Scalable Estimation on Linear and Nonlinear Regression Models via Decentralized Processing: Adaptive LMS Filter and Gaussian Process Regression

by Ayano NAKAI

Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Informatics

Supervisor: Toshiyuki TANAKA

November 2021

KYOTO UNIVERSITY

Abstract

Large amounts of data collected from a variety of Internet of Things (IoT) devices or various services on the Internet give us rich information but in general require high computational and/or communication costs and complicated processing. Decentralization of the centralized data processing is a solution that enables scalable data treatments. The idea of the decentralization is also beneficial for in-network processing in terms of scalability in network size. However, performance of decentralized processing can be degraded compared with that of the centralized processing when the decentralized processing is derived by approximately dividing the centralized processing. The purpose of the thesis is to enhance performance of such decentralized processing.

The thesis covers two estimation problems, in-network adaptive least-mean-square (LMS) filter and Gaussian process regression (GPR), as the specific themes of the decentralization. The former aims at tracking an unknown deterministic vector by using measurements at nodes obtained via linear regression models in the network such as sensor networks. Full decentralization of the algorithm is achieved by approximately dividing a cost function and by performing estimation in cooperation with neighboring nodes. In the latter, desired outputs are predicted by using training data under the assumption that the relation follows nonlinear Gaussian process regression models. The algorithm yields flexible prediction but requires a high computational cost. Partial decentralization of GPR is realized by dividing the data and then aggregating locally processed estimations.

We tackle improvements of the decentralized algorithms. For the fully decentralized adaptive LMS filter, we adopt a more efficient cooperation method known as message propagation to reduce performance degradation caused by the approximate divisions of the problem. We propose two types of algorithms with different use of the message propagation method and extend the algorithm to be a sparsity-aware one. For the partially decentralized GPR, we provide a reinterpretation of a conventional decentralization by introducing the idea of sketching and propose a novel algorithm that allows to balance computational complexity and performance. The efficiency of the proposed algorithms is evaluated via computer simulations.

Acknowledgments

I would like to thank my supervisor Professor Toshiyuki Tanaka for his thoughtful guidance through my doctoral course. I am also grateful to the members of my thesis committee, Professor Hidetoshi Shimodaira and Associate Professor Kazunori Sakurama for their insightful comments. I also appreciate Professor Kazunori Hayashi for his kind support since my master's course. I have had the encouragement of Associate Professor Tomoyuki Obuchi. I am grateful for the assistance given by all members of our laboratory, Mathematical Information Systems Group. Finally, I thank my family for their generous support.

List of Figures

List of Tables

List of Publications

Related Journal Papers

1. **A. Nakai-Kasai** and T. Tanaka, “Nested aggregation of experts using inducing points for approximated Gaussian process regression,” Machine Learning, Springer. (accepted for publication)[1]

2. **A. Nakai-Kasai** and K. Hayashi, “An acceleration method of sparse diffusion LMS based on message propagation,”' IEICE Trans. Commun., vol. E104-B, no.2, pp. 141–148, Feb. 2021. Copyright© 2021 IEICE[2]

3. **A. Nakai-Kasai** and K. Hayashi, “Diffusion LMS based on message passing algorithm,” IEEE Access, vol. 7, pp. 47022–47033, Apr. 2019. Copyright© 2019 IEEE

Related International Conference Proceedings

4. **A. Nakai-Kasai** and K. Hayashi, “Optimal combination weight for sparse diffusion least-mean-square based on consensus propagation,” Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2020), pp. 228–235, virtual conference, Dec. 2020. Copyright© 2020 IEEE

5. **A. Nakai-Kasai** and K. Hayashi, “Diffusion strategies based on consensus propagation,” Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2018), no. TU-A1-9.15, Honolulu, USA, Nov. 2018. Copyright© 2018 IEEE

6. **A. Nakai** and K. Hayashi, “An adaptive combination rule for diffusion LMS based on consensus propagation,” IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2018), pp. 3839–3843, Calgary, Canada, Apr. 2018. Copyright© 2018 IEEE

[1]Reproduced with permission from Springer Nature.

[2]Part I in this dissertation is based on “An acceleration method of sparse diffusion LMS based on message propagation” [2], by the same author, which appeared in IEICE Transactions on Communications, Copyright©2021 IEICE. The material in this dissertation was presented in part at IEICE Transactions on Communications [2], and a part of the figures of this dissertation is reused from [2] under the permission of the IEICE.

7. **A. Nakai** and K. Hayashi, "Diffusion LMS using consensus propagation," Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2017), pp. 943–948, Kuala Lumpur, Malaysia, Dec. 2017. Copyright© 2017 IEEE

Other Journal Paper

8. R. Hayakawa, **A. Nakai-Kasai**, and K. Hayashi, "Discreteness and group sparsity aware detection for uplink overloaded MU-MIMO systems," APSIPA Transactions on Signal and Information Processing, vol. 9, no. E21, pp. 1–12, Oct. 2020.

Other International Conference Proceedings

9. K. Hayashi, **A. Nakai-Kasai**, A. Hirayama, H. Honda, T. Sasaki, H. Yasukawa, and R. Hayakawa, "An overloaded IoT signal detection method using non-convex sparse regularizers," Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2020), pp. 1490–1496, virtual conference, Dec. 2020.

10. K. Hayashi, **A. Nakai-Kasai**, and R. Hayakawa, "An overloaded SC-CP IoT signal detection method via sparse complex discrete-valued vector reconstruction," Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2019), pp. 1473–1478, Lanzhou, China, Nov. 2019.

11. K. Hayashi, **A. Nakai-Kasai**, R. Hayakawa, and S. Ha, "Uplink overloaded MU-MIMO OFDM signal detection methods using convex optimization," Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2018), pp. 1421–1427, Honolulu, USA, Nov. 2018.

Contents

1: Introduction

1.1 Demand for Scalability

Today large amounts of data (or big data) are collected from a lot of sensors placed in equipment or spaces and from various services on the Internet. Improvement of hardware performance has been promoting centralized analysis of such data, which is providing new generation mobile communication systems or practical realization of machine learning technologies (Kambatla et al., 2014; Liu et al., 2020b). However, the more we collect such data, generally the more difficulties we are faced with in processing them. Large amounts of data require high computational and/or communication cost, and complicate the centralized processing operations. For scalability to the large amount of data, the idea of parallel or distributed computing can be incorporated into algorithm designs (Dean and Ghemawat, 2004; Low et al., 2010). The data themselves or (a part of) centralized processes are exactly or approximately divided into sub-data or sub-tasks, respectively, which are assigned to multiple processors or computers. This enables us to decentralize the processing and handle large amounts of data while keeping the computational cost low.

On the other hand, decentralized processing is also gathering much attention in a perspective of scalability to large networks, especially in wireless communication field. One of the important technologies for new standards, 5G or 6G, is communication including or among Internet of Things (IoT) devices having abilities of wireless communication, computation, and sensing (Gupta and Jha, 2015; Zhang et al., 2019). The number of sensors equipped with the IoT devices is said to become a million per square kilometer (Bi et al., 2015). This causes an enormous increase in data traffic in a general centralized processing, which may result in communication delays or failures of a central data processing unit. Moreover, the diversity of data collected from different IoT devices complicates the processing. Even in such a case, exact or approximate division of a part or all of the data processing can be one of the solutions of the problems for preventing the concentration of load at the central unit. The divided processes are assigned to local devices, which enables decentralized processing in networks. For instance, in edge computing (Li et al., 2018), data are partially processed at edge servers before being sent to the cloud; in federated learning (Konečnỳ et al., 2016), data are learned at local IoT devices and then sent to the central server; or in application to sensor networks or industrial IoT (Dimakis et al., 2008; Savazzi et al., 2020), devices process data by themselves and

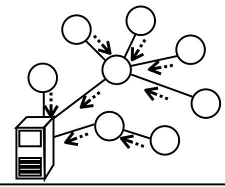
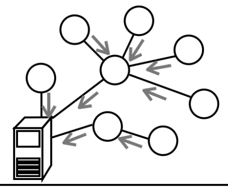
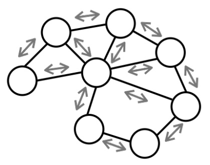
	Centralized	Decentralized	
		Partially distributed	Fully distributed
processing	process at central unit 	process at local and aggregate at central unit 	process at local 
characteristics	• all at once • high speed/cost • highly-biased load	• sequentially or simultaneously • privacy consideration • low speed/cost	
		• little-biased load	• unbiased load
examples	• general communication • batch processing	• edge computing • federated learning • parallel processing	• sensor network • industrial IoT • machine-to-machine

Figure 1.1: Centralized and decentralized processing.

cooperate with the neighboring devices. Such decentralized processing may have an additional advantage in applications that deal with personal data with privacy, for example, as it does not exchange the raw data that may enable identification of an individual.

In the thesis, we focus on decentralized processing composed of divided processes of centralized processing for scalability in network size and the amounts of data. The decentralized processing includes partially distributed processing that supposes aggregation at the central processing unit and fully distributed processing that does not suppose the existence of the central unit. Fig. 1.1 summarizes conceptual diagrams, characteristics, and examples for the centralized, partially distributed, and fully distributed processing. If a problem that should be originally solved by centralized processing can be exactly divided into sub-problems, the solution by the decentralized processing according to the division becomes identical with that by the centralized one. On the other hand, if it cannot be done so and if an approximated division is employed, performance of the decentralized processing will be degraded. Therefore, it is required to develop a decentralized algorithm that can reduce the degradation as much as possible.

1.2 Thesis Overview and Contributions

1.2.1 Objective and Overview of Thesis

The objective of the thesis is to enhance performance of decentralized processing brought about by approximate divisions of the corresponding centralized processing. In the thesis, we tackle two estimation problems, namely, in-network adaptive least-mean-square (LMS) filter and Gaussian Process Regression (GPR), as the specific themes of the decentralized processing. Decentralization of the former problem is in demand to enhance scalability in network size, which is motivated by IoT environments where the number of devices is increasing recently. On the other hand, the latter problem is desired to be scalable in the amount of data because it requires high computational complexity. Some approximation methods of GPR reduce the complexity by decentralization of a part of the processes. Decentralization of both of the problems includes in common some degradation coming from approximate divisions, so that there is some room for improving the performance of the algorithms. In the following, the decentralization of each problem and the contributions of the thesis are reviewed.

1.2.2 Part I: Fully Distributed LMS based on Message Propagation

Adaptive filters (Farhang-Boroujeny, 2013) aim to adaptively estimate filter coefficients $\boldsymbol{w}^{\mathrm{o}}$ so that a system can yield desirable outputs, according to measurements $d^{(i)}$ and inputs $\boldsymbol{u}^{(i)}$ that can be observed at each time i. One may notice that the settings are the same as online learning (Shalev-Shwartz, 2012) but the main interest in online learning is often to estimate outputs corresponding to test inputs instead of the filter coefficients. LMS is the simplest adaptive filter derived on the basis of a mean-square error of a linear measurement model $d^{(i)} = \boldsymbol{u}^{(i)\mathrm{H}}\boldsymbol{w}^{\mathrm{o}} + \boldsymbol{v}^{(i)}$, where $\boldsymbol{v}^{(i)}$ is additive noise, and has relatively low complexity because it does not include calculations of matrix inversion or expectation.

LMS itself does not necessarily require high computational cost but the decentralized estimation is in demand from the application side. In communication networks such as sensor networks and multi-agent systems (Olfati-Saber et al., 2007), LMS can be used for model parameter tracking or target localization (Sayed, 2014) on the basis of the measurements at the sensors or agents which are referred to as nodes. For the centralized LMS, a central processor (or fusion node) collects all the measurements and processes them. However, when the number of nodes in the network is large, most of the load concentrates around the fusion node. The heavily biased load may cause the failure of the fusion node and stop all the processing. Under such a situation, there have been proposed fully distributed LMS algorithms, where each node in the network iterates updates of its own estimate by adaptation and by communications with its neighbors, to make the process scalable and robust. The algorithms include approximated divisions of the centralized LMS in order to run only with locally available information at each node, so that the performance is generally degraded. Thus, it is important to reduce the degradation caused by the approximated divisions and improve the performance of the fully distributed LMS.

In the thesis, to reduce the degradation, we focus on ways for exchanging information in a fully distributed LMS, which are significantly related to approximate divisions of the problem considered in the centralized LMS. Specifically, we introduce a message propagation algorithm as the method for information exchange to achieve faster convergence or lower error than the conventional distributed LMS. The detailed derivations are described and the performance is evaluated via computer simulation shown in Chapter 3. We extend the proposed algorithm to the case where the parameter of interest is sparse. The derivation and evaluation are shown in Chapter 4.

1.2.3 Part II: New Representation of Scalable GPR

GPR (Rasmussen and Williams, 2006) mainly aims to estimate output z corresponding to input $\boldsymbol{x}$ by using more flexible models than the linear measurement model used in LMS, whose use is not limited to time-dependent data. It supposes a nonlinear measurement model $z = f(\boldsymbol{x}) + \epsilon$, where $f(\cdot)$ follows a Gaussian process and where ϵ is additive noise. The prediction can be obtained as the posterior distribution of Bayesian estimation. GPR enables flexible estimation but is known to require high computational cost when the number N of the input data is large because of the inversion of an $N \times N$ matrix. Note that the linear measurement model in LMS is actually included in GPR as a special case.

There exist a lot of approximation methods of GPR. One approach called aggregation method divides all data into relatively small sub-datasets, runs GPR for each local sub-dataset, and then aggregates the local predictions. This can be interpreted as partially distributed GPR. Also in the aggregation method, the performance is generally degraded compared with the centralized GPR because a part of covariance information of the data is lost due to the division of the processing. Thus, it is desirable to develop an algorithm that balances high availability of the covariance information and low complexity.

In the thesis, we enhance performance of the conventional aggregation method by adopting richer covariance information to reduce the degradation. We show that the conventional aggregation method can be generalized via the concept of dimensionality reduction called sketching (Liberty, 2013; Woodruff, 2014) and propose a novel distributed GPR. The proposed algorithm can use richer information at the cost of a slightly higher computational complexity than the conventional method or can significantly reduce the complexity by using less information. The detailed derivation is described and the performance is evaluated on synthetic and real datasets shown in Chapter 5.

1.3 Notations

In the rest of the thesis, we use the following notations. Boldface indicates vector or matrix. Let $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{N}$ be the sets of real numbers, complex numbers, and nonnegative integers, respectively. Superscripts $(\cdot)^{\mathrm{T}}$ and $(\cdot)^{\mathrm{H}}$ denote the transpose and the Hermitian transpose, respectively. $\mathrm{E}[\cdot]$, $\det[\cdot]$, $\mathrm{Tr}[\cdot]$, and $\mathrm{Re}[\cdot]$ stand for the expectation, the determinant, the trace operators, and the real part of a complex number, respectively. $\|\boldsymbol{w}\|_p$ for a vector $\boldsymbol{w} = [w_1, \ldots, w_M]^{\mathrm{T}}$ and $p \geq 1$ is ℓ_p-norm defined by $\left(\sum_{m=1}^{M} |w_m|^p\right)^{1/p}$. The abbreviated form $\|\cdot\|$ represents ℓ_2-norm. $\|\boldsymbol{w}\|_0$ for a vector $\boldsymbol{w}$ is the number of nonzero elements of $\boldsymbol{w}$ and is called ℓ_0-norm. $\mathrm{vec}(\cdot)$ and $\mathrm{vec}^{-1}(\cdot)$ denote the vectorization and the inverse vectorization operators. $\otimes$ denotes Kronecker product. $\mathrm{diag}[\cdot]$ is the diagonal matrix composed of the elements in the square brackets. The largest eigenvalue of matrix $\boldsymbol{A}$ is represented as $\lambda_{\max}(\boldsymbol{A})$. $\boldsymbol{0}$, $\boldsymbol{1}$, $\boldsymbol{O}$, and $\boldsymbol{I}$ stand for the vectors where all elements are composed of 0 or 1, zero matrix, and identity matrix, respectively. $\boldsymbol{1}_M$ and $\boldsymbol{I}_M$ are also the vector with size M where all elements are composed of 1 and identity matrix with size $M \times M$, respectively. $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ is Gaussian distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$. $\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}]$ means the covariance matrix of random vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. $\ker \boldsymbol{A} := \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}$ is the kernel (null space) of the matrix $\boldsymbol{A}$. $[\cdot]_{ab}$, $[\cdot]_a$, and $[\cdot]_{[a][b]}$ denote the (a, b)th element of the matrix, the ath row of the matrix or ath element of the vector, and the (a, b)th block of the block matrix, respectively. $\mathrm{sign}(\boldsymbol{w})$ for $\boldsymbol{w} = [w_1, \ldots, w_M]^{\mathrm{T}}$ is a vector of size M whose m-th element $[\mathrm{sign}(\boldsymbol{w})]_m$ is defined as

$$[\mathrm{sign}(\boldsymbol{w})]_m = \begin{cases} w_m/|w_m|, & \text{if } w_m \neq 0 \\ 0, & \text{if } w_m = 0. \end{cases} \quad (m = 1, \ldots, M)$$

1.4 Organization

This thesis is organized as follows. In Part I, we discuss fully distributed LMS based on message propagation method.

- Chapter 2: Network model and mathematical characterizations are presented. Diffusion LMS (D-LMS), which is one of the decentralized LMS algorithms, and consensus propagation (CP), which is employed in the thesis to improve the performance of D-LMS, are introduced.

- Chapter 3: We propose two algorithms: 1. A novel distributed LMS based on the exact version of CP, which runs by extracting a spanning tree from the original network. This work is based on Nakai and Hayashi (2017); Nakai-Kasai and Hayashi (2019). 2. D-LMS based on the suboptimal version of CP, which can run even in a network with cycles and which results in novel combination weights for D-LMS. This work is based on Nakai and Hayashi (2018); Nakai-Kasai and Hayashi (2019). The convergence analysis and computational complexity of the two proposed algorithms are discussed. The performance is evaluated via computer simulation.

- Chapter 4: We propose an algorithm that extends the second algorithm in Chapter 3 to a sparsity-aware variant of D-LMS. The convergence analysis and computational complexity of the proposed algorithm are discussed. The performance is evaluated via computer simulation. This work is based on Nakai-Kasai and Hayashi (2020, 2021).

In Part II, we address the new representation of scalable GPR.

- Chapter 5: The centralized processing of GPR and its approximation methods are introduced. We show a generalization of one of the approximation methods through a sketching technique which lowers a dimension of a large-size vector by multiplying it by a matrix. The generalization brings about a novel scalable approximation that uses some auxiliary points. There are five options for the choice of the points, which affect the predictive performance and the computational complexity. The proposed method has a theoretical guarantee known as consistency. The performance of the proposed algorithm with the five options is evaluated by using synthetic and real data. This work is based on Nakai-Kasai and Tanaka (2021).

In Part III Chapter 6, the summary of the thesis and future directions are discussed.

Part I

Fully Distributed LMS based on Message Propagation

2: Introduction of Distributed LMS

2.1 Introduction

2.1.1 Background and Related Work

On large-scale communication networks composed of a number of small nodes, such as sensor networks and M2M (machine-to-machine), the subject of distributed signal processing (Hara et al., 2006) is gathering much attention recently. For the centralized method, all measurements at the nodes are processed at once in a fusion node, but it suffers from problems that nodes around the fusion node tend to consume much energy for the relaying and also that the network is vulnerable to the failure of the fusion node. On the other hand, for the distributed processing, each node processes by itself its own measurements and information obtained from neighboring nodes and estimates a desired parameter. This method is scalable in network size and robust because there is less imbalance of computation and communication loads among the nodes.

For tracking unknown parameters of interest in networks, several interpretations of adaptive filters (Farhang-Boroujeny, 2013) as distributed signal processing have been proposed (Sayed and Lopes, 2007; Cattivelli et al., 2008; Wang and Dekorsy, 2019; Lopes and Sayed, 2008; Schizas et al., 2009; Cattivelli and Sayed, 2010; Sayed et al., 2013). The most famous and simplest adaptive filter, least-mean-square (LMS) algorithm (Mandic et al., 2015), has also been decentralized in various ways (Lopes and Sayed, 2007, 2008; Schizas et al., 2009; Cattivelli and Sayed, 2010; Sayed et al., 2013). In these algorithms, each node iteratively updates its estimate by the LMS algorithm using local measurements and by averaging the estimates of its neighboring nodes obtained via wireless communications, and finally all nodes in the network achieve a common estimate. The incremental LMS (Lopes and Sayed, 2007) is applicable to any connected networks but it suffers from slow convergence because it assumes that each node communicates only with a single specific node among its neighboring nodes. In order to achieve faster convergence, the Combine-then-Adapt (CTA) diffusion LMS (D-LMS) (Lopes and Sayed, 2008) is derived by modifying the updating rules of the incremental LMS so that it allows each node to utilize the estimates of all neighboring nodes. Moreover, the consensus-based LMS (Schizas et al., 2009) achieves faster convergence than the CTA D-LMS (Lopes and Sayed, 2008) while it requires uneven processing of the nodes. Furthermore, a modified D-LMS named Adapt-then-Combine (ATC) D-LMS, which is obtained by just interchanging the order of the updating rules of

CTA diffusion (Lopes and Sayed, 2008), has been proposed in Cattivelli and Sayed (2010). The ATC D-LMS can outperform not only the original version (Lopes and Sayed, 2008) but also the consensus-based LMS (Schizas et al., 2009) in terms of convergence rate. The estimate of D-LMS converges to that of the centralized solution if sufficiently small step-size parameters are employed (Zhao and Sayed, 2012; Sayed et al., 2013; Sayed, 2014). More recently, some improvements have been proposed such as a doubly-compressed D-LMS (Harrane et al., 2019) for communication reduction and an Adapt-Multi-Combine (AMC) D-LMS (Kong et al., 2017) for better convergence performance at the cost of frequent communications. D-LMS for more specific settings has also attracted interests in the context of multitask learning where each node in the network estimates its own target vector and the neighboring nodes have related targets (Nassif et al., 2017).

In addition to this, in many applications, unknown parameters could be sparse in some representation domain, meaning that most elements of the parameters are zero or very small. It is known that compressed sensing (Donoho, 2006) is effective for the sparse signal estimation, and methods based on the compressed sensing have been proposed for the distributed signal processing as well (Duarte et al., 2005; Hayakawa et al., 2018; Wee and Yamada, 2013; Hu and Zhan, 2016). Among such methods, an extension of D-LMS for sparse estimation has been proposed and called sparse diffusion LMS (SD-LMS) (Lorenzo and Sayed, 2013). The theoretical analysis in Lorenzo and Sayed (2013) guarantees convergence in the mean sense and describes mean-square behaviors of SD-LMS. It has been also revealed that SD-LMS shows better convergence performance than that of D-LMS in the case of sparse estimation under realistic conditions.

Both D-LMS and SD-LMS are derived by approximately dividing each corresponding centralized problem into sub-problems that can be solved in a fully distributed manner. The performance degradation by the approximate divisions is compensated by cooperation with neighboring nodes. In D-LMS and SD-LMS, conventional average consensus protocol (Olfati-Saber et al., 2007) has been employed for the cooperation, where all nodes in the network aim to obtain the average of all nodes' initial state values by exchanging the current estimates of the average with their neighbors. This protocol is known to require a lot of iterations to achieve consensus, especially in large networks. It is also known that, even in small-scale networks, the choice of combination weights used in the updates has a great impact on the convergence performance of D-LMS (Takahashi et al., 2010). Thus, several combination rules have been proposed in the literature, such as uniform rule (Blondel et al., 2005), maximum degree rule (Scherber and Papadopoulos, 2004), Metropolis rule (Xiao and Boyd, 2004), and relative degree rule (Cattivelli and Sayed, 2010). More sophisticated static rules are considered in Cattivelli and Sayed (2010); Takahashi et al. (2010); Zhao et al. (2012), which are derived by solving some optimization problems. In particular, a closed-form solution that minimizes the steady-state error has been derived in Tu and Sayed (2011) and Zhao et al. (2012), which is referred to as relative-variance rule. Since the relative-variance rule requires network statistics such as noise variance at all nodes, which are not locally available at each node in general, adaptive estimation methods of the parameters have been also proposed in Tu and Sayed (2011) and Zhao et al. (2012). On the other hand, there exists another kind of average consensus algorithms named consensus propagation (CP) (Moallemi and Roy, 2006). CP is based on the efficient exchange of information called belief propagation (Pearl, 1988). The algorithm can be interpreted in two ways. The one is exact CP that achieves exact average consensus with the minimum number of iterations required for message propagation through the network, i.e., the diameter of the tree, while it restricts the use to the specific structure of the network such as a tree. Another is loopy CP that can be applied to networks with some cycles but becomes suboptimal. Convergence of the loopy CP is controlled by some constants whose optimal values have not been known.

2.1.2 Contributions

In this thesis, we enhance the convergence performance of D-LMS (Cattivelli and Sayed, 2010) and SD-LMS (Lorenzo and Sayed, 2013) by improving ways for cooperation among nodes that compensate for performance degradation caused by approximated divisions of the centralized LMS. Specifically, we first propose a novel distributed LMS algorithm by applying the exact CP. Since networks for in-network signal processing do not necessarily have a tree structure in general, the proposed algorithm is applied to a spanning tree extracted from the original network using some spanning tree protocols (Herzen et al., 2011; Bui et al., 2004). Moreover, we apply the loopy CP to D-LMS and derive another novel distributed LMS algorithm which is directly applicable to the original networks having cycles. This thesis also extends it to an analytically tractable algorithm, which results in a novel combination rule of D-LMS. The constants involved in the loopy CP control the convergence property but they are known to be difficult to optimize in general. We thus select the constants by minimizing an upper bound of steady-state mean-squared-deviation (MSD) of D-LMS. We further extend the proposed combination rule to an adaptive version, which can be implemented in a fully distributed manner. Finally, we apply the above method to SD-LMS and optimize the constants in terms of MSD at the steady-state under practical assumptions. We derive two combination rules for SD-LMS according to different assumptions. Simulation results show the efficiency of the proposed methods. The proposed algorithms based on the exact CP and the loopy CP can achieve faster convergence than the conventional ATC D-LMS for large-scale network. The proposed combination rule inspired by the loopy CP achieves better convergence performance than the conventional combination rules. SD-LMS using the proposed method also achieves faster convergence than that using conventional combination rules.

2.2 Preliminaries

2.2.1 Network Model

Consider a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges, and where $|\mathcal{V}| = N$. The network $\mathcal{G}$ is assumed to be fixed and connected, i.e., there is at least one path for any pair of nodes in the network. The nodes that are directly connected by edges can communicate and share information with each other. Each node $k \in \mathcal{V}$ $(k = 1, \ldots, N)$ can perform only single-hop communications with its neighbors. $\mathcal{N}_k$ is the set of the neighbors of node k including node k itself, and $\mathcal{N}_k \setminus \{k\}$ is the set of the neighbors except for node k. Fig. 2.1 shows an example of a network with $N = 10$. Weighted adjacency matrix $\mathrm{adj}(\mathcal{G}) \in \mathbb{R}^{N \times N}$ of the network $\mathcal{G}$ is given by

$$[\mathrm{adj}(\mathcal{G})]_{lk} = \begin{cases} w_{lk}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ 0, & \text{if } l \notin \mathcal{N}_k \setminus \{k\}, \end{cases} \quad \text{for } k, l \in \mathcal{V}. \tag{2.2.1}$$

where $w_{lk} > 0$ is a weight of the edge between node l and k.

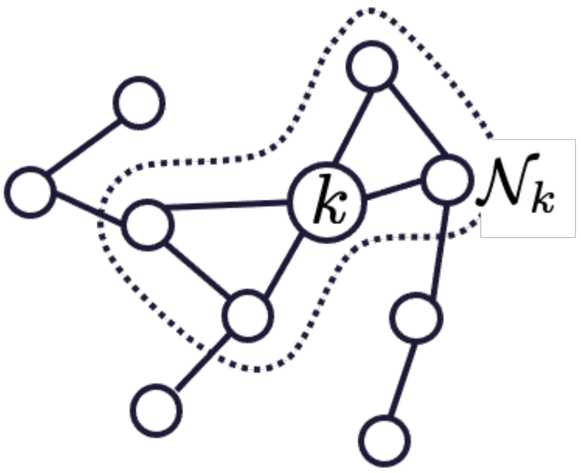


Figure 2.1: An example of a network with 10 nodes. $\mathcal{N}_k$ is the set of neighbors of node k (including node k itself).

2.2.2 Diffusion LMS

D-LMS Algorithm

On a network $\mathcal{G}$, consider a problem of tracking an unknown deterministic vector of interest $\boldsymbol{w}^{\mathrm{o}} \in \mathbb{C}^M$ by using measurements on the unknown vector and measurement vectors obtained at each node and each time that are related via linear regression model. For the centralized processing, the LMS filter (Mandic et al., 2015) is employed to track the unknown vector in a low computational cost. In this section, the fully distributed method, D-LMS (Lopes and Sayed, 2008; Cattivelli and Sayed, 2010; Sayed et al., 2013; Sayed, 2014), is introduced, where all nodes in the network track the unknown vector by themselves in a computationally efficient manner by exploiting LMS-like updates and by cooperating with the neighboring nodes. For simplicity, we assume that all communications between neighboring nodes are perfect, i.e., we do not consider any communication error, while the following discussion is applicable to networks with noisy links as considered in Zhao et al. (2012). Note that we also assume that nodes do not exchange the raw measurements and measurement vectors but only estimates processed in some way, which is convenient for private data for example, but the following discussion can be extended to the more general situation where each node exchanges the raw measurements and measurement vectors.

Consider random measurement process $\mathcal{D}_k^{(i)} \in \mathbb{C}$ at each node $k \in \mathcal{V}$ and time $i \in \mathbb{N}$ given by the following linear model:

$$\mathcal{D}_k^{(i)} = \mathcal{U}_k^{(i)\mathrm{H}} \boldsymbol{w}^{\mathrm{o}} + \mathcal{V}_k^{(i)}. \tag{2.2.2}$$

$\mathcal{U}_k^{(i)} \in \mathbb{C}^M$ is a random measurement vector process. $\mathcal{V}_k^{(i)} \in \mathbb{C}$ is a zero-mean additive complex noise process with variance of σ_k^2, which is pairwise independent of $\mathcal{V}_l^{(j)}$ for $l \in \mathcal{V} \setminus \{k\}$ and $j \in \mathbb{N} \setminus \{i\}$, and is pairwise independent of $\mathcal{U}_l^{(j)}$ for all $k, l \in \mathcal{V}, i, j \in \mathbb{N}$. The variance σ_k^2 is assumed to be unknown to each node. The measurement and measurement vector processes $\{\mathcal{D}_k^{(i)}, \mathcal{U}_k^{(i)}\}$ at each node k are assumed to be jointly wide-sense stationary and have zero-mean.

First, a global problem that should be considered at all nodes is shown and then it is approximated to derive a fully distributed algorithm. All nodes in the network aim to obtain a common estimate of $\boldsymbol{w}^{\mathrm{o}}$ basically by minimizing the following global cost function (Cattivelli and Sayed, 2010):

$$\mathcal{J}^{\mathrm{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N} \mathrm{E}\left[\left|\mathcal{D}_k^{(i)} - \mathcal{U}_k^{(i)\mathrm{H}} \boldsymbol{w}\right|^2\right]. \tag{2.2.3}$$

The function also does not depend on time index i because of the joint wide-sense stationarity. This includes all nodes' information of random processes, but, in the fully distributed situation, each node can directly obtain only a part of the information, i.e., its own and neighbors' information. In D-LMS, the following approximated local cost function has been considered at each node k,

$$\mathcal{J}_k^{\mathrm{loc}}(\boldsymbol{w}) = \mathrm{E}\left[\left|\mathcal{D}_k^{(i)} - \boldsymbol{\mathcal{U}}_k^{(i)\mathrm{H}}\boldsymbol{w}\right|^2\right] + \sum_{l\in\mathcal{N}_k\backslash\{k\}} b_{lk}\|\boldsymbol{w} - \boldsymbol{\phi}_l^{(i)}\|^2, \tag{2.2.4}$$

where $\boldsymbol{\phi}_l^{(i)} \in \mathbb{C}^M$ is the current estimate of $\boldsymbol{w}^{\mathrm{o}}$ at node l, and where $b_{lk} \geq 0$ is the weight to be determined later. The second term on the right-hand side of (2.2.4) means the penalty for the difference between node k's and the neighbors' estimates. The local function $\mathcal{J}_k^{\mathrm{loc}}(\boldsymbol{w})$ is an approximate division of the global one $\mathcal{J}^{\mathrm{glob}}(\boldsymbol{w})$ and includes compensation by the second term. If each node k took the first term on the right-hand side of (2.2.4) into account and if it knew moments related to $\mathcal{D}_k^{(i)}$ and $\boldsymbol{\mathcal{U}}_k^{(i)}$, it could obtain the optimal value $\hat{\boldsymbol{w}}$ from the derivative of the first term as

$$\hat{\boldsymbol{w}} = \boldsymbol{R}_{u_k}^{-1}\boldsymbol{r}_{d_k u_k},$$

where $\boldsymbol{r}_{d_k u_k} = \mathrm{E}[\mathcal{D}_k^{(i)}\boldsymbol{\mathcal{U}}_k^{(i)}]$ and $\boldsymbol{R}_{u_k} = \mathrm{E}[\boldsymbol{\mathcal{U}}_k^{(i)}\boldsymbol{\mathcal{U}}_k^{(i)\mathrm{H}}]$, which are independent of time index i because of the joint wide-sense stationarity of $\{\mathcal{D}_k^{(i)}, \boldsymbol{\mathcal{U}}_k^{(i)}\}$. It coincides with the unknown vector because the same formula can be obtained by multiplying $\boldsymbol{\mathcal{U}}_k^{(i)}$ and taking the expectation of both sides in (2.2.2). In other words, the unknown vector could be calculated at each node k if the moments $\{\boldsymbol{r}_{d_k u_k}, \boldsymbol{R}_{u_k}\}$ were known. However, in applications, each node cannot obtain the moments but realizations of the processes. The second term of the function (2.2.4) is motivated to improve convergence performance of the algorithm by cooperating with the neighboring nodes.

If we employ the steepest descent method (Farhang-Boroujeny, 2013) to minimize the function $\mathcal{J}_k^{\mathrm{loc}}$, node k's estimate $\boldsymbol{\phi}_k^{(i)}$ is updated by using the gradient $\nabla\mathcal{J}_k^{\mathrm{loc}}(\boldsymbol{w})$ of (2.2.4) as

$$\begin{aligned}\boldsymbol{\phi}_k^{(i)} &= \boldsymbol{\phi}_k^{(i-1)} - \frac{\mu_k}{2}\nabla\mathcal{J}_k^{\mathrm{loc}}(\boldsymbol{\phi}_k^{(i-1)}) \\ &= \boldsymbol{\phi}_k^{(i-1)} + \mu_k(\boldsymbol{r}_{d_k u_k} - \boldsymbol{R}_{u_k}\boldsymbol{\phi}_k^{(i-1)}) + \mu_k \sum_{l\in\mathcal{N}_k\backslash\{k\}} b_{lk}(\boldsymbol{\phi}_l^{(i)} - \boldsymbol{\phi}_k^{(i-1)}),\end{aligned} \tag{2.2.5}$$

where $\mu_k > 0$ is a step-size parameter, which controls a balance between the convergence rate and the steady-state error. Larger μ_k accelerates convergence to the steady-state but error at the steady-state becomes larger. Smaller μ_k promotes smaller steady-state error but the convergence becomes slower. One of the famous adaptive filters, LMS algorithm (Farhang-Boroujeny, 2013; Mandic et al., 2015), replaces the moments in (2.2.5) with instantaneous values. Let a realization of $\{\mathcal{D}_k^{(i)}, \boldsymbol{\mathcal{U}}_k^{(i)}\}$ be $\{d_k^{(i)}, \boldsymbol{u}_k^{(i)}\}$. The moments are replaced as $\boldsymbol{r}_{d_k u_k} \simeq d_k^{(i)}\boldsymbol{u}_k^{(i)}$ and

$\boldsymbol{R}_{u_k} \simeq \boldsymbol{u}_k^{(i)}\boldsymbol{u}_k^{(i)\mathrm{H}}$, respectively. The LMS-type update is given by

$$\boldsymbol{\phi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\boldsymbol{\phi}_k^{(i-1)}) + \mu_k \sum_{l\in\mathcal{N}_k\setminus\{k\}} b_{lk}(\boldsymbol{\phi}_l^{(i)} - \boldsymbol{\phi}_k^{(i-1)}). \tag{2.2.6}$$

The equation (2.2.6) can be divided into two steps. Each node k first updates an immediate estimate $\boldsymbol{\psi}_k^{(i)} \in \mathbb{C}^M$ by using its own measurement, and then updates the estimate $\boldsymbol{\phi}_k^{(i)}$ by using $\boldsymbol{\psi}_k^{(i)}$ and $\boldsymbol{\psi}_l^{(i)}$ received from its neighbors $l \in \mathcal{N}_k \setminus \{k\}$.

$$\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\boldsymbol{\phi}_k^{(i-1)}), \tag{2.2.7}$$

$$\boldsymbol{\phi}_k^{(i)} = \boldsymbol{\psi}_k^{(i)} + \mu_k \sum_{l\in\mathcal{N}_k\setminus\{k\}} b_{lk}(\boldsymbol{\psi}_l^{(i)} - \boldsymbol{\psi}_k^{(i)}), \tag{2.2.8}$$

where $\boldsymbol{\phi}_l^{(i)}$ and $\boldsymbol{\phi}_k^{(i-1)}$ in (2.2.6) are replaced with the immediate estimates $\boldsymbol{\psi}_l^{(i)}$ and $\boldsymbol{\psi}_k^{(i)}$, respectively. Here, let $\boldsymbol{A} = \{a_{lk}\} \in \mathbb{R}^{N\times N}$ be

$$a_{lk} = \begin{cases} \mu_k b_{lk}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ 1 - \mu_k \sum_{l\in\mathcal{N}_k\setminus\{k\}} b_{lk}, & \text{if } l = k, \\ 0, & \text{if } l \notin \mathcal{N}_k, \end{cases} \tag{2.2.9}$$

and then (2.2.8) can be rewritten as

$$\boldsymbol{\phi}_k^{(i)} = \sum_{l\in\mathcal{N}_k} a_{lk}\boldsymbol{\psi}_l^{(i)}, \tag{2.2.10}$$

where a_{lk} is the nonnegative combination weight defined in (2.2.9) and the matrix $\boldsymbol{A}$ satisfies

$$\boldsymbol{1}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{1}^{\mathrm{T}}. \tag{2.2.11}$$

The update (2.2.7) of the immediate estimate $\boldsymbol{\psi}_k^{(i)}$ is called adaptation step and that of $\boldsymbol{\phi}_k^{(i)}$ (2.2.10) is called combination step. From (2.2.4), (2.2.9), and (2.2.10), the second term in (2.2.4) corresponds to the combination step of D-LMS.

