EFFICIENT LEARNING STRATEGIES OVER DISTRIBUTED NETWORKS FOR BIG DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Osman Fatih KILIÇ
July, 2017

Efficient Learning Strategies over Distributed Networks for Big Data
By Osman Fatih KILIÇ
July, 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Sinan Gezici

Çağatay Candan

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

EFFICIENT LEARNING STRATEGIES OVER DISTRIBUTED NETWORKS FOR BIG DATA

Osman Fatih KILIÇ
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
July, 2017

We study the problem of online learning over a distributed network, where agents in the network collaboratively estimate an underlying parameter of interest using noisy observations. For the applicability of such systems, sustaining a communication and computation efficiency while providing a comparable performance plays a crucial role. To this end, in this work, we propose computation and communication wise highly efficient distributed online learning methods that present superior performance compared to the state-of-the-art. In the first part of the thesis, we study distributed centralized estimation schemes, where such approaches require high communication bandwidth and high computational load. We introduce a novel approach based on set-membership filtering to reduce such burdens of the system. In the second part of our work, we study distributed decentralized estimation schemes, where nodes in the network individually and collaboratively estimate a dynamically evolving parameter using noisy observations. We present an optimal decentralized learning algorithm through disclosure of local estimates and prove that optimal estimation in such systems is possible only over certain network topologies. We then derive an iterative algorithm to recursively construct the optimal combination weights and the estimation. Through series of simulations over generated and real-life benchmark data, we demonstrate the superior performance of the proposed methods compared to state-of-the-art distributed learning methods. We show that the introduced algorithms provide improved learning rates and lower steady-state error levels while requiring much less communication and computation load on the system.

Keywords: Distributed estimation, adaptive networks, efficient learning, centralized estimation, decentralized estimation.

ÖZET

BÜYÜK VERİLER İÇİN DAĞINIK AĞLARDA ETKİLİ ÖĞRENME TEKNİKLERİ

Osman Fatih KILIÇ
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
Temmuz 2017

Dağınık ağlarda çevrimiçi öğrenme teknikleri üzerinde çalışmaktayız. Dağınık ağlarda iletişim ve işleme kapasitesine sahip ajanlar ağ üzerinde birlikte çalışarak gürültülü gözlemler üzerinden altta yatan bir parametreyi kestirmeye çalışırlar. Bu sistemlerin uygulanabilirliği için yüksek kestirim performansı sağlanırken, etkili iletişim ve işleme kabiliyetlerine sahip olmak da esastır. Bu bağlamda, tezimizde düşük iletişim ve işleme ağırlığı ile literatüre göre yüksek performans sağlayan dağınık çevrimiçi öğrenme metodları sunmaktayız. Tezin ilk kısmında yüksek iletişim ağırlığı gerektiren merkezi dağınık ağlar için küme üyeliği tabanlı yeni bir yöntem sunarak, bu tarz ağların gerektirdiği iletişim ve işlem ağırlığını azaltarak, literatürde bulunan benzer algoritmalara göre çok daha üstün bir performans elde etmekteyiz. Tezin ikinci kısmında ise dinamik parametre kestirimi için merkezi olmayan dağınık öğrenme teknikleri üzerinde çalışmaktayız. Bu ağlarda her ajan kendi kestirimini gözlemlerine ve etraftan aldıkları bilgilere göre oluşturmaktadır. Bu bağlamda biz merkezi olmayan ağlarda optimum öğrenmenin gerçekleşmesi için gerekli şartları ortaya koymakta ve sadece kestirimlerin paylaşılması ile optimum öğrenim performansına ulaşmanın sadece belirli ağ topolojilerinde mümkün olduğunu göstermekteyiz. Daha sonra bu çıkarımlar üzerinden optimum öğrenme performansına erişecek yinelemeli algoritmayı ortaya koyarak gerekli birleşim ağırlıklarını ve kestirimi gerçekleştirecek çıkarımları yapmaktayız. Tez sırasında, yaratılmış ve reel datalar üzerinde gerçekleştirdiğimiz simülasyonlar ile ortaya koyduğumuz bu yöntemlerin hem öğrenme hızı olarak hem de nihai hata seviyeleri olarak literatürde bulunan yöntemlere göre çok daha üstün bir performans gösterdiklerini sergilemekteyiz.

Anahtar sözcükler: Dağıtık kestirim, uyarlanır ağlar, etkili öğrenme, merkezi kestirim, merkezi olmayan kestirim.

