag{1.17}$$

where $\boldsymbol{\Pi} = \text{diag}\{\boldsymbol{\pi}\}$.

When the graph and edge weights are not given for an application, one can define the weight $w_{k\ell}$ of an edge connecting k and ℓ based on their distance. To obtain an undirected graph, a choice is to define the weight $w_{k\ell}$ via a Gaussian kernel [Shuman 2013]:

$$w_{k\ell} = \begin{cases} e^{-\frac{d_{k\ell}^2}{2\theta^2}} & \text{if} \quad d_{k\ell} \leq \kappa, \\ 0 & \text{otherwise}, \end{cases} \tag{1.18}$$

for some parameters θ, κ, and $d_{k\ell}$ may represent a physical distance between vertices k and ℓ, or the Euclidean distance between two feature vectors describing k and ℓ. To construct a directed graph, each vertex is connected to nearest K vertices with directed edges weighted by the normalized inverse exponents of the squared distances [Sandryhaila 2013]:

$$w_{k\ell} = \frac{e^{-d_{k\ell}^2}}{\sqrt{\sum_{m\in\mathcal{N}_k} e^{-d_{km}^2} \sum_{n\in\mathcal{N}_\ell} e^{-d_{\ell n}^2}}}. \tag{1.19}$$

There are scenarios where it is desirable to learn a graph from given data sets, see [Giannakis 2018, Dong 2019, Mateos 2019] and references therein. These approaches aim to find a graph Laplacian or adjacency matrix that can capture the structure of data or explain a data distribution [Zhang 2015].

A graph signal is defined as $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample at vertex k. One of key ingredients of GSP is the definition of the graph shift operator (GSO) which provides a basic operation on a graph signal. A GSO $\boldsymbol{S}$ can be defined as an $N \times N$ matrix which captures the graph topology such that its entries satisfy:

$$s_{k\ell} \neq 0, \text{ if } \{k, \ell\} \in \mathcal{E} \text{ or } k = \ell, \text{ and } s_{k\ell} = 0, \text{ otherwise.} \tag{1.20}$$

Apparently, the adjacency matrix $\boldsymbol{W}$, the Laplacian matrix $\boldsymbol{L}$, and its normalized one $\boldsymbol{L}_{norm}$ can be adopted as the graph shift operator. They are common choices for GSP analysis, e.g., the adjacency matrix $\boldsymbol{W}$ is analogous to the shift in classical time DSP

[Sandryhaila 2013], the Laplacian matrix $\boldsymbol{L}$ have been widely studied in the field of spectral graph theory [Chung 1997]. However, there is no universal operator that is proper for all applications. The choice of operators depends on the problem at hand. This has led several authors to introduce specific ones. For instance, in [Girault 2015b], the authors propose an isometric graph shift operator for stationary graph signals. In [Gavili 2017], the authors introduce a set of operators preserving the energy content of graph signals in the frequency domain. A unitary shift operator is proposed in [Dees 2019], which exhibits property of energy preservation over both backward and forward graph shifts. The authors in [Singh 2016] consider an extension for directed graphs of the symmetric Laplacian matrix. Combination of operators has been considered in [Anis 2016, Sevi 2018a], we will discuss more in Chapter 5 how to model graph signals with combination of operators.

Given a graph shift operator $\boldsymbol{S}$, and assume that $\boldsymbol{S}$ is diagonalizable, such that

$$\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}, \tag{1.21}$$

where $\boldsymbol{V}$ collects the eigenvectors of $\boldsymbol{S}$ as columns, and $\boldsymbol{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_N\}$ collects the eigenvalues. In the view of graph spectral theory, the eigenvalues correspond to graph frequencies and the eigenvectors correspond to frequency components. The graph Fourier transform (GFT) $\hat{\boldsymbol{x}}$ of a graph signal $\boldsymbol{x}$ is defined as

$$\hat{\boldsymbol{x}} = \boldsymbol{V}^{-1}\boldsymbol{x} \tag{1.22}$$

and the inverse GFT is

$$\boldsymbol{x} = \boldsymbol{V}\hat{\boldsymbol{x}}. \tag{1.23}$$

The GFT projects the graph signal $\boldsymbol{x}$ into eigenvector basis and obtains the weights of the frequency content. The inverse GFT reconstructs the graph signal by combining graph frequency components weighted by the coefficients of the signal's graph Fourier transform.

The polynomial shift-invariant graph filter can be defined as [Sandryhaila 2013]:

$$\boldsymbol{H} \triangleq \sum_{\ell=0}^{L-1} h_\ell \boldsymbol{S}^\ell \tag{1.24}$$

where $h_\ell \in \mathbb{R}$ is the filter coefficient, L is called the order of the filter and $L-1$ is called the degree of the filter. Consider a graph signal defined as $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample at vertex k. The filtering process can be expressed as:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} = \sum_{\ell=0}^{L-1} h_\ell \boldsymbol{S}^\ell \boldsymbol{x} \tag{1.25}$$

with $\boldsymbol{y}$ the filter output vector.

Much of the GSP literature has focused on static graph signals, that is, signals that need not evolve with time. However, a wide spectrum of network-structured problems requires adaptation to time-varying dynamics. Prior to the more recent GSP literature, many earlier works on adaptive networks have addressed problems dealing with this challenge by developing processing strategies that are well-suited to data streaming into graphs as reviewed in Section 1.1. We will consider the problem of adaptation and learning on streaming graph signals in Chapter 4.

1.3 Organization of the contents

The objective of this dissertation is to contribute to the investigation of adaptation and learning over multitask networks and graphs. The work in this dissertation is composed of four main parts, as described below.

In Chapter 2, we consider the multitask problem over network (1.11) where each agent is interested in estimating its own parameter vector and where the tasks are related according to a set of linear equality constraints. We assume that each agent possesses its own cost and that the set of constraints is distributed among the agents. Based on the penalty method, we propose an adaptive multitask estimation algorithm in order to allow the network to optimize the individual costs subject to all constraints. We derive conditions on the step-sizes ensuring the stability of the algorithm and we carry out the theoretical performance analysis in the mean and mean-square-error sense.

In Chapter 3, we continue to consider the multitask problem over network whereas agents involved in the same constraint are not necessarily one-hop neighbors. In order to devise a fully distributed algorithm, we employ a multi-hop relay protocol and the penalty method. The resulting algorithm contains delayed intermediate estimates comparing with the algorithm developed in Chapter 2. To see how these delays affect on the stability and performance, we derive a detailed stochastic behavior in an alternative way compared to Chapter 2. We show that the distributed algorithm can continue to converge by selecting proper small step-size.

In Chapter 4, we consider the problem of adaptive and distributed estimation of graph filter coefficients from streaming data. We first formulate this problem as a consensus estimation problem over graphs, which can be addressed with single task distributed strategies. Several diffusion based algorithms are proposed. Performance analysis in the mean and mean-square sense is provided. Then, we extend the single task problem to multitask problem and consider a more general problem where the filter coefficients to estimate may vary

over the graph. To avoid a large estimation bias, we introduce an unsupervised clustering method for splitting the global estimation problem into local ones. Numerical results show the effectiveness of the proposed algorithms and validate the theoretical results.

In Chapter 5, we consider the problem of modeling graph signals with graph filters. In order to enhance interpretability, we propose to use combination of graph filters where each graph has its own graph shift operators. The objective is to learn multiple graph filters coefficients. Due to their extra degrees of freedom, these models might suffer from overfitting. We address this problem through a weighted ℓ_2-norm regularization formulation to perform model selection by encouraging group sparsity. We show that the proposed formulation is actually a smooth convex optimization problem that can be solved efficiently. Experiments on real-world data structured by undirected and directed graphs show the effectiveness of this method.

Finally, the last Chapter concludes the thesis by summarizing main contributions of the methods developed in the previous chapters and provides possible future research directions.

Several publications were presented during the preparation of this dissertation:

1. Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. "Diffusion LMS with communication delays: Stability and performance analysis." *IEEE Signal Processing Letters*, vol. 27, pp. 730–734, 2020.

2. Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. "Online distributed learning over graphs with multitask graph-filter models." *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, no. 1, pp. 63–77, 2020.

3. Fei Hua, Cédric Richard, Jie Chen, Haiyan Wang, Pierre Borgnat and Paulo Gonçalves, "Learning combination of graph filters for graph signal modeling." *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1912–1916, Dec. 2019.

4. Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. "Decentralized clustering for node-variant graph filtering with graph diffusion LMS." in *Proc. 52nd Asilomar Conference on Signals, Systems, and Computers (ASILOMAR),* Pacific Grove, CA, USA, Oct. 2018, pp. 1418–1422.

5. Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *Proc. 26th European Signal Processing Conference (EUSIPCO),* Rome, Italy, Sep. 2018, pp. 111–115.

6. Fei Hua, Roula Nassif, Cédric Richard and Haiyan Wang. "Penalty-based multitask estimation with non-local linear equality constraints," in *Proc. IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP),* Curacao, Netherlands Antilles, Dec. 2017, pp. 1–5.

7. Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Jianguo Huang, "Penalty-based multitask distributed adaptation over networks with constraints," in *Proc. 51st Asilomar Conference on Signals, Systems, and Computers (ASILOMAR),* Pacific Grove, CA, USA, Oct. 2017, pp. 908–912.

8. Yongsheng Yan, Xiaohong Shen, Fei Hua, and Xionghu Zhong, "On the semidefinite programming algorithm for energy-based acoustic source localization in sensor networks," *IEEE Sensors Journal,* vol. 18, no. 21, pp. 8835–8846, Nov. 2018.

9. Mircea Moscu, Roula Nassif, Fei Hua, and Cédric Richard, "Learning causal networks topology from streaming graph signals," in *Proc. 27th European Signal Processing Conference (EUSIPCO),* A Coruña, Spain, Sep. 2019, pp. 1–5.

Chapter 2

Distributed adaptation over multitask networks with constraints

Contents

Multitask distributed optimization over networks enables the agents to cooperate locally to estimate multiple related parameter vectors. In this chapter, we consider multitask estimation problems over mean-square-error (MSE) networks where each agent is interested in estimating its own parameter vector, also called *task*, and where the tasks are related according to a set of linear equality constraints. We assume that each agent possesses its own cost and that the set of constraints is distributed among the agents. In order to solve the multitask problem, a cooperative algorithm based on penalty method is derived. Results

on its stability and convergence properties are also provided. Simulations are conducted to illustrate the theoretical results and show the effectiveness of the strategy. The material in this chapter is based on the work [Hua 2017b].

2.1 Introduction

Distributed adaptive learning strategies over networks enable the agents to accomplish a certain task such as parameter estimation collaboratively from streaming data, and endow the agents with continuous adaptation and learning ability to track possible drifts in the underlying model. Although a centralized strategy may benefit more from information collected throughout the network, in most cases, distributed strategies are more attractive since they are scalable and robust. There is an extensive literature on distributed adaptive methods for single-task problems, where all the agents over the network have a common parameter vector to estimate [Sayed 2014a, Sayed 2014b, Sayed 2014c, Towfic 2014]. However, many applications are multi-task oriented in the sense that the agents have to infer multiple parameter vectors simultaneously. In this case, the agents do not share a common minimizer. It is shown in [Chen 2013a] that the network converges to a Pareto solution corresponding to a multi-objective optimization problem. Multitask diffusion strategies, by exploiting prior information about relationships between the tasks, can let the agents or clusters of agents converge to their own respective models. One useful way to model relationships among tasks is to formulate optimization problems with appropriate regularizer between agents [Chen 2014c, Nassif 2016a, Nassif 2016b]. In [Bogdanovic 2014, Plata-Chaves 2015, Plata-Chaves 2016], distributed algorithms are derived to estimate node-specific parameter vectors where agents are interested in estimating parameters of local interest and parameters of global interest. In other works [Chen 2014b, Chen 2017], the parameter space is decomposed into two orthogonal subspaces. The relations among tasks are modeled by assuming that they all share one of the subspaces. In [Yu 2017], it is assumed that each agent has only access to a subset of the entries of a global parameter vector and only shares the common entries with its neighbors.

In some applications, it happens that each agent has its own parameter vector to estimate and these vectors are coupled together through a set of linear constraints. Examples include the network flow control problem [Bertsekas 1998], the interference management problem in communication networks [Shen 2012], and the basis pursuit problem [Mota 2012]. In this work, we consider multitask estimation problems where the pa-

rameter vectors to be estimated at neighboring agents are related according to a set of linear equality constraints. Therefore, the objective of the network is to optimize the aggregate cost across all nodes subject to all constraints:

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N} \quad J^{\text{glob}}(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N) \triangleq \sum_{k=1}^{N} J_k(\boldsymbol{w}_k), \tag{2.1a}$$

$$\text{s. t.} \quad \sum_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{w}_\ell + \boldsymbol{b}_p = 0, \quad p = 1,\ldots,P \tag{2.1b}$$

with N the number of agents in the network. Each agent k seeks to estimate its own parameter vector $\boldsymbol{w}_k \in \mathbb{R}^{M_k\times 1}$, and has knowledge of its local cost $J_k(\cdot)$ and the set of linear equality constraints that it is involved in. The dimension of the parameter vectors can differ from one node to another. Each constraint is indexed by p, and defined by the $L_p \times M_\ell$ matrix $\boldsymbol{D}_{p\ell}$, the $L_p \times 1$ vector $\boldsymbol{b}_p$, and the set $\mathcal{I}_p$ of agent indices involved in the p-th constraint. It is assumed that each agent k in $\mathcal{I}_p$ can collect information from the other agents in $\mathcal{I}_p$, i.e., $\mathcal{I}_p \subseteq \mathcal{N}_k$ for all $k \in \mathcal{I}_p$ where $\mathcal{N}_k$ denotes the neighborhood of agent k. Total number of constraints in the network is denoted by P.

In [Nassif 2017b], the authors address (2.1) by combining diffusion adaptation with a stochastic projection method. The nodes involved in several constraints are divided into virtual sub-nodes in order to circumvent the problem of projecting their local parameter vector onto several constraint subspaces simultaneously. In this work, we propose an alternative method that consists of reformulating (2.1) as an unconstrained problem with penalty functions. We devise a distributed learning strategy relying on an adaptation step and a penalization step. Although we consider only the case of equality constraints, the algorithm can be easily extended to solve problems with inequality constraints. We analyze its behavior in the mean and mean-square-error sense. Simulations are conducted to show the effectiveness of the proposed strategy.

2.2 Problem formulation

Consider a network of N agents, labeled with $k = 1,\ldots,N$. At each time instant $i \geq 0$, each agent k is assumed to have access to a zero-mean scalar measurement $d_k(i)$ and a real-valued regression vector $\boldsymbol{x}_k(i) \in \mathbb{R}^{M_k\times 1}$ with positive covariance matrix $\boldsymbol{R}_{x,k} = \mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)\}$. The data $\{d_k(i), \boldsymbol{x}_k(i)\}$ are assumed to be related via the linear regression model:

$$d_k(i) = \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k^o + z_k(i) \tag{2.2}$$

where $\boldsymbol{w}_k^o$ is an unknown parameter vector, and $z_k(i)$ is a zero-mean measurement noise with variance $\sigma_{z,k}^2$ assumed to be spatially and temporally independent.

Let $\boldsymbol{w}_k \in \mathbb{R}^{M_k \times 1}$ denote the parameter vector associated with agent k. The objective at agent k is to estimate $\boldsymbol{w}_k^o$ by minimizing the cost function $J_k(\boldsymbol{w}_k)$ given by:

$$J_k(\boldsymbol{w}_k) = \mathbb{E}|d_k(i) - \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k|^2 \tag{2.3}$$

We assume that $J_k(\cdot)$ is strongly convex and second-order differentiable. In addition, we consider that the optimum parameter vectors at neighboring agents are related according to a set of linear equality constraints of the form (2.1b). Each agent k has knowledge of its cost and the set of constraints that it is involved in. We use $\mathcal{J}_k$ to denote the set of constraint indices involving agent k, i.e., $\mathcal{J}_k \triangleq \{p|k \in \mathcal{I}_p\}$.

We collect the parameter vectors $\boldsymbol{w}_k$ and $\boldsymbol{w}_k^o$ from across all nodes into the following $N \times 1$ block vectors:

$$\boldsymbol{w} \triangleq \text{col}\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}, \quad \boldsymbol{w}^o \triangleq \text{col}\{\boldsymbol{w}_1^o, \ldots, \boldsymbol{w}_N^o\}. \tag{2.4}$$

The constraints in (2.1b) can be written more compactly as:

$$\boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b} = \boldsymbol{0} \tag{2.5}$$

where $\boldsymbol{\mathcal{D}}$ is a $P \times N$ block matrix, with each block $\boldsymbol{D}_{p\ell}$ having dimension $L_p \times M_\ell$, and $\boldsymbol{b}$ is a $P \times 1$ block column vector with each block $\boldsymbol{b}_p$ having dimensions $L_p \times 1$. This leads to the network constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \sum_{k=1}^{N} \mathbb{E}|d_k(i) - \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k|^2 \\ \text{s. t.} \quad & \boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b} = \boldsymbol{0}. \end{aligned} \tag{2.6}$$

Let $\boldsymbol{r}_{dx,k} \triangleq \mathbb{E}\{d_k(i)\boldsymbol{x}_k(i)\}$ and $\sigma_{d,k}^2 \triangleq \mathbb{E}|d_k(i)|^2$. Problem (2.6) can be written equivalently as:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \boldsymbol{\mathcal{R}}_x \boldsymbol{w} - 2\boldsymbol{r}_{dx}^\top \boldsymbol{w} + \boldsymbol{\sigma}_d^\top \boldsymbol{1}_{N\times 1}, \\ \text{s. t.} \quad & \boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b} = \boldsymbol{0} \end{aligned} \tag{2.7}$$

where the $N \times N$ block diagonal matrix $\boldsymbol{\mathcal{R}}_x$, the $N \times 1$ block vector $\boldsymbol{r}_{dx}$, and the $N \times 1$ vector $\boldsymbol{\sigma}_d$ are given by:

$$\boldsymbol{\mathcal{R}}_x \triangleq \text{diag}\{\boldsymbol{R}_{x,1}, \ldots, \boldsymbol{R}_{x,N}\}, \tag{2.8}$$

$$\boldsymbol{r}_{dx} \triangleq \text{col}\{\boldsymbol{r}_{dx,1}, \ldots, \boldsymbol{r}_{dx,2}\}, \tag{2.9}$$

$$\boldsymbol{\sigma}_d \triangleq \text{col}\{\sigma_{d,1}^2, \ldots, \sigma_{d,N}^2\}. \tag{2.10}$$

2.3 Centralized and distributed solution

2.3.1 Centralized optimal solution and iterative solution

Because $\boldsymbol{\mathcal{R}}_x$ is positive definite, problem (2.7) is a positive definite quadratic programming problem with equality constraints which has a unique global minimum. We introduce a Lagrange multiplier $\boldsymbol{\lambda}$ and study the Lagrange function

$$\min_{\boldsymbol{w},\boldsymbol{\lambda}} \quad \mathcal{L}(\boldsymbol{w},\boldsymbol{\lambda}) = \boldsymbol{w}^\top \boldsymbol{\mathcal{R}}_x \boldsymbol{w} - 2\boldsymbol{r}_{dx}^\top \boldsymbol{w} + \boldsymbol{\sigma}_d^\top \mathbf{1}_{N\times 1} + \boldsymbol{\lambda}^\top(\boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b}). \tag{2.11}$$

Differentiating $\mathcal{L}(\boldsymbol{w},\boldsymbol{\lambda})$ w.r.t. $\boldsymbol{w}$, we find its gradient vector:

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w},\boldsymbol{\lambda}) = 2\boldsymbol{\mathcal{R}}_x\boldsymbol{w} - 2\boldsymbol{r}_{dx} + \boldsymbol{\mathcal{D}}^\top\boldsymbol{\lambda}. \tag{2.12}$$

Let $\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w},\boldsymbol{\lambda}) = 0$, we get:

$$\boldsymbol{w}^\star = \boldsymbol{w}^o - \frac{1}{2}\boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^\top\boldsymbol{\lambda}. \tag{2.13}$$

Since the optimum $\boldsymbol{w}^\star$ satisfies the constraints, namely $\boldsymbol{\mathcal{D}}\boldsymbol{w}^\star + \boldsymbol{b} = \mathbf{0}$, we get:

$$\boldsymbol{\mathcal{D}}\boldsymbol{w}^o - \frac{1}{2}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^\top\boldsymbol{\lambda} + \boldsymbol{b} = \mathbf{0}. \tag{2.14}$$

Then $\boldsymbol{\lambda}^\star$ can be obtained by solving the above equation:

$$\boldsymbol{\lambda}^\star = 2(\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^\top)^{-1}(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}), \tag{2.15}$$

substituting it into (2.13), the closed form of optimum solution for problem (2.7) is given by:

$$\boldsymbol{w}^\star = \boldsymbol{w}^o - \boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^\top(\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^\top)^{-1}(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}). \tag{2.16}$$

Let Ω denote the linear manifold:

$$\Omega \triangleq \{\boldsymbol{w} : \boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b} = \mathbf{0}\}. \tag{2.17}$$

Observe that if $\boldsymbol{w}^o \in \Omega$, i.e. $\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}$, the optimum $\boldsymbol{w}^\star$ coincides with $\boldsymbol{w}^o$ according to (2.16). In order to solve the problem (2.7) iteratively, we can apply the gradient projection method [Bertsekas 1999] on top of a gradient-descent iteration

$$\boldsymbol{w}(i+1) = P_\Omega\big(\boldsymbol{w}(i) + \mu(\boldsymbol{r}_{dx} - \boldsymbol{\mathcal{R}}_x\boldsymbol{w}(i))\big) \tag{2.18}$$

where $\boldsymbol{w}(i)$ denotes the estimate of $\boldsymbol{w}^\star$ at iteration i and P_Ω denotes projection operation onto Ω. The projection of any vector $\boldsymbol{y} \in \mathbb{R}^M$ on Ω can be expressed as:

$$P_\Omega(\boldsymbol{y}) = \boldsymbol{\mathcal{P}}\boldsymbol{y} - \boldsymbol{f}, \tag{2.19}$$

where

$$\boldsymbol{\mathcal{P}} \triangleq \boldsymbol{I}_M - \boldsymbol{\mathcal{D}}^{\dagger}\boldsymbol{\mathcal{D}}, \tag{2.20}$$

$$\boldsymbol{f} \triangleq \boldsymbol{\mathcal{D}}^{\dagger}\boldsymbol{b}, \tag{2.21}$$

with $\boldsymbol{\mathcal{D}}^{\dagger}$ denoting the pseudo-inverse of the matrix $\boldsymbol{\mathcal{D}}$ given by $\boldsymbol{\mathcal{D}}^{\top}(\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{D}}^{\top})^{-1}$. Notice that, the gradient projection iteration (2.18) requires the second-order statistical moments $\{\boldsymbol{r}_{dx}, \boldsymbol{\mathcal{R}}_x\}$ which are, however, usually unavailable beforehand. We can replace the components of them by the instantaneous approximations $\boldsymbol{R}_{x,k} \approx \boldsymbol{x}_k(i)\boldsymbol{x}_k^{\top}(i), \boldsymbol{r}_{dx,k} \approx d_k(i)\boldsymbol{x}_k(i)$. To this end, we arrive at the stochastic-gradient algorithm:

$$\boldsymbol{w}(i+1) = \boldsymbol{\mathcal{P}}\mathrm{col}\big\{\boldsymbol{w}_k(i) + \mu\boldsymbol{x}_k(i)(d_k(i) - \boldsymbol{x}_k^{\top}(i)\boldsymbol{w}_k(i))\big\}_{k=1}^{N} - \boldsymbol{f} \tag{2.22}$$

which is refer to as the constrained least-mean-square (CLMS) algorithm, which was originally proposed in [Frost 1972] as an online linearly constrained minimum variance (LCMV) filter for solving mean-square-error estimation problems subject to linear constraints in array signal processing.

2.3.2 Penalty functions

The solutions given in last subsection require explicit projection operation onto the constraints. Considering the computational cost, we resort to augmentation-based methods, which offer a simple straightforward way for handling constrained problems. The idea is to approximate the original constrained problem by an unconstrained problem. Basically, there are two alternative methods, called barrier method and penalty method [Bazaraa 2013, Chap. 9], [Luenberger 2015, Chap. 13], to augment the original objective function with a "penalty" term. Barrier method, also known as *interior* penalty function method, augments the original objective function with a barrier penalty term that prevents the points generated from leaving the feasible region. Penalty method, also known as *exterior* penalty function method, adds a term to the objective function to penalize any violation of the constraints. Barrier method requires a strictly feasible initialization and full knowledge of feasible set, this makes it not suitable in distributed settings [Towfic 2014]. In the following, we will focus on the penalty method, which avoids this disadvantage.

To motivate penalty functions, consider the following problem having the single constraint $h(\boldsymbol{w}) = 0$:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & f(\boldsymbol{w}) \\ \text{s. t.} \quad & h(\boldsymbol{w}) = 0. \end{aligned} \tag{2.23}$$

This problem can be replaced by the following unconstrained problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) + \eta(h(\boldsymbol{w}))^2, \tag{2.24}$$

where η is a large number. We can see that an optimal solution to the above problem must have $(h(\boldsymbol{w}))^2$ close to zero, otherwise, a large penalty $\eta(h(\boldsymbol{w}))^2$ will be incurred. This indicates that for equality constraint, a possible choice of a continuous, convex, and twice-differentiable penalty function is the quadratic penalty $\delta^{\text{SEP}}(x) = x^2$.

Consider the following problem having single inequality constraint $g(\boldsymbol{w}) \leq 0$:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & f(\boldsymbol{w}) \\ \text{s.t.} \quad & g(\boldsymbol{w}) \leq 0. \end{aligned} \tag{2.25}$$

This problem can be replaced by the following unconstrained problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) + \eta \max\{0, g(\boldsymbol{w})\}, \tag{2.26}$$

If $g(\boldsymbol{w}) \leq 0$, then $\max\{0, g(\boldsymbol{w})\} = 0$ and no penalty is added. On the other hand, if $g(\boldsymbol{w}) > 0$, then $\max\{0, g(\boldsymbol{w})\} > 0$ and the penalty term $\eta g(\boldsymbol{w})$ is incurred. Notice that $\max\{0, g(\boldsymbol{w})\} > 0$ might not be differentiable. One continuous, convex, non-decreasing, and twice-differentiable penalty function for inequality constraint could be $\delta^{\text{SIP}}(x) = \max\{0, \frac{x^3}{\sqrt{x^2+\epsilon^2}}\}$ with $\epsilon > 0$ a positive value.

2.3.3 Penalty-based distributed solution

By augmenting the objective function with a penalty term, problem (2.1) can be approximated into an unconstrained problem of the following form:

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N} \quad \sum_{k=1}^{N} J_k(\boldsymbol{w}_k) + \eta \sum_{p=1}^{P} \| \sum_{\ell \in \mathcal{I}_p} \boldsymbol{D}_{p\ell} \boldsymbol{w}_\ell + \boldsymbol{b}_p \|^2 \tag{2.27}$$

where $\eta > 0$ is a scalar parameter that controls the relative importance of adhering to the constraints. The approximation (2.27) of (2.1) improves in quality as η increases [Polyak 1987, Bazaraa 2013]. Problem (2.27) can be written alternatively as:

$$\min_{\boldsymbol{w}} \quad J_\eta^{\text{glob}}(\boldsymbol{w}) \triangleq J^{\text{glob}}(\boldsymbol{w}) + \eta \|\boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b}\|^2 \tag{2.28}$$

where $J^{\text{glob}}(\cdot)$ is given in (2.1a). The above problem is strongly convex for any η and its closed form solution parameterized by η is given by:

$$\boldsymbol{w}^o(\eta) = (\boldsymbol{\mathcal{R}}_x + \eta \boldsymbol{\mathcal{D}}^\top \boldsymbol{\mathcal{D}})^{-1} (\boldsymbol{\mathcal{R}}_x \boldsymbol{w}^o - \eta \boldsymbol{\mathcal{D}}^\top \boldsymbol{b}). \tag{2.29}$$

Node k can apply a steepest-descent iteration to minimize the cost in (2.27) with respect to $\boldsymbol{w}_k$. Starting from an initial condition $\boldsymbol{w}_k(0)$, we obtain the following steepest descent iteration at node k:

$$\boldsymbol{w}_k(i+1) = \boldsymbol{w}_k(i) - \mu\Big[\boldsymbol{R}_{x,k}\boldsymbol{w}_k(i) - \boldsymbol{r}_{dx,k} + \eta \sum_{p\in\mathcal{J}_k} \boldsymbol{D}_{pk}^{\top}\big(\sum_{\ell\in\mathcal{I}_p\subseteq\mathcal{N}_k} \boldsymbol{D}_{p\ell}\boldsymbol{w}_\ell(i) + \boldsymbol{b}_p\big)\Big]. \quad (2.30)$$

Replacing the second-order moments $\boldsymbol{R}_{x,k}$, $\boldsymbol{r}_{dx,k}$ by instantaneous approximations:

$$\boldsymbol{R}_{x,k} \approx \boldsymbol{x}_k(i)\boldsymbol{x}_k^{\top}(i), \quad \boldsymbol{r}_{dx,k} \approx d_k(i)\boldsymbol{x}_k(i) \quad (2.31)$$

and implementing the update iteration into two successive steps by introducing the intermediate estimate $\boldsymbol{\phi}_k(i+1)$, we obtain the following adaptive algorithm at agent k:

$$\boldsymbol{\phi}_k(i+1) = \boldsymbol{w}_k(i) + \mu\boldsymbol{x}_k(i)\Big(d_k(i) - \boldsymbol{x}_k^{\top}(i)\boldsymbol{w}_k(i)\Big), \quad (2.32a)$$

$$\boldsymbol{w}_k(i+1) = \boldsymbol{\phi}_k(i+1) - \mu\eta \sum_{p\in\mathcal{J}_k} \boldsymbol{D}_{pk}^{\top}\Big(\sum_{\ell\in\mathcal{I}_p\subseteq\mathcal{N}_k} \boldsymbol{D}_{p\ell}\boldsymbol{\phi}_\ell(i+1) + \boldsymbol{b}_p\Big). \quad (2.32b)$$

Note that in the second step (2.32b), $\boldsymbol{w}_\ell(i)$ is replaced by the intermediate estimate $\boldsymbol{\phi}_\ell(i+1)$, which is a better estimate for the solution at agent ℓ. In the first step (2.32a), which corresponds to the adaptation step, node k uses its data to update its estimate $\boldsymbol{w}_k(i)$ to an intermediate estimate $\boldsymbol{\phi}_k(i+1)$. In the second step (2.32b), which is the penalization step, node k collects the intermediate estimates $\boldsymbol{\phi}_\ell(i+1)$ from its neighbors and moves along the gradient of the penalty function. Note that, in this step, instead of sending $\boldsymbol{\phi}_\ell(i+1)$ to agent k, agent ℓ may send the vector $\boldsymbol{D}_{p\ell}\boldsymbol{\phi}_\ell(i+1)$. In this sense, our algorithm has privacy-preserving property.

2.4 Performance analysis

We shall now analyze the mean and mean-square-error behaviors of the adaptive algorithm (2.32) with respect to the optimal parameter vector $\boldsymbol{w}^o$, the solution $\boldsymbol{w}^\star$ of the constrained optimization problem (2.7), and the solution $\boldsymbol{w}^o(\eta)$ of the approximated unconstrained optimization problem (2.28). Before proceeding, let us introduce the following assumption.

Assumption 2.1 *The regressor $\boldsymbol{x}_k(i)$ arises from a stationary random process that is temporally white and spatially independent with covariance matrix $\boldsymbol{R}_{x,k} = \mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^{\top}(i)\} \succ 0$.*

This is the well-known *Independence Assumption*, which is commonly employed for analyzing adaptive filters and networks [Sayed 2003, Sayed 2008, Sayed 2014a]. Under this

assumption, $\boldsymbol{x}_k(i)$ is independent of $\boldsymbol{w}_\ell(j)$ for all $i > j$ and for all ℓ. Although not true in general, it simplifies the derivations without constraining the conclusions. Furthermore, there are extensive results in the adaptive filtering literature indicating that the performance results obtained using this assumption match well the actual performance for sufficiently small step-sizes.

2.4.1 Error recursion

Let us introduce the error vector w.r.t. $\boldsymbol{w}_k^o$ at node k and time instant i:

$$\widetilde{\boldsymbol{w}}_k(i) \triangleq \boldsymbol{w}_k^o - \boldsymbol{w}_k(i), \tag{2.33}$$

and the intermediate error vector

$$\widetilde{\boldsymbol{\phi}}_k(i) \triangleq \boldsymbol{w}_k^o - \boldsymbol{\phi}_k(i). \tag{2.34}$$

We further collect them into the network block error vectors:

$$\widetilde{\boldsymbol{w}}(i) = \mathrm{col}\{\widetilde{\boldsymbol{w}}_1(i), \ldots, \widetilde{\boldsymbol{w}}_N(i)\}, \tag{2.35}$$

$$\widetilde{\boldsymbol{\phi}}(i) = \mathrm{col}\{\widetilde{\boldsymbol{\phi}}_1(i), \ldots, \widetilde{\boldsymbol{\phi}}_N(i)\}. \tag{2.36}$$

Let $M = \sum_{k=1}^N M_k$ denote the length of the network error vector. Using similar arguments, we define the block error vectors w.r.t. $\boldsymbol{w}_k^o(\eta)$ and $\boldsymbol{w}_k^\star$:

$$\widetilde{\boldsymbol{w}}'(i) = \boldsymbol{w}^o(\eta) - \boldsymbol{w}(i) = \boldsymbol{w}^o - \boldsymbol{w}(i) + \boldsymbol{w}^o(\eta) - \boldsymbol{w}^o = \widetilde{\boldsymbol{w}}(i) + \boldsymbol{w}_\eta^\delta, \tag{2.37}$$

$$\widetilde{\boldsymbol{w}}''(i) = \boldsymbol{w}^\star - \boldsymbol{w}(i) = \boldsymbol{w}^\star - \boldsymbol{w}(i) + \boldsymbol{w}^\star - \boldsymbol{w}^o = \widetilde{\boldsymbol{w}}(i) + \boldsymbol{w}_\star^\delta, \tag{2.38}$$

with

$$\boldsymbol{w}_\eta^\delta \triangleq \boldsymbol{w}^o(\eta) - \boldsymbol{w}^o, \tag{2.39}$$

$$\boldsymbol{w}_\star^\delta \triangleq \boldsymbol{w}^\star - \boldsymbol{w}^o. \tag{2.40}$$

Observe that, the behaviors of algorithm (2.32) w.r.t. $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$ can be deduced from its behavior w.r.t. $\boldsymbol{w}^o$ using the relations (2.37) and (2.38). Therefore, in the squeal, we first study the stochastic behavior of algorithm (2.32) w.r.t. $\boldsymbol{w}^o$ and then extend its behaviors w.r.t. $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$.

Using the linear model (2.2), the estimation error in the adaptation step (2.32a) can be written as:

$$\begin{aligned} d_k(i) - \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k(i) &= \boldsymbol{x}_k^\top(i)(\boldsymbol{w}_k^o - \boldsymbol{w}_k(i)) + z_k(i) \\ &= \boldsymbol{x}_k^\top(i)(\widetilde{\boldsymbol{w}}_k(i)) + z_k(i). \end{aligned} \tag{2.41}$$

Subtracting $\boldsymbol{w}_k^o$ from both sides of the adaptation step (2.32a) and using (2.41), we get

$$\widetilde{\boldsymbol{\phi}}_k(i+1) = [\boldsymbol{I}_{M_k} - \mu\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)]\widetilde{\boldsymbol{w}}_k(i) - \mu\boldsymbol{x}_k(i)z_k(i). \tag{2.42}$$

Collecting the error vector $\widetilde{\boldsymbol{\phi}}_k(i)$ into column block vector, we obtain:

$$\widetilde{\boldsymbol{\phi}}(i+1) = [\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x(i)]\widetilde{\boldsymbol{w}}(i) - \mu\boldsymbol{p}_{zx}(i) \tag{2.43}$$

where

$$\boldsymbol{\mathcal{R}}_x(i) \triangleq \text{bdiag}\Big\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)\Big\}_{k=1}^N, \tag{2.44}$$

$$\boldsymbol{p}_{zx}(i) \triangleq \text{col}\Big\{\boldsymbol{x}_k(i)z_k(i)\Big\}_{k=1}^N. \tag{2.45}$$

Observe that, $\boldsymbol{\mathcal{R}}_x = \mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\}$ and $\mathbb{E}\{\boldsymbol{p}_{zx}(i)\} = 0$.

Subtracting $\boldsymbol{w}_k^o$ from both sides of the penalization step (2.32b), we obtain:

$$\widetilde{\boldsymbol{w}}_k(i+1) = \widetilde{\boldsymbol{\phi}}_k(i+1) + \mu\eta\sum_{p\in\mathcal{J}_k}\boldsymbol{D}_{pk}^\top\Big(\sum_{\ell\in\mathcal{I}_p}\boldsymbol{D}_{p\ell}(\boldsymbol{w}_\ell^o - \widetilde{\boldsymbol{\phi}}_\ell(i+1)) + \boldsymbol{b}_p\Big). \tag{2.46}$$

Collecting into column block vector, we have:

$$\widetilde{\boldsymbol{w}}(i+1) = [\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}]\widetilde{\boldsymbol{\phi}}(i+1) + \mu\eta\boldsymbol{\mathcal{D}}^\top(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}). \tag{2.47}$$

Combing error recursion (2.43) and (2.47), we obtain the network error vector evolves as follow:

$$\widetilde{\boldsymbol{w}}(i+1) = [\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}]\big([\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x(i)]\widetilde{\boldsymbol{w}}(i) - \mu\boldsymbol{p}_{zx}(i)\big) + \mu\eta\boldsymbol{\mathcal{D}}^\top(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}). \tag{2.48}$$

The above recursion can be rewritten in a more compact form:

$$\widetilde{\boldsymbol{w}}(i+1) = \boldsymbol{\mathcal{B}}(i)\widetilde{\boldsymbol{w}}(i) - \mu\boldsymbol{g}(i) + \mu\eta\boldsymbol{f}_o \tag{2.49}$$

by introducing the following notations

$$\boldsymbol{\mathcal{B}}(i) \triangleq \boldsymbol{\mathcal{H}}[\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x(i)], \tag{2.50}$$

$$\boldsymbol{\mathcal{H}} \triangleq [\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}], \tag{2.51}$$

$$\boldsymbol{g}(i) \triangleq \boldsymbol{\mathcal{H}}\boldsymbol{p}_{zx}(i), \tag{2.52}$$

$$\boldsymbol{f}_o \triangleq \boldsymbol{\mathcal{D}}^\top(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}). \tag{2.53}$$

Using the relations (2.37) and (2.38), the error recursion forms w.r.t. $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$ are given by

$$\widetilde{\boldsymbol{w}}'(i+1) = \boldsymbol{\mathcal{B}}(i)\widetilde{\boldsymbol{w}}'(i) - \mu\boldsymbol{g}(i) + \mu(\boldsymbol{r}_\eta(i) + \eta\boldsymbol{f}_\eta), \tag{2.54}$$

$$\widetilde{\boldsymbol{w}}''(i+1) = \boldsymbol{\mathcal{B}}(i)\widetilde{\boldsymbol{w}}''(i) - \mu\boldsymbol{g}(i) + \mu\boldsymbol{r}_s(i), \tag{2.55}$$

with

$$\boldsymbol{r}_{\eta}(i) \triangleq \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{w}_{\eta}^{\delta}, \tag{2.56}$$

$$\boldsymbol{f}_{\eta} \triangleq \boldsymbol{\mathcal{D}}^{\top}(\boldsymbol{\mathcal{D}}\boldsymbol{w}^{o}(\eta) + \boldsymbol{b}), \tag{2.57}$$

$$\boldsymbol{r}_{s}(i) \triangleq \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{w}_{\star}^{\delta}. \tag{2.58}$$

2.4.2 Mean error behavior analysis

Taking the expectation of both sides of recursion (2.48), using Assumption 2.1 and the fact that $\mathbb{E}\boldsymbol{p}_{zx}(i) = 0$, it can be verified that the network mean error vector $\mathbb{E}\widetilde{\boldsymbol{w}}(i)$ evolves according to:

$$\mathbb{E}\widetilde{\boldsymbol{w}}(i+1) = \boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}(i) + \mu\eta\boldsymbol{f}_{o}, \tag{2.59}$$

with

$$\boldsymbol{\mathcal{B}} \triangleq \mathbb{E}\boldsymbol{\mathcal{B}}(i) = \boldsymbol{\mathcal{H}}[\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x]. \tag{2.60}$$

Similarly, we can find that the mean error vectors $\mathbb{E}\widetilde{\boldsymbol{w}}'(i)$ and $\mathbb{E}\widetilde{\boldsymbol{w}}''(i)$ evolve according to:

$$\mathbb{E}\widetilde{\boldsymbol{w}}'(i+1) = \boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}'(i) + \mu(\boldsymbol{r}_{\eta} + \eta\boldsymbol{f}_{\eta}), \tag{2.61}$$

$$\mathbb{E}\widetilde{\boldsymbol{w}}''(i+1) = \boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}''(i) + \mu\boldsymbol{r}_{s}, \tag{2.62}$$

where

$$\boldsymbol{r}_{\eta} \triangleq \mathbb{E}\boldsymbol{r}_{\eta}(i) = \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x\boldsymbol{w}_{\eta}^{\delta}, \tag{2.63}$$

$$\boldsymbol{r}_{s} \triangleq \mathbb{E}\boldsymbol{r}_{s}(i) = \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x\boldsymbol{w}_{\star}^{\delta}. \tag{2.64}$$

Theorem 2.1 (Convergence in the mean) *Assume that data model (2.2) and Assumption 2.1 hold. Then, for any initial condition, algorithm (2.32) converges in the mean, i.e., recursions (2.59), (2.61), and (2.62) converge as $i \to \infty$, if matrix $\boldsymbol{\mathcal{B}}$ given by (2.60) is stable. A sufficient step-size condition for ensuring the stability of matrix $\boldsymbol{\mathcal{B}}$ is:*

$$0 < \mu < \min\left\{\frac{2}{\lambda_{\max}(\boldsymbol{R}_{x,k})}, \frac{2}{\eta \cdot \lambda_{\max}(\boldsymbol{\mathcal{D}}^{\top}\boldsymbol{\mathcal{D}})}\right\},\ k = 1, \ldots, N. \tag{2.65}$$

In this case, the asymptotic mean biases are given by:

$$\mathbb{E}\widetilde{\boldsymbol{w}}(\infty) = \lim_{i\to\infty}\mathbb{E}\widetilde{\boldsymbol{w}}(i) = \mu\eta(\boldsymbol{I}_M - \boldsymbol{\mathcal{B}})^{-1}\boldsymbol{f}_{o}, \tag{2.66}$$

$$\mathbb{E}\widetilde{\boldsymbol{w}}'(\infty) = \lim_{i\to\infty}\mathbb{E}\widetilde{\boldsymbol{w}}'(i) = \mu(\boldsymbol{I}_M - \boldsymbol{\mathcal{B}})^{-1}(\boldsymbol{r}_{\eta} + \eta\boldsymbol{f}_{\eta}), \tag{2.67}$$

$$\mathbb{E}\widetilde{\boldsymbol{w}}''(\infty) = \lim_{i\to\infty}\mathbb{E}\widetilde{\boldsymbol{w}}''(i) = \mu(\boldsymbol{I}_M - \boldsymbol{\mathcal{B}})^{-1}\boldsymbol{r}_{s}. \tag{2.68}$$

PROOF. The mean error recursions (2.59), (2.61), and (2.62) converge if the matrix $\boldsymbol{\mathcal{B}}$ is a stable matrix, which means its spectral radius is strictly less than 1, i.e., $\rho(\boldsymbol{\mathcal{B}}) < 1$. Since the spectral radius of a matrix is upper bounded by any of its induced norms, we have the following relation in terms of the 2-induced norm:

$$\rho(\boldsymbol{\mathcal{B}}) \leq \|(\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x)(\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}})\|_2 \tag{2.69}$$

$$\leq \|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 \cdot \|\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}\|_2 \tag{2.70}$$

where the second inequality follows from the submultiplicative property of matrix norm. In order to ensure $\rho(\boldsymbol{\mathcal{B}}) < 1$, a sufficient condition is to choose step-size μ such that $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$ and $\|\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}\|_2 \leq 1$. Observe that $\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}$ is a symmetric matrix, its 2-induced norm agrees with its spectral radius. It can be verified that $\|\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}}\|_2 \leq 1$ by choosing μ satisfies

$$0 \leq \mu \leq \frac{2}{\eta \cdot \lambda_{\max}(\boldsymbol{\mathcal{D}}^\top\boldsymbol{\mathcal{D}})}. \tag{2.71}$$

Observe that $\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x$ is a block diagonal symmetric matrix, its 2-induced norm is equal to the largest spectral radius of its block components:

$$\begin{aligned}\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 &= \rho(\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x) \\ &= \max_{1\leq k\leq N} \rho(\boldsymbol{I}_{M_k} - \mu\boldsymbol{R}_{x,k}).\end{aligned} \tag{2.72}$$

The condition $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$ is satisfied by choosing μ such that

$$0 < \mu < \min\left\{\frac{2}{\lambda_{\max}(\boldsymbol{R}_{x,k})}\right\},\ k = 1, \ldots, N. \tag{2.73}$$

Combing conditions (2.71) and (2.73) yields to the condition (2.65).