The combination weight a_{lk} does not have to be necessarily determined by the form in (2.2.9) but can be chosen freely as long as the constraint (2.2.11) is satisfied. A specific choice of the combination weights is called combination rule and has a great impact on the convergence performance of the algorithm (Takahashi et al., 2010). There are two types of the combination rules; a static rule that is independent of time index i and an adaptive rule that is dependent on i. Possible choices for the static one will be uniform rule (Blondel et al., 2005)

$$a_{lk}^{\mathrm{ave}} = \begin{cases} 1/|\mathcal{N}_k|, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise}, \end{cases} \tag{2.2.12}$$

Metropolis rule (Xiao and Boyd, 2004)

$$a_{lk}^{\mathrm{met}} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ 1 - \sum_{l\in\mathcal{N}_k\setminus\{k\}} a_{lk}^{\mathrm{met}}, & \text{if } l = k, \\ 0, & \text{otherwise}, \end{cases} \tag{2.2.13}$$

Algorithm 1 D-LMS (ATC version) (Cattivelli and Sayed, 2010)

```
1: Initialize $\phi_k^{(-1)} = 0,\ \forall k \in \mathcal{V}$
2: for each time $i \in \mathbb{N}$ and each node $k \in \mathcal{V}$ do
3:     $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \phi_k^{(i-1)})$   (adaptation step)
4:     $\phi_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk} \psi_l^{(i)}$   (combination step)
5: end for
```

relative degree rule (Cattivelli and Sayed, 2010)

$$a_{lk}^{\mathrm{rd}} = \begin{cases} \frac{|\mathcal{N}_l|}{\sum_{m \in \mathcal{N}_k} |\mathcal{N}_m|}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.2.14}$$

relative-variance rule (Tu and Sayed, 2011)

$$a_{lk}^{\mathrm{rv}} = \begin{cases} \frac{[\gamma_l^2]^{-1}}{\sum_{m \in \mathcal{N}_k} [\gamma_m^2]^{-1}}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.2.15}$$

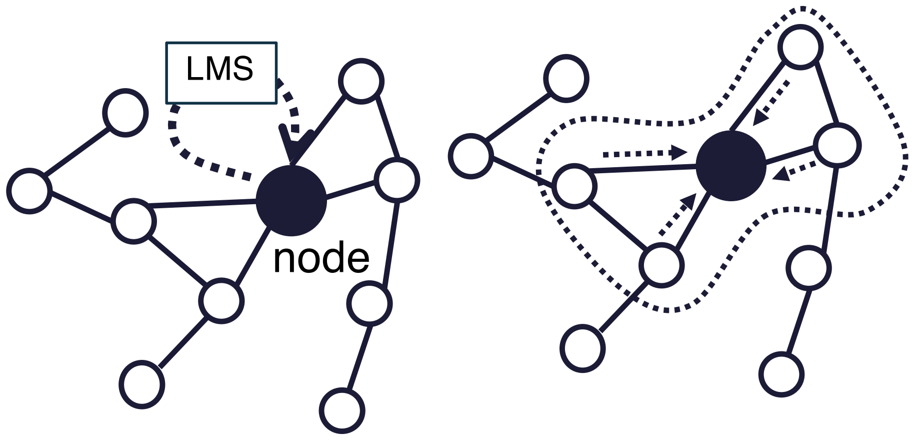
where $\gamma_l^2 = \mu_l^2 \sigma_l^2 \mathrm{Tr}[\boldsymbol{R}_{u_l}]$, and so on (Sayed, 2014). An example of the adaptive rule is adaptive relative-variance rule (Zhao et al., 2012)

$$a^{\mathrm{rv}(i)}_{lk} = \begin{cases} \frac{[(\gamma_{lk}^2)^{(i)}]^{-1}}{\sum_{m \in \mathcal{N}_k} [(\gamma_{mk}^2)^{(i)}]^{-1}}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.2.16}$$

where $(\gamma_{lk}^2)^{(i)} = (1 - \nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \left\| \psi_l^{(i)} - \phi_k^{(i-1)} \right\|^2$ and where $0 < \nu_k < 1$.

Summarizing the adaptation step (2.2.7) and the combination step (2.2.10), the updating rules of D-LMS are shown in *Algorithm* 1 and outlined in Fig. 2.2. Note that this formulations are called ATC version of D-LMS (Cattivelli and Sayed, 2010) and an alternative algorithm obtained by interchanging the order of the updates (2.2.7) and (2.2.10) is referred to as CTA version (Lopes and Sayed, 2008). In this part, we focus on ATC because it generally outperforms CTA (Sayed, 2014, Table 6), (Zhao and Sayed, 2012), though the following discussion can hold in both versions.

AMC diffusion LMS algorithm (Kong et al., 2017) has been also proposed and is constructed on the basis of D-LMS. The algorithm iterates the combination steps to collect more estimates and increase available information at each node.



(a) Adaptation step (b) Combination step

Figure 2.2: Two steps of D-LMS.

The update rules are described as

$$\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)} (d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \boldsymbol{\phi}_k^{(i-1)}), \tag{2.2.17}$$

$$\boldsymbol{\phi}_k^{(i)[1]} = \sum_{l \in \mathcal{N}_k} a_{lk}^{[1]} \boldsymbol{\psi}_l^{(i)}, \tag{2.2.18}$$

$$\boldsymbol{\phi}_k^{(i)[2]} = \sum_{l \in \mathcal{N}_k} a_{lk}^{[2]} \boldsymbol{\phi}_l^{(i)[1]}, \tag{2.2.19}$$

$$\vdots$$

$$\boldsymbol{\phi}_k^{(i)} = \boldsymbol{\phi}_k^{(i)[J']} = \sum_{l \in \mathcal{N}_k} a_{lk}^{[J']} \boldsymbol{\phi}_l^{(i)[J'-1]}, \tag{2.2.20}$$

where J' denotes the number of iterations of the combination steps, and where $\boldsymbol{\psi}_k^{(i)[j']} \in \mathbb{C}^M$ and $a_{lk}^{[j']} > 0$ are the immediate estimate and a combination weight at j'th iteration ($j' = 1, \ldots, J'$), respectively. The equations (2.2.18)–(2.2.20) are summarized as follows:

$$\boldsymbol{\phi}_k^{(i)} = \sum_{l_1 \in \mathcal{N}_k} a_{l_1 k}^{[J']} \sum_{l_2 \in \mathcal{N}_{l_1}} a_{l_2 l_1}^{[J'-1]} \cdots \sum_{l_{J'} \in \mathcal{N}_{l_{J'-1}}} a_{l_{J'} l_{J'-1}}^{[1]} \boldsymbol{\psi}_{l_{J'}}^{(i)}. \tag{2.2.21}$$

The adaptation step of AMC (2.2.17) is the same as that of D-LMS (2.2.7), but the

AMC diffusion LMS communicates J' times to collect immediate estimates of the nodes within J' hops.

Connection with Discrete-time Average Consensus Protocol

The combination step (2.2.10) of D-LMS is closely related to conventional average consensus protocol using a weighted average of neighbors' values in multi-agent networked systems (Olfati-Saber et al., 2007). In the protocol, all nodes in the network aim to obtain the average of all nodes' initial state values by exchanging the current estimates of the average with their neighbors. This problem is called average consensus problem. An iterative update equation of the conventional discrete-time protocol is given by

$$\hat{x}_k^{(i'+1)} = \bar{x}_k^{(i')} + \mu' \sum_{l \in \mathcal{N}_k \backslash \{k\}} w_{lk} (\hat{x}_l^{(i')} - \hat{x}_k^{(i')}), \tag{2.2.22}$$

where $\hat{x}_k^{(i')}$ is the estimate at node k and time i' and can be vector or scalar, and where μ' is a small positive number. In order to relate the combination step equation (2.2.10) with (2.2.22), we substitute $\boldsymbol{\phi}_k^{(i)}$ to $\hat{x}_k^{(i'+1)}$ and $\boldsymbol{\psi}_k^{(i)}$ to $\hat{x}_k^{(i')}$. Then, we have

$$\begin{aligned}
\boldsymbol{\phi}_k^{(i)} &= \boldsymbol{\psi}_k^{(i)} + \mu' \sum_{l \in \mathcal{N}_k \backslash \{k\}} w_{lk} (\boldsymbol{\psi}_l^{(i)} - \boldsymbol{\psi}_k^{(i)}) \\
&= (1 - \sum_{l \in \mathcal{N}_k \backslash \{k\}} a_{lk}) \boldsymbol{\psi}_k^{(i)} + \sum_{l \in \mathcal{N}_k \backslash \{k\}} a_{lk} \boldsymbol{\psi}_l^{(i)} \\
&= \sum_{l \in \mathcal{N}_k} a_{lk} \boldsymbol{\psi}_l^{(i)},
\end{aligned}$$

where we set $a_{lk} = \mu' w_{lk}$. Therefore, the combination step (2.2.10) can be regarded as an example of the conventional average consensus protocol.

Convergence Analysis

In this section, the convergence of D-LMS is discussed. We define convergence in the mean sense as in the following definition.

Definition 1. *Let* $\{\boldsymbol{\phi}\}_{i \in \mathbb{N}}$ *be a sequence of estimates of* $\boldsymbol{w}^{\mathrm{o}}$ *generated by an algorithm. We say that the algorithm converges in the mean sense when the estimation error* $\tilde{\boldsymbol{w}}^{(i)} = \boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}^{(i)}$ *satisfies* $\lim_{i \to \infty} \mathrm{E}[\tilde{\boldsymbol{w}}^{(i)}] = 0$.

The estimation error of D-LMS at node k and time i is given by

$$\tilde{\boldsymbol{w}}_k^{(i)} = \boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_k^{(i)}. \tag{2.2.23}$$

Moreover, for the mean-square error analysis, we evaluate behaviors of $\mathrm{E}[\|\tilde{\boldsymbol{w}}_k^{(i)}\|^2]$ at time i or at the steady-state. We define instantaneous network MSD, MSD at each node k at time i, network MSD at time i, the steady-state MSD at each node k, and the steady-state network MSD as

$$\mathrm{MSD}^{\mathrm{nw,in}} = \frac{1}{N}\sum_{k=1}^{N} \|\tilde{\boldsymbol{w}}_k^{(i)}\|^2, \tag{2.2.24}$$

$$\mathrm{MSD}_{k,i} = \mathrm{E}[\|\tilde{\boldsymbol{w}}_k^{(i)}\|^2], \tag{2.2.25}$$

$$\mathrm{MSD}_i^{\mathrm{nw}} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{MSD}_{k,i} = \mathrm{E}\left[\mathrm{MSD}^{\mathrm{nw,in}}\right], \tag{2.2.26}$$

$$\mathrm{MSD}_k = \lim_{i\to\infty} \mathrm{MSD}_{k,i}, \tag{2.2.27}$$

$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{MSD}_k = \lim_{i\to\infty} \mathrm{MSD}_i^{\mathrm{nw}} = \lim_{i\to\infty} \mathrm{E}\left[\mathrm{MSD}^{\mathrm{nw,in}}\right], \tag{2.2.28}$$

respectively.

We introduce the following two reasonable assumptions.

Assumption 1 (Cattivelli and Sayed (2010); Sayed (2014))**.** *The measurement vector process* $\{\mathcal{U}_k^{(i)}\}$ *is spatially independent and temporally white, i.e.,* $\mathrm{E}\left[\mathcal{U}_k^{(i)}\mathcal{U}_l^{(j)\mathrm{H}}\right] = \boldsymbol{R}_{u_k}\delta_{kl}\delta_{ij}$.

Assumption 2 (Lorenzo and Sayed (2013))**.** *The step-sizes* $\{\mu_k\}$ *are sufficiently small so that higher order terms of the step-sizes can be ignored.*

Assumption 1 does not match actual measurement situations. However, not only it simplifies the analysis of adaptive filters but also in general the analyzed performance under Assumption 2 reproduces the actual situations reasonably well (Sayed, 2011).

The convergence of D-LMS in the mean sense is guaranteed under Assumption 1.

Theorem 1 (Cattivelli and Sayed (2010, Theorem 1), Sayed (2014, Theorem 6.1))**.** *Consider the measurement model given by* (2.2.2) *and the combination weight satisfying* (2.2.11). *The convergence of D-LMS in the mean sense is guaranteed under Assumption 1 if the step-sizes satisfy*

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{u_k})} \quad (k \in \mathcal{V}).$$

Proof sketch: The error recursion can be expressed as

$$\mathrm{E}\left[\tilde{\boldsymbol{w}}^{(i)}\right] = \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{NM} - \mathcal{M}\mathcal{D})\mathrm{E}\left[\tilde{\boldsymbol{w}}^{(i-1)}\right], \tag{2.2.29}$$

where $\tilde{\boldsymbol{w}}^{(i)} = \left[(\tilde{\boldsymbol{w}}_1^{(i)})^{\mathrm{T}}, \ldots, (\tilde{\boldsymbol{w}}_N^{(i)})^{\mathrm{T}}\right]^{\mathrm{T}}$, $\mathcal{A} = \boldsymbol{A} \otimes \boldsymbol{I}_M$, $\mathcal{D} = \operatorname{diag}[\boldsymbol{R}_{u_1}, \ldots, \boldsymbol{R}_{u_N}]$, and $\mathcal{M} = (\operatorname{diag}[\mu_1, \ldots, \mu_N]) \otimes \boldsymbol{I}_M$. The recursion converges to zero if all the eigenvalue of $\boldsymbol{I}_{MN} - \mathcal{M}\mathcal{D}$ lie in a unit disc because $\mathcal{A}$ is composed of stochastic matrices, the magnitude of whose eigenvalues is less than or equal to 1. Thus, the theorem is derived by constraining the maximum eigenvalue of $\boldsymbol{I}_{MN} - \mathcal{M}\mathcal{D}$ to be less than 1. □

Moreover, the transient behavior of the mean-square error is represented in Cattivelli and Sayed (2010, Sect. 4) and Sayed (2014, Sect. 6) under Assumptions 1 and 2. Now let

$$\boldsymbol{F}' = (\boldsymbol{I}_{(MN)^2} - \boldsymbol{I}_{MN} \otimes (\mathcal{D}\mathcal{M}) - (\mathcal{D}^{\mathrm{T}}\mathcal{M}) \otimes \boldsymbol{I}_{MN})(\mathcal{A} \otimes \mathcal{A}), \tag{2.2.30}$$

and then the theoretical network MSD learning curve of D-LMS is given by

$$\mathrm{MSD}_i^{\mathrm{nw}} = \mathrm{MSD}_{i-1}^{\mathrm{nw}} + \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}\boldsymbol{F}'^{i}\boldsymbol{q}' - \frac{1}{N}\boldsymbol{w}^{\mathrm{T}}\left[\operatorname{vec}^{-1}\left(\boldsymbol{F}'^{i}\left[\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}'\right]\boldsymbol{q}'\right)\right]\boldsymbol{w}, \tag{2.2.31}$$

where $\boldsymbol{r}' = \operatorname{vec}\left(\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathcal{G}\mathcal{M}\mathcal{A}\right)$, $\mathcal{G} = \operatorname{diag}\left[\sigma_1^2\boldsymbol{R}_{u_1}, \ldots, \sigma_N^2\boldsymbol{R}_{u_N}\right]$, $\boldsymbol{w} = \boldsymbol{1}_N \otimes \boldsymbol{w}^{\mathrm{o}}$, and $\boldsymbol{q}' = \operatorname{vec}(\boldsymbol{I}_{MN})$. The performance at the steady-state is given by

$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{q}', \tag{2.2.32}$$

$$\mathrm{MSD}_k = \boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{p}', \tag{2.2.33}$$

where $\boldsymbol{p}' = \operatorname{vec}(C_k)$ and C_k is a block diagonal matrix with size $MN \times MN$ whose blocks are $M \times M$ matrices that are zero matrices for all blocks except for the identity matrix on the k-th diagonal block.

Proof sketch: Consider weighted ℓ_2-norm of the error recursion (2.2.29) with any weight Σ. If $\Sigma = \frac{1}{N}\boldsymbol{I}_{NM}$, the norm coincides with $\mathrm{MSD}_i^{\mathrm{nw}}$ and if $\Sigma = C_k$, it coincides with $\mathrm{MSD}_{k,i}$. By calculating $\mathrm{MSD}_i^{\mathrm{nw}}$ directly under Assumptions 1 and 2, $\boldsymbol{F}'$ is extracted as a coefficient matrix that does not depend on time indices. The learning curve of the network MSD can be derived by the recursion for $\Sigma = \frac{1}{N}\boldsymbol{I}_{NM}$. Taking the limit $i \to \infty$ yields the steady-state network MSD. □

SD-LMS Algorithm

SD-LMS is an extension of D-LMS which is applicable when the unknown vector $\boldsymbol{w}^{\mathrm{o}}$ is known to be sparse. All nodes in the network aim to obtain a common estimate of $\boldsymbol{w}^{\mathrm{o}}$ basically by minimizing the following global cost function (Lorenzo and Sayed, 2013):

$$\mathcal{J}^{\mathrm{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N}\mathrm{E}\left[\left|\mathcal{D}_k^{(i)} - \mathcal{U}_k^{(i)\mathrm{H}}\boldsymbol{w}\right|^2\right] + \lambda' f(\boldsymbol{w}), \tag{2.2.34}$$

where $\lambda' > 0$ is a regularization parameter and $f(\boldsymbol{w})$ is a real-valued sparsity-promoting regularization function such as ℓ_1-norm of $\boldsymbol{w}$ motivated by least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996), which is further discussed later. In the same way as in Sect. 2.2.2, the alternative problem considered to perform at each node k is minimizing the following approximated local cost function

$$\mathcal{J}_k^{\mathrm{loc}}(\boldsymbol{w}) = \mathrm{E}\left[\left|\mathcal{D}_k^{(i)} - \mathcal{U}_k^{(i)\mathrm{H}}\boldsymbol{w}\right|^2\right] + \lambda f(\boldsymbol{w}) + \sum_{l\in\mathcal{N}_k\setminus\{k\}} b'_{lk}\|\boldsymbol{w} - \boldsymbol{\phi}_l^{(i)}\|^2, \tag{2.2.35}$$

where $\lambda > 0$ is a regularization parameter and $b'_{lk} \geq 0$ is the weight to be determined later. The LMS-type update for the local cost function can be derived by using the gradient and replacing the moments with the instantaneous values, and also divided into two steps as before, as

$$\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k\boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\boldsymbol{\phi}_k^{(i-1)}) - \mu_k\lambda\partial f(\boldsymbol{\phi}_k^{(i-1)}), \tag{2.2.36}$$

$$\boldsymbol{\phi}_k^{(i)} = \boldsymbol{\psi}_k^{(i)} + \mu_k\sum_{l\in\mathcal{N}_k\setminus\{k\}} b'_{lk}(\boldsymbol{\psi}_l^{(i)} - \boldsymbol{\psi}_k^{(i)}), \tag{2.2.37}$$

where $\partial f(\cdot)$ is the sub-gradient of $f(\cdot)$. Letting $b'_{lk} = b_{lk}$ and introducing $\boldsymbol{A} = \{a_{lk}\}$ in (2.2.9) lead to

$$\boldsymbol{\phi}_k^{(i)} = \sum_{l\in\mathcal{N}_k} a_{lk}\boldsymbol{\psi}_l^{(i)}. \tag{2.2.38}$$

Summarizing the adaptation step (2.2.36) and the combination step (2.2.38), the updating rules of SD-LMS (Lorenzo and Sayed, 2013) are shown in *Algorithm* 2.

Remark 1. *SD-LMS is an extended algorithm of D-LMS (Cattivelli and Sayed, 2010) by adding the regularization term to the adaptation step.*

As the regularization function $f(\boldsymbol{w})$, a widely used form is ℓ_1-norm which enables us to avoid solving ℓ_0-regularization problems, which are known to be NP-hard, but in this thesis, we employ the following regularization function

$$f(\boldsymbol{w}) = \|\boldsymbol{W}_\epsilon\boldsymbol{w}\|_1 = \sum_{m=1}^{M}\frac{1}{\epsilon + |w_m|}|w_m|, \tag{2.2.39}$$

where w_m is the m-th element of $\boldsymbol{w}$, $\epsilon > 0$ is a parameter, and $\boldsymbol{W}_\epsilon = \mathrm{diag}\left[\frac{1}{\epsilon+|w_1|}, \ldots, \frac{1}{\epsilon+|w_M|}\right]$. This is regarded as reweighting of ℓ_1-norm (Candes et al., 2008) and, according to Lorenzo and Sayed (2013), achieves better performance than ℓ_1-norm, as anticipated by the fact that it is a better approximation of $\|\boldsymbol{w}\|_0$ than $\|\boldsymbol{w}\|_1$ as long as ϵ is sufficiently small. As an alternative to sub-gradient of (2.2.39), the following formula is exploited:

$$\partial f(\boldsymbol{w}) = \boldsymbol{W}_\epsilon\mathrm{sign}(\boldsymbol{w}), \tag{2.2.40}$$

where $\boldsymbol{W}_\epsilon$ in (2.2.39) are regarded as weights independent of $\boldsymbol{w}$.

The convergence of SD-LMS in the mean sense is also guaranteed in Lorenzo and Sayed (2013) under the same assumption as D-LMS.

Theorem 2 (Lorenzo and Sayed (2013, Theorem 1))**.** *Consider the measurement model given by* (2.2.2) *and the combination weight satisfying* (2.2.11)*. Assume the regularization function $f(\cdot)$ so that $\partial f(\cdot)$ is bounded. The convergence of SD-LMS in the mean sense is guaranteed under Assumption 1 if the step-sizes satisfy*

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{u_k})} \quad (k \in \mathcal{V}).$$

Proof sketch: The error recursion can be expressed as

$$\begin{aligned} \mathrm{E}\left[\tilde{\boldsymbol{w}}^{(i)}\right] &= \mathcal{B}\mathrm{E}\left[\tilde{\boldsymbol{w}}^{(i-1)}\right] + \lambda\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-1)})\right] \\ &= \mathcal{B}^i\mathrm{E}\left[\tilde{\boldsymbol{w}}^{(0)}\right] + \lambda\sum_{n=0}^{i-1}\mathcal{B}^n\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-n-1)})\right], \end{aligned} \tag{2.2.41}$$

where $\mathcal{B} = \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{NM} - \mathcal{M}\mathcal{D})$ and $\boldsymbol{\phi}^{(i-1)} = \left[(\boldsymbol{\phi}_1^{(i-1)})^{\mathrm{T}}, \ldots, (\boldsymbol{\phi}_N^{(i-1)})^{\mathrm{T}}\right]^{\mathrm{T}}$. The first term on the right-hand side can be handled in the same manner as in Theorem 1. For the convergence of the second term, comparison test is employed that bounds the term by another series where the absolute convergence can be shown. From the triangle inequality, each element of $\mathcal{B}^n\lambda\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-n-1)})\right]$ can be bounded by the norms of $\mathcal{B}$ and $\partial f(\boldsymbol{\phi}^{(\cdot)})$. The norm of $\mathcal{B}$ is bounded under the discussion in Theorem 1 and that of $\partial f(\boldsymbol{\phi}^{(\cdot)})$ is also bounded from the assumption of the theorem. Considering a series composed of these bounds, it becomes absolutely convergent, so that the series of $\mathcal{B}^n\lambda\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-n-1)})\right]$ in the second term of (2.2.41) is bounded by the absolutely convergent series. Therefore the second term is also absolutely convergent. □

The theoretical network MSD learning curve is expressed under Assumptions 1 and 2 as in D-LMS but has a more complicated form.

$$\begin{aligned} \mathrm{MSD}_i^{\mathrm{nw}} &= \mathrm{MSD}_{i-1}^{\mathrm{nw}} + \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}\boldsymbol{F}'^i\boldsymbol{q}' - \frac{1}{N}\boldsymbol{w}^{\mathrm{T}}\left[\mathrm{vec}^{-1}\left(\boldsymbol{F}'^i\left[\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}'\right]\boldsymbol{q}'\right)\right]\boldsymbol{w} \\ &\quad + g_{\boldsymbol{q}'/N}^{(i)} + \sum_{j=0}^{i-1}\left(g_{(\boldsymbol{F}')^{i-j}\boldsymbol{q}'/N}^{(j)} - g_{(\boldsymbol{F}')^{i-j-1}\boldsymbol{q}'/N}^{(j)}\right), \end{aligned} \tag{2.2.42}$$

where $g_{\boldsymbol{\sigma}}^{(j)} = \frac{\lambda}{N}\left(\lambda g_{1,\boldsymbol{\sigma}}^{(j)} - g_{2,\boldsymbol{\sigma}}^{(j)}\right)$, and where

$$g_{1,\boldsymbol{\sigma}}^{(j)} = \mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(j-1)})^{\mathrm{H}}\mathcal{M}\mathcal{A}\left(\mathrm{vec}^{-1}(\boldsymbol{\sigma})\right)\mathcal{A}^{\mathrm{T}}\mathcal{M}\partial f(\boldsymbol{\phi}^{(j-1)})\right], \tag{2.2.43}$$

Algorithm 2 SD-LMS (Lorenzo and Sayed, 2013)

```
1: Initialize $\phi_k^{(-1)} = 0,\ \forall k \in \mathcal{V}$
2: for each time $i \in \mathbb{N}$ and each node $k \in \mathcal{V}$ do
3:     $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \phi_k^{(i-1)}) - \mu_k \lambda \partial f(\phi_k^{(i-1)})$   (adaptation step)
4:     $\phi_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk} \psi_l^{(i)}$   (combination step)
5: end for
```

$$g_{2,\boldsymbol{\sigma}}^{(j)} = -2\mathrm{E}\Big[\partial f(\phi^{(j-1)})^{\mathrm{H}} \mathcal{M}\mathcal{A}\Big(\mathrm{vec}^{-1}(\boldsymbol{\sigma})\Big)\mathcal{B}\tilde{\boldsymbol{w}}^{(j-1)}\Big], \tag{2.2.44}$$

for any vector $\boldsymbol{\sigma}$. The steady-state network MSD and the steady-state MSD at each node k are shown in Lorenzo and Sayed (2013, Sect. 3) and given by

$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{q}' + \frac{1}{N}\lambda g_{1,\Sigma,\infty}\left(\lambda - \frac{g_{2,\Sigma,\infty}}{g_{1,\Sigma,\infty}}\right), \tag{2.2.45}$$

$$\mathrm{MSD}_k = \boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{p}' + \lambda g_{1,\Sigma_k,\infty}\left(\lambda - \frac{g_{2,\Sigma_k,\infty}}{g_{1,\Sigma_k,\infty}}\right), \tag{2.2.46}$$

respectively, where

$$g_{1,\Sigma,\infty} = \lim_{i\to\infty} g^{(i)}_{1,(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}\boldsymbol{q}'/N}, \tag{2.2.47}$$

$$g_{1,\Sigma_k,\infty} = \lim_{i\to\infty} g^{(i)}_{1,(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}C_k}. \tag{2.2.48}$$

$$g_{2,\Sigma,\infty} = \lim_{i\to\infty} g^{(i)}_{2,(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}\boldsymbol{q}'/N}, \tag{2.2.49}$$

$$g_{2,\Sigma_k,\infty} = \lim_{i\to\infty} g^{(i)}_{2,(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}C_k}, \tag{2.2.50}$$

The derivation is done in the same manner as D-LMS. Compared with (2.2.32) and (2.2.33), the last terms in (2.2.45) and (2.2.46) result from the regularization term and complicate the analysis of the steady-state performance. If the last terms are negative, SD-LMS is superior to the original D-LMS. In Lorenzo and Sayed (2013, Sect. 3-C), this is likely to be true in sparse estimation under the condition that $g_{2,\Sigma_k,\infty} > 0$ under Assumption 2 and that the regularization parameter satisfies $0 < \lambda < \frac{g_{2,\Sigma_k,\infty}}{g_{1,\Sigma_k,\infty}}$.

2.2.3 Consensus Propagation

CP (Moallemi and Roy, 2006) is an iterative algorithm to achieve the average consensus by regarding an undirected network as a bidirectional network and by using the idea of message passing algorithm (Pearl, 1988). Assume that each node k has an initial state value $x_k \in \mathbb{C}$. The goal of CP is the same as that of average consensus protocol (Olfati-Saber et al., 2007), that each node obtains the average of the initial state values $\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$. The formulations of CP can be categorized into two schemes: one is for a tree structured network and another is for the other structures that include some cycles. We name in the thesis the former exact CP and the latter loopy CP.

In the network with a tree structured network, exact CP can obtain the exact average. CP consists of two types of updates, namely, message update between neighboring nodes and state update at each node, to calculate the average using locally available information only. The updates of exact CP at the j-th iteration ($j \in \mathbb{N} \setminus \{0\}$) are given as follows:

$$K^{[j]}_{(u\to k)} = 1 + \sum_{m\in\mathcal{N}_u\setminus\{k,u\}} K^{[j-1]}_{(m\to u)}, \tag{2.2.51}$$

$$\theta^{[j]}_{(u\to k)} = \frac{x_u + \sum_{m\in\mathcal{N}_u\setminus\{k,u\}} K^{[j-1]}_{(m\to u)}\theta^{[j-1]}_{(m\to u)}}{1 + \sum_{m\in\mathcal{N}_u\setminus\{k,u\}} K^{[j-1]}_{(m\to u)}}, \tag{2.2.52}$$

where $K^{[0]}_{(u\to k)} = 0$, $\forall u, k \in \mathcal{V}$, and where $K^{[j]}_{(u\to k)}$ and $\theta^{[j]}_{(u\to k)}$ are the messages sent from node u to k. After iterating (2.2.51) and (2.2.52) between all neighboring nodes with the same number J as the diameter of the tree, the estimate of the average $\bar{x}$ at node k is given by

$$x^{[J]}_k = \frac{x_k + \sum_{u\in\mathcal{N}_k\setminus\{k\}} K^{[J]}_{(u\to k)}\theta^{[J]}_{(u\to k)}}{1 + \sum_{u\in\mathcal{N}_k\setminus\{k\}} K^{[J]}_{(u\to k)}}, \tag{2.2.53}$$

$x^{[J]}_k$ is exactly identical with $\bar{x}$ meaning that all nodes obtain $\bar{x}$. Figs. 2.3 and 2.4 show the updates at node k and its neighbors. Since the diameter is the minimum number of iterations required for the message propagation through the entire network, exact CP is a fast and efficient algorithm to achieve average consensus.

For networks involving some cycles, loopy CP is employed where the update of the message $K^{[j]}_{(u\to k)}$ is replaced with

$$K^{[j]}_{(u\to k)} = \frac{1 + \sum_{m\in\mathcal{N}_u\setminus\{k,u\}} K^{[j-1]}_{(m\to u)}}{1 + \frac{1}{\beta_k}(1 + \sum_{m\in\mathcal{N}_u\setminus\{k,u\}} K^{[j-1]}_{(m\to u)})}. \tag{2.2.54}$$

Here, β_k is a positive constant and it should be noted that β_k can be also made dependent on u as $\beta_{k,u}$ so that it depends not only on k but also on u. However, unlike the exact CP, what the sufficient number of iterations is has not been understood yet. By iterating (2.2.54) and (2.2.52) T times, the messages are shown to converge to unique constants as $T \to \infty$ and the resulting value of $x^{[T]}_k$ also becomes a constant, which is an estimate of the average. The convergence time for networks of any structure has not been fully understood.[1] It is known that β_k in (2.2.54)

[1]The convergence time for regular graphs has been analyzed in Moallemi and Roy (2006) but we now consider broader classes of networks.

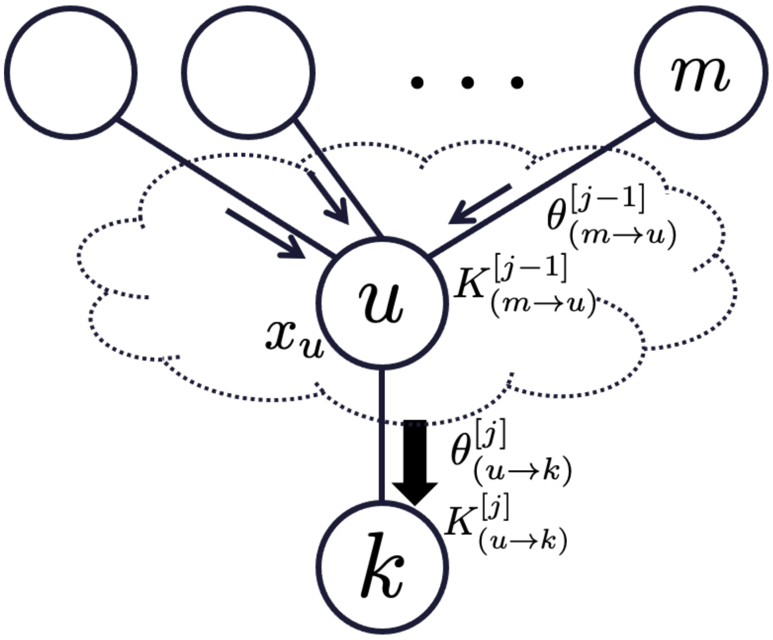


Figure 2.3: Update of message from node u to node k in the j-th iteration. The message is generated from the messages received in the $(j-1)$-th iteration from its neighbors except for node k, and its initial value x_u.

plays an important role for the convergence, but, to the best of our knowledge, the optimal value of β_k in terms of the convergence time has not been derived.

The message updates of CP can be represented by the non-backtracking operator (Coja-Oghlan et al., 2009; Decelle et al., 2011; Krzakala et al., 2013), which is a matrix defined in terms of connections between edges. We expect that the representation may enable the convergence analysis of loopy CP by using a spectrum of the operator (Bordenave et al., 2015).

2.3 Overview of Proposed Methods

In this part, we aim to reduce performance degradation of D-LMS and SD-LMS caused by approximate divisions of the cost functions and improve the part related to the compensation of the local cost functions in the algorithms, i.e., the combination steps which are closely related to the conventional average consensus protocol. Specifically, we propose three novel distributed LMS algorithms that employ CP as the alternative to the average consensus protocol to achieve better convergence performance.

The first algorithm is based on exact CP which converges to the exact average on the network with a tree structure. It requires extraction of a spanning tree

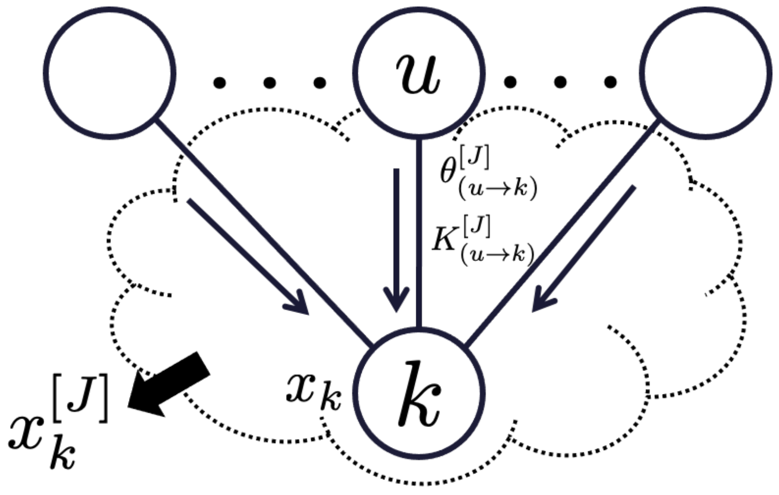


Figure 2.4: State update at node k. Node k calculates the estimate $x_k^{[J]}$ using the messages received in the J-th iteration from its all neighbors.

beforehand because the network of interest is not necessarily a tree. We then construct the algorithm by substituting the combination step of D-LMS with exact CP. Unlike the original D-LMS that iterates a pair of one adaptation step and one combination step, the proposed algorithm iterates a set of one adaptation step and J updates of exact CP (i.e., iterations of exact CP until achieving average consensus). Although in small networks the proposed algorithm may take more iterations than D-LMS, it expects faster convergence, especially in large networks, as long as the diameter of the tree is not so large. The algorithm is named CP-LMS.

The second algorithm is based on loopy CP. It does not require extraction of a spanning tree and can perform on networks of general structure. There can be two approaches to apply loopy CP to the distributed LMS. One approach is to replace the exact CP of CP-LMS with the loopy CP. However, constants involved in the approach and the required iteration number cannot be determined because appropriate values for them in loopy CP have not been known. Another approach considers a special case that employs only one iteration of loopy CP. This approach actually results in the proposal of a novel combination rule of D-LMS in this thesis and enables us to analyze convergence performance. The combination rule is named CP rule. Moreover, it enables optimization of the constant involved in loopy CP and extension to an adaptive version of the combination rule. The adaptive extension is named adaptive CP rule.

The last algorithm is an extension to SD-LMS. We apply the latter approach mentioned above to SD-LMS. In this case, there are two methods for the optimization of the constant involved in loopy CP. One method introduces a rough approximation, which results in the same combination rule as the above, adaptive CP rule.

Another method relaxes the approximation and yields a novel combination rule of SD-LMS. The rule is named adaptive CP with Optimization (CPO) rule.

3: D-LMS based on CP

3.1 D-LMS based on Exact CP

3.1.1 Proposed Algorithm

In the first approach CP-LMS, we firstly extract a spanning tree from the original network which possibly has some cycles using some centralized or distributed spanning tree protocol such as Herzen et al. (2011) and Bui et al. (2004). For example, we can use the distributed algorithm in Bui et al. (2004) to find a minimum diameter spanning tree of any graph $G = \{V, E\}$ with $O(|V|)$ time complexity and $O(|V||E|)$ message complexity. We then apply exact CP in the combination step of D-LMS on the extracted tree network. Every time each node obtains a linear measurement, the estimate $\psi_k^{(i)}$ at node k is updated by the adaptation step (2.2.7) as in the conventional method. The subsequent estimate $\phi_k^{(i)}$ is obtained as the consensus value achieved by exact CP on each element of the vector $\psi_k^{(i)}$, i.e., the average of all nodes' estimates at time i. After each node performs the same number of exaxt CP updates as the diameter J of the spanning tree, the average is substituted to the new estimate $\phi_k^{(i)}$, and then the algorithm proceeds to the next step.

The proposed updating rules are summarized in *Algorithm* 3. $K_{p,(k\to l)}^{(i)[j]}$ and $\theta_{p,(k\to l)}^{(i)[j]}$ are the p-th elements ($p = 1, \ldots, M$) of the messages transmitted from node k to node l at the j-th iteration of exact CP at time i. $(\phi_k^{(i)})_p$ becomes the p-th element of the average of all nodes' estimates at time i.