Acknowledgement

I would like to thank Assoc. Prof. Dr. Suleyman S. Kozat whose vision and insight on the contemporary signal processing research made this thesis possible. I also would like to thank for his fruitful comments and feedback that he gave during our study.

I acknowledge that this work is supported by TUBITAK BIDEB 2210A Scholarship Programme.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, due to the advancements in information technologies, distributed learning and estimation has attracted significant attention thanks to their high convergence and robustness behaviors over fast streaming data [1, 2, 3, 4, 5]. In a distributed learning framework, we consider a network of agents observing a temporal signal about an underlying state, possibly coming from different spatial sources with different statistics. Each agent in the network is equipped with communication and processing capabilities. The aim of each agent is to estimate the underlying parameter of interest using its observations by solving an optimization problem of minimizing the expected Euclidean distance between the estimation and the true value of the state, i.e. minimum mean-square estimation (MMSE). However, agents in the network are connected to a set of neighboring nodes and can exchange information, i.e. observations and/or estimations, to augment the learning process. As an example, assume a network of emission sensors distributed over a greenhouse to monitor the CO_2 levels for precision agriculture applications [6]. Since the agents would collect different observations from different parts of the area, they can cooperate in the network to promptly learn the true CO_2 levels for an enhanced intervention.

There exists an extensive research on distributed learning for estimating a fixed or a dynamic underlying parameter, which mainly goes under centralized and

decentralized distributed learning frameworks [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. In the centralized framework, all the agents in the network are connected to a fusion center and each agent sends their information to this center for constructing a final estimate[7]. Distributed agents may have internal processing unit and send their local estimations along with their observations to fusion center or they can just send the peripheral observations if they are not equipped with processing power. After collecting all the information, fusion center then construct the final estimate of the system.

For certain learning and adaptive filtering scenarios, we can select an appropriate adaptation algorithm with its parameters, e.g., the length of the filter or the learning rate, based on the *a priori* knowledge about the structure and statistics of the data model [17, 18]. However, the performance of the algorithm might degrade severely due to the improper design in the lack of *a priori* information. As an example, conventional learning algorithms, e.g., the least mean square (LMS) algorithm, in general demonstrate degraded performance in the impulsive noise environment while the algorithms robust against impulsive interferences, e.g., the sign algorithm (SA), achieve inferior performance over the conventional algorithms in the impulse-free noise environments [19]. On the other hand, in the centralized distributed estimation framework, combining various adaptive filters with different configurations running on distributed nodes through a fusion center achieve better performance than any of the single filters on the peripheral nodes [17, 20, 21, 22, 23, 24, 25]. Particularly, through this approach, we can achieve enhanced performance in a wider range of learning and estimation applications.

In the centralized distributed estimation method, the fusion center combines the various information coming from the distributed nodes such that the final output of the system better estimates the underlying parameter [7]. As the combination weights on the fusion center could be fixed with hindsight about the temporal data, we can also adapt those combination weights sequentially based on the observed data and the information coming from the individual nodes. Since all the information is collected on a single node, solving the global optimization problem regarding MSE performance is therefore trivial in centralized systems. However, it has serious disadvantages regarding communication and

computation load on the network [7, 11]. Especially the fusion center requires a high communication bandwidth and processing power if the network becomes too large. Hence, these approaches cannot be used for applications involving big data due to their impractical computational and communication need.

To this end, in the first part of this thesis, we introduce a centralized learning approach using the SMF in order to reduce communication and computational load and achieve improved performance. In the conventional least squares algorithms, e.g., the LMS algorithm (or the stochastic gradient descent algorithm), we minimize a cost function of the error term defined as the difference between the desired and the estimated signals. On the contrary, the set membership filtering approach seeks to find any parameter yielding smaller error terms than a predefined bound. SMF approach achieves relatively fast convergence performance in addition to the reduced communication and computation load since we do not update the parameter unless we obtain larger error than the bound and nodes do not need to send information to fusion center unless an update occurs [26, 27]. Later in the thesis, through series of simulations over generated and real-life benchmark data, we show the superior performance of the proposed method over state-of-the-art filtering algorithms in the centralized estimation framework regarding both convergence and steady-state performance. We also show that the proposed algorithm significantly reduces computation and communication load compared to the conventional estimation schemes.

In the alternative decentralized framework, each agent in the network has a different set of neighboring nodes consisting of spatially close ones and exchange information only with them to overcome the former problems with the centralized framework [15, 16, 14, 8, 9, 10, 11, 12]. In these approaches, agents only disclose their local estimations about the underlying event and combine the received information to produce final estimates. This way, the information efficiently propagates through the network to improve the overall performance. In consensus decentralized framework, after processing and collecting information, nodes reach to a consensus about their estimate either immediately or iteratively through time [15, 16].