Provided matrix $\boldsymbol{\mathcal{B}}$ is stable, recursions (2.59), (2.61), and (2.62) will converge. Taking limitations of both sides of them, we can find the asymptotic mean biases (2.66), (2.67), and (2.68). ■

Observe that when $\boldsymbol{w}^o$ satisfies the linear constraints, i.e., $\boldsymbol{w}^o = \boldsymbol{w}^o(\eta) = \boldsymbol{w}^\star$, all the biases reduce to zero. When $\boldsymbol{w}^o$ does not satisfy the linear constraints, the algorithm will converge with a bias to optimum in the mean. We can observe that this bias given by (2.68) can be arbitrary small by choosing small step-size.

2.4.3 Mean-square-error behavior analysis

Ensuring the stability in the mean sense is not sufficient. Even the error vectors $\widetilde{\boldsymbol{w}}_k(i)$ are converging on average, they may have large fluctuations around the steady-state values

due to the effect of measurement noises. Therefore, it requires to examine the mean-square analysis to evaluate how the variances $\mathbb{E}\|\widetilde{\boldsymbol{w}}_k(i)\|^2$ evolve with time i and what their steady-state values are.

To do so, we evaluate the mean-square quantity weighted by any positive semi-definite matrix $\boldsymbol{\Sigma}$. Let $\|\boldsymbol{a}\|^2_{\boldsymbol{\Sigma}}$ denote the weighted square quantity $\boldsymbol{a}^\top\boldsymbol{\Sigma}\boldsymbol{a}$, for any vector $\boldsymbol{a}$ and matrix $\boldsymbol{\Sigma}$. The freedom in selecting $\boldsymbol{\Sigma}$ allows us to extract various types of information about the nodes or network. To see this, consider the excess-mean-square-error (EMSE) at agent k and over network which are defined as [Sayed 2008]:

$$\text{EMSE}_k(i) = \mathbb{E}|\boldsymbol{x}_k^\top(i)\widetilde{\boldsymbol{w}}_k(i-1)|^2 \tag{2.74}$$

$$\text{EMSE}_{\text{network}}(i) = \frac{1}{N}\sum_{k=1}^{N}\mathbb{E}|\boldsymbol{x}_k^\top(i)\widetilde{\boldsymbol{w}}_k(i-1)|^2 \tag{2.75}$$

can be obtained from $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{\boldsymbol{\Sigma}}$ by selecting $\boldsymbol{\Sigma}$ as $\text{bdiag}\{\boldsymbol{0},\ldots,\boldsymbol{R}_{x,k},\ldots,\boldsymbol{0}\}$ and $\frac{1}{N}\boldsymbol{\mathcal{R}}_x$, respectively. Matrix $\text{bdiag}\{\boldsymbol{0},\ldots,\boldsymbol{R}_{x,k},\ldots,\boldsymbol{0}\}$ represents a block diagonal matrix where all blocks on the diagonal are zero except for $\boldsymbol{R}_{x,k}$ on the diagonal block of index k. Let us consider the mean-square-deviation (MSD) at agent k and over network which are defined as [Sayed 2008]:

$$\text{MSD}_k(i) = \mathbb{E}\|\widetilde{\boldsymbol{w}}_k(i)\|^2 \tag{2.76}$$

$$\text{MSD}_{\text{network}}(i) = \frac{1}{N}\sum_{k=1}^{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}_k(i)\|^2 \tag{2.77}$$

can be obtained from $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{\boldsymbol{\Sigma}}$ by selecting $\boldsymbol{\Sigma}$ as $\text{bdiag}\{\boldsymbol{0},\ldots,\boldsymbol{I}_{M_k},\ldots,\boldsymbol{0}\}$ and $\frac{1}{N}\boldsymbol{I}_M$, respectively. Therefore, in the following, we will focus on analyzing the weighted mean-square-error $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\Sigma}}$.

From error vector recursion (2.49) and Assumption 2.1, we obtain the weighted mean-square-error relation:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\Sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\Sigma}'} + \mu^2\mathbb{E}\|\boldsymbol{g}(i)\|^2_{\boldsymbol{\Sigma}} + \mu^2\eta^2\|\boldsymbol{f}_o\|^2_{\boldsymbol{\Sigma}} + 2\mu\eta\boldsymbol{f}_o^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}(i), \tag{2.78}$$

with $\boldsymbol{\Sigma}'$ given by

$$\boldsymbol{\Sigma}' = \mathbb{E}\big\{\boldsymbol{\mathcal{B}}^\top(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\big\}. \tag{2.79}$$

It is also convenient to introduce the alternative notation $\|\boldsymbol{a}\|^2_{\boldsymbol{\sigma}}$ to refer to the weighted square quantity $\|\boldsymbol{a}\|^2_{\boldsymbol{\Sigma}}$, where $\boldsymbol{\sigma} \triangleq \text{bvec}(\boldsymbol{\Sigma})$. In the squeal, we will use these two notations interchangeably to denote the same quantity. Note that, we use the block Kronecker product operator $\otimes_b$ [Koning 1991] instead of the Kronecker product $\otimes$, and the

block vectorization operator bvec(·) instead of the vectorization operator vec(·), see Appendix 2.A. These block operators preserve the locality of the blocks in the original matrix arguments [Sayed 2014a, Zhao 2015b]. It will allow us to evaluate the exact expressions of quantities involving fourth-order moments of the regression data. Consider the property (2.110) of *block* matrices, we find that

$$\boldsymbol{\sigma}' \triangleq \text{bvec}(\boldsymbol{\Sigma}') = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma} \tag{2.80}$$

where $\boldsymbol{\mathcal{F}}$ is the $M^2 \times M^2$ matrix given by:

$$\boldsymbol{\mathcal{F}} \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i) \otimes_b \boldsymbol{\mathcal{B}}^\top(i)\}. \tag{2.81}$$

Consider the sufficiently small step-size case where the influence of high-order terms of μ can be neglected [Sayed 2008, Sayed 2014c], matrix $\boldsymbol{\mathcal{F}}$ can be approximated as

$$\boldsymbol{\mathcal{F}} \approx \boldsymbol{\mathcal{B}}^\top \otimes_b \boldsymbol{\mathcal{B}}^\top. \tag{2.82}$$

The first term on the RHS of relation (2.78) can be rewritten as

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\Sigma}'} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\sigma}'} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}. \tag{2.83}$$

The second term on the RHS of relation (2.78) can be evaluated as

$$\mu^2\mathbb{E}\|\boldsymbol{g}(i)\|^2_{\boldsymbol{\Sigma}} = \mu^2\mathbb{E}\{\boldsymbol{g}^\top(i)\boldsymbol{\Sigma}\boldsymbol{g}(i)\} = \mu^2\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}) \overset{(2.110)}{=} \mu^2[\text{bvec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\sigma}. \tag{2.84}$$

where $\boldsymbol{\mathcal{G}}$ is the $M \times M$ matrix given by:

$$\boldsymbol{\mathcal{G}} \triangleq \mathbb{E}\{\boldsymbol{g}(i)\boldsymbol{g}^\top(i)\} = \boldsymbol{\mathcal{H}}\text{bdiag}\{\sigma^2_{z,k}\boldsymbol{R}_{x,k}\}_{k=1}^N\boldsymbol{\mathcal{H}}^\top. \tag{2.85}$$

The third and fourth term on the RHS of relation (2.78) can be expressed as

$$\mu^2\eta^2\|\boldsymbol{f}_o\|^2_{\boldsymbol{\Sigma}} = \mu^2\eta^2\boldsymbol{f}_o^\top\boldsymbol{\Sigma}\boldsymbol{f}_o = \mu^2\eta^2\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{f}_o\boldsymbol{f}_o^\top) = \mu^2\eta^2\big[\text{bvec}(\boldsymbol{f}_o\boldsymbol{f}_o^\top)\big]^\top\boldsymbol{\sigma}, \tag{2.86}$$

$$2\mu\eta\boldsymbol{f}_o^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}(i) = 2\mu\eta\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}\mathbb{E}\widetilde{\boldsymbol{w}}(i)\boldsymbol{f}_o^\top) = 2\mu\eta\big[\text{bvec}\big(\boldsymbol{f}_0\mathbb{E}\widetilde{\boldsymbol{w}}^\top(i)\boldsymbol{\mathcal{B}}^\top\big)\big]^\top\boldsymbol{\sigma}. \tag{2.87}$$

Therefore, the weighted mean-square-error relation (2.78) can be rewritten as:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}} + [\text{bvec}(\boldsymbol{\mathcal{Y}}(i))]^\top\boldsymbol{\sigma}, \tag{2.88}$$

where the matrix $\boldsymbol{\mathcal{Y}}(i)$ is given by:

$$\boldsymbol{\mathcal{Y}}(i) \triangleq \mu^2\boldsymbol{\mathcal{G}}^\top + \mu^2\eta^2\boldsymbol{f}_o\boldsymbol{f}_o^\top + 2\mu\eta\boldsymbol{f}_0\mathbb{E}\widetilde{\boldsymbol{w}}^\top(i)\boldsymbol{\mathcal{B}}^\top. \tag{2.89}$$

Theorem 2.2 (Mean-square stability) *Assume that data model (2.2) and Assumption 2.1 hold. Then, for any initial condition, algorithm (2.32) is mean-square stable if the error recursion (2.59) is mean stable and the matrix $\boldsymbol{\mathcal{F}}$ defined by (2.81) is stable. Assume further that the step-size μ is sufficiently small such that approximation (2.82) is justified by neglecting higher-order powers of μ, and relation (2.88) can be used as a reasonable representation for the evolution of the weighted mean-square-error $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}}$ w.r.t. $\boldsymbol{w}^o$. Then, the mean-square stability of algorithm (2.32) is guaranteed by sufficiently small step-size that also satisfies (2.65).*

PROOF. Iterating (2.88) starting from $i = 0$, we find that:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(0)\|^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}} + \sum_{j=0}^{i}[\text{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^{\top}\boldsymbol{\mathcal{F}}^{j}\boldsymbol{\sigma} \tag{2.90}$$

where $\widetilde{\boldsymbol{w}}(0) = \boldsymbol{w}^o - \boldsymbol{w}(0)$ is an initial condition. Provided that $\boldsymbol{\mathcal{F}}$ is stable, the first term on the RHS of (2.90) converges to 0 as $i \to \infty$. The second term is related to $\boldsymbol{\mathcal{Y}}(i)$ given by (2.89), since μ, η, $\boldsymbol{\mathcal{G}}$, $\boldsymbol{f}_o$, and $\boldsymbol{\mathcal{B}}$ are constant and finite terms, the boundedness of $\boldsymbol{\mathcal{Y}}(i)$ depends on the $\mathbb{E}\widetilde{\boldsymbol{w}}(i)$ being bounded. Notice that (2.59) is a bounded-input bounded-output (BIBO) stable recursion with a bounded driving term $\mu\eta\boldsymbol{f}_o$. It follows that the sum appearing as the right-most term in (2.90) converges as $i \to \infty$. We conclude that $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}}$ converges to a bounded value as $i \to \infty$, and the algorithm is said to be mean-square stable. Under sufficiently small step-size assumption, matrix $\boldsymbol{\mathcal{F}}$ can be reasonably approximated as (2.82). In this case, we have $\rho(\boldsymbol{\mathcal{F}}) = \big(\rho(\boldsymbol{\mathcal{B}})\big)^2$ (c.f. (2.109)), and therefore $\boldsymbol{\mathcal{F}}$ will be stable if $\boldsymbol{\mathcal{B}}$ is stable, which corresponds to condition (2.65). ■

Theorem 2.3 (Transient performance) *Assume the same settings as Theorem 2.2 and sufficiently small step-size μ that ensures mean and mean-square stability. The learning curve $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}}$ evolves according to the following recursion for $i \geq 0$:*

$$\begin{aligned}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\sigma}} &+ \big[\text{bvec}\big(\widetilde{\boldsymbol{w}}(0)\widetilde{\boldsymbol{w}}^{\top}(0)\big)\big]^{\top}(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{M^2})\boldsymbol{\mathcal{F}}^{i}\boldsymbol{\sigma} \\ &+ [\text{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top}\boldsymbol{\sigma} + \boldsymbol{\gamma}(i)\boldsymbol{\sigma}\end{aligned} \tag{2.91}$$

where $\widetilde{\boldsymbol{w}}(0)$ is the initial condition and $\boldsymbol{\gamma}(i+1)$ is an $M^2 \times 1$ vector that can be evaluated from $\boldsymbol{\gamma}(i)$ according to:

$$\boldsymbol{\gamma}(i+1) = [\text{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top}(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{M^2}) + \boldsymbol{\gamma}(i)\boldsymbol{\mathcal{F}} \tag{2.92}$$

with $\boldsymbol{\gamma}(0) = \boldsymbol{0}_{M^2}$.

PROOF. Comparing the expression (2.90) at instant $i+1$ and i, we have:

$$\begin{aligned}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{\boldsymbol{\sigma}}^2 + \mathbb{E}\|\widetilde{\boldsymbol{w}}(0)\|_{(\boldsymbol{\mathcal{F}}^{i+1}-\boldsymbol{\mathcal{F}}^i)\boldsymbol{\sigma}}^2 + \\ \sum_{j=0}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^\top \boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma} - \sum_{j=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j))]^\top \boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}\end{aligned} \tag{2.93}$$

The last two terms on the RHS of (2.93) can be rewritten as:

$$\begin{aligned}&\sum_{j=0}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^\top \boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma} - \sum_{j=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j))]^\top \boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma} \\ &= [\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^\top\boldsymbol{\sigma} + \left(\sum_{j=1}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^\top \boldsymbol{\mathcal{F}}^j - \sum_{j=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j))]^\top \boldsymbol{\mathcal{F}}^j\right)\boldsymbol{\sigma}.\end{aligned} \tag{2.94}$$

Introducing the notation $\boldsymbol{\gamma}(i)$ leads to recursion (2.91):

$$\boldsymbol{\gamma}(i) = \sum_{j=1}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^\top \boldsymbol{\mathcal{F}}^j - \sum_{j=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j))]^\top \boldsymbol{\mathcal{F}}^j. \tag{2.95}$$

Note that, $\boldsymbol{\gamma}(i+1)$ can be evaluated from $\boldsymbol{\gamma}(i)$ recursively according to (2.92), see Appendix 2.C. ■

The *network* MSD learning curve, defined as $\zeta(i) = \frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2$, can be obtained by replacing $\boldsymbol{\sigma}$ with $\frac{1}{N}\mathrm{bvec}(\boldsymbol{I}_M)$ in (2.91):

$$\begin{aligned}\zeta(i+1) = \zeta(i) + \frac{1}{N}\left[\mathrm{bvec}\big(\widetilde{\boldsymbol{w}}(0)\widetilde{\boldsymbol{w}}^\top(0)\big)\right]^\top(\boldsymbol{\mathcal{F}}-\boldsymbol{I}_{M^2})\boldsymbol{\mathcal{F}}^i\mathrm{bvec}(\boldsymbol{I}_M) + \\ \frac{1}{N}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^\top\mathrm{bvec}(\boldsymbol{I}_M) + \frac{1}{N}\boldsymbol{\gamma}(i)\mathrm{bvec}(\boldsymbol{I}_M).\end{aligned} \tag{2.96}$$

Theorem 2.4 (Steady-state performance) *Assume the same settings as Theorem 2.3. The steady-state performance* $\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{\boldsymbol{\sigma}}^2$ *of the algorithm* (2.32) *is given by*

$$\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{\boldsymbol{\sigma}}^2 = [\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(\infty))]^\top(\boldsymbol{I}_{M^2}-\boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}, \tag{2.97}$$

where $\boldsymbol{\mathcal{Y}}(\infty)$ *can be obtained from* (2.66) *and* (2.89).

PROOF. If the matrix $\boldsymbol{\mathcal{F}}$ is stable, from the relation (2.88) and groupe the terms, we obtain as $i\to\infty$:

$$\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|_{(\boldsymbol{I}_{M^2}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}}^2 = [\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(\infty))]^\top\boldsymbol{\sigma}, \tag{2.98}$$

replacing $\boldsymbol{\sigma}$ by $(\boldsymbol{I}_{M^2}-\boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}$ yields (2.97). ■

The steady-state *network* MSD is defined as $\zeta^\star = \lim_{i\to\infty} \frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2$. Replacing $\boldsymbol{\sigma}$ in (2.98) by $\frac{1}{N}(\boldsymbol{I}_{M^2} - \boldsymbol{\mathcal{F}})^{-1}\text{bvec}(\boldsymbol{I}_M)$ we obtain:

$$\zeta^\star = \frac{1}{N}[\text{bvec}(\boldsymbol{\mathcal{Y}}(\infty))]^\top (\boldsymbol{I}_{M^2} - \boldsymbol{\mathcal{F}})^{-1}\text{bvec}(\boldsymbol{I}_M). \tag{2.99}$$

Making use of the relations (2.37) and (2.38), the transient and steady-state behaviors of $\mathbb{E}\|\widetilde{\boldsymbol{w}}'(i)\|^2_{\boldsymbol{\sigma}}$ and $\mathbb{E}\|\widetilde{\boldsymbol{w}}''(i)\|^2_{\boldsymbol{\sigma}}$ can be derived from the model for $\widetilde{\boldsymbol{w}}(i)$ according to:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}'(i)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\sigma}} + 2\mathbb{E}\widetilde{\boldsymbol{w}}(i)\boldsymbol{\Sigma}\boldsymbol{w}^\delta_\eta + \|\boldsymbol{w}^\delta_\eta\|^2_{\boldsymbol{\Sigma}}, \tag{2.100}$$

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}''(i)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\sigma}} + 2\mathbb{E}\widetilde{\boldsymbol{w}}(i)\boldsymbol{\Sigma}\boldsymbol{w}^\delta_\star + \|\boldsymbol{w}^\delta_\star\|^2_{\boldsymbol{\Sigma}}. \tag{2.101}$$

Remark 2.1 *As the statement in Theorem 2.2, the algorithm (2.32) is mean-square stable if $\boldsymbol{\mathcal{F}}$ in (2.81) is stable. For sufficiently small step-size, matrix $\boldsymbol{\mathcal{F}}$ can be approximated by (2.82) and the mean-square stability is ensured by sufficiently small-step size that also satisfies condition (2.65). It is worth noting that, from experiments, the theoretical analyses (2.91) and (2.97) for the mean square error $\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2_{\boldsymbol{\sigma}}$ w.r.t. $\boldsymbol{w}^o$ match well the simulation results in both the cases of explicit $\boldsymbol{\mathcal{F}}$ (2.81) and approximated $\boldsymbol{\mathcal{F}}$ (2.82). While, regarding the mean square errors $\mathbb{E}\|\widetilde{\boldsymbol{w}}'(i)\|^2_{\boldsymbol{\sigma}}$ and $\mathbb{E}\|\widetilde{\boldsymbol{w}}''(i)\|^2_{\boldsymbol{\sigma}}$, the performance evaluations which can be deduced from (2.100) and (2.101) require explicit expression of $\boldsymbol{\mathcal{F}}$. We give the explicit evaluation of matrix $\boldsymbol{\mathcal{F}}$ in Appendix 2.B in the case of zero-mean real Gaussian regressors.*

2.5 Simulations

We shall now provide simulation examples to illustrate the behavior of algorithm (2.32). We considered a network consisting of 15 nodes with connections shown in Fig. 2.1. The regression vectors $\boldsymbol{x}_k(i)$ were zero-mean Gaussian with covariance matrix $\boldsymbol{R}_{x,k} = \sigma^2_{x,k}\boldsymbol{I}_2$. The noises $z_k(i)$ were zero-mean i.i.d. Gaussian random variables independent of any other signal with variances $\sigma^2_{z,k}$. The variances $\sigma^2_{x,k}$ and $\sigma^2_{z,k}$ are shown in Fig. 2.2. We randomly sampled 9 linear constraints of the form $\sum_{\ell\in\mathcal{I}_p} d_{p\ell}\boldsymbol{w}_\ell = b_p \cdot \boldsymbol{1}_{2\times 1}$, where the coefficients $d_{p\ell}$ and b_p were randomly chosen from $\{-2,-1,1,2\}$. The results were averaged over 200 Monte-Carlo runs.

In the first scenario, we considered the case of a perfect model where the parameter vector $\boldsymbol{w}^o$ satisfies the constraints, i.e. $\boldsymbol{w}^o = \boldsymbol{w}^\star$. The step size μ was set to 0.02 for all nodes. In Fig. 2.3, we compare three algorithms: the non-cooperative LMS algorithm, the centralized CLMS algorithm [Frost 1972], and the proposed algorithm (2.32). We observe that the simulation results match well the actual performance. Furthermore, compared to

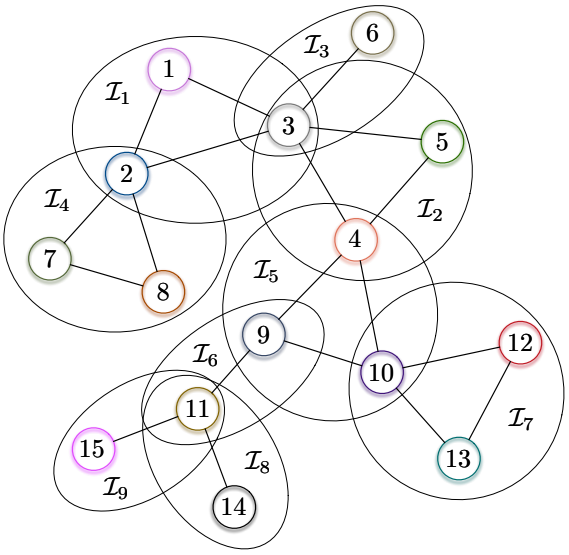


Figure 2.1: Multitask MSE network with local constraints.

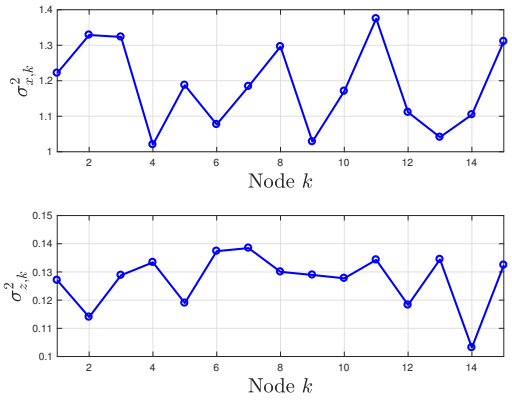


Figure 2.2: Regression and noise variances.

the non-cooperative strategy, the network MSD is improved by the penalty term which, in this case, promotes the relationship between tasks. Finally, our algorithm performs well compared to the centralized solution and the gap w.r.t. centralized performance decreases as η increases.

In a second scenario, we considered the case when $\boldsymbol{w}^o$ does not satisfy the constraints. We perturbed $\boldsymbol{w}^o$ as $\boldsymbol{w}_{\text{pert}} = \boldsymbol{w}^o + \boldsymbol{u}$. The entries of $\boldsymbol{u}$ were sampled from Gaussian distribution $\mathcal{N}(0, \sigma^2)$. We set $\mu = 0.02$ and $\eta = 8$. The theoretical and simulated learning curves with respect to $\boldsymbol{w}^o$, $\boldsymbol{w}^o(\eta)$, and $\boldsymbol{w}^\star$ are depicted in Fig. 2.4, Fig. 2.6, and Fig. 2.6, respectively. Although the performance with respect to $\boldsymbol{w}^o$ deteriorates as σ increases, the

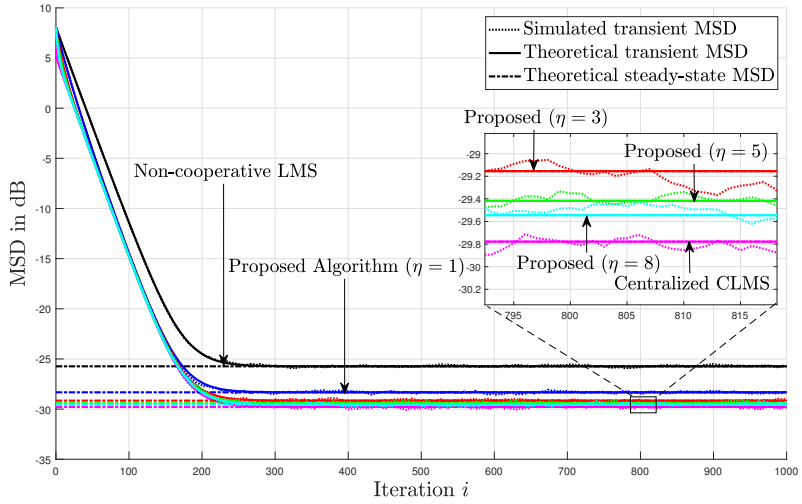


Figure 2.3: MSD comparison of different algorithms for the perfect model scenario.

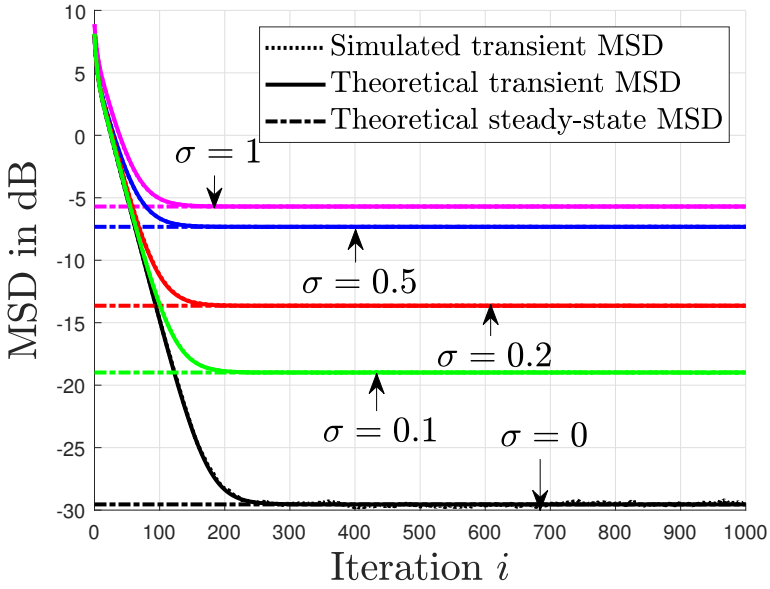


Figure 2.4: MSD w.r.t. $\boldsymbol{w}^o$ of different σ.

algorithm performs well with respect to $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$.

Next, we illustrate in Fig. 2.7 the MSD curves of the centralized algorithm and the proposed algorithm w.r.t. $\boldsymbol{w}^\star$ for different values of σ. We set $\mu = 0.02$ and $\eta = 8$. Observe that the performance gap between the proposed distributed algorithm and the centralized solution increases as the error vector $\boldsymbol{w}_\star^\delta$ increases.

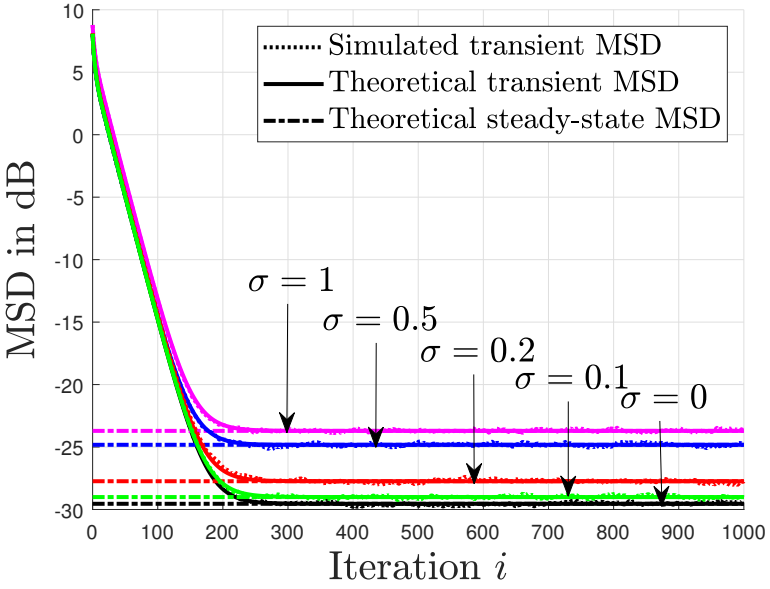


Figure 2.5: MSD w.r.t. $\boldsymbol{w}^o(\eta)$ of different σ.

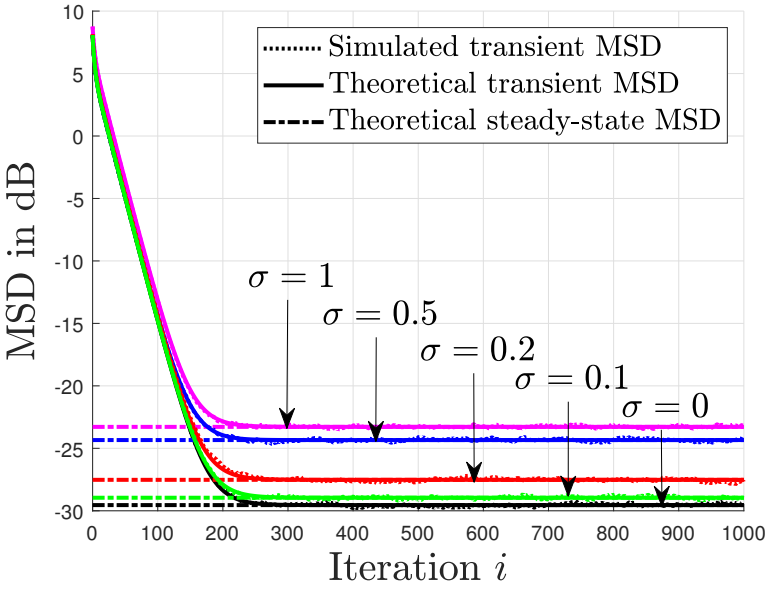


Figure 2.6: MSD w.r.t. $\boldsymbol{w}^\star$ of different σ.

Finally, we illustrate in Fig. 2.8 the influence of μ and η on the performance of the proposed algorithm (2.32). For comparison purposes, we set $\sigma = 1$ for all the μ and η. We observe that, as expected, the larger μ is, the faster the convergence rate is but the worse

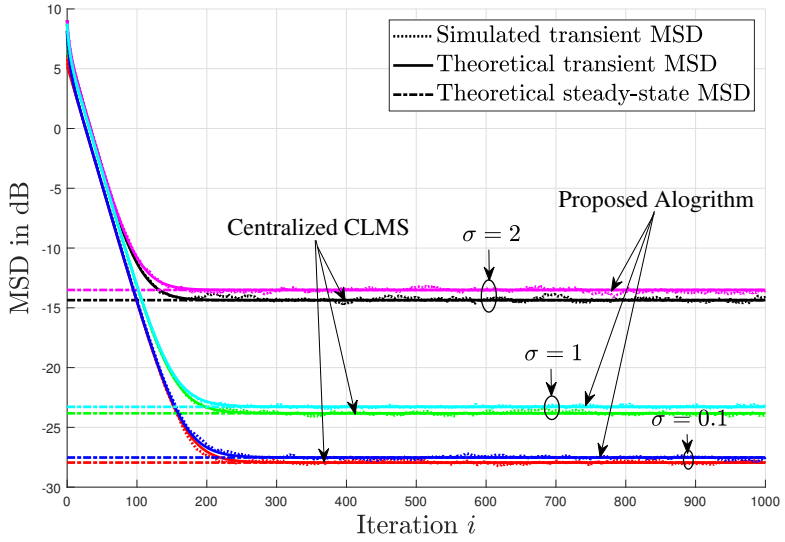


Figure 2.7: MSD comparison with centralized CLMS for different σ.

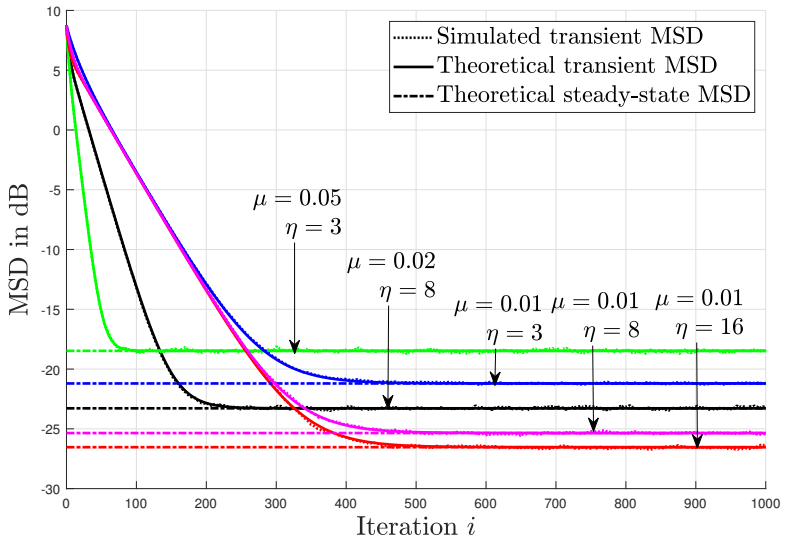


Figure 2.8: MSD comparison for different μ, η.

the MSD performance is. In addition, the performance improves by increasing η.

2.6 Conclusion

In this Chapter, we proposed a distributed multitask LMS algorithm for solving problems that require the simultaneous estimation of multiple parameter vectors that are related locally via linear equality constraints. We approximated the original constrained problem by an unconstrained one based on the penalty method. The behavior of the algorithm in the mean and in the mean-square-error sense was analyzed. Finally, the simulations showed the efficiency of the proposed method. In the next Chapter, we will consider a more general case of constraints that are not necessary local.