Algorithm 3 CP-LMS

```
 1: Extract a spanning tree
 2: Initialize $\phi_k^{(-1)} = 0, \ \forall k \in \mathcal{V}$
 3: for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each element $p = 1, \ldots, M$ do
 4:     $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)} (d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \phi_k^{(i-1)})$
 5:     $\theta_{p,(k\to l)}^{(i)[0]} = 0, K_{p,(k\to l)}^{(i)[0]} = 0$
 6:     for $j = 1$ to $J$ do
 7:         $K_{p,(k\to l)}^{(i)[j]} = 1 + \sum_{u\in\mathcal{N}_k\setminus\{l,k\}} K_{p,(u\to k)}^{(i)[j-1]}$
 8:         $\theta_{p,(k\to l)}^{(i)[j]} = \frac{(\psi_k^{(i)})_p + \sum_{u\in\mathcal{N}_k\setminus\{l,k\}} K_{p,(u\to k)}^{(i)[j-1]} \theta_{p,(u\to k)}^{(i)[j-1]}}{K_{p,(k\to l)}^{(i)[j]}}$
 9:     end for
10:     $(\phi_k^{(i)})_p = \frac{(\psi_k^{(i)})_p + \sum_{u\in\mathcal{N}_k\setminus\{k\}} K_{p,(u\to k)}^{(i)[J]} \theta_{p,(u\to k)}^{(i)[J]}}{1 + \sum_{u\in\mathcal{N}_k\setminus\{k\}} K_{p,(u\to k)}^{(i)[J]}}$
11: end for
```

3.1.2 Convergence Analysis

In this section, we analyze the convergence in the mean sense and the mean-square behavior of the proposed CP-LMS. In order to make the analysis tractable, we appropriately use the assumptions in Sect. 2.2.2.

It should be noted here that the proposed CP-LMS realizes the centralized fusion-based solution known as block LMS (Cattivelli and Sayed, 2011; Zhao and Sayed, 2012), where a fusion node collects measurements and measurement vectors of all other nodes, in a fully distributed manner if we introduce further assumption:

Assumption 3. *All nodes have the same step-size parameter, i.e., $\mu_k = \mu$ for all k,*

because all nodes can collect all nodes' estimates at each iteration after J updates of exact CP. To be more specific, under Assumption 3, the update equations of CP-LMS can be summarized as

$$\begin{cases} \boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\boldsymbol{\phi}_k^{(i-1)}), \\ \boldsymbol{\phi}_k^{(i)} = \frac{1}{N}\sum_{l=1}^{N} \boldsymbol{\psi}_l^{(i)}. \end{cases}$$

Since $\boldsymbol{\phi}_k^{(i)}$ does not depend on k, we replace $\boldsymbol{\phi}_k^{(i)}$ with a new notation $\boldsymbol{\psi}^{(i)}$, and then the whole update is simplified as

$$\begin{aligned} \boldsymbol{\psi}^{(i)} &= \frac{1}{N}\sum_{l=1}^{N}[\boldsymbol{\psi}^{(i-1)} + \mu \boldsymbol{u}_l^{(i)}(d_l^{(i)} - \boldsymbol{u}_l^{(i)\mathrm{H}}\boldsymbol{\psi}^{(i-1)})] \\ &= \boldsymbol{\psi}^{(i-1)} + \frac{\mu}{N}\sum_{l=1}^{N} \boldsymbol{u}_l^{(i)}(d_l^{(i)} - \boldsymbol{u}_l^{(i)\mathrm{H}}\boldsymbol{\psi}^{(i-1)}), \end{aligned} \tag{3.1.1}$$

which corresponds to the centralized block LMS algorithm. This means that the conventional performance analysis for the block LMS (Cattivelli and Sayed, 2011, Sect. 4-B) is directly applicable to the proposed CP-LMS. For example, the following theorem, which was originally proved for the block LMS, is valid for the CP-LMS as well.

Theorem 3. *Consider the measurement model given by* (2.2.2)*. The convergence of CP-LMS in the mean sense is guaranteed under Assumptions 1 and 3 if the step-size satisfies*

$$0 < \mu < \frac{2}{\lambda_{\max}(\boldsymbol{R})},$$

where $\boldsymbol{R} = \sum_{k=1}^{N} \boldsymbol{R}_{u_k}$.

The transient behavior of CP-LMS can be also represented by the same expression as the conventional block LMS (Cattivelli and Sayed, 2011, Sect. 4-C) under Assumptions 1 and 3. Now let $\boldsymbol{r}_{u_k} = \mathrm{vec}(\boldsymbol{R}_{u_k})$ and

$$\boldsymbol{F} = \boldsymbol{I}_{M^2} - \mu'(\boldsymbol{I}_M \otimes \boldsymbol{R}) - \mu'(\boldsymbol{R}^{\mathrm{T}} \otimes \boldsymbol{I}_M) + \mu'^2(\boldsymbol{R}^{\mathrm{T}} \otimes \boldsymbol{R}) + \mu'^2 \sum_{k=1}^{N} \boldsymbol{r}_{u_k} \boldsymbol{r}_{u_k}^{\mathrm{H}},$$

where $\mu' = \mu/N$. The theoretical network MSD learning curve of the CP-LMS algorithm is given by

$$\mathrm{MSD}_i^{\mathrm{nw}} = \mathrm{MSD}_{i-1}^{\mathrm{nw}} + \mu'^2 \sum_{k=1}^{N} \sigma_k^2 \boldsymbol{r}_{u_k}^{\mathrm{H}} \boldsymbol{F}^i \boldsymbol{q} - \boldsymbol{w}^{\mathrm{oH}} \left[\mathrm{vec}^{-1} \left(\boldsymbol{F}^i \left[\boldsymbol{I}_{M^2} - \boldsymbol{F} \right] \boldsymbol{q} \right) \right] \boldsymbol{w}^{\mathrm{o}}, \tag{3.1.2}$$

where $\boldsymbol{q} = \mathrm{vec}(\boldsymbol{I}_M)$. It is clear that $\mathrm{MSD}_k = \mathrm{MSD}^{\mathrm{nw}}$ holds in the case of CP-LMS because all nodes obtain the same estimate $\psi^{(i)}$ at each step i. Thus, the steady-state MSD is given by

$$\mathrm{MSD}_k = \mathrm{MSD}^{\mathrm{nw}} = \mu'^2 \sum_{l=1}^{N} \sigma_l^2 \boldsymbol{r}_{u_l}^{\mathrm{H}} (\boldsymbol{I}_{M^2} - \boldsymbol{F})^{-1} \boldsymbol{q}. \tag{3.1.3}$$

Note that it is one of the important merits of CP-LMS that the transient behavior can be described by the same expression as that of the centralized solution. Unlike the conventional D-LMS, the calculation of the theoretical transient behavior of the centralized solution requires much smaller computational complexity than that of the conventional D-LMS.

3.2 D-LMS based on Loopy CP

3.2.1 Proposed Algorithm – General Case

The extraction of a spanning tree in the algorithm in Sect. 3.1.1 can ensure the perfect consensus at each iteration of LMS. However, the use of a spanning tree may also degrade the convergence performance because some communication links available in the original network are not utilized at all in the algorithm working on the extracted spanning tree. In this section, we consider applying loopy CP to D-LMS without extracting any spanning tree.

Here, we describe a modified CP-LMS that can be directly applicable to networks with some cycles. Every time each node obtains a linear measurement, the estimate $\psi_k^{(i)}$ at node k is updated by LMS (2.2.7) as in the conventional method. Since it is difficult to know the required number of updates for the case of loopy CP, we set a fixed number of updates. After each node performs the updates of loopy

Algorithm 4 LCP-LMS

```
1:  Set $T > 0, \beta_k > 0, \forall k \in \mathcal{V}$
2:  Initialize $\phi_k^{(-1)} = 0,\ \beta_k > 0,\ \forall k \in \mathcal{V}, T > 0$
3:  for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each element $p = 1, \dots, M$ do
4:      $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)} (d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \phi_k^{(i-1)})$
5:      $\theta_{p,(k \to l)}^{(i)[0]} = 0,\ K_{p,(k \to l)}^{(i)[0]} = 0$
6:      for $t = 1$ to $T$ do
7:          $K_{p,(k \to l)}^{(i)[t]} = \frac{1 + \sum_{u \in \mathcal{N}_k \setminus \{l,k\}} K_{p,(u \to k)}^{(i)[t-1]}}{1 + \frac{1}{\beta_l}(1 + \sum_{u \in \mathcal{N}_k \setminus \{l,k\}} K_{p,(u \to k)}^{(i)[t-1]})}$
8:          $\theta_{p,(k \to l)}^{(i)[t]} = \frac{(\psi_k^{(i)})_p + \sum_{u \in \mathcal{N}_k \setminus \{l,k\}} K_{p,(u \to k)}^{(i)[t-1]} \theta_{p,(u \to k)}^{(i)[t-1]}}{1 + \sum_{u \in \mathcal{N}_k \setminus \{l,k\}} K_{p,(u \to k)}^{(i)[t-1]}}$
9:      end for
10:     $(\phi_k^{(i)})_p = \frac{(\psi_k^{(i)})_p + \sum_{u \in \mathcal{N}_k \setminus \{k\}} K_{p,(u \to k)}^{(i)[T]} \theta_{p,(u \to k)}^{(i)[T]}}{1 + \sum_{u \in \mathcal{N}_k \setminus \{k\}} K_{p,(u \to k)}^{(i)[T]}}$
11: end for
```

CP T times, the calculated estimate is assigned to the p-th element of the new estimate $\phi_k^{(i)}$, and then the algorithm proceeds to the next step. Note that $(\phi_k^{(i)})_p$ will not be exactly the same as the average of the p-th elements of $\psi_k^{(i)}$ in general for any T.

The proposed updating rules are summarized in *Algorithm* 4. We call this algorithm *Loopy CP-LMS (LCP-LMS)*. As mentioned in Sect. 2.2.3, the sufficient number of iterations for convergence has not been understood and the optimal value of constants β_k that minimizing convergence time has not been derived, so that the behavior of LCP-LMS cannot be also analyzed. It is necessary to adjust values of T and β_k ($\forall k \in \mathcal{V}$) depending on a structure of a network or desired performance.

3.2.2 Proposed Algorithm – Special Case

Derivation

LCP-LMS has difficulties in analyzing the performance and determining the constants β_k because the behavior of loopy CP has not been fully understood. However, we can analytically evaluate the convergence performance if we consider a special case that the number T of iterations of loopy CP updates is limited to one. This idea also brings about an advantage to track fluctuations of an unknown vector more quickly than CP-LMS or the general case of LCP-LMS which repeat communications while keeping information of measurements fixed. We describe the algorithm for the special case that employs only one update of loopy CP and then the method turns out to be D-LMS using a novel combination rule. We show how to determine the constants β_k in terms of the performance of D-LMS. This idea that T is limited to one does not enjoy benefits of CP but we focus on the analytical aspect and further extend the proposed combination rule to an adaptive one.

From the first updates of the messages in Algorithm 4, we have

$$K_{p,(u\to k)}^{(i)[1]} = \frac{\beta_k}{1+\beta_k}, \tag{3.2.1}$$

$$\theta_{p,(u\to k)}^{(i)[1]} = x_{p,u}^{(i)[0]} = (\psi_u^{(i)})_p. \tag{3.2.2}$$

We omit the subscript p in $K_{p,(u\to k)}^{(i)[1]}$ as $K_{(u\to k)}^{(i)[1]}$ because $K_{p,(u\to k)}^{(i)[1]}$ does not depend on p, and thus the element-wise update rule in Algorithm 4 can be rewritten as a vector-wise update as

$$K_{(u\to k)}^{(i)[1]} = \frac{\beta_k}{1+\beta_k}, \tag{3.2.3}$$

$$\theta_{(u\to k)}^{(i)[1]} = \psi_u^{(i)}. \tag{3.2.4}$$

Moreover, deriving $x_k^{(i)[1]}$, namely, $\phi_k^{(i)}$ by using (3.2.3) and (3.2.4) gives

$$\phi_k^{(i)} = \frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\psi_k^{(i)} + \frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\sum_{u\in\mathcal{N}_k\setminus\{k\}}\psi_u^{(i)} = \sum_{l\in\mathcal{N}_k} a_{lk}\psi_l^{(i)}. \tag{3.2.5}$$

This can be regarded as a novel combination rule summarized as

$$a_{lk} = \begin{cases} \frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ \frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}, & \text{if } k = l, \\ 0, & \text{otherwise,} \end{cases} \tag{3.2.6}$$

which satisfies $\mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}$.

Optimal Combination Weights

As mentioned in Sect. 2.2.3, how to select β_k has been an open issue (Moallemi and Roy, 2006). In this section, we choose β_k in terms of the steady-state network MSD of D-LMS (2.2.28). The steady-state network MSD itself includes an infinite sum and has a complicated form, but an upper bound of it can be expressed simply by a product of some parameters. We thus optimize β_k by minimizing the upper bound of the steady-state network MSD.

Under Assumptions 1 and 2 in Sect. 2.2.2, the steady-state network MSD can be rewritten by expanding $\left(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}'\right)^{-1}$ as

$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\sum_{j=0}^{\infty}\mathrm{Tr}\left[\mathcal{B}^j\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathcal{G}\mathcal{M}\mathcal{A}\left(\mathcal{B}^{\mathrm{H}}\right)^j\right], \tag{3.2.7}$$

where $\mathcal{B} = \mathcal{A}^{\mathrm{T}}(\boldsymbol{I} - \mathcal{M}\mathcal{D})$. The right-hand side can be shown to be bounded (Sayed, 2014, Sect. 8.2) such that

$$\mathrm{MSD}^{\mathrm{nw}} \le c\,\mathrm{Tr}\left[\mathcal{A}^{\mathrm{T}}\mathcal{M}\mathcal{G}\mathcal{M}\mathcal{A}\right], \tag{3.2.8}$$

where c is a positive constant. Recalling the definitions $\mathcal{A} = \boldsymbol{A} \otimes \boldsymbol{I}_M$, $\mathcal{M} = (\mathrm{diag}[\mu_1, \ldots, \mu_N]) \otimes \boldsymbol{I}_M$, and $\mathcal{G} = \mathrm{diag}[\sigma_1^2 \boldsymbol{R}_{u_1}, \ldots, \sigma_N^2 \boldsymbol{R}_{u_N}]$, the trace can be calculated as follows (Tu and Sayed, 2011; Zhao et al., 2012; Sayed, 2014):

$$\mathrm{Tr}\left[\mathcal{A}^{\mathrm{T}} \mathcal{M} \mathcal{G} \mathcal{M} \mathcal{A}\right] = \sum_{k=1}^{N} \sum_{l=1}^{N} \gamma_l^2 a_{lk}^2, \tag{3.2.9}$$

where $\gamma_l^2 = \mu_l^2 \sigma_l^2 \mathrm{Tr}[\boldsymbol{R}_{u_l}]$. For the proposed combination rule (3.2.6), the optimization problem to determine $\{\beta_k\}$ is written as

$$\begin{aligned} \{\beta_k^{\mathrm{opt}}\}_{k=1}^N = \; & \underset{\{\beta_k\}_{k=1}^N}{\arg\min} \sum_{k=1}^{N} \sum_{l=1}^{N} \gamma_l^2 a_{lk}^2, \\ & \text{s.t. } (3.2.6),\ \beta_k > 0\ (k = 1, \ldots, N), \end{aligned} \tag{3.2.10}$$

where the constraint $\beta_k > 0$ comes from the definition in loopy CP. By substituting (3.2.6) into the problem, this optimization problem can be separated into N independent subproblems as

$$\beta_k^{\mathrm{opt}} = \underset{\beta_k}{\arg\min}\ h(\beta_k), \tag{3.2.11}$$

where

$$h(\beta_k) = \gamma_k^2 \left(\frac{1 + \beta_k}{1 + |\mathcal{N}_k| \beta_k} \right)^2 + \sum_{l \in \mathcal{N}_k \setminus \{k\}} \gamma_l^2 \left(\frac{\beta_k}{1 + |\mathcal{N}_k| \beta_k} \right)^2.$$

The above function $h(\beta_k)$ is differentiable and we have

$$\frac{\partial h}{\partial \beta_k} = \frac{2}{(1 + |\mathcal{N}_k| \beta_k)^3} \left(\left(\sum_{l \in \mathcal{N}_k} \gamma_l^2 - |\mathcal{N}_k| \gamma_k^2 \right) \beta_k - (|\mathcal{N}_k| - 1) \gamma_k^2 \right). \tag{3.2.12}$$

In line with this equation, the shape of the cost function $h(\beta_k)$ largely depends on $\Gamma_k = \sum_{l \in \mathcal{N}_k} \gamma_l^2 - |\mathcal{N}_k| \gamma_k^2 \neq 0$ in the first term of the right-hand side of (3.2.12). When $\Gamma_k > 0$, the function $h(\beta_k)$ has a global minimum in $\beta_k > 0$ and the optimal value is obtained as

$$\beta_k^{\min} = \frac{(|\mathcal{N}_k| - 1) \gamma_k^2}{\Gamma_k}. \tag{3.2.13}$$

On the other hand, it becomes a monotonically decreasing function of $\beta_k > 0$ when $\Gamma_k < 0$. Thus, in summary, the optimum β_k is given by

$$\beta_k^{\mathrm{opt}} = \begin{cases} \beta_k^{\min}, & \text{if } \Gamma_k > 0, \\ +\infty, & \text{otherwise,} \end{cases} \tag{3.2.14}$$

and the corresponding combination weights are given by

$$a_{lk}^{\mathrm{cp}} = \begin{cases} \frac{\beta_k^{\mathrm{opt}}}{1 + |\mathcal{N}_k| \beta_k^{\mathrm{opt}}}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ \frac{1 + \beta_k^{\mathrm{opt}}}{1 + |\mathcal{N}_k| \beta_k^{\mathrm{opt}}}, & \text{if } k = l, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.15}$$

We name this combination rule as *CP rule*. Note that if $\Gamma_k < 0$, i.e., $\beta_k^{\mathrm{opt}} = +\infty$, the weight a_{lk}^{cp} coincides with the uniform rule (2.2.12).

In the existing works (Tu and Sayed, 2011; Zhao et al., 2012; Sayed, 2014), $\{a_{lk}\}$ in (2.2.10) is directly derived by minimizing the upper bound (3.2.9) of $\mathrm{MSD}^{\mathrm{nw}}$ with respect to $\{a_{lk}\}$ as

$$\begin{aligned} \{a_{lk}^{\mathrm{opt}}\}_{k=1}^{N} = &\ \underset{\{a_{lk}\}_{k=1}^{N}}{\arg\min} \sum_{l=1}^{N} \gamma_l^2 a_{lk}^2, \\ \text{s.t.} &\ \sum_{l=1}^{N} a_{lk} = 1,\ a_{lk} = 0 \text{ if } l \notin \mathcal{N}_k. \end{aligned} \tag{3.2.16}$$

The solution results in the relative-variance rule (2.2.15) (Tu and Sayed, 2011) given by

$$a_{lk}^{\mathrm{rv}} = \begin{cases} \frac{[\gamma_l^2]^{-1}}{\sum_{m \in \mathcal{N}_k} [\gamma_m^2]^{-1}}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.17}$$

Adaptive Combination Weights

The CP rule (3.2.15) and the relative-variance rule (3.2.17) require the knowledge of γ_l^2, which depends on locally unavailable network statistics such as the correlation matrices $\boldsymbol{R}_{u_l}$ of the measurement vectors and the measurement noise variance σ_l^2. Thus, we employ adaptive estimations of γ_l^2 for D-LMS proposed in Tu and Sayed (2011); Zhao et al. (2012); Sayed (2014). The estimate $\phi_k^{(i)}$ should approach the unknown vector $\boldsymbol{w}^{\mathrm{o}}$ as the algorithm iterates (2.2.36) and (2.2.38), and reach the steady-state under Assumption 2. By using (2.2.36) and (2.2.2), the estimate $\psi_l^{(i)}$ can be rewritten as

$$\psi_l^{(i)} \approx \boldsymbol{w}^{\mathrm{o}} + \mu_l \mathcal{U}_l^{(i)} \mathcal{V}_l^{(i)}.$$

Taking the expectation leads to

$$\mathrm{E}\left[\left\|\psi_l^{(i)} - \boldsymbol{w}^{\mathrm{o}}\right\|^2\right] \approx \mu_l^2 \sigma_l^2 \mathrm{Tr}(\boldsymbol{R}_{u_l}).$$

The right-hand side is γ_l^2 itself. To estimate it adaptively, the instantaneous values are substituted into the expectation. Let $(\gamma_{lk}^2)^{(i)}$ be an estimate of γ_l^2 at node k and time i. Such an estimate $(\gamma_{lk}^2)^{(i)}$ is obtained by employing the following update:

$$(\gamma_{lk}^2)^{(i)} = (1 - \nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \left\|\psi_l^{(i)} - \phi_k^{(i-1)}\right\|^2, \tag{3.2.18}$$

where ν_k $(0 < \nu_k < 1)$ is the forgetting factor. By using this estimate, the adaptive version of (3.2.17) named the adaptive relative-variance rule (2.2.16) is proposed

Algorithm 5 D-LMS with adaptive relative-variance rule (Zhao et al., 2012)

```
1: Initialize $\phi_k^{(-1)} = 0, \ \forall k \in \mathcal{V}$
2: for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each neighbor $l \in \mathcal{N}_k$ do
3:     $\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \boldsymbol{\phi}_k^{(i-1)})$
4:     $(\gamma_{lk}^2)^{(i)} = (1-\nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \|\boldsymbol{\psi}_l^{(i)} - \boldsymbol{\phi}_k^{(i-1)}\|^2$
5:     $a_{lk}^{(i)} = \frac{[(\gamma_{lk}^2)^{(i)}]^{-1}}{\sum_{m \in \mathcal{N}_k} [(\gamma_{mk}^2)^{(i)}]^{-1}}$
6:     $\boldsymbol{\phi}_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk}^{(i)} \boldsymbol{\psi}_l^{(i)}$
7: end for
```

in Zhao et al. (2012) as

$$a_{lk}^{\mathrm{rv}(i)} = \begin{cases} \frac{[(\gamma_{lk}^2)^{(i)}]^{-1}}{\sum_{m \in \mathcal{N}_k} [(\gamma_{mk}^2)^{(i)}]^{-1}}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.19}$$

In the same manner, the adaptive version of the CP rule is given by

$$\beta_k^{(i)} = \begin{cases} \frac{(|\mathcal{N}_k|-1)(\gamma_{kk}^2)^{(i)}}{\Gamma_k^{(i)}}, & \text{if } \Gamma_k^{(i)} > 0, \\ +\infty, & \text{otherwise,} \end{cases} \tag{3.2.20}$$

$$a_{lk}^{\mathrm{cp}(i)} = \begin{cases} \frac{\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ \frac{1+\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}, & \text{if } k = l, \\ 0, & \text{otherwise,} \end{cases} \tag{3.2.21}$$

where $\Gamma_k^{(i)} = \sum_{l \in \mathcal{N}_k} (\gamma_{lk}^2)^{(i)} - |\mathcal{N}_k| (\gamma_{kk}^2)^{(i)}$.

The algorithms of D-LMS using the conventional adaptive combination rule in (3.2.19) and the proposed rule in (3.2.21) are summarized in *Algorithm* 5 and *Algorithm* 6, respectively. The computational complexity of Algorithm 5 and Algorithm 6 are almost the same because the complexity of the proposed rule (3.2.21) becomes comparable to that of the conventional rule (3.2.19) by substituting (3.2.20) into (3.2.21).

Convergence Analysis

In this section, we discuss the convergence of D-LMS using static (i.e., non-adaptive) CP rule (3.2.15) in the mean sense and the mean-square behavior. Note that the transient behavior for the adaptive combination rules (not only adaptive CP rule but also adaptive relative-variance rule) remains an open issue. So, the theoretical network MSD learning curve below will not be valid for the adaptive CP rule, while theoretical results at the steady-state are applicable to the adaptive CP rule as well. The conventional performance analysis framework for D-LMS in

Algorithm 6 D-LMS with proposed adaptive CP rule

```
 1: Initialize $\phi_k^{(-1)} = 0,\ \forall k \in \mathcal{V}$
 2: for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each neighbor $l \in \mathcal{N}_k$ do
 3:     $\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}} \boldsymbol{\phi}_k^{(i-1)})$
 4:     $(\gamma_{lk}^2)^{(i)} = (1-\nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \|\boldsymbol{\psi}_l^{(i)} - \boldsymbol{\phi}_k^{(i-1)}\|^2$
 5:     if $\sum_{l \in \mathcal{N}_k} (\gamma_{lk}^2)^{(i)} - (\gamma_{kk}^2)^{(i)} |\mathcal{N}_k| > 0$ then
 6:         $\beta_k^{(i)} = \frac{(|\mathcal{N}_k|-1)(\gamma_{kk}^2)^{(i)}}{\sum_{l \in \mathcal{N}_k} (\gamma_{lk}^2)^{(i)} - (\gamma_{kk}^2)^{(i)} |\mathcal{N}_k|}$
 7:         $a_{lk}^{(i)} = \frac{\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}\ (l \in \mathcal{N}_k \setminus \{k\}),\quad \frac{1+\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}\ (l = k)$
 8:     else
 9:         $a_{lk}^{(i)} = \frac{1}{|\mathcal{N}_k|}$
10:     end if
11:     $\boldsymbol{\phi}_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk}^{(i)} \boldsymbol{\psi}_l^{(i)}$
12: end for
```

Sect. 2.2.2 is applicable to the proposed D-LMS using CP rule because CP rule can be regarded as one of the combination rules.

Corollary 1. *Consider the measurement model given by* (2.2.2) *and the combination weight given by CP rule* (3.2.15)*. The convergence of D-LMS using the proposed CP rule in the mean sense is guaranteed under Assumption 1 if the step-sizes satisfy*

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{u_k})} \quad (k \in \mathcal{V}).$$

The theoretical network MSD learning curve of D-LMS using CP rule (3.2.15) can be represented by the same expression as (2.2.31) given by

$$\mathrm{MSD}_i^{\mathrm{nw}} = \mathrm{MSD}_{i-1}^{\mathrm{nw}} + \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}\boldsymbol{F}'^{i}\boldsymbol{q}' - \frac{1}{N}\boldsymbol{w}^{\mathrm{T}}\left[\mathrm{vec}^{-1}\left(\boldsymbol{F}'^{i}\left[\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}'\right]\boldsymbol{q}'\right)\right]\boldsymbol{w}. \tag{3.2.22}$$

The steady-state network MSD and the steady-state MSD at each node k are also represented by the same expressions as (2.2.32) and (2.2.33) given by

$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{q}', \tag{3.2.23}$$

$$\mathrm{MSD}_k = \boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{p}', \tag{3.2.24}$$

respectively.

3.3 Computational Complexity and Communication Cost

We compare the computational complexity and the communication cost of the proposed schemes with those of the conventional methods in this section. Table 3.1

shows the complexity and the cost of the proposed algorithms and other algorithms in each combination step at a node. The computational complexity is defined as the number of scalar multiplication (division) or addition (subtraction) par one-way synchronous communications to node k from its neighboring nodes (for example for CP-LMS, not added up J times). The communication cost is defined as the number of scalar values that node k has to receive par one synchronous communication (for CP-LMS, not added up J times). Note that, since the adaptation step is common for all algorithms including the proposed schemes, the complexities only related to the combination step are evaluated. We can find that the proposed CP-LMS and LCP-LMS require higher computational complexity and communication cost than other algorithms because they contain exchanges of two types of messages between nodes. However, for all the algorithms compared here the communication cost is of the same order. As the communication time typically dominates the execution time in applications such as wireless sensor networks, one may conclude that the required time is comparable.

Here, we also mention communication cost of centralized block LMS. In this case, a fusion node collects measurements and measurement vectors of all other nodes, so that it receives $M + 1$ scalar values from $N - 1$ nodes, namely, the communication cost is $(M + 1)(N - 1)$. This is higher than those of the distributed methods discussed here in terms of cost par a processing node and the difference will be remarkable, especially in large-scale networks.

Table 3.1: Computational complexity and communication cost for the proposed algorithms and other conventional algorithms par one synchronous communication at node k.

Algorithm	$\times,\div$	$+,-$	Communication cost
D-LMS w/ static rules	$M\lvert\mathcal{N}_k\rvert$	$M(\lvert\mathcal{N}_k\rvert - 1)$	$M(\lvert\mathcal{N}_k\rvert - 1)$
AMC diffusion	$M\lvert\mathcal{N}_k\rvert$	$M(\lvert\mathcal{N}_k\rvert - 1)$	$M(\lvert\mathcal{N}_k\rvert - 1)$
CP-LMS (proposed)	$M((\lvert\mathcal{N}_k\rvert - 1)^2$ $+\lvert\mathcal{N}_k\rvert/J)$	$2M(\lvert\mathcal{N}_k\rvert - 1)$ $\cdot(\lvert\mathcal{N}_k\rvert - 2 + 1/J)$	$2M(\lvert\mathcal{N}_k\rvert - 1)$
LCP-LMS (proposed)	$M(\lvert\mathcal{N}_k\rvert^2 - 1$ $+\lvert\mathcal{N}_k\rvert/T)$	$M(\lvert\mathcal{N}_k\rvert - 1)$ $\cdot(3\lvert\mathcal{N}_k\rvert - 5 + 2/T)$	$2M(\lvert\mathcal{N}_k\rvert - 1)$
D-LMS w/ adaptive relative-variance rule	$(2M + 5)\lvert\mathcal{N}_k\rvert$	$(3M+2)\lvert\mathcal{N}_k\rvert-M-1$	$M(\lvert\mathcal{N}_k\rvert - 1)$
D-LMS w/ adaptive CP rule (proposed)	$(2M + 4)\lvert\mathcal{N}_k\rvert + 3$	$(3M+3)\lvert\mathcal{N}_k\rvert-M+2$	$M(\lvert\mathcal{N}_k\rvert - 1)$

3.4 Simulation Results

3.4.1 Settings

In this section, we compare the learning curves of instantaneous network MSD, $\mathrm{MSD}^{\mathrm{nw,in}}$, of the proposed schemes with the theoretical results and those of the conventional schemes via computer simulations. All the simulation results were obtained by using MATLAB, and we implemented all algorithms by ourselves without any toolbox. We assumed that the measurement vectors $\{\boldsymbol{u}_k^{(i)}\}$ were zero-mean real Gaussian random vectors with size $M = 5$ and had time-correlated shift structures (Lopes and Sayed, 2007). The specific structure was given by

$$\boldsymbol{u}_k^{(i)} = [u_k(i),\ u_k(i-1),\ \cdots,\ u_k(i-M+1)]^{\mathrm{T}}, \tag{3.4.1}$$

where

$$u_k(i) = \alpha_k u_k(i-1) + \sqrt{\sigma_{u_k}^2(1-\alpha_k^2)}z_k(i). \quad (i \in \mathbb{N}) \tag{3.4.2}$$

$\alpha_k \in [0,1)$ and $\sigma_{u_k}^2 \in (0,1]$ were chosen from uniform distribution. $z_k(i)$ was a white Gaussian process with zero mean and unit variance, and independent of $z_l(j)$ for $l \in \mathcal{V}\backslash\{k\}$ and $j \neq i$. The initial value $u_k(0)$ was also chosen from i.i.d. Gaussian with zero mean and unit variance. In this case, the trace of the correlation matrix is $\mathrm{Tr}[\boldsymbol{R}_{u_k}] = M\sigma_{u_k}^2$. We used the common step-size parameters $\mu_k = \mu$, the common forgetting factors $\nu_k = \nu$, and the common initial values $(\gamma_{lk}^2)^{(0)} = (\gamma^2)^{(0)}$, for all k, l. The unknown vector was set to be $\boldsymbol{w}^{\mathrm{o}} = \frac{1}{\sqrt{M}}\mathbf{1}_M$. All simulation results were obtained by averaging over 100 independent trials. All networks considered in the following sections were connected and the topologies were fixed throughout the simulations.

3.4.2 Comparison with Theoretical Learning Curve

In Figs. 3.1 and 3.2, we compare the learning curves of the CP-LMS and D-LMS using the static CP rule (3.2.15) obtained by simulations with the theoretical curves using (3.1.2) and (3.2.22), respectively. We used only a small network with $N = 20$ in Fig. 3.3 due to the computational difficulty in obtaining the theoretical values. The simulation of CP-LMS and its theoretical calculation were performed on the spanning tree shown in Fig. 3.3 (b), and that of D-LMS using CP rule and its theoretical calculation were performed on the original network shown in Fig. 3.3 (a). The step-size parameters were $\mu = 0.08$ ($\mu' = 0.004$) in CP-LMS and $\mu = 0.01$ in D-LMS with CP rule, respectively. The measurement noise power was set to $\sigma_k^2 = 10^{-3}$ ($k = 1, \ldots, N$). Fig. 3.1 shows that the simulation result agrees well with the theoretical curve. On the other hand, in Fig. 3.2, the curve before the steady-state matches the theoretical one but we found a slight disagreement between the simulation and the theoretical results at the steady-state. This may be

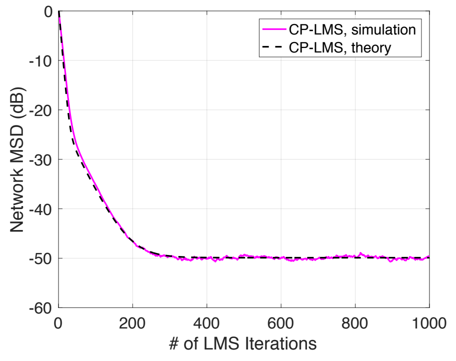


Figure 3.1: Network MSD learning curves of the proposed CP-LMS.

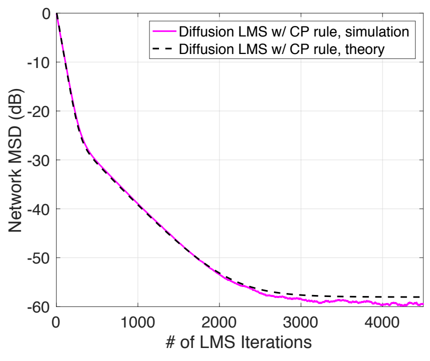
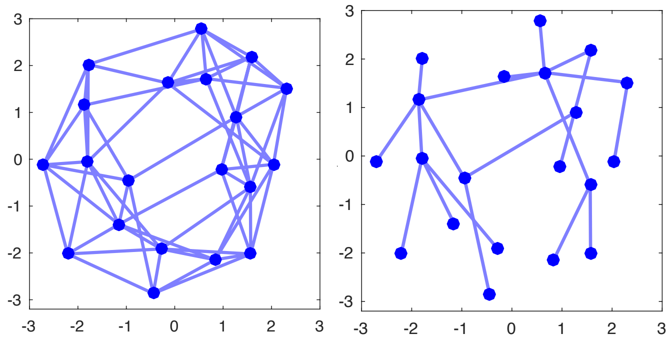


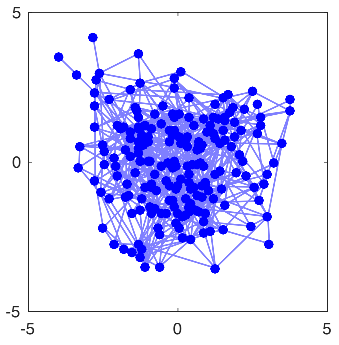
Figure 3.2: Network MSD learning curves of D-LMS with proposed CP rule.

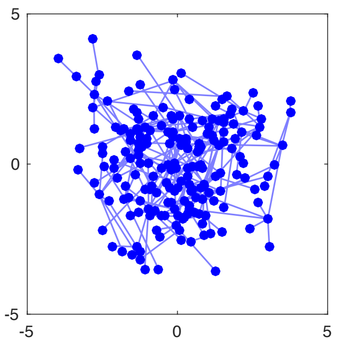


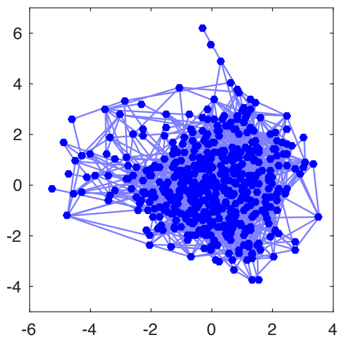
(a) Original network (b) Spanning tree

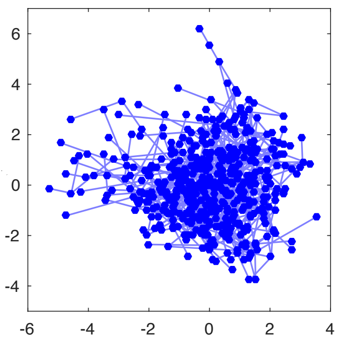
Figure 3.3: Network topologies with $N = 20$.

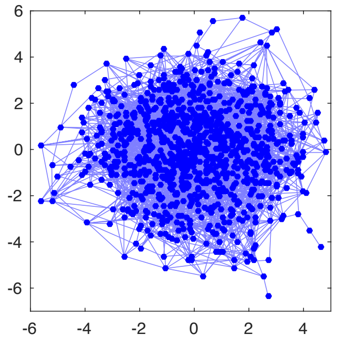
due to the gap between the model of measurement vectors (3.4.1) and Assumption 1 that imposes the temporal whiteness. Although this gap also affects Fig. 3.1, the disagreement becomes more pronounced in Fig. 3.2 because, even before the assumption is introduced, the theoretical result of D-LMS is more complicated than that of CP-LMS so that more information is lost in the former in approximation by introducing the assumption.


(a) Original network $N = 200$


(b) Spanning tree $N = 200$


(c) Original network $N = 500$


(d) Spanning tree $N = 500$


(e) Original network $N = 1000$

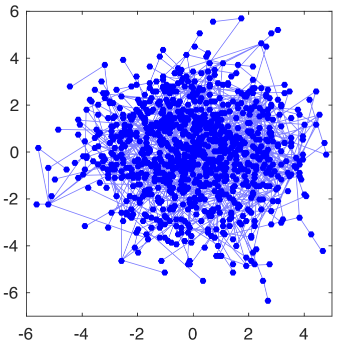

(f) Spanning tree $N = 1000$

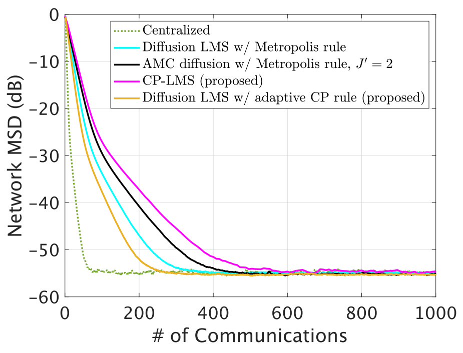
Figure 3.4: Network topologies.