In the original implementation of consensus algorithms two time scale is being used among nodes; one time scale is for collecting the measurements and the other one is to iterate over the collected data sufficiently enough to reach an agreement among nodes [28, 29, 30]. Nonetheless, the use of two time scale prevents such system from working in real-time over online-streaming data [12]. In the implementation of iterative consensus algorithms, the learning rate on the individual nodes reduces with time as the nodes in the network reaches to a consensus about the data, which enables such systems to use only one time scale and iteratively reach to a consensus over time [31, 2, 32]. However, due to this decay in the learning rate, as the system reaches to a consensus, the network stops adapting and becomes inefficient against estimation of dynamic events [12].

In [8, 11] and [14], authors present different solutions to the decentralized distributed estimation problem by introducing the diffusion strategy to the framework. The motivation behind the strategy is to develop a distributed estimation scheme that is able to promptly respond to the online-streaming data in real-time. The nodes in diffusion strategy combine their local estimate with the information coming from the neighboring agent within the same time scale so that the overall network can rapidly adapt itself to the temporal and spatial variations in the data statistics. However, these methods do not consider the network topology, information path, and disclosure to obtain a globally optimal solution. In [33, 34], distributed incremental solutions are presented to reach the optimal estimate by defining a certain path through the network,i.e. cyclic path among nodes, which may not be practical against fast streaming data or dynamic configurations.

In [35], authors presented a novel approach to obtain an optimal estimation of a fixed underlying parameter by exploiting the network structure and optimal information disclosure and combination without any incremental and path requirements. We note that it is more realistic to model the underlying parameter as it is subject to change over time, i.e. non-stationary sources. Although there exists a literature on distributed dynamic parameter estimation, these algorithms again do not consider reaching a globally optimal solution [3, 36, 37].

To this end, in the second part of our study, we extend the work of [35]

for optimal estimation of dynamic parameters over distributed networks. Since the underlying event is evolving with time and information is only propagating through disclosure of local estimates, it requires a different approach than the existing methods. We first use the framework of [35] to establish the model and the problem. Then, we introduce efficient and optimal distributed learning (EODL) algorithm for dynamic parameters and prove that it is only applicable over certain network topologies. Later in the thesis, we also show the superior performance of the proposed method compared to the literature through numerical examples.

We organize the rest of the thesis as follows. In Chapter 2, we introduce communication and computation wise highly efficient distributed estimation framework. We introduce efficient learning algorithms for centralized estimation over distributed networks. In Chapter 3, we propose efficient and optimal learning algorithm for dynamic parameters over distributed networks. We present concluding remarks in Chapter 4.

Notation: Through this paper, bold lower case letters denote column vectors and bold upper case letter denote matrices. For a vector $\boldsymbol{a}$ (or matrix $\boldsymbol{A}$), $\boldsymbol{a}^T$ (or $\boldsymbol{A}^T$) is its ordinary transpose. The operator $\text{col}\{\cdot\}$ produces a column vector or a matrix in which the arguments of $\text{col}\{\cdot\}$ are stacked one under the other. For a given vector $\boldsymbol{w}$, $\boldsymbol{w}^{(i)}$ denotes the ith individual entry of $\boldsymbol{w}$. Similarly for a given matrix $\boldsymbol{G}$, $\boldsymbol{G}^{(i)}$ is the ith row of $\boldsymbol{G}$. For a vector argument, $\text{diag}\{\cdot\}$ creates a diagonal matrix whose diagonal entries are elements of the associated vector. In addition, all random variables are presented as uppercase calligraphic letters, i.e. $\mathcal{X}$ and all realizations of these variables are presented as their lowercase characters, i.e. x.

Chapter 2

Communication and Computation wise Highly Efficient Distributed Learning

In this part of the thesis, we study distributed estimation over centralized networks. We introduce communication and computation wise highly efficient distributed learning algorithm based on set-membership filtering. We deploy set-membership filters on peripheral nodes to reduce the communication load and also we construct adaptive combination weights on the fusion center again based on set-membership filtering to reduce the computational load on the network. Through series of simulation, we demonstrate the superior performance of the introduced algorithm over the state-of-the-art regarding transient and steady-state performance.