Appendices

2.A Block Kronecker product

Let $\boldsymbol{\mathcal{A}}$ denote an $N \times N$ *block* matrix with blocks $\{\boldsymbol{A}_{ij}\}$ of size $P \times P$. Likewise, let $\boldsymbol{\mathcal{B}}$ denote an $M \times M$ *block* matrix with blocks $\{\boldsymbol{B}_{ij}\}$ of size $P \times P$. The *block* Kronecker product of these two matrices is denoted by $\boldsymbol{\mathcal{C}} \triangleq \boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{B}}$ and is defined as the following $NMP^2 \times NMP^2$ matrix [Koning 1991, Sayed 2014a]:

$$\boldsymbol{\mathcal{C}} \triangleq \boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} & \dots & \boldsymbol{C}_{1N} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} & \dots & \boldsymbol{C}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}_{N1} & \boldsymbol{C}_{N2} & \dots & \boldsymbol{C}_{NN} \end{bmatrix} \tag{2.102}$$

with each block $\boldsymbol{C}_{ij}$ is $MP^2 \times MP^2$ and is constructed as:

$$\boldsymbol{C}_{ij} = \begin{bmatrix} \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{11} & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{12} & \dots & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{1M} \\ \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{21} & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{22} & \dots & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{M1} & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{M2} & \dots & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{MM} \end{bmatrix}. \tag{2.103}$$

The bvec($\boldsymbol{\mathcal{A}}$) operator vectorizes each block entry of the matrix $\boldsymbol{\mathcal{A}}$ and then stacks the resulting columns on top of each other, i.e.,

$$\begin{aligned} \text{bvec}(\boldsymbol{\mathcal{A}}) = \text{col}\big\{&\text{vec}(\boldsymbol{A}_{11}), \text{vec}(\boldsymbol{A}_{21}), \dots, \text{vec}(\boldsymbol{A}_{N1}), \\ &\text{vec}(\boldsymbol{A}_{12}), \text{vec}(\boldsymbol{A}_{22}), \dots, \text{vec}(\boldsymbol{A}_{N2}), \\ &\vdots \\ &\text{vec}(\boldsymbol{A}_{1N}), \text{vec}(\boldsymbol{A}_{2N}), \dots, \text{vec}(\boldsymbol{A}_{NN})\big\}. \end{aligned} \tag{2.104}$$

For block matrices $\{\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{D}}\}$ with blocks of size $P \times P$, we have some useful properties of block Kronecker products and block vectorization operations:

$$(\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}) \otimes_b \boldsymbol{\mathcal{C}} = (\boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{C}}) + (\boldsymbol{\mathcal{B}} \otimes_b \boldsymbol{\mathcal{C}}) \tag{2.105}$$

$$(\boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{B}})(\boldsymbol{\mathcal{C}} \otimes_b \boldsymbol{\mathcal{D}}) = (\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{C}} \otimes_b \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{D}}) \tag{2.106}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B}) \otimes_b (\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A} \otimes \boldsymbol{C}) \otimes (\boldsymbol{B} \otimes \boldsymbol{D}) \tag{2.107}$$

$$(\boldsymbol{\mathcal{A}} \otimes_b \boldsymbol{\mathcal{B}})^\top = \boldsymbol{\mathcal{A}}^\top \otimes_b \boldsymbol{\mathcal{B}}^\top \tag{2.108}$$

$$\{\lambda(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}})\} = \{\lambda_i(\boldsymbol{\mathcal{A}})\lambda_j(\boldsymbol{\mathcal{B}})\}_{i=1,j=1}^{NP,MP} \tag{2.109}$$

$$\mathrm{Tr}(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{B}}) = [\mathrm{bvec}(\boldsymbol{\mathcal{B}}^\top)]^\top \mathrm{bvec}(\boldsymbol{\mathcal{A}}) \tag{2.110}$$

$$\mathrm{bvec}(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{B}}) = (\boldsymbol{\mathcal{B}}^\top \otimes_b \boldsymbol{\mathcal{A}})\mathrm{bvec}(\mathcal{C}) \tag{2.111}$$

$$\mathrm{bvec}(\boldsymbol{x}\boldsymbol{y}^\top) = \boldsymbol{y} \otimes_b \boldsymbol{x}. \tag{2.112}$$

2.B Evaluation of matrix $\mathcal{F}$ for zero-mean real Gaussian regressors

Without loss of generality, we assume that $M_k = M_0$ for all k across the network. Recall (2.79), we have

$$\boldsymbol{\Sigma}' = \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\} = \mathbb{E}\{(\boldsymbol{I} - \mu\boldsymbol{\mathcal{R}}_x(i))^\top \boldsymbol{\mathcal{H}}^\top \boldsymbol{\Sigma}\boldsymbol{\mathcal{H}}(\boldsymbol{I} - \mu\boldsymbol{\mathcal{R}}_x(i))\} \tag{2.113}$$

$$= \boldsymbol{\mathcal{H}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{H}} - \mu\boldsymbol{\mathcal{R}}_x\boldsymbol{\mathcal{H}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{H}} - \mu\boldsymbol{\mathcal{H}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x + \mu^2\mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{\mathcal{H}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{R}}_x(i)\}. \tag{2.114}$$

Observe that, in order to evaluate $\boldsymbol{\Sigma}'$, we need to evaluate the fourth item on the RHS of (2.114). Let

$$\boldsymbol{\mathcal{V}} \triangleq \mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{\mathcal{U}}\boldsymbol{\mathcal{R}}_x(i)\}, \tag{2.115}$$

$$\boldsymbol{\mathcal{U}} \triangleq \boldsymbol{\mathcal{H}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{H}}. \tag{2.116}$$

Note that, for any square matrix $\boldsymbol{A}$ and zero-mean Gaussian regressors $\{\boldsymbol{x}_k(i), \boldsymbol{x}_\ell(i)\}$, we have [Sayed 2003, Petersen 2012]:

$$\mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)\boldsymbol{A}\boldsymbol{x}_\ell(i)\boldsymbol{x}_\ell^\top(i)\} = \boldsymbol{R}_{x,k}\boldsymbol{A}\boldsymbol{R}_{x,\ell} + \delta_{k,\ell}(\boldsymbol{R}_{x,k}\boldsymbol{A}^\top\boldsymbol{R}_{x,k} + \boldsymbol{R}_{x,k}\mathrm{Tr}(\boldsymbol{R}_{x,k}\boldsymbol{A})). \tag{2.117}$$

Using the above equality, it can be verified that the (k,ℓ)-th block of matrix $\boldsymbol{\mathcal{V}}$ can be evaluated as

$$[\boldsymbol{\mathcal{V}}]_{k,\ell} = \mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)[\boldsymbol{\mathcal{U}}]_{k,\ell}\boldsymbol{x}_\ell(i)\boldsymbol{x}_\ell^\top(i)\} \tag{2.118}$$

$$= \boldsymbol{R}_{x,k}[\boldsymbol{\mathcal{U}}]_{k,\ell}\boldsymbol{R}_{x,\ell} + \delta_{k,\ell}(\boldsymbol{R}_{x,k}[\boldsymbol{\mathcal{U}}]_{k,\ell}^\top\boldsymbol{R}_{x,k} + \boldsymbol{R}_{x,k}\mathrm{Tr}(\boldsymbol{R}_{x,k}[\boldsymbol{\mathcal{U}}]_{k,\ell})) \tag{2.119}$$

then, matrix $\boldsymbol{\mathcal{V}}$ can be written as

$$\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{R}}_x \boldsymbol{\mathcal{U}} \boldsymbol{\mathcal{R}}_x + \sum_{k=1}^{N} \Big(\boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \boldsymbol{\mathcal{U}}^{\top} (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \boldsymbol{\mathcal{S}}_k + \boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \boldsymbol{\mathcal{Z}}_k \boldsymbol{\mathcal{S}}_k \Big), \tag{2.120}$$

where $\boldsymbol{\mathcal{S}}_k \triangleq \mathrm{diag}(\boldsymbol{e}_k^{\top}) \otimes \boldsymbol{I}_{M_0}$ is an $N \times N$ block diagonal matrix and $\boldsymbol{e}_k$ is an $N \times 1$ column vector with a unit entry at position k and zeros elsewhere. $\boldsymbol{\mathcal{Z}}_k$ is an $N \times N$ block matrix with the (k, ℓ)-th block given by:

$$[\boldsymbol{\mathcal{Z}}_k]_{k,\ell} = [\mathrm{vec}(\boldsymbol{R}_{x,k})]^{\top} \mathrm{vec}([\boldsymbol{\mathcal{U}}]_{k,\ell}) \boldsymbol{I}_{M_0}. \tag{2.121}$$

Applying the block-vectorization operator to $\boldsymbol{\mathcal{V}}$ and using property (2.111), we obtain:

$$\begin{aligned} \mathrm{bvec}(\boldsymbol{\mathcal{V}}) = (\boldsymbol{\mathcal{R}}_x \otimes_b \boldsymbol{\mathcal{R}}_x) \mathrm{bvec}(\boldsymbol{\mathcal{U}}) &+ \sum_{k=1}^{N} \Big(\big(\boldsymbol{\mathcal{S}}_k^{\top} (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \otimes_b \big(\boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \Big) \mathrm{bvec}(\boldsymbol{\mathcal{U}}^{\top}) \\ &+ \sum_{k=1}^{N} \Big(\boldsymbol{\mathcal{S}}_k^{\top} \otimes_b \big(\boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \Big) \mathrm{bvec}(\boldsymbol{\mathcal{Z}}_k), \end{aligned} \tag{2.122}$$

where $\mathrm{bvec}(\boldsymbol{\mathcal{Z}}_k)$ and $\mathrm{bvec}(\boldsymbol{\mathcal{U}})$ given by:

$$\mathrm{bvec}(\boldsymbol{\mathcal{Z}}_k) = \big(\boldsymbol{I}_{N^2} \otimes \mathrm{vec}(\boldsymbol{I}_{M_0}) \otimes [\mathrm{vec}(\boldsymbol{R}_{x,k})]^{\top} \big) \mathrm{bvec}(\boldsymbol{\mathcal{U}}) \tag{2.123}$$

$$\mathrm{bvec}(\boldsymbol{\mathcal{U}}) = (\boldsymbol{\mathcal{H}}^{\top} \otimes_b \boldsymbol{\mathcal{H}}^{\top}) \mathrm{bvec}(\boldsymbol{\Sigma}) = (\boldsymbol{\mathcal{H}}^{\top} \otimes_b \boldsymbol{\mathcal{H}}^{\top}) \boldsymbol{\sigma}. \tag{2.124}$$

Combing (2.114), the explicit expression of $\boldsymbol{\mathcal{F}}$ in (2.81) for zero-mean real Gaussian regressors can be written as:

$$\begin{aligned} \boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{B}}^{\top} \otimes_b \boldsymbol{\mathcal{B}}^{\top} &+ \mu^2 \sum_{k=1}^{N} \Big(\big(\boldsymbol{\mathcal{S}}_k^{\top} (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \otimes_b \big(\boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \Big) (\boldsymbol{\mathcal{H}}^{\top} \otimes_b \boldsymbol{\mathcal{H}}^{\top}) \\ &+ \mu^2 \sum_{k=1}^{N} \Big(\boldsymbol{\mathcal{S}}_k^{\top} \otimes_b \big(\boldsymbol{\mathcal{S}}_k (\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k}) \big) \Big) \big(\boldsymbol{I}_{N^2} \otimes \mathrm{vec}(\boldsymbol{I}_{M_0}) \otimes [\mathrm{vec}(\boldsymbol{R}_{x,k})]^{\top} \big) (\boldsymbol{\mathcal{H}}^{\top} \otimes_b \boldsymbol{\mathcal{H}}^{\top}). \end{aligned} \tag{2.125}$$

2.C Proof of recursion (2.92)

Recall the expression (2.95) of $\boldsymbol{\gamma}(i)$, we find that $\boldsymbol{\gamma}(i+1)$ can be expressed as

$$
\begin{aligned}
\boldsymbol{\gamma}(i+1) &= \sum_{j=1}^{i+1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i+1-j))]^{\top}\boldsymbol{\mathcal{F}}^{j} - \sum_{j=0}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^{\top}\boldsymbol{\mathcal{F}}^{j} \\
&= \sum_{j=1}^{i+1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i+1-j))]^{\top}\boldsymbol{\mathcal{F}}^{j} - \left([\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top} + \sum_{j=1}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j))]^{\top}\boldsymbol{\mathcal{F}}^{j}\right) \\
&\overset{j'\triangleq j-1}{=\!=} \sum_{j'=0}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'}\boldsymbol{\mathcal{F}} - \left([\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top} + \sum_{j'=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'}\boldsymbol{\mathcal{F}}\right) \\
&= \left([\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top}\boldsymbol{\mathcal{F}} + \sum_{j'=1}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'}\boldsymbol{\mathcal{F}}\right) - \\
&\qquad\qquad \left([\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top} + \sum_{j'=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'}\boldsymbol{\mathcal{F}}\right) \\
&= [\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top}(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{M^2}) + \left(\sum_{j'=1}^{i}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'} - \sum_{j'=0}^{i-1}[\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i-1-j'))]^{\top}\boldsymbol{\mathcal{F}}^{j'}\right)\boldsymbol{\mathcal{F}} \\
&= [\mathrm{bvec}(\boldsymbol{\mathcal{Y}}(i))]^{\top}(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{M^2}) + \boldsymbol{\gamma}(i)\boldsymbol{\mathcal{F}}.
\end{aligned}
$$

This concludes that $\boldsymbol{\gamma}(i+1)$ can be recursively evaluated from $\boldsymbol{\gamma}(i)$ according to (2.92).

Chapter 3

Distributed estimation over multitask networks with non-local constraints

Contents

In Chapter 2, we considered distributed estimation over multitask networks where the parameter vectors at distinct agents are coupled via a set of linear equality constraints. However, the constraints were assumed to be local where the agents involved in the same constraint are neighboring agents. In this Chapter, this assumption is relaxed such that the agents are not necessarily one-hop neighbors. At each time instant, we assume that each agent has access to the instantaneous estimates of its one-hop neighbors and the past estimates of its multi-hop neighbors through a multi-hop relay protocol. A *fully* distributed penalty-based algorithm is then derived. The result algorithm employs delayed intermediate estimates comparing with the algorithm in Chapter 2. This makes the performance analysis more sophisticated and requires to re-check the stability conditions. We prove that the distributed algorithm can continue to converge by selecting proper step-size and provide the stochastic behaviors in the mean and mean-square-error sense. Simulation results show the effectiveness of the strategy and validate the theoretical models. The main results established in this chapter were published in [Hua 2017a, Hua 2020a].

3.1 Introduction

Distributed estimation is used in a wide range of applications including communication [Barbarossa 2013], spectrum sensing [Zhang 2014], distributed localization [Meyer 2016], and power system monitoring [Xie 2012]. Several useful distributed solutions, such as incremental strategies [Bertsekas 1997], diffusion strategies [Cattivelli 2010a, Sayed 2014c, Tu 2012, Towfic 2015, Sayed 2013, Sayed 2014a], and consensus strategies [Xiao 2005, Olfati-Saber 2007] have been proposed in the literature to address single-task problems where all agents in the network collaborate to estimate a common parameter vector from noisy measurements. Among them, diffusion strategies are advantageous in terms of stability range, robustness, and performance [Tu 2012, Towfic 2015, Sayed 2013].

In many applications, however, it happens that the agents in the network have to infer multiple parameter vectors simultaneously. Networks of this type are referred to as multitask networks [Chen 2014c, Chen 2015a]. Multitask diffusion strategies were derived by exploiting prior information on the relationships among tasks. For example, appropriate regularization terms can be used to promote similarities between the tasks [Chen 2014c, Wang 2017, Nassif 2016b]. In [Plata-Chaves 2015], a diffusion-based algorithm is proposed to solve node-specific estimation problems where each node consists of a set of local parameters and a set of network global parameters. In [Chen 2014b, Chen 2017], the parameter space is decomposed into two orthogonal subspaces. The relations among tasks are modeled by assuming that they all share one of the subspaces. In some applications, such as the network flow problem [Bertsekas 1998], the basis pursuit problem [Mota 2012], and the interference management problem [Shen 2012], the parameter vectors may be related via a set of linear equality constraints. Distributed projection-based [Nassif 2017b] and penalty-based [Hua 2017b] estimation algorithms were proposed to solve multitask estimation problems where each agent is interested in estimating its own parameter vector and where the parameter vectors at neighboring agents are related according to a set of linear equality constraints. In the current work, we consider a more general multitask scenario where each equality constraint involves agents that are not necessarily one-hop neighbors.

Let N denote the number of agents in the network, and let P denote the total number of constraints. We are interested in devising a distributed adaptive solution to solve the

following optimization problem:

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N} \quad J^{\text{glob}}(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N) \triangleq \sum\nolimits_{k=1}^{N} J_k(\boldsymbol{w}_k), \tag{3.1a}$$

$$\text{s.t.} \quad \sum\nolimits_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{w}_\ell + \boldsymbol{b}_p = 0, \quad p = 1,\ldots,P \tag{3.1b}$$

Each agent k seeks to estimate its parameter vector $\boldsymbol{w}_k \in \mathbb{R}^{M_k\times 1}$, and has knowledge of its local cost $J_k(\cdot)$ and the set of linear equality constraints that it is involved in. Each constraint is indexed by p, and defined by the $L_p \times M_\ell$ matrices $\boldsymbol{D}_{p\ell}$, the $L_p \times 1$ vector $\boldsymbol{b}_p$, and the set $\mathcal{I}_p$ of agent indices involved in the p-th constraint. The previous works [Nassif 2017b, Hua 2017b] assumed that $\mathcal{I}_p \subseteq \mathcal{N}_k$ for all $k \in \mathcal{I}_p$ with $\mathcal{N}_k$ denoting the one-hop neighborhood of agent k that consists of all agents that are connected to k by an edge. In this Chapter, we relax this assumption by considering scenarios where the constraints involve agents that are not necessarily one-hop neighbors. In order to derive a distributed solution relying solely on local interactions between neighbors, we shall employ multi-hop relay protocols to enable non-neighboring agents to share their estimates in order to satisfy their constraints. A penalty-based distributed estimation algorithm is derived and its stochastic behaviors in the mean and mean-square-error sense are analyzed. Simulations are conducted to illustrate the effectiveness of the proposed algorithm and validate the theoretical models.

3.2 Problem formulation and penalty-based solution

Consider a *connected* network of N agents. At each time instant i, each agent k has access to a zero-mean scalar observation $d_k(i)$, and a zero-mean regression vector $\boldsymbol{x}_k(i) \in \mathbb{R}^{M_k\times 1}$ with positive covariance matrix $\boldsymbol{R}_{x,k} = \mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)\} \succ 0$. The observations $\{d_k(i), \boldsymbol{x}_k(i)\}$ are assumed to satisfy a linear regression model:

$$d_k(i) = \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k^o + z_k(i), \tag{3.2}$$

where $\boldsymbol{w}_k^o$ is an $M_k \times 1$ unknown parameter vector to be estimated by agent k, and $z_k(i)$ is a zero-mean noise with variance $\sigma_{z,k}^2$ assumed to be spatially and temporally independent. In order to estimate $\boldsymbol{w}_k^o$, we associate with agent k the mean-square-error cost, which is strongly convex and second-order differentiable :

$$J_k(\boldsymbol{w}_k) = \mathbb{E}|d_k(i) - \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k|^2. \tag{3.3}$$

Let us collect the parameter vectors $\boldsymbol{w}_k$ and $\boldsymbol{w}_k^o$ from across the network into the following vectors of length $M = \sum_{k=1}^{N} M_k$:

$$\boldsymbol{w} \triangleq \text{col}\{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_N\}, \quad \boldsymbol{w}^o \triangleq \text{col}\{\boldsymbol{w}_1^o,\ldots,\boldsymbol{w}_N^o\}.$$

Problem (3.1) can be written equivalently as:

$$\min_{\boldsymbol{w}} \quad \sum\nolimits_{k=1}^{N} \mathbb{E}|d_k(i) - \boldsymbol{x}_k^{\top}(i)\boldsymbol{w}_k|^2, \tag{3.4a}$$

$$\text{s.t.} \quad \boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b} = \boldsymbol{0}. \tag{3.4b}$$

where $\boldsymbol{\mathcal{D}}$ is a $P \times N$ block matrix with block entries $\boldsymbol{D}_{p\ell}$ and $\boldsymbol{b}$ is a $P \times 1$ block column vector with block entries $\boldsymbol{b}_p$. We shall assume that (3.4b) has at least one solution.

The positive-definite quadratic problem (3.4) has a unique global minimum given by:

$$\boldsymbol{w}^{\star} = \boldsymbol{w}^o - \boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^{\top}(\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_x^{-1}\boldsymbol{\mathcal{D}}^{\top})^{-1}(\boldsymbol{\mathcal{D}}\boldsymbol{w}^o + \boldsymbol{b}), \tag{3.5}$$

where $\boldsymbol{\mathcal{R}}_x \triangleq \text{diag}\{\boldsymbol{R}_{x,1}, \ldots, \boldsymbol{R}_{x,N}\}$.

Instead of using (3.5), we are interested in an adaptive distributed solution that is able to learn from streaming data, and relies on local interactions between neighboring agents. Penalty methods offer a simple way for tackling constrained optimization problems. These methods consist of approximating the constrained problem (3.4) into an unconstrained one by adding to the objective function a penalty term that penalizes any violation of the constraints:

$$\min_{\boldsymbol{w}} \quad J_{\eta}^{\text{glob}}(\boldsymbol{w}) \triangleq \sum\nolimits_{k=1}^{N} J_k(\boldsymbol{w}_k) + \eta\|\boldsymbol{\mathcal{D}}\boldsymbol{w} + \boldsymbol{b}\|^2, \tag{3.6}$$

with $\eta > 0$ a scalar parameter that controls the relative importance of adhering to the constraints. Increasing η improves the approximation (3.6) in quality [Polyak 1987, Bazaraa 2013, Luenberger 2015, Towfic 2014], i.e., $\boldsymbol{w}^o(\eta)$ gets closer to $\boldsymbol{w}^{\star}$. The above problem is strongly convex for any η and its closed form solution parameterized by η is given by:

$$\boldsymbol{w}^o(\eta) = (\boldsymbol{\mathcal{R}}_x + \eta\boldsymbol{\mathcal{D}}^{\top}\boldsymbol{\mathcal{D}})^{-1}(\boldsymbol{\mathcal{R}}_x\boldsymbol{w}^o - \eta\boldsymbol{\mathcal{D}}^{\top}\boldsymbol{b}). \tag{3.7}$$

Applying a steepest-descent iteration to minimize the cost in (3.6) with respect to $\boldsymbol{w}_k$ and starting from an initial condition $\boldsymbol{w}_k(0)$, we obtain the following algorithm at node k:

$$\boldsymbol{w}_k(i+1) = \boldsymbol{w}_k(i) - \mu\Big[\boldsymbol{R}_{x,k}\boldsymbol{w}_k(i) - \boldsymbol{r}_{dx,k} + \eta \sum_{p\in\mathcal{J}_k} \boldsymbol{D}_{pk}^{\top}\big(\sum_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{w}_{\ell}(i) + \boldsymbol{b}_p\big)\Big], \tag{3.8}$$

where $\boldsymbol{r}_{dx,k} \triangleq \mathbb{E}\{d_k(i)\boldsymbol{x}_k(i)\}$, $\mathcal{J}_k$ denotes the set of constraint indices involving agent k, i.e., $\mathcal{J}_k \triangleq \{p|k \in \mathcal{I}_p\}$. In order to evaluate $\sum_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{w}_{\ell}(i) + \boldsymbol{b}_p$ in (3.8), agent k needs the estimates $\boldsymbol{w}_{\ell}(i)$ from all agents $\ell \in \mathcal{I}_p$. These agents are not necessarily in the one-hop neighborhood of k, and need to employ multi-hop relay protocols to share their own estimates. Since the network is connected, there is at least one path between any pair of nodes k and ℓ. We shall assume that the route from agent $\ell \in \mathcal{I}_p$ to agent k with the smallest number of relays or hops, which is often the most energy-efficient route [Dargie 2010], is

known. Instead of using $\boldsymbol{w}_\ell(i)$ since it may not be available, agent k will use past estimate $\boldsymbol{w}_\ell(i-j)$ of agent ℓ where the delay j depends on the smallest number of hops from agent ℓ to agent k, denoted by $h_{\ell k}$. In the sequel, we shall assume that $j = h_{\ell k} - 1$. With this multi-hop relay protocol, at each time instant i, agent k has access to $\boldsymbol{w}_\ell(i+1-h_{\ell k})$.

Usually, the second-order moments $\boldsymbol{R}_{x,k}$ and $\boldsymbol{r}_{dx,k}$ in (3.8) are not available beforehand. We replace them by the instantaneous approximations [Sayed 2008]:

$$\boldsymbol{R}_{x,k} \approx \boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i), \quad \boldsymbol{r}_{dx,k} \approx d_k(i)\boldsymbol{x}_k(i), \tag{3.9}$$

Replacing $\boldsymbol{w}_\ell(i)$ in (3.8) by $\boldsymbol{w}_\ell(i+1-h_{\ell k})$, and splitting the update iteration into two incremental steps by introducing the intermediate estimate $\boldsymbol{\phi}_k(i+1)$, we obtain the following adaptive algorithm at agent k:

$$\boldsymbol{\phi}_k(i+1) = \boldsymbol{w}_k(i) + \mu\boldsymbol{x}_k(i)\Big(d_k(i) - \boldsymbol{x}_k^\top(i)\boldsymbol{w}_k(i)\Big), \tag{3.10a}$$

$$\boldsymbol{w}_k(i+1) = \boldsymbol{\phi}_k(i+1) - \mu\eta \sum_{p\in\mathcal{J}_k} \boldsymbol{D}_{pk}^\top\Big(\sum_{\ell\in\mathcal{I}_p} \boldsymbol{D}_{p\ell}\boldsymbol{\phi}_\ell(i+2-h_{\ell k}) + \boldsymbol{b}_p\Big), \tag{3.10b}$$

where in the second step (3.10b), we replaced $\boldsymbol{w}_\ell(i+1-h_{\ell k})$ by the intermediate estimate $\boldsymbol{\phi}_\ell(i+2-h_{\ell k}))$, which is a better estimate for the solution at agent ℓ. We set $\boldsymbol{\phi}_\ell(i+2-h_{\ell k}) = \boldsymbol{0}$ if $i+2-h_{\ell k} < 0$, and $h_{kk} = 1$. In the first step (3.10a), which is the adaptation step, node k uses its own data to update its estimate $\boldsymbol{w}_k(i)$ to an intermediate estimate $\boldsymbol{\phi}_k(i+1)$. In the second step (3.10b), which corresponds to the penalization step, node k collects the intermediate estimates $\boldsymbol{\phi}_\ell(i+2-h_{\ell k})$ from nodes $\ell \in \mathcal{I}_p$ for all $p \in \mathcal{J}_k$.

3.3 Stochastic behavior analysis

In this section, we study the mean and the mean-square-error behavior of algorithm (3.10) w.r.t. $\boldsymbol{w}^o(\eta)$ (3.7) and $\boldsymbol{w}^\star$ (3.5).

Remark 3.1 *From the results in Chapter 2, we know that the penalty-based distributed algorithm converges to the optimum in the perfect model scenario where $\boldsymbol{w}^o = \boldsymbol{w}^o(\eta) = \boldsymbol{w}^\star$. While in the imperfect model scenario, the algorithm will converge around $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$ with biases but far away from $\boldsymbol{w}^o$. To this end, we will directly derive the performance analysis w.r.t. $\boldsymbol{w}^o(\eta)$ and $\boldsymbol{w}^\star$.*

To perform the theoretical analysis, we introduce the following assumption before proceeding.

Assumption 3.1 *The regressor $\boldsymbol{x}_k(i)$ arises from a zero-mean random process that is temporally white and spatially independent.*

As already explained in Assumption 2.1, this assumption is commonly used in the adaptive filtering literature. It helps to simplify the derivations without constraining the conclusions.

3.3.1 Extended error recursion

Let $\widetilde{\boldsymbol{w}}_k(i) \triangleq \boldsymbol{w}_k^o(\eta) - \boldsymbol{w}_k(i)$, $\widetilde{\boldsymbol{\phi}}_k(i) \triangleq \boldsymbol{w}_k^o(\eta) - \boldsymbol{\phi}_k(i)$ denote the error vector and the error vector w.r.t. $\boldsymbol{w}_k^o(\eta)$ at agent k and time instant i. We further introduce the network $N \times 1$ block error vectors:

$$\widetilde{\boldsymbol{w}}(i) = \mathrm{col}\{\widetilde{\boldsymbol{w}}_1(i), \ldots, \widetilde{\boldsymbol{w}}_N(i)\}, \quad (M \times 1) \tag{3.11}$$

$$\widetilde{\boldsymbol{\phi}}(i) = \mathrm{col}\{\widetilde{\boldsymbol{\phi}}_1(i), \ldots, \widetilde{\boldsymbol{\phi}}_N(i)\}. \quad (M \times 1) \tag{3.12}$$

Let $M = \sum_{k=1}^N M_k$, the length of the block error vectors is then M.

Subtracting $\boldsymbol{w}_k^o(\eta)$ from both sides of the adaptation step (3.10a) and using the data model (3.2), similar to the process in Chapter 2, it can be verified that

$$\widetilde{\boldsymbol{\phi}}(i+1) = [\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x(i)]\widetilde{\boldsymbol{w}}(i) + \mu\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{w}^\delta - \mu\boldsymbol{p}_{zx}(i) \tag{3.13}$$

where

$$\boldsymbol{\mathcal{R}}_x(i) \triangleq \mathrm{bdiag}\Big\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^\top(i)\Big\}_{k=1}^N, \qquad (M \times M) \tag{3.14}$$

$$\boldsymbol{w}^\delta \triangleq \boldsymbol{w}^o(\eta) - \boldsymbol{w}^o, \qquad (M \times 1) \tag{3.15}$$

$$\boldsymbol{p}_{zx}(i) \triangleq \mathrm{col}\Big\{\boldsymbol{x}_k(i)z_k(i)\Big\}_{k=1}^N. \qquad (M \times 1) \tag{3.16}$$

Note that, $\boldsymbol{\mathcal{R}}_x = \mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\}$ and $\mathbb{E}\{\boldsymbol{p}_{zx}(i)\} = 0$. However, the error relation between $\widetilde{\boldsymbol{w}}(i+1)$ and $\widetilde{\boldsymbol{\phi}}(i+1)$ cannot be obtained by subtracting $\boldsymbol{w}_k^o(\eta)$ from both sides of (3.10b) due to delayed information. In order to proceed, taking into account the delayed information emerging from the multi-hop protocol, we introduce the following extended network error vectors:

$$\begin{aligned}\widetilde{\boldsymbol{w}}_e(i) \triangleq \mathrm{col}\big\{&\widetilde{\boldsymbol{w}}_1(i), \ldots, \widetilde{\boldsymbol{w}}_N(i), \widetilde{\boldsymbol{\phi}}_1(i), \ldots, \widetilde{\boldsymbol{\phi}}_N(i), \ldots, \\ &\widetilde{\boldsymbol{\phi}}_1(i-H+2), \ldots, \widetilde{\boldsymbol{\phi}}_N(i-H+2)\big\}, \quad (MH \times 1)\end{aligned} \tag{3.17}$$

$$\begin{aligned}\widetilde{\boldsymbol{\phi}}_e(i) \triangleq \mathrm{col}\big\{&\widetilde{\boldsymbol{\phi}}_1(i), \ldots, \widetilde{\boldsymbol{\phi}}_N(i), \widetilde{\boldsymbol{\phi}}_1(i-1), \ldots, \widetilde{\boldsymbol{\phi}}_N(i-1), \\ &\ldots, \widetilde{\boldsymbol{\phi}}_1(i-H+1), \ldots, \widetilde{\boldsymbol{\phi}}_N(i-H+1)\big\}, \quad (MH \times 1)\end{aligned} \tag{3.18}$$

where $H = \max_{k,\ell}\{h_{\ell k}\}$. It can be verified that, from (3.13), the extended block error $\widetilde{\boldsymbol{\phi}}_e(i+1)$ is related to $\widetilde{\boldsymbol{w}}_e(i)$ as

$$\widetilde{\boldsymbol{\phi}}_e(i+1) = [\boldsymbol{I}_{MH} - \mu\boldsymbol{\mathcal{R}}_{x,e}(i)]\widetilde{\boldsymbol{w}}_e(i) + \mu\boldsymbol{\mathcal{R}}_{x,e}(i)\boldsymbol{w}_e^\delta + \mu\boldsymbol{p}_{zx,e}(i) \tag{3.19}$$

where

$$\boldsymbol{\mathcal{R}}_{x,e}(i) \triangleq \text{bdiag}\big\{\boldsymbol{\mathcal{R}}_x(i), \underbrace{\boldsymbol{0}_{M\times M},\ldots,\boldsymbol{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\big\}, \quad (MH\times MH) \tag{3.20}$$

$$\boldsymbol{w}_e^{\delta} \triangleq \boldsymbol{1}_H \otimes \boldsymbol{w}^{\delta}, \quad (MH\times 1) \tag{3.21}$$

$$\boldsymbol{p}_{zx,e}(i) \triangleq \text{col}\Big\{\boldsymbol{p}_{zx}(i), \underbrace{\boldsymbol{0}_{M\times 1},\ldots,\boldsymbol{0}_{M\times 1}}_{\text{in total } H\text{-1 block items}}\Big\}. \quad (MH\times 1) \tag{3.22}$$

The extended block error $\widetilde{\boldsymbol{w}}_e(i+1)$ can be verified to be related to $\widetilde{\boldsymbol{\phi}}_e(i+1)$ as

$$\widetilde{\boldsymbol{w}}_e(i+1) = \boldsymbol{\mathcal{H}}_e\widetilde{\boldsymbol{\phi}}_e(i+1) + \mu\eta\boldsymbol{f}_{\eta,e} \tag{3.23}$$

where

$$\boldsymbol{\mathcal{H}}_e \triangleq [\boldsymbol{\mathcal{I}} - \mu\eta\,\boldsymbol{\mathcal{C}}_{\mathcal{D},e}], \quad (MH\times MH) \tag{3.24}$$

$$\boldsymbol{\mathcal{I}} = \begin{bmatrix} \boldsymbol{I}_M & \boldsymbol{0}_{M\times M(H-1)} \\ \boldsymbol{I}_{M(H-1)} & \boldsymbol{0}_{M(H-1)\times M} \end{bmatrix}, \quad (MH\times MH) \tag{3.25}$$

$$\boldsymbol{\mathcal{C}}_{\mathcal{D},e} = \begin{bmatrix} \boldsymbol{C}_{D,1} & \boldsymbol{C}_{D,2} & \cdots & \boldsymbol{C}_{D,H} \\ & \boldsymbol{0}_{M(H-1)\times MH} & & \end{bmatrix}, \quad (MH\times MH) \tag{3.26}$$

$$\boldsymbol{f}_{\eta,e} \triangleq \text{col}\Big\{\boldsymbol{\mathcal{D}}^{\top}(\boldsymbol{\mathcal{D}}\boldsymbol{w}^{o}(\eta)+\boldsymbol{b}), \underbrace{\boldsymbol{0}_{M\times 1},\ldots,\boldsymbol{0}_{M\times 1}}_{\text{in total } H\text{-1 block items}}\Big\}. \quad (MH\times 1) \tag{3.27}$$

and $\boldsymbol{C}_{D,h}$ is an $N\times N$ block matrix with (k,ℓ)-th block given by:

$$[\boldsymbol{C}_{D,h}]_{k,\ell} = \begin{cases} \sum_{p\in\mathcal{J}_k}\boldsymbol{D}_{pk}^{\top}\boldsymbol{D}_{p\ell} & \text{if} \quad \ell\in\mathcal{I}_p\cap\mathcal{N}_k^{(h)}, \\ \boldsymbol{0} & \text{otherwise.} \end{cases} \quad (M\times M) \tag{3.28}$$

Observe that $\sum_{h=1}^{H}\boldsymbol{C}_{D,h} = \boldsymbol{\mathcal{D}}^{\top}\boldsymbol{\mathcal{D}}$. Note that matrix $\boldsymbol{\mathcal{I}}$ given by (3.25) is neither an identity matrix nor a symmetric matrix. The resulting extended matrix $\boldsymbol{\mathcal{H}}_e$ given by (3.24) is not a symmetric matrix, whereas $\boldsymbol{\mathcal{H}} \triangleq [\boldsymbol{I}_M - \mu\eta\boldsymbol{\mathcal{D}}^{\top}\boldsymbol{\mathcal{D}}]$ (c.f. (2.51)) is a symmetric matrix.

Introducing the following notations

$$\boldsymbol{\mathcal{B}}_e(i) \triangleq \boldsymbol{\mathcal{H}}_e\,[\boldsymbol{I}_{MH} - \mu\boldsymbol{\mathcal{R}}_{x,e}(i)], \quad (MH\times MH) \tag{3.29}$$

$$\boldsymbol{g}_e(i) \triangleq \boldsymbol{\mathcal{H}}_e\boldsymbol{p}_{zx,e}(i), \quad (MH\times 1) \tag{3.30}$$

$$\boldsymbol{r}_{\eta,e}(i) \triangleq \boldsymbol{\mathcal{H}}_e\boldsymbol{\mathcal{R}}_{x,e}(i)\boldsymbol{w}_e^{\delta}, \quad (MH\times 1) \tag{3.31}$$

the block vector $\widetilde{\boldsymbol{w}}_e(i)$ evolves according to the following form:

$$\widetilde{\boldsymbol{w}}_e(i+1) = \boldsymbol{\mathcal{B}}_e(i)\widetilde{\boldsymbol{w}}_e(i) - \mu\boldsymbol{g}_e(i) + \mu\boldsymbol{r}_{\eta,e}(i) + \mu\eta\boldsymbol{f}_{\eta,e}. \tag{3.32}$$

3.3.2 Mean error behavior analysis

Taking the expectation of both sides of (3.32), using Assumption 3.1 and the fact that $\mathbb{E}\{\boldsymbol{g}_e(i)\} = \mathbf{0}$, we obtain:

$$\mathbb{E}\widetilde{\boldsymbol{w}}_e(i+1) = \boldsymbol{\mathcal{B}}_e \mathbb{E}\widetilde{\boldsymbol{w}}_e(i) + \mu \boldsymbol{r}_{\eta,e} + \mu\eta \boldsymbol{f}_{\eta,e}, \tag{3.33}$$

where

$$\boldsymbol{\mathcal{B}}_e \triangleq \boldsymbol{\mathcal{H}}_e \left[\boldsymbol{I}_{MH} - \mu \boldsymbol{\mathcal{R}}_{x,e}\right], \tag{3.34}$$

$$\boldsymbol{r}_{\eta,e} \triangleq \boldsymbol{\mathcal{H}}_e \boldsymbol{\mathcal{R}}_{x,e} \boldsymbol{w}_e^{\delta}, \tag{3.35}$$

$$\boldsymbol{\mathcal{R}}_{x,e} \triangleq \text{bdiag}\{\boldsymbol{\mathcal{R}}_x, \underbrace{\mathbf{0}_{M\times M}, \ldots, \mathbf{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\}. \tag{3.36}$$

Observe that, the mean error behavior depends on matrix $\boldsymbol{\mathcal{B}}_e$. Let us introduce the following Lemma:

Lemma 3.1 *If $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$, $\|\boldsymbol{\mathcal{H}}\|_2 \leq 1$ and $\|\boldsymbol{\mathcal{H}}_e\|_2 \leq 1$, then the matrix $\boldsymbol{\mathcal{B}}_e$ in (3.29) is stable, i.e., its spectral radius $\rho(\boldsymbol{\mathcal{B}}_e)$ is less than 1.*

Proof. The argument is similar of the proof for the Lemma 1 presented in [Hua 2020a]. The sketch is as follow: First, it is easy to verify that $\rho(\boldsymbol{\mathcal{B}}_e) \leq 1$ if $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$ and $\|\boldsymbol{\mathcal{H}}_e\|_2 \leq 1$. Then assume that $\boldsymbol{\mathcal{B}}_e$ has an eigenvalue equals to one, it will contradict the condition $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$ and $\|\boldsymbol{\mathcal{H}}\|_2 \leq 1$. Finally, it can be concluded that the matrix $\boldsymbol{\mathcal{B}}_e$ cannot have an eigenvalue equals to 1, i.e., $\rho(\boldsymbol{\mathcal{B}}_e) < 1$. ■

Remark 3.2 *Notice that, the stability condition in Chapter 2 requires $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$ and $\|\boldsymbol{\mathcal{H}}\|_2 \leq 1$. In the non-local constraints case, in addition, it requires $\|\boldsymbol{\mathcal{H}}_e\|_2 \leq 1$ where $\boldsymbol{\mathcal{H}}_e$ is a variant form of $\boldsymbol{\mathcal{H}}$ due to delayed information. This indicates that the delayed information has an impact on convergence condition, and it needs to be more careful to choose proper step-size μ and parameter η to ensure stability. It is also worth noting that, in other works, e.g. [Hua 2020a], where delayed information is also used but aggregated by other forms, the delayed information there does not affect convergence condition.*

Provided that $\boldsymbol{\mathcal{B}}_e$ is stable, we arrive at the following statement of convergence in the mean sense:

Theorem 3.1 (Convergence in the mean) *Assume that data model (3.2) and Assumption 3.1 hold. Then, for any initial condition, algorithm (3.10) converges in the mean as*

$i \to \infty$ if $\boldsymbol{\mathcal{B}}_e$ is stable. A sufficient step-size condition for ensuring the mean stability is:

$$0 < \mu < \min\Big\{\frac{2}{\lambda_{\max}(\boldsymbol{R}_{x,k})}, \frac{2}{\eta \cdot \lambda_{\max}(\boldsymbol{\mathcal{D}}^{\top}\boldsymbol{\mathcal{D}})}\Big\},\ k = 1, \ldots, N \tag{3.37}$$

and $\{\mu, \eta\}$ should also ensure $\|\boldsymbol{\mathcal{H}}_e\|_2 \leq 1$.

In this case, the asymptotic mean bias is given by:

$$\mathbb{E}\widetilde{\boldsymbol{w}}_e(\infty) = \lim_{i\to\infty} \mathbb{E}\widetilde{\boldsymbol{w}}_e(i) = \mu(\boldsymbol{I}_{MH} - \boldsymbol{\mathcal{B}}_e)^{-1}(\boldsymbol{r}_{\eta,e} + \eta \boldsymbol{f}_{\eta,e}). \tag{3.38}$$

Proof. The convergence of mean error recursions (3.33) requires the matrix $\boldsymbol{\mathcal{B}}$ to be a stable matrix. According to Lemma 3.1, a sufficient condition is to choose μ and η that satisfy $\|\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x\|_2 < 1$, $\|\boldsymbol{\mathcal{H}}\|_2 \leq 1$ and $\|\boldsymbol{\mathcal{H}}_e\|_2 \leq 1$. From the proof of Theorem 2.1, step-size given by (3.37) ensures the first two conditions. However, for given $\{\mu, \eta\}$ which ensure $\|\boldsymbol{\mathcal{H}}\|_2 \leq 1$ may not ensure $\rho(\boldsymbol{\mathcal{H}}_e) = \rho([\boldsymbol{\mathcal{I}} - \mu\eta\,\boldsymbol{\mathcal{C}}_{\mathcal{D},e}]) \leq 1$, it further requires to check that $\{\mu, \eta\}$ ensure the stability of $\boldsymbol{\mathcal{H}}_e$. ■

Using similar arguments, we find that the mean error recursion with respect to $\boldsymbol{w}^{\star}$ evolves according to:

$$\mathbb{E}\widetilde{\boldsymbol{w}}'_e(i+1) = \boldsymbol{\mathcal{B}}_e \mathbb{E}\widetilde{\boldsymbol{w}}'_e(i) + \mu \boldsymbol{r}_{s,e}. \tag{3.39}$$

where $\widetilde{\boldsymbol{w}}'_e(i)$ is the extended error vector w.r.t. $\boldsymbol{w}^{*}$ defined similarly as $\mathbb{E}\widetilde{\boldsymbol{w}}_e(i)$, and

$$\boldsymbol{r}_{s,e} \triangleq \boldsymbol{\mathcal{H}}_e \boldsymbol{\mathcal{R}}_{x,e} \boldsymbol{w}_e^{\delta\prime}, \qquad (MH \times MH) \tag{3.40}$$

$$\boldsymbol{w}_e^{\delta\prime} \triangleq \boldsymbol{1}_H \otimes \boldsymbol{w}^{\delta\prime}, \qquad (MH \times 1) \tag{3.41}$$

$$\boldsymbol{w}^{\delta\prime} \triangleq \boldsymbol{w}^{\star} - \boldsymbol{w}^{o}. \qquad (M \times 1) \tag{3.42}$$

When $\boldsymbol{\mathcal{B}}_e$ is stable, we obtain:

$$\mathbb{E}\widetilde{\boldsymbol{w}}'_e(\infty) = \lim_{i\to\infty} \mathbb{E}\widetilde{\boldsymbol{w}}'_e(i) = \mu(\boldsymbol{I}_{MH} - \boldsymbol{\mathcal{B}}_e)^{-1}\boldsymbol{r}_{s,e}. \tag{3.43}$$

We observe that, when $\boldsymbol{w}^{o}$ satisfies the constraints, we have $\boldsymbol{w}^{o} = \boldsymbol{w}^{o}(\eta) = \boldsymbol{w}^{\star}$, $\boldsymbol{r}_{\eta,e} = \boldsymbol{0}$, $\boldsymbol{f}_{\eta,e} = \boldsymbol{0}$, and $\boldsymbol{r}_{s,e} = \boldsymbol{0}$. In this case, the asymptotic mean biases $\mathbb{E}\widetilde{\boldsymbol{w}}_e(\infty)$ and $\mathbb{E}\widetilde{\boldsymbol{w}}'_e(\infty)$ reduce to zero.