3.4.3 Performance of Proposed CP-LMS and D-LMS w/ Adaptive CP rule

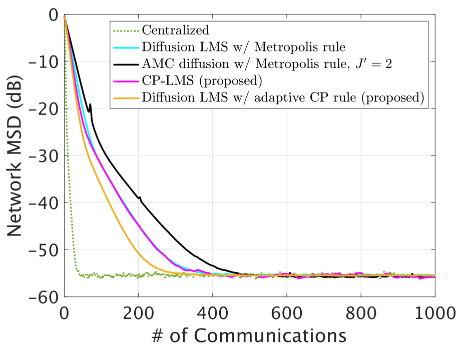
Here, we compare the performance of the proposed CP-LMS and D-LMS using the proposed adaptive CP rule (3.2.21) with that of D-LMS and the AMC diffusion

LMS using the conventional Metropolis rule (2.2.13). In order to compare the performance in networks of different sizes, we have generated Erdős-Rényi random networks with $N = 200,\ 500$, and 1000 as shown in Figs. 3.4 (a), (c), and (e), whose average degree was set to 6. Figs. 3.4 (b), (d), and (f) show the spanning trees extracted from the original networks so that the diameter of the tree becomes the smallest among all possible choices, i.e., the minimum diameter spanning trees. The diameters of the trees with $N = 200,\ 500$, and 1000 were $J = 8, 10$, and 10, respectively. Here, we adopted the minimum diameter spanning trees to CP-LMS and the original connected networks to D-LMS algorithms. The reason of the latter is that the original networks have more edges than the trees, which increases available information at the combination step in D-LMS. The number of iterations of the AMC diffusion LMS and the measurement noise power were set to $J' = 2$ and $\sigma_k^2 = 10^{-3}$ ($k = 1, \dots, N$), respectively. The performance of centralized block LMS in a star topology where measurements of all nodes were collected in a single synchronous communication was also evaluated.

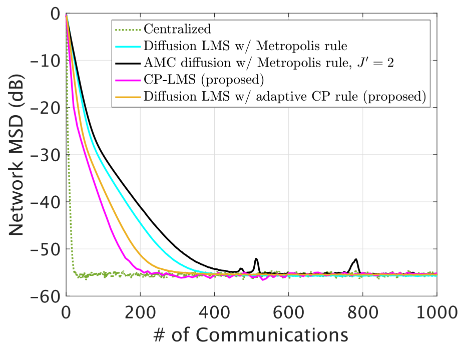
Fig. 3.5 shows the network MSD learning curves of the centralized block LMS at a star topology, the proposed methods, and the conventional methods for the networks with different sizes versus the number of communications. The number of communications is defined as the number of the combination steps, where the exchanges of measurements, messages, or immediate estimates are assumed to be synchronous among all nodes. We have controlled the step-size parameters of the algorithms so that the steady-state performance becomes comparable for all algorithms. In the figures, D-LMS using the proposed adaptive CP rule outperformed the conventional diffusion methods for all cases. As for CP-LMS, though the conventional methods achieved faster convergence in Fig. 3.5 (a), CP-LMS converged faster as the number of nodes increases and it outperformed the conventional methods and D-LMS using the proposed adaptive CP rule in Fig. 3.5 (c). These results are consistent with the expectation that the proposed CP-LMS would be useful, especially for large-scale networks. This is because the consensus protocol employed in the conventional LMS generally achieves slower convergence when the network size is large, while the convergence rate of CP depends only on the diameter of the graph. Note that the reason why the AMC diffusion LMS converged slower than D-LMS was that this chapter evaluated the network MSD with respect to the number of communications instead of the number of the adaptation steps. The centralized block LMS achieved the fastest convergence if it worked on a star topology but the convergence became slower in a case of relaying measurements in the networks shown in Fig. 3.4, which was understood from the reason that the situation can be regarded as CP-LMS.



(a) $N = 200$



(b) $N = 500$



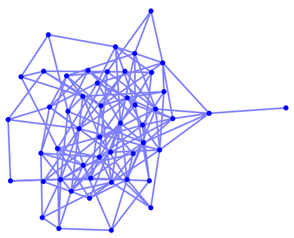
(c) $N = 1000$

Figure 3.5: Network MSD learning curves of proposed methods and conventional methods versus number of communications.

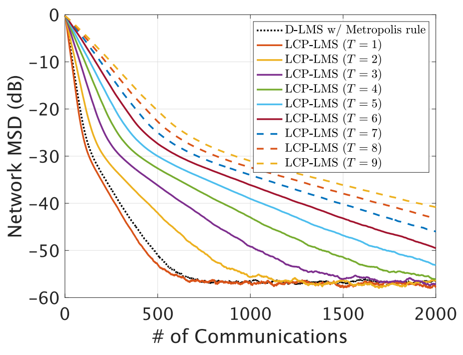
3.4.4 Performance of Proposed LCP-LMS

Next, we evaluated the performance of LCP-LMS where the iteration number T was varied from 1 to 9. For performance comparison in a small network, we used Erdős-Rényi random networks with $N = 50$ shown in Fig. 3.6 (a) and, for that in a large network, we used the network with $N = 1000$ in Fig. 3.4 (e). The constant and the measurement noise power were set to $\beta_k = 10^4$ and $\sigma_k^2 = 10^{-3}$ $(k = 1, \dots, N)$, respectively.

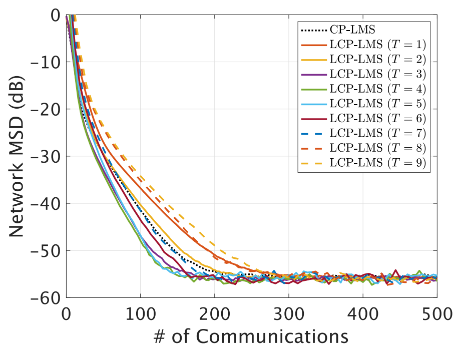
Figs. 3.6 (b) and (c) show the network MSD learning curves for the proposed LCP-LMS for the networks with different sizes versus the number of communications. We compare the performance of LCP-LMS in the small network with the conventional D-LMS in Fig. 3.6 (b). In this case, convergence of LCP-LMS with $T > 1$ was slower than D-LMS, so that the method could not work well in the small network. In Fig. 3.6 (c), we compare the performance of LCP-LMS in the large network with the proposed CP-LMS that achieved the best performance in Fig. 3.5 (c). The performance of LCP-LMS with $T = 4$ and $T = 5$ was the best among the



(a) Network $N = 50$



(b) Learning curves ($N = 50$)



(c) Learning curves ($N = 1000$)

Figure 3.6: Network MSD learning curves of LCP-LMS versus number of communications.

methods in this case. The figure suggests that LCP-LMS achieves faster convergence than CP-LMS and conventional diffusion schemes in large scale networks if we can set an appropriate value of T.

3.4.5 Proposed Rules vs Conventional Rules

Finally, we compare the performance of the proposed D-LMS using the static CP rule (3.2.15) and the adaptive CP rule (3.2.21) with that of conventional D-LMS using the static Metropolis rule (2.2.13), the static relative-variance rule (3.2.17), and the adaptive relative-variance rule (3.2.19) in a small network. We used the network with $N = 20$ shown in Fig. 3.3 (a). The measurement noise power σ_k^2 was varied among nodes in order to confirm the estimation ability of the proposed adaptive CP rule. We set σ_k^2 in proportion to the square of the distance of node k from the origin $(0, 0)$ of Fig. 3.3 (a). This setting can be alternatively understood as assuming the situation that the observation target is located at the origin, and the power of the target signal decays with the square of the distance while keeping the measurement noise power to be uniform for all observation nodes. The initial value and the forgetting factor of the adaptation were $(\gamma^2)^{(0)} = 4.5 \times 10^{-2}$ and $\nu = 0.07$ in

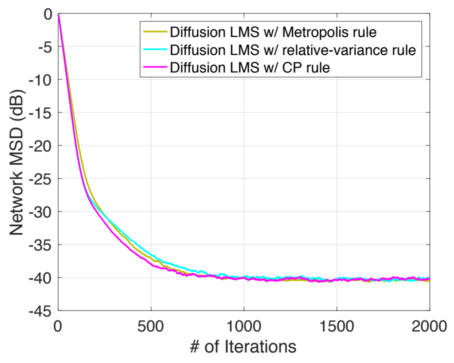


Figure 3.7: Network MSD learning curves of proposed CP rule and conventional static combination rules with $N = 20$.

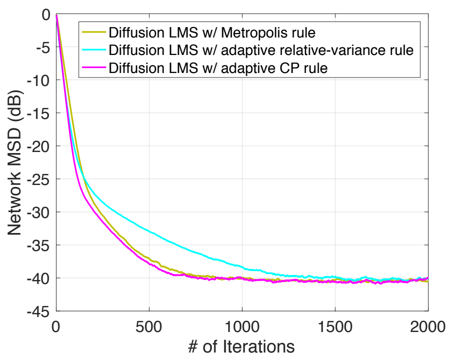


Figure 3.8: Network MSD learning curves of proposed adaptive CP rule and conventional combination rules with $N = 20$.

both Algorithms 5 and 6. We have controlled the step-size parameters of the algorithms so that the steady-state performance becomes comparable for all methods. Fig. 3.7 shows the learning curves of D-LMS using the static CP rule, Metropolis rule, and the static relative-variance rule in terms of the network MSD assuming that true γ_l^2s were known to each node. In the figure, we see that D-LMS using the CP rule converged faster than that using other rules. Fig. 3.8 shows the learning curves of D-LMS using the proposed adaptive CP rule, Metropolis rule, and the adaptive relative-variance rule in order to verify the influence of weight adaptation. The adaptive CP rule achieved comparable performance as in Fig. 3.7, while the performance of the adaptive relative-variance rule was significantly degraded in the settings here.

3.5 Conclusion

In this chapter, we have proposed novel D-LMS algorithms for in-network signal processing on the basis of the idea of the message passing algorithm of CP. By using exact CP on the spanning tree of the original network, CP-LMS can achieve the same solution as the centralized LMS in a fully distributed manner. LCP-LMS and its special case of CP rule are based on loopy CP, and we have optimized the constants involved in CP rule in terms of the steady-state MSD of D-LMS. Moreover, we have shown that their theoretical learning curves and steady-state MSDs can be obtained using existing frameworks. Also, we have extended the CP rule to an adaptive implementation named adaptive CP rule. Simulation results have shown that the proposed CP-LMS and LCP-LMS with an appropriate iteration number achieved better performance than the conventional D-LMS, especially in large-scale networks, and that D-LMS with the static and the adaptive CP rules achieved better performance than that with the conventional combination rules.

Future work includes extension of the proposed CP rule to more flexible weight control methods using asymmetric updates in loopy CP, i.e., using different constant at each node depending on the direction of the messages.

4: SD-LMS based on CP

4.1 SD-LMS based on Loopy CP

4.1.1 Derivation

We discuss the case that the unknown vector $\boldsymbol{w}^{\mathrm{o}}$ is known to be sparse. SD-LMS (Lorenzo and Sayed, 2013) is an extension of D-LMS for estimating sufficiently sparse unknown vectors. To improve the performance of SD-LMS, we apply loopy CP, instead of the conventional average consensus protocol, to SD-LMS. Especially in this chapter, to focus on analytical tractability of the algorithm, we deal with a special case of loopy CP, where only the first iteration ($T = 1$) is employed, in the same way as Sect. 3.2.2. This section shows that combining the special case of loopy CP with SD-LMS results in an SD-LMS that employs the same combination weight as that derived in Sect. 3.2.2, which includes the constant β_k. Optimization of β_k is discussed in the next section.

We apply the special case of loopy CP to the combination step of SD-LMS (2.2.38) and describe the update by using the messages of loopy CP, $K^{[1]}_{(u\to k)}$ and $\theta^{[1]}_{(u\to k)}$. The former is directly calculated as

$$K^{[1]}_{(u\to k)} = \frac{\beta_k}{1+\beta_k}. \tag{4.1.1}$$

Substituting the current estimate $\psi_k^{(i)}$ in (2.2.36) for the initial value x_k of loopy CP in (2.2.52) ($k = 1, 2, \ldots, N$), we have

$$\theta^{[1]}_{(u\to k)} = x_u = \psi_u^{(i)}. \tag{4.1.2}$$

Moreover, the estimate $\phi_k^{(i)}$ is obtained by the derivation of $x_k^{[1]}$ in (2.2.53) instead of (2.2.38) and can be calculated as

$$\begin{aligned}\phi_k^{(i)} &= \frac{x_k + \sum_{u\in\mathcal{N}_k\setminus\{k\}} K^{[1]}_{(u\to k)}\theta^{[1]}_{(u\to k)}}{1+\sum_{u\in\mathcal{N}_k\setminus\{k\}} K^{[1]}_{(u\to k)}}\\ &= \frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\psi_k^{(i)} + \frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\sum_{u\in\mathcal{N}_k\setminus\{k\}}\psi_u^{(i)}\\ &= \sum_{l\in\mathcal{N}_k} a_{lk}^{\mathrm{cp}}\psi_l^{(i)},\end{aligned}$$

where

$$a_{lk} = \begin{cases} \frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}, & \text{if } l \in \mathcal{N}_k\setminus\{k\},\\ \frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}, & \text{if } l = k,\\ 0, & \text{otherwise.}\end{cases} \tag{4.1.3}$$

This coincides with (3.2.6) for D-LMS. Therefore, the proposed method results in SD-LMS algorithm that uses the parameter β_k for the combination weights as in Sect. 3.2.2.

4.1.2 Optimization and Adaptive Combination Weights

Optimization Problem

In the following sections, we optimize β_k in (4.1.3) in terms of an upper bound of the steady-state network MSD of SD-LMS in the same way as Sect. 3.2.2. First, we introduce a specific formula of the steady-state network MSD of SD-LMS, $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}}$, that has been concretely calculated in Lorenzo and Sayed (2013) as

$$\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}} = \mathrm{MSD}^{\mathrm{nw}}_{\mathrm{dif}} + \frac{1}{N}\lambda g_{\lambda,\epsilon}(\boldsymbol{A}), \tag{4.1.4}$$

where $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{dif}}$ is the steady-state network MSD of D-LMS (2.2.32), where $g_{\lambda,\epsilon}(\boldsymbol{A})$ is a function depending on the parameters λ and ϵ, and where $\boldsymbol{A}$ is the matrix composed of the combination weights $\{a_{lk}\}$. The specific formula of $g_{\lambda,\epsilon}(\boldsymbol{A})$ has been shown in Lorenzo and Sayed (2013) as

$$g_{\lambda,\epsilon}(\mathbf{A}) = \lambda g_{1,\boldsymbol{\Sigma},\infty} - g_{2,\boldsymbol{\Sigma},\infty}, \tag{4.1.5}$$

where

$$g_{1,\boldsymbol{\Sigma},\infty} = \lim_{i\to\infty} \mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-1)})^{\mathrm{H}}\mathcal{M}\mathcal{A}\boldsymbol{\Sigma}\mathcal{A}^{\mathrm{T}}\mathcal{M}\partial f(\boldsymbol{\phi}^{(i-1)})\right], \tag{4.1.6}$$

$$g_{2,\boldsymbol{\Sigma},\infty} = -2\lim_{i\to\infty} \mathrm{E}\Big[\partial f(\boldsymbol{\phi}^{(i-1)})^{\mathrm{H}}\mathcal{M}\mathcal{A}\boldsymbol{\Sigma}\mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{NM} - \mathcal{M}\mathcal{D})\tilde{\boldsymbol{w}}^{(i-1)}\Big], \tag{4.1.7}$$

and where $\boldsymbol{\Sigma} = \mathrm{vec}^{-1}\left((\boldsymbol{I}_{(MN)^2} - \boldsymbol{F}')^{-1}\boldsymbol{q}'/N\right)$. Equations (4.1.6) and (4.1.7) are the same as (2.2.47) and (2.2.49), respectively, but they are reproduced here for convenience. Remark 1 in Sect. 2.2.2 demonstrates that the second term of (4.1.4) is due to the regularization term of SD-LMS.

Adaptive CP rule

The function $g_{\lambda,\epsilon}(\boldsymbol{A})$ has the complicated form so that it is difficult to use $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}}$ directly for the optimization of β_k. We thus derive an upper bound of $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}}$ by introducing some approximations.

In this section, we consider approximating $g_{\lambda,\epsilon}(\boldsymbol{A}) \simeq g_{\lambda,\epsilon}(\boldsymbol{I}_N)$, that is, the second term in (4.1.4) is assumed not to depend on $\boldsymbol{A}$, i.e., β_k. This is motivated by the fact that only the adaptation step includes the regularization term and that any information is not exchanged between nodes in the step. From the approximation, we only need to find an upper bound of $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{dif}}$ because the second term of (4.1.4) is independent of β_k. The optimization problem is identical with (3.2.10), so that we can derive the optimal parameter β_k^{opt} in the same manner, namely,

$$\beta_k^{\text{opt}} = \begin{cases} \frac{(|\mathcal{N}_k|-1)\gamma_k^2}{\Gamma_k}, & \text{if } \Gamma_k > 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.1.8}$$

The resulting combination weight using (4.1.8) has been named CP rule. As discussed in Sect. 3.2.2, in the case of $\beta_k^{\text{opt}} = \infty$, the resulting combination weight (4.1.3) coincides with the uniform rule (2.2.12).

Furthermore, we can also derive an adaptive solution to estimate $\gamma_l^2 = \mu_l^2\sigma_l^2\text{Tr}(\boldsymbol{R}_{u_l})$ and then update β_k in a similar way to that of D-LMS (shown in Sect. 3.2.2) to avoid the direct calculation of γ_l^2 that depends on locally unavailable network statistics such as noise variance σ_l^2 and the correlation matrix $\boldsymbol{R}_{u_l}$. The estimate $\phi_k^{(i)}$ approaches the unknown vector $\boldsymbol{w}^{\text{o}}$ as the algorithm iterates (2.2.36) and (2.2.38), and reaches steady-state under the assumption that the step-sizes are sufficiently small. By using (2.2.36) and (2.2.2), we can rewrite

$$\psi_l^{(i)} \approx \boldsymbol{w}^{\text{o}} + \mu_l \mathcal{U}_l^{(i)} \mathcal{V}_l^{(i)} - \mu_l\lambda\partial f(\boldsymbol{w}^{\text{o}}).$$

Taking the expectation leads to

$$\text{E}\left[\left\|\psi_l^{(i)} - \boldsymbol{w}^{\text{o}} + \mu_l\lambda\partial f(\boldsymbol{w}^{\text{o}})\right\|^2\right] \approx \mu_l^2\sigma_l^2\text{Tr}(\boldsymbol{R}_{u_l}).$$

We substitute the instantaneous values into the expectation and use it to estimate γ_l^2. Let $(\gamma_{lk}^2)^{(i)}$ be the estimate of γ_l^2 at node k and time i. We adaptively obtain the estimate by employing the following update:

$$(\gamma_{lk}^2)^{(i)} = (1-\nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k\left\|\psi_l^{(i)} - \phi_k^{(i-1)} + \mu_l\lambda\partial f(\phi_k^{(i-1)})\right\|^2, \tag{4.1.9}$$

where ν_k $(0 < \nu_k < 1)$ is the forgetting factor.

The subsequent estimate $\beta_k^{(i)}$ of β_k^{opt} and the combination weight $a_{lk}^{\text{cp},(i)}$ at time i are described as

$$\beta_k^{(i)} = \begin{cases} \frac{(|\mathcal{N}_k|-1)(\gamma_{kk}^2)^{(i)}}{\Gamma_k^{(i)}}, & \text{if } \Gamma_k^{(i)} > 0, \\ +\infty, & \text{otherwise,} \end{cases} \tag{4.1.10}$$

$$a_{lk}^{\text{cp}(i)} = \begin{cases} \frac{\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}, & \text{if } l \in \mathcal{N}_k \setminus \{k\}, \\ \frac{1+\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}}, & \text{if } k = l, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.11}$$

This combination weight has been named adaptive CP rule. SD-LMS using the adaptive CP rule is summarized in *Algorithm* 7.

Algorithm 7 SD-LMS with adaptive CP rule

```
1: Initialize $\phi_k^{(-1)} = 0, (\gamma_{lk}^2)^{(-1)}, \forall k \in \mathcal{V}, l \in \mathcal{N}_k$
2: for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each neighbor $l \in \mathcal{N}_k$ do
3:     $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\phi_k^{(i-1)}) - \mu_k \lambda \partial f(\phi_k^{(i-1)})$
4:     $(\gamma_{lk}^2)^{(i)} = (1-\nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \left\| \psi_l^{(i)} - \phi_k^{(i-1)} + \mu_l \lambda \partial f(\phi_k^{(i-1)}) \right\|^2$
5:     if $\sum_{l \in \mathcal{N}_k} (\gamma_{lk}^2)^{(i)} - (\gamma_{kk}^2)^{(i)} |\mathcal{N}_k| > 0$ then
6:         $\beta_k^{(i)} = \frac{(|\mathcal{N}_k|-1)(\gamma_{kk}^2)^{(i)}}{\sum_{l \in \mathcal{N}_k} (\gamma_{lk}^2)^{(i)} - (\gamma_{kk}^2)^{(i)} |\mathcal{N}_k|}$
7:     else
8:         $\beta_k^{(i)} = +\infty$ (large positive constant)
9:     end if
10:    $a_{lk}^{(i)} = \frac{\beta_k^{(i)}}{1+\beta_k^{(i)}|\mathcal{N}_k|}$ $(l \in \mathcal{N}_k \setminus \{k\})$, $\frac{1+\beta_k^{(i)}}{1+\beta_k^{(i)}|\mathcal{N}_k|}$ $(l = k)$
11:    $\phi_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk}^{(i)} \psi_l^{(i)}$
12: end for
```

Adaptive CPO rule

The optimization in Sect. 4.1.2 has some room for improvement because it has been derived by ignoring the influence of the sparsity-promoting regularization term in $\mathrm{MSD}_{\mathrm{spa}}^{\mathrm{nw}}$ (4.1.4) by approximating $g_{\lambda,\epsilon}(\boldsymbol{A}) \simeq g_{\lambda,\epsilon}(\boldsymbol{I}_N)$. In this section, we introduce another approximation to capture the contribution of $g_{\lambda,\epsilon}(\boldsymbol{A})$ on $\mathrm{MSD}_{\mathrm{spa}}^{\mathrm{nw}}$ and optimize the constant β_k.

For the first term of (4.1.4), we utilize the upper bound as in the discussion above. The following upper bound has been already derived in Cattivelli and Sayed (2010) as

$$\mathrm{MSD}_{\mathrm{dif}}^{\mathrm{nw}} \leq c \sum_{k=1}^{N} \sum_{l \in \mathcal{N}_k} \gamma_l^2 a_{lk}^2. \tag{4.1.12}$$

By substituting (4.1.3), the equation (4.1.12) can be rewritten as

$$\mathrm{MSD}_{\mathrm{dif}}^{\mathrm{nw}} \leq \sum_{k=1}^{N} \left(c\gamma_k^2 \left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k} \right)^2 + \sum_{l \in \mathcal{N}_k \setminus \{k\}} c\gamma_l^2 \left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k} \right)^2 \right). \tag{4.1.13}$$

The second term of (4.1.4) is difficult to directly express as a function of β_k because it takes the limit $i \to \infty$ and the expectation. Thus, we consider simplifying (4.1.6) and (4.1.7) by introducing three approximations. First, since the step-sizes are sufficiently small under Assumption 2, we ignore the term in (4.1.7) which is quadratic in $\mathcal{M}$ and approximate it as

$$g_{2,\boldsymbol{\Sigma},\infty} \simeq -2 \lim_{i \to \infty} \mathrm{E}\left[\partial f(\boldsymbol{\phi}^{(i-1)})^{\mathrm{T}} \mathcal{M} \mathcal{A} \boldsymbol{\Sigma} \mathcal{A}^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(i-1)} \right]. \tag{4.1.14}$$

Second, we remove the limit and the expectation in (4.1.6) and (4.1.7), and use the instantaneous values. Third, we approximate $\boldsymbol{\Sigma} = \boldsymbol{I}$. This approximation of $\boldsymbol{\Sigma}$ is reasonable because the instantaneous approximation of $g_{\lambda,\epsilon}(\boldsymbol{A})$ actually corresponds to the fourth term of (2.2.42) when the coefficient is ignored and because $\mathrm{vec}^{-1}(\boldsymbol{q}') = \mathrm{vec}^{-1}(\mathrm{vec}(\boldsymbol{I})) = \boldsymbol{I}$. Specifically, the approximated values of $g_{1,\boldsymbol{\Sigma},\infty}$ and $g_{2,\boldsymbol{\Sigma},\infty}$ are given by

$$g_{1,i} \simeq \partial f(\boldsymbol{\phi}^{(i-1)})^{\mathrm{H}} \mathcal{M}\mathcal{A}\mathcal{A}^{\mathrm{T}}\mathcal{M}\partial f(\boldsymbol{\phi}^{(i-1)}), \tag{4.1.15}$$

$$g_{2,i} \simeq -2\partial f(\boldsymbol{\phi}^{(i-1)})^{\mathrm{H}} \mathcal{M}\mathcal{A}\mathcal{A}^{\mathrm{T}}\tilde{\boldsymbol{w}}^{(i-1)}, \tag{4.1.16}$$

respectively. By using these terms, we have the approximated $g_{\lambda,\epsilon}(\boldsymbol{A})$ as

$$\begin{aligned} g_{\lambda,\epsilon}(\boldsymbol{A}) &\approx \lambda g_{1,i} - g_{2,i} \\ &= \sum_{k=1}^{N} \sum_{j\in\mathcal{N}_k} \sum_{l\in\mathcal{N}_k} \Big[\lambda\mu_j\mu_l a_{jk} a_{lk} \partial f(\boldsymbol{\phi}_j^{(i-1)})^{\mathrm{H}} \partial f(\boldsymbol{\phi}_l^{(i-1)}) \\ &\quad + 2\mu_j a_{jk} a_{lk} \partial f(\boldsymbol{\phi}_j^{(i-1)})^{\mathrm{H}} \left(\boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_l^{(i-1)}\right)\Big]. \end{aligned} \tag{4.1.17}$$

It seems hard to optimize N^2 combination weights $\{a_{lk}\}$ directly from (4.1.17) but it can be captured by the optimization of N coefficients $\{\beta_k\}$ in our framework assuming (4.1.3). Substituting (4.1.3) into (4.1.17) leads to

$$g_{\lambda,\epsilon}(\boldsymbol{A}) \approx \sum_{k=1}^{N} \left[\tilde{\varrho}_k^{(i)} \left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)\left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right) + \tilde{\varepsilon}_k^{(i)} \left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2 + \tilde{\varsigma}_k^{(i)} \left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2\right], \tag{4.1.18}$$

where

$$\begin{aligned} \tilde{\varrho}_k^{(i)} &= \sum_{l\in\mathcal{N}_k\backslash\{k\}} \mathrm{Re}\left[\mu_k \partial f(\boldsymbol{\phi}_k^{(i-1)})^{\mathrm{H}} \left\{\lambda\mu_l \partial f(\boldsymbol{\phi}_l^{(i-1)}) + 2\left(\boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_l^{(i-1)}\right)\right\}\right] \\ &\quad + \sum_{l\in\mathcal{N}_k\backslash\{k\}} \mathrm{Re}\left[\mu_l \partial f(\boldsymbol{\phi}_l^{(i-1)})^{\mathrm{H}} \left\{\lambda\mu_k \partial f(\boldsymbol{\phi}_k^{(i-1)}) + 2\left(\boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_k^{(i-1)}\right)\right\}\right], \\ \tilde{\varepsilon}_k^{(i)} &= \mathrm{Re}\left[\mu_k \partial f(\boldsymbol{\phi}_k^{(i-1)})^{\mathrm{H}} \left\{\lambda\mu_k \partial f(\boldsymbol{\phi}_k^{(i-1)}) + 2\left(\boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_k^{(i-1)}\right)\right\}\right], \\ \tilde{\varsigma}_k^{(i)} &= \sum_{l\in\mathcal{N}_k\backslash\{k\}} \sum_{j\in\mathcal{N}_k\backslash\{k\}} \mathrm{Re}\left[\mu_j \partial f(\boldsymbol{\phi}_j^{(i-1)})^{\mathrm{H}} \left\{\lambda\mu_l \partial f(\boldsymbol{\phi}_l^{(i-1)}) + 2\left(\boldsymbol{w}^{\mathrm{o}} - \boldsymbol{\phi}_l^{(i-1)}\right)\right\}\right]. \end{aligned}$$

In the above equations, we extract real parts of terms to guarantee that $\mathrm{MSD}_{\mathrm{spa}}^{\mathrm{nw}}$ and resulting weights are real numbers.

By incorporating (4.1.13) and (4.1.18), we can obtain an approximated upper bound of $\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}}$ as

$$\mathrm{MSD}^{\mathrm{nw}}_{\mathrm{spa}} \lessapprox c\sum_{k=1}^{N}\sum_{l\in\mathcal{N}_k}\gamma_l^2 a_{lk}^2 + \frac{\lambda}{N}g_{\lambda,\epsilon}(\boldsymbol{A}) \tag{4.1.19}$$

$$= \sum_{k=1}^{N}\Big[\frac{\lambda}{N}\tilde{\varrho}_k^{(i)}\left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)\left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)$$

$$+\left(c\gamma_k^2+\frac{\lambda}{N}\tilde{\varepsilon}_k^{(i)}\right)\left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2+\left(\sum_{l\in\mathcal{N}_k\setminus\{k\}}c\gamma_l^2+\frac{\lambda}{N}\tilde{\varsigma}_k^{(i)}\right)\left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2\Big]$$

$$= \sum_{k=1}^{N}\left[\varrho_k^{(i)}\left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)\left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)+\varepsilon_k^{(i)}\left(\frac{1+\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2+\varsigma_k^{(i)}\left(\frac{\beta_k}{1+|\mathcal{N}_k|\beta_k}\right)^2\right]$$

$$:= \sum_{k=1}^{N}F_k(\beta_k), \tag{4.1.20}$$

where $\varrho_k^{(i)} = \frac{\lambda}{N}\tilde{\varrho}_k^{(i)}$, $\varepsilon_k^{(i)} = c\gamma_k^2 + \frac{\lambda}{N}\tilde{\varepsilon}_k^{(i)}$, and $\varsigma_k^{(i)} = \sum_{l\in\mathcal{N}_k\setminus\{k\}}c\gamma_l^2 + \frac{\lambda}{N}\tilde{\varsigma}_k^{(i)}$. Hence, we optimize $\{\beta_k\}_{k=1}^N$ by minimizing the approximated upper bound (4.1.20), i.e.,

$$\min_{\{\beta_k\}_{k=1}^N}\sum_{k=1}^{N}F_k(\beta_k). \tag{4.1.21}$$

We can divide the problem into the following N problems,

$$\beta_k^{\mathrm{opt}} = \arg\min_{\beta_k}F_k(\beta_k) \quad k=1,\dots,N. \tag{4.1.22}$$

The derivative of F_k with respect to β_k can be calculated as

$$\frac{\partial F_k}{\partial\beta_k} = \Big[\left\{2\varsigma_k^{(i)}-2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-(|\mathcal{N}_k|-2)\varrho_k^{(i)}\right\}\beta_k$$

$$-\left\{2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-\varrho_k^{(i)}\right\}\Big]\cdot\frac{1}{(1+|\mathcal{N}_k|\beta_k)^3}. \tag{4.1.23}$$

Since the denominator is positive, $\frac{\partial F_k}{\partial\beta_k}=0$ when

$$\beta_k = \frac{2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-\varrho_k^{(i)}}{2\varsigma_k^{(i)}-2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-(|\mathcal{N}_k|-2)\varrho_k^{(i)}}. \tag{4.1.24}$$

We put $\Gamma'_k = 2\varsigma_k^{(i)}-2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-(|\mathcal{N}_k|-2)\varrho_k^{(i)}$ and $\Lambda'_k = 2(|\mathcal{N}_k|-1)\varepsilon_k^{(i)}-\varrho_k^{(i)}$. Considering $\beta_k>0$, we can derive the following optimal parameter:

$$\begin{cases}\beta_k^{\mathrm{opt}} = \frac{\Lambda'_k}{\Gamma'_k}, & \text{if } \Gamma'_k>0 \text{ and } \Lambda'_k>0,\\ \beta_k^{\mathrm{opt}} \to +0, & \text{if } \Gamma'_k>0 \text{ and } \Lambda'_k\le 0,\\ \beta_k^{\mathrm{opt}} \to +\infty, & \text{if } \Gamma'_k\le 0.\end{cases} \tag{4.1.25}$$

Note that sufficiently small β_k implies that the node does not use the neighbors' estimates at the combination step but only uses its own information. When β_k is very large, the resulting combination weight (4.1.3) coincides with the conventional uniform rule (2.2.12).

The optimal parameter (4.1.25) includes unavailable information such as the unknown vector $\boldsymbol{w}^{\mathrm{o}}$, noise variance $\{\sigma_k^2\}$, correlation matrices $\{\boldsymbol{R}_{u_k}\}$, and the number N of all nodes in the network. Therefore, we consider approximating the unknown vector and plugging in adaptive estimates to the other factors, and derive an adaptive algorithm. First, as an adaptive update of the estimate of γ_l^2, we adopt the same equation as (4.1.9). Second, we approximate the unknown vector $\boldsymbol{w}^{\mathrm{o}}$ and the number N of all nodes with the instantaneous estimate $\boldsymbol{\psi}_k^{(i)}$ and the number $|\mathcal{N}_k|$ of neighbors, respectively. The coefficients $\varepsilon_k^{(i)}$, $\varrho_k^{(i)}$, and $\varsigma_k^{(i)}$ are redefined as

$$\hat{\varepsilon}_k^{(i)} = c(\gamma_{kk}^2)^{(i)} + \frac{\lambda}{|\mathcal{N}_k|}\mu_k \mathrm{Re}\left[\partial f(\boldsymbol{\phi}_k^{(i-1)})^{\mathrm{H}}\left\{\lambda\mu_k\partial f(\boldsymbol{\phi}_k^{(i-1)}) + 2\left(\boldsymbol{\psi}_k^{(i)} - \boldsymbol{\phi}_k^{(i-1)}\right)\right\}\right], \tag{4.1.26}$$

$$\hat{\varrho}_k^{(i)} = \frac{\lambda}{|\mathcal{N}_k|}\mathrm{Re}\left[\boldsymbol{\zeta}_k^{(i)\mathrm{H}}\left\{\lambda\mu_k\partial f(\boldsymbol{\phi}_k^{(i-1)}) + 2\left(\boldsymbol{\psi}_k^{(i)} - \boldsymbol{\phi}_k^{(i-1)}\right)\right\} + \mu_k\partial f(\boldsymbol{\phi}_k^{(i-1)})^{\mathrm{H}}\boldsymbol{\iota}_k^{(i)}\right], \tag{4.1.27}$$

$$\hat{\varsigma}_k^{(i)} = c\sum_{l\in\mathcal{N}_k\setminus\{k\}}(\gamma_{lk}^2)^{(i)} + \frac{\lambda}{|\mathcal{N}_k|}\mathrm{Re}\left[\boldsymbol{\zeta}_k^{(i)\mathrm{T}}\boldsymbol{\iota}_k^{(i)}\right], \tag{4.1.28}$$

where $\boldsymbol{\zeta}_k^{(i)} = \sum_{j\in\mathcal{N}_k\setminus\{k\}}\mu_j\partial f(\boldsymbol{\phi}_j^{(i-1)})$ and $\boldsymbol{\iota}_k^{(i)} = \lambda\boldsymbol{\zeta}_k^{(i)} + 2(|\mathcal{N}_k|-1)\boldsymbol{\psi}_k^{(i)} - 2\sum_{j\in\mathcal{N}_k\setminus\{k\}}\boldsymbol{\phi}_j^{(i-1)}$. We further redefine Γ_k' and Λ_k' by using (4.1.26)–(4.1.28) as

$$\Gamma_k^{\prime(i)} = 2\hat{\varsigma}_k^{(i)} - 2(|\mathcal{N}_k|-1)\hat{\varepsilon}_k^{(i)} - (|\mathcal{N}_k|-2)\hat{\varrho}_k^{(i)}, \tag{4.1.29}$$

$$\Lambda_k^{\prime(i)} = 2(|\mathcal{N}_k|-1)\hat{\varepsilon}_k^{(i)} - \hat{\varrho}_k^{(i)}, \tag{4.1.30}$$

respectively.

Summarizing these plug-ins, we can derive an adaptive form of the parameter β_k as follows:

$$\begin{cases} \beta_k^{\mathrm{o},(i)} = \frac{\Lambda_k^{\prime(i)}}{\Gamma_k^{\prime(i)}}, & \text{if } \Gamma_k^{\prime(i)} > 0 \text{ and } \Lambda_k^{\prime(i)} > 0,\\ \beta_k^{\mathrm{o},(i)} \to +0, & \text{if } \Gamma_k^{\prime(i)} > 0 \text{ and } \Lambda_k^{\prime(i)} \le 0,\\ \beta_k^{\mathrm{o},(i)} \to +\infty, & \text{if } \Gamma_k^{\prime(i)} \le 0. \end{cases} \tag{4.1.31}$$

The corresponding combination weight $a_{lk}^{\mathrm{cpo}(i)}$ is described as

$$a_{lk}^{\mathrm{cpo}(i)} = \begin{cases} \frac{\beta_k^{\mathrm{o},(i)}}{1+|\mathcal{N}_k|\beta_k^{\mathrm{o},(i)}}, & \text{if } l \in \mathcal{N}_k\setminus\{k\},\\ \frac{1+\beta_k^{\mathrm{o},(i)}}{1+|\mathcal{N}_k|\beta_k^{\mathrm{o},(i)}}, & \text{if } l = k,\\ 0, & \text{otherwise.} \end{cases} \tag{4.1.32}$$

Algorithm 8 SD-LMS with adaptive CPO rule

```
1: Initialize $\phi_k^{(-1)} = 0, \{(\gamma_{lk}^2)^{(-1)}\}, \ \forall k \in \mathcal{V}, l \in \mathcal{N}_k$
2: for each time $i \in \mathbb{N}$, each node $k \in \mathcal{V}$, and each neighbor $l \in \mathcal{N}_k$ do
3:     $\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k \boldsymbol{u}_k^{(i)}(d_k^{(i)} - \boldsymbol{u}_k^{(i)\mathrm{H}}\phi_k^{(i-1)}) - \mu_k\lambda\partial f(\phi_k^{(i-1)})$
4:     $(\gamma_{lk}^2)^{(i)} = (1-\nu_k)(\gamma_{lk}^2)^{(i-1)} + \nu_k \left\| \psi_l^{(i)} - \phi_k^{(i-1)} + \mu_l\lambda\partial f(\phi_k^{(i-1)}) \right\|^2$
5:     Calculate $\hat{\varepsilon}_k^{(i)}, \hat{\varrho}_k^{(i)}, \hat{\varsigma}_k^{(i)}$ as in (4.1.26)–(4.1.28)
6:     Calculate $\Gamma_k^{'(i)}$ and $\Lambda_k^{'(i)}$ as in (4.1.29) and (4.1.30)
7:     if $\Gamma_k^{'(i)} > 0$ then
8:         if $\Lambda_k^{'(i)} > 0$ then
9:             $\beta_k^{(i)} = \frac{\Lambda_l^{(i)}}{\Gamma_k^{'(i)}}$
10:        else
11:            $\beta_k^{(i)} = +0$ (small positive constant)
12:        end if
13:    else
14:        $\beta_k^{(i)} = +\infty$ (large positive constant)
15:    end if
16:    $a_{lk}^{(i)} = \frac{\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}} \ (l \in \mathcal{N}_k \setminus \{k\}), \quad \frac{1+\beta_k^{(i)}}{1+|\mathcal{N}_k|\beta_k^{(i)}} \ (l = k)$
17:    $\phi_k^{(i)} = \sum_{l \in \mathcal{N}_k} a_{lk}^{(i)} \psi_l^{(i)}$
18: end for
```

SD-LMS using the proposed adaptive optimization is summarized in *Algorithm* 8. We name this weight *adaptive CPO rule*.