2.1 Centralized Distributed Estimation Framework

Considering an on-line setting where only the current feature vector $\mathbf{x}(t)$ at time $t \geq 1$ is available for corresponding data $d(t)$. Our aim is to sequentially estimate $d(t)$ that is produced by an underlying parameter $\mathbf{w}_o$ and corresponding feature vector $\mathbf{x}(t)$ such that

$$d(t) = \mathbf{x}(t)^T \mathbf{w}_o$$

through a function

$$\hat{d}(t) = f(\mathbf{x}(t))$$

and for the estimation, in this work we use centralized distributed network. In the centralized network, the system consist of two parts as presented in Fig.2.1. The first part have m peripheral nodes running an individual learning algorithm to estimate their corresponding desired signal $d_i(t)$ for $i = 1, \cdots, m$. Each filter with their parameter vector $\mathbf{w}_i(t)$, $i = 1, \cdots, m$ and input vector $\mathbf{x}(t)$ produce an estimate $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$ and in next step we update their parameter vector according to their estimation error

$$e_i(t) \triangleq d_i(t) - \hat{d}_i(t).$$

In the second part of the system, nodes then send their estimate regarding the desired signal to the fusion center, where we combine the information coming from the peripheral nodes and obtain the final estimate of the system such that

$$\hat{d}(t) = \mathbf{w}^T(t)\mathbf{y}(t),$$

where $\mathbf{y}(t) = col\{\hat{d}_1(t), \cdots, \hat{d}_m(t)\}$ and $\mathbf{w}(t) = col\{w^{(1)}(t), \cdots, w^{(m)}(t)\}$ is the combination weights vector. Linear combination parameters of this stage is updated adaptively according to the final estimation error

$$e(t) \triangleq d(t) - \hat{d}(t).$$

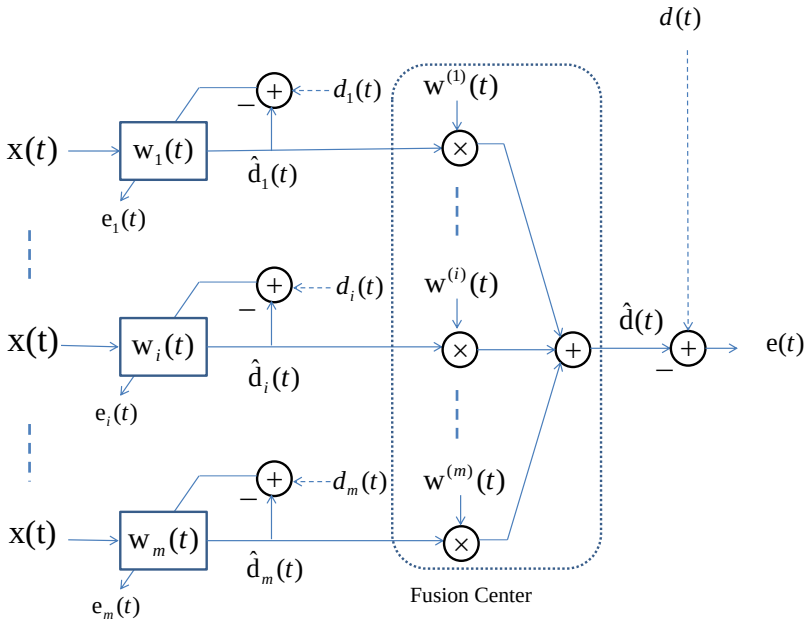

Figure 2.1: Centralized distributed estimation structure.

The use of conventional least squares algorithms such as least mean square algorithm in these centralized systems result in an update of parameter vectors at each step and sending them to the fusion center. This notion is not advantageous for most big data applications due to high computation and communication load that this feature will create. Therefore, as a solution, we employ set membership filters on peripheral nodes and their combination at fusion center for this structure.

In subsequent sections, we first introduce the structure of the set membership filters (SMF), then we introduce methods for constructing adaptive combination weights for the central node.