3.3.3 Mean-square-error behavior analysis

We shall evaluate the weighted variance $\mathbb{E}\{\|\widetilde{\boldsymbol{w}}_e(i+1)\|^2_{\boldsymbol{\Sigma}}\}$ where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix that we are free to choose. Let $\boldsymbol{\sigma} \triangleq \mathrm{vec}(\boldsymbol{\Sigma})$. In the following, we use the alternative notation $\|\boldsymbol{w}\|^2_{\boldsymbol{\sigma}}$ to refer to the same weighted squared norm $\|\boldsymbol{w}\|^2_{\boldsymbol{\Sigma}}$. Following the same

line of reasoning as in [Sayed 2013, Sayed 2014c] for single-task diffusion strategies, and extending the arguments to our multitask scenario, we find the variance relation:

$$\begin{aligned}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i+1)\|^2_{\boldsymbol{\Sigma}} &= \mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|^2_{\boldsymbol{\Sigma}'} + \mu^2\mathbb{E}\|\boldsymbol{g}_e(i)\|^2_{\boldsymbol{\Sigma}} + \mu^2\mathbb{E}\|\boldsymbol{r}_{\eta,e}(i)\|^2_{\boldsymbol{\Sigma}} + \mu^2\eta^2\|\boldsymbol{f}_{\eta,e}\|^2_{\boldsymbol{\Sigma}} \\ &\quad + 2\mu\mathbb{E}\{\boldsymbol{r}^\top_{\eta,e}(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\widetilde{\boldsymbol{w}}_e(i)\} + 2\mu\eta\boldsymbol{f}^\top_{\eta,e}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\mathbb{E}\widetilde{\boldsymbol{w}}_e(i) + 2\mu^2\eta\boldsymbol{f}^\top_{\eta,e}\boldsymbol{\Sigma}\boldsymbol{r}_{\eta,e}\end{aligned} \tag{3.44}$$

with $\boldsymbol{\Sigma}'$ given by

$$\boldsymbol{\Sigma}' = \mathbb{E}\big\{\boldsymbol{\mathcal{B}}^\top_e(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e(i)\big\}. \tag{3.45}$$

It can be verified that

$$\boldsymbol{\sigma}' \triangleq \mathrm{vec}(\boldsymbol{\Sigma}') = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma} \tag{3.46}$$

where $\boldsymbol{\mathcal{F}}$ is the $(MH)^2 \times (MH)^2$ matrix given by:

$$\boldsymbol{\mathcal{F}} \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top_e(i) \otimes \boldsymbol{\mathcal{B}}^\top_e(i)\} \tag{3.47}$$

$$\approx \boldsymbol{\mathcal{B}}^\top_e \otimes \boldsymbol{\mathcal{B}}^\top_e. \qquad \text{(small step-size)} \tag{3.48}$$

As explained before, this approximation is reasonable by ignoring terms depending on higher order power of the step-sizes. Moreover, the small step-size condition is prevalent in the stochastic gradient approximation literature and diffusion strategies [Sayed 2003, Sayed 2008, Chen 2015a, Plata-Chaves 2015, Nassif 2016b].

Remark 3.3 *Note that, unlike Chapter 2, we use the traditional Kronecker product, see Appendix 3.A. Recall that in Chapter 2, to deduce the mean-square performance w.r.t. $\boldsymbol{w}^o_\eta$ and $\boldsymbol{w}^*$, it requires to evaluate the explicit expression of $\boldsymbol{\mathcal{F}}$ which is of dimension $M^2 \times M^2$. While in this Chapter, the matrix $\boldsymbol{\mathcal{F}}$ is of dimension $(MH)^2 \times (MH)^2$, it would be much more involved to evaluate in block Kronecker product. In the sequel, we show that how to proceed the analysis by using traditional Kronecker product. Simulation results in Section 3.4 will validate that the derived theoretical models match well the simulation curves.*

We move to evaluate each term on the RHS of (3.44). The first term can be rewritten as

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|^2_{\boldsymbol{\Sigma}'} = \mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}. \tag{3.49}$$

The second term can be evaluated as

$$\mu^2\mathbb{E}\|\boldsymbol{g}_e(i)\|^2_{\boldsymbol{\Sigma}} = \mu^2\mathbb{E}\{\boldsymbol{g}^\top_e(i)\boldsymbol{\Sigma}\boldsymbol{g}_e(i)\} = \mu^2\mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}_e) \overset{(3.84)}{=} \mu^2[\mathrm{vec}(\boldsymbol{\mathcal{G}}^\top_e)]^\top\boldsymbol{\sigma}. \tag{3.50}$$

where $\boldsymbol{\mathcal{G}}_e$ is the $H \times H$ block matrix whose each block is of dimension $M \times M$ given by:

$$\boldsymbol{\mathcal{G}}_e \triangleq \mathbb{E}\{\boldsymbol{g}_e(i)\boldsymbol{g}_e^{\top}(i)\} = \boldsymbol{\mathcal{H}}_e \text{bdiag}\big\{\boldsymbol{\mathcal{T}}, \underbrace{\boldsymbol{0}_{M\times M}, \ldots, \boldsymbol{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\big\}\boldsymbol{\mathcal{H}}_e^{\top}, \quad (MH \times MH) \tag{3.51}$$

$$\boldsymbol{\mathcal{T}} \triangleq \text{bdiag}\{\sigma_{z,k}^2 \boldsymbol{R}_{x,k}\}_{k=1}^{N}. \quad (M \times M) \tag{3.52}$$

Similarly, the third term can be evaluated as

$$\mu^2 \mathbb{E}\|\boldsymbol{r}_{\eta,e}(i)\|_{\boldsymbol{\Sigma}}^2 = \mu^2 \mathbb{E}\{\boldsymbol{r}_{\eta,e}^{\top}(i)\boldsymbol{\Sigma}\boldsymbol{r}_{\eta,e}(i)\} = \mu^2 \text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{Q}}_e) \overset{(3.84)}{=} \mu^2 [\text{vec}(\boldsymbol{\mathcal{Q}}_e^{\top})]^{\top}\boldsymbol{\sigma}. \tag{3.53}$$

where $\boldsymbol{\mathcal{Q}}_e$ is given by:

$$\boldsymbol{\mathcal{Q}}_e \triangleq \mathbb{E}\{\boldsymbol{r}_{\eta,e}(i)\boldsymbol{r}_{\eta,e}^{\top}(i)\} = \boldsymbol{\mathcal{H}}_e \text{bdiag}\big\{\boldsymbol{\mathcal{T}}_1, \underbrace{\boldsymbol{0}_{M\times M}, \ldots, \boldsymbol{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\big\}\boldsymbol{\mathcal{H}}_e^{\top}, \quad (MH \times MH) \tag{3.54}$$

$$\boldsymbol{\mathcal{T}}_1 \triangleq \mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{\mathcal{K}}_1\boldsymbol{\mathcal{R}}_x(i)\}, \quad (M \times M) \tag{3.55}$$

$$\boldsymbol{\mathcal{K}}_1 \triangleq \boldsymbol{w}^{\delta}(\boldsymbol{w}^{\delta})^{\top}. \quad (M \times M) \tag{3.56}$$

Notice that, the evaluation of $\boldsymbol{\mathcal{T}}_1$ depends on higher order moments of the regressors. In the following, we shall evaluate $\boldsymbol{\mathcal{T}}_1$ when the regressors are zero-mean real Gaussian. Note that, for any square matrix $\boldsymbol{A}$ and zero-mean Gaussian regressors, we have:

$$\mathbb{E}\{\boldsymbol{x}_k(i)\boldsymbol{x}_k^{\top}(i)\boldsymbol{A}\boldsymbol{x}_{\ell}(i)\boldsymbol{x}_{\ell}^{\top}(i)\} = \boldsymbol{R}_{x,k}\boldsymbol{A}\boldsymbol{R}_{x,\ell} + \delta_{k,\ell}(\boldsymbol{R}_{x,k}\boldsymbol{A}^{\top}\boldsymbol{R}_{x,k} + \boldsymbol{R}_{x,k}\text{Tr}(\boldsymbol{R}_{x,k}\boldsymbol{A})). \tag{3.57}$$

Using (3.57), it can be verified that $\boldsymbol{\mathcal{T}}_1$ can be expressed as:

$$\boldsymbol{\mathcal{T}}_1 = \boldsymbol{\mathcal{R}}_x\boldsymbol{\mathcal{K}}_1\boldsymbol{\mathcal{R}}_x + \sum_{k=1}^{N}\Big(\boldsymbol{\mathcal{S}}_k(\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k})\boldsymbol{\mathcal{K}}_1^{\top}(\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k})\boldsymbol{\mathcal{S}}_k + \boldsymbol{\mathcal{S}}_k(\boldsymbol{I}_N \otimes \boldsymbol{R}_{x,k})\boldsymbol{\mathcal{Z}}_k\boldsymbol{\mathcal{S}}_k\Big), \tag{3.58}$$

where $\boldsymbol{\mathcal{S}}_k$ is an $N \times N$ block diagonal matrix whose k-th diagonal entry is $\boldsymbol{I}_{M_k}$ and zeros elsewhere, matrix $\boldsymbol{\mathcal{Z}}_k$ is an $N \times N$ block matrix with the (k,ℓ)-th block given by:

$$[\boldsymbol{\mathcal{Z}}_k]_{k,\ell} = [\text{vec}(\boldsymbol{R}_{x,k})]^{\top}\text{vec}([\boldsymbol{\mathcal{K}}_1]_{k,\ell})\boldsymbol{I}_{M_k}. \tag{3.59}$$

The fourth term is:

$$\mu^2\eta^2\|\boldsymbol{f}_{\eta,e}\|_{\boldsymbol{\Sigma}}^2 = \mu^2\eta^2\mathbb{E}\{\boldsymbol{f}_{\eta,e}^{\top}\boldsymbol{\Sigma}\boldsymbol{f}_{\eta,e}\} = \mu^2\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{f}_{\eta,e}\boldsymbol{f}_{\eta,e}^{\top}) \overset{(3.84)}{=} \mu^2\eta^2[\text{vec}(\boldsymbol{f}_{\eta,e}\boldsymbol{f}_{\eta,e}^{\top})]^{\top}\boldsymbol{\sigma} \tag{3.60}$$

The fifth term $2\mu\mathbb{E}\{\boldsymbol{r}_{\eta,e}(i)^{\top}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\widetilde{\boldsymbol{w}}_e(i)\}$ is evaluated as follows:

$$2\mu\mathbb{E}\{\boldsymbol{r}_{\eta,e}^{\top}(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\widetilde{\boldsymbol{w}}_e(i)\} = 2\mu\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\boldsymbol{\mathcal{P}}(i)) = 2\mu[\text{vec}(\boldsymbol{\mathcal{P}}^{\top}(i))]^{\top}\boldsymbol{\sigma} \tag{3.61}$$

The time dependent $H \times H$ block matrix $\boldsymbol{\mathcal{P}}(i)$ is given by:

$$\boldsymbol{\mathcal{P}}(i) \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}_e(i)\widetilde{\boldsymbol{w}}_e(i)\boldsymbol{r}_{\eta,e}^{\top}(i)\} = \boldsymbol{\mathcal{H}}_e \mathrm{diag}\{\boldsymbol{\mathcal{T}}_2(i), \underbrace{\boldsymbol{0}_{M\times M}, \ldots, \boldsymbol{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\}\boldsymbol{\mathcal{H}}_e^{\top}, \ (MH \times MH) \tag{3.62}$$

with

$$\boldsymbol{\mathcal{T}}_2(i) \triangleq \mathbb{E}\{\boldsymbol{\mathcal{R}}_x(i)\boldsymbol{\mathcal{K}}_2(i)(\boldsymbol{I}_M - \mu\boldsymbol{\mathcal{R}}_x(i))\}, \qquad (M \times M) \tag{3.63}$$

$$\boldsymbol{\mathcal{K}}_2(i) \triangleq \mathbb{E}\{\widetilde{\boldsymbol{w}}(i)\}(\boldsymbol{w}^{\delta})^{\top}. \qquad (M \times M) \tag{3.64}$$

Note that, $\mathbb{E}\{\widetilde{\boldsymbol{w}}(i)\}$ can be obtained from the mean error recursion (3.33). Then $\boldsymbol{\mathcal{T}}_2(i)$ can be evaluated as it has been done for $\boldsymbol{\mathcal{T}}_1$. The last two terms on the RHS of (3.44) can be evaluated as

$$2\mu\eta\boldsymbol{f}_{\eta,e}^{\top}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\mathbb{E}\widetilde{\boldsymbol{w}}_e(i) = 2\mu\eta\mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_e\mathbb{E}\widetilde{\boldsymbol{w}}_e(i)\boldsymbol{f}_{\eta,e}^{\top}) = 2\mu\eta[\mathrm{vec}(\boldsymbol{f}_{\eta,e}\mathbb{E}\widetilde{\boldsymbol{w}}_e^{\top}(i)\boldsymbol{\mathcal{B}}_e^{\top})]^{\top}\boldsymbol{\sigma}, \tag{3.65}$$

$$2\mu^2\eta\boldsymbol{f}_{\eta,e}^{\top}\boldsymbol{\Sigma}\boldsymbol{r}_{\eta,e} = 2\mu^2\eta\mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{r}_{\eta,e}\boldsymbol{f}_{\eta,e}^{\top}) = 2\mu^2[\mathrm{vec}(\boldsymbol{f}_{\eta,e}\boldsymbol{r}_{\eta,e}^{\top})]^{\top}\boldsymbol{\sigma}. \tag{3.66}$$

Let us define the $MH \times MH$ time dependent matrix $\boldsymbol{\mathcal{Y}}_e(i)$ given by:

$$\boldsymbol{\mathcal{Y}}_e(i) \triangleq \mu^2\boldsymbol{\mathcal{G}}_e^{\top} + \mu^2\boldsymbol{\mathcal{Q}}_e^{\top} + 2\mu\boldsymbol{\mathcal{P}}^{\top}(i) + \mu^2\eta^2\boldsymbol{f}_{\eta,e}\boldsymbol{f}_{\eta,e}^{\top} + 2\mu\eta\boldsymbol{f}_{\eta,e}\mathbb{E}\widetilde{\boldsymbol{w}}_e^{\top}(i)\boldsymbol{\mathcal{B}}_e^{\top} + 2\mu^2\eta\boldsymbol{f}_{\eta,e}\boldsymbol{r}_{\eta,e}^{\top}, \tag{3.67}$$

therefore, the variance relation (3.44) can be expressed as:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}^2 + [\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(i))]^{\top}\boldsymbol{\sigma}. \tag{3.68}$$

Theorem 3.2 (Mean-square stability) *Assume that data model (3.2) and Assumption 3.1 hold. Assume further that the step-size μ is sufficiently small. Then, for any initial condition, algorithm (3.10) is mean-square stable if the error recursion (3.33) is mean stable and the matrix $\boldsymbol{\mathcal{F}}$ defined by (3.47) is stable. Using the approximation (3.48) and property (3.83), the condition for mean stability in Theorem 3.1 will also ensure mean-square stability.*

Proof. Iterating (3.68) starting from $i = 0$ and an initial condition $\widetilde{\boldsymbol{w}}_e(0) = \boldsymbol{1}_H \otimes (\boldsymbol{w}^o(\eta) - \boldsymbol{w}(0))$, we obtain:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\widetilde{\boldsymbol{w}}_e(0)\|_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}}^2 + \sum_{j=0}^{i}[\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(i-j))]^{\top}\boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}. \tag{3.69}$$

Similar with the proof in Theorem 2.2, the mean square error will converge when $\boldsymbol{\mathcal{F}}$ is stable and the mean error is bounded. When the step-size is sufficiently small, $\boldsymbol{\mathcal{F}}$ can

be approximated as $\boldsymbol{\mathcal{B}}_e^\top \otimes \boldsymbol{\mathcal{B}}_e^\top$, stabilty of $\boldsymbol{\mathcal{F}}$ coincides to stabilty of $\boldsymbol{\mathcal{B}}_e$, therefore mean stability condition also ensures mean-square stability. ■

Theorem 3.3 (Learning curve) *Assume the same settings as Theorem 3.2. The learning curve* $\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i+1)\|_{\boldsymbol{\sigma}}^2$ *ends up evolving according to the following recursion:*

$$\begin{aligned}\mathbb{E}\{\|\widetilde{\boldsymbol{w}}_e(i+1)\|_{\boldsymbol{\sigma}}^2\} = \mathbb{E}\{\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\sigma}}^2\} + \big[\mathrm{vec}(\widetilde{\boldsymbol{w}}_e(0)\widetilde{\boldsymbol{w}}_e^\top(0))\big]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(MH)^2})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma} \\ + [\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(i))]^\top\boldsymbol{\sigma} + \boldsymbol{\gamma}_e(i)\boldsymbol{\sigma},\end{aligned} \tag{3.70}$$

where $\boldsymbol{\gamma}_e(i+1)$ *is an* $(MH)^2 \times 1$ *vector that can be evaluated from* $\boldsymbol{\gamma}_e(i)$ *according to:*

$$\boldsymbol{\gamma}_e(i+1) = [\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(i))]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(MH)^2}) + \boldsymbol{\gamma}_e(i)\boldsymbol{\mathcal{F}}, \tag{3.71}$$

with $\boldsymbol{\gamma}_e(0) = \boldsymbol{0}_{(MH)^2\times 1}$.

Proof. The argument is similar to the proof in Theorem 2.3. ■

Let

$$\boldsymbol{\Sigma}_{net} \triangleq \mathrm{bdiag}\{\boldsymbol{I}_M, \underbrace{\boldsymbol{0}_{M\times M}, \ldots, \boldsymbol{0}_{M\times M}}_{\text{in total } H\text{-1 block items}}\}, \qquad (MH \times MH) \tag{3.72}$$

$$\boldsymbol{\sigma}_{net} = \mathrm{vec}(\boldsymbol{\Sigma}_{net}). \qquad ((MH)^2 \times 1) \tag{3.73}$$

The *network* MSD learning curve, defined as $\zeta(i) \triangleq \frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}(i)\|^2 = \frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\Sigma}_{net}}^2$, can be obtained by replacing $\boldsymbol{\sigma}$ with $\boldsymbol{\sigma}_{net}$ in (3.70):

$$\begin{aligned}\zeta(i+1) = \zeta(i) + \frac{1}{N}\big[\mathrm{vec}(\widetilde{\boldsymbol{w}}_e(0)\widetilde{\boldsymbol{w}}_e^\top(0))\big]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(MH)^2})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}_{net} \\ + \frac{1}{N}[\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(i))]^\top\boldsymbol{\sigma}_{net} + \frac{1}{N}\boldsymbol{\gamma}_e(i)\boldsymbol{\sigma}_{net},\end{aligned} \tag{3.74}$$

Theorem 3.4 (Steady-state performance) *Assume the same settings as Theorem 3.2. The steady-state performance* $\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\sigma}}^2$ *of the algorithm (3.10) is given by*

$$\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\sigma}}^2 = [\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(\infty))]^\top(\boldsymbol{I}_{(MH)^2} - \boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}, \tag{3.75}$$

where $\boldsymbol{\mathcal{Y}}_e(\infty)$ *can be obtained from (3.38) and (3.67).*

Proof. The argument is similar to the proof in Theorem 2.4. ■

Define steady-state *network* MSD as $\zeta^\star \triangleq \lim_{i\to\infty}\frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|^2 = \lim_{i\to\infty}\frac{1}{N}\mathbb{E}\|\widetilde{\boldsymbol{w}}_e(i)\|_{\boldsymbol{\Sigma}_{net}}^2$, it can be obtained by:

$$\zeta^\star = \frac{1}{N}[\mathrm{vec}(\boldsymbol{\mathcal{Y}}_e(\infty))]^\top(\boldsymbol{I}_{(MH)^2} - \boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}_{net}. \tag{3.76}$$

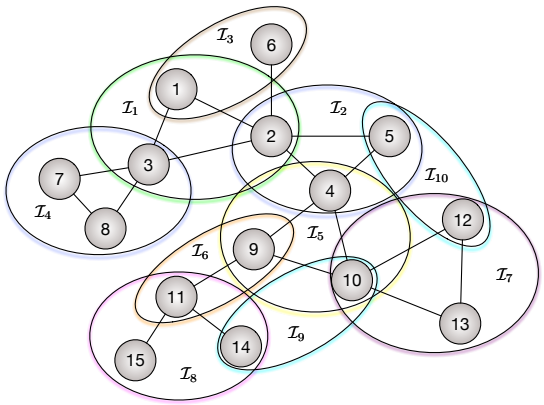


Figure 3.1: Multitask MSE network with constraints.

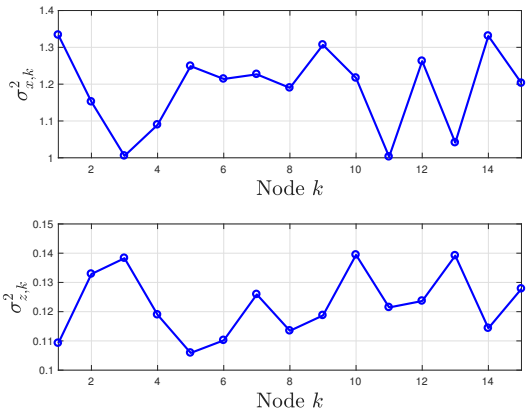


Figure 3.2: Regressors and noise variances.

The behavior in the mean and mean-square sense w.r.t. $\boldsymbol{w}^o(\eta)$ can be predicted by the theoretical findings (3.33), (3.38), (3.70), and (3.75). Finally, the transient and steady-state behavior of $\mathbb{E}\|\widetilde{\boldsymbol{w}}'_e(i)\|^2_{\boldsymbol{\sigma}}$ w.r.t. $\boldsymbol{w}^*$ can be derived from the following relation:

$$\mathbb{E}\{\|\widetilde{\boldsymbol{w}}'_e(i)\|^2_{\boldsymbol{\sigma}}\} = \mathbb{E}\{\|\widetilde{\boldsymbol{w}}_e(i)\|^2_{\boldsymbol{\sigma}}\} + 2\mathbb{E}\{\widetilde{\boldsymbol{w}}_e(i)\}\boldsymbol{\Sigma}\boldsymbol{w}_{d,e} + \|\boldsymbol{w}_{d,e}\|^2_{\boldsymbol{\Sigma}} \tag{3.77}$$

with $\boldsymbol{w}_{d,e} = \mathbf{1}_H \otimes (\boldsymbol{w}^\star - \boldsymbol{w}^o(\eta))$.

3.4 Simulations

In this section, we provide experimental results to illustrate the convergence of algorithm (3.10) and validate our theoretical models. We considered a network of 15 nodes with the topology and the constraints shown in Fig. 3.1. We randomly sampled 10 linear

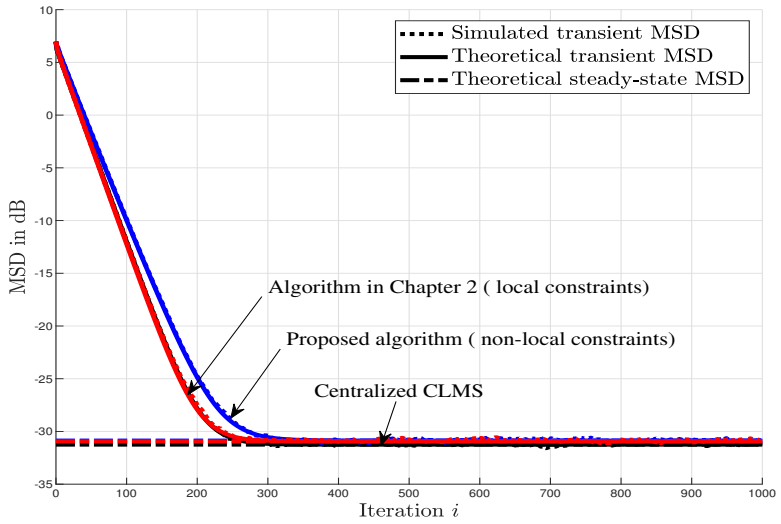


Figure 3.3: MSD comparison (perfect model scenario).

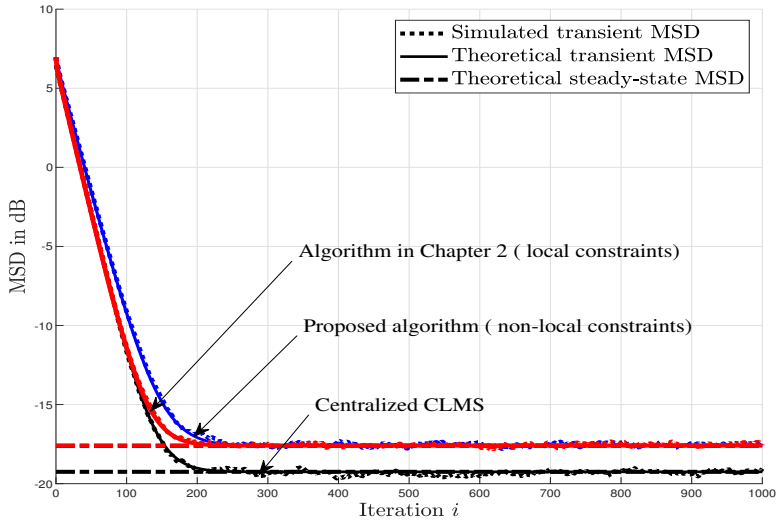


Figure 3.4: MSD comparison (imperfect model scenario).

constraints of the form $\sum_{\ell\in\mathcal{I}_p} d_{p\ell}\boldsymbol{w}_\ell = b_p \cdot \mathbf{1}_2$, where the coefficients $d_{p\ell}$ and b_p were randomly chosen from $\{-2,-1,1,2\}$. The regression vectors $\boldsymbol{x}_k(i)$ were zero-mean Gaussian with covariance matrix $\boldsymbol{R}_{x,k} = \sigma_{x,k}^2 \boldsymbol{I}_2$. The noises $z_k(i)$ were zero-mean i.i.d. Gaussian random variables independent of any other signal with variances $\sigma_{z,k}^2$. The variances $\sigma_{x,k}^2$ and $\sigma_{z,k}^2$ used in the simulations are shown in Fig. 3.2. The results were averaged over 200

Monte-Carlo runs.

We considered two scenarios: i) the parameter vector $\boldsymbol{w}^o = \boldsymbol{w}_o$ where $\boldsymbol{w}_o$ satisfies the constraints, i.e. $\boldsymbol{w}^o = \boldsymbol{w}^\star$(Fig.3.3); ii) $\boldsymbol{w}^o$ does not satisfy the constraints, specifically, we perturbed it as $\boldsymbol{w}^o = \boldsymbol{w}_o + \boldsymbol{u}$ where $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (Fig.3.4). The step-size μ was set to 0.02. We compared our algorithm (3.10) with the centralized CLMS algorithm [Nassif 2017b, Frost 1972]. Furthermore, for comparison purposes, we assumed additional links connecting nodes 1 to 6, 5 to 12, 10 to 14, and 14 to 15. In this case, the constraints are local and the algorithm derived in Chapter 2 [Hua 2017b] can be applied. We set $\mu = 0.018$ for the centralized CLMS algorithm and for algorithm [Hua 2017b] (with additional links) so that their steady-state MSD match. Observe that the simulation results match well the theoretical models. Furthermore, our algorithm performs well in the mean-square-error compared to the centralized solution. Finally, observe that, as expected, the delays emerging from the multi-hop protocols required in non-local constraints scenarios will lead to a slower convergence rate.

3.5 Conclusion

We proposed a distributed multitask algorithm for estimating multiple parameter vectors that are coupled through non-local linear equality constraints. Based on the penalty method, we solved the original constrained problem by approximating it into an unconstrained one. A multi-hop relay protocol was employed in order to deal with the non-local constraints and to devise a distributed algorithm. The stochastic behavior of the algorithm in the mean and in the mean-square-error sense was studied. Simulation results were conducted to show the effectiveness of the proposed method and to validate our theoretical performance analysis.

In Chapter 2 and 3, we considered multitask estimation problems where agents are interested in estimating different but linearly related tasks. The relations between agents are known beforehand. In next Chapter, we will consider the multitask estimation problem in Graph Signal Processing (GSP) applications where the agents do not know in prior that their neighbors are estimating the same task.

3.A Kronecker product

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be denote $N \times N$ and $M \times M$ matrices, respectively, whose individual (i, j)-th entries are denoted by a_{ij} and b_{ij}. Their Kronecker product is denoted by $\boldsymbol{A} \otimes \boldsymbol{B}$ and is

defined as the $NM \times NM$ matrix given by [Horn 2012]:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1N}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{21}\boldsymbol{B} & \dots & a_{2N}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}\boldsymbol{B} & a_{N1}\boldsymbol{B} & \dots & a_{NN}\boldsymbol{B} \end{bmatrix}. \tag{3.78}$$

For any two vectors $\{\boldsymbol{x}, \boldsymbol{y}\}$, we have:

$$\text{vec}(\boldsymbol{x}\boldsymbol{y}^\top) = \boldsymbol{y} \otimes \boldsymbol{x}, \tag{3.79}$$

where vec$(\cdot)$ operator transforms a matrix into a vector by stacking the columns of the matrix on top of each other.

For matrices $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}\}$ with compatible dimensions and column vectors $\{\boldsymbol{x}, \boldsymbol{y}\}$, we list some useful properties of traditional Kronecker products:

$$(\boldsymbol{A} + \boldsymbol{B}) \otimes \boldsymbol{C} = (\boldsymbol{A} \otimes \boldsymbol{C}) + (\boldsymbol{B} \otimes \boldsymbol{C}) \tag{3.80}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C} \otimes \boldsymbol{B}\boldsymbol{D}) \tag{3.81}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B})^\top = \boldsymbol{A}^\top \otimes \boldsymbol{B}^\top \tag{3.82}$$

$$\{\lambda(\boldsymbol{A} \otimes \boldsymbol{B})\} = \{\lambda_i(\boldsymbol{A})\lambda_j(\boldsymbol{B})\}_{i=1,j=1}^{N,M} \tag{3.83}$$

$$\text{Tr}(\boldsymbol{A}\boldsymbol{B}) = [\text{vec}(\boldsymbol{B}^\top)]^\top \text{vec}(\boldsymbol{A}) \tag{3.84}$$

$$\text{vec}(\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}) = (\boldsymbol{B}^\top \otimes \boldsymbol{A})\text{vec}(\boldsymbol{C}). \tag{3.85}$$

Chapter 4

Online Distributed Learning over Graphs with Multitask Graph-Filter Models

Contents

In this chapter, we are interested in adaptive and distributed estimation of graph filters from streaming data. We formulate this problem as a consensus estimation problem over graphs, which can be addressed with diffusion LMS strategies. Most popular graph-shift operators such as those based on the graph Laplacian matrix, or the adjacency matrix, are not energy preserving. This may result in an ill-conditioned estimation problem, and reduce the convergence speed of the distributed algorithms. To address this issue and improve the transient performance, we introduce a preconditioned graph diffusion LMS algorithm. We also propose a computationally efficient version of this algorithm by approximating the Hessian matrix with local information. Performance analyses in the mean and mean-square sense are provided. Finally, we consider a more general problem where the filter coefficients to estimate may vary over the graph. To avoid a large estimation bias, we introduce an unsupervised clustering method for splitting the global estimation problem into local ones. Numerical results show the effectiveness of the proposed algorithms and validate the theoretical results. The main results established in this chapter were published in [Hua 2018a, Hua 2018b, Hua 2020b].

4.1 Introduction

Data generated by network-structured applications often exhibit non-Euclidean structures, which make traditional signal processing techniques inefficient to analyze them. In contrast, graph signal processing (GSP) provides useful tools to analyze and process signals on graphs. They represent them as samples at the vertices of a possibly weighted graph, and use algebraic and spectral properties of the graph to study the signals. These graph representations are useful in applications ranging from social and economic networks to smart grids [Djurić 2018, Ortega 2018, Sandryhaila 2014a, Shuman 2013]. Recent results in the area include sampling [Chen 2015b, Anis 2016, Tsitsvero 2016], filtering [Sandryhaila 2013], and inference and learning [Nassif 2019, Defferrard 2016, Gama 2018b, Anis 2019], to cite a few.

In order to cope with graph signals, GSP relies on two ingredients: the graph shift operator (GSO) on one hand, which accounts for the topology of the graph, and the graph Fourier transform (GFT) on the other hand, which allows to represent graph signals in the graph frequency domain. Built upon the definition of the GFT, graph filters play a central role in processing graph signal spectra by selectively amplifying or attenuating frequency components. Various architectures of graph filters have been proposed in the

literature, including finite impulse response (FIR) [Shuman 2013, Sandryhaila 2013] and infinite impulse response (IIR) [Shi 2015, Liu 2019] filters. From a perspective of scalability, and considering energy constraints and band-limited communications that may be encountered in large networks of distributed nodes such as sensor networks, significant efforts have been made recently to derive distributed graph filters. These filtering procedures allow each node to exchange only local information with its neighboring nodes [Loukas 2015, Isufi 2017b, Segarra 2017, Coutino 2019, Shuman 2018].

Much of the GSP literature has focused on static graph signals, that is, signals that need not evolve with time. However, a wide spectrum of network-structured problems requires adaptation to time-varying dynamics. Sensor networks, social networks, vehicular networks, communication networks, and power grids are some typical examples. Prior to the more recent GSP literature, many earlier works on adaptive networks have addressed problems dealing with this challenge by developing processing strategies that are well-suited to data streaming into graphs; see, e.g., [Sayed 2013, Sayed 2014b, Sayed 2014c]. Several diffusion strategies have been introduced, and their performance studied in various situations, such as diffusion LMS [Lopes 2008], RLS [Cattivelli 2008], and APA [Li 2009]. By referring to the problem of estimating an optimal parameter vector at a node as a "task", and depending on the relations between the parameter vectors across the entire network, adaptive networks can be divided into single or multitask networks. In single-task networks, all nodes estimate the same parameter vector. Typical works include [Sayed 2013, Sayed 2014b, Sayed 2014c]. With multitask networks, multiple but related parameter vectors are inferred simultaneously in a cooperative manner, so as to improve the estimation accuracy by using the similarities between tasks [Chen 2015a, Chen 2014b, Nassif 2016a, Nassif 2016b].

In this Chapter, we are interested in online learning of linear graph models for representing streaming graph signals in a distributed manner. We focus on diffusion strategies because they are scalable, robust, and enable network adaptation and learning. Recently, some research works have considered time-varying graph signals. An adaptive graph signal reconstruction algorithm based on the LMS is proposed in [Di Lorenzo 2016] but it operates in a centralized manner. In [Mei 2017], the authors focus on estimating a network structure capturing the dependencies among time series in the form of a possibly directed, weighted adjacency matrix. A causal autoregressive process is introduced in the time series to capture the intuition that information travels over the network at some fixed speed. In [Isufi 2019], vector autoregressive (VAR) and vector autoregressive moving average (VARMA) models are proposed for predicting time-varying processes on graphs. Joint time-vertex stationarity is introduced for time-varying graph signals in [Loukas 2016, Loukas 2017], and a joint

time-vertex harmonic analysis for graph signals is proposed in [Grassi 2018]. It is shown that joint stationarity facilitates estimation or recovery tasks when compared to purely time or graph methods.

All the aforementioned works focus on centralized solutions whereas distributed algorithms may be more appropriate within the context of big data applications. In [Isufi 2016, Isufi 2017c], the behavior of some distributed graph filters on time-varying graph signals is studied. Considering graph signal sampling and reconstruction, several distributed algorithms have been proposed to track time-varying band-limited graph signals, e.g., LMS-based algorithms in [Di Lorenzo 2017], RLS-based methods in [Di Lorenzo 2018], Kalman-based methods in [Isufi 2017a], and kernel-based algorithms in [Romero 2017]. In [Qiu 2017], the authors are interested in time-varying graph signals with temporal smoothness prior. They devise distributed gradient descent algorithms to reconstruct the signals. Most of these works assume that the graph signals are band-limited. Another limitation is that the graph Fourier decomposition (eigenvectors) is needed beforehand, which is impractical for large networks.

Recently, several works successfully applied adaptive algorithms to graph signals. In [Di Lorenzo 2016, Di Lorenzo 2017, Di Lorenzo 2018] for instance, graph Fourier coefficients are learned from streaming graph signals under band-limited assumption to perform adaptive reconstruction and tracking of time-varying graph signals from partial observations. In this work, we are interested in online distributed learning of linear graph models without assumption of band-limited processes. We use graph filter models in the time-vertex domain where there is no need to decompose the graph shift operator. The formulated optimization problem relies on minimizing a global cost consisting of the aggregate sum of individual costs at each vertex. To address this problem, we blend concepts from adaptive networks [Sayed 2014a] and graph signal processing to devise graph diffusion LMS strategies. Considering that most popular shift operators are not energy preserving and may result in a slow convergence speed, we introduce a preconditioned optimization strategy to improve the transient performance. As this may lead to an increased computational complexity, we further propose a computationally efficient algorithm. Explicit theoretical performance analyses in the mean and mean-square-error sense are provided. We also give alternative theoretical results that are tractable for large networks. Simulation results show the efficiency of the proposed algorithms and validate the theoretical models. Finally, we extend these node-invariant filter models to more flexible ones where each node in the graph seeks to estimate a local node-varying graph filter. This allows us to exploit more degrees of freedom in the filter coefficients to better model graph signals. We introduce an unsu-

pervised clustering strategy to determine which nodes in the graph share the same graph filter and may collaborate to estimate its parameters. Numerical results on a real-word dataset illustrate the efficiency of the proposed methods.

The rest of this Chapter is organized as follows. Section 4.2 formulates the problem and provides the centralized solution. Section 4.3 introduces the distributed algorithms, namely, the graph diffusion LMS strategy and its preconditioned counterparts. Section 4.4 provides their theoretical analyses in the mean and the mean-square sense. A clustering method is devised to estimate local node-varying graph filters in Section 4.5. Numerical results in Section 4.6 show the effectiveness of these algorithms and validate the theoretical models.

4.2 Problem formulation and centralized solution

We consider an undirected, weighted and connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \boldsymbol{W})$ of N nodes, where $\mathcal{N} = \{1, 2, \ldots, N\}$ is the set of nodes, and $\mathcal{E}$ is the set of edges such that if node k is connected to node ℓ, then $(k, \ell) \in \mathcal{E}$. We denote by $\mathcal{N}_k$ the neighborhood of node k including itself, namely, $\mathcal{N}_k = \{\ell : \ell = k \vee (\ell, k) \in \mathcal{E}\}$. Matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is the adjacency matrix whose (k, ℓ)-th entry $w_{k\ell}$ assigns a weight to the relation between vertices k and ℓ. Since the graph is undirected, $\boldsymbol{W}$ is a symmetric matrix. The degree matrix $\boldsymbol{D} \triangleq \text{diag}(\boldsymbol{W}\boldsymbol{1})$ is a diagonal matrix whose i-th diagonal entry is the degree of node i, which is equal to the sum of all the weights of edges incident at node i. The combinatorial Laplacian matrix is defined as $\boldsymbol{L} \triangleq \boldsymbol{D} - \boldsymbol{W}$ which is a real, symmetric, positive semi-definite matrix. We further assume that the graph is endowed with a graph shift operator defined as an $N \times N$ shift matrix $\boldsymbol{S}$ whose entry $s_{k\ell}$ can be non-zero only if $k = \ell$ or $(k, \ell) \in \mathcal{E}$. Although the shift matrix can be any matrix that captures the graph topology for the problem at hand [Ortega 2018], popular choices are the graph Laplacian matrix [Shuman 2013], the adjacency matrix [Sandryhaila 2013], and their normalized counterparts. A graph signal is defined as $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample associated with node k. Let $\boldsymbol{x}(i)$ denote the graph signal at time i. Operation $\boldsymbol{S}\boldsymbol{x}$ is called graph shift. It can be performed locally at each node k by linearly combining the samples x_ℓ from its neighboring nodes $\ell \in \mathcal{N}_k$.