4.1.3 Convergence Analysis

In this section, we discuss the convergence of SD-LMS using static CP rule (3.2.15), adaptive CP rule (4.1.11), and adaptive CPO rule (4.1.32) in the mean sense and the mean-square behaviors. The conventional performance analysis framework for SD-LMS in Sect. 2.2.2 is applicable to the proposed SD-LMS with CP and CPO rules because the rules can be regarded as ones of the combination rules.

Corollary 2. *Consider the measurement model given by* (2.2.2) *and the combination weight given by CP rule* (3.2.15)*, adaptive CP rule* (4.1.11)*, and adaptive CPO rule* (4.1.32)*. The convergence of SD-LMS using the proposed rules in the mean sense is guaranteed under Assumption 1 if the step-sizes satisfy*

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{u_k})} \quad (k \in \mathcal{V}).$$

If we use the static CP rule (3.2.15), the mean-square behaviors can be expressed as in Sect 2.2.2. The theoretical network MSD learning curve of SD-LMS

Table 4.1: Computational complexity and communication cost for the proposed and conventional rules par one synchronous communication at node k.

Algorithm	$\times,\div$	$+,-$	Communication cost
SD-LMS w/ static rules	$M\lvert\mathcal{N}_k\rvert$	$M(\lvert\mathcal{N}_k\rvert-1)$	$M(\lvert\mathcal{N}_k\rvert-1)$
SD-LMS w/ adaptive relative-variance rule	$(3M+6)\lvert\mathcal{N}_k\rvert$	$(4M+2)\lvert\mathcal{N}_k\rvert-M-1$	$M(\lvert\mathcal{N}_k\rvert-1)$
SD-LMS w/ adaptive CP rule (proposed)	$(3M+5)\lvert\mathcal{N}_k\rvert+3$	$(4M+3)\lvert\mathcal{N}_k\rvert-M+2$	$M(\lvert\mathcal{N}_k\rvert-1)$
SD-LMS w/ adaptive CPO rule (proposed)	$(4M+5)\lvert\mathcal{N}_k\rvert+10M+20$	$(6M+3)\lvert\mathcal{N}_k\rvert+6M+4$	$2M(\lvert\mathcal{N}_k\rvert-1)$

for the static CP rule can be represented by the same expression as (2.2.42) given by

$$\begin{aligned}\mathrm{MSD}_i^{\mathrm{nw}} &= \mathrm{MSD}_{i-1}^{\mathrm{nw}} + \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}\boldsymbol{F}'^{i}\boldsymbol{q}' - \frac{1}{N}\boldsymbol{w}^{\mathrm{T}}\left[\mathrm{vec}^{-1}\left(\boldsymbol{F}'^{i}\left[\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}'\right]\boldsymbol{q}'\right)\right]\boldsymbol{w}\\ &\quad + g_{\boldsymbol{q}'}^{(i)} + \sum_{j=0}^{i-1}\left(g_{(\boldsymbol{F}')^{i-j}\boldsymbol{q}'}^{(j)} - g_{(\boldsymbol{F}')^{i-j-1}\boldsymbol{q}'}^{(j)}\right). \end{aligned} \tag{4.1.33}$$

The steady-state network MSD and the steady-state MSD at each node k under Assumptions 1 and 2 are also the same as (2.2.45) and (2.2.46) given by

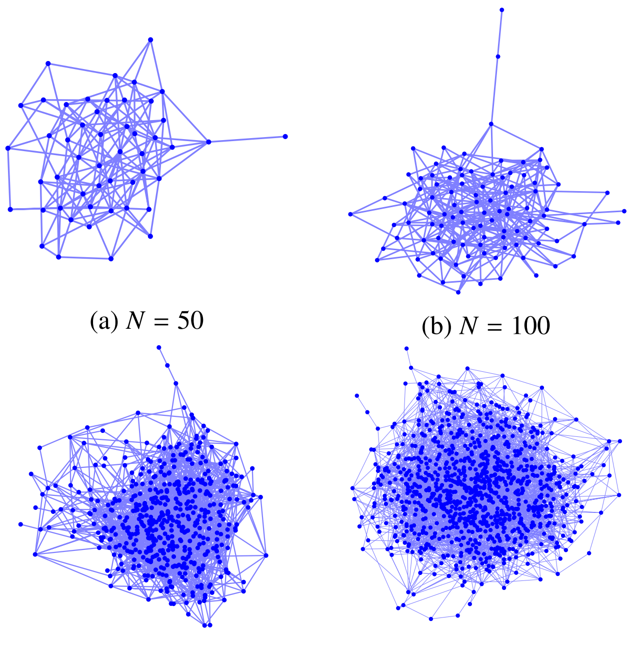
$$\mathrm{MSD}^{\mathrm{nw}} = \frac{1}{N}\boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}\boldsymbol{q}' + \frac{1}{N}\lambda\beta_{\Sigma,\infty}\left(\lambda-\frac{\alpha_{\Sigma,\infty}}{\beta_{\Sigma,\infty}}\right), \tag{4.1.34}$$

$$\mathrm{MSD}_k = \boldsymbol{r}'^{\mathrm{T}}(\boldsymbol{I}_{(MN)^2}-\boldsymbol{F}')^{-1}\boldsymbol{p}' + \lambda\beta_{\Sigma_k,\infty}\left(\lambda-\frac{\alpha_{\Sigma_k,\infty}}{\beta_{\Sigma_k,\infty}}\right), \tag{4.1.35}$$

respectively.

4.1.4 Computational Complexity

We discuss the computational complexity and the communication cost of SD-LMS using the proposed and conventional rules. Table 4.1 shows the complexity and the cost in each combination step at a node, whose definitions are the same as in Sect. 3.3. Note that, since the common adaptation step and calculation of $\partial f(\cdot)$ are required for all algorithms, the complexities only related to the combination step are evaluated. Adaptive CPO rule (Algorithm 8) requires higher computational complexity than the conventional rules and adaptive CP rule (Algorithm 7) because it computes additional coefficients given by (4.1.26)–(4.1.28). Moreover, the


(a) $N = 50$ (b) $N = 100$

(c) $N = 500$ (d) $N = 1000$

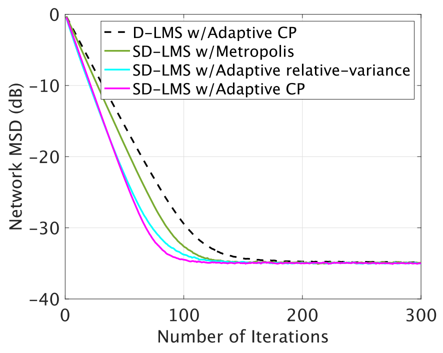
Figure 4.1: Network topologies.

communication cost of the adaptive CPO rule becomes twice because it requires exchanging $\bar{\phi}_l^{(i-1)}$ with neighboring nodes to calculate the additional coefficients but the order is the same as those of the other rules. As mentioned in Sect. 3.3, the communication time typically dominates the execution time in applications such as wireless sensor networks, one may conclude that the required time is comparable.

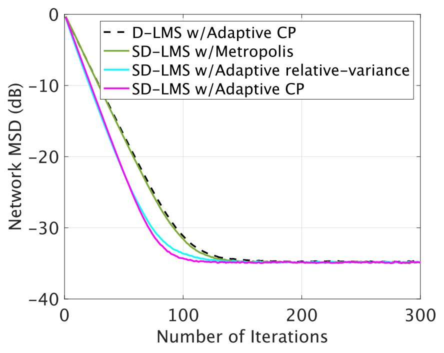
4.2 Simulation Results

4.2.1 Evaluation of Adaptive CP Rule

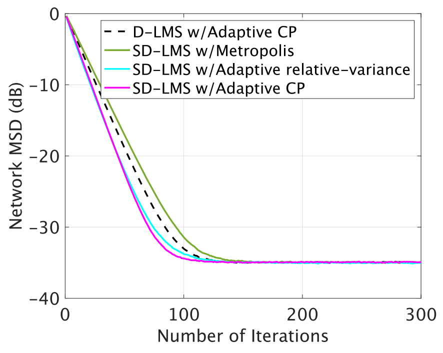
In this section, we verify the performance of SD-LMS with the proposed adaptive CP rule via computer simulations. All the simulation results were obtained by using MATLAB. In order to compare the performance in networks of different sizes, we have generated Erdős-Rényi random networks with $N = 50, 100, 500$, and 1000, as shown in Figs. 4.1 (a)–(d). The average degree of each network was fixed to 6. Note that for the adaptive rules, the results in this subsection employed (3.2.18) for the estimation of γ_l^2. We have used node-independent step-size parameters $\mu_k = \mu$, forgetting factors $\nu_k = \nu$, and initial values $(\gamma_{lk}^2)^{(i)} = (\gamma^2)^{(i)}$, for all k, l. The initial values $(\gamma^2)^{(-1)}$ were set to be 0.1 and the forgetting factor was fixed to be $\nu = 0.05$. The size of the measurement vector was set to be $M = 20$. We



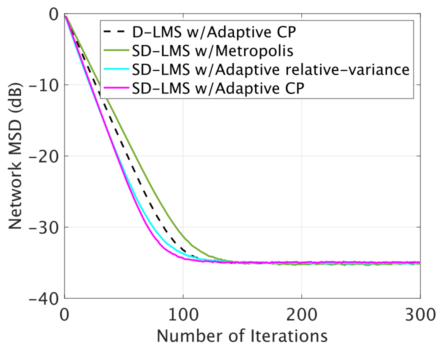
(a) $N = 50$



(b) $N = 100$



(c) $N = 500$



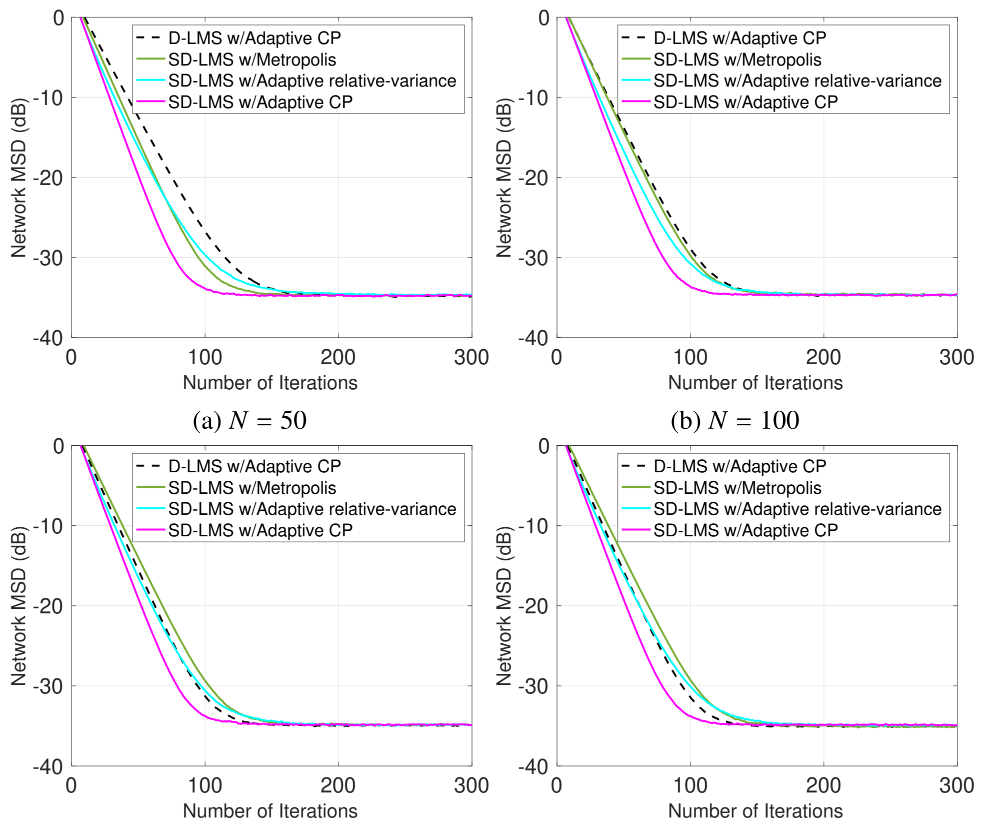
(d) $N = 1000$

Figure 4.2: Network MSD learning curves for 1-sparse.

have investigated instantaneous network MSD $\mathrm{MSD}^{\mathrm{nw,in}}$ of the proposed method and conventional methods.

All the simulation results were obtained by averaging over 20 independent trials. The measurement noise power σ_k^2 was independently generated by uniform distribution over $[0.01, 0.02]$. The measurement vectors $\{\boldsymbol{u}_k^{(i)}\}$ had the same structures as (3.4.1) except that α_k is set to 0 for all k. The indices of the non-zero elements of unknown vector were randomly selected in each trial.

We compare the convergence performance of SD-LMS with that of D-LMS, both of which employ the proposed adaptive CP rule (4.1.11). Also, we have evaluated the performance of SD-LMS using conventional weights, static Metropolis rule a_{lk}^{met} (2.2.13) and adaptive RV rule $a_{lk}^{\mathrm{rv}(i)}$ (3.2.19). The unknown vector was assumed to be 1-sparse whose non-zero element is equal to 1. We have fixed the regularization parameter as $\lambda = 0.0005$ for Metropolis rule and $\lambda = 0.00025$ for adaptive relative-variance rule and adaptive CP rule, with which the steady-state performance of each rule has become the best among our trials. We have controlled the step-size parameters of the algorithms so that the steady-state performance be-



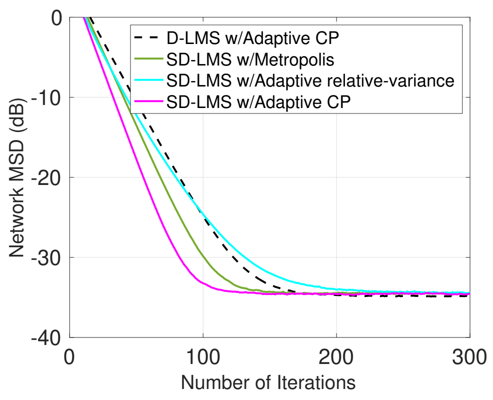
(a) $N = 50$ (b) $N = 100$

(c) $N = 500$ (d) $N = 1000$

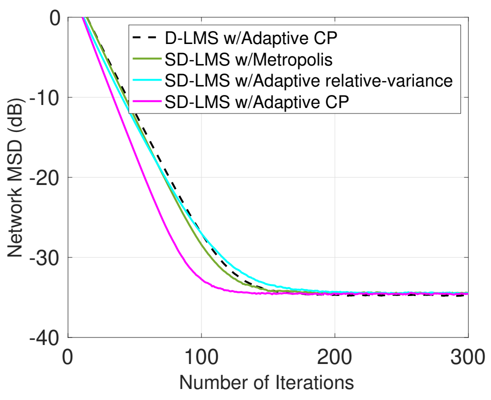
Figure 4.3: Network MSD learning curves for 2-sparse.

came comparable for all algorithms. The parameter ϵ in the regularization function (2.2.39) was set to be $\epsilon = 0.01$. Figs. 4.2 (a)–(d) show the learning curves for the networks of different sizes. The figures demonstrated that SD-LMS with the proposed adaptive CP rule achieved faster convergence than D-LMS and than SD-LMS with the conventional weights regardless of the network size. It should be noted here that we do not see so much difference in the convergence rate among different network sizes. This may be because a larger network has more measurements, which will be beneficial for faster convergence, while a larger network requires more steps for average consensus, so that one can understand both types of the influences have been canceled. We further evaluated the performance when the unknown vector was 2-sparse and 3-sparse. Fig. 4.3 shows the learning curves for 2-sparse and Fig. 4.4 is those for 3-sparse. The figures indicate that the validity of the proposed method is not specific to 1-sparse.

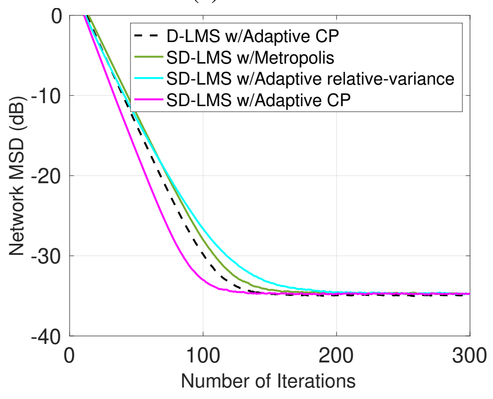
Finally, we verify the impact of the choice of ϵ on the convergence performance of SD-LMS using the proposed adaptive CP rule (4.1.11). Figs. 4.5 (a)–(d) show



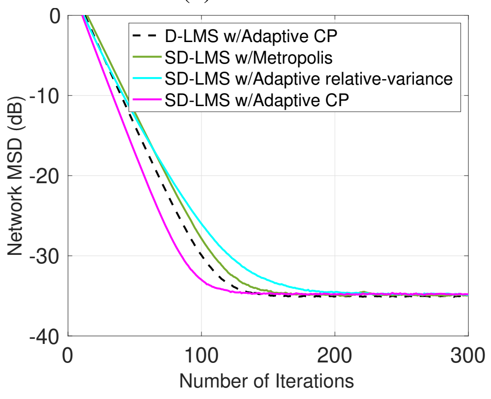
(a) $N = 50$



(b) $N = 100$



(c) $N = 500$

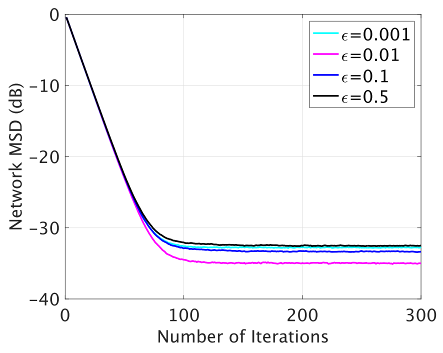


(d) $N = 1000$

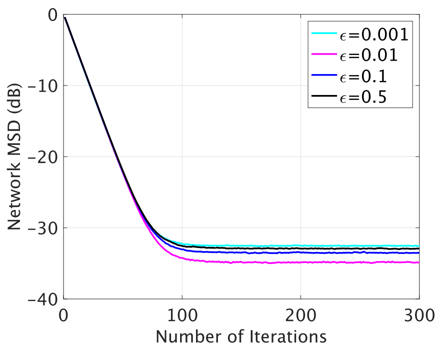
Figure 4.4: Network MSD learning curves for 3-sparse.

the learning curves for the networks of different sizes with $\epsilon = 0.001, 0.01, 0.1, 0.5$. We have fixed step-size parameters $\mu = 0.098$ in Fig. 4.5 (a), $\mu = 0.1$ in Fig. 4.5 (b), $\mu = 0.108$ in Fig. 4.5 (c), and $\mu = 0.109$ in Fig. 4.5 (d). The rest of the parameters were the same as in Fig. 4.2. In our settings, $\epsilon = 0.01$ was the best regardless of the network size and the worst choice resulted in degradation of about 3dB.

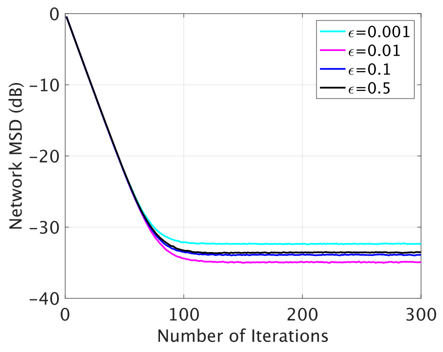
We have also evaluated the optimal value of ϵ for the case with different sparsity and measurement noise variance. Fig. 4.6 demonstrates the learning curves for the network with $N = 50$ shown in Fig. 4.1 (a) when the unknown vector had lower sparsity, namely 10-sparse. Fig. 4.7 shows those when the measurement noise power σ_k^2 was larger, which was independently generated by uniform distribution over $[0.11, 0.12]$. The rest of the parameters in each experiment were the same as in Fig. 4.2. From the figures, we can see that the optimal value of ϵ in Fig. 4.6 was the same as that in Fig. 4.5 (a), while it was different in Fig. 4.7. Thus, the value of ϵ should be appropriately determined, especially depending on the signal-to-noise ratio of measurements.



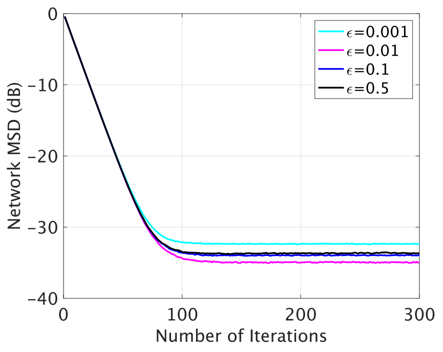
(a) $N = 50$



(b) $N = 100$



(c) $N = 500$



(d) $N = 1000$

Figure 4.5: Network MSD learning curves for comparison of ϵ.

4.2.2 Evaluation of Adaptive CPO Rule

In order to compare the performance in networks of different density, we have generated Erdős-Rényi random networks with $N = 20$ where the mean degrees are $D = 12, 14, 16$, and 18, as shown in Figs. 4.8 (a)–(d). The step-size parameter, the forgetting factor, and the parameter in the regularization function were fixed to be $\mu = 0.05$, $\nu = 0.005$, and $\epsilon = 0.001$, respectively. In this subsection, the equation (4.1.9) was employed for the estimation of γ_l^2. The initial values $(\gamma^2)^{(-1)}$ were set to be 1. We have fixed the regularization parameter as $\lambda = 0.0005$ for $D = 12$ and 14, and $\lambda = 0.0004$ for $D = 16$ and 18, with which the steady-state performance has become the best among our trials. The parameter c that controls the balance of MSD (4.1.19) has been chosen as $c = 0.1, 1$, or 5. The unknown vector was with size $M = 100$ and the number of nonzero element was 1, where the index of the nonzero element switches every 1000 iterations in order to evaluate the tracking performance of the proposed and conventional methods. The measurement vectors $\{\boldsymbol{u}_k^{(i)}\}$ had the same structures as (3.4.1) except that α_k was set to 0 for all k. All simulation results were obtained by averaging 100 independent

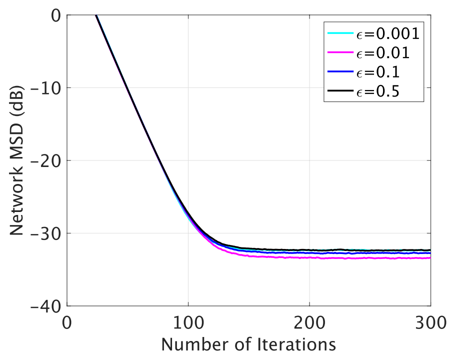


Figure 4.6: Network MSD learning curves with $N = 50$ when unknown vector has lower sparsity.

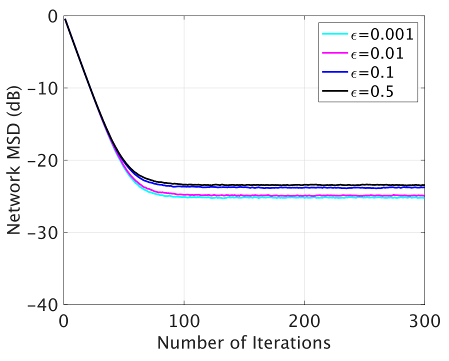


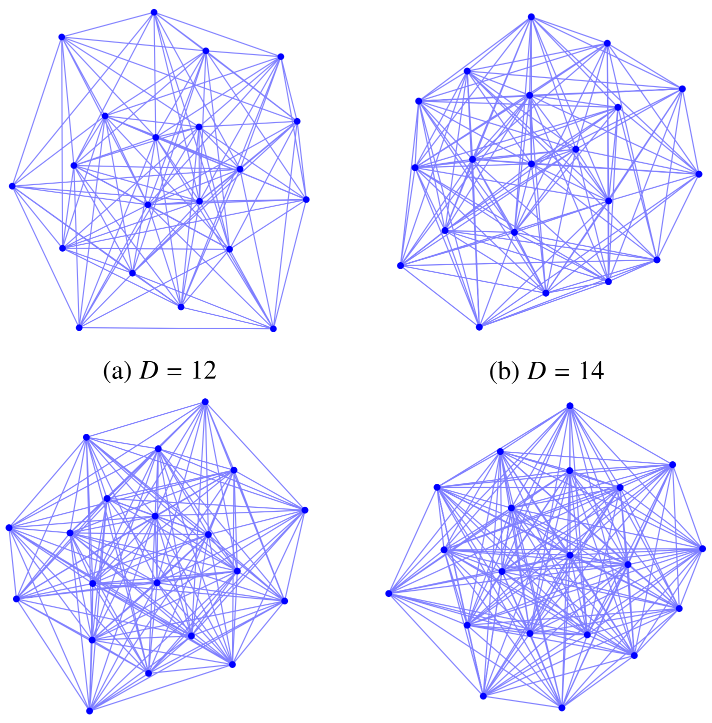
Figure 4.7: Network MSD learning curves with $N = 50$ when measurement noise variance is larger.

trials. The measurement noise power σ_k^2 was independently generated by uniform distribution over $[0.1, 0.2]$ in each trial. We compare learning curves of SD-LMS in terms of instant network MSD $\frac{1}{N}\sum_{k=1}^{N}\|\phi_k^{(i)} - \boldsymbol{w}^{\mathrm{o}}\|^2$ using the proposed adaptive CPO rule (4.1.32) with that using the adaptive CP rule (4.1.11), static Metropolis rule (2.2.13), and adaptive relative-variance (RV) rule (3.2.19).

Figs. 4.9 (a)–(d) show the learning curves in the case of $D = 12, 14, 16,$ and 18, respectively. In Fig. 4.9 (a), the adaptive RV rule converged slightly slower than the proposed adaptive CPO rule and CP rule but achieved the lowest error. However, as the density of the network increased, the algorithm with the proposed adaptive CPO rule achieved faster convergence and lower MSD under the suitable choice of c than that with all the other rules. Namely, the proposed adaptive CPO rule showed the best tracking performance to the change of the unknown vector.

4.3 Conclusion

This chapter has considered achieving faster convergence of SD-LMS for the adaptive sparse signal processing in networks. We have applied loopy CP to SD-LMS and shown that the proposed algorithm results in SD-LMS that uses CP rule determined by the parameters of CP. We have further optimized the parameters in terms of the steady-state MSD of SD-LMS and extended the weight to an adaptive version, adaptive CP rule. We have further improved the approximation of the optimization problem to achieve better convergence performance and robustness, and optimized the parameters. Moreover, we have shown an adaptive implementation for tracking the optimal coefficients, which has been named adaptive CPO rule. Simulation results demonstrated that the proposed method achieves faster convergence than the original SD-LMS. We have also numerically verified that the

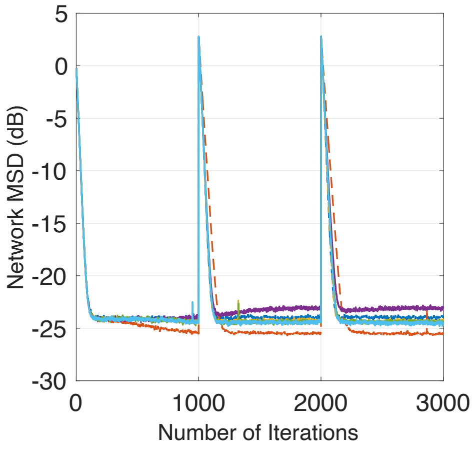


(a) $D = 12$ (b) $D = 14$

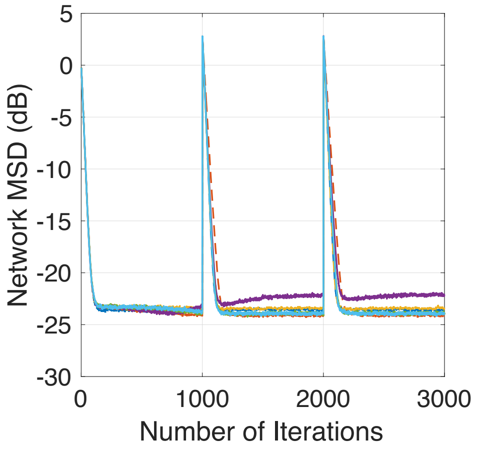
(c) $D = 16$ (d) $D = 18$

Figure 4.8: Network topologies.

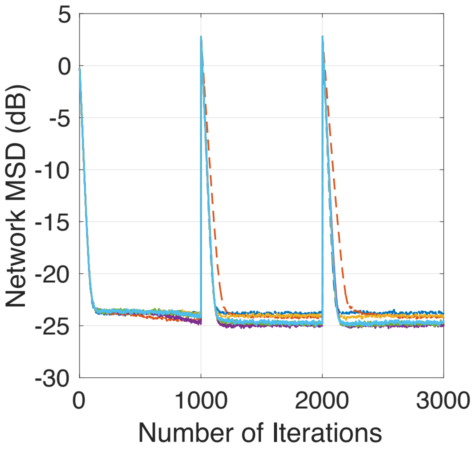
optimal value of ϵ in the regularization function with respect to the convergence performance of the proposed adaptive CP rule significantly depended not on the sparsity of unknown vector but on the signal-to-noise ratio of measurements. The algorithm with the proposed adaptive CPO rule has shown better tracking performance for the change of the unknown vector, especially in dense networks, under the suitable choice of the parameter c, at the cost of a slightly higher computational complexity.



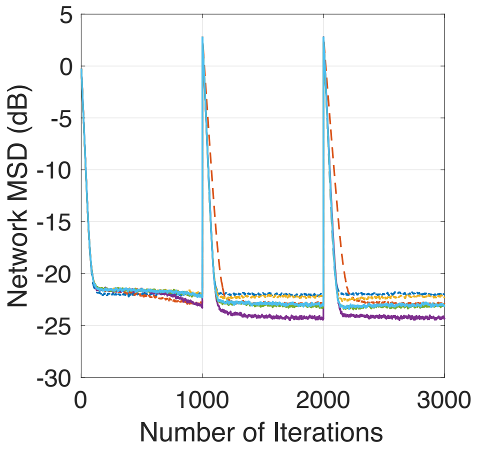
(a) $D = 12$



(b) $D = 14$



(c) $D = 16$



(d) $D = 18$

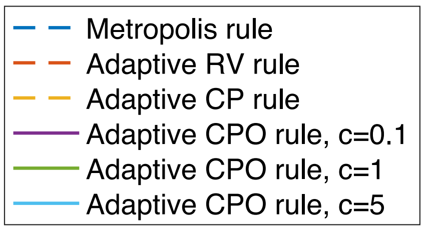


Figure 4.9: Network MSD learning curves for networks with different density.

Part II

New Representation of Scalable GPR

5: Scalable GPR with Sketching

5.1 Introduction

5.1.1 Background and Related Work

Gaussian process regression (GPR or full GPR) (Rasmussen and Williams, 2006) is a non-parametric regression model that assumes a Gaussian process prior on regression functions. This is also known as Kriging (Cressie, 1993; Stein, 2012) especially in geostatistics, geoscience, and mining. Its application in machine learning includes data visualization (Lawrence, 2005), reinforcement learning (Deisenroth et al., 2015), multi-task learning (Ashton and Sollich, 2012), distributed learning (Tavassolipour et al., 2020), to mention a few. Despite its advantage of allowing nonlinear regression, its computational complexity and required memory can be a serious problem in the cases where the number N of training data is large. Full GPR (which we mean the GPR with no approximation) includes inversion of an $N \times N$ matrix, so that it takes $O(N^3)$ time complexity[1] (via conventional methods like Gauss-Jordan elimination and LU decomposition) for training. This would restrict the applicability of full GPR to problems with $N \lesssim 10^4$.

In order to circumvent the limitation, various approximation methods have been proposed. These can be divided into two main categories, global and local approximations (Liu et al., 2020a). The global approximations replace the global representation of the $N \times N$ matrix with small-sized matrices. The simplest approach is Subset-of-Data (SoD) (Quiñonero-Candela and Rasmussen, 2005; Chalupka et al., 2013), which uses the only m ($< N$) data out of all data. The time complexity can be reduced to $O(m^3)$. Although the sophisticated selections of the subset have been proposed in Lawrence et al. (2003) and Keerthi and Chu (2005), but the rest of the data becomes wasteful. For the approaches that adopt information of all the data, compactly supported covariance functions such as Wu (1995); Gneiting (2002); Melkumyan and Ramos (2009) help the covariance matrix to be sparse (Furrer et al., 2006; Gu and Hu, 2012), which reduce the complexity to $O(\alpha N^3)$, where the parameter $0 < \alpha < 1$ is defined depending on the sparsity. However, some functions do not make the covariance matrix positive-definite (Wendland, 2004). The idea that is categorically termed sparse GP typically uses some training points or virtual points, called inducing points or pseudo datapoints (Snelson and Ghahramani, 2005; Quiñonero-Candela and Rasmussen, 2005; Wilson and Nick-

[1]In what follows we consider time complexity under the assumption that arithmetic with matrix elements has complexity $O(1)$.

isch, 2015; Bauer et al., 2016). Sparse GP using m inducing points can reduce time complexity to $O(Nm^2)$. Locations of the inducing points can furthermore be optimized via stochastic variational inference (Hensman et al., 2013). However, the sparse GP methods are not suitable when the underlying function has quick-varying features because in such cases they require a large number of inducing points to achieve good performance, yielding high complexity (Bui and Turner, 2014).

The local approximations first split training data into a number of sub-datasets, then assign an “expert” to each of them, and finally summarize local predictions made by these experts to arrive at the final prediction. The procedure enables us to capture such quick-varying features. Models based on Gaussian mixture of the experts (Yuan and Neubauer, 2009; Zhang and Williamson, 2019) mainly focus on tracking non-stationary features by exploiting MCMC (Markov Chain Monte Carlo). One of the state-of-the-art local approximations is the aggregation method, which includes product-of-experts (PoE) (Hinton, 2002), generalized PoE (GPoE) (Cao and Fleet, 2014), Bayesian committee machine (BCM) (Tresp, 2000), robust BCM (RBCM) (Deisenroth and Ng, 2015), generalized RBCM (GRBCM) (Liu et al., 2018), query-aware BCM (QBCM) (He et al., 2019). Different aggregation methods summarize the local predictions of the experts by using different schemes under some assumptions of independence of the experts. In the methods, covariance information of data is partially lost by approximate divisions of the processing into the experts and the more covariance information becomes lost due to the independence assumptions. One of the aggregation methods known as nested pointwise aggregation of experts (NPAE) (Rullière et al., 2018) does not assume the independence of the experts at the cost of higher computational complexity. The time complexity of the aggregation methods except for NPAE with sub-dataset size n_0 is reduced to $O(Nn_0^2) + O(CNn_0)$, where C is independent of N and varies depending on the method.

An important theoretical property for the aggregation methods is *consistency*, which means that the aggregated prediction converges to the value of the true underlying function when N approaches infinity. The aggregation methods without consistency do not necessarily yield good predictions even in large-sample situations. NPAE and GRBCM are proven to have consistency under appropriate conditions (Bachoc et al., 2017, 2021; Liu et al., 2018). Furthermore, NPAE usually achieves better predictive performance than other methods by using richer information but at the same time requires higher computational complexity.

5.1.2 Contributions

In this thesis, we aim to improve performance of the conventional aggregation method by focusing on the use of covariance information of data that is lost by the approximate divisions. Specifically, we propose a novel aggregation method inspired by NPAE. We first generalize the prediction of NPAE via an idea of low-dimensional projection known as sketching (Liberty, 2013; Woodruff, 2014) of the training samples, and then extend it to more informative versions via introducing inducing points. With its higher flexibility, this method is expected to achieve a better trade-off between predictive performance and computational complexity. We name the proposed method Nested Aggregation of Experts using Inducing Points (NAE-IP). The Gaussian process approximation via sketching has also been considered by Calandriello et al. (2019) but their construction is based on matrix sketching and falls within the category of sparse GP methods. On the other hand, the dimensionality reduction in NAE-IP is different from that of sparse GP in that NAE-IP exploits linear sketching of signals, extending NPAE. Furthermore, NAE-IP, similarly to NPAE, allows parallelization of some part of processing by employing a block-diagonal sketching matrix. NAE-IP is expected to be advantageous in two alternative fronts: one is that it prioritizes predictive performance at the cost of an increase of computational complexity, and another is that it uses a less informative set of inducing points while allowing reduction of the computational complexity. Furthermore, we prove that NAE-IP has consistency under certain conditions. Simulation results show that the proposed method achieves lower prediction errors than conventional aggregation methods, while keeping less computing time than the original NPAE.