4.2.1 Graph filter and data model

In this Chapter, we focus on linear shift-invariant FIR graph filters $\boldsymbol{H} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ of order M, which are polynomials of the graph-shift operator [Sandryhaila 2013]:

$$\boldsymbol{H} \triangleq \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m, \tag{4.1}$$

where $\boldsymbol{h}^o = \{h_m^o\}_{m=0}^{M-1}$ are the scalar filter coefficients. With the definitions of graph signal and graph shift operator, one common filtering model assumes that the filtered graph signal $\boldsymbol{y}(i)$ is generated from the input graph signal $\boldsymbol{x}(i)$ as follows [Sandryhaila 2013, Nassif 2017a]:

$$\boldsymbol{y}(i) = \boldsymbol{H}\boldsymbol{x}(i) + \boldsymbol{v}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i) + \boldsymbol{v}(i), \tag{4.2}$$

where $\boldsymbol{v}(i) = [v_1(i), \ldots, v_N(i)]^\top \in \mathbb{R}^N$ denotes an i.i.d. zero-mean noise independent of any other signal and with covariance matrix $\boldsymbol{R}_v = \text{diag}\{\sigma_{v,k}^2\}_{k=1}^N$. For each node k, the filtered signal $y_k(i)$ can be computed by linearly combining the input signals at nodes located in an $(M-1)$-hop neighborhood [Shuman 2013]. This model however assumes the instantaneous diffusion of information over the graph since $\boldsymbol{S}^m\boldsymbol{x}(i)$ translates $\boldsymbol{x}(i)$ without time delay. As this assumption may appear as a serious limitation, we consider the more general model embedding the temporal dimension as follows [Isufi 2016, Nassif 2018]:

$$\boldsymbol{y}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i-m) + \boldsymbol{v}(i). \tag{4.3}$$

Observe that the input signal $\boldsymbol{x}(i)$ in (4.2) has been replaced by $\boldsymbol{x}(i-m)$ in (4.3), i.e., the m-hop spatial shift $\boldsymbol{S}^m$ is now carried out in m time slots. This model implements an FIR filter in both graph domain and temporal domain. By retaining the following shifted signals that form the $N \times M-1$ matrix:

$$\boldsymbol{X}_r = \Big[\boldsymbol{x}(i-1), \boldsymbol{S}\boldsymbol{x}(i-2), \ldots, \boldsymbol{S}^{M-2}\boldsymbol{x}(i-M+1)\Big], \tag{4.4}$$

note that only one shift is required at time instant i to produce the filtered signal $\boldsymbol{y}(i)$. This means that $y_k(i)$ can be computed using only local information available within the one-hop neighborhood of node k. Let $\boldsymbol{Z}(i)$ denote the $N \times M$ matrix given by:

$$\boldsymbol{Z}(i) \triangleq \left[\boldsymbol{x}(i),\, \boldsymbol{S}\boldsymbol{x}(i-1), \ldots, \boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)\right], \tag{4.5}$$

then model (4.3) can be written alternatively as:

$$\boldsymbol{y}(i) = \boldsymbol{Z}(i)\boldsymbol{h}^o + \boldsymbol{v}(i) \tag{4.6}$$

From model (4.6), sample $y_k(i)$ at node k can be written as:

$$y_k(i) = \boldsymbol{z}_k^{\top}(i)\boldsymbol{h}^o + v_k(i), \tag{4.7}$$

where $\boldsymbol{z}_k^{\top}(i)$ is the k-th row of $\boldsymbol{Z}(i)$ given by:

$$\boldsymbol{z}_k(i) \triangleq \text{col}\big\{[\boldsymbol{x}(i)]_k,\, [\boldsymbol{S}\boldsymbol{x}(i-1)]_k, \ldots, [\boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)]_k\big\}. \tag{4.8}$$

Observe in (4.7) that each node shares the same filter coefficient vector $\boldsymbol{h}^o$. The corresponding graph filter (4.1) is referred to as *node-invariant* graph filter. A more flexible model was introduced in [Segarra 2017], and called a *node-variant* graph filter. It allows the filter coefficients to vary across nodes as follows:

$$\boldsymbol{H} \triangleq \sum_{m=0}^{M-1} \text{diag}(\boldsymbol{h}^{(m)})\boldsymbol{S}^m, \tag{4.9}$$

with $\boldsymbol{h}^{(m)} \in \mathbb{R}^N$. If $\boldsymbol{h}^{(m)} = h_m\mathbf{1}$ for all m, model (4.9) reduces to the node-invariant model (4.1). Otherwise, each node applies different weights to the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$. Then $y_k(i)$ in (4.7) can be re-written as:

$$y_k(i) = \boldsymbol{z}_k^{\top}(i)\boldsymbol{h}_k^o + v_k(i), \tag{4.10}$$

where $\boldsymbol{h}_k^o \in \mathbb{R}^M$ is the filter coefficient vector at node k collected into $\boldsymbol{h}^{(m)}$, i.e., $[\boldsymbol{h}_k^o]_m = [\boldsymbol{h}^{(m)}]_k$. In this work, we seek to estimate $\boldsymbol{h}_k^o$ from the filtered graph signal $y_k(i)$ and inputs $\boldsymbol{z}_k(i)$, in a distributed, collaborative and adaptive manner. Distributed algorithms such as the diffusion LMS exist in the literature to address single-task and multitask inference problems with similar data models as (4.7) or (4.10). In this work, however, regressors $\boldsymbol{z}_k(i)$ in (4.8) are raised from graph shifted signals. This Chapter aims to exploit the graph shift structure in the regression data and incorporate it into the formulation of the distributed algorithm – see expression (4.22) further ahead. In the sequel, first, we shall study the case where the filter coefficients are common for all nodes, i.e. $\boldsymbol{h}_k^o = \boldsymbol{h}^o, \forall k \in \mathcal{N}$. We shall show how to estimate $\boldsymbol{h}^o$ from streaming data $\{\boldsymbol{y}(i), \boldsymbol{x}(i)\}$ in a centralized way and then, in a distributed way. Next, we shall assume that there are clusters of nodes within the graph, and each node in the same cluster uses the same filter. This model is called a *hybrid node-varying* graph filter [Gama 2018a]. We shall introduce an unsupervised clustering method to allow each node to identify which neighboring nodes it should collaborate with.

4.2.2 Centralized solution

Before introducing the distributed method, we first introduce the centralized solution. Consider the data model (4.6) and assume that $\{\boldsymbol{y}(i), \boldsymbol{x}(i), \boldsymbol{v}(i)\}$ are zero-mean jointly

wide-sense stationary random processes. Estimating $\boldsymbol{h}^o$ from $\{\boldsymbol{y}(i), \boldsymbol{Z}(i)\}$ can be performed by solving the following problem:

$$\boldsymbol{h}^o = \arg\min_{\boldsymbol{h}} J(\boldsymbol{h}), \tag{4.11}$$

where $J(\boldsymbol{h})$ denotes the mean-square-error criterion:

$$\begin{aligned} J(\boldsymbol{h}) &= \mathbb{E}\|\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}\|^2 \\ &= \mathbb{E}\{\boldsymbol{y}^\top(i)\boldsymbol{y}(i)\} - 2\boldsymbol{h}^\top \boldsymbol{r}_{Zy} + \boldsymbol{h}^\top \boldsymbol{R}_Z \boldsymbol{h}, \end{aligned} \tag{4.12}$$

and the $M \times M$ matrix $\boldsymbol{R}_Z$ and the $M \times 1$ vector $\boldsymbol{r}_{Zy}$ are given by:

$$\boldsymbol{R}_Z \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)\}, \quad \boldsymbol{r}_{Zy} \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{y}(i)\}. \tag{4.13}$$

By setting the gradient vector of $J(\boldsymbol{h})$ to zero, the optimal parameter vector $\boldsymbol{h}^o$ can be found by solving:

$$\boldsymbol{R}_Z \boldsymbol{h}^o = \boldsymbol{r}_{Zy}. \tag{4.14}$$

It can be verified that the (m, n)-th entry of $\boldsymbol{R}_Z$ is given by:

$$[\boldsymbol{R}_Z]_{m,n} = \mathrm{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1} \boldsymbol{R}_x(m-n)\right) \tag{4.15}$$

where $\boldsymbol{R}_x(m) \triangleq \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i-m)\}$. The m-th entry of the vector $\boldsymbol{r}_{Zy}$ is given by:

$$[\boldsymbol{r}_{Zy}]_m = \mathrm{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{R}_{xy}(m)\right), \tag{4.16}$$

with $\boldsymbol{R}_{xy}(m) \triangleq \mathbb{E}\{\boldsymbol{y}(i)\boldsymbol{x}^\top(i-m)\}$ denoting cross correlation function, which is assumed independent of time i.

Instead of solving (4.14), $\boldsymbol{h}^o$ can be sought iteratively by using the gradient-descent method:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu\big[\boldsymbol{r}_{Zy} - \boldsymbol{R}_Z \boldsymbol{h}(i)\big], \tag{4.17}$$

with $\mu > 0$ a small step-size. Since the statistical moments are usually unavailable beforehand, one way is to replace them by the instantaneous approximations $\boldsymbol{R}_Z \approx \boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)$ and $\boldsymbol{r}_{Zy} \approx \boldsymbol{Z}^\top(i)\boldsymbol{y}(i)$. This yields the LMS graph filter:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu \boldsymbol{Z}^\top(i)\big[\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}(i)\big]. \tag{4.18}$$

This stochastic-gradient algorithm is referred to as the *centralized graph-LMS* algorithm. In this centralized setting, each node k at each time instant i sends its data $\{x_k(i), y_k(i)\}$ to a fusion center which will update $\boldsymbol{h}(i)$ according to (4.18). Note that the step-size μ in (4.18) must satisfy $0 < \mu < \frac{2}{\lambda_{\max}(\boldsymbol{R}_Z)}$ in order to guarantee stability in the mean under certain independence conditions on the data [Sayed 2008].

4.3 Diffusion LMS strategies over graph signals

In this section, we seek to estimate the graph filter coefficients in a distributed fashion. First, we review the graph diffusion LMS strategy [Nassif 2018]. Then, a preconditioned algorithm is proposed to improve the transient performance. We also devise a computationally efficient counterpart of this algorithm.

4.3.1 Graph diffusion LMS

Consider the local data model (4.7) at node k. It is worth noting that, by retaining the past shifted signals $\{[\boldsymbol{S}^{m-1}\boldsymbol{x}(i-m)]_\ell : m = 1, \ldots, M-1\}$ at each node ℓ in the network from previous iterations, $\boldsymbol{z}_k(i)$ can be computed locally at node k from its one-hop neighbors at each iteration i. Let $\boldsymbol{R}_{z,k} \triangleq \mathbb{E}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}$ denote the $M \times M$ covariance matrix with (m,n)-th entry given by [Nassif 2018]:

$$[\boldsymbol{R}_{z,k}]_{m,n} = \mathrm{Tr}\left([\boldsymbol{S}^{m-1})]_{k,\bullet}^\top [\boldsymbol{S}^{n-1}]_{k,\bullet}\boldsymbol{R}_x(m-n)\right). \tag{4.19}$$

Considering the local cost $J_k(\boldsymbol{h})$ at node k:

$$J_k(\boldsymbol{h}) = \mathbb{E}|y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}|^2, \tag{4.20}$$

the global cost (4.12) is now the aggregation of the local costs over the graph:

$$J(\boldsymbol{h}) = \sum_{k=1}^{N} J_k(\boldsymbol{h}). \tag{4.21}$$

In order to minimize (4.12) in a decentralized fashion, there are several useful techniques, e.g., incremental strategy [Bertsekas 1997], consensus strategy [Xiao 2005] and diffusion strategy [Sayed 2014c]. Diffusion strategies are attractive since they are scalable, robust, and enable continuous learning and adaptation. In particular, the adapt-then-combine (ATC) diffusion LMS takes the following form at node k [Nassif 2018]:

$$\boldsymbol{z}_k^\top(i) = \Big[x_k(i), \sum_{\ell\in\mathcal{N}_k} s_{k\ell}\,[\boldsymbol{z}_\ell(i-1)]_1, \ldots, \sum_{\ell\in\mathcal{N}_k} s_{k\ell}\,[\boldsymbol{z}_\ell(i-1)]_{M-1}\Big], \tag{4.22a}$$

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\boldsymbol{z}_k(i)\big[y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)\big], \tag{4.22b}$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1), \tag{4.22c}$$

where $\mu_k > 0$ is a local step-size parameter and $\{a_{\ell k}\}$ are non-negative combination coefficients chosen to satisfy:

$$a_{\ell k} > 0, \quad \sum_{\ell=1}^{N} a_{\ell k} = 1, \ \text{ and } \ a_{\ell k} = 0 \ \text{ if } \ \ell \notin \mathcal{N}_k. \tag{4.23}$$

This implies that the matrix $\boldsymbol{A}$ with (ℓ, k)-th entry $a_{\ell k}$ is a left-stochastic matrix, which means that the sum of each of its columns is equal to 1. In the first step (4.22a), each node k uses the first $M-1$ entries of $\boldsymbol{z}_\ell(i-1)$ from its one-hop neighbors and its own input sample $x_k(i)$ to compute $\boldsymbol{z}_k(i)$. Note that the first $M-1$ entries of $\boldsymbol{z}_k(i)$ then need to be retained for the next iteration. In the adaptation step (4.22b), each node k updates its local estimate $\boldsymbol{h}_k(i)$ to an intermediate estimate $\boldsymbol{\psi}_k(i+1)$. In the combination step (4.22c), node k aggregates all the intermediate estimates $\boldsymbol{\psi}_\ell(i+1)$ from its neighbors to obtain the updated estimate $\boldsymbol{h}_k(i+1)$.

4.3.2 Graph diffusion preconditioned LMS

The regressor $\boldsymbol{z}_k(i)$ used in the adaptation step (4.22b) results from shifted graph signals while the shift matrix $\boldsymbol{S}$ is not energy preserving in general [Gavili 2017]. This is due to the fact that the magnitude of the eigenvalues of the shift operator $\boldsymbol{S}$ are not uniformly equal to 1; the energy of the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$ changes exponentially with m. Thus, the eigenvalue spread of $\boldsymbol{R}_{z,k}$ may be large and the LMS update may suffer from slow convergence speed in this case [Sayed 2008]. To address this issue, albeit at an increased computational cost, we resort to a form of Newton's method. Focusing on the adaptation step, we have:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) - \mu_k[\nabla^2_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))]^{-1}[\nabla_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))], \tag{4.24}$$

where $\nabla^2_{\boldsymbol{h}} J_k(\cdot)$ denotes the Hessian matrix for $J_k(\cdot)$ and $\nabla_{\boldsymbol{h}} J_k(\cdot)$ is its gradient vector, if available.

For the quadratic cost function (4.20), expression (4.24) would lead to:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{R}^{-1}_{z,k}\big[\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)\big], \tag{4.25}$$

where $\boldsymbol{r}_{zy,k} = \mathbb{E}\{\boldsymbol{z}_k(i)y_k(i)\}$. Note that the second term on the RHS of (4.25) requires second-order moments. Since they are rarely available beforehand, we can replace $\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)$ by the instantaneous approximation:

$$\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i) \approx \boldsymbol{z}_k(i)e_k(i) \tag{4.26}$$

with $e_k(i) = y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)$. The adaptation step (4.25) becomes:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \widehat{\boldsymbol{R}}^{-1}_{z,k}(i)\boldsymbol{z}_k(i)e_k(i), \tag{4.27}$$

where $\widehat{\boldsymbol{R}}_{z,k}(i)$ is an estimate for $\boldsymbol{R}_{z,k}(i)$, which can possibly be obtained recursively:

$$\widehat{\boldsymbol{R}}_{z,k}(i) = (1-\mu)\,\widehat{\boldsymbol{R}}_{z,k}(i-1) + \mu\big[\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\big], \quad i \geq 1, \tag{4.28}$$

where μ is a small factor that can be chosen in $(0, 0.1]$ in practice. It can be verified that $\mathbb{E}\{\widehat{\boldsymbol{R}}_{z,k}(i)\} = \boldsymbol{R}_{z,k}$ is an unbiased estimate when $i \to \infty$. As discussed before, $\boldsymbol{S}$ may not be energy preserving and results in a large eigenvalue spread of $\boldsymbol{R}_{z,k}$, which may even be close to singular. The inverse $\boldsymbol{R}_{z,k}^{-1}$ would then be ill-conditioned and lead to undesirable effects. To address this problem, it is common to use regularization [Sayed 2008]. We obtain the diffusion LMS-Newton algorithm:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \big[\epsilon \boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1} \boldsymbol{z}_k(i) e_k(i), \tag{4.29a}$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_\ell(i+1), \tag{4.29b}$$

with $\epsilon \geq 0$ a small regularization parameter. Compared with the diffusion LMS algorithm (4.22), algorithm (4.29) requires first to recursively estimate the Hessian matrix according to (4.28) and then calculate $\big[\epsilon \boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1}$. This algorithm can lead to improved performance as shown in the sequel, but at the expense of additional computation cost.

In order to reduce the computational complexity of the LMS-Newton algorithm, we propose to use a preconditioning matrix $\boldsymbol{P}_k$ that does not depend on the graph signal $\boldsymbol{x}(i)$ in the adaptation step, instead of the Hessian matrix $\boldsymbol{R}_{z,k}$ or its estimate $\widehat{\boldsymbol{R}}_{z,k}$. Since the large eigenvalue spread of the input covariance matrix $\boldsymbol{R}_{z,k}$ results mainly from the shift matrix $\boldsymbol{S}$ and the filter order M, we construct an $M \times M$ preconditioning matrix $\boldsymbol{P}_k$ as follows:

$$\boldsymbol{P}_k \triangleq \operatorname{diag}\{\|[\boldsymbol{S}^{(m-1)}]_{k,\bullet}\|^2\}_{m=1}^{M}. \tag{4.30}$$

The rationale behind (4.30) is that, in the case where $\boldsymbol{x}(i)$ is i.i.d. with variance σ^2, it follows from (4.19) that $\boldsymbol{R}_{z,k} = \sigma^2 \boldsymbol{P}_k$. According to (4.30), matrix $\boldsymbol{P}_k$ does not depend on $\boldsymbol{x}(i)$ and can be evaluated beforehand at each node k during an initial step. Each node k only requires to know the edge weights in its M-hop neighborhood, which can be performed in a decentralized manner. Interestingly, $\boldsymbol{P}_k$ is a diagonal matrix, which means that the matrix product in the adaptation step does not require expensive matrix inversion. Following the same line of reasoning as for the Newton algorithm (4.29), a regularization term $\epsilon \boldsymbol{I}_M$ can be added to $\boldsymbol{P}_k$. This leads to:

$$\boldsymbol{D}_k = (\epsilon \boldsymbol{I}_M + \boldsymbol{P}_k)^{-1}. \tag{4.31}$$

We arrive at the following preconditioned graph diffusion LMS strategy:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{D}_k \boldsymbol{z}_k(i) e_k(i), \tag{4.32a}$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_\ell(i+1). \tag{4.32b}$$

At each iteration i, node k uses the local information to update the intermediate estimate $\boldsymbol{\psi}_k(i+1)$ in the adaptation step (4.32a). Then, in the combination step (4.32b), the intermediate estimates $\boldsymbol{\psi}_\ell(i+1)$ from the neighborhood of node k are combined to get $\boldsymbol{h}_k(i+1)$. Although the preconditioning matrix $\boldsymbol{D}_k$ is not the true Hessian matrix, we prove in Section 4.4 that the algorithm converges to the optimal solution $\boldsymbol{h}^o$ provided that it is stable.

4.3.3 Comparison with the graph diffusion LMS

We explain how preconditioning with (4.30) improves performance. For comparison purposes, let us first focus on the adaptation step of diffusion LMS. At each node k, the m-th entry of $\boldsymbol{h}_k(i)$ is updated as follows:

$$[\boldsymbol{\psi}_k(i+1)]_m = [\boldsymbol{h}_k(i)]_m + \mu_k \big[\boldsymbol{z}_k(i)e_k(i)\big]_m. \tag{4.33}$$

During the transient phase, the m-th entry $[\boldsymbol{h}_k(i)]_m$ exponentially converges to its optimal value with a time constant [Sayed 2008]:

$$\tilde{\tau}_m \approx \frac{1}{2\mu_k \lambda_m} \tag{4.34}$$

where λ_m denotes the m-th eigenvalue of $\boldsymbol{R}_{z,k}$. Given μ_k, the convergence rate of each entry of $\boldsymbol{h}_k(i)$ then depends on the corresponding eigenvalue. Disparity between entries increases as the eigenvalue spread defined as $\lambda_{\max}/\lambda_{\min}$ increases.

The preconditioning matrix $\boldsymbol{D}_k$ is diagonal at each node k, which means that the m-th entry of $\boldsymbol{h}_k(i)$ in (4.32a) converges to its optimal value with a time constant:

$$\tilde{\tau}_m \approx \frac{1}{2\mu_k d_{k,m} \lambda_m} \tag{4.35}$$

where $d_{k,m}$ denotes the m-th diagonal entry of $\boldsymbol{D}_k$. The convergence speed now depends on $d_{k,m}\lambda_m$. Considering the case where $\boldsymbol{P}_k$ is proportional to $\boldsymbol{R}_{z,k}$, then $d_{k,m}$ is inversely proportional to λ_m, which mitigates the effects of the eigenvalues spread. We shall analyze and illustrate in Section 4.4 and Section 4.6, respectively, how this preconditioning improves convergence speed in more general cases.

4.4 Performance analysis

We shall now analyze the stochastic behavior of the diffusion preconditioned LMS (PLMS) algorithm (4.32) in the sense of mean and mean-square error. We introduce the following weight error vectors at each node k:

$$\tilde{\boldsymbol{h}}_k(i) = \boldsymbol{h}^o - \boldsymbol{h}_k(i), \qquad \tilde{\boldsymbol{\psi}}_k(i) = \boldsymbol{h}^o - \boldsymbol{\psi}_k(i), \tag{4.36}$$

and we collect them across the nodes into the network weight error vectors :

$$\tilde{\boldsymbol{h}}(i) \triangleq \text{col}\{\tilde{\boldsymbol{h}}_1(i), \tilde{\boldsymbol{h}}_2(i), \ldots, \tilde{\boldsymbol{h}}_N(i)\}, \tag{4.37}$$

$$\tilde{\boldsymbol{\psi}}(i) \triangleq \text{col}\{\tilde{\boldsymbol{\psi}}_1(i), \tilde{\boldsymbol{\psi}}_2(i), \ldots, \tilde{\boldsymbol{\psi}}_N(i)\}. \tag{4.38}$$

We refer to *mean stability* of the error vector $\tilde{\boldsymbol{h}}(i)$ if the limit superior of $\mathbb{E}\tilde{\boldsymbol{h}}(i)$ is bounded. Furthermore, we will claim that the algorithm converges in the mean to the optimum if $\mathbb{E}\tilde{\boldsymbol{h}}(i)$ converges to zero as i tends to $+\infty$ regardless of the starting point. *Mean-square stability* refers to the case where the superior limit of $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2$ is bounded.

Let us introduce the following $N \times N$ block matrices with individual entries of size $M \times M$:

$$\boldsymbol{\mathcal{A}} \triangleq \boldsymbol{A} \otimes \boldsymbol{I}_M, \tag{4.39}$$

$$\boldsymbol{\mathcal{M}} \triangleq \text{bdiag}\{\mu_k \boldsymbol{I}_M\}_{k=1}^N, \tag{4.40}$$

$$\boldsymbol{\mathcal{D}} \triangleq \text{bdiag}\{\boldsymbol{D}_k\}_{k=1}^N. \tag{4.41}$$

The estimation error in (4.32a) can be written as:

$$e_k(i) = y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i) = \boldsymbol{z}_k^\top(i)\tilde{\boldsymbol{h}}_k(i) + v_k(i). \tag{4.42}$$

Subtracting $\boldsymbol{h}^o$ from both sides of (4.32a) and using the above relation, then stacking $\tilde{\boldsymbol{\psi}}_k(i)$ across the nodes, we obtain

$$\tilde{\boldsymbol{\psi}}(i+1) = \left(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z(i)\right)\tilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i), \tag{4.43}$$

where $\boldsymbol{\mathcal{R}}_z(i)$ is an $N \times N$ block matrix with entries of size $M \times M$ define as:

$$\boldsymbol{\mathcal{R}}_z(i) \triangleq \text{bdiag}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}_{k=1}^N, \tag{4.44}$$

and $\boldsymbol{p}_{zv}(i)$ is an $N \times 1$ block column vector with entries of size $M \times 1$ given by:

$$\boldsymbol{p}_{zv}(i) \triangleq \text{col}\{\boldsymbol{z}_k(i)v_k(i)\}_{k=1}^N. \tag{4.45}$$

Subtracting $\boldsymbol{h}^o$ from both sides of (4.32b), we obtain the block weight error vector:

$$\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{A}}^\top\tilde{\boldsymbol{\psi}}(i+1). \tag{4.46}$$

Finally, combing (4.43) and (4.46), the network weight error vector $\tilde{\boldsymbol{h}}(i)$ of algorithm (4.32) evolves according to:

$$\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}(i)\tilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i), \tag{4.47}$$

with

$$\boldsymbol{\mathcal{B}}(i) = \boldsymbol{\mathcal{A}}^\top\Big(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z(i)\Big). \tag{4.48}$$

To proceed with the analysis, we introduce the following assumption.

Assumption 4.1 (independent inputs) *The inputs $\boldsymbol{z}_k(i)$ arise from a zero-mean random process that is temporally white with $\boldsymbol{R}_{z,k} \succ 0$.*

A consequence of Assumption 4.1 is that $\boldsymbol{z}_k(i)$ is independent of $\boldsymbol{h}_\ell(j)$ for all ℓ and $j < i$. This independence assumption is not true in the current work. Two successive regressors $\boldsymbol{z}_k$ involve common entries that cannot be statistically independent as in a conventional FIR implementation. However, when the step-size is sufficiently small, conclusions derived under this assumption tend to be realistic. For more details and discussions, see [Sayed 2008, Section 16.4]. Since this assumption helps to simplify the derivations without constraining the conclusions, it is widely used in the literature of adaptive filters and adaptive networks [Sayed 2014c, Sayed 2008]. We shall see in Section 4.6 that the resulting expressions match well the simulation results for sufficiently small step-sizes.

4.4.1 Mean-error behavior analysis

Taking expectations of both sides of (4.47), using the fact that $\mathbb{E}\boldsymbol{p}_{zv}(i) = 0$, and applying Assumption 4.1, we find that the network mean error vector evolves according to:

$$\mathbb{E}\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}\,\mathbb{E}\tilde{\boldsymbol{h}}(i), \tag{4.49}$$

where:

$$\boldsymbol{\mathcal{B}} \triangleq \mathbb{E}\boldsymbol{\mathcal{B}}(i) = \boldsymbol{\mathcal{A}}^{\top}(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z), \tag{4.50}$$

$$\boldsymbol{\mathcal{R}}_z \triangleq \mathbb{E}\boldsymbol{\mathcal{R}}_z(i) = \mathrm{bdiag}\{\boldsymbol{R}_{z,k}\}_{k=1}^{N}. \tag{4.51}$$

Theorem 4.1 (Convergence in the mean) *Assume that data model (4.7) and Assumption 4.1 hold. Then, for any initial condition, algorithm (4.32) converges asymptotically in the mean toward the optimal vector $\boldsymbol{h}^o$ if, and only if, the step-sizes in $\boldsymbol{\mathcal{M}}$ are chosen to satisfy:*

$$\rho\Big(\boldsymbol{\mathcal{A}}^{\top}(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z)\Big) < 1, \tag{4.52}$$

where $\rho(\cdot)$ denotes the spectral radius of its matrix argument. In the case where the signal $\boldsymbol{x}(i)$ is i.i.d, a sufficient condition for (4.52) to hold is to choose μ_k such that:

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{D}_k\boldsymbol{R}_{z,k})}, \qquad k = 1, \ldots, N. \tag{4.53}$$

Proof. The weight error vector $\tilde{\boldsymbol{h}}(i)$ converges to zero if, and only if, the coefficient matrix $\boldsymbol{\mathcal{B}}$ in (4.49) is a stable matrix, namely, $\rho(\boldsymbol{\mathcal{B}}) < 1$. Since any induced matrix norm

is lower bounded by the spectral radius, we have the following relation in terms of block maximum norm [Sayed 2014c]:

$$\begin{aligned}\rho(\boldsymbol{\mathcal{B}}) &\leq \|\boldsymbol{\mathcal{A}}^{\top}(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z)\|_{b,\infty} \\ &\leq \|\boldsymbol{\mathcal{A}}^{\top}\|_{b,\infty} \cdot \|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty} \\ &= \|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty}, \end{aligned} \tag{4.54}$$

where the last equality follows from the fact that $\boldsymbol{\mathcal{A}}$ is left stochastic, which implies that $\|\boldsymbol{\mathcal{A}}^{\top}\|_{b,\infty} = 1$ from Lemma D.4 of [Sayed 2014c]. Matrix $\boldsymbol{\mathcal{R}}_z$ is block diagonal if $\boldsymbol{x}(i)$ is i.i.d. Since $\boldsymbol{\mathcal{D}}$ is also diagonal, their product is symmetric, and $\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z$ is a block diagonal symmetric matrix. Then, following Lemma D.5 of [Sayed 2014c], its block maximum norm agrees with its spectral radius:

$$\|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty} = \rho(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z). \tag{4.55}$$

Combining (4.54) and (4.55), we verify that condition (4.53) ensures the stability of $\boldsymbol{\mathcal{B}}$. ■

4.4.2 Mean-square-error behavior analysis

We shall now study the mean-square-error behavior of algorithm (4.32). Let $\boldsymbol{\Sigma}$ be any $NM \times NM$ positive semi-definite matrix that we are free to choose. The freedom in selecting $\boldsymbol{\Sigma}$ will allow us to derive different performance measures about the network and the nodes. We consider the weighted mean-square error vector, i.e., $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}}$, where $\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}} \triangleq \tilde{\boldsymbol{h}}^{\top}(i)\boldsymbol{\Sigma}\tilde{\boldsymbol{h}}(i)$. From Assumption 4.1 and $\mathbb{E}\boldsymbol{p}_{zv}(i) = \boldsymbol{0}$, using (4.47), we obtain the following variance relation:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\Sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}'} + \mathbb{E}\|\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}}, \tag{4.56}$$

where $\boldsymbol{\Sigma}' \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^{\top}(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\}$. Let $\boldsymbol{\sigma}$ denote the $(NM)^2 \times 1$ vector obtained by vectorizing matrix $\boldsymbol{\Sigma}$, namely, $\boldsymbol{\sigma} = \mathrm{vec}(\boldsymbol{\Sigma})$. With some abuse of notation, we shall use $\|\cdot\|^2_{\boldsymbol{\sigma}}$ to also refer to the quantity $\|\cdot\|^2_{\boldsymbol{\Sigma}}$ when it is more convenient. Let $\boldsymbol{\sigma}' = \mathrm{vec}(\boldsymbol{\Sigma}')$. Considering that $\mathrm{vec}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}) = (\boldsymbol{W}^{\top} \otimes \boldsymbol{U})\boldsymbol{\sigma}$, it can be verified that:

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma}, \tag{4.57}$$

where $\boldsymbol{\mathcal{F}}$ is the $(NM)^2 \times (NM)^2$ matrix given by

$$\begin{aligned}\boldsymbol{\mathcal{F}} \triangleq& \mathbb{E}\{\boldsymbol{\mathcal{B}}^{\top}(i) \otimes \boldsymbol{\mathcal{B}}^{\top}(i)\} \\ =& \Big(\boldsymbol{I}_{(NM)^2} - \boldsymbol{I}_{NM} \otimes \boldsymbol{\mathcal{R}}_z^{\top}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{R}}_z^{\top}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{NM} + \mathcal{O}(\boldsymbol{\mathcal{M}}^2)\Big)(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}), \end{aligned} \tag{4.58}$$

where $\mathcal{O}(\boldsymbol{\mathcal{M}}^2)$ denotes $\mathbb{E}\{\boldsymbol{\mathcal{R}}_z^\top(i)\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{\mathcal{R}}_z^\top(i)\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}}\}$, which depends on the square of the step-sizes, $\{\mu_k^2\}$. While we can continue the analysis by taking this factor into account as was done in other studies [Sayed 2003], it is sufficient for the exposition to focus on the case of sufficiently small step-sizes where terms involving higher powers of the step-sizes $\{\mu_k\}$ can be ignored. Following the same line of reasoning, for sufficiently small step-sizes $\{\mu_k\}$, $\boldsymbol{\mathcal{F}}$ can be approximated by:

$$\boldsymbol{\mathcal{F}} \approx \boldsymbol{\mathcal{B}}^\top \otimes \boldsymbol{\mathcal{B}}^\top. \tag{4.59}$$

The second term on the RHS of (4.56) can be written as:

$$\mathbb{E}\|\boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|_{\boldsymbol{\Sigma}}^2 = \mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}), \tag{4.60}$$

where

$$\boldsymbol{\mathcal{G}} \triangleq \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}, \tag{4.61}$$

$$\boldsymbol{\mathcal{S}} \triangleq \mathbb{E}\{\boldsymbol{p}_{zv}(i)\boldsymbol{p}_{zv}^\top(i)\} = \mathrm{bdiag}\{\boldsymbol{\sigma}_{v,k}^2 \boldsymbol{R}_{z,k}\}_{k=1}^N. \tag{4.62}$$

Using the property $\mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{W}) = [\mathrm{vec}(\boldsymbol{W}^\top)]^\top \boldsymbol{\sigma}$, combining (4.57) and (4.60), the variance relation (4.56) can be re-written as:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}^2 + [\mathrm{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top \boldsymbol{\sigma}. \tag{4.63}$$

Theorem 4.2 (Stability in the mean-square) *Assume that data model (4.7) and Assumption 4.1 hold. Algorithm (4.32) converges in the mean-square sense if the matrix $\boldsymbol{\mathcal{F}}$ in (4.58) is stable. Assume further that the step-sizes are sufficiently small such that (4.59) is a reasonable approximation. In that case, the stability of $\boldsymbol{\mathcal{F}}$ is ensured if $\boldsymbol{\mathcal{B}}$ is stable.*

Proof. Iterating (4.63) starting from $i = 0$, we obtain

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\tilde{\boldsymbol{h}}(0)\|_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}}^2 + [\mathrm{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top \sum_{j=0}^{i} \boldsymbol{\mathcal{F}}^j \boldsymbol{\sigma}, \tag{4.64}$$

with initial condition $\tilde{\boldsymbol{h}}(0) = \boldsymbol{h}^o - \boldsymbol{h}(0)$. Provided $\boldsymbol{\mathcal{F}}$ is stable, $\boldsymbol{\mathcal{F}}^i \to 0$ as $i \to \infty$, then the first item on the RHS of (4.64) converges to zero and the second item converges to a finite value. The weighted mean-square error converges to a finite value as $i \to \infty$ which implies that the algorithm (4.32) will converge in the mean-square sense if $\boldsymbol{\mathcal{F}}$ is stable. Under the sufficiently small step-sizes assumption where the higher-order terms of $\boldsymbol{\mathcal{F}}$ in (4.58) can be neglected, approximation (4.59) is reasonable. The eigenvalues of $\boldsymbol{\mathcal{F}}$

are all the products of the eigenvalues of $\boldsymbol{\mathcal{B}}$, which means that $\rho(\boldsymbol{\mathcal{F}}) = [\rho(\boldsymbol{\mathcal{B}})]^2$. It follows that $\boldsymbol{\mathcal{F}}$ is stable if $\boldsymbol{\mathcal{B}}$ is stable. Therefore, when the graph signal $\boldsymbol{x}(i)$ is i.i.d, according to Theorem 4.1, condition (4.53) ensures mean-square stability of the algorithm under the assumed approximation (4.59). ■

Theorem 4.3 (Network transient MSD) *Assume sufficiently small step-sizes that ensure mean and mean-square stability. The network transient mean-square deviation (MSD) defined as* $\zeta(i) = \frac{1}{N}\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2$ *evolves according to the following recursion for* $i \geq 0$*:*

$$\zeta(i+1) = \zeta(i) + \frac{1}{N}\Big([\mathrm{vec}(\tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0))]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(NM)^2}) + [\mathrm{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\Big)\boldsymbol{\mathcal{F}}^i\mathrm{vec}(\boldsymbol{I}_{NM}). \tag{4.65}$$

Proof. Comparing (4.64) at time $i+1$ and i, $\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}}$ is related to $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}}$ as follows:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}} + [\mathrm{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma} + [\mathrm{vec}(\tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0))]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(NM)^2})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}. \tag{4.66}$$

Substituting $\boldsymbol{\sigma}$ by $\frac{1}{N}\mathrm{vec}(\boldsymbol{I}_{NM})$ leads to (4.65). ■

Although expression (4.65) gives a compact form of the transient MSD model, it may not be practical to use since $\boldsymbol{\mathcal{F}}$ is of size $(NM)^2 \times (NM)^2$ and may become huge for large networks or high order filters. For example, in the simulation Section 4.6, we considered a network consisting of $N = 60$ nodes and a graph filter of degree $M = 5$. Matrix $\boldsymbol{\mathcal{F}}$ is of size 90000×90000 and requires a prohibitive amount of computational time and memory space (about 60GB in that case). To tackle this issue, we make use of the following properties of the Kronecker product:

$$\mathrm{vec}(\boldsymbol{X}\boldsymbol{Y}\boldsymbol{Z}) = (\boldsymbol{Z}^\top \otimes \boldsymbol{X})\mathrm{vec}(\boldsymbol{Y}) \tag{4.67}$$

$$\mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y}) = \big(\mathrm{vec}(\boldsymbol{Y}^\top)\big)^\top\mathrm{vec}(\boldsymbol{X}). \tag{4.68}$$

This leads to:

Corollary 4.1 (Alternative network transient MSD expression)

$$\zeta(i+1) = \zeta(i) + \frac{1}{N}\mathrm{Tr}\Big(\boldsymbol{\mathcal{B}}^i\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{B}}^i)^\top + \tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0)\left((\boldsymbol{\mathcal{B}}^{i+1})^\top\boldsymbol{\mathcal{B}}^{i+1} - (\boldsymbol{\mathcal{B}}^i)^\top\boldsymbol{\mathcal{B}}^i\right)\Big). \tag{4.69}$$

While the update with $\boldsymbol{\mathcal{F}}$ has a computation complexity of order $\mathcal{O}((NM)^2)$, using $\boldsymbol{\mathcal{B}}$ only requires matrix manipulations of order $\mathcal{O}(NM)$.

Corollary 4.2 (Network steady-state MSD) *Consider sufficiently small step-sizes to ensure mean and mean-square convergence. The network steady-state MSD is given by*

$$\zeta^{\star} = \frac{1}{N}[\mathrm{vec}(\boldsymbol{\mathcal{G}}^{\top})]^{\top}(\boldsymbol{I}_{(NM)^2} - \boldsymbol{\mathcal{F}})^{-1}\ \mathrm{vec}(\boldsymbol{I}_{NM}). \tag{4.70}$$

Proof. The network steady-state MSD is defined as:

$$\zeta^{\star} = \lim_{i\to\infty} \frac{1}{N}\mathbb{E}\{\|\tilde{\boldsymbol{h}}(i)\|^2\}. \tag{4.71}$$

If $\boldsymbol{\mathcal{F}}$ is stable, we obtain from (4.63) as $i \to \infty$:

$$\lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{(\boldsymbol{I}_{(NM)^2}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = [\mathrm{vec}(\boldsymbol{\mathcal{G}}^{\top})]^{\top}\boldsymbol{\sigma}. \tag{4.72}$$

We obtain (4.70) by substituting $\boldsymbol{\sigma}$ in (4.72) by $\frac{1}{N}(\boldsymbol{I}_{(NM)^2} - \boldsymbol{\mathcal{F}})^{-1}\ \mathrm{vec}(\boldsymbol{I}_{NM})$. ■

Following the same line of reasoning as for the transient MSD model, the steady-state MSD given by (4.70) can be equivalently expressed as:

Corollary 4.3 (Alternative steady-state MSD expression)

$$\zeta^{\star} = \frac{1}{N}\sum_{i=0}^{\infty} \mathrm{Tr}\Big(\boldsymbol{\mathcal{B}}^{i}\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{B}}^{i})^{\top}\Big) \tag{4.73}$$

This expression is obtained by a series expansion of (4.70). In practice, a limited number of iterations can be used instead of the upper limit index $i \to \infty$ to obtain an accurate result.

4.5 Unsupervised clustering for hybrid node-varying graph filter

In Section 4.3, we investigated the scenario where the nodes in a graph share a common filter coefficient vector. Now we extend this model to the more flexible case (4.10), which allows the filter coefficients to vary across the graph. We further assume that the graph is decomposed into Q clusters of nodes $\mathcal{C}_q$ and, within each cluster $\mathcal{C}_q$, there is a common filter coefficient vector $\boldsymbol{h}_q^o$ to estimate, namely,

$$\boldsymbol{h}_k^o = \boldsymbol{h}_q^o, \qquad \text{if } k \in \mathcal{C}_q. \tag{4.74}$$

We assume that there is no prior information on the clusters composition and that the nodes do not know which other nodes share the same estimation task. Applying the algorithm (4.32) within this context may result a bias due to aggregate intermediate estimates

from different data models. To address this issue, automatic network clustering strategies may be used [Chen 2015a, Zhao 2015c, Plata-Chaves 2016, Khawatmi 2017] in order to inhibit cooperation between nodes from different clusters. These methods are based on local stand-alone estimation strategies that may not be efficient for the current context. Basically, the polynomial form (4.1) of graph filters does not make the estimation of the filter coefficients reliable enough for the higher degrees. In the following, we tackle this problem by devising a clustering strategy based on the PLMS.