5.2 GPR and Its Approximation Methods

5.2.1 Full GPR

Consider a problem that predicts the corresponding output given any input on a region $\mathcal{Q} \subset \mathbb{R}^D$ by using training data composed of pairs of input and noisy output, namely, regression problem. In the full GPR, given a training dataset with N samples, $\mathcal{D} = \{(\boldsymbol{x}_n, z_n) \in \mathcal{Q} \times \mathbb{R}\}_{n=1,\dots,N}$, the following regression model is assumed:

$$z_n = f(\boldsymbol{x}_n) + \epsilon_n, \tag{5.2.1}$$

where the regression function f is assumed to follow a Gaussian process (GP), and where the residual error ϵ_n is assumed to be a white Gaussian noise with mean 0 and variance σ^2, that is, one has $[\epsilon_1, \dots, \epsilon_N]^{\mathrm{T}} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$. The mean function of the GP can be assumed to be 0 without loss of generality. The covariance function

$k(\cdot,\cdot)$ of the GP represents properties of the regression function. Commonly used covariance functions are the squared exponential (SE) function:

$$k(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{r^2}{2}\right), \tag{5.2.2}$$

and the Matérn-($\nu + 1/2$) function:

$$k(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\left(-\sqrt{2\nu+1}r\right)\frac{\nu!}{(2\nu)!}\sum_{\nu'=0}^{\nu}\frac{(\nu+\nu')!}{\nu'!(\nu-\nu')!}\left(2\sqrt{2\nu+1}r\right)^{\nu-\nu'}, \tag{5.2.3}$$

where $r = \sqrt{(\boldsymbol{x}-\boldsymbol{x}')^{\mathrm{T}}\boldsymbol{L}^{-1}(\boldsymbol{x}-\boldsymbol{x}')}$ is the Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ and covariance matrix $\boldsymbol{L} = \mathrm{diag}[\ell_1,\ldots,\ell_D]$, where $\nu \in \mathbb{N}^+$ is a model parameter for Matérn function, and where $\sigma_f^2 > 0$ and $\ell_d > 0$ ($d = 1,\ldots,D$) are hyperparameters. The hyperparameters of these models are thus $\boldsymbol{\Theta} = \{\sigma_f^2, \{\ell_d\}_{d=1,\ldots,D}, \sigma^2\}$, the values of which may be determined via maximizing the log-marginal likelihood

$$\log p(\boldsymbol{z}|\boldsymbol{X},\boldsymbol{\Theta}) = -\frac{1}{2}\left(\boldsymbol{z}^{\mathrm{T}}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{z} + \log\det\left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I}\right]\right), \tag{5.2.4}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D}$, $\boldsymbol{z} = [z_1,\cdots,z_N]^{\mathrm{T}} \in \mathbb{R}^N$, and where the covariance matrix $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}')$ is such that $[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}')]_{nn'} = k(([\boldsymbol{X}]_n)^{\mathrm{T}}, ([\boldsymbol{X}']_{n'})^{\mathrm{T}})$.

Assume that we wish to estimate the values of f at N_T test points $\{\boldsymbol{x}_t^*\}_{t=1,\ldots,N_T}$. All the test points and the corresponding outputs are summarized as $\boldsymbol{X}^* = [\boldsymbol{x}_1^*,\cdots,\boldsymbol{x}_{N_T}^*]^{\mathrm{T}} \in \mathbb{R}^{N_T\times D}$ and $\boldsymbol{z}^* = [z_1^*,\ldots,z_{N_T}^*]^{\mathrm{T}} \in \mathbb{R}^{N_T}$, respectively. The values of the regression function corresponding to $\boldsymbol{X}$ and $\boldsymbol{X}^*$ are summarized as $\boldsymbol{f} = [f(\boldsymbol{x}_1),\cdots,f(\boldsymbol{x}_N)]^{\mathrm{T}} \in \mathbb{R}^N$ and $\boldsymbol{f}^* = [f(\boldsymbol{x}_1^*),\cdots,f(\boldsymbol{x}_{N_T}^*)]^{\mathrm{T}} \in \mathbb{R}^{N_T}$, respectively. On the assumption that the prior of f is GP, the joint distribution of $\boldsymbol{z}$ and $\boldsymbol{f}^*$ is given by

$$p\left(\begin{bmatrix}\boldsymbol{z}\\ \boldsymbol{f}^*\end{bmatrix}\right) = \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\\ \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}^*)\end{bmatrix}\right). \tag{5.2.5}$$

The predictive distribution of $\boldsymbol{f}^*$ given $\mathcal{D}$ is obtained as $p(\boldsymbol{f}^*|\boldsymbol{X}^*,\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\mathrm{full}}(\boldsymbol{X}^*), \boldsymbol{\Sigma}_{\mathrm{full}}(\boldsymbol{X}^*))$, where

$$\boldsymbol{\mu}_{\mathrm{full}}(\boldsymbol{X}^*) = \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{z}, \tag{5.2.6}$$

$$\boldsymbol{\Sigma}_{\mathrm{full}}(\boldsymbol{X}^*) = \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}^*) - \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*). \tag{5.2.7}$$

The prediction of $\boldsymbol{z}^*$ is similarly obtained as $p(\boldsymbol{z}^*|\boldsymbol{X}^*,\mathcal{D}) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathrm{full}}(\boldsymbol{X}^*), \boldsymbol{\Sigma}_{\mathrm{full}}(\boldsymbol{X}^*)+\sigma^2\boldsymbol{I}\right)$. The matrix inversion in (5.2.6) and (5.2.7) has $O(N^3)$ time complexity and $O(N^2)$ memory consumption, so that various approximations have been proposed to circumvent the complexity.

5.2.2 Aggregation Methods

Problem Settings and Training

In this subsection, we introduce the common settings among the aggregation methods. The whole training dataset is first divided into p subsets, $\mathcal{D}_i = (\boldsymbol{X}_i, \boldsymbol{z}_i)$ $(i = 1, \ldots, p)$, where each subset has $n^{(i)}$ data points, namely, $\boldsymbol{X}_i \in \mathbb{R}^{n^{(i)} \times D}$ and $\boldsymbol{z}_i \in \mathbb{R}^{n^{(i)}}$. Sub-models that make predictions using the sub-datasets are referred to as "experts". Each expert $\mathcal{M}_i$ makes own predictions by using its own sub-dataset $\mathcal{D}_i$. The local prediction $p_i(z^*|\boldsymbol{x}^*, \mathcal{D}_i) = \mathcal{N}\left(\mu_i(\boldsymbol{x}^*), \sigma_i^2(\boldsymbol{x}^*)\right)$ at a test point $\boldsymbol{x}_t^* := \boldsymbol{x}^*$ $(t = 1, \ldots, N_T)$ is obtained by applying full GPR to the sub-dataset $\mathcal{D}_i$, as

$$\mu_i(\boldsymbol{x}^*) = \boldsymbol{K}_{*i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{z}_i, \tag{5.2.8}$$

$$\sigma_i^2(\boldsymbol{x}^*) = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{K}_{*i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{K}_{i*} + \sigma^2, \tag{5.2.9}$$

respectively, where $\boldsymbol{K}_{*i} = \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{X}_i)$, $\boldsymbol{K}_{i*} = \boldsymbol{K}_{*i}^{\mathrm{T}}$, and $\boldsymbol{K}_{ij} = \boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j)$ for $i, j = 1, \ldots, p$. Aggregation methods described in the subsequent sections integrate the experts' predictions and yield the final prediction in different manners.

For learning hyperparameters $\boldsymbol{\Theta}$, it is reasonable under these settings to introduce a factorized training process (Deisenroth and Ng, 2015). In the process, the exact marginal likelihood (5.2.4) is approximated by assuming independence of the marginal likelihoods of the experts, i.e.,

$$p(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{\Theta}) \approx \prod_{i=1}^{p} p_i(\boldsymbol{z}_i|\boldsymbol{X}_i, \boldsymbol{\Theta}), \tag{5.2.10}$$

where the experts share the same hyperparameters $\boldsymbol{\Theta}$. The computational complexity for the training process can be reduced compared with full GPR thanks to the independence assumption.

Predictions That Ignore Some Covariance of Expert

PoE (Hinton, 2002), gPoE (Cao and Fleet, 2014), BCM (Tresp, 2000), and rBCM (Deisenroth and Ng, 2015) are aggregation methods that ignore covariance between experts. The original PoE and gPoE assume independence of experts $\{\mathcal{M}_i\}_{i=1,\ldots,p}$. Specifically, the following product approximation is employed,

$$p(z^*|\boldsymbol{x}^*, \mathcal{D}) = \prod_{i=1}^{p} \left(p_i(z^*|\boldsymbol{x}^*, \mathcal{D}_i)\right)^{\beta_{i1}}, \tag{5.2.11}$$

where β_{i1} is the weight assigned to expert $\mathcal{M}_i$. BCM and rBCM assume conditional independence of experts $\mathcal{D}_i \perp \mathcal{D}_j \mid z^*$ for $i, j = 1, \ldots, p$ and $i \neq j$, the specific

Table 5.1: Recommended weight choice for aggregation methods. The choice marked † means that it lacks theoretical justification.

METHOD	β_{i1}	β_{i2}	CONSTRAINT
PoE	1	$1/p$	
GPoE	$1/p$†	$1/p$	$\sum_{i=1}^{p} \beta_{i1} = 1$
BCM	1	β_{i1}	
RBCM	$\frac{\log \sigma_{**}^2 - \log \sigma_i^2}{2}$†	β_{i1}	

approximation of which is given by

$$p(z^*|\boldsymbol{x}^*, \mathcal{D}) = \frac{\prod_{i=1}^{p} (p_i(z^*|\boldsymbol{x}^*, \mathcal{D}_i))^{\beta_{i1}}}{(p(z^*|\boldsymbol{x}^*))^{\sum_{i=1}^{p} \beta_{i1} - 1}}. \tag{5.2.12}$$

The local prediction $p_i(z^*|\boldsymbol{x}^*, \mathcal{D}_i)$ is Gaussian with mean (5.2.8) and variance (5.2.9), so that the aggregated prediction $p(z^*|\boldsymbol{x}^*, \mathcal{D})$, i.e., $p\left(z^*\middle|\boldsymbol{x}^*, \{\mu_i(\boldsymbol{x}^*), \sigma_i^2(\boldsymbol{x}^*)\}_{i=1,\dots,p}\right)$, becomes also Gaussian. Thus the resulting prediction with mean $\mu_{\text{poe/bcm}}(\boldsymbol{x}^*)$ and variance $\sigma^2_{\text{poe/bcm}}(\boldsymbol{x}^*)$ can be collectively formulated as

$$\mu_{\text{poe/bcm}}(\boldsymbol{x}^*) = \sigma^2_{\text{poe/bcm}}(\boldsymbol{x}^*) \sum_{i=1}^{p} \beta_{i1} \sigma_i^{-2}(\boldsymbol{x}^*) \mu_i(\boldsymbol{x}^*), \tag{5.2.13}$$

$$\sigma^{-2}_{\text{poe/bcm}}(\boldsymbol{x}^*) = \sum_{i=1}^{p} \beta_{i1} \sigma_i^{-2}(\boldsymbol{x}^*) + (1 - \sum_{i=1}^{p} \beta_{i2}) \sigma_{**}^{-2}, \tag{5.2.14}$$

where $\sigma_{**}^2 = k(\boldsymbol{x}^*, \boldsymbol{x}^*) + \sigma^2$, and where β_{i2} is the coefficient determined for each model. The choices of the weight and the values of the coefficient recommended in the respective papers, as well as constraints, of those aggregation methods are summarized in Table 5.1.

Two extensions of rBCM, called GRBCM (Liu et al., 2018) and QBCM (He et al., 2019), are recently proposed. These methods assume existence of an informative "global expert" $\mathcal{M}_g := \mathcal{M}_1$, and that every expert can access, in addition to the sub-dataset assigned to it, the sub-dataset $\mathcal{D}_g = \mathcal{D}_1$ assigned to the global expert. Therefore, these methods take account of covariances between the global expert and other experts, but ignore covariance between non-global experts, and assume conditional independence $\mathcal{D}_i \perp \mathcal{D}_j \mid z^*, \mathcal{D}_g$ for $i, j = 2, \dots, p$ and $i \neq j$. Each expert $\mathcal{M}_i$ ($i = 2, \dots, p$) possesses sub-dataset $\mathcal{D}_{+i} = \mathcal{D}_g \cup \mathcal{D}_i$ and makes own predictions with mean $\mu_{+i}(\boldsymbol{x}^*)$ and variance $\sigma^2_{+i}(\boldsymbol{x}^*)$. The global expert also makes prediction with mean $\mu_g(\boldsymbol{x}^*)$ and variance $\sigma_g^2(\boldsymbol{x}^*)$ by using only the global sub-dataset $\mathcal{D}_g$. The aggregated predictions are given by

$$p(z^*|\boldsymbol{x}^*, \mathcal{D}) = \frac{\prod_{i=2}^{p} (p_i(z^*|\boldsymbol{x}^*, \mathcal{D}_{+i}))^{\beta_i}}{(p_1(z^*|\boldsymbol{x}^*, \mathcal{D}_1))^{\sum_{i=2}^{p} \beta_i - 1}} \tag{5.2.15}$$

with the following mean $\mu_{\mathrm{grbcm/qbcm}}(\boldsymbol{x}^*)$ and variance $\sigma^2_{\mathrm{grbcm/qbcm}}(\boldsymbol{x}^*)$,

$$\mu_{\mathrm{grbcm/qbcm}}(\boldsymbol{x}^*) = \sigma^2_{\mathrm{grbcm/qbcm}}(\boldsymbol{x}^*)\left[\sum_{i=2}^{p}\beta_i\sigma_{+i}^{-2}(\boldsymbol{x}^*)\mu_{+i}(\boldsymbol{x}^*) + \left(1-\sum_{i=2}^{p}\beta_i\right)\sigma_g^{-2}(\boldsymbol{x}^*)\mu_g(\boldsymbol{x}^*)\right], \tag{5.2.16}$$

$$\sigma^{-2}_{\mathrm{grbcm/qbcm}}(\boldsymbol{x}^*) = \sum_{i=2}^{p}\beta_i\sigma_{+i}^{-2}(\boldsymbol{x}^*) + \left(1-\sum_{i=2}^{p}\beta_i\right)\sigma_g^{-2}(\boldsymbol{x}^*), \tag{5.2.17}$$

where the experts' weights $\{\beta_i\}_{i=2,\dots,p}$ are chosen in the same manner as rBCM. For GRBCM, the global sub-dataset $\mathcal{D}_g$ is randomly selected from the entire training samples, and for QBCM, $\mathcal{D}_g$ is selected as the sub-dataset with its centroid closest to the test point.

NPAE: Prediction That Uses Covariance between All Experts

NPAE (Rullière et al., 2018) for GPR is also one of the aggregation methods but it yields "consistent" prediction by taking account of covariance between experts, at the cost of computational complexity. The consistency is discussed in the next subsection. In NPAE, the collection of the local predictions $\boldsymbol{\mu}_* = [\mu_1(\boldsymbol{x}^*),\dots,\mu_p(\boldsymbol{x}^*)]^{\mathrm{T}} \in \mathbb{R}^p$ can be regarded as the outputs like $\boldsymbol{z}$ and the prediction is done by GPR with regards to $\boldsymbol{\mu}_*$ by using the covariances of the local predictions. The aggregated prediction is obtained as follows:

$$\mu_{\mathrm{npae}}(\boldsymbol{x}^*) = \boldsymbol{k}_{\mathcal{A}*}^{\mathrm{T}}\boldsymbol{K}_{\mathcal{A}*}^{-1}\boldsymbol{\mu}_*, \tag{5.2.18}$$

$$\sigma^2_{\mathrm{npae}}(\boldsymbol{x}^*) = k(\boldsymbol{x}^*,\boldsymbol{x}^*) - \boldsymbol{k}_{\mathcal{A}*}^{\mathrm{T}}\boldsymbol{K}_{\mathcal{A}*}^{-1}\boldsymbol{k}_{\mathcal{A}*} + \sigma^2, \tag{5.2.19}$$

where $\boldsymbol{k}_{\mathcal{A}*} = \mathrm{Cov}[\boldsymbol{\mu}_*, z^*] \in \mathbb{R}^p$ and $\boldsymbol{K}_{\mathcal{A}*} = \mathrm{Cov}[\boldsymbol{\mu}_*,\boldsymbol{\mu}_*] \in \mathbb{R}^{p\times p}$. This formulation means that NPAE uses the covariance between all experts, that is, uses richer information than those aggregation methods described in Sect. 5.2.2.

Note that the original NPAE is restricted to test-point-wise processing and requires $p \times p$ matrix inversion $\boldsymbol{K}_{\mathcal{A}*}^{-1}$ for each test point, so that its computational complexity is higher than other aggregation methods. Rullière et al. (2018) have also proposed additional complexity reduction of NPAE by considering hierarchical organization of the experts, in which case the subsequent prediction becomes different from (5.2.18) and (5.2.19).

Consistency

Consistency is one of the important properties for the aggregation methods, which means that the aggregated prediction converges to the value of the true underlying function when the number N of training points approaches infinity. It should be noted that the definition of consistency in this thesis is such that an aggregation method for a finite number of test points is said to be consistent if the aggregated predictions provided by the method converges to the values of the true underlying function at those test points in probability, as $N \to \infty$. In particular, the definition is different from, and much weaker than, the consistency in functional spaces (van der Vaart and van Zanten, 2011): Consistency of a method in the above definition does not necessarily imply that the posterior on the functional space provided by the method converges to the Dirac measure at the true underlying function in the limit $N \to \infty$.

NPAE in Sect. 5.2.2 is proven to be consistent in the noiseless case ($\sigma^2 = 0$) (Bachoc et al., 2017) and the noisy case ($\sigma^2 \neq 0$) (Bachoc et al., 2021). For the latter case, NPAE is consistent when the placement of all input points is not too irregular on $\boldsymbol{Q}$ or when the training data is divided by typical clustering algorithms, e.g., k-means. Consistency including noisy observations is also discussed in Liu et al. (2018), where they have concluded that GRBCM is consistent as long as the input points in the global sub-dataset are randomly selected on $\boldsymbol{Q}$. Bachoc et al. (2017, 2021) have also proven that, under some assumptions on the kernel[2], there are cases where consistency of PoE, gPoE, BCM, and rBCM does not hold depending on the distribution of the input points.

5.3 GPR with Sketching

5.3.1 Understanding of Aggregation Method as Sketching

In this subsection, we represent the predictions by NPAE (5.2.18), (5.2.19) in an alternative formulation, with the aim of extending it to a generalized method. As mentioned in Sect. 5.2.1, the high computational complexity of full GPR arises primarily from the necessity of inverting the Gram matrix $(\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})$ with size equal to the number N of training samples. Consequently, all the existing approximation schemes include some ideas of reducing the size of the matrix to be inverted, and accordingly, when evaluating the conditional mean in these schemes one projects $\boldsymbol{z}$ to a low-dimensional subspace determined by the matrix of reduced size. In this thesis, rather than considering a reduced-size matrix to be inverted, we focus on the latter projection procedure. More specifically, we consider a *linear sketch* $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{z} \in \mathbb{R}^{N_u}$ of $\boldsymbol{z}$, where N_u is the dimension of the linear sketch $\boldsymbol{u}$, and where $\boldsymbol{A} \in \mathbb{R}^{N_u \times N}$ is a sketching matrix, and study the problem of estimating the function values at test points not on the basis of $\boldsymbol{z}$ but on the basis of its sketch $\boldsymbol{u}$. As detailed in the following, this approach has advantages in that it provides

[2]Many stationary kernels including the Matérn kernel satisfy the assumption, but the SE kernel does not.

a novel interpretation of NPAE as well as its extensions, and that it allows us to provide a full characterization of the optimal sketching matrix.

In what follows we assume, without loss of generality, that the rows of the sketching matrix $\boldsymbol{A}$ are linearly independent, as adding linearly dependent rows does not add any useful information of $\boldsymbol{z}$ to its linear sketch $\boldsymbol{u}$. The joint probability of $\{\boldsymbol{u}, \boldsymbol{z}^*\}$ is

$$\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{z}^*\end{bmatrix} = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{A}\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I}\right)\boldsymbol{A}^{\mathrm{T}} & \boldsymbol{A}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\\ \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{A}^{\mathrm{T}} & \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}^*)+\sigma^2\boldsymbol{I}\end{bmatrix}\right). \tag{5.3.1}$$

The conditional distribution of $\boldsymbol{z}^*$ given $\boldsymbol{u}$ is calculated as

$$\boldsymbol{z}^*|\boldsymbol{u} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathcal{A}}(\boldsymbol{X}^*), \boldsymbol{\Sigma}_{\mathcal{A}}(\boldsymbol{X}^*)\right), \tag{5.3.2}$$

where

$$\boldsymbol{\mu}_{\mathcal{A}}(\boldsymbol{X}^*) = \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}}\right)^{-1}\boldsymbol{u}, \tag{5.3.3}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathcal{A}}(\boldsymbol{X}^*) = {} & \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}^*)+\sigma^2\boldsymbol{I}\\ & - \boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}}\right)^{-1}\boldsymbol{A}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*).\end{aligned} \tag{5.3.4}$$

The matrix to be inverted in the above formulae is of size $N_u \times N_u$, implying that the time complexity can be significantly reduced by taking $N_u \ll N$. It should be noted that this reduction is different from that of sparse GP methods and that in Calandriello et al. (2019), where the matrix to be inverted is $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_u)\boldsymbol{K}(\boldsymbol{X}_u,\boldsymbol{X}_u)^{-1}\boldsymbol{K}(\boldsymbol{X}_u,\boldsymbol{X})$ ($\boldsymbol{X}_u \in \mathbb{R}^{N_u\times D}$ is the set of inducing points) with size $N \times N$ and the reduction is granted via Woodbury matrix identity.

For sketching matrix $\boldsymbol{A}$ with a general structure, the following proposition holds.

Proposition 1. *Assume row independence of the sketching matrix $\boldsymbol{A}$. The conditional distribution of $\boldsymbol{z}^*$ given the linear sketch $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{z}$ depends on $\boldsymbol{A}$ only through its kernel* $\ker \boldsymbol{A}$.

Proof: Under the row independence, the size of the sketching matrix $\boldsymbol{A}$ is $N_u \times N$ with $N_u = N - \dim\ker\boldsymbol{A}$. For two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N_u\times N}$, $\ker\boldsymbol{A} = \ker\boldsymbol{B}$ holds if and only if $\boldsymbol{A}$ and $\boldsymbol{B}$ are row equivalent, that is, there exists an invertible matrix $\boldsymbol{T} \in \mathbb{R}^{N_u\times N_u}$ satisfying $\boldsymbol{B} = \boldsymbol{T}\boldsymbol{A}$. The conditional distribution of $\boldsymbol{z}^*$ given a linear sketch $\boldsymbol{u}' = \boldsymbol{B}\boldsymbol{z}$ with $\boldsymbol{B} = \boldsymbol{T}\boldsymbol{A}$ is the same as that given the linear sketch $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{z}$, as can be confirmed by the fact that replacing $\boldsymbol{A}$ with $\boldsymbol{B} = \boldsymbol{T}\boldsymbol{A}$ in (5.3.3) and (5.3.4) with $\boldsymbol{z}$ fixed keeps $\boldsymbol{\mu}_{\mathcal{A}}(\boldsymbol{X}^*)$ and $\boldsymbol{\Sigma}_{\mathcal{A}}(\boldsymbol{X}^*)$ invariant. □

One expects that the conditional mean with sketching given in (5.3.3) would give a good approximation of the conditional mean in the full GPR. Goodness of this approximation may be measured via the mean squared error $\mathcal{E} = \mathrm{E}[\|\boldsymbol{\mu}_{\mathcal{A}}(\boldsymbol{X}^*) - \boldsymbol{\mu}_{\text{full}}(\boldsymbol{X}^*)\|^2)$ between the conditional means with and without sketching. It is evaluated as

$$\begin{aligned}\mathcal{E} &= \mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\right] \\ &\quad - \mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}}\right)^{-1}\boldsymbol{A}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\right]. \qquad (5.3.5)\end{aligned}$$

The next proposition provides a full characterization of the optimal sketching matrix in the sense of minimizing $\mathcal{E}$.

Proposition 2. *For a given dimension N_u of the linear sketching $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{z} \in \mathbb{R}^{N_u}$ of $\boldsymbol{z}$, the optimal sketching matrix $\boldsymbol{A} \in \mathbb{R}^{N_u \times N}$ in the sense of minimizing the mean squared error $\mathcal{E}$ is such that the N_u row vectors of $\boldsymbol{A}$ span the subspace spanned by the eigenvectors of $(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})$ corresponding to its N_u largest eigenvalues.*

Proof: Since the first term on the right-hand side of (5.3.5) is independent of $\boldsymbol{A}$, the optimum sketching matrix $\boldsymbol{A}$ minimizing the mean squared error $\mathcal{E}$ is the matrix that maximizes

$$J(\boldsymbol{A}) = \frac{1}{2}\mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{A}^{\mathrm{T}}\left(\boldsymbol{A}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}}\right)^{-1}\boldsymbol{A}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\right].$$

The matrix $\boldsymbol{C} = \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}$ is symmetric and positive definite, so that it is diagonalized by an orthogonal matrix $\boldsymbol{V}$ as $\boldsymbol{C} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{V}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with the diagonal elements consisting of the eigenvalues of $\boldsymbol{C}$. Letting $\boldsymbol{C}^{1/2} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}$ and $\boldsymbol{A}' = \boldsymbol{T}\boldsymbol{A}\boldsymbol{C}^{1/2}$, where $\boldsymbol{T}$ is an invertible matrix corresponding to the Gram-Schmidt orthogonalization applied to the row vectors of $\boldsymbol{A}\boldsymbol{C}^{1/2}$ such that $\boldsymbol{A}'(\boldsymbol{A}')^{\mathrm{T}} = \boldsymbol{I}$ holds, one has

$$\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{V}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A}\boldsymbol{C}^{1/2}\left(\boldsymbol{A}\boldsymbol{C}^{1/2}\right)^{\mathrm{T}} = \boldsymbol{T}^{-1}\boldsymbol{A}'(\boldsymbol{A}')^{\mathrm{T}}\boldsymbol{T}^{-\mathrm{T}} = \boldsymbol{T}^{-1}\boldsymbol{T}^{-\mathrm{T}}.$$

The cost function $J(\boldsymbol{A})$ can then be written as

$$\begin{aligned}J(\boldsymbol{A}) &= \frac{1}{2}\mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{C}^{-1/2}(\boldsymbol{A}')^{\mathrm{T}}\boldsymbol{A}'\boldsymbol{C}^{-1/2}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\right] \\ &= \frac{1}{2}\mathrm{Tr}\left[\boldsymbol{A}'\boldsymbol{C}^{-1/2}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{C}^{-1/2}(\boldsymbol{A}')^{\mathrm{T}}\right].\end{aligned}$$

Therefore, the optimal sketching matrix is such that the N_u row vectors of $\boldsymbol{A}' = \boldsymbol{T}\boldsymbol{A}\boldsymbol{C}^{1/2}$ span the subspace spanned by the eigenvectors of $\boldsymbol{C}^{-1/2}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})\boldsymbol{C}^{-1/2}$ corresponding to its N_u largest eigenvalues. This coincides with the statement of the proposition. $\square$

Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$ be the eigenvalues of $(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X})$. Then, the mean squared error with the optimal sketching matrix is given by

$$\mathcal{E} = \sum_{i=N_u+1}^{N} \lambda_i.$$

Since $\operatorname{rank}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*) = \operatorname{rank}\boldsymbol{K}(\boldsymbol{X}^*,\boldsymbol{X}) \leq \min\{N, N_T\}$, one has $\lambda_i = 0$ for $i > \min\{N, N_T\}$. Therefore, in order to make the mean squared error $\mathcal{E}$ smaller, it would make no sense to take $N_u > N_T$ if there is no restriction in the choice of the sketching matrix $\boldsymbol{A}$, because $N_u > N_T$ allows us to make $\mathcal{E} = 0$ with the optimal choice of $\boldsymbol{A}$.

The approach of optimizing the sketching matrix with a general structure, however, would require inversion of $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}$ and/or solving a (generalized) eigenvalue problem with a large full-rank matrix, so that its computational complexity should be high.

We next consider block-structured sketching, in which one assumes $\boldsymbol{A}$ to have the following block structure:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A}_2 & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{A}_p \end{bmatrix},$$

where $\boldsymbol{A}_i \in \mathbb{R}^{n_u^{(i)} \times n^{(i)}}$ with $\sum_{i=1}^{p} n_u^{(i)} = N_u$ and $\sum_{i=1}^{p} n^{(i)} = N$. This block-structured sketching allows us to perform a certain fraction of the calculations in a distributed manner, with p computing agents (i.e., experts). The prediction in (5.3.2) exactly coincides with that of NPAE when $n_u^{(i)} = 1$ and $\boldsymbol{A}_i$ is chosen as

$$\boldsymbol{A}_i = \boldsymbol{K}_{*i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1}, \tag{5.3.6}$$

for all experts. In this case, $\boldsymbol{A}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}^*)$ and $\boldsymbol{A}\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right)\boldsymbol{A}^{\mathrm{T}}$ in (5.3.1) are replaced as $\boldsymbol{k}_{\mathcal{A}*}$ and $\boldsymbol{K}_{\mathcal{A}*}$, respectively. Thanks to this formulation, we can regard the choice of $\boldsymbol{A}_i$ in NPAE as a dimensionality reduction from the size $n^{(i)}$ of the sub-dataset $\mathcal{D}_i$ to $n_u^{(i)} = 1$.

5.3.2 Proposed Algorithm: NAE-IP

Derivation

The choice of the matrix $\boldsymbol{A}_i$ in (5.3.3) and (5.3.4) is not limited to that of NPAE (5.3.6). Furthermore, $\boldsymbol{A}_i$ does not even have to be dependent on $\boldsymbol{X}^*$. We then propose a novel aggregation method on the basis of (5.3.2) and name it *Nested*

Aggregation of Experts using Inducing Points (NAE-IP), which is not limited to be “pointwise,” that is, it allows simultaneous prediction on multiple test points. In the proposed method, we select the following choice for $\boldsymbol{A}_i$:

$$\boldsymbol{A}_i = \boldsymbol{K}_{\eta_i i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1}, \tag{5.3.7}$$

where $\boldsymbol{K}_{\eta_i i} = \boldsymbol{K}(\bar{\boldsymbol{X}}_i, \boldsymbol{X}_i) \in \mathbb{R}^{n_u^{(i)} \times n^{(i)}}$ and $\boldsymbol{K}_{i\eta_i} = \boldsymbol{K}_{\eta_i i}^{\mathrm{T}}$ for a collection $\bar{\boldsymbol{X}}_i \in \mathbb{R}^{n_u^{(i)} \times D}$ of $n_u^{(i)}$ inducing points. It should be noticed that (5.3.7) is the same as (5.3.6) except that the test points $\boldsymbol{X}^*$ in the latter is replaced by the collection $\bar{\boldsymbol{X}}_i$ of inducing points. In other words, we consider the projection from the size $n^{(i)}$ of the sub-dataset $\mathcal{D}_i$ to the number $n_u^{(i)}$ of inducing points.

We show the prediction scheme using (5.3.7) in a way that follows NPAE. Assume that each expert $\mathcal{M}_i$ has a set of inducing points $\bar{\boldsymbol{X}}_i$ in addition to its own sub-dataset $\mathcal{D}_i$. There is no constraint on the choice of the inducing points but their total number N_u is assumed to be less than N for achieving dimensionality reduction. First, each expert defines an estimator $\bar{\boldsymbol{\mu}}_i$ on the basis of its observation $\boldsymbol{z}_i$ as

$$\bar{\boldsymbol{\mu}}_i = \boldsymbol{K}_{\eta_i i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1} \boldsymbol{z}_i = \boldsymbol{A}_i \boldsymbol{z}_i \in \mathbb{R}^{n_u^{(i)}}, \quad i = 1, \dots, p. \tag{5.3.8}$$

Second, the estimators are concatenated to form a random vector $\bar{\boldsymbol{\mu}} = [\bar{\boldsymbol{\mu}}_1^{\mathrm{T}}, \dots, \bar{\boldsymbol{\mu}}_p^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_u}$. The covariances involving $\bar{\boldsymbol{\mu}}$ and $\boldsymbol{z}^*$ are calculated as

$$\bar{\boldsymbol{k}}_{\mathcal{A}} = \mathrm{Cov}[\bar{\boldsymbol{\mu}}, \boldsymbol{z}^*] = \begin{bmatrix} \boldsymbol{A}_1\boldsymbol{K}_{1*} \\ \vdots \\ \boldsymbol{A}_p\boldsymbol{K}_{p*} \end{bmatrix} \in \mathbb{R}^{N_u \times N_T}, \tag{5.3.9}$$

$$\bar{\boldsymbol{K}}_{\mathcal{A}} = \mathrm{Cov}[\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\mu}}] = \left[\left[\bar{\boldsymbol{K}}_{\mathcal{A}}\right]_{[i][j]}\right]_{i,j=1,\dots,p} \in \mathbb{R}^{N_u \times N_u},$$

$$\left[\bar{\boldsymbol{K}}_{\mathcal{A}}\right]_{[i][j]} = \begin{cases} \boldsymbol{K}_{\eta_i i}\left(\boldsymbol{K}_{ii} + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{K}_{i\eta_i} & \text{if } i = j, \\ \boldsymbol{A}_i\boldsymbol{K}_{ij}\boldsymbol{A}_j^{\mathrm{T}} & \text{if } i \neq j. \end{cases} \tag{5.3.10}$$

Finally, the predictive mean $\bar{\boldsymbol{\mu}}_{\mathcal{A}}$ and covariance $\bar{\boldsymbol{\Sigma}}_{\mathcal{A}}$ of NAE-IP are derived as

$$\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*) = \bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{\mu}}, \tag{5.3.11}$$

$$\bar{\boldsymbol{\Sigma}}_{\mathcal{A}}(\boldsymbol{X}^*) = \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) - \bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{k}}_{\mathcal{A}} + \sigma^2\boldsymbol{I}. \tag{5.3.12}$$

These formulae correspond to $\boldsymbol{\mu}_{\mathcal{A}}(\boldsymbol{X}^*)$ and $\boldsymbol{\Sigma}_{\mathcal{A}}(\boldsymbol{X}^*)$ in (5.3.2), respectively, when the choice of (5.3.7) for $\boldsymbol{A}_i$ is employed.

The following proposition holds and is used for proving the consistency of NAE-IP discussed later.

Proposition 3. *$\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*)$ in (5.3.11) is the best linear unbiased estimator of $\boldsymbol{f}^*$ on the basis of $\bar{\boldsymbol{\mu}}$, where the coefficient matrix $\phi = [\phi_1^{\mathrm{T}} \dots \phi_p^{\mathrm{T}}]^{\mathrm{T}}$ of $\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*) = \phi^{\mathrm{T}}\bar{\boldsymbol{\mu}} = \sum_{i=1}^{p} \phi_i^{\mathrm{T}}\bar{\boldsymbol{\mu}}_i$ is given by $\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{k}}_{\mathcal{A}}$. The mean squared error $v(\boldsymbol{X}^*) = \mathrm{E}\left[\|\boldsymbol{f}^* - \bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*)\|^2\right]$ of the estimator $\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*)$ of $\boldsymbol{f}^*$ is given by $\mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) - \bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{k}}_{\mathcal{A}}\right]$.*

Proof: Using (5.3.11) and (5.3.12), the mean squared error of an estimator $\phi^{\mathrm{T}}\bar{\boldsymbol{\mu}}$ of $\boldsymbol{f}^*$, with $\bar{\boldsymbol{\mu}}$ defined above, is written as

$$\mathrm{E}\left[\|\boldsymbol{f}^* - \phi^{\mathrm{T}}\bar{\boldsymbol{\mu}}\|^2\right] = \mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) - 2\phi^{\mathrm{T}}\bar{\boldsymbol{k}}_{\mathcal{A}} + \phi^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}\phi\right].$$

The value of $\hat{\phi}$ minimizing it is found by differentiation: $-2\bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}} + 2\hat{\phi}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}} = \mathbf{O}$, which leads to $\hat{\phi} = \bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{k}}_{\mathcal{A}}$ and $\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*) = \hat{\phi}^{\mathrm{T}}\bar{\boldsymbol{\mu}}$. Then, $v(\boldsymbol{X}^*) = \mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) - 2\hat{\phi}^{\mathrm{T}}\bar{\boldsymbol{k}}_{\mathcal{A}} + \hat{\phi}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}\hat{\phi}\right]$ and the statement follows. □

One may perform prediction on the N_T test points in a pointwise manner, repeating predition on a single point N_T times, or all at once, predicting for the N_T test points simultaneously. In view of the computational complexity to be discussed later, we consider a more general framework in which the N_T test points are partitioned into S subsets $\boldsymbol{X}_1^*, \dots, \boldsymbol{X}_S^*$ with $\bigcup_{s=1}^{S} \boldsymbol{X}_s^* = \boldsymbol{X}^*$ and $\boldsymbol{X}_s \cap \boldsymbol{X}_{s'} = \emptyset$ for $s \neq s'$, and the prediction is performed on each of these subsets separately. Assume now that the prediction is to be made on the target subset $\boldsymbol{X}_s^*$ of $n_t^{(s)}$ test points. Then, there are 5 possible options of inducing points $\bar{\boldsymbol{X}}_i$ for expert i in NAE-IP:

1. $\bar{\boldsymbol{X}}_i = \boldsymbol{X}_s^*$: Use the test points themselves as the inducing points. In this case $n_u^{(i)} = n_t^{(s)}$. [Blockwise Test points (**BT**)]

2. $\bar{\boldsymbol{X}}_i = \{\boldsymbol{x} \in \boldsymbol{X}_e^* \mid \boldsymbol{X}_e^* \subset \boldsymbol{X}^*, \boldsymbol{X}_e^* \neq \boldsymbol{X}_s^*\}$: Use a part of test points, $\boldsymbol{X}_e^*$, which is not equal to the target subset $\boldsymbol{X}_s^*$. We can set $n_u^{(i)}$ arbitrarily while satisfying $n_u^{(i)} < N_T$. [Blockwise Test points and Other Test points (**BT+OT**), Arbitrary Test points (**AT**)]

3. $\bar{\boldsymbol{X}}_i = \{\boldsymbol{x} \in (\boldsymbol{X}_o \cup \boldsymbol{X}_s^*) \mid \boldsymbol{X}_o \cap \boldsymbol{X}^* = \emptyset, \boldsymbol{X}_o \neq \emptyset\}$: Use both the target subset of test points, $\boldsymbol{X}_s^*$, and non-test points. We can set $n_u^{(i)}$ arbitrarily while satisfying $n_u^{(i)} > n_t^{(s)}$. [Blockwise Test points and Non-Test points (**BT+NT**)]

4. $\bar{\boldsymbol{X}}_i = \{\boldsymbol{x} \in \boldsymbol{X}_o \mid \boldsymbol{X}_o \cap \boldsymbol{X}^* = \emptyset, \boldsymbol{X}_o \neq \emptyset\}$: Use only non-test points as the inducing points. We can set $n_u^{(i)}$ arbitrarily. [Non-Test points (**NT**)]

5. $\bar{\boldsymbol{X}}_i = \{\boldsymbol{x} \in (\boldsymbol{X}_o \cup \boldsymbol{X}_e^*) \mid \boldsymbol{X}_o \cap \boldsymbol{X}^* = \emptyset, \boldsymbol{X}_e^* \subset \boldsymbol{X}^*, \boldsymbol{X}_e^* \neq \boldsymbol{X}_s^*, \boldsymbol{X}_o \neq \emptyset\}$: Use both a part of test points, $\boldsymbol{X}_e^*$, which is not equal to the target subset $\boldsymbol{X}_s^*$, and non-test points. We can set $n_u^{(i)}$ arbitrarily.

Option 1 is an extension of the original NPAE (Rullière et al., 2018) to multiple dimensions. As option 2, we can consider two natural choices, one that completely includes test points themselves (BT+OT) and another that partially or never includes them (AT). BT+OT and BT+NT use higher-dimensional sketching at each expert by incorporating auxiliary points as its inducing points. Extension of sketching dimensions employed in these options is expected to improve prediction accuracy, at the expense of increased computational complexity. The idea of BT, BT+OT, and BT+NT are known as transduction (Quiñonero-Candela and Rasmussen, 2005) that uses the test points of interest for prediction. The transduction could be beneficial because the test points should have some information about the corresponding outputs. A drawback with these options is that the covariance matrix $\bar{\boldsymbol{K}}_{\mathcal{A}}$ depends on all or some test points in the target subset $\boldsymbol{X}_s^*$, so that one has to construct it, as well as to perform matrix inversion, for every target subset. On the other hand, AT and NT require the construction of $\bar{\boldsymbol{K}}_{\mathcal{A}}$ only once for all the target subsets of test points, as long as the inducing points are fixed. It brings about a significant reduction of the complexity. Option 5 might not yield a better prediction than BT+OT or BT+NT. Therefore we focus on BT, BT+OT, AT, BT+NT, and NT in the rest of this chapter and expect an improvement of the predictive performance by using the extended dimensions $n_u^{(i)} \geq n_t^{(s)}$.

Summary of Proposed Algorithm

In this subsection, we summarize the procedure of the proposed NAE-IP. Definitions of symbols used for NAE-IP are summarized in Table 5.2. We write the covariance matrix as $\boldsymbol{K}_{\boldsymbol{\theta}}(\cdot,\cdot)$ in order to make explicit its dependence on the hyperparameters $\boldsymbol{\theta}$ of the covariance function. The whole training dataset is divided into p subsets by using some clustering algorithms or at random. Each sub-dataset $(\boldsymbol{X}_i, \boldsymbol{z}_i)$ is assigned to an expert. To learn hyperparameters, we adopt the factorized training process (Deisenroth and Ng, 2015). We first specify an option from BT, BT+OT, BT+NT, AT, or NT and construct inducing points $\{\bar{\boldsymbol{X}}_i\}_{i=1}^p$ by Algorithm 9. We then perform NAE-IP as shown in Algorithm 10.

Consistency of NAE-IP

We study consistency of NAE-IP in the noisy case by extending the proof of consistency of NPAE in Bachoc et al. (2021). The following assumption is necessary only for NAE-IP.

Assumption 4. *For a test point $\boldsymbol{x}^* \in Q$, estimation of $f(\boldsymbol{x}^*)$ is done by including the test point $\boldsymbol{x}^*$ as an inducing point of all experts.*

For $N \in \mathbb{N}$, let p_N be the number of experts, which may depend on N, and let

Table 5.2: Definitions.

$k(\boldsymbol{x}, \boldsymbol{x}')$	Covariance function (e.g., SE function (Eq. (5.2.2)) or Matérn-$(\nu + 1/2)$ function (Eq. (5.2.3))
$\boldsymbol{\theta}$	Set of hyperparameters included in covariance function
σ^2	Noise variance
$\Theta = \{\boldsymbol{\theta}, \sigma^2\}$	Set of hyperparameters
$\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_i, \boldsymbol{X}_j)$	Covariance matrix whose (m, l)th element is $k(([\boldsymbol{X}_i]_m)^{\mathrm{T}}, ([\boldsymbol{X}_j]_l)^{\mathrm{T}})$ with hyperparameters $\boldsymbol{\theta}$
$\{\boldsymbol{X}_i, \boldsymbol{z}_i\}$	Sub-dataset of ith expert
$\bar{\boldsymbol{X}}_i$	Inducing points of ith expert
$n_u^{(i)}$	Number of inducing points of ith expert
p	Number of experts
$\boldsymbol{X}^*$	Test data points
$\boldsymbol{X}_s^*$	sth subset of test data points
$n_t^{(s)}$	Number of test data points in sth subset
S	Number of subsets of test data points

$\boldsymbol{X}_1, \dots, \boldsymbol{X}_{p_N}$ be the sub-datasets, where $\boldsymbol{X}_i$ $(i = 1, \dots, p_N)$, being a subset of $\boldsymbol{X}$, is the sub-dataset assigned to expert i. We also require the following assumption on the sub-datasets.

Assumption 5. *There exists a sequence $\{j_N(\boldsymbol{x}^*)\}_{N \in \mathbb{N}}$ of indices $j_N(\boldsymbol{x}^*) \in \{1, \dots, p_N\}$ depending on a given test point $\boldsymbol{x}^*$ such that, for any $\rho > 0$, the number of the input points in $\boldsymbol{X}_{j_N(\boldsymbol{x}^*)}$ lying within the ρ-ball $B_\rho(\boldsymbol{x}^*) = \{\boldsymbol{x} \in Q : \|\boldsymbol{x} - \boldsymbol{x}^*\| < \rho\}$ centered at $\boldsymbol{x}^*$ goes to infinity as $N \to \infty$.*

Under these assumptions, Proposition 4 below establishes consistency of NAE-IP at a fixed test point $\boldsymbol{x}^* \in Q$.

Proposition 4. *Let Q be a compact nonempty subset of $\mathbb{R}^D$. Let f be a Gaussian process on Q with mean zero and continuous covariance function k. Let $\{\boldsymbol{x}_{Nn}\}_{1 \le n \le N, N \in \mathbb{N}}$ be a triangular array of input points, all of which lie in Q. For $N \in \mathbb{N}$, let $\boldsymbol{X} = [\boldsymbol{x}_{N1}, \dots, \boldsymbol{x}_{NN}]^{\mathrm{T}}$, and let $\bar{\boldsymbol{\mu}}_1, \dots, \bar{\boldsymbol{\mu}}_{p_N}$ be the collection of p_N experts' estimates defined in* (5.3.8) *on the basis of respective sub-datasets $(\boldsymbol{X}_1, \boldsymbol{z}_1), \dots, (\boldsymbol{X}_{p_N}, \boldsymbol{z}_{p_N})$. Assume that each row of $\boldsymbol{X}$ is a row of at least one $\boldsymbol{X}_i$. For a test point $\boldsymbol{x}^* \in Q$, assume further that $\boldsymbol{X}_1, \dots, \boldsymbol{X}_{p_N}$ satisfy Assumption 5. For such a test point $\boldsymbol{x}^*$, under Assumption 4 we have*

$$\lim_{N \to \infty} \mathrm{E}\left[(f(\boldsymbol{x}^*) - \bar{\mu}_{\mathcal{A}}(\boldsymbol{x}^*))^2\right] = 0. \tag{5.3.13}$$

where $\bar{\mu}_{\mathcal{A}}(\boldsymbol{x}^*)$ is as in (5.3.11).

Algorithm 9 Construction of inducing points

```
1: Input: Option (BT, BT+OT, BT+NT, AT, or NT), $\boldsymbol{X}^*$, $\boldsymbol{X}_s^*$, $n_t^{(s)}$, $p$, $n_u^{(i)}(\geq n_t^{(s)},\ i = 1,\dots,p)$
2: Output: $\{\bar{\boldsymbol{X}}_i\}_{i=1}^p$
3: switch (Option)
4:     case BT:
5:         $\bar{\boldsymbol{X}}_i = \boldsymbol{X}_s^* \quad (i = 1,\dots,p)$
6:         $n_u^{(i)} = n_t^{(s)} \quad (i = 1,\dots,p)$
7:         break
8:     case BT+OT:
9:         Choose $n_u^{(i)} - n_t^{(s)}$ test points $\boldsymbol{X}_e^*$ where $\boldsymbol{X}_e^* \subset \boldsymbol{X}^*, \boldsymbol{X}_e^* \cap \boldsymbol{X}_s^* = \emptyset \quad (i = 1,\dots,p)$
10:        $\bar{\boldsymbol{X}}_i = (\boldsymbol{X}_e^* \cup \boldsymbol{X}_s^*) \quad (i = 1,\dots,p)$
11:        break
12:    case BT+NT:
13:        Choose $n_u^{(i)} - n_t^{(s)}$ non-test points $\boldsymbol{X}_o$ where $\boldsymbol{X}_o \cap \boldsymbol{X}^* = \emptyset \quad (i = 1,\dots,p)$
14:        $\bar{\boldsymbol{X}}_i = (\boldsymbol{X}_o \cup \boldsymbol{X}_s^*) \quad (i = 1,\dots,p)$
15:        break
16:    case AT:
17:        Choose $n_u^{(i)}$ test points $\boldsymbol{X}_e^*$ where $\boldsymbol{X}_e^* \subset \boldsymbol{X}^*, \boldsymbol{X}_e^* \neq \boldsymbol{X}_s^* \quad (i = 1,\dots,p)$
18:        $\bar{\boldsymbol{X}}_i = \boldsymbol{X}_e^* \quad (i = 1,\dots,p)$
19:        break
20:    case NT:
21:        Choose $n_u^{(i)}$ non-test points $\boldsymbol{X}_o$ where $\boldsymbol{X}_o \cap \boldsymbol{X}^* = \emptyset \quad (i = 1,\dots,p)$
22:        $\bar{\boldsymbol{X}}_i = \boldsymbol{X}_o \quad (i = 1,\dots,p)$
23:        break
24: end switch
```

Proof: By Assumption 4, expert $j_N(\boldsymbol{x}^*)$ has the test point $\boldsymbol{x}^*$ as its inducing point, that is, $\boldsymbol{x}^*$ is a component of $\bar{\boldsymbol{X}}_{j_N(\boldsymbol{x}^*)}$. Let $a_j(\boldsymbol{x}^*)$ be the index of the test point $\boldsymbol{x}^*$ in $\bar{\boldsymbol{X}}_{j_N(\boldsymbol{x}^*)}$. With these notations, since $\bar{\mu}_{\mathcal{A}}(\boldsymbol{x}^*)$ is a linear combination of the elements of $\bar{\boldsymbol{\mu}}$ with minimal square prediction errors from Proposition 3, its square prediction error is not larger than that of any single element of $\bar{\boldsymbol{\mu}}$. We hence have

$$\mathrm{E}\left[(f(\boldsymbol{x}^*) - \bar{\mu}_{\mathcal{A}}(\boldsymbol{x}^*))^2\right] \leq \mathrm{E}\left[\left(f(\boldsymbol{x}^*) - [\bar{\boldsymbol{\mu}}_{j_N(\boldsymbol{x}^*)}(\boldsymbol{x}^*)]_{a_j(\boldsymbol{x}^*)}\right)^2\right] \tag{5.3.14}$$

From Assumption 5, for any fixed $\rho > 0$, the number ι of input points lying within $B_\rho(\boldsymbol{x}^*)$ goes to infinity as $N \to \infty$. Let $\boldsymbol{x}_{j_N^{(1)}(\boldsymbol{x}^*)}, \dots, \boldsymbol{x}_{j_N^{(\iota)}(\boldsymbol{x}^*)}$ and $z_{j_N^{(1)}(\boldsymbol{x}^*)}, \dots, z_{j_N^{(\iota)}(\boldsymbol{x}^*)}$ be such input points and the corresponding observations, respectively. Since $[\bar{\boldsymbol{\mu}}_{j_N(\boldsymbol{x}^*)}(\boldsymbol{x}^*)]_{a_j(\boldsymbol{x}^*)} = \boldsymbol{K}_{*j_N(\boldsymbol{x}^*)}\left(\boldsymbol{K}_{j_N(\boldsymbol{x}^*)j_N(\boldsymbol{x}^*)} + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{z}_{j_N(\boldsymbol{x}^*)}$ is also a linear combination of the ele-

Algorithm 10 NAE-IP

```
1: **Input**: Option, $\boldsymbol{X}^*$, $p$, $S$, $n_t^{(s)}$ $(s = 1,\dots,S)$, $\{\boldsymbol{X}_i, \boldsymbol{z}_i\}, n_u^{(i)} (\geq \max_s n_t^{(s)})$ $(i = 1,\dots,p)$
2: **Output**: $\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}^*), \bar{\boldsymbol{\Sigma}}_{\mathcal{A}}(\boldsymbol{X}^*)$
3: Obtain hyperparameters $\boldsymbol{\Theta}$ via factorized training process
4:     $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \sigma^2\} = \arg\max_{\tilde{\boldsymbol{\theta}},\tilde{\sigma}^2}\Big\{-\sum_{i=1}^{p}\Big(\boldsymbol{z}_i^{\mathrm{T}}(\boldsymbol{K}_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{X}_i, \boldsymbol{X}_i)$
5:                 $+\tilde{\sigma}^2\boldsymbol{I})^{-1}\boldsymbol{z}_i + \log\det\Big[\boldsymbol{K}_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{X}_i, \boldsymbol{X}_i) + \tilde{\sigma}^2\boldsymbol{I}\Big]\Big)\Big\}$
6: Partition test points $\boldsymbol{X}^*$ into $S$ subsets $\boldsymbol{X}_s^*$, where the number of points is $n_t^{(s)}$ $(s = 1,\dots,S)$
7: Construct inducing points by **Algorithm 1**
8: **for** each partition $\boldsymbol{X}_s^*$ **do**
9:     **for** each expert $i = 1,\dots,p$ **do**
10:        $\boldsymbol{A}_i = \boldsymbol{K}_{\boldsymbol{\theta}}(\bar{\boldsymbol{X}}_i, \boldsymbol{X}_i)\Big(\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_i, \boldsymbol{X}_i) + \sigma^2\boldsymbol{I}\Big)^{-1}$
11:        $\bar{\boldsymbol{\mu}}_i = \boldsymbol{A}_i\boldsymbol{z}_i$
12:    **end for**
13:    $\bar{\boldsymbol{\mu}} = [\bar{\boldsymbol{\mu}}_1^{\mathrm{T}}, \dots, \bar{\boldsymbol{\mu}}_p^{\mathrm{T}}]^{\mathrm{T}}$
14:    $\bar{\boldsymbol{k}}_{\mathcal{A}} = [\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_s^*, \boldsymbol{X}_1)\boldsymbol{A}_1^{\mathrm{T}}, \dots, \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_s^*, \boldsymbol{X}_p)\boldsymbol{A}_p^{\mathrm{T}}]^{\mathrm{T}}$
15:    $\Big[\bar{\boldsymbol{K}}_{\mathcal{A}}\Big]_{[i][j]} = \begin{cases} \boldsymbol{K}_{\boldsymbol{\theta}}(\bar{\boldsymbol{X}}_i, \boldsymbol{X}_i)\Big(\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_i, \boldsymbol{X}_i) + \sigma^2\boldsymbol{I}\Big)^{-1}\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_i, \bar{\boldsymbol{X}}_i) & \text{if } i = j \\ \boldsymbol{A}_i\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}_i, \boldsymbol{X}_j)\boldsymbol{A}_j^{\mathrm{T}} & \text{if } i \neq j \end{cases}$ $(i, j = 1,\dots,p)$
16:    $\bar{\boldsymbol{\mu}}_{\mathcal{A}}(\boldsymbol{X}_s^*) = \bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{\mu}}$
17:    $\bar{\boldsymbol{\Sigma}}_{\mathcal{A}}(\boldsymbol{X}_s^*) = \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{X}^*, \boldsymbol{X}^*) - \bar{\boldsymbol{k}}_{\mathcal{A}}^{\mathrm{T}}\bar{\boldsymbol{K}}_{\mathcal{A}}^{-1}\bar{\boldsymbol{k}}_{\mathcal{A}} + \sigma^2\boldsymbol{I}$
18: **end for**
```

ments of $\boldsymbol{z}_{j_N(\boldsymbol{x}^*)}$ with minimal square prediction errors, we have, similarly as above,

$$\mathrm{E}\left[\left(f(\boldsymbol{x}^*) - [\bar{\boldsymbol{\mu}}_{j_N(\boldsymbol{x}^*)}(\boldsymbol{x}^*)]_{a_j(\boldsymbol{x}^*)}\right)^2\right] \leq \mathrm{E}\left[\left(f(\boldsymbol{x}^*) - \frac{1}{\iota}\sum_{a=1}^{\iota} z_{j_N^{(a)}(\boldsymbol{x}^*)}\right)^2\right]. \tag{5.3.15}$$

From the independence of the noise process and Cauchy-Schwarz inequality, the right-hand side can further be bounded as

$$\begin{aligned}
\mathrm{E}\left[\left(f(\boldsymbol{x}^*) - \frac{1}{\iota}\sum_{a=1}^{\iota} z_{j_N^{(a)}(\boldsymbol{x}^*)}\right)^2\right] &= \mathrm{E}\left[\left(f(\boldsymbol{x}^*) - \frac{1}{\iota}\sum_{a=1}^{\iota}\left(f\left(\boldsymbol{x}_{j_N^{(a)}(\boldsymbol{x}^*)}\right) + \epsilon_{j_N^{(a)}(\boldsymbol{x}^*)}\right)\right)^2\right] \\
= \mathrm{E}\left[\left(\frac{1}{\iota}\sum_{a=1}^{\iota}\left(f(\boldsymbol{x}^*) - f\left(\boldsymbol{x}_{j_N^{(a)}(\boldsymbol{x}^*)}\right)\right)\right)^2\right] &+ \mathrm{E}\left[\left(\frac{1}{\iota}\sum_{a=1}^{\iota}\epsilon_{j_N^{(a)}(\boldsymbol{x}^*)}\right)^2\right] \\
\leq \left(\max_{a=1,\dots,\iota}\mathrm{E}\left[\left(f(\boldsymbol{x}^*) - f\left(\boldsymbol{x}_{j_N^{(a)}(\boldsymbol{x}^*)}\right)\right)^2\right]\right) &+ \frac{\sigma^2}{\iota}
\end{aligned} \tag{5.3.16}$$

The second term on the rightmost side of (5.3.16) converges to zero as $\iota \to \infty$ because σ^2 is finite. From the continuity of k as in Bachoc et al. (2021, Appendix

E), one has

$$\begin{aligned}\limsup_{N\to\infty} \mathrm{E}\left[\left(f(\boldsymbol{x}^*) - \frac{1}{\iota}\sum_{a=1}^{\iota} z_{j_N^{(a)}(\boldsymbol{x}^*)}\right)^2\right] &\le \sup_{\boldsymbol{\xi}\in B_\rho(\boldsymbol{x}^*)} \mathrm{E}\left[(f(\boldsymbol{x}^*) - f(\boldsymbol{\xi}))^2\right] \\ &= \sup_{\boldsymbol{\xi}\in B_\rho(\boldsymbol{x}^*)} \left[k(\boldsymbol{x}^*,\boldsymbol{x}^*) + k(\boldsymbol{\xi},\boldsymbol{\xi}) - 2k(\boldsymbol{x}^*,\boldsymbol{\xi})\right] \\ &\overset{\rho\to 0}{\to} 0. \end{aligned} \tag{5.3.17}$$

The fact that the limit supremum of a nonnegative sequence converges to 0 implies that the limit also converges. We therefore obtain the statement of the proposition.

□

Note that Proposition 4 proves convergence of $\bar{\mu}_{\mathcal{A}}(\boldsymbol{x}^*)$ to $f(\boldsymbol{x}^*)$ in the mean square sense, which in turn implies convergence in probability, hence establishing the desired consistency.

Options BT, BT+OT, and BT+NT satisfy Assumption 4 from their definitions. Options AT and NT can also include the test point as an inducing point of all experts under the additional assumptions.

Corollary 3. *Under the conditions of Proposition 4, NAE-IP-BT, BT+OT, and BT+NT have consistency.*

Corollary 4. *Under the conditions of Proposition 4 and the assumption that* $n_u^{(i)} \to N_T$ $(i = 1, \dots, p_N)$ *as* $N \to \infty$*, NAE-IP-AT has consistency.*

Corollary 5. *Under the conditions of Proposition 4 and the assumption that* $n_u^{(i)} \to \infty$ $(i = 1, \dots, p_N)$ *as* $N \to \infty$*, NAE-IP-NT has consistency.*

Proof: In NAE-IP-NT, let $\{\boldsymbol{x}_{\eta_i u^{(i)}}\}_{1\le u^{(i)}\le n_u^{(i)}, 1\le i\le p}$ be an array of inducing points. For each $\boldsymbol{x}^* \in Q$ and for $i = 1, \dots, p$, there exists at least one inducing point such that $\lim_{n_u^{(i)}\to\infty} \min_{u^{(i)}} \|\boldsymbol{x}_{\eta_i u^{(i)}} - \boldsymbol{x}^*\| = 0$ and then the estimation of $f(\boldsymbol{x}^*)$ satisfies Assumption 4. □

Assumption 5 holds in typical division of the training data into sub-datasets (Bachoc et al., 2021). When we adopt clustering algorithms such as k-means for the division, the condition holds under the assumption $\min_{i=1,\dots,p_N} n^{(i)} \to \infty$ as $N \to \infty$. In the case where the input points are distributed with strictly positive density on Q, it also holds under the assumption that the number p_N of experts is $o(N)$ as $N \to \infty$.

Time Complexity

The complexity in time is one of the main interests of approximated Gaussian process regression. We show the complexity of the conventional aggregation methods and our methods proposed in this chapter in Table 5.3 under a sufficiently large

Table 5.3: Time complexity of aggregation methods under sufficiently large N, where α, β, and γ are constants larger than 1.

METHOD	TIME COMPLEXITY
(G)PoE/(R)BCM	$O\left(\frac{N^3}{p^2}\right) + O\left(\frac{N_T N^2}{p}\right)$
GRBCM	$O\left(\frac{\alpha N^3}{p^2}\right) + O\left(\frac{\beta N_T N^2}{p}\right)$
QBCM	$O\left(\frac{\gamma\alpha N^3}{p^2}\right) + O\left(\frac{\beta N_T N^2}{p}\right)$
(ORIGINAL) NPAE	$O\left(\frac{N^3}{p^2}\right) + O\left(N_T N^2\right)$
NAE-IP-BT	$O\left(\frac{N^3}{p^2}\right) + O\left(N_T N^2\right)$
NAE-IP-BT+OT/BT+NT	$O\left(\frac{N^3}{p^2}\right) + O\left(\frac{N_T n_u N^2}{n_t}\right)$
NAE-IP-AT/NT	$O\left(\frac{N^3}{p^2}\right) + O\left(n_u N^2\right)$

N, where, for simplicity, we consider the case of equal dimensions $n_u^{(i)} = n_u$ and equally divided sub-datasets $n^{(i)} = N/p$ among the experts, and of equally divided partitions $n_t^{(s)} = N_T/S = n_t$. GRBCM and QBCM require slightly higher complexity than PoE, gPoE, BCM, and rBCM because each expert uses the modified sub-dataset $\mathcal{D}_{+i}$. As mentioned in Sect. 5.2.2, the complexity of NPAE is higher than these methods but keeps lower than that of full GPR when $N_T < N$. Rullière et al. (2018) reported the complexity as $O\left(\frac{N^3}{p^2}\right) + O\left(N_T N^2\right)$ which is the same as NAE-IP-BT, but the frequency of memory access can be reduced by a factor of n_t in the proposed methods. NAE-IP-BT+OT and NAE-IP-BT+NT require higher complexity than the original NPAE by considering n_u dimensions. On the other hand, the complexity of NAE-IP-AT and NAE-IP-NT can be lower than that of not only the original NPAE but also the other methods, depending on the choice of n_u.

In the following, we briefly describe a proof sketch of the complexity of NAE-IP. The main factors that affect the complexity are calculation of the inverse $\left(\boldsymbol{K}_{ii} + \sigma^2 \boldsymbol{I}\right)^{-1}$ in (5.3.8), which is to be performed by every expert, and the construction of $\bar{\boldsymbol{K}}_{\mathcal{A}}$ in (5.3.10). The former takes $O((n^{(i)})^3)$ at p experts, thus resulting in $O(p(n^{(i)})^3) = O(N^3/p^2)$. Next, each block of $\bar{\boldsymbol{K}}_{\mathcal{A}}$ includes the product of $n_u \times n^{(i)}$ matrix and $n^{(i)} \times n^{(i)}$ matrix, and there are p^2 blocks in $\bar{\boldsymbol{K}}_{\mathcal{A}}$, so that these amount to $O(p^2 n_u (n^{(i)})^2) = O(n_u N^2)$ computation. The construction of $\bar{\boldsymbol{K}}_{\mathcal{A}}$ is repeated $S = N_T/n_t$ times for BT, BT+OT, and BT+NT, resulting in $O(N_T N^2)$ for BT and $O(N_T n_u N^2/n_t)$ for the others, whereas it is performed only once for AT and NT, resulting in $O(n_u N^2)$.

5.4 Simulation Results

5.4.1 Datasets and settings

We evaluated the predictive performance and the computing time of NAE-IP in comparison with conventional methods. All the results were obtained by using GPML MATLAB Code[3] (Rasmussen and Williams, 2006). We measured total CPU time on a linux computer with two CPUs (Intel Xeon Gold 5222, 4 cores, 3.8 GHz base clock) and 768 GB RAM. The datasets used in the numerical experiments are summarized below.

- 1-D *Synthetic data*: Synthetic data generated by $z_n = \mathrm{sinc}(x_n) + \epsilon_n$, $n = 1, \ldots, N$, where the training points lie in the interval $[-4, 4]$ uniformly, where N_T test points are uniformly chosen in $[-5, 5]$, and where $[\epsilon_1, \ldots, \epsilon_N]^{\mathrm{T}} \sim \mathcal{N}(\mathbf{0}, 0.04\boldsymbol{I})$.
- 8-D *KIN8NM dataset*[4] (Vanschoren et al., 2013): The data related to the forward dynamics of an 8-link robot arm. There are 8,192 samples in total. We randomly split them into 7,373 samples for training and 819 samples for testing.
- 21-D *SARCOS dataset*[5] (Rasmussen and Williams, 2006): The data related to the inverse dynamics problem of robot arms. There are 44,484 training samples and 4,449 test samples.
- 26-D *POL dataset*[6]: Pole telecom dataset. There are 10,000 training samples and 5,000 test samples.

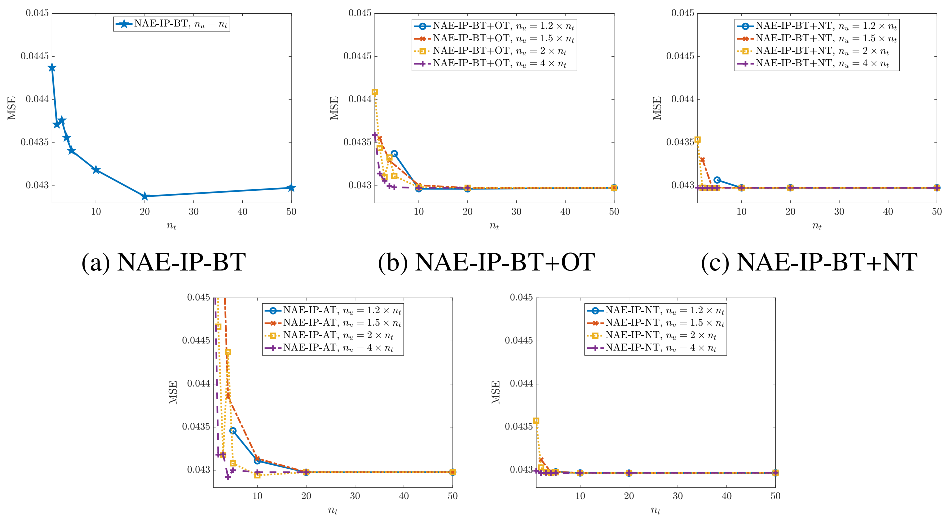
For each experimental condition, we performed 10 trials or more, each consisting of training and prediction procedures. We have used two performance measures, mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{N_T} \sum_{i=1}^{N_T} (\hat{\mu}(x_{t,i}) - z(x_{t,i}))^2, \tag{5.4.1}$$

and mean standardized log loss (MSLL):

$$\mathrm{MSLL} = \frac{1}{N_T} \sum_{i=1}^{N_T} \left(\frac{1}{2} \log\left(2\pi\hat{\sigma}^2(x_{t,i})\right) + \frac{(\hat{\mu}(x_{t,i}) - z(x_{t,i}))^2}{2\hat{\sigma}^2(x_{t,i})} \right), \tag{5.4.2}$$

[3]http://www.gaussianprocess.org/gpml/code/matlab/doc/
[4]https://www.openml.org/d/189
[5]http://www.gaussianprocess.org/gpml/data/
[6]https://cims.nyu.edu/~andrewgw/pattern/



(a) NAE-IP-BT (b) NAE-IP-BT+OT (c) NAE-IP-BT+NT

(d) NAE-IP-AT (e) NAE-IP-NT

Figure 5.1: MSE versus the number n_t of test points.

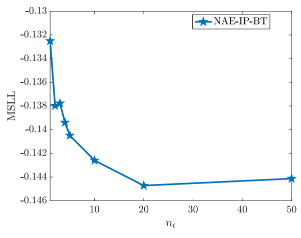
where $\hat{\mu}(x_{t,i})$, $\hat{\sigma}^2(x_{t,i})$, $z(x_{t,i})$ are the predictive mean, variance, and true value at the test point $x_{t,i}$, respectively. MSLL is the mean of pointwise negative log losses of the Gaussian models with mean $\hat{\mu}(x_{t,i})$ and variance $\hat{\sigma}^2(x_{t,i})$ given data $\{z(x_{t,i})\}$, and takes into account uncertainty of the predictions via the posterior variances. The lower MSE and MSLL imply the better prediction.

For the proposed methods, we have assumed equal dimensions $n_u^{(i)} = n_u$ among all experts and almost equally divided partitions $n_t^{(s)} = n_t$ of the test points. Other test points for NAE-IP-OT and arbitrary test points for NAE-IP-AT have been chosen randomly from the remaining test points and from the entire test points, respectively.

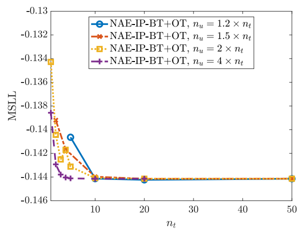
5.4.2 Synthetic data

For the synthetic data, the training data were divided into sub-datasets by k-means. The SE function (5.2.2) was employed as the covariance function of GP. The non-test points of each expert in NAE-IP-BT+NT and NAE-IP-NT were generated from the multivariate Gaussian distribution with the same mean and covariance as those of the sub-dataset assigned to that expert.

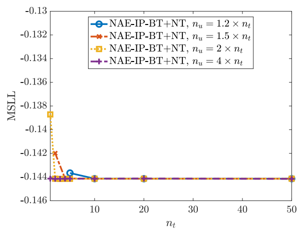
First, we investigate the influence on NAE-IP's performance of the dimension n_u and the number of test points n_t processed at once. Figures 5.1–5.3 show the performance measures and computing time versus n_t when $N = 10^4, N_T = 100$, and $p = 20$. We set the dimension n_u to be $1.2 \times n_t$, $1.5 \times n_t$, $2 \times n_t$, and $4 \times n_t$. When $n_t \leq 20$, the larger n_t and n_u showed the better predictive performance, and



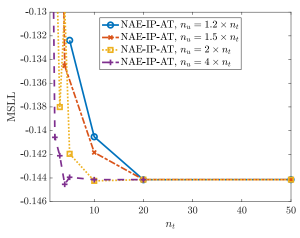
(a) NAE-IP-BT



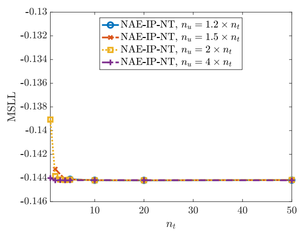
(b) NAE-IP-BT+OT



(c) NAE-IP-BT+NT



(d) NAE-IP-AT

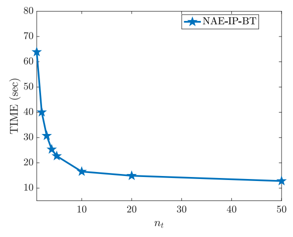


(e) NAE-IP-NT

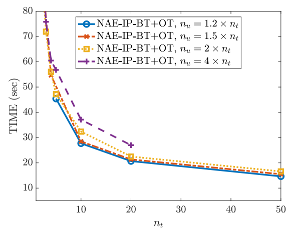
Figure 5.2: MSLL versus the number n_t of test points.

the performance became stable in most cases. The higher dimensions required the more computing time, and the computing time of BT, BT+OT, and BT+NT decreased as n_t increased. On the other hand, the computing time of AT and NT was kept small regardless of n_t. Note that, depending on the value of p or N_T, the larger n_t ($\leq n_u$ in this chapter) does not always yield the shorter computing time because the complexity required for evaluating the inversion $\boldsymbol{K}_{\mathcal{A}*}^{-1}$ is $O(n_u^3 p^3)$ and it could be higher than that of other factors.

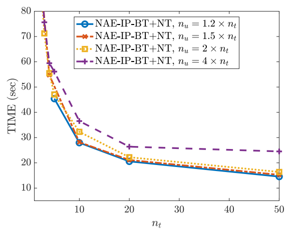
Second, we compare the predictive performance and computing time of NAE-IP with those of conventional aggregation methods, PoE, GPoE, BCM, RBCM, GRBCM, QBCM, and the original NPAE in 30 trials. We evaluated the cases $N = 10^4, 5 \times 10^4$, and 10^5 with $N_T = N \times 10^{-2}$ and $p = N/500$, and chose $(n_t, n_u) = (50, 75)$ for NAE-IP except for NAE-IP-BT. Fig. 5.4 shows the performance measures versus N. Table 5.4 summarizes the results of statistical significance testing for a difference between the best-performing method and each of the other 11 methods. We first checked the normality of data via the one-sample two-sided Kolmogorov-Smirnov test ($P < 0.05$), and then employed paired t-test with Bonferroni multiple testing correction ($P < 0.05/11$) for the statistical significance testing. The results for the cases of $N = 5 \times 10^4$ and 10^5 reveal that the MSE of the proposed NAE-IP was equal to or better than the other methods and the MSLL was the lowest among the methods. This indicates that NAE-IP can obtain not only better predictive means but also smaller predictive variances. Moreover, NAE-IP took less time than the original NPAE. This might be ascribed to difference in mem-



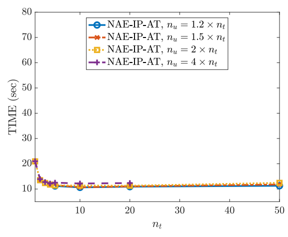
(a) NAE-IP-BT



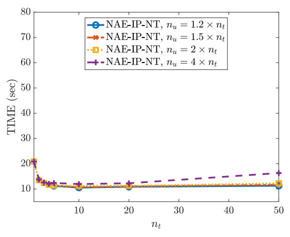
(b) NAE-IP-BT+OT



(c) NAE-IP-BT+NT



(d) NAE-IP-AT



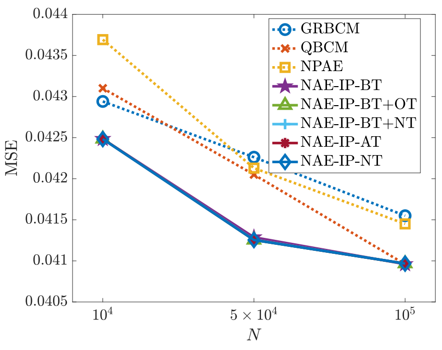
(e) NAE-IP-NT

Figure 5.3: Computing time versus the number n_t of test points.

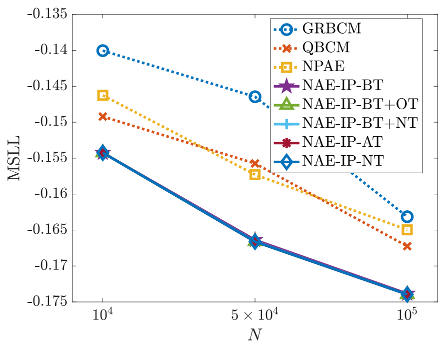
ory access patterns. Especially for NAE-IP-BT, NAE-IP-AT, and NAE-IP-NT, the computing time was shorter than QBCM. Note that we omitted the results of PoE, gPoE, BCM, and rBCM from Figs. 5.4 because the difference between those and the other methods was significant and those showed the comparable computing time with GRBCM. Figures 5.5 and 5.6 show examples with the predictive means and 95% confidence intervals in the case of $N = 10^4$, $n_t = 20$, and $n_u = 30$ except for NAE-IP-BT.

5.4.3 Real data

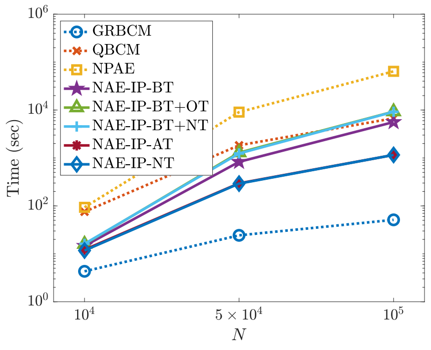
For SARCOS and POL datasets, the training data were divided respectively by constrained k-means (Bradley et al., 2000), which can avoid generating weak sub-models by setting the minimum size of clusters. We have set the minimum size to 300 for SARCOS dataset and 200 for POL dataset. For KIN8NM dataset, the training data were divided by k-means. The Matérn-5/2 function (5.2.3) was employed as the covariance function of GP. For NAE-IP-BT+NT and NAE-IP-NT, we employed the optimization of each expert's non-test points by Hensman et al. (2013) under fully independent training conditional assumption (Snelson and Ghahramani, 2005; Quiñonero-Candela and Rasmussen, 2005). We used the assumption only in the optimization of inducing points, and not in the predictions. The mini-batch size and the number of epochs were set to 100 and 10, respectively. It should be noted that the computational complexity of the optimization is $O((n_u^{(i)})^3)$, so that we can ignore the complexity as long as we set $n_u^{(i)}$ to be smaller



(a) MSE



(b) MSLL



(c) Computing time

Figure 5.4: Performance measures versus the number N of training data.

than $n^{(i)}$.