First, we introduce the $N \times N$ instantaneous clustering matrix $\boldsymbol{E}_i$, whose (ℓ, k)-th entry shows if node k believes at time i that its neighboring node ℓ belongs to the same cluster or not, namely,

$$[\boldsymbol{E}_i]_{\ell k} = \begin{cases} 1, \text{ if } \ell \in \mathcal{N}_k \text{ and } k \text{ believes that } \boldsymbol{h}_k^o = \boldsymbol{h}_\ell^o, \\ 0, \text{ otherwise.} \end{cases} \tag{4.75}$$

At each time instant i, node k infers which neighbors belong to its cluster based on the non-zeros entries of the k-th column of $\boldsymbol{E}_i$. We collect these entries into a set $\mathcal{N}_{k,i}$, so that node k only combines the intermediate estimates from its neighbors in $\mathcal{N}_{k,i}$. Condition (4.23) on the combination coefficients becomes:

$$a_{\ell k} > 0, \quad \sum_{\ell=1}^{N} a_{\ell k} = 1, \text{ and } a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_{k,i}, \tag{4.76}$$

Since the clustering information is unknown beforehand, we propose to learn $\boldsymbol{E}_i$ in an online way by computing a Boolean variable $b_{\ell k}(i)$ defined as follows:

$$b_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{\psi}_\ell(i+1) - \boldsymbol{h}_k(i)\|^2 \leq \beta, \\ 0, & \text{otherwise,} \end{cases} \tag{4.77}$$

with $\beta > 0$ a preset threshold. Variable $b_{\ell k}(i)$ is defined from the ℓ_2-norm distance between the estimates at two neighboring nodes. If this distance is smaller than the threshold β, the two nodes are then assigned to the same cluster. Note that the distance between $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ is used in (4.77), instead of the distance between $\boldsymbol{h}_\ell(i+1)$ and $\boldsymbol{h}_k(i+1)$, in order to merge the learning and the clustering processes. Consider the learning process at any node k defined in (4.32). Information about clusters is used in the combination step (4.32b), where $\mathcal{N}_k$ now denotes the neighboring nodes of k that share the same estimation task as node k. This information should be available as soon as possible in order to avoid estimation bias. Considering the distance between $\boldsymbol{h}_\ell(i+1)$ and $\boldsymbol{h}_k(i+1)$ to decide if nodes k and ℓ are in the same cluster would allow to update the composition of sets $\mathcal{N}_k$ and $\mathcal{N}_\ell$ in

the combination step (4.32b) used to calculate parameter vectors $\boldsymbol{h}_\ell(i+2)$ and $\boldsymbol{h}_k(i+2)$. This latency time can be shortened by considering the distance between $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ right after the adaptation step (4.32a), and using this information to define $\mathcal{N}_k$ in the combination step.

This strategy usually fails if left as is, because the estimation of higher-order coefficients is not reliable enough and results in bad clustering performance. We propose to estimate this distance from M_k principal components of the estimates. Because $\boldsymbol{R}_{z,k}$ cannot be reasonably used to perform a Principal Component Analysis (PCA) of the input data, as it is rarely available beforehand and would involve significant additional computational effort, we suggest instead using matrix $\boldsymbol{P}_k$. As for the PLMS, the rationale behind this is that $\boldsymbol{R}_{z,k} = \sigma^2 \boldsymbol{P}_k$ when $\boldsymbol{x}(i)$ is i.i.d. with variance σ^2. Another interest lies in that $\boldsymbol{P}_k$ is a diagonal matrix, which greatly simplifies calculations. Without loss of generality, consider that the diagonal entries of $\boldsymbol{P}_k$ are in decreasing order. Projecting data onto the first M_k principal axes then reduces to selecting their first M_k entries and set the other entries to zero. Dimension M_k can be determined by setting the ratio of explained variance to total variance to some desired level τ as follows:

$$\begin{aligned} \min \quad & M_k \\ \text{s. t.} \quad & \sum_{m=1}^{M_k} \hat{\pi}_{k,m} \geq \tau \end{aligned} \tag{4.78}$$

with $\hat{\pi}_{k,m} = [\boldsymbol{p}_k]_m / \mathrm{Tr}(\boldsymbol{P}_k)$ an approximation of the *proportion of total variance* [Jolliffe 2016]. In practice, we can use a predefined threshold $\tau \in [0.9, 1)$ to decide how many entries should be retained. The Boolean variable (4.77) becomes:

$$b_{\ell k}(i) = \begin{cases} 1, & \text{if } \frac{\|\boldsymbol{\psi}'_\ell(i+1) - \boldsymbol{h}'_k(i)\|^2}{\|\boldsymbol{h}'_k(i)\|^2} \leq \beta, \\ 0, & \text{otherwise,} \end{cases} \tag{4.79}$$

with $\boldsymbol{\psi}'_\ell(i+1)$ and $\boldsymbol{h}'_k(i)$ the first M_k entries of $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ respectively. Compared to (4.77), note that the distance in (4.79) that accounts for the similarity between $\boldsymbol{\psi}'_\ell(i+1)$ and $\boldsymbol{h}'_k(i)$ has been normalized. We suggest to choose $\beta \in (0, 0.01]$ to lower the false detection rate. To reduce noise effects, we further introduce a smoothing step:

$$t_{\ell k}(i) = \nu t_{\ell k}(i-1) + (1-\nu) b_{\ell k}(i), \tag{4.80}$$

where $t_{\ell k}(i)$ is a trust level, and ν is a forgetting factor in $(0, 1)$ to balance the past and present cluster assignments. Once the trust level $t_{\ell k}(i)$ exceeds a preset threshold θ, which

can be chosen in $[0.5, 1)$, node k concludes that node ℓ belongs to its cluster, that is,

$$[\boldsymbol{E}_i]_{\ell k} = \begin{cases} 1, & \text{if } t_{\ell k}(i) \geq \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.81}$$

Based on $[\boldsymbol{E}_i]_{\ell k}$, each node k determines at each time instant i those nodes ℓ that it believes they belong to the same cluster, updates the combination coefficients according to (4.76), and finally combines the estimates from its neighbors with (4.32b).

4.6 Numerical results

4.6.1 Experiment with i.i.d. input data

We first considered a zero-mean i.i.d. Gaussian graph signal $\boldsymbol{x}(i)$ with covariance $\boldsymbol{R}_x = \text{diag}\{\sigma^2_{x,k}\}_{k=1}^N$. Variances $\sigma^2_{x,k}$ were randomly generated from the uniform distribution $\mathcal{U}(1, 1.5)$. In this setting, the graph signal sample $x_k(i)$ was independent of $x_\ell(j)$ for all ℓ and $j \leq i$. We assumed the linear data model (4.7). The graph filter order was set to $M = 5$ and the coefficients h^o_m were randomly generated from the uniform distribution $\mathcal{U}(0, 1)$. Noise $\boldsymbol{v}(i)$ was zero-mean Gaussian with covariance $\boldsymbol{R}_v = \text{diag}\{\sigma^2_{v,k}\}_{k=1}^N$. Variances $\sigma^2_{v,k}$ were randomly generated from the uniform distribution $\mathcal{U}(0.1, 0.15)$. We considered

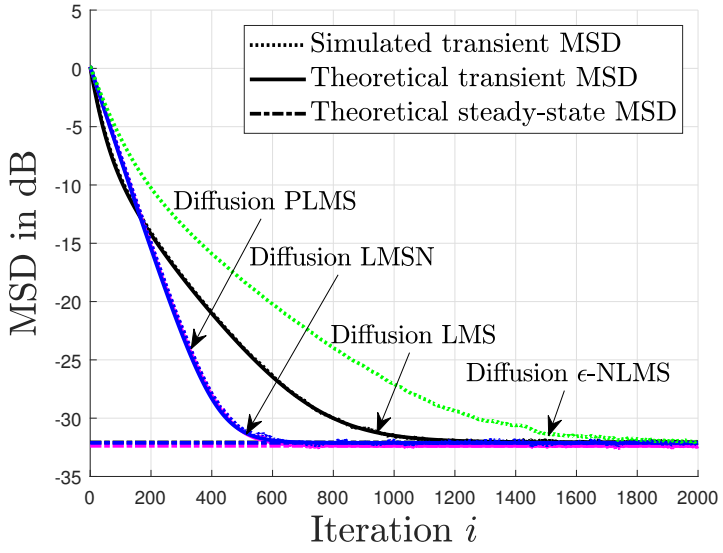
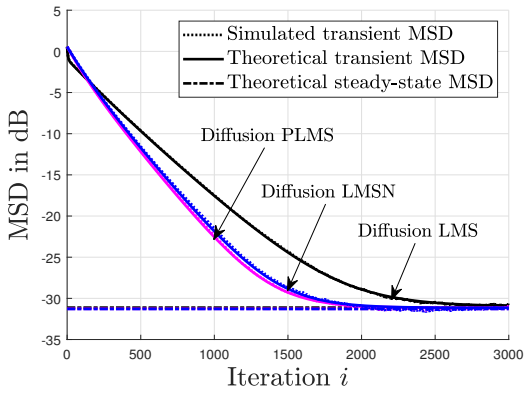
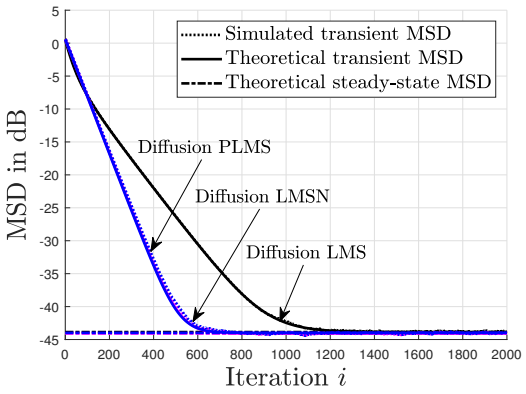


Figure 4.1: Network MSD performance with the Erdős-Rényi graph.

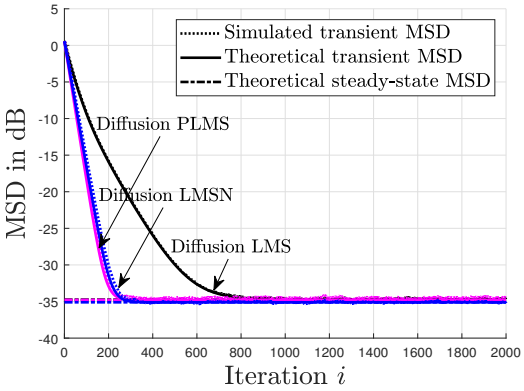
this data model with an Erdős-Rényi random graph and on a sensor network graph. Both



(a) Normalized Adjacency Matrix



(b) Normalized Laplacian Matrix



(c) Adjacency Matrix

Figure 4.2: Network MSD performance for different types of shift operators with the sensor network.

consisted of $N = 60$ nodes. The Erdős-Rényi random graph was generated in a similar construction as in [Mei 2017]. Namely, it was obtained by generating an $N \times N$ symmetric matrix $\boldsymbol{S}$ whose entries were governed by the Gaussian distribution $\mathcal{N}(0, 1)$, and then thresholding edges to be between 1.2 and 1.8 in absolute value. Then, the edges were soft thresholded by 1.1 to be between 0.1 and 0.7 in magnitude. The shift matrix $\boldsymbol{S}$ was normalized by 1.1 times its largest eigenvalue. The sensor network was generated by using GSPBOX [Perraudin 2014]. Each node was connected to its 5 nearest neighbors. The shift matrix was the normalized adjacency matrix, that is, $\boldsymbol{S} = \frac{\boldsymbol{W}}{1.1\lambda_{\max}(\boldsymbol{W})}$. In this case, all the eigenvalues of $\boldsymbol{S}$ are smaller than 1 and the energy of the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$ diminishes for large m. The smallest eigenvalue $\lambda_{\min}(\boldsymbol{R}_{z,k})$ was very small, and, for some node, it was close to 0.

With this setting, we compared the diffusion LMS algorithm (4.22), the diffusion LMS-Newton (LMSN) algorithm (4.29), and the diffusion preconditioned LMS (PLMS) algorithm (4.32). Simulated results were averaged over 500 Monte-Carlo runs. For the LMSN and PLMS algorithms, we set the regularization parameter as $\epsilon = 0.01$. We ran algorithms (4.22), (4.29) and (4.32) by setting $a_{\ell,k} = \frac{1}{|\mathcal{N}_k|}$ for $\ell \in \mathcal{N}_k$. We used a uniform step-size for all nodes, i.e., $\mu_k = \mu$ for all k. We also considered the ϵ-normalized LMS (ϵ-NLMS) method for comparison purposes. In this case, the adaptation step (4.32a) is substituted by:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \frac{\mu_k}{\|\boldsymbol{z}_k(i)\|^2 + \epsilon}\boldsymbol{z}_k(i)e_k(i). \tag{4.82}$$

With the Erdős-Rényi graph, we compared the performance of the LMS, PLMS, LMSN and ϵ-NLMS algorithms. We set $\mu = \{0.08, 0.008, 0.01, 0.05\}$, respectively. The network MSD performance of each algorithm is reported in Fig. 4.1. The theoretical transient and steady-state MSD are also reported. With the sensor network graph, we compared the performance of the LMS, PLMS and LMSN algorithms. We set $\mu = \{0.08, 0.005, 0.0055\}$, respectively. The performance of each algorithm is reported in Fig. 4.2(a). In Fig. 4.1, we observe that the diffusion ϵ-NLMS converged slower than all other algorithms. Observe that the diffusion LMSN and PLMS algorithms converged faster than the LMS algorithm for both graphs, and the diffusion PLMS performed similarly compared with the LMSN in terms of convergence rate. Also, note that the theoretical results match well the simulated curves.

In a second experiment, we considered the normalized graph Laplacian matrix $\boldsymbol{S} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$, and the adjacency matrix $\boldsymbol{W}$, as shift operators. For the normalized graph Laplacian, $\lambda_{\max}(\boldsymbol{R}_{z,k})$ was large for all nodes. Therefore, for the diffusion LMS algorithm, the step-size was chosen relatively small to guarantee convergence. We set $\mu = \{0.004, 0.01, 0.008\}$ for the LMS, LMSN and PLMS. The results are reported in Fig. 4.2(b). For the adjacency matrix, we used uniform step-sizes $\mu = \{0.02, 0.018\}$ for the LMSN and PLMS, respectively. The step-size was set to $\mu_k = 0.05 \cdot \frac{2}{\lambda_{\max}(\boldsymbol{R}_{z,k})}$ for each node k for the LMS update in order to achieve the same steady-state MSD. The results are reported in Fig. 4.2(c). We observe in Fig. 4.2 that the diffusion LMSN and PLMS algorithms converged faster than the LMS algorithm with the three graph shift operators. The PLMS algorithm achieved the same performance as the LMSN algorithm with a lower computational complexity.

4.6.2 Experiment with correlated input data

We tested the algorithms over the sensor network graph with correlated graph signals. We first considered a zero-mean i.i.d. Gaussian graph signal driven by a non-diagonal covariance matrix $\boldsymbol{R}_x$. This means that the input data were correlated over the vertex domain, but uncorrelated over time. Matrix $\boldsymbol{R}_x$ was generated as $\boldsymbol{R}_x = \boldsymbol{V}\text{diag}\{\sigma_{x,k}^2\}_{k=1}^N \boldsymbol{V}^\top$, with $\sigma_{x,k}^2$ randomly chosen from the uniform distribution $\mathcal{U}(1, 1.5)$ and $\boldsymbol{V}$ is the graph Fourier transform matrix. The graph shift operator was defined by the normalized adjacency matrix. The filter degree was set as $M = 3$. We observe in Fig. 4.3 that the proposed diffusion PLMS algorithm performed as well as the LMSN algorithm, and converged faster than the diffusion LMS algorithm.

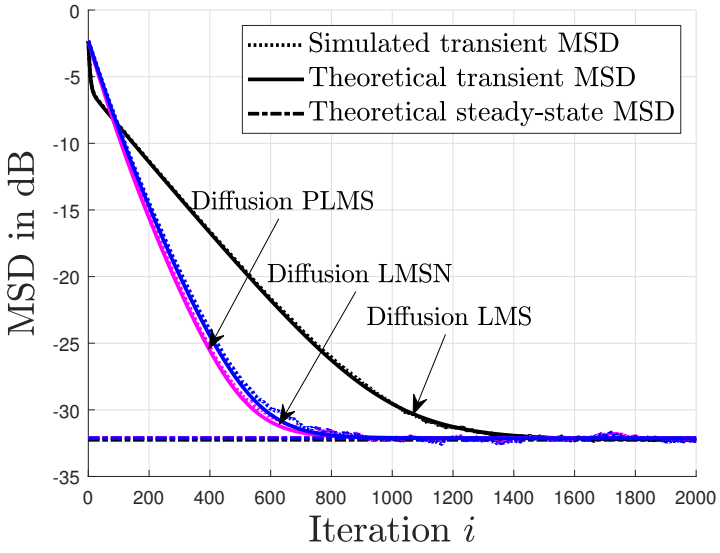


Figure 4.3: Network MSD performance with a vertex domain correlated input signal.

Next, we considered a graph signal $\boldsymbol{x}(i)$ that was correlated over both vertex and time domains. We assumed that $\boldsymbol{x}(i)$ is a Gaussian process with zero mean and covariance matrix $\boldsymbol{R}_x$ satisfying the discrete Lyapunov equation:

$$\boldsymbol{S}\boldsymbol{R}_x\boldsymbol{S}^\top - \boldsymbol{R}_x + \boldsymbol{I} = 0. \tag{4.83}$$

Graph signal sample $\boldsymbol{x}(i)$ was related to $\boldsymbol{x}(i-1)$ as follows:

$$\boldsymbol{x}(i) = \boldsymbol{S}\boldsymbol{x}(i-1) + \boldsymbol{w}(i) \tag{4.84}$$

with $\boldsymbol{S}$ the normalized adjacency matrix and $\boldsymbol{w}(i)$ a zero-mean i.i.d. Gaussian noise with covariance $\boldsymbol{I}_N$. It can be checked that $\boldsymbol{x}(i)$ is wide-sense stationary with $\mathbb{E}\{\boldsymbol{x}(i)\} = 0$,

and $\boldsymbol{R}_x(\tau) = \boldsymbol{S}^\tau \boldsymbol{R}_x(0)$ for all $\tau > 0$, where $\boldsymbol{R}_x(0)$ satisfies the Lyapunov equation (4.83). The graph filter order was set as $M = 3$. The step-sizes were set to $\mu = \{0.1, 0.038, 0.03\}$ for the LMS, LMSN and PLMS, respectively. The regularization parameter ϵ was set to $\epsilon = 0.1$. Fig. 4.4 depicts the simulated and theoretical MSD performance. We observe that, due to correlation over time, the diffusion LMSN method converged faster than the PLMS. However, the proposed PLMS algorithm still performed better than the diffusion LMS method.

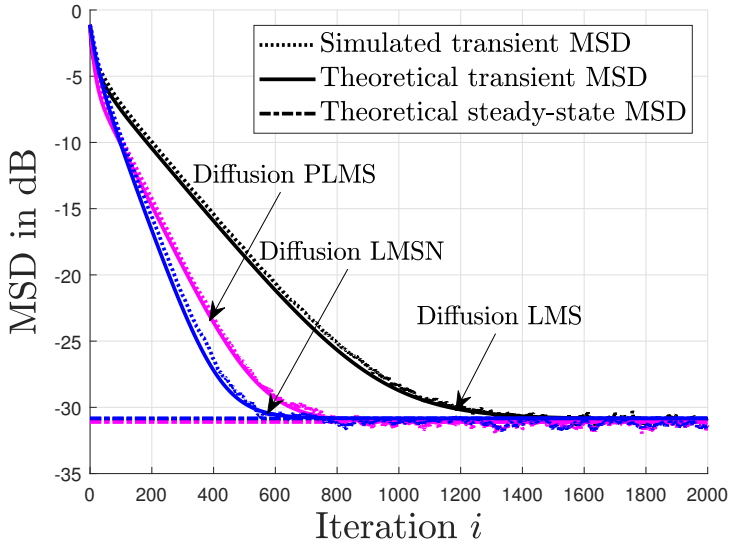


Figure 4.4: Network MSD performance with input graph signal correlated over both vertex and time domains.

4.6.3 Clustering method for node-varying graph filter

Finally, we considered a scenario where nodes do not share the same filter coefficients. We assumed the linear data model (4.10). The graph shift operator was defined by the normalized adjacency matrix, and the graph filter order was set as $M = 3$. The nodes were grouped into three clusters: $\mathcal{C}_1 = \{1, \ldots, 20\}$, $\mathcal{C}_2 = \{21, \ldots, 40\}$, and $\mathcal{C}_3 = \{41, \ldots, 60\}$. The optimal graph filter coefficients $\boldsymbol{h}_k^o$ were set according to $[0.5\ 0.4\ 0.9]^\top$ if $k \in \mathcal{C}_1$, $[0.3\ 0.1\ 0.4]^\top$ if $k \in \mathcal{C}_2$, and $[0.9\ 0.3\ 0.7]^\top$ if $k \in \mathcal{C}_3$. We considered for comparison purpose the PLMS algorithm with clustering mechanism (4.78)-(4.81) , with basic clustering mechanism (4.77) and $M_k = M$ for all k, the oracle PLMS algorithm where the clusters are assumed to be known a priori, the PLMS algorithm without clustering mechanism, and the non-cooperative algorithm where $a_{\ell k} = 1$ if $k = \ell$ and zero otherwise. All al-

gorithms used the adaptation step (4.32a) with the same step-size $\mu_k = 0.01$ for all k. Parameters $\{\tau, \beta, \theta, \nu\}$ were set to $\{0.9, 0.01, 0.5, 0.98\}$, respectively. As shown in Fig. 4.5, the non-cooperative method did not achieve acceptable MSD level. The main reason is that, with the normalized adjacency matrix as graph shift operator $\boldsymbol{S}$, the entries of $\boldsymbol{z}_k(i)$ in (4.8) corresponding to higher powers of $\boldsymbol{S}$ are significantly diminished, resulting in poor estimation performance of filter coefficients when nodes cannot cooperate. The PLMS algorithm without clustering mechanism did not achieve good performance too because it has been designed to converge toward a consensual solution $\boldsymbol{h}^o$, which does not make sense for this scenario. The PMLS with clustering mechanism (71) and $M_k = M$ achieved slightly improved performance because the estimation of higher-order coefficients in $\boldsymbol{h}_k$ was not reliable enough, leading to incorrect clustering. The proposed PLMS algorithm with clustering mechanism (4.78)-(4.81) performed as well as the oracle algorithm. Fig. 4.6 (a) shows the topology of the graph given by the adjacency matrix $\boldsymbol{A}$ (and the shift matrix $\boldsymbol{S}$). Fig. 4.6 (b) presents the clusters inferred by the proposed method. These clusters perfectly match the ground truth clusters $\mathcal{C}_1$ to $\mathcal{C}_3$.

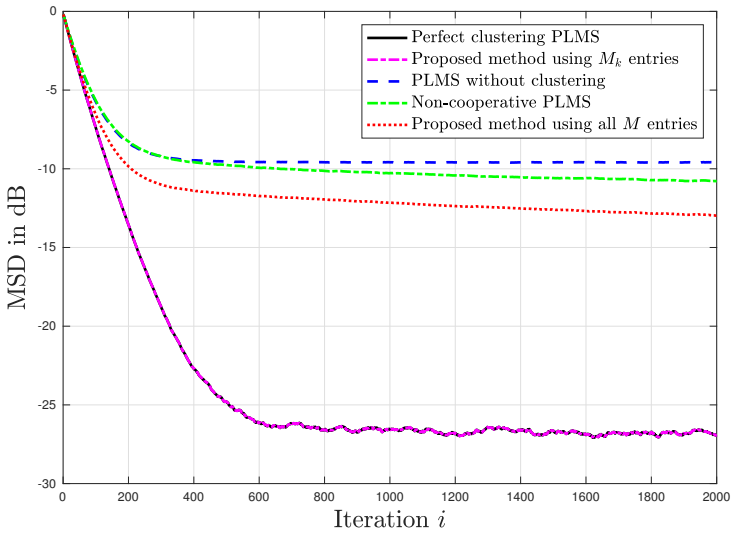
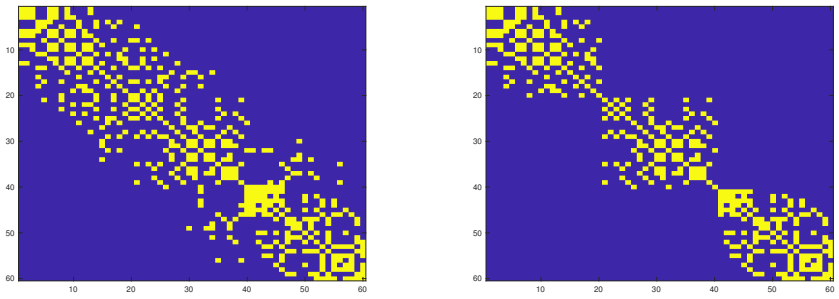


Figure 4.5: Network MSD performance for different clustering algorithms.

Next, we considered that optimal parameter vectors $\boldsymbol{h}_k^o$ change over time while clusters remain unchanged. Nodes were grouped into two clusters $\mathcal{C}_1 = \{1, \ldots, 30\}$ and $\mathcal{C}_2 = \{31, \ldots, 60\}$. The optimal parameter vectors changed for both clusters at time instant $i = 1000$. Simulation results in Fig. 4.7 show that the proposed clustering method was able to track well this change. Finally, we considered the scenario where clusters and models



(a) Adjacency Matrix (b) Inferred clusters

Figure 4.6: Graph topology and clusters.

change simultaneously. At Stage 1, the nodes were grouped into two clusters defined as $\mathcal{C}_1 = \{1, \ldots, 30\}$ and $\mathcal{C}_2 = \{31, \ldots, 60\}$. Stage 2 started at time instant $i = 1000$ with three clusters $\mathcal{C}_1 = \{1, \ldots, 20\}$, $\mathcal{C}_2 = \{21, \ldots, 40\}$, and $\mathcal{C}_3 = \{41, \ldots, 60\}$. Stage 3 started at time instant $i = 2000$ with two clusters $\mathcal{C}_1 = \{1, \ldots, 25\}$, $\mathcal{C}_2 = \{26, \ldots, 60\}$. At each stage, the optimal parameter vectors $\boldsymbol{h}_k^o$ changed accordingly. Ground truth clusters for the three stages are depicted in Fig. 4.9 (Top). Parameters $\{\mu, \tau, \beta, \theta, \nu\}$ were set to $\{0.01, 0.9, 0.01, 0.5, 0.4\}$, respectively. Fig. 4.8 shows the simulated transient MSD of the proposed PLMS algorithm with clustering mechanism. It is compared with the oracle PLMS algorithm where the clusters are assumed to be known a priori. Fig. 4.9 (Bottom) depicts the inferred clusters at $i = 1000, 2000, 3000$ during one Monte Carlo run. The proposed algorithm was able to track changes in both clusters and models.

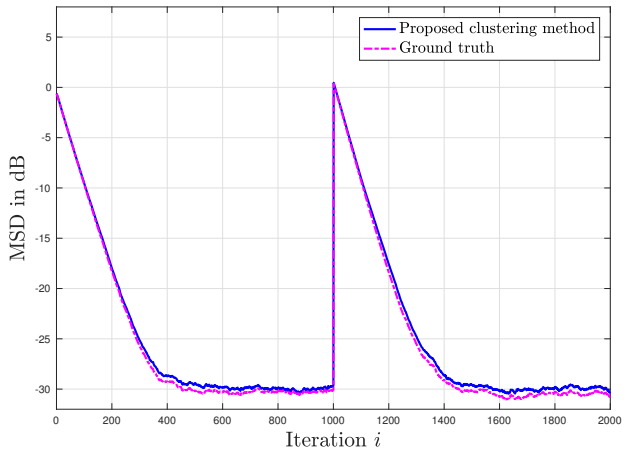


Figure 4.7: Network MSD performance with model change.

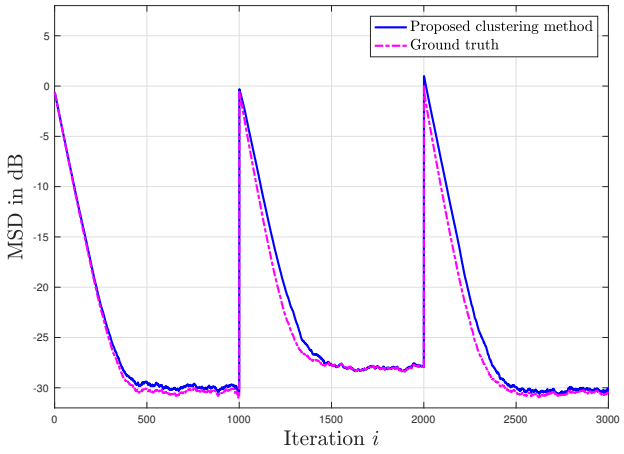


Figure 4.8: Network MSD performance with model and clusters change.

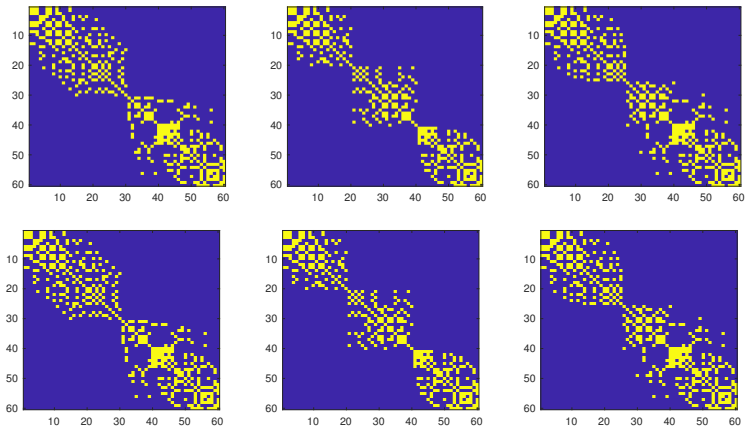
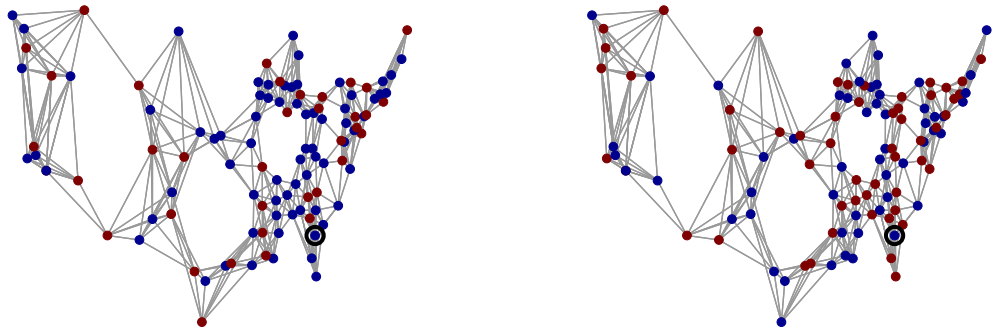


Figure 4.9: Ground truth cluster *(Top)*. Inferred clusters at steady-state of a single Monte Carlo run *(Bottom)*. From left to right: Stage 1, Stage 2, Stage 3.

4.6.4 Reconstruction on U.S. temperature dataset

We considered a dataset that collects hourly temperature measurements at $N = 109$ stations for $T = 8759$ hours across the United States in 2010 [noa]. An undirected graph, illustrated in Fig. 4.10, was constructed according to the nodes coordinates by using the k-NN approach ($k = 7$) of GSPBOX.

In the first experiment, the dataset was divided into a training set containing $T_{train} = 6570$ hours data (about 75% of total). The remaining data were assigned to the test set. The goal of this experiment was to learn a graph filter that minimizes the reconstruction



(a) (b)

Figure 4.10: Graph topology for the U.S. temperatures dataset. Temperatures were sampled at the red nodes in red. Data at the blue nodes were unobserved. (a) 37 sampled nodes. (b) 54 sampled nodes.

error over the training set, i.e.,

$$\min \sum_{i=1}^{T_{train}} \sum_{k=1}^{N} |y_k(i) - \sum_{m=1}^{M} h_{m,k}[\boldsymbol{S}^m \boldsymbol{x}(i-m+1)]_k|^2, \tag{4.85}$$

where $\boldsymbol{y}(i)$ is the ground truth temperature at time i, and $\boldsymbol{x}(i)$ is the partial observation given by $\boldsymbol{x}(i) = \text{diag}(\mathbf{1}_{\mathcal{S}})\boldsymbol{y}(i)$. Here $\mathbf{1}_{\mathcal{S}}$ denotes the set indicator vector, whose k-th entry is equal to one if node k is sampled, and zero otherwise. The sampling set, illustrated in Fig. 4.10 (a) was fixed over time in the first experiment. The normalized adjacency matrix was set as graph shift operator. Graph filter degree was set to $M = 4$. Note that if $h_{m,k} = h_m$ for all k, problem (4.85) refers to the single-task problem where all the nodes seek to find common graph filter coefficients; see model (4.1). Otherwise, problem (4.85) refers to the multitask problem; see model (4.9). We ran different models and algorithms on the training set to learn graph filter coefficients. In Fig. 4.11, we provide the true temperature and the reconstructed ones obtained by the different algorithms at an unobserved node, black circled in Fig. 4.10, over the last 120 hours samples of the test set. For comparison purposes, reconstruction results of the Kernel Kalman Filter (KKF) and the Kernel Ridge Regression (KRR) in [Romero 2017] are also reported in Fig. 4.11. We observe that the single-task diffusion LMSN and multitask diffusion LMS were not able to reconstruct the true temperature, whereas the multitask diffusion PLMS and the multitask diffusion LMSN showed a good reconstruction performance, and performed better than the KKF and KRR at the selected unobserved node. To evaluate the performance over all unobserved nodes

on the test set, we considered the normalized mean square error (NMSE) defined as:

$$\text{NMSE} = \frac{\sum_{i=T_{train}+1}^{T} \|\text{diag}(\mathbf{1}_{\bar{\mathcal{S}}}) \, (\boldsymbol{y}(i) - \hat{\boldsymbol{y}}(i)) \, \|^2}{\sum_{i=T_{train}+1}^{T} \|\text{diag}(\mathbf{1}_{\bar{\mathcal{S}}}) \boldsymbol{y}(i)\|^2} \tag{4.86}$$

where $\hat{\boldsymbol{y}}(i)$ denotes the reconstructed estimate at time i, $\mathbf{1}_{\bar{\mathcal{S}}}$ is the set indicator vector whose k-th entry is equal to one if node k has not been sampled, and zero otherwise. The results are reported in Table 4.1. We observe that the multitask diffusion PLMS performed as well as the LMSN at a lower computational cost, and both performed better than the KFF and KRR. Finally, Figure 4.12 reports the original topology and the clusters learned by the multitask diffusion PLMS.

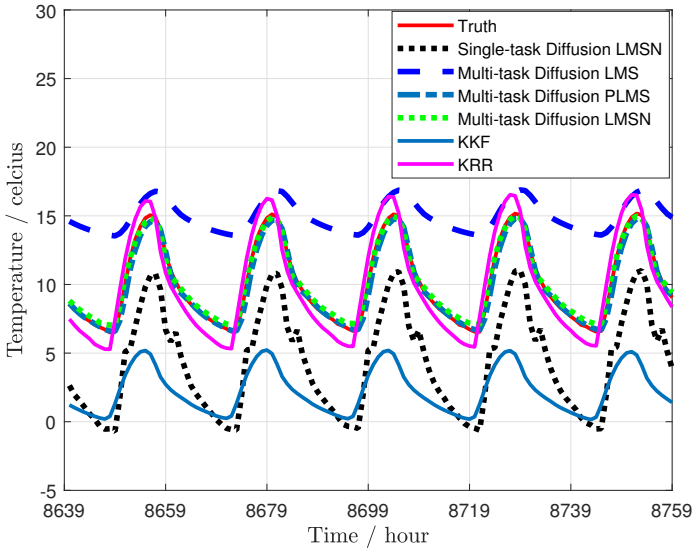
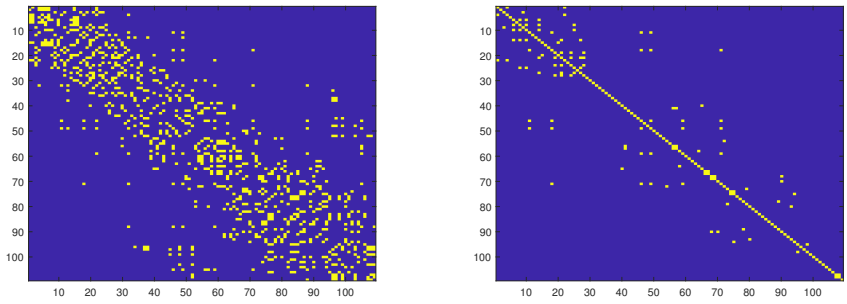


Figure 4.11: True temperatures and reconstructed ones at an unobserved node. $\mu_{\text{LMS}} = 10^{-5}$, $\mu_{\text{PLMS}} = \mu_{\text{LMSN}} = 10^{-4}$.

Table 4.1: NMSE of different algorithms.

Algorithm	NMSE
KKF	0.1093
KRR	0.0479
Multitask diffusion LMS	0.1152
Multitask diffusion PLMS	0.0090
Multitask diffusion LMSN	0.0031



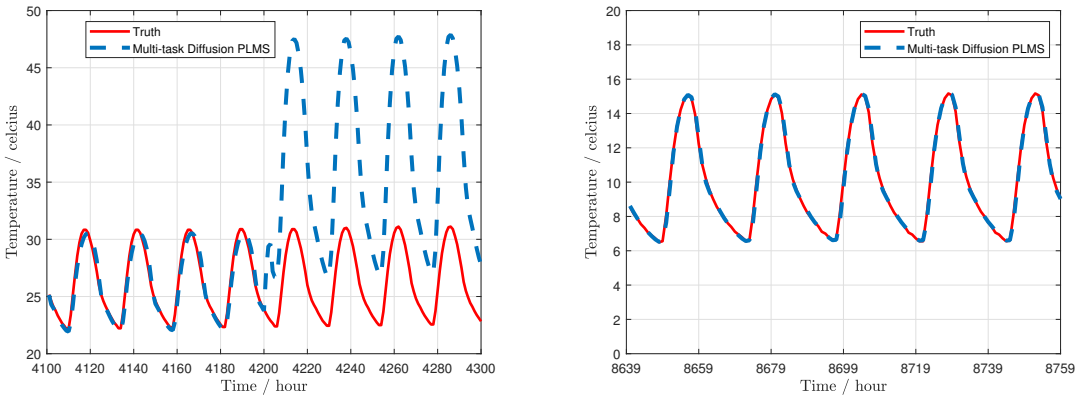
(a) Adjacency Matrix (b) Inferred clusters

Figure 4.12: U.S. temperature graph topology and learned clusters.

In the second experiment, we divided the dataset into two parts. The first part contained the first 4200 hours sampled at the nodes showed in Fig. 4.10 (a), and the second part contained the remaining hours sampled at the nodes showed in Fig. 4.10 (b). This means that the sampling set abruptly changed at time $t = 4201$ (the black circled node was unobserved in both case). We applied the multitask diffusion PLMS method over the entire dataset. In Fig. 4.13 (a), the reconstructed temperature at the unobserved black circled node is reported from time $t = 4100$ to $t = 4300$ by using the filter coefficients learned up to time $t = 4000$. As expected, we can notice that the reconstruction performance was successful from $t = 4100$ to $t = 4200$, and dramatically deteriorated after $t = 4201$. This is due to the fact that the sampling set changed, which led to a drift in the filter coefficients to estimate. Figure 4.13 (b) depicts the reconstruction behavior over last 120 hours using the filter coefficients learned up to time $t = 8600$. It can be observed that the proposed method was able to track the drift in the filter coefficients.

4.7 Conclusion

In this Chapter, diffusion LMS strategies were considered to estimate graph filter coefficients in an adaptive and distributed manner. A diffusion LMS with Newton-like descent procedure was first proposed to achieve improved convergence rate, since usual algorithms may suffer from ill conditioning effects due to the use of non-energy preserving graph shift operators. A preconditioned diffusion LMS strategy, which does not require computationally intensive matrix inversion and only uses local information, was then devised to reduce the computational burden. Its convergence behavior was analyzed in the mean and mean-square-error sense. Finally, for hybrid node-varying graph filters, a clustering mechanism to



(a) (b)

Figure 4.13: True temperatures and reconstructed ones at an unobserved node. For clarity purposes, focus on the intervals (a) $[4100, 4300]$ and (b) $[8640, 8759]$.

be used with the preconditioned diffusion LMS was proposed. Simulation results validated the theoretical models and showed the efficiency of the proposed algorithms.