We compare the predictive performance of NAE-IP with that of the conventional methods in 10 trials by using the real datasets. We set $(p, n_t) = (8, 20)$ for KIN8NM dataset, $(72, 20)$ for SARCOS dataset, and $(25, 50)$ for POL dataset. The dimension n_u for NAE-IP except for NAE-IP-BT was set to $n_u = 1.5 \times n_t$. Table 5.5 summarizes the performance measures of the aggregation methods and the results of statistical significance testing for a difference between the best-performing method and each of the other 11 methods (Wilcoxon signed rank test with Bonferroni multiple testing correction, $P < 0.05$). For KIN8NM and POL, the extension of sketching dimensions in NAE-IP-BT+NT or NAE-IP-BT+OT improved the performance compared with that of NAE-IP-BT, and those methods achieved better performance than the other methods. On the other hand, for SARCOS, the performance of the conventional methods was the best. The fact that the performance of NAE-IP-AT and NAE-IP-NT was worse might reflect the lack of consistency of these methods under the setting of the dimension n_u.

Table 5.4: Significance of differences in performance. The methods with ⋆ mean that performance of the methods is the best. The results with † mean that differences between each of those and the method with ⋆ are statistically significant (paired t-test with Bonferroni multiple testing correction, $P < 0.05/11$). The results with ◦ mean that the null hypothesis is not rejected.

	$N = 10^4$		$N = 5 \times 10^4$		$N = 10^5$	
Method	MSE	MSLL	MSE	MSLL	MSE	MSLL
PoE	†	†	†	†	†	†
gPoE	†	†	†	†	†	†
BCM	†	†	†	†	†	†
rBCM	†	†	†	†	†	†
GRBCM	◦	†	†	†	◦	†
QBCM	◦	◦	◦	†	⋆	†
NPAE	◦	◦	◦	†	◦	†
NAE-IP-BT	⋆	⋆	◦	◦	◦	◦
NAE-IP-BT+OT	◦	◦	◦	◦	◦	◦
NAE-IP-BT+NT	◦	◦	◦	◦	◦	⋆
NAE-IP-AT	◦	◦	◦	◦	◦	◦
NAE-IP-NT	◦	◦	⋆	⋆	◦	◦

5.5 Conclusion

In this part, we have introduced the idea of linear sketching into approximate Gaussian process regression and have proposed NAE-IP with 5 options for inducing points. The options NAE-IP-BT, NAE-IP-BR+OT, and NAE-IP-BT+NT include test points as inducing points and inherit consistency at the cost of higher computational complexity. On the other hand, NAE-IP-AT and NAE-IP-NT do not have consistency but yield a significant reduction of the computational complexity. Numerical experiments using synthetic and real datasets were conducted. The experimental results show that the proposed method achieved not only the lowest predictive error but also less computing time than the conventional NPAE. Future work includes optimization of a block-structured sketching matrix $\boldsymbol{A}$.

Table 5.5: Results of the aggregation methods on KIN8NM, SARCOS, and POL datasets. The methods with ⋆ mean that performance of the methods is the best. The unmarked results mean that differences between each of those and the method with ⋆ are statistically significant (Wilcoxon signed rank test with Bonferroni multiple testing correction, $P < 0.05/11$). The results with ∘ mean that the null hypothesis is not rejected.

	KIN8NM		SARCOS		POL	
Method	MSE	MSLL	MSE	MSLL	MSE	MSLL
PoE	0.00799	−0.0635	27.1	10.4	116.	11.2
gPoE	0.00799	−0.933	27.1	3.18	116.	3.64
BCM	0.00731	−0.393	3.82	2.73	52.5	3.55
rBCM	0.00625	−0.301	2.24	2.86	22.1	3.19
GRBCM	0.00595	−1.15	1.51	**1.62⋆**	19.5	2.67
QBCM	0.00574	−1.16	1.73	1.73	15.2	2.58
NPAE	0.00540	−1.19	**1.42⋆**	1.68	13.0	2.57
NAE-IP-BT	**0.00532∘**	**−1.20∘**	1.44	1.68	12.3	2.56
NAE-IP-BT+OT	**0.00530⋆**	**−1.20⋆**	1.44	1.68	**12.2∘**	**2.55∘**
NAE-IP-BT+NT	**0.00531∘**	−1.20	1.44	1.68	**12.1⋆**	**2.55⋆**
NAE-IP-AT	0.0178	−0.650	34.4	2.90	54.0	3.27
NAE-IP-NT	0.0141	−0.756	30.2	2.71	26.7	2.85

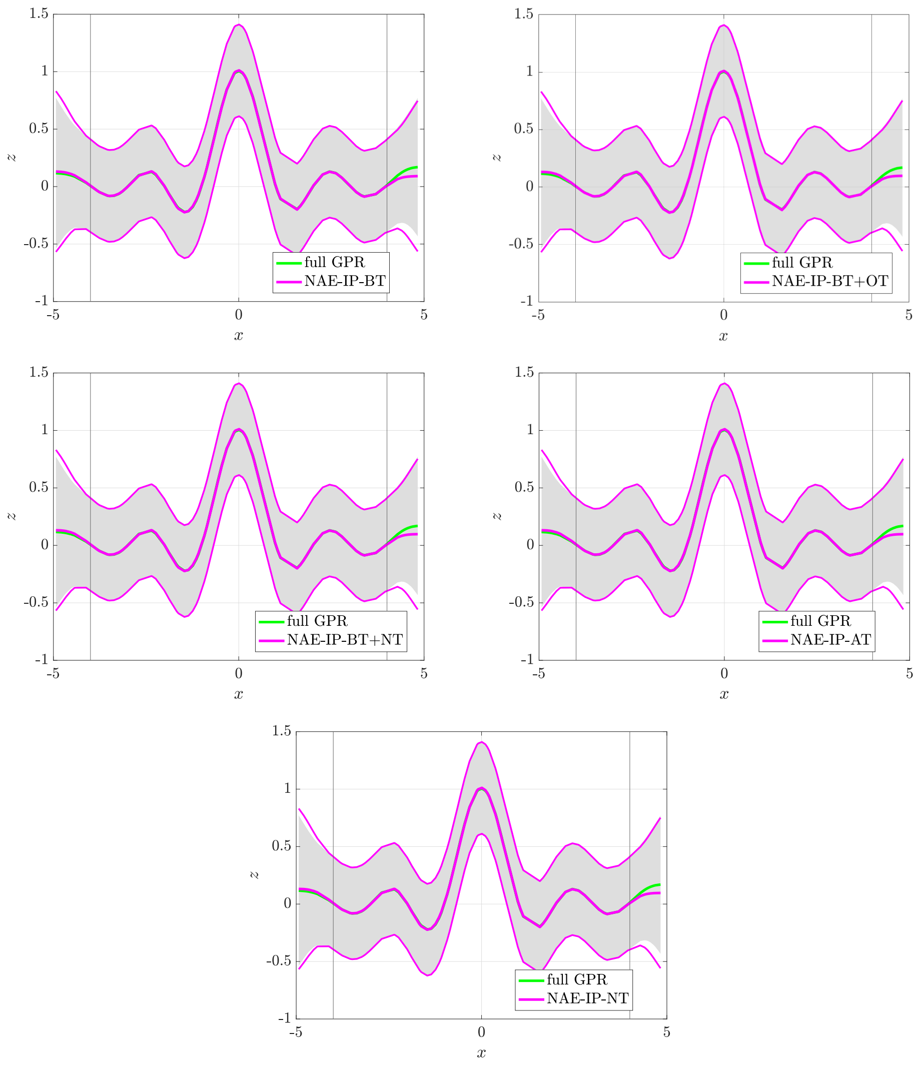


Figure 5.5: Examples: Predictive distribution of NAE-IP with the means and 95% confidence intervals in case of $N = 10^4, n_t = 20$. The dimension of NAE-IP except for NAE-IP-BT is $n_u = 1.5 \times n_t = 30$. The outside of vertical lines shows extrapolation.

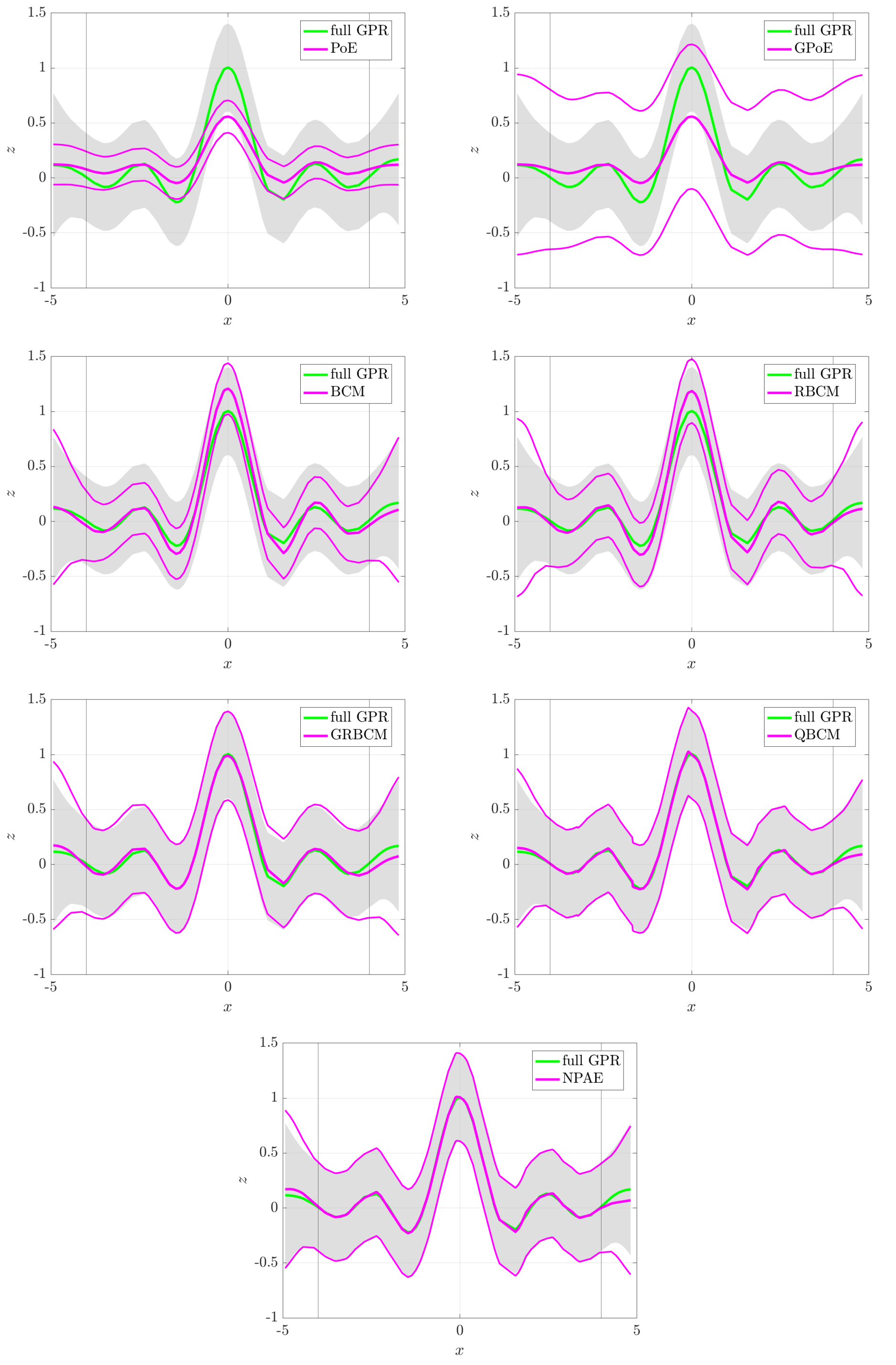


Figure 5.6: Examples: Predictive distribution of conventional aggregation methods with the means and 95% confidence intervals in case of $N = 10^4$. The outside of vertical lines shows extrapolation.

Part III

Conclusion and Future Directions

6: Conclusion

6.1 Summary

In this thesis, we have studied how to enhance performance of decentralized algorithms derived by approximate divisions of corresponding centralized processing. Specifically, we have coped with two scalable estimation problems via decentralized processing, whose subjects have been introduced in Chapter 1, namely, in-network adaptive least-mean-square (LMS) filter and Gaussian process regression (GPR). The former's target is tracking an unknown deterministic vector by using the measurements at nodes in the network obtained from the linear regression model. Diffusion LMS (D-LMS) is the fully distributed LMS algorithm in such networks that enhances scalability with respect to network size. The latter adopts a nonlinear regression model, so that it achieves more flexible estimation. However, it requires high computational complexity, and therefore aggregation methods have been proposed as approximation methods of GPR on the basis of partial decentralization.

In Part I, we have aimed to improve convergence performance of the fully distributed D-LMS and its sparsity-aware extension, sparse diffusion LMS (SD-LMS), by focusing on ways of cooperation among nodes that compensate for the performance deterioration caused by approximate divisions of the centralized problems. The cooperation method in both D-LMS and SD-LMS can be regarded as an example of average consensus protocol but the protocol is known to require a lot of iterations, especially in large networks. We have thus employed another method for the cooperation based on message propagation, i.e., consensus propagation (CP), which can be categorized into two schemes, exact CP and loopy CP. We have proposed a novel fully distributed LMS, CP-LMS, by applying exact CP, although the algorithm requires extraction of tree structure of the original network to achieve fast convergence. We have also proposed Loopy CP-LMS (LCP-LMS) by applying loopy CP to D-LMS, which can eliminate the extraction of tree structure but the required number of iteration for convergence has not been known. The special case where the iteration of loopy CP is limited to one has improved interpretability of the proposed algorithm and enabled optimization of constants involved in loopy CP. The resulting algorithm has become novel combination weights of D-LMS named (static) CP rule. We have optimized the constants in terms of the steady-state network mean-squared-error (MSD) and derived an adaptive implementation named adaptive CP rule. We have also applied the special case to SD-LMS and derived two kinds of adaptive combination weights under some assumptions. One is the adaptive CP rule that is the same as for D-LMS and another is adaptive CP with Optimization (CPO) rule. The theoretical performance has been analyzed via the framework for D-LMS and SD-LMS. Efficiency of the proposed methods has been shown via computer simulations.

In Part II, we have aimed to improve performance of the aggregation methods where covariance information of data is lost by the decentralization. Among the aggregation methods proposed so far, we have focused on one of them, nested pointwise aggregation of experts (NPAE), that uses the richest covariance information among them and generalized the prediction process via sketching. The proposed generalization has yielded a more flexible approximation of GPR than NPAE, which has been named nested aggregation of experts using inducing points (NAE-IP). The proposed NAE-IP can control the computational complexity and the performance of the algorithm by choices of the inducing points. Simulation results on synthetic and real data have shown that NAE-IP can achieve lower complexity than the original NPAE and better predictive error than the conventional methods.

This thesis is no more than having coped with the two specific problems for in-network processing and for general data processing. However, results in the thesis have shown that the improvements focusing on factors that affect performance degradation caused by decentralization enhance performance with a little increase or even decrease in complexity. As for other decentralization problems, the improvements are expected to be realized while maintaining scalability by properly identifying the parts that can affect the performance degradation and reconsidering the parts in a different context.

6.2 Future Directions

The decentralized processing that the thesis have focused on is expected to further improve the performance and to extend the range of applicable applications. This section points out the future directions, thus concluding the thesis.

Reconsideration of cost function in D-LMS

In Part I, we have derived proposed algorithms by applying CP to the combination step of D-LMS. This was motivated by the fact that the combination step has a tight relation to average consensus. One can also reconsider minimizing estimation error with a constraint for consensus among nodes. If one chose Gaussian belief propagation as a distributed solution of the problem, one may derive another distributed algorithm for the same problem settings as D-LMS. Note that it requires taking time-dependent potentials into accounts. Moreover, it may be possible to obtain interpretation of the constants β_k involved in the proposed methods in relation to weights for controlling balance between the error and the constraint.

Analysis of convergence time using non-backtracking operator

For the proposed method in Part I, we have optimized parameters in the fully distributed LMS by minimizing cost function in terms of steady-state error. On the other hand, in order to explicitly accelerate the convergence, another approach is conceivable that minimizes the convergence time of the algorithm. The convergence time for algorithms on the basis of average consensus protocol has been analyzed by using spectra of adjacency matrices but the analysis for CP has not had much progress. As discussed at the end of Sect. 2.2.3, we expect that algorithms on the basis of CP can be represented by the non-backtracking operator (Coja-Oghlan et al., 2009; Decelle et al., 2011; Krzakala et al., 2013) and that the spectrum of the operator (Bordenave et al., 2015) may enable the analysis of the convergence time. It should be a next challenge to solve an optimization problem in terms of the convergence time represented by using the spectrum of the non-backtracking operator.

Acceleration of iterative decentralized algorithms by deep unfolding

For iterative algorithms such as the centralized LMS filter, the required number of iterations for the convergence is also affected by some parameters such as step-size, the control of which largely affects the performance of the algorithms. Especially for the decentralized cases such as D-LMS, the available information is less than that for the centralized cases and the required number of iterations is generally large, so that the acceleration is an important matter. The control of the parameters of the iterative algorithms has been achieved by the aid of deep learning, which is known as deep unfolding (Gregor and LeCun, 2010; Balatsoukas-Stimming and Studer, 2019). Its application to decentralized algorithms also gathers attention recently (Kishida et al., 2020). Future directions include the extension of the deep unfolding approach to decentralized adaptive filters or online learning, where the parameters are needed to be learned in a decentralized manner as well.

Supervised optimization of sketching matrix

In Part II, we have introduced approximation methods for full GPR. Most of the aggregation methods divide covariance information $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ on the basis of the training data $\boldsymbol{X}$ and many of sparse GP methods lower the dimension of $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ on the basis of inducing points. These can be interpreted as unsupervised learning that does not take the information of test data $\boldsymbol{X}^*$ into consideration. However, Proposition 2 has shown that optimal sketching matrix depends not only on the covariance $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ of the training data but also on the covariance $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}^*)$ related to the test data. This suggests that both training and test data should be incorporated into the design of approximation methods. Therefore, it seems beneficial to utilize the test data in optimization of a block-structured sketching matrix.

Fully distributed online GPR

Fully distributed GPR is also in demand from not only computational complexity but also application point of view, for example, in environmental monitoring by multiple unmanned aerial vehicles (Gu and Hu, 2012; Choi et al., 2014; Tiwari et al., 2018). The temporal variation of the phenomena should be also taken into account (Garg et al., 2012). In addition, the algorithm that can track the phenomena is desirable, namely, online GPR (Nguyen-Tuong et al., 2008; Hoang et al., 2019). The theoretical property must be discussed as well as to manage both scalability and predictive performance.

Bibliography

Ashton SRF, Sollich P (2012) Learning curves for multi-task Gaussian process regression. In: Advances in Neural Information Processing Systems, pp 1393–1428

Bachoc F, Durrande N, Rullière D, Chevalier C (2017) Some properties of nested Kriging predictors. arXiv preprint arxiv:1707.05708

Bachoc F, Durrande N, Rullière D, Chevalier C (2021) Properties and comparison of some Kriging sub-model aggregation. arXiv preprint arxiv:1707.05708v2

Balatsoukas-Stimming A, Studer C (2019) Deep unfolding for communications systems: A survey and some new directions. In: 2019 IEEE International Workshop on Signal Processing Systems (SiPS), pp 266–271, DOI 10.1109/SiPS47522.2019.9020494

Bauer M, van der Wilk M, Rasmussen CE (2016) Understanding probabilistic sparse gaussian process approximations. In: Advances in Neural Information Processing Systems, pp 1533–1541

Bi S, Zhang R, Ding Z, Cui S (2015) Wireless communications in the era of big data. IEEE Communications Magazine 53(10):190–199, DOI 10.1109/MCOM.2015.7295483

Blondel VD, Hendrickx JM, Olshevsky A, Tsitsiklis JN (2005) Convergence in multiagent coordination, consensus, and flocking. In: Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, pp 2996–3000, DOI 10.1109/CDC.2005.1582620

Bordenave C, Lelarge M, Massoulié L (2015) Non-backtracking spectrum of random graphs: Community detection and non-regular ramanujan graphs. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp 1347–1357, DOI 10.1109/FOCS.2015.86

Bradley PS, Bennett KP, Demiriz A (2000) Constrained k-means clustering. Tech. rep., MSR-TR-2000-65, Microsoft Research, Redmond, WA

Bui M, Butelle F, Lavault C (2004) A distributed algorithm for constructing a minimum diameter spanning tree. Journal of Parallel and Distributed Computing 64(5):571–577, DOI 10.1016/j.jpdc.2004.03.009

Bui TD, Turner RE (2014) Tree-structured Gaussian process approximations. In: Advances in Neural Information Processing Systems, pp 2213–2221

Calandriello D, Carratino L, Lazaric A, Valko M, Rosasco L (2019) Gaussian process optimization with adaptive sketching: Scalable and no regret. In: Beygelzimer A, Hsu D (eds) Proceedings of the Thirty-Second Conference on Learning Theory, PMLR, Proceedings of Machine Learning Research, vol 99, pp 533–557

Candes EJ, Wakin MB, Boyd SP (2008) Enhancing sparsity by reweighted ℓ_1 minimization. Journal of Fourier Analysis and Applications 14(5):877–905, DOI 10.1007/s00041-008-9045-x

Cao Y, Fleet DJ (2014) Generalized product of experts for automatic and principled fusion of Gaussian process predictions. arXiv preprint arxiv:1410.7827

Cattivelli FS, Sayed AH (2010) Diffusion LMS strategies for distributed estimation. IEEE Transactions on Signal Processing 58(3):1035–1048, DOI 10.1109/TSP.2009.2033729

Cattivelli FS, Sayed AH (2011) Analysis of spatial and incremental LMS processing for distributed estimation. IEEE Transactions on Signal Processing 59(4):1465–1480, DOI 10.1109/TSP.2010.2100386

Cattivelli FS, Lopes CG, Sayed AH (2008) Diffusion recursive least-squares for distributed estimation over adaptive networks. IEEE Transactions on Signal Processing 56(5):1865–1877, DOI 10.1109/TSP.2007.913164

Chalupka K, Williams CK, Murray I (2013) A framework for evaluating approximation methods for Gaussian process regression. Journal of Machine Learning Research 14:330–350

Choi S, Jadaliha M, Choi J, Oh S (2014) Distributed Gaussian Process Regression Under Localization Uncertainty. Journal of Dynamic Systems, Measurement, and Control 137(3), DOI 10.1115/1.4028148, 031007

Coja-Oghlan A, Mossel E, Vilenchik D (2009) A spectral approach to analysing belief propagation for 3-colouring. Combinatorics, Probability and Computing 18(6):881–912, DOI 10.1017/S096354830900981X

Cressie NAC (1993) Statistics for Spatial Data, Revised Edition. Wiley, New York, NY, DOI 10.1002/9781119115151

Dean J, Ghemawat S (2004) Mapreduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Operating

Systems Design & Implementation - Volume 6, USENIX Association, USA, OSDI'04, pp 137–149

Decelle A, Krzakala F, Moore C, Zdeborová L (2011) Inference and phase transitions in the detection of modules in sparse networks. Phys Rev Lett 107:065701, DOI 10.1103/PhysRevLett.107.065701

Deisenroth MP, Ng JW (2015) Distributed Gaussian processes. In: Proceedings of the 32th International Conference on Machine Learning, PMLR, pp 1481–1490

Deisenroth MP, Fox D, Rasmussen CE (2015) Gaussian processes for data-efficient learning in robotics and control. IEEE Transactions on Pattern Analysis and Machine Intelligence 37(2):408–423, DOI 10.1109/TPAMI.2013.218

Dimakis AD, Sarwate AD, Wainwright MJ (2008) Geographic gossip: Efficient averaging for sensor networks. IEEE Transactions on Signal Processing 56(3):1205–1216, DOI 10.1109/TSP.2007.908946

Donoho D (2006) Compressed sensing. IEEE Transactions on Information Theory 52(4):1289–1306, DOI 10.1109/TIT.2006.871582

Duarte M, Sarvotham S, Baron D, Wakin M, Baraniuk R (2005) Distributed compressed sensing of jointly sparse signals. In: Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005., pp 1537–1541, DOI 10.1109/ACSSC.2005.1600024

Farhang-Boroujeny B (2013) Adaptive Filters: Theory and Applications, 2nd ed. John Wiley & Sons, New York, NY

Furrer R, Genton MG, Nychka D (2006) Covariance tapering for interpolation of large spatial datasets. Journal of Computational and Graphical Statistics 15(3):502–523, DOI 10.1198/106186006X132178

Garg S, Singh A, Ramos F (2012) Learning non-stationary space-time models for environmental monitoring. In: Twenty-Sixth AAAI Conference on Artificial Intelligence

Gneiting T (2002) Compactly supported correlation functions. Journal of Multivariate Analysis 83(2):493–508, DOI 10.1006/jmva.2001.2056

Gregor K, LeCun Y (2010) Learning fast approximations of sparse coding. In: Proceedings of the 27th International Conference on International Conference on Machine Learning, pp 399–406

Gu D, Hu H (2012) Spatial Gaussian process regression with mobile sensor networks. IEEE Transactions on Neural Networks and Learning Systems 23(8):1279–1290, DOI 10.1109/TNNLS.2012.2200694

Gupta A, Jha RK (2015) A survey of 5G network: Architecture and emerging technologies. IEEE Access 3:1206–1232, DOI 10.1109/ACCESS.2015.2461602

Hara S, Yomo H, Popovski P, Hayashi K (2006) New paradigms in wireless communication systems. Wireless Personal Communications 37(3–4):233–241, DOI 10.1007/s11277-006-9036-7

Harrane IEK, Flamary R, Richard C (2019) On reducing the communication cost of the diffusion LMS algorithm. IEEE Trans Signal Inf Process Over Netw 5(1):100–112, DOI 10.1109/TSIPN.2018.2863218

Hayakawa R, Nakai A, Hayashi K (2018) Distributed approximate message passing with summation propagation. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 4104–4108, DOI 10.1109/ICASSP.2018.8462333

He J, Qi J, Ramamohanarao K (2019) Query-aware Bayesian committee machine for scalable Gaussian process regression. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp 208–216

Hensman J, Fusi N, Lawrence ND (2013) Gaussian processes for big data. In: Proceedings of the 29th Conference on Uncertainly in Artificial Intelligence, pp 282–290

Herzen J, Westphal C, Thiran P (2011) Scalable routing easy as PIE: A practical isometric embedding protocol. In: 2011 19th IEEE International Conference on Network Protocols, pp 49–58, DOI 10.1109/ICNP.2011.6089081

Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Computation 14(8):1771–1800, DOI 10.1162/089976602760128018

Hoang TN, Hoang QM, Low KH, How J (2019) Collective online learning of gaussian processes in massive multi-agent systems. Proceedings of the AAAI Conference on Artificial Intelligence 33(01):7850–7857, DOI 10.1609/aaai.v33i01.33017850

Hu W, Zhan F (2016) Sparse diffusion apa for distributed estimation. In: 2016 IEEE 13th International Conference on Signal Processing (ICSP), pp 253–256, DOI 10.1109/ICSP.2016.7877835

Kambatla K, Kollias G, Kumar V, Grama A (2014) Trends in big data analytics. Journal of Parallel and Distributed Computing 74(7):2561–2573, DOI 10.1016/j.jpdc.2014.01.003, special Issue on Perspectives on Parallel and Distributed Processing

Keerthi SS, Chu W (2005) A matching pursuit approach to sparse Gaussian process regression. In: Proceedings of the 18th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, NIPS'05, pp 643–650

Kishida M, Ogura M, Yoshida Y, Wadayama T (2020) Deep learning-based average consensus. IEEE Access 8:142404–142412, DOI 10.1109/ACCESS.2020.3014148

Konečnỳ J, McMahan HB, Ramage D, Richtárik P (2016) Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:161002527

Kong JT, Lee JW, Kim SE, Song WJ (2017) Diffusion LMS algorithms with multi combination for distributed estimation: Formulation and performance analysis. Digital Signal Processing 71:117–130, DOI 10.1016/j.dsp.2017.09.004

Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborová L, Zhang P (2013) Spectral redemption in clustering sparse networks. Proceedings of the National Academy of Sciences 110(52):20935–20940, DOI 10.1073/pnas.1312486110

Lawrence N (2005) Probabilistic non-linear principal component analysis with Gaussian process latent variable models. Journal of Machine Learning Research 6:1783–1816

Lawrence N, Seeger M, Herbrich R (2003) Fast sparse gaussian process methods: The informative vector machine. In: Proceedings of the 16th Annual Conference on Neural Information Processing Systems, pp 609–616

Li H, Ota K, Dong M (2018) Learning iot in edge: Deep learning for the internet of things with edge computing. IEEE Network 32(1):96–101, DOI 10.1109/MNET.2018.1700202

Liberty E (2013) Simple and deterministic matrix sketching. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 581–588

Liu H, Cai J, Wang Y, Ong YS (2018) Generalized robust Bayesian committee machine for large-scale Gaussian process regression. In: Proceedings of the 35th International Conference on Machine Learning, PMLR, pp 3131–3140

Liu H, Ong YS, Shen X, Cai J (2020a) When Gaussian process meets big data: A review of scalable GPs. IEEE Transactions on Neural Networks and Learning Systems 31(11):4405–4423, DOI 10.1109/TNNLS.2019.2957109

Liu Y, Bi S, Shi Z, Hanzo L (2020b) When machine learning meets big data: A wireless communication perspective. IEEE Vehicular Technology Magazine 15(1):63–72, DOI 10.1109/MVT.2019.2953857

Lopes CG, Sayed AH (2007) Incremental adaptive strategies over distributed networks. IEEE Transactions on Signal Processing 55(8):4064–4077, DOI 10.1109/TSP.2007.896034

Lopes CG, Sayed AH (2008) Diffusion least-mean-squares over adaptive networks: Formulation and performance analysis. IEEE Transactions on Signal Processing 56(7):3122–3136, DOI 10.1109/TSP.2008.917383

Lorenzo P, Sayed AH (2013) Sparse distributed learning based on diffusion adaptation. IEEE Transactions on Signal Processing 61(6):1419–1433, DOI 10.1109/TSP.2012.2232663

Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein J (2010) Graphlab: A new framework for parallel machine learning. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI2010), pp 340–349

Mandic DP, Kanna S, Constantinides AG (2015) On the intrinsic relationship between the least mean square and Kalman filters [lecture notes]. IEEE Signal Processing Magazine 32(6):117–122, DOI 10.1109/MSP.2015.2461733

Melkumyan A, Ramos F (2009) A sparse covariance function for exact gaussian process inference in large datasets. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'09, pp 1936–1942

Moallemi CC, Roy BV (2006) Consensus propagation. IEEE Transactions on Information Theory 52(11):4753–4766, DOI 10.1109/TIT.2006.883539

Nakai A, Hayashi K (2017) Diffusion LMS using consensus propagation. In: 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp 943–948, DOI 10.1109/APSIPA.2017.8282158

Nakai A, Hayashi K (2018) An adaptive combination rule for diffusion lms based on consensus propagation. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 3839–3843, DOI 10.1109/ICASSP.2018.8462277

Nakai-Kasai A, Hayashi K (2019) Diffusion LMS based on message passing algorithm. IEEE Access 7:47022–47033, DOI 10.1109/ACCESS.2019.2909775

Nakai-Kasai A, Hayashi K (2020) Optimal combination weight for sparse diffusion least-mean-square based on consensus propagation. In: 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, pp 228–235

Nakai-Kasai A, Hayashi K (2021) An acceleration method of sparse diffusion LMS based on message propagation. IEICE Transactions on Communications E104-B(2):141–148, DOI 10.1587/transcom.2020EBT0001

Nakai-Kasai A, Tanaka T (2021) Nested aggregation of experts using inducing points for approximated Gaussian process regression. Machine Learning (accepted for publication)

Nassif R, Richard C, Ferrari A, Sayed AH (2017) Diffusion LMS for multitask problems with local linear equality constraints. IEEE Transactions on Signal Processing 65(19):4979–4993, DOI 10.1109/TSP.2017.2721930

Nguyen-Tuong D, Peters J, Seeger M (2008) Local gaussian process regression for real time online model learning and control. In: Proceedings of the 21st International Conference on Neural Information Processing Systems, pp 1193–1200

Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 95(1):215–233, DOI 10.1109/JPROC.2006.887293

Pearl J (1988) Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann Publishers Inc.

Quiñonero-Candela J, Rasmussen CE (2005) A unifying view of sparse approximate Gaussian process regression. Journal of Machine Learning Research 6:1939–1959

Rasmussen CE, Williams CKI (2006) Gaussian Process for Machine Learning. The MIT Press, Cambridge, MA

Rullière D, Durrande N, Bachoc F, Chevalier C (2018) Nested Kriging predictions for datasets with a large number of observations. Statistics and Computing 28(4):849–867, DOI 10.1007/s11222-017-9766-2

Savazzi S, Nicoli M, Rampa V (2020) Federated learning with cooperating devices: A consensus approach for massive iot networks. IEEE Internet of Things Journal 7(5):4641–4654, DOI 10.1109/JIOT.2020.2964162

Sayed AH (2011) Adaptive filters. John Wiley & Sons

Sayed AH (2014) Chapter 9 - diffusion adaptation over networks*the work was supported in part by nsf grants eecs-060126, eecs-0725441, ccf-0942936, and ccf-1011918*. In: Zoubir AM, Viberg M, Chellappa R, Theodoridis S (eds) Academic Press Library in Signal Processing: Volume 3, Academic Press Library in Signal Processing, vol 3, Elsevier, pp 323–453, DOI 10.1016/B978-0-12-411597-2.00009-6

Sayed AH, Lopes CG (2007) Adaptive processing over distributed networks. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E90-A(8):1504–1510, DOI 10.1093/ietfec/e90-a.8.1504

Sayed AH, Tu S, Chen J, Zhao X, Tpwfic ZJ (2013) Diffusion strategies for adaptation and learning over networks. IEEE Signal Processing Magazine 30(3):155–171, DOI 10.1109/MSP.2012.2231991

Scherber DS, Papadopoulos HC (2004) Locally constructed algorithms for distributed computations in ad-hoc networks. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, Association for Computing Machinery, New York, NY, USA, IPSN '04, pp 11–19, DOI 10.1145/984622.984625

Schizas ID, Mateos G, Giannakis GB (2009) Distributed LMS for consensus-based in-network adaptive processing. IEEE Transactions on Signal Processing 57(6):2365–2382, DOI 10.1109/TSP.2009.2016226

Shalev-Shwartz S (2012) Online learning and online convex optimization. Foundations and Trends in Machine Learning 4(2):107–194, DOI 10.1561/2200000018

Snelson E, Ghahramani Z (2005) Sparse Gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems, pp 1257–1264

Stein ML (2012) Interpolation of Spatial Data: Some Theory for Kriging. Springer Science & Business Media, New York, NY, DOI 10.1007/978-1-4612-1494-6

Takahashi N, Yamada I, Sayed AH (2010) Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis. IEEE Transactions on Signal Processing 58(9):4795–4810, DOI 10.1109/TSP.2010.2051429

Tavassolipour M, Motahari SA, Shalmani MTM (2020) Learning of Gaussian processes in distributed and communication limited systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 42(8):1928–1941, DOI 10.1109/TPAMI.2019.2906207

Tibshirani R (1996) Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) 58(1):267–288, DOI 10.1111/j.2517-6161.1996.tb02080.x

Tiwari K, Jeong S, Chong NY (2018) Point-wise fusion of distributed gaussian process experts (fudge) using a fully decentralized robot team operating in communication-devoid environment. IEEE Transactions on Robotics 34(3):820–828, DOI 10.1109/TRO.2018.2794535

Tresp V (2000) A Bayesian committee machine. Neural Computation 12(11):2719–2741, DOI 10.1162/089976600300014908

Tu SY, Sayed AH (2011) Optimal combination rules for adaptation and learning over networks. In: 2011 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), San Juan, Puerto Rico, pp 317–320, DOI 10.1109/CAMSAP.2011.6136014

van der Vaart A, van Zanten H (2011) Information rates of nonparametric Gaussian process methods. Journal of Machine Learning Research 12:2095–2119

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2013) Openml: Networked science in machine learning. SIGKDD Explorations 15(2):49–60, DOI 10.1145/2641190.2641198

Wang S, Dekorsy A (2019) Distributed consensus-based extended Kalman filtering: A Bayesian perspective. In: 2019 27th European Signal Processing Conference (EUSIPCO), pp 1–5, DOI 10.23919/EUSIPCO.2019.8902553

Wee WM, Yamada I (2013) A proximal splitting approach to regularized distributed adaptive estimation in diffusion networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp 5420–5424, DOI 10.1109/ICASSP.2013.6638699

Wendland H (2004) Scattered data approximation, vol 17. Cambridge university press

Wilson A, Nickisch H (2015) Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In: Proceedings of the 32nd International Conference on Machine Learning, PMLR, pp 1775–1784

Woodruff DP (2014) Sketching as a tool for numerical linear algebra. Foundations and Trends in Theoretical Computer Science 10(1–2):1–157, DOI 10.1561/0400000060

Wu Z (1995) Compactly supported positive definite radial functions. Advances in Computational Mathematics 4(1):283–292, DOI 10.1007/BF03177517

Xiao L, Boyd S (2004) Fast linear iterations for distributed averaging. Systems & Control Letters 53(1):65–78, DOI 10.1016/j.sysconle.2004.02.022

Yuan C, Neubauer C (2009) Variational mixture of gaussian process experts. In: Advances in Neural Information Processing Systems, pp 1897–1904

Zhang MM, Williamson SA (2019) Embarrassingly parallel inference for gaussian processes. Journal of Machine Learning Research 20:1–26

Zhang Z, Xiao Y, Ma Z, Xiao M, Ding Z, Lei X, Karagiannidis GK, Fan P (2019) 6g wireless networks: Vision, requirements, architecture, and key technologies. IEEE Vehicular Technology Magazine 14(3):28–41, DOI 10.1109/MVT.2019.2921208

Zhao X, Sayed AH (2012) Performance limits for distributed estimation over LMS adaptive networks. IEEE Transactions on Signal Processing 60(10):5107–5124, DOI 10.1109/TSP.2012.2204985

Zhao X, Tu SY, Sayed AH (2012) Diffusion adaptation over networks under imperfect information exchange and non-stationary data. IEEE Transactions on Signal Processing 60(7):3460–3475, DOI 10.1109/TSP.2012.2192928