4.A Block maximum norm

Let $\boldsymbol{x} = \mathrm{col}\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ denote an $N \times 1$ block column vector with each block $\boldsymbol{x}_k$ of size $M \times 1$. The block maximum norm of $\boldsymbol{x}$ is denoted by $\|\boldsymbol{x}\|_{b,\infty}$ and is defined as [Sayed 2014c, Appendix D]:

$$\|\boldsymbol{x}\|_{b,\infty} \triangleq \max_{1 \le k \le N} \|\boldsymbol{x}_k\| \tag{4.87}$$

where $\|\boldsymbol{x}_k\|$ is the Euclidean norm of $\boldsymbol{x}_k$. Let $\boldsymbol{\mathcal{A}}$ denote an $N \times N$ block matrix whose individual block entries are of size $M \times M$ each. The block maximum norm of $\boldsymbol{\mathcal{A}}$ can be induced from the block maximum norm of vector, and is defined as:

$$\|\boldsymbol{\mathcal{A}}\|_{b,\infty} \triangleq \max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{\mathcal{A}}\boldsymbol{x}\|_{b,\infty}}{\|\boldsymbol{x}\|_{b,\infty}}. \tag{4.88}$$

There are some useful properties of block maximum matrix norm. Let $\boldsymbol{A}$ be an $N \times N$ matrix with bounded entries and introduce the block matrix

$$\boldsymbol{\mathcal{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_M. \tag{4.89}$$

Then the block maximum matrix norm of $\boldsymbol{A}$ is equal to the maximum absolute row sum of the matrix $\boldsymbol{A}$ [Sayed 2014c], i.e.

$$\|\boldsymbol{\mathcal{A}}\|_{b,\infty} = \|\boldsymbol{A}\|_{\infty}. \tag{4.90}$$

Let $\boldsymbol{A}$ denote an $N \times N$ left stochastic matrix, i.e., its entries are nonnegative and it satisfies $\boldsymbol{A}^{\top}\mathbf{1} = \mathbf{1}$. Let $\boldsymbol{C}$ denote an $N \times N$ right stochastic matrix, i.e., its entries are nonnegative and it satisfies $\boldsymbol{C}\mathbf{1} = \mathbf{1}$. Let $\boldsymbol{\mathcal{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_M$ and $\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_M$, it holds that [Sayed 2014c]:

$$\|\boldsymbol{\mathcal{A}}^{\top}\|_{b,\infty} = 1, \tag{4.91}$$

$$\|\boldsymbol{\mathcal{C}}\|_{b,\infty} = 1. \tag{4.92}$$

Let $\boldsymbol{\mathcal{D}} = \text{bdiag}\{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_N\}$ denote an $N \times N$ block diagonal symmetric matrix whose each block $\boldsymbol{D}_k$ is an $M \times M$ symmetric matrix. Then, we have [Sayed 2014c]:

$$\|\boldsymbol{\mathcal{D}}\|_{b,\infty} = \rho(\boldsymbol{\mathcal{D}}) = \max_{1 \le k \le N} \rho(\boldsymbol{D}_k). \tag{4.93}$$

Chapter 5

Learning Combination of Graph Filters for Graph Signal Modeling

Contents

In Chapter 4, we considered the problem of learning multitask graph filter coefficients. The graph filter was based on the same graph shift operator, the individual node may apply different coefficients. In this Chapter, to enhance interpretability, we consider the graph filter with different graph shift operators, i.e., combination of graph filters. We study the problem of parametric modeling of network-structured signals with graph filters. To benefit from the properties of several graph shift operators simultaneously, we investigate combinations of parallel graph filters with different shift operators. Due to their extra degrees of freedom, these models might suffer from over-fitting. We address this problem through a weighted ℓ_2-norm regularization formulation to perform model selection by encouraging group sparsity. What makes this formulation interesting is that it is actually a smooth convex optimization problem. Experiments on real-world data structured by undirected and directed graphs show the effectiveness of this method. The material in this chapter is largely based on the work [Hua 2019].

5.1 Introduction

The field of Graph Signal Processing (GSP) has been introduced to exploit the complex data relations embedded in a graph, in the processing techniques. Similarly to filtering in the time domain, but built upon the Graph Fourier Transform (GFT), graph filters process graph signals by selectively amplifying and attenuating their graph Fourier coefficients. This makes them central in a number of GSP applications such as sampling [Anis 2016], modeling [Nassif 2018], reconstruction [Isufi 2018], denoising [Chen 2014d] and graph clustering [Tremblay 2016b]. Moreover, graph filters are the fundamental building block of graph wavelets and filter banks [Hammond 2011, Narang 2013, Tanaka 2014, Tremblay 2016a], and graph neural networks [Defferrard 2016, Gama 2018b]. Several architectures of graph filters have been proposed in the literature, including FIR filters [Shuman 2013, Sandryhaila 2013] and IIR filters [Shi 2015, Loukas 2015, Isufi 2017b]. Recently, several extensions have been introduced to fully exploit the structure in the graph data, such as node-variant graph filters [Segarra 2017] and edge-variant graph filters [Coutino 2019]. Driven by the need to deal with big data, all allow nodes to exchange only local information.

A key ingredient of GSP is the graph shift operator. It accounts for the topology of the graph in the vertex domain, and provides a basis for the GFT. It is also involved in designing, analyzing and implementing graph filters. The Laplacian matrix and the adjacency matrix are typical choices of graph shift operators for undirected graphs [Shuman 2013], and directed graphs [Sandryhaila 2013]. However, there is no consensus as to which operator should be considered for a given application. This has led several authors to introduce specific ones. For instance, in [Girault 2015b], the authors propose an isometric graph shift operator for stationary graph signals. In [Gavili 2017], the authors introduce a set of operators preserving the energy content of graph signals in the frequency domain. The authors in [Singh 2016] consider an extension for directed graphs of the symmetric Laplacian matrix.

In order to benefit from the properties of several graph shift operators simultaneously, and to enhance interpretability, some authors have experienced alternative strategies consisting of combining several graph shift operators. In [Coutino 2019], the authors combine edge-weighting matrices to obtain edge-variant graph filters. In [Anis 2016], the authors study the problem of selecting the best sampling set for band-limited reconstruction of signals on graphs. They define the Hub-authority operator as a convex combination of two operators. This model distinguishes between two types of nodes, hub nodes and authority

nodes, and allows to process specific directed graphs such as a hyperlinked environments. In [Sevi 2018b, Sevi 2018a], for modeling signals over directed graphs, the authors consider a convex combination of the random walk operator and its time reversed counterpart as a graph shift operator. All these works show that combinations of shift operators can perform better than single shift operators, but no work has so far addressed the problem of adjusting the balance between the shift operators in an optimal way.

In this chapter, we consider learning polynomial graph filters for graph signal modeling. We investigate combinations of two graph filters, i.e., parallel graph filters [Isufi 2017b], with different shift operators. Due to their extra degrees of freedom, these models might suffer from over-fitting unless appropriate regularization is performed. We address the problem of jointly estimating the coefficients of both filters and the combination coefficient through a weighted ℓ_2-norm regularization formulation to perform model selection by encouraging group sparsity. This helps to explain which shift operator contributes more to the combination. What makes this formulation interesting is that it is actually a smooth convex optimization problem, with connections with a mixed-norm regularization formulation.

5.2 Parametric modeling via graph filters

Let $\boldsymbol{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}\}$ denote a graph with $\mathcal{V}$ a set of N vertices and $\mathcal{E}$ a set of edges such that $\{k, \ell\} \in \mathcal{E}$ if there is an edge from vertex k to vertex ℓ. Graph $\boldsymbol{\mathcal{G}}$ can be represented by its N-by-N real-valued adjacency matrix $\boldsymbol{W}$ whose (k, ℓ)-th entry $w_{k\ell}$ assigns a weight to the relation between vertices k and ℓ such that:

$$w_{k\ell} > 0, \text{ if } \{k, \ell\} \in \mathcal{E}, \quad \text{and} \quad w_{k\ell} = 0, \text{ otherwise.} \tag{5.1}$$

Let $\boldsymbol{D} = \text{diag}\{d_1, \ldots, d_N\}$ denote the diagonal degree matrix whose k-th diagonal entry d_k is the sum of the k-th row entries of $\boldsymbol{W}$, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ the combinatorial Laplacian matrix. Matrix $\boldsymbol{L}$ is a symmetric positive semi-definite matrix for an undirected graph. The normalized Laplacian matrix is defined as follows: $\boldsymbol{L}_{\text{norm}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$. For a directed graph, the random walk operator is a probability transition matrix defined as $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}$. If the random walk is irreducible, it has a unique stationary distribution $\boldsymbol{\pi}$ that satisfies $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$. Its time reversed ergodic random walk is given by $\boldsymbol{P}^* = \boldsymbol{\Pi}^{-1}\boldsymbol{P}^\top\boldsymbol{\Pi}$, where $\boldsymbol{\Pi} = \text{diag}\{\pi_1, \ldots, \pi_N\}$.

A graph shift operator $\boldsymbol{S}$ is defined as an N-by-N matrix that captures the graph topology such that its entries satisfy:

$$s_{k\ell} \neq 0, \text{ if } \{k, \ell\} \in \mathcal{E} \text{ or } k = \ell, \text{ and } s_{k\ell} = 0, \text{ otherwise.} \tag{5.2}$$

In that sense, any matrix that satisfies condition (5.2) can be used as a graph shift operator. This property of locality allows a distributed implementation [Segarra 2017, Coutino 2019, Nassif 2018, Hua 2018b]. There are, however, other types of shift operators that do not satisfy (5.2); see for instance the isometric operator in [Girault 2015b]. Given a graph shift operator $\boldsymbol{S}$, we can define a polynomial shift-invariant graph filter as follows:

$$\boldsymbol{H} \triangleq \sum_{\ell=0}^{L-1} h_\ell \boldsymbol{S}^\ell \tag{5.3}$$

with $h_\ell \in \mathbb{R}$ the filter coefficients. Consider a graph signal defined as $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample at vertex k. The filtering process can be expressed as:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} = \sum_{\ell=0}^{L-1} h_\ell \boldsymbol{S}^\ell \boldsymbol{x} = \boldsymbol{M}\boldsymbol{h}, \tag{5.4}$$

with $\boldsymbol{y}$ the filter output vector, $\boldsymbol{M}$ the N-by-L matrix whose ℓ-th column is $[\boldsymbol{M}]_{\cdot,\ell} = \boldsymbol{S}^{\ell-1}\boldsymbol{x}$, and $\boldsymbol{h} = \text{col}\{h_\ell\}_{\ell=0}^{L-1}$ where operator $\text{col}\{\cdot\}$ stacks its scalar arguments on top of each other.

We consider a graph where each vertex k has access to a measurement y_k and a regression data x_k, assumed to be related by the linear model:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{M}\boldsymbol{h} + \boldsymbol{e} \tag{5.5}$$

for some unknown N-by-1 vector $\boldsymbol{h}$, where $\boldsymbol{e}$ denotes an i.i.d. zero-mean Gaussian noise. The model parameters $\boldsymbol{h}$ can be learned by minimizing the following cost function:

$$J(\boldsymbol{h}) = \|\boldsymbol{y} - \boldsymbol{M}\boldsymbol{h}\|^2 = \sum_{k=1}^{N} (y_k - \boldsymbol{h}^\top \boldsymbol{m}_k)^2 \tag{5.6}$$

with $\boldsymbol{m}_k^\top$ the k-th row vector of $\boldsymbol{M}$. As there is no rule on how to choose $\boldsymbol{S}$ given $\boldsymbol{x}$ and $\boldsymbol{y}$, some recent studies have shown that using a convex combination of shift operators, namely,

$$\boldsymbol{S} = \alpha \boldsymbol{S}_1 + (1-\alpha)\boldsymbol{S}_2, \quad \alpha \in [0,1] \tag{5.7}$$

instead of a single one can improve the performance and the interpretability of the regression function [Anis 2016, Sevi 2018b, Sevi 2018a]. Possible choices for $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are, e.g., the adjacency $\boldsymbol{W}$ and the Laplacian $\boldsymbol{L}$ matrices for undirected graphs, and the random walk matrix $\boldsymbol{P}$ and its time reversed counterpart $\boldsymbol{P}^*$ for directed graphs. In the same spirit as multiple kernel learning [Rakotomamonjy 2008, Chen 2013b, Chen 2014a] in Machine Learning, one might want to simultaneously learn a valid graph shift operator of the form (5.7) and the filter coefficients $\boldsymbol{h}$, in a supervised learning setting. Nevertheless,

considering the convex combination (5.7), and minimizing $J(\cdot)$ in (5.6) with respect to variables $\boldsymbol{h}$ and α, gives rise to a non-convex optimization problem. For this reason, we shall not pursue this idea further.

As an alternative to (5.7), this letter focuses on combinations of two graph filters, i.e., parallel graph filters [Isufi 2017b], which can also mix different graph shift operators. The problem of jointly estimating the coefficients of both filters and the combination coefficient is addressed in the next section, through an ℓ_2-norm regularization formulation that allows us to perform model selection by encouraging group sparsity.

5.3 Jointly estimating graph filters coefficients and the combination coefficient

Before proceeding with parallel graph filters, for clarity, we begin with (5.6) where we introduce an ℓ_2-regularization term:

$$J_{\text{reg}}(\boldsymbol{h}) = \frac{1}{2}\|\boldsymbol{h}\|^2 + \frac{1}{2\mu}\sum_{k=1}^{N}(y_k - \boldsymbol{h}^\top \boldsymbol{m}_k)^2 \tag{5.8}$$

with μ a small positive value to control the trade-off between the regularization and the fitting terms. Indeed, as studied in [Hua 2018b], the problem of estimating graph filter coefficients $\boldsymbol{h}$ in the mean-square error sense is inherently ill-conditioned. The reason is that the shift operator $\boldsymbol{S}$ is not energy preserving in general [Gavili 2017], because the magnitudes of its eigenvalues are not uniformly equal to 1; the energy of the shifted signal $\boldsymbol{S}^\ell \boldsymbol{x}$ then changes exponentially with ℓ. Thus the eigenvalue spread of matrix $\boldsymbol{M}^\top \boldsymbol{M}$ is usually large, which results in numerical instability and large sensitivity to noisy measurements when estimating $\boldsymbol{h}$ from (5.6). Tikhonov regularizer in (5.8), which imposes penalty on $\|\boldsymbol{h}\|^2$, allows to address this issue. The closed-form solution of (5.8) is given by

$$\boldsymbol{h} = (\mu \boldsymbol{I} + \boldsymbol{M}^\top \boldsymbol{M})^{-1} \boldsymbol{M}^\top \boldsymbol{y}. \tag{5.9}$$

While we can also derive the optimum $\boldsymbol{h}^*$ alternatively in the dual domain. It can be obtained by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{h}^* = &\arg\min_{\boldsymbol{h}\in\mathbb{R}^L} \frac{1}{2}\|\boldsymbol{h}\|^2 + \frac{1}{2\mu}\sum_{k=1}^{N} e_k^2 \\ &\text{s.t.} \quad e_k = y_k - \boldsymbol{h}^\top \boldsymbol{m}_k, \quad k \in \{1, \dots, N\}. \end{aligned} \tag{5.10}$$

The dual problem can be derived by introducing Lagrange multipliers λ_k as follows:

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{h}\|^2 + \frac{1}{2\mu}\sum_{k=1}^{N} e_k^2 - \sum_{k=1}^{N} \lambda_k (e_k - y_k + \boldsymbol{h}^\top \boldsymbol{m}_k). \tag{5.11}$$

The optimality conditions w.r.t. the primal variables lead to:

$$\boldsymbol{h}^* = \sum_{k=1}^{N} \lambda_k^* \boldsymbol{m}_k, \quad e_k^* = \mu \lambda_k^* \tag{5.12}$$

where the dual variables are estimated by solving:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\arg\max} -\frac{1}{2}\boldsymbol{\lambda}^\top (\boldsymbol{M}\boldsymbol{M}^\top + \mu \boldsymbol{I})\boldsymbol{\lambda} + \boldsymbol{\lambda}^\top \boldsymbol{y} \tag{5.13}$$

which is a quadratic programming (QP) problem. Consider now two parallel graph filters $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ defined w.r.t. shift operators $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. Model (5.5) becomes:

$$\boldsymbol{y} = \boldsymbol{M}_1 \boldsymbol{h}_1 + \boldsymbol{M}_2 \boldsymbol{h}_2 + \boldsymbol{e} \tag{5.14}$$

with $[\boldsymbol{M}_p]_{\cdot,\ell} = \boldsymbol{S}_p^{\ell-1}\boldsymbol{x}$ and $\boldsymbol{h}_p \triangleq \text{col}\{h_{p,\ell}\}_{\ell=0}^{L-1}$ the coefficients of filter $\boldsymbol{H}_p$. We shall use $\boldsymbol{m}_{p,k}^\top$ to denote the k-th row vector of $\boldsymbol{M}_p$.

Compared to (5.5), model (5.14) might suffer from overfitting due to extra degrees of freedom, unless regularization is used. Jointly estimating the coefficients of both filters and balancing their contributions through the weighted ℓ_2-norm regularization formulation (5.15) shall allow us to perform model selection by encouraging group sparsity. More precisely, let us consider now the following problem derived from the multiple kernel learning literature [Rakotomamonjy 2008, Chen 2013b, Chen 2014a]:

$$\begin{aligned} \underset{\boldsymbol{h}_1, \boldsymbol{h}_2, \alpha}{\arg\min} \quad & \frac{1}{2}\left(\frac{\|\boldsymbol{h}_1\|^2}{\alpha} + \frac{\|\boldsymbol{h}_2\|^2}{1-\alpha}\right) + \frac{1}{2\mu}\sum_{k=1}^{N} e_k^2 \\ \text{s.t. :} \quad & \\ & e_k = y_k - \boldsymbol{h}_1^\top \boldsymbol{m}_{1,k} - \boldsymbol{h}_2^\top \boldsymbol{m}_{2,k}, \quad k \in \{1, \ldots, N\} \\ & 0 < \alpha < 1, \end{aligned} \tag{5.15}$$

where α allows to adjust the balance between $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ via their norms. We show in Subsection 5.3.3 that this formulation is equivalent to a mixed-norm penalization which promotes sparsity at the group level. Indeed, the solution of problem (5.15) tends to that of problem (5.8) with $\boldsymbol{h}_2$ (resp., $\boldsymbol{h}_1$) as α tends to 0 (resp., 1). Note that function $\|\boldsymbol{h}\|^2/\alpha$, called the perspective function, is jointly convex w.r.t. $\boldsymbol{h}$ and α [Boyd 2004]. It follows that problem (5.15) is jointly convex w.r.t. $\boldsymbol{h}_1$, $\boldsymbol{h}_2$ and α.

In order to solve problem (5.15), we consider the following constrained optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & J(\alpha) \\ \text{s.t.} \quad & 0 < \alpha < 1 \end{aligned} \tag{5.16}$$

where $J(\alpha)$ is given by

$$J(\alpha) = \begin{cases} \min\limits_{\boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{R}^L} F(\alpha, \boldsymbol{h}_1, \boldsymbol{h}_2) = \frac{1}{2}\left(\frac{1}{\alpha}\|\boldsymbol{h}_1\|^2 + \frac{1}{1-\alpha}\|\boldsymbol{h}_2\|^2\right) + \frac{1}{2\mu}\sum_{k=1}^N e_k^2 \\ \text{s. t.} \quad e_k = y_k - \boldsymbol{h}_1^\top \boldsymbol{m}_{1,k} - \boldsymbol{h}_2^\top \boldsymbol{m}_{2,k}, \quad k \in \{1, \dots, N\} \end{cases}. \tag{5.17}$$

Problem (5.16) is an optimization problem that is jointly convex w.r.t. $\alpha, \boldsymbol{h}_1, \boldsymbol{h}_2$. It can be solved with a two-step procedure w.r.t $\boldsymbol{h}_1, \boldsymbol{h}_2$ and α successively.

5.3.1 Solving w.r.t. $\boldsymbol{h}_1, \boldsymbol{h}_2$

The Lagrangian of problem (5.17) can be written as:

$$\mathcal{L}' = \frac{1}{2}\left(\frac{1}{\alpha}\|\boldsymbol{h}_1\|^2 + \frac{1}{1-\alpha}\|\boldsymbol{h}_2\|^2\right) + \frac{1}{2\mu}\sum_{k=1}^N e_k^2 - \sum_{k=1}^N \lambda_k \left(e_k - y_k + \boldsymbol{h}_1^\top \boldsymbol{m}_{1,k} + \boldsymbol{h}_2^\top \boldsymbol{m}_{2,k}\right). \tag{5.18}$$

The optimality conditions for $\mathcal{L}'$ w.r.t. the primal variables are:

$$\begin{cases} \boldsymbol{h}_1^* &= \alpha \sum_{k=1}^N \lambda_k^* \boldsymbol{m}_{1,k} \\ \boldsymbol{h}_2^* &= (1-\alpha) \sum_{k=1}^N \lambda_k^* \boldsymbol{m}_{2,k} \\ e_k^* &= \mu \lambda_k^* \end{cases}. \tag{5.19}$$

Note that coefficients $\boldsymbol{h}_1^*$ and $\boldsymbol{h}_2^*$ are coupled through α in the dual domain. Substituting (5.19) into (5.18) yields:

$$\begin{aligned} \boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda} \in \mathbb{R}^N} -\frac{1}{2}\boldsymbol{\lambda}^\top (\boldsymbol{R}_\alpha + \mu \boldsymbol{I})\boldsymbol{\lambda} + \boldsymbol{\lambda}^\top \boldsymbol{y} \\ \text{with } \boldsymbol{R}_\alpha = \alpha \boldsymbol{M}_1 \boldsymbol{M}_1^\top + (1-\alpha)\boldsymbol{M}_2 \boldsymbol{M}_2^\top. \end{aligned} \tag{5.20}$$

Problem (5.20) is a QP problem which can be efficiently solved. Given $\boldsymbol{\lambda}^*$, coefficients $\boldsymbol{h}_1^*$ and $\boldsymbol{h}_2^*$ can be computed with (5.19).

5.3.2 Solving w.r.t. α

First, note that function:

$$f_{p,q}(\alpha) = \frac{p}{\alpha} + \frac{q}{1-\alpha} \quad \text{with} \quad p, q \geq 0 \tag{5.21}$$

is convex over $0 < \alpha < 1$. Its optimum is given by:

$$\alpha^* = (1 + \sqrt{q/p})^{-1}. \tag{5.22}$$

Then, considering (5.17), and substituting $\boldsymbol{h}_1^*$, $\boldsymbol{h}_2^*$ from (5.19) in (5.21), the optimum value α_i^* at iteration i is provided by:

$$\alpha_i^* = \left(1 + \frac{1-\alpha_{i-1}^*}{\alpha_{i-1}^*}\sqrt{\frac{\boldsymbol{\lambda}^{*\top}\boldsymbol{M}_2\boldsymbol{M}_2^\top\boldsymbol{\lambda}^*}{\boldsymbol{\lambda}^{*\top}\boldsymbol{M}_1\boldsymbol{M}_1^\top\boldsymbol{\lambda}^*}}\right)^{-1} \tag{5.23}$$

where α_{i-1}^* is the value obtained from the previous iteration. The algorithm can be stopped based on Karush-Kuhn-Tucker conditions, or the duality gap, up to an error tolerance provided by the user. The procedure is summarized in Algorithm 1.

Algorithm 1

Input: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S}_1, \boldsymbol{S}_2, L$.

Initialize: randomly choose $0 < \alpha_{-1}^* < 1$, compute $\boldsymbol{M}_1, \boldsymbol{M}_2$.

Repeat:

1: solve (5.20) with a generic QP solver to get $\boldsymbol{\lambda}^*$
compute $\boldsymbol{h}_1^*, \boldsymbol{h}_2^*$ from (5.19),

2: update α_i^* by using (5.23).

Until: stopping condition is satisfied.

Output: $\boldsymbol{h}_1^*, \boldsymbol{h}_2^*, \alpha^*$.

5.3.3 Mixed-norm formulation

Let us show now that (5.15) admits a mixed-norm equivalent form that is known to perform model selection by encouraging group sparsity. By Cauchy-Schwartz inequality, we have:

$$\big(\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|\big)^2 \leq \frac{\|\boldsymbol{h}_1\|^2}{\alpha} + \frac{\|\boldsymbol{h}_2\|^2}{1-\alpha} \tag{5.24}$$

because

$$\begin{aligned}\big(\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|\big)^2 &= \left(\frac{\|\boldsymbol{h}_1\|}{\sqrt{\alpha}}\sqrt{\alpha} + \frac{\|\boldsymbol{h}_2\|}{\sqrt{1-\alpha}}\sqrt{1-\alpha}\right)^2 \\ &\leq \left[\left(\frac{\|\boldsymbol{h}_1\|}{\sqrt{\alpha}}\right)^2 + \left(\frac{\|\boldsymbol{h}_2\|}{\sqrt{1-\alpha}}\right)^2\right](\alpha + 1 - \alpha).\end{aligned} \tag{5.25}$$

Equality is achieved when vectors $\big[\|\boldsymbol{h}_1\|/\sqrt{\alpha}\,;\, \|\boldsymbol{h}_2\|/\sqrt{1-\alpha}\big]$ and $\big[\sqrt{\alpha}\,;\,\sqrt{1-\alpha}\big]$ are collinear, i.e., $\alpha = \|\boldsymbol{h}_1\|/\big[\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|\big]$, which implies that:

$$\min_{\alpha\in(0,1)} \quad \left(\frac{\|\boldsymbol{h}_1\|^2}{\alpha} + \frac{\|\boldsymbol{h}_2\|^2}{1-\alpha}\right) = \big(\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|\big)^2 \tag{5.26}$$

Problem (5.15) is then equivalent to:

$$\begin{aligned} \min_{\boldsymbol{h}_1,\boldsymbol{h}_2} \quad & \frac{1}{2}\big(\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|\big)^2 + \frac{1}{2\mu}\sum_{k=1}^{N} e_k^2 \\ \text{s. t.} \quad & e_k = y_k - \boldsymbol{h}_1^\top \boldsymbol{m}_{1,k} - \boldsymbol{h}_2^\top \boldsymbol{m}_{2,k}, \quad k \in \{1,\ldots,N\} \end{aligned} \tag{5.27}$$

Regularizer $\big(\|\boldsymbol{h}_1\|+\|\boldsymbol{h}_2\|\big)^2$ is a mixed-norm penalization term that promotes sparsity at the group level [Simon 2013]. That is, problem (5.27) performs model selection by encouraging sparsity at the level of the two entire groups of variables defined by the entries of $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. Unlike (5.15), the objective function in (5.27) is not smooth since $\|\boldsymbol{h}\|$ is not differentiable.

5.4 Numerical results

We considered the Molène temperature data set of hourly weather observations collected during January 2014 in Brittany, France, [Girault 2015a] for undirected graphs, and the data set of the political blogs of the 2004 US presidential election [Adamic 2005] for directed graphs. We compared the performance of model (5.5), model (5.5) with mixed operator (5.7), and model (5.14). On the one hand, problem (5.8) was considered to estimate models (5.5), and (5.5) with operator (5.7). For the later, to determine the best α, we followed the strategy in [Sevi 2018a], i.e., α was sampled over $(0, 1)$ and problem (5.8) was solved for each candidate value. On the other hand, problem (5.15) was solved to estimate model (5.14). Operators $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ used for both experiments are provided in Table 5.1. Note that $\boldsymbol{W}_{\text{norm}}$ denotes the normalized adjacency matrix $\boldsymbol{W}_{\text{norm}} = \boldsymbol{W}/|\lambda_{\max}\{\boldsymbol{W}\}|$, where $\lambda_{\max}\{\cdot\}$ denotes the maximum eigenvalue of its matrix argument. The graph filters order was arbitrarily set to $L = 10$ after we observed that the performance of the compared methods varied little for orders larger than 7. Parameter μ was arbitrarily set to $5 \cdot 10^{-4}$.

Table 5.1: Shift operators used in the experiments.

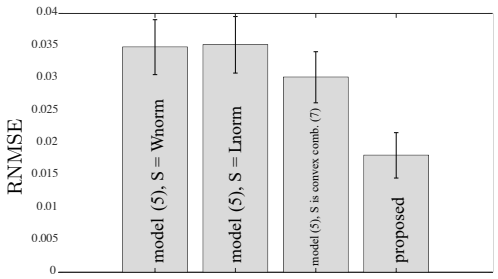
Graph type	$\boldsymbol{S}_1$	$\boldsymbol{S}_2$
Molène data (undirected)	$\boldsymbol{W}_{\text{norm}}$	$\boldsymbol{L}_{\text{norm}}$
Political blog (directed)	$\boldsymbol{P}$	$\boldsymbol{P}^*$

Undirected graph: The Molène data set consists of 32 vertices, with 744 observations each. The undirected graph was generated by using GSPBOX [Perraudin 2014]. Each vertex was connected to its 6 nearest neighbors. We considered the following denoising problem. Let $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{e}$ be a noisy graph signal measurement, where $\boldsymbol{x}$ is the original

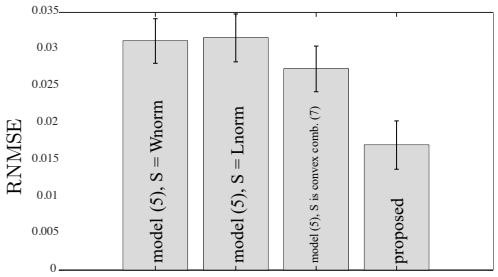
signal and $\boldsymbol{e}$ an additive noise. The goal was to learn a graph filter $\hat{\boldsymbol{H}}$ that minimizes the reconstruction error: $\|\boldsymbol{x} - \boldsymbol{H}\tilde{\boldsymbol{x}}\|^2$. For this purpose, we randomly selected one realization in the Molène data set, and we corrupted it with an i.i.d. additive noise. Two noises were experimented: Gaussian $\mathcal{N}(0, 10^2)$, and uniform $\mathcal{U}([-15, 15])$. Then we learned the graph filter $\hat{\boldsymbol{H}}$; the three models outlined above were considered. The other 743 observations were also corrupted with the additive noise, and denoised with $\hat{\boldsymbol{H}}$. Figure 5.1 depicts the reconstruction root normalized mean square error defined as:

$$\mathsf{RNMSE} = \sqrt{\|\hat{\boldsymbol{H}}\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 / \|\boldsymbol{x}\|^2}. \tag{5.28}$$

averaged over the 743 independent test observations. It can be observed that proposed method provided the best performance.



(a) Gaussian noise $\mathcal{N}(0, 10^2)$



(b) Uniform noise $\mathcal{U}(-15, 15)$

Figure 5.1: Denoising performance over Molène data set.

Directed graph: The political blogs data set consists of 1224 blogs where each blog i is either conservative and labeled as $y_i = +1$, or liberal and labeled as $y_i = -1$. This data set can be represented by a directed graph where vertices represent blogs, and a directed edge is considered to be present from vertex i to vertex j if there is a hyperlink from blog i to j. We considered a strongly connected part of this graph composed of 793 blogs, where 351 are liberal and the others conservative. The goal was to learn a graph filter $\hat{\boldsymbol{H}}$ that minimizes the reconstruction error $\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2$, where $\boldsymbol{x}$ is an input vector obtained

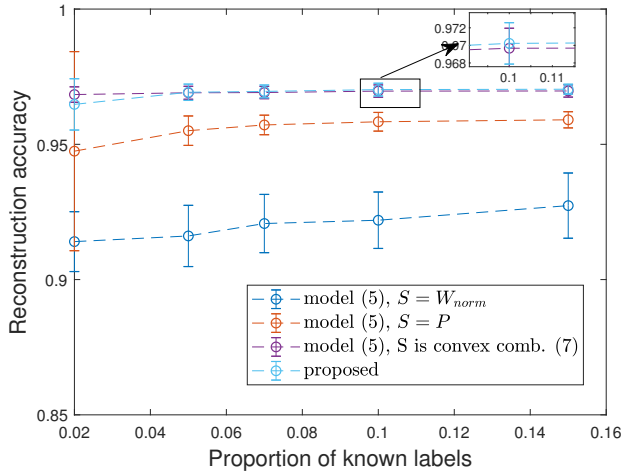


Figure 5.2: Reconstruction accuracy for different proportions of known labels of the political blogs data.

by randomly setting part of the entries of $\boldsymbol{y}$ to zero. The reconstruction accuracy was obtained by comparing $\boldsymbol{y}$ with reconstructed labels $\hat{\boldsymbol{y}} = \text{sgn}(\hat{\boldsymbol{H}}\boldsymbol{x})$. Figure 5.2 reports the reconstruction accuracy for different proportions of known labels. The results are based on 100 realizations of $\boldsymbol{x}$ for each proportion. Observe that the combination models performed better than the filters based on $\boldsymbol{W}_{\text{norm}}$ or $\boldsymbol{P}$. The proposed model (5.14) also performed slightly better than model (5.5) when the latter used the shift operator $\boldsymbol{S}$ defined as in (5.7).

5.5 Conclusion

In this Chapter, we considered the problem of learning parallel combinations of polynomial graph filters for graph signal modeling, in order to benefit from the properties of several graph shift operators simultaneously. The problem of jointly estimating the coefficients of both filters and the combination coefficient was addressed through an ℓ_2-norm regularization formulation that allows to perform model selection by encouraging group sparsity. Numerical results on real-world data, for undirected and directed graphs, demonstrated the efficiency and robustness of the proposed method.

Chapter 6

Conclusion and future works

We are living in a highly interconnected world. There are prevalent networked systems collecting streaming data in a distributed fashion. This thesis focused on adaptation and learning over networks and graphs which are able to deal with streaming data and time-varying dynamics. In this Chapter, we briefly summarize the main results proposed in each chapter and discuss some possible future directions.

6.1 Summary

We proposed a distributed adaptive estimation algorithm based on penalty method for solving constrained multitask problem in Chapter 2. We studied the performance of proposed algorithm in the mean and mean-square sense. It was shown that the algorithm is able to converge to the optimum with a bias, while the bias can be arbitrary small by choosing small step-size (Theorem 2.1). The mean-square stability can also be ensured from sufficiently small-step size (Theorem 2.2). The expressions of transient performance (Theorem 2.3) and steady-state performance (Theorem 2.4) are provided as well.

In Chapter 3, in order to solve the multitask problem with non-local constraints in a fully distributed manner, we proposed an adaptive algorithm based on penalty method and multi-hop relay protocols. We derived a detailed performance analysis. We showed that the delays indeed have impacts on stability condition, however, the algorithm can continue to converge if a sufficient stability condition is satisfied (Theorem 3.1, Theorem 3.2). We also provided closed form expressions to predict the learning behavior which can be used to tune parameters.

In Chapter 4, we proposed a framework for adaptation of time-varying graph signals. Diffusion LMS strategies were considered to estimate graph filter coefficients in an adaptive and distributed manner. A diffusion LMS with Newton-like descent procedure was first proposed to achieve improved convergence rate. A preconditioned diffusion LMS strategy, which does not require computationally intensive matrix inversion and only uses local information, was then devised to reduce the computational burden. Its convergence behavior

was analyzed in the mean and mean-square-error sense (Theorem 4.1, Theorem 4.2). Finally, for hybrid node-varying graph filters, a clustering mechanism to be used with the preconditioned diffusion LMS was proposed. Simulation results validated the theoretical models and showed the effectiveness of the proposed algorithms.

In Chapter 5, we proposed to use parallel combinations of polynomial graph filters for graph signal modeling in order to benefit from the properties of several graph shift operators simultaneously. To avoid over-fitting and ill-condition, we proposed to use an ℓ_2-norm regularization formulation. We showed that weighted ℓ_2-norm regularization allows to perform model selection by encouraging group sparsity. Numerical results on real-world data, for undirected and directed graphs, demonstrated the efficiency and robustness of the proposed method.

6.2 Future works

Throughout the thesis, we restricted ourself to linear models where each agent collects data through a linear regression model. Learning nonlinear input-output relations via kernel-based models or non-parametric methods [Richard 2009, Gao 2014, Koppel 2018, Pradhan 2018] is a possible direction to be considered.

In Chapter 2 and Chapter 3, we considered the multitask problem where the parameters are coupled through linear equality constraints. While, the global cost function is considered to be sum aggregation of local cost functions which means they are separable. There are applications where the global cost function is coupled as well, such as smart grid and distributed sparse regression. Developing distributed algorithm to such non-separable costs problem would be challenging but very useful in some applications.

One the other hand, the cost function considered in this thesis is convex. Nevertheless, many optimization problems in signal processing and machine learning are non-convex [Vlaski 2019, Chang 2020]. Understanding the behavior of distributed multitask algorithms in non-convex environments could be considered.

In this thesis, we assumed that the graph shift operator is known and time invariant. In future, we could consider ways to estimate the graph shift operator and the filter coefficients simultaneously. Moreover, taking into account the nonlinearity, kernel or non-parametric methods could also be considered to be incorporated into GSP framework.

Bibliography

[Abdolee 2016] Reza Abdolee, Benoit Champagne and Ali H Sayed. *Diffusion adaptation over multi-agent networks with wireless link impairments.* IEEE Transactions on Mobile Computing, vol. 15, no. 6, pages 1362–1376, 2016. (Cited on page 5.)

[Adamic 2005] Lada A Adamic and Natalie Glance. *The political blogosphere and the 2004 US election: divided they blog.* In International workshop on Link discovery, pages 36–43, New York, USA, 2005. ACM. (Cited on page 101.)

[Aldous 1995] David Aldous and James Fill. Reversible markov chains and random walks on graphs. Berkeley, Berkeley, USA, 1995. (Cited on page 9.)

[Alghunaim 2020] Sulaiman A Alghunaim, Kun Yuan and Ali H Sayed. *A proximal diffusion strategy for multi-agent optimization with sparse affine constraints.* to apper in IEEE Transactions on Automatic Control, 2020. (Cited on page 7.)

[Anis 2016] Aamir Anis, Akshay Gadde and Antonio Ortega. *Efficient sampling set selection for bandlimited graph signals using graph spectral proxies.* IEEE Transactions on Signal Processing, vol. 64, no. 14, pages 3775–3789, Jul. 2016. (Cited on pages 8, 10, 60, 94 and 96.)

[Anis 2019] A. Anis, A. El Gamal, A. S. Avestimehr and A. Ortega. *A Sampling Theory Perspective of Graph-Based Semi-Supervised Learning.* IEEE Transactions on Information Theory, vol. 65, no. 4, pages 2322–2342, Apr. 2019. (Cited on page 60.)

[Arablouei 2014] Reza Arablouei, Stefan Werner, Yih-Fang Huang and Kutluyıl Doğançay. *Distributed least mean-square estimation with partial diffusion.* IEEE Transactions on Signal Processing, vol. 62, no. 2, pages 472–484, 2014. (Cited on page 5.)

[Barbarossa 2013] Sergio Barbarossa, Stefania Sardellitti and Paolo Di Lorenzo. *Distributed Detection and Estimation in Wireless Sensor Networks.* In Sergios Theodoridis and Rama Chellappa, Editors, Academic Press Library in Signal Processing, volume 2, pages 329–408. Academic Press, Elsevier, Cambridge, MA, USA, 2013. (Cited on pages 4 and 42.)

[Bazaraa 2013] Mokhtar S Bazaraa, Hanif D Sherali and Chitharanjan M Shetty. Nonlinear programming: theory and algorithms. John Wiley & Sons, Hoboken, NJ, USA, 2013. (Cited on pages 20, 21 and 44.)

[Bertrand 2010] Alexander Bertrand and Marc Moonen. *Distributed adaptive node-specific signal estimation in fully connected sensor networks–Part I: Sequential node updating.* IEEE Transactions on Signal Processing, vol. 58, no. 10, pages 5277–5291, 2010. (Cited on page 5.)

[Bertrand 2011] Alexander Bertrand and Marc Moonen. *Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology.* IEEE Transactions on Signal Processing, vol. 59, no. 5, pages 2196–2210, 2011. (Cited on page 5.)

[Bertsekas 1997] Dimitri P Bertsekas. *A new class of incremental gradient methods for least squares problems.* SIAM Journal on Optimization, vol. 7, no. 4, pages 913–926, 1997. (Cited on pages 3, 42 and 67.)

[Bertsekas 1998] Dimitri P Bertsekas. Network optimization: continuous and discrete models. Athena Scientific, Belmont, MA, USA, 1998. (Cited on pages 7, 16 and 42.)

[Bertsekas 1999] Dimitri P Bertsekas. Nonlinear programming. Athena Scientific, Belmont, MA, USA, 1999. (Cited on page 19.)

[Blatt 2007] Doron Blatt, Alfred O Hero and Hillel Gauchman. *A convergent incremental gradient method with a constant step size.* SIAM Journal on Optimization, vol. 18, no. 1, pages 29–51, 2007. (Cited on page 3.)

[Bogdanovic 2014] Nikola Bogdanovic, Jorge Plata-Chaves and Kostas Berberidis. *Distributed incremental-based LMS for node-specific adaptive parameter estimation.* IEEE Transactions on Signal Processing, vol. 62, no. 20, pages 5382–5397, Oct. 2014. (Cited on pages 5 and 16.)

[Boyd 2004] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge University Press, Cambridge, U.K., 2004. (Cited on page 98.)

[Boyd 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato and Jonathan Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Foundations and Trends® in Machine learning, vol. 3, no. 1, pages 1–122, 2011. (Cited on page 4.)

[Braca 2008] Paolo Braca, Stefano Marano and Vincenzo Matta. *Enforcing consensus while monitoring the environment in wireless sensor networks.* IEEE Transactions on Signal Processing, vol. 56, no. 7, pages 3375–3380, Jun. 2008. (Cited on page 3.)

[Bronstein 2017] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam and Pierre Vandergheynst. *Geometric deep learning: going beyond euclidean data.* IEEE Signal Processing Magazine, vol. 34, no. 4, pages 18–42, 2017. (Cited on page 8.)

[Cattivelli 2008] F. Cattivelli, C. G. Lopes and A. H. Sayed. *Diffusion recursive least-squares for distributed estimation over adaptive networks.* IEEE Transactions on Signal Processing, vol. 56, no. 5, pages 1865–1877, May 2008. (Cited on pages 4 and 61.)

[Cattivelli 2010a] Federico S Cattivelli and Ali H Sayed. *Diffusion LMS strategies for distributed estimation.* IEEE Transactions on Signal Processing, vol. 58, no. 3, pages 1035–1048, Mar. 2010. (Cited on pages 3 and 42.)

[Cattivelli 2010b] Federico S Cattivelli and Ali H Sayed. *Diffusion strategies for distributed Kalman filtering and smoothing.* IEEE Transactions on automatic control, vol. 55, no. 9, pages 2069–2084, 2010. (Cited on page 4.)

[Cattivelli 2011] Federico S Cattivelli and Ali H Sayed. *Distributed detection over adaptive networks using diffusion adaptation.* IEEE Transactions on Signal Processing, vol. 59, no. 5, pages 1917–1932, 2011. (Cited on page 5.)

[Chang 2020] Tsung-Hui Chang, Mingyi Hong, Hoi-To Wai, Xinwei Zhang and Songtao Lu. *Distributed Learning in the Non-Convex World: From Batch to Streaming Data, and Beyond.* IEEE Signal Processing Magazine, vol. 37, no. 3, pages 26–38, May 2020. (Cited on page 106.)

[Chen 2009] Yilun Chen, Yuantao Gu and Alfred O Hero. *Sparse LMS for system identification.* In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 3125–3128, Taipei, China, 2009. IEEE. (Cited on page 5.)

[Chen 2012] Jianshu Chen and Ali H Sayed. *Diffusion adaptation strategies for distributed optimization and learning over networks.* IEEE Transactions on Signal Processing, vol. 60, no. 8, pages 4289–4305, 2012. (Cited on page 3.)

[Chen 2013a] Jianshu Chen and Ali H Sayed. *Distributed Pareto optimization via diffusion strategies.* IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 2, pages 205–220, Apr. 2013. (Cited on pages 5 and 16.)

[Chen 2013b] Jie Chen, Cédric Richard and Paul Honeine. *Nonlinear unmixing of hyperspectral data based on a linear-mixture/nonlinear-fluctuation model.* IEEE Trans-

actions on Signal Processing, vol. 61, no. 2, pages 480–492, Jan. 2013. (Cited on pages 96 and 98.)

[Chen 2014a] Jie Chen, Cédric Richard and Alfred O Hero. *Nonlinear unmixing of hyperspectral images using a semiparametric model and spatial regularization.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7954–7958, Florence, Italy, 2014. IEEE. (Cited on pages 96 and 98.)

[Chen 2014b] Jie Chen, Cédric Richard, Alfred O. Hero and Ali H. Sayed. *Diffusion LMS for multitask problems with overlapping hypothesis subspaces.* In IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Reims, France, 2014. IEEE. (Cited on pages 6, 16, 42 and 61.)

[Chen 2014c] Jie Chen, Cédric Richard and Ali Sayed. *Multitask diffusion adaptation over networks.* IEEE Transactions on Signal Processing, vol. 62, no. 16, pages 4129–4144, Aug. 2014. (Cited on pages 7, 16 and 42.)

[Chen 2014d] Siheng Chen, Aliaksei Sandryhaila, Jose M. F. Moura and Jelena Kovacevic. *Signal denoising on graphs via graph filtering.* In IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 872–876, Atlanta, Georgia, USA, 2014. IEEE. (Cited on page 94.)

[Chen 2015a] Jie Chen, Cédric Richard and Ali H Sayed. *Diffusion LMS over multitask networks.* IEEE Transactions on Signal Processing, vol. 63, no. 11, pages 2733–2748, Jun. 2015. (Cited on pages 5, 7, 42, 50, 61 and 77.)

[Chen 2015b] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila and Jelena Kovačević. *Discrete Signal Processing on Graphs: Sampling Theory.* IEEE Transactions on Signal Processing, vol. 63, no. 24, pages 6510–6523, Dec. 2015. (Cited on pages 8 and 60.)

[Chen 2016] Jie Chen, Shang Kee Ting, Cédric Richard and Ali H Sayed. *Group diffusion LMS.* In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4925–4929, Shanghai, China, 2016. IEEE. (Cited on page 5.)

[Chen 2017] Jie Chen, Cédric Richard and Ali Sayed. *Multitask diffusion adaptation over networks with common latent representations.* IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 3, pages 563–579, Apr. 2017. (Cited on pages 6, 16 and 42.)

[Chung 1997] Fan RK Chung. *Spectral graph theory*. In CBMS Regional Conference Series in Mathematics 92, CA, US, 1997. American Mathematical Society. (Cited on pages 8 and 10.)

[Coutino 2019] M. Coutino, E. Isufi and G. Leus. *Advances in Distributed Graph Filtering*. IEEE Transactions on Signal Processing, vol. 67, no. 9, pages 2320–2333, May 2019. (Cited on pages 8, 61, 94 and 96.)

[Dargie 2010] Waltenegus Dargie and Christian Poellabauer. Fundamentals of wireless sensor networks: Theory and practice. John Wiley & Sons, 2010. (Cited on page 44.)

[Dees 2019] Bruno Scalzo Dees, Ljubisa Stankovic, Milos Dakovic, Anthony G Constantinides and Danilo P Mandic. *Unitary Shift Operators on a Graph*. arXiv preprint arXiv:1909.05767, 2019. (Cited on page 10.)

[Defferrard 2016] Michaël Defferrard, Xavier Bresson and Pierre Vandergheynst. *Convolutional neural networks on graphs with fast localized spectral filtering*. In Advances in Neural Information Processing Systems (NeurIPS), pages 3844–3852, Barcelona, Spain, 2016. Curran Associates, Inc. (Cited on pages 60 and 94.)

[Di Lorenzo 2012] Paolo Di Lorenzo and Ali H Sayed. *Sparse distributed learning based on diffusion adaptation*. IEEE Transactions on signal processing, vol. 61, no. 6, pages 1419–1433, 2012. (Cited on page 5.)

[Di Lorenzo 2016] Paolo Di Lorenzo, Sergio Barbarossa, Paolo Banelli and Stefania Sardellitti. *Adaptive Least Mean Squares Estimation of Graph Signals*. IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 4, pages 555–568, Dec. 2016. (Cited on pages 61 and 62.)

[Di Lorenzo 2017] Paolo Di Lorenzo, Paolo Banelli, Sergio Barbarossa and Stefania Sardellitti. *Distributed adaptive learning of graph signals*. IEEE Transactions on Signal Processing, vol. 65, no. 16, pages 4193–4208, Aug. 2017. (Cited on page 62.)

[Di Lorenzo 2018] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa and G. Leus. *Adaptive Graph Signal Processing: Algorithms and Optimal Sampling Strategies*. IEEE Transactions on Signal Processing, vol. 66, no. 13, pages 3584–3598, Jul. 2018. (Cited on page 62.)

[Di Lorenzo 2020] Paolo Di Lorenzo, Sergio Barbarossa and Stefania Sardellitti. *Distributed signal processing and optimization based on in-network subspace projections*. IEEE Transactions on Signal Processing, vol. 68, pages 2061–2076, 2020. (Cited on page 7.)

[Dimakis 2010] Alexandros G Dimakis, Soummya Kar, José MF Moura, Michael G Rabbat and Anna Scaglione. *Gossip algorithms for distributed signal processing.* Proceedings of the IEEE, vol. 98, no. 11, pages 1847–1864, 2010. (Cited on page 3.)

[Djurić 2018] P. Djurić and C. Richard. Cooperative and graph signal processing: Principles and applications. Academic Press, Elsevier, Cambridge, MA, USA, 2018. (Cited on pages 1 and 60.)

[Dong 2019] Xiaowen Dong, Dorina Thanou, Michael Rabbat and Pascal Frossard. *Learning graphs from data: A signal representation perspective.* IEEE Signal Processing Magazine, vol. 36, no. 3, pages 44–63, 2019. (Cited on page 9.)

[Frost 1972] Otis Lamont Frost. *An algorithm for linearly constrained adaptive array processing.* Proceedings of the IEEE, vol. 60, no. 8, pages 926–935, Aug. 1972. (Cited on pages 20, 31 and 56.)

[Gama 2018a] F. Gama, G. Leus, A. G. Marques and A. Ribeiro. *Convolutional Neural Networks Via Node-Varying Graph Filters.* In IEEE Data Science Workshop (DSW), pages 1–5, Lausanne, Switzerland, Jun. 2018. IEEE. (Cited on page 65.)

[Gama 2018b] Fernando Gama, Antonio G Marques, Geert Leus and Alejandro Ribeiro. *Convolutional neural network architectures for signals supported on graphs.* IEEE Transactions on Signal Processing, vol. 67, no. 4, pages 1034–1049, Feb. 2018. (Cited on pages 60 and 94.)

[Gao 2014] Wei Gao, Jie Chen, Cedric Richard and Jianguo Huang. *Online dictionary learning for kernel LMS.* IEEE Transactions on Signal Processing, vol. 62, no. 11, pages 2765–2777, 2014. (Cited on page 106.)

[Gavili 2017] Adnan Gavili and Xiao-Ping Zhang. *On the Shift Operator, Graph Frequency, and Optimal Filtering in Graph Signal Processing.* IEEE Transactions on Signal Processing, vol. 65, no. 23, pages 6303–6318, Dec. 2017. (Cited on pages 10, 68, 94 and 97.)

[Giannakis 2018] Georgios B Giannakis, Yanning Shen and Georgios Vasileios Karanikolas. *Topology identification and learning over graphs: Accounting for nonlinearities and dynamics.* Proceedings of the IEEE, vol. 106, no. 5, pages 787–807, 2018. (Cited on page 9.)

[Girault 2015a] Benjamin Girault. *Stationary graph signals using an isometric graph translation.* In 23rd European Signal Processing Conference (EUSIPCO), pages 1516–1520, Nice, France, 2015. IEEE. (Cited on page 101.)

[Girault 2015b] Benjamin Girault, Paulo Gonçalves and Eric Fleury. *Translation on graphs: An isometric shift operator.* IEEE Signal Processing Letters, vol. 22, no. 12, pages 2416–2420, Dec. 2015. (Cited on pages 10, 94 and 96.)

[Grassi 2018] Francesco Grassi, Andreas Loukas, Nathanaël Perraudin and Benjamin Ricaud. *A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs.* IEEE Transactions on Signal Processing, vol. 66, no. 3, pages 817–829, Feb. 2018. (Cited on page 62.)

[Hammond 2011] David K Hammond, Pierre Vandergheynst and Rémi Gribonval. *Wavelets on graphs via spectral graph theory.* Applied and Computational Harmonic Analysis, vol. 30, no. 2, pages 129–150, 2011. (Cited on page 94.)

[Harrane 2019] I. E. K. Harrane, R. Flamary and C. Richard. *On Reducing the Communication Cost of the Diffusion LMS Algorithm.* IEEE Transactions on Signal and Information Processing over Networks, vol. 5, no. 1, pages 100–112, 2019. (Cited on page 5.)

[Haykin 2002] Simon Haykin. Adaptive filter theory. Prentice-Hall, NJ, USA, 2002. (Cited on page 2.)

[Horn 2012] Roger A Horn and Charles R Johnson. Matrix analysis. Cambridge University Press, Cambridge, U.K., 2012. (Cited on pages 9 and 57.)

[Hua 2017a] Fei Hua, Roula Nassif, Cédric Richard and Haiyan Wang. *Penalty-based multitask estimation with non-local linear equality constraints.* In IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pages 1–5, Curacao, Netherlands Antilles, Dec. 2017. IEEE. (Cited on pages 7 and 41.)

[Hua 2017b] Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Jianguo Huang. *Penalty-based multitask distributed adaptation over networks with constraints*, 2017. 51st Asilomar Conference on Signals, Systems, and Computers (ASILOMAR). (Cited on pages 7, 16, 42, 43 and 56.)

[Hua 2018a] Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H Sayed. *Decentralized clustering for node-variant graph filtering with graph diffusion LMS*. In 52nd Asilomar Conference on Signals, Systems, and Computers (ASILOMAR), pages 1418–1422, Pacific Grove, CA, USA, Oct. 2018. IEEE. (Cited on page 60.)

[Hua 2018b] Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H Sayed. *A Preconditioned Graph Diffusion LMS for Adaptive Graph Signal Processing*. In 26th European Signal Processing Conference (EUSIPCO), pages 111–115, Rome, Italy, Sep. 2018. IEEE. (Cited on pages 60, 96 and 97.)

[Hua 2019] Fei Hua, Cédric Richard, Jie Chen, Haiyan Wang, Pierre Borgnat and Paulo Gonçalves. *Learning Combination of Graph Filters for Graph Signal Modeling*. IEEE Signal Processing Letters, vol. 26, no. 12, pages 1912–1916, Dec. 2019. (Cited on page 93.)

[Hua 2020a] Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. *Diffusion LMS with Communication Delays: Stability and Performance Analysis*. IEEE Signal Processing Letters, vol. 27, pages 730–734, 2020. (Cited on pages 5, 41 and 48.)

[Hua 2020b] Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang and Ali H. Sayed. *Online distributed learning over graphs with multitask graph-filter models*. IEEE Transactions on Signal and Information Processing over Networks, vol. 6, no. 1, pages 63–77, 2020. (Cited on page 60.)

[Isufi 2016] Elvin Isufi, Geert Leus and Paolo Banelli. *2-dimensional finite impulse response graph-temporal filters*. In IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 405–409, Greater Washington, D.C., USA, 2016. IEEE. (Cited on pages 62 and 64.)

[Isufi 2017a] Elvin Isufi, Paolo Banelli, Paolo Di Lorenzo and Geert Leus. *Observing and Tracking Bandlimited Graph Processes*. arXiv preprint arXiv:1712.00404, 2017. (Cited on page 62.)

[Isufi 2017b] Elvin Isufi, Andreas Loukas, Andrea Simonetto and Geert Leus. *Autoregressive moving average graph filtering*. IEEE Transactions on Signal Processing, vol. 65, no. 2, pages 274–288, Jan. 2017. (Cited on pages 8, 61, 94, 95 and 97.)

[Isufi 2017c] Elvin Isufi, Andreas Loukas, Andrea Simonetto and Geert Leus. *Filtering Random Graph Processes Over Random Time-Varying Graphs*. IEEE Transactions on Signal Processing, vol. 65, no. 16, pages 4406–4421, Aug. 2017. (Cited on page 62.)

[Isufi 2018] Elvin Isufi, Paolo Di Lorenzo, Paolo Banelli and Geert Leus. *Distributed wiener-based reconstruction of graph signals*. In IEEE Workshop on Statistical Signal Processing (SSP), pages 21–25, Freiburg, Germany, 2018. IEEE. (Cited on page 94.)

[Isufi 2019] E. Isufi, A. Loukas, N. Perraudin and G. Leus. *Forecasting Time Series With VARMA Recursions on Graphs*. IEEE Transactions on Signal Processing, vol. 67, no. 18, pages 4870–4885, Sep. 2019. (Cited on page 61.)

[Jin 2020] Danqi Jin, Jie Chen, Cedric Richard, Jingdong Chen and Ali H Sayed. *Affine combination of diffusion strategies over networks*. IEEE Transactions on Signal Processing, vol. 68, pages 2087–2104, 2020. (Cited on page 5.)

[Jolliffe 2016] Ian T. Jolliffe and Jorge Cadima. *Principal component analysis: a review and recent developments*. Philosophical Transactions of the Royal Society A, vol. 374, no. 2065, pages 1–16, 2016. (Cited on page 78.)

[Kar 2011] Soummya Kar and José MF Moura. *Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs*. IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 4, pages 674–690, 2011. (Cited on page 3.)

[Kar 2012] Soummya Kar, José MF Moura and Kavita Ramanan. *Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication*. IEEE Transactions on Information Theory, vol. 58, no. 6, pages 3575–3605, 2012. (Cited on page 3.)

[Khalili 2011] Azam Khalili, Mohammad Ali Tinati, Amir Rastegarnia and Jonathon A Chambers. *Steady-state analysis of diffusion LMS adaptive networks with noisy links*. IEEE Transactions on Signal Processing, vol. 60, no. 2, pages 974–979, 2011. (Cited on page 5.)

[Khawatmi 2017] Sahar Khawatmi, Ali H Sayed and Abdelhak M Zoubir. *Decentralized Clustering and Linking by Networked Agents*. IEEE Transactions on Signal Processing, vol. 65, no. 13, pages 3526–3537, Jul. 2017. (Cited on pages 7 and 77.)

[Koning 1991] Ruud H. Koning, Heinz Neudecker and Tom Wansbeek. *Block Kronecker products and the vecb operator*. Linear Algebra and its Applications, vol. 149, pages 165–184, 1991. (Cited on pages 27 and 36.)

[Koppel 2018] A. Koppel, S. Paternain, C. Richard and A. Ribeiro. *Decentralized Online Learning With Kernels*. IEEE Transactions on Signal Processing, vol. 66, no. 12, pages 3240–3255, Jun. 2018. (Cited on page 106.)

[Lewis 2011] Ted G Lewis. Network science: Theory and applications. John Wiley & Sons, Hoboken, NJ, USA, 2011. (Cited on page 2.)

[Li 2009] L. Li and J. Chambers. *Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology*. In IEEE Workshop on Statistical Signal Processing (SSP), pages 757–760, Cardiff, UK, 2009. (Cited on pages 4 and 61.)

[Liu 2019] J. Liu, E. Isufi and G. Leus. *Filter Design for Autoregressive Moving Average Graph Filters*. IEEE Transactions on Signal and Information Processing over Networks, vol. 5, no. 1, pages 47–60, Mar. 2019. (Cited on pages 8 and 61.)

[Lopes 2007] Cassio G Lopes and Ali H Sayed. *Incremental adaptive strategies over distributed networks*. IEEE Transactions on Signal Processing, vol. 55, no. 8, pages 4064–4077, 2007. (Cited on page 3.)

[Lopes 2008] C. G. Lopes and A. H. Sayed. *Diffusion least-mean squares over adaptive networks: Formulation and performance analysis*. IEEE Transactions on Signal Processing, vol. 56, no. 7, pages 3122–3136, Jul. 2008. (Cited on pages 3 and 61.)

[Loukas 2015] Andreas Loukas, Andrea Simonetto and Geert Leus. *Distributed Autoregressive Moving Average Graph Filters*. IEEE Signal Processing Letters, vol. 22, no. 11, pages 1931–1935, Nov. 2015. (Cited on pages 61 and 94.)

[Loukas 2016] Andreas Loukas and Nathanaël Perraudin. *Stationary time-vertex signal processing*. arXiv preprint arXiv:1611.00255, 2016. (Cited on page 61.)

[Loukas 2017] A. Loukas, E. Isufi and N. Perraudin. *Predicting the evolution of stationary graph signals*. In 51st Asilomar Conference on Signals, Systems, and Computers (ASILOMAR), pages 60–64, Pacific Grove, CA, USA, Oct. 2017. IEEE. (Cited on page 61.)

[Lovász 1993] László Lovász*et al.* *Random walks on graphs: A survey*, 1993. (Cited on page 9.)

[Luenberger 2015] David G Luenberger and Yinyu Ye. Linear and nonlinear programming, volume 228. Springer Cham, Cham, Switzerland, 2015. (Cited on pages 20 and 44.)

[Mateos 2009] Gonzalo Mateos, Ioannis D Schizas and Georgios B Giannakis. *Distributed recursive least-squares for consensus-based in-network adaptive estimation.* IEEE Transactions on Signal Processing, vol. 57, no. 11, pages 4583–4588, 2009. (Cited on page 4.)

[Mateos 2019] Gonzalo Mateos, Santiago Segarra, Antonio G Marques and Alejandro Ribeiro. *Connecting the dots: Identifying network structure via graph signal processing.* IEEE Signal Processing Magazine, vol. 36, no. 3, pages 16–43, 2019. (Cited on page 9.)

[Mei 2017] Jonathan Mei and José MF Moura. *Signal processing on graphs: Causal modeling of unstructured data.* IEEE Transactions on Signal Processing, vol. 65, no. 8, pages 2077–2092, Apr. 2017. (Cited on pages 61 and 80.)

[Meyer 2016] Florian Meyer, Ondrej Hlinka, Henk Wymeersch, Erwin Riegler and Franz Hlawatsch. *Distributed localization and tracking of mobile networks including noncooperative objects.* IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 1, pages 57–71, Mar. 2016. (Cited on page 42.)

[Mota 2012] João FC Mota, João MF Xavier, Pedro MQ Aguiar and Markus Puschel. *Distributed basis pursuit.* IEEE Transactions on Signal Processing, vol. 60, no. 4, pages 1942–1956, Apr. 2012. (Cited on pages 7, 16 and 42.)

[Narang 2013] Sunil K Narang and Antonio Ortega. *Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs.* IEEE Transactions on Signal Processing, vol. 61, no. 19, pages 4673–4685, Oct. 2013. (Cited on page 94.)

[Nassif 2016a] Roula Nassif, Cédric Richard, André Ferrari and Ali H Sayed. *Multitask Diffusion Adaptation Over Asynchronous Networks.* IEEE Transactions on Signal Processing, vol. 64, no. 11, pages 2835–2850, Jun. 2016. (Cited on pages 7, 16 and 61.)

[Nassif 2016b] Roula Nassif, Cédric Richard, André Ferrari and Ali H Sayed. *Proximal multitask learning over networks with sparsity-inducing coregularization.* IEEE Trans-

actions on Signal Processing, vol. 64, no. 23, pages 6329–6344, Dec. 2016. (Cited on pages 7, 16, 42, 50 and 61.)

[Nassif 2017a] Roula Nassif, Cédric Richard, Jie Chen and Ali H Sayed. *A graph diffusion LMS strategy for adaptive graph signal processing.* In 51st Asilomar Conference on Signals, Systems, and Computers (ASILOMAR), pages 1973–1976, Pacific Grove, CA, USA, Oct. 2017. IEEE. (Cited on page 64.)

[Nassif 2017b] Roula Nassif, Cédric Richard, André Ferrari and Ali H Sayed. *Diffusion LMS for Multitask Problems with Local Linear Equality Constraints.* IEEE Transactions on Signal Processing, vol. 65, no. 19, pages 4979–4993, 2017. (Cited on pages 7, 17, 42, 43 and 56.)

[Nassif 2018] Roula Nassif, Cédric Richard, Jie Chen and Ali H Sayed. *Distributed Diffusion Adaptation Over Graph Signals.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4129–4133, Calgary, AB, Canada, Apr. 2018. IEEE. (Cited on pages 64, 67, 94 and 96.)

[Nassif 2019] R. Nassif, S. Vlaski, C. Richard and A. H. Sayed. *A Regularization Framework for Learning Over Multitask Graphs.* IEEE Signal Processing Letters, vol. 26, no. 2, pages 297–301, Feb. 2019. (Cited on pages 7 and 60.)

[Nassif 2020a] Roula Nassif, Stefan Vlaski, Cédric Richard, Jie Chen and Ali H Sayed. *Multitask Learning over Graphs.* IEEE Signal Processing Magazine, vol. 37, no. 3, page 14–25, May 2020. (Cited on page 5.)

[Nassif 2020b] Roula Nassif, Stefan Vlaski and Ali H Sayed. *Adaptation and learning over networks under subspace constraints Part II: Performance Analysis.* IEEE Transactions on Signal Processing, 2020. (Cited on page 7.)

[Nassif 2020c] Roula Nassif, Stefan Vlaski and Ali H Sayed. *Adaptation and learning over networks under subspace constraints—Part I: Stability analysis.* IEEE Transactions on Signal Processing, vol. 68, pages 1346–1360, 2020. (Cited on page 7.)

[Nedic 2001] Angelia Nedic and Dimitri P Bertsekas. *Incremental subgradient methods for nondifferentiable optimization.* SIAM Journal on Optimization, vol. 12, no. 1, pages 109–138, 2001. (Cited on page 3.)

[Nedic 2009] Angelia Nedic and Asuman Ozdaglar. *Distributed subgradient methods for multi-agent optimization.* IEEE Transactions on Automatic Control, vol. 54, no. 1, pages 48–61, 2009. (Cited on page 3.)

[Newman 2010] Mark Newman. Networks: An introduction. Oxford University Press, Oxford, U.K., 2010. (Cited on page 2.)

[noa] *1981-2010 U.S. climate normals.* [Online]. Available: https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/climate-normals/1981-2010-normals-data. (Cited on page 86.)

[Olfati-Saber 2007] Reza Olfati-Saber, J Alex Fax and Richard M Murray. *Consensus and cooperation in networked multi-agent systems.* Proceedings of the IEEE, vol. 95, no. 1, pages 215–233, Jan. 2007. (Cited on pages 3 and 42.)

[Ortega 2018] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura and Pierre Vandergheynst. *Graph Signal Processing: Overview, Challenges, and Applications.* Proceedings of the IEEE, vol. 106, no. 5, pages 808–828, May 2018. (Cited on pages 8, 60 and 63.)

[Perraudin 2014] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst and David K Hammond. *GSPBOX: A toolbox for signal processing on graphs.* arXiv preprint arXiv:1408.5781, 2014. (Cited on pages 80 and 101.)

[Petersen 2012] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*, 2012. (Cited on page 37.)

[Piggott 2016] Marc James Piggott and Victor Solo. *Diffusion LMS with correlated regressors Part–I: Realization-wise stability.* IEEE Transactions on Signal Processing, vol. 64, no. 21, pages 5473–5484, Jun. 2016. (Cited on page 5.)

[Piggott 2017] Marc James Piggott and Victor Solo. *Diffusion LMS with correlated regressors–Part II: Performance.* IEEE Transactions on Signal Processing, vol. 65, no. 15, pages 3934–3947, May 2017. (Cited on page 5.)

[Plata-Chaves 2015] Jorge Plata-Chaves, Nikola Bogdanović and Kostas Berberidis. *Distributed diffusion-based LMS for node-specific adaptive parameter estimation.* IEEE Transactions on Signal Processing, vol. 63, no. 13, pages 3448–3460, Jul. 2015. (Cited on pages 5, 16, 42 and 50.)

[Plata-Chaves 2016] Jorge Plata-Chaves, Mohamad Hasan Bahari, Marc Moonen and Alexander Bertrand. *Unsupervised diffusion-based LMS for node-specific parameter estimation over wireless sensor networks.* In IEEE International Conference on

Acoustics, Speech and Signal Processing (ICASSP), pages 4159–4163, Shanghai, China, 2016. IEEE. (Cited on pages 5, 7, 16 and 77.)

[Plata-Chaves 2017] Jorge Plata-Chaves, Alexander Bertrand, Marc Moonen, Sergios Theodoridis and Abdelhak M Zoubir. *Heterogeneous and multitask wireless sensor networks—algorithms, applications, and challenges.* IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 3, pages 450–465, 2017. (Cited on page 5.)

[Polyak 1987] Boris T Polyak. Introduction to optimization. Optimization Software, New York, USA, 1987. (Cited on pages 21 and 44.)

[Pradhan 2018] H. Pradhan, A. S. Bedi, A. Koppel and K. Rajawat. *Exact nonparametric decentralized online optimization.* In IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 643–647, Anaheim, CA, USA, 2018. IEEE. (Cited on page 106.)

[Qiu 2017] Kai Qiu, Xianghui Mao, Xinyue Shen, Xiaohan Wang, Tiejian Li and Yuantao Gu. *Time-varying graph signal reconstruction.* IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pages 870–883, Sep. 2017. (Cited on page 62.)

[Rabbat 2005] Michael G Rabbat and Robert D Nowak. *Quantized incremental algorithms for distributed optimization.* IEEE Journal on Selected Areas in Communications, vol. 23, no. 4, pages 798–808, Apr. 2005. (Cited on page 3.)

[Rakotomamonjy 2008] Alain Rakotomamonjy, Francis R Bach, Stéphane Canu and Yves Grandvalet. *SimpleMKL.* Journal of Machine Learning Research, vol. 9, pages 2491–2521, 2008. (Cited on pages 96 and 98.)

[Richard 2009] Cédric Richard, José Carlos M Bermudez and Paul Honeine. *Online prediction of time series data with kernels.* IEEE Transactions on Signal Processing, vol. 57, no. 3, pages 1058–1067, 2009. (Cited on page 106.)

[Romero 2017] Daniel Romero, Vassilis N Ioannidis and Georgios B Giannakis. *Kernel-based reconstruction of space-time functions on dynamic graphs.* IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pages 856–869, Sep. 2017. (Cited on pages 62 and 87.)

[Sandryhaila 2013] Aliaksei Sandryhaila and José MF Moura. *Discrete signal processing on graphs.* IEEE Transactions on Signal Processing, vol. 61, no. 7, pages 1644–1656, Apr. 2013. (Cited on pages 9, 10, 60, 61, 63, 64 and 94.)

[Sandryhaila 2014a] Aliaksei Sandryhaila and Jose MF Moura. *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure.* IEEE Signal Processing Magazine, vol. 31, no. 5, pages 80–90, Sep. 2014. (Cited on pages 8 and 60.)

[Sandryhaila 2014b] Aliaksei Sandryhaila and Jose MF Moura. *Discrete signal processing on graphs: Frequency analysis.* IEEE Transactions on Signal Processing, vol. 62, no. 12, pages 3042–3054, 2014. (Cited on page 8.)

[Sayed 2003] A.H. Sayed. Fundamentals of adaptive filtering. Wiley, Hoboken, NJ, USA, 2003. (Cited on pages 22, 37, 50 and 74.)

[Sayed 2008] Ali H Sayed. Adaptive filters. John Wiley & Sons, Hoboken, NJ, USA, 2008. (Cited on pages 2, 22, 27, 28, 45, 50, 66, 68, 69, 70 and 72.)

[Sayed 2013] Ali H Sayed, Sheng-Yuan Tu, Jianshu Chen, Xiaochuan Zhao and Zaid J Towfic. *Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior.* IEEE Signal Processing Magazine, vol. 30, no. 3, pages 155–171, May. 2013. (Cited on pages 2, 42, 50 and 61.)

[Sayed 2014a] Ali H Sayed. *Adaptation, learning, and optimization over networks.* Foundations and Trends® in Machine Learning, vol. 7, no. 4-5, pages 311–801, 2014. (Cited on pages 1, 2, 3, 16, 22, 28, 36, 42 and 62.)

[Sayed 2014b] Ali H Sayed. *Adaptive networks.* Proceedings of the IEEE, vol. 102, no. 4, pages 460–497, Apr. 2014. (Cited on pages 2, 16 and 61.)

[Sayed 2014c] Ali H Sayed. *Diffusion adaptation over networks.* In Sergios Theodoridis and Rama Chellappa, Editors, Academic Press Library in Signal Processing, volume 3, pages 322–454. Academic Press, Elsevier, Cambridge, MA, USA, 2014. (Cited on pages 16, 28, 42, 50, 61, 67, 72, 73, 90 and 91.)

[Schizas 2009] Ioannis D Schizas, Gonzalo Mateos and Georgios B Giannakis. *Distributed LMS for consensus-based in-network adaptive processing.* IEEE Transactions on Signal Processing, vol. 57, no. 6, pages 2365–2382, 2009. (Cited on page 4.)

[Segarra 2017] Santiago Segarra, Antonio G Marques and Alejandro Ribeiro. *Optimal graph-filter design and applications to distributed linear network operators.* IEEE Transactions on Signal Processing, vol. 65, no. 15, pages 4117–4131, Aug. 2017. (Cited on pages 8, 61, 65, 94 and 96.)

[Sevi 2018a] Harry Sevi, Gabriel Rilling and Pierre Borgnat. *Harmonic analysis on directed graphs and applications: from Fourier analysis to wavelets.* arXiv preprint arXiv:1811.11636, 2018. (Cited on pages 10, 95, 96 and 101.)

[Sevi 2018b] Harry Sevi, Gabriel Rilling and Pierre Borgnat. *Modeling signals over directed graphs through filtering.* In IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 718–722, Anaheim, CA, USA, 2018. IEEE. (Cited on pages 95 and 96.)

[Shen 2012] Chao Shen, Tsung-Hui Chang, Kun-Yu Wang, Zhengding Qiu and Chong-Yung Chi. *Distributed robust multicell coordinated beamforming with imperfect CSI: An ADMM approach.* IEEE Transactions on signal processing, vol. 60, no. 6, pages 2988–3003, Jun. 2012. (Cited on pages 7, 16 and 42.)

[Shi 2015] Xuesong Shi, Hui Feng, Muyuan Zhai, Tao Yang and Bo Hu. *Infinite Impulse Response Graph Filters in Wireless Sensor Networks.* IEEE Signal Processing Letters, vol. 22, no. 8, pages 1113–1117, Aug. 2015. (Cited on pages 8, 61 and 94.)

[Shuman 2013] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega and Pierre Vandergheynst. *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains.* IEEE Signal Processing Magazine, vol. 30, no. 3, pages 83–98, May 2013. (Cited on pages 8, 9, 60, 61, 63, 64 and 94.)

[Shuman 2018] David I Shuman, Pierre Vandergheynst, Daniel Kressner and Pascal Frossard. *Distributed Signal Processing via Chebyshev Polynomial Approximation.* IEEE Transactions on Signal and Information Processing over Networks, vol. 4, no. 4, pages 736–751, Dec. 2018. (Cited on page 61.)

[Simon 2013] Noah Simon, Jerome Friedman, Trevor Hastie and Robert Tibshirani. *A Sparse-Group Lasso.* Journal of Computational and Graphical Statistics, vol. 22, no. 2, pages 231–245, 2013. (Cited on page 101.)

[Singh 2016] Rahul Singh, Abhishek Chakraborty and BS Manoj. *Graph Fourier transform based on directed Laplacian.* In IEEE International Conference on Signal Processing and Communications (SPCOM), pages 1–5, Bangalore, India, 2016. IEEE. (Cited on pages 10 and 94.)

[Tanaka 2014] Yuichi Tanaka and Akie Sakiyama. *M-channel oversampled graph filter banks.* IEEE Transactions on Signal Processing, vol. 62, no. 14, pages 3578–3590, Jul. 2014. (Cited on page 94.)

[Towfic 2014] Zaid J Towfic and Ali H Sayed. *Adaptive penalty-based distributed stochastic convex optimization.* IEEE Transactions on Signal Processing, vol. 62, no. 15, pages 3924–3938, Aug. 2014. (Cited on pages 16, 20 and 44.)

[Towfic 2015] Zaid J Towfic and Ali H Sayed. *Stability and performance limits of adaptive primal-dual networks.* IEEE Transactions on Signal Processing, vol. 63, no. 11, pages 2888–2903, Jun. 2015. (Cited on pages 4 and 42.)

[Tremblay 2016a] Nicolas Tremblay and Pierre Borgnat. *Subgraph-based filterbanks for graph signals.* IEEE Transactions on Signal Processing, vol. 64, no. 15, pages 3827–3840, Aug. 2016. (Cited on page 94.)

[Tremblay 2016b] Nicolas Tremblay, Gilles Puy, Rémi Gribonval and Pierre Vandergheynst. *Compressive spectral clustering.* In International Conference on Machine Learning (ICML), pages 1002–1011, New York City, NY, USA, 2016. IMLS. (Cited on page 94.)

[Tsitsvero 2016] M. Tsitsvero, S. Barbarossa and P. Di Lorenzo. *Signals on Graphs: Uncertainty Principle and Sampling.* IEEE Transactions on Signal Processing, vol. 64, no. 18, pages 4845–4860, Sep. 2016. (Cited on pages 8 and 60.)

[Tu 2012] Sheng-Yuan Tu and Ali H Sayed. *Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks.* IEEE Transactions on Signal Processing, vol. 60, no. 12, pages 6217–6234, Dec. 2012. (Cited on pages 4 and 42.)

[Vlaski 2019] Stefan Vlaski and Ali H Sayed. *Distributed Learning in Non-Convex Environments–Part I: Agreement at a Linear Rate.* arXiv preprint arXiv:1907.01848, 2019. (Cited on page 106.)

[Wang 2015] Xiaohan Wang, Pengfei Liu and Yuantao Gu. *Local-set-based graph signal reconstruction.* IEEE Transactions on Signal Processing, vol. 63, no. 9, pages 2432–2444, 2015. (Cited on page 8.)

[Wang 2017] Yuan Wang, Wee Peng Tay and Wuhua Hu. *A multitask diffusion strategy with optimized inter-cluster cooperation.* IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 3, pages 504–517, 2017. (Cited on pages 7 and 42.)

[Widrow 1985] Widrow, Bernard, Stearns and SamuelD. Adaptive signal processing. Prentice-Hall, NJ, USA, 1985. (Cited on page 2.)

[Wu 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang and S Yu Philip. *A comprehensive survey on graph neural networks*. IEEE Transactions on Neural Networks and Learning Systems, 2020. (Cited on page 8.)

[Xiao 2005] L. Xiao, S. Boyd and S. Lall. *A scheme for robust distributed sensor fusion based on average consensus*. In International Symposium on Information Processing in Sensor Networks (IPSN), pages 63–70, Boise, ID, USA, 2005. IEEE. (Cited on pages 3, 42 and 67.)

[Xie 2012] Le Xie, Dae-Hyun Choi, Soummya Kar and H Vincent Poor. *Fully distributed state estimation for wide-area monitoring systems*. IEEE Transactions on Smart Grid, vol. 3, no. 3, pages 1154–1169, Sep. 2012. (Cited on page 42.)

[Xie 2018] Siyu Xie and Lei Guo. *Analysis of distributed adaptive filters based on diffusion strategies over sensor networks*. IEEE Transactions on Automatic Control, vol. 63, no. 11, pages 3643–3658, 2018. (Cited on page 4.)

[Xiong 2019] Naixue Xiong, Mou Wu, Victor CM Leung and Laurence T Yang. *The Effective Cooperative Diffusion Strategies With Adaptation Ability by Learning Across Adaptive Network-Wide Systems*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019. (Cited on page 5.)

[Yu 2017] Chung-Kai Yu and Ali H Sayed. *Learning by Networked Agents under Partial Information*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3874–3878, New Orleans, LA, USA, 2017. IEEE, IEEE. (Cited on page 16.)

[Yuan 2019a] K. Yuan, B. Ying, X. Zhao and Ali H. Sayed. *Exact diffusion for distributed optimization and learning–Part I: Algorithm development*. IEEE Transactions on Signal Processing, vol. 67, no. 3, pages 708–723, Feb. 2019. (Cited on page 5.)

[Yuan 2019b] Kun Yuan, Sulaiman A Alghunaim, Bicheng Ying and Ali H Sayed. *On the performance of exact diffusion over adaptive networks*. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 4898–4903, Nice, France, 2019. IEEE. (Cited on page 5.)

[Zhang 2014] Yi Zhang, Wee Peng Tay, Kwok Hung Li and Dominique Gaïti. *Distributed boundary estimation for spectrum sensing in cognitive radio networks*. IEEE Journal on Selected Areas in Communications, vol. 32, no. 11, pages 1961–1973, Nov. 2014. (Cited on page 42.)

[Zhang 2015] Cha Zhang, Philip Chou and Dinei Florencio. *Graph signal processing–a probabilistic framework*. Rapport technique MSR-TR-2015-31, Microsoft Research, WA, USA, April 2015. (Cited on page 9.)

[Zhang 2020] Ziwei Zhang, Peng Cui and Wenwu Zhu. *Deep learning on graphs: A survey*. IEEE Transactions on Knowledge and Data Engineering, 2020. (Cited on page 8.)

[Zhao 2012] Xiaochuan Zhao, Sheng-Yuan Tu and Ali H Sayed. *Diffusion adaptation over networks under imperfect information exchange and non-stationary data*. IEEE Transactions on Signal Processing, vol. 60, no. 7, pages 3460–3475, 2012. (Cited on page 5.)

[Zhao 2015a] Xiaochuan Zhao and Ali H Sayed. *Asynchronous adaptation and learning over networks–Part I: Modeling and stability analysis*. IEEE Transactions on Signal Processing, vol. 63, no. 4, pages 811–826, Feb. 2015. (Cited on page 5.)

[Zhao 2015b] Xiaochuan Zhao and Ali H. Sayed. *Asynchronous Adaptation and Learning Over Networks–Part II: Performance Analysis*. IEEE Transactions on Signal Processing, vol. 63, no. 4, pages 827–842, Feb. 2015. (Cited on pages 5 and 28.)

[Zhao 2015c] Xiaochuan Zhao and Ali H Sayed. *Distributed clustering and learning over networks*. IEEE Transactions on Signal Processing, vol. 63, no. 13, pages 3285–3300, Jul. 2015. (Cited on pages 7 and 77.)