2.2 Structure of Set-Membership Filters

For the general linear-in-parameter filters whose input is $\mathbf{x} \in \mathbb{R}^n$, the desired output is real scalar d and the output of the filter is $\hat{d} = \mathbf{x}^T\mathbf{w}$ where $\mathbf{w} \in \mathbb{R}^n$ is the

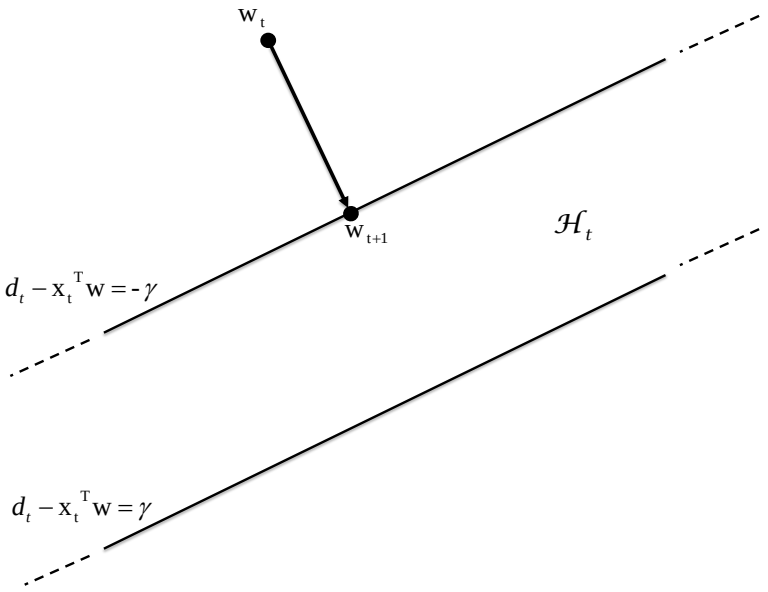

Figure 2.2: Parameter vector update by projection onto constraint set.

parameter vector for the filter, the filter error is defined as $e(\mathbf{w}) = d - \hat{d}$. In the general setting, filter estimates the parameter vector to minimize the cost which is a function of the filter error [18]. However, in the set membership filtering scheme, we update the parameter vector to satisfy a predefined upper bound γ on the filter error for all data pairs $(d, \mathbf{x})$ in a model space $\mathcal{S}$ such that

$$|e(\mathbf{w})|^2 \leq \gamma, \ \forall (d, x) \in \mathcal{S}. \tag{2.1}$$

Therefore any parameter vector satisfying (2.1) is an acceptable solution and the set of these solutions forms the feasibility set which is defined as

$$\Gamma \triangleq \bigcap_{(d,\mathbf{x})\in\mathcal{S}} \{\mathbf{w} \in \mathbb{R}^n : |d - \mathbf{x}^T\mathbf{w}|^2 \leq \gamma^2\}. \tag{2.2}$$

If the model space $\mathcal{S}$ is known priorly, then it is possible to estimate the feasibility set or a parameter vector in it. However, there is no closed form solution for an arbitrary $\mathcal{S}$ and in practice the model space is not known completely or it is time-varying [26, 27]. Therefore we estimate the feasibility set or one of its members using set-membership adaptive recursive techniques (SMART).

Considering a practical case, where only measured data pair $(d(t), \mathbf{x}(t)) \in \mathcal{S}$ is available, the constraint set $\mathcal{H}_t$ containing all parameter vectors satisfying (2.1)

is defined as

$$\mathcal{H}(t) \triangleq \{\mathbf{w} \in \mathbb{R}^n : |d(t) - \mathbf{w}^T\mathbf{x}(t)| \leq \gamma\}. \tag{2.3}$$

Here the constraint set is a region enclosed by the parallel hyperplanes defined with

$$|d(t) - \mathbf{x}(t)^T\mathbf{w}| = \gamma$$

and an estimate for the feasibility set at time t is membership set $\phi_t \triangleq \bigcap_{\tau=1}^{t} \mathcal{H}(\tau)$. We approximate the membership set for tractable and computable results by projecting current parameter vector $\mathbf{w}(t)$ onto constraint set $\mathcal{H}(t)$ if it is not contained in it and assure an error upper bound of γ [26] as we present in Fig.2.2.

We express the problem defined above as

$$\mathbf{w}(t+1) = \arg \min_{\mathbf{w} \in \mathcal{H}(t+1)} \|\mathbf{w} - \mathbf{w}(t)\|^2. \tag{2.4}$$

We solve the optimization problem with constraint in (2.4) with the method of Lagrange multipliers. The Lagrangian equation to the optimization problem in (2.4) is

$$\mathcal{L}(\mathbf{w}, \tau) = \|\mathbf{w} - \mathbf{w}(t)\|^2 + \tau(|e(t)| - \gamma). \tag{2.5}$$

Solution to the Lagrangian in (2.5) is

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{x}(t)e(t)}{\mathbf{x}^T(t)\mathbf{x}(t)} \tag{2.6}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\gamma}{|e(t)|} & \text{if } |e(t)| > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

The resulting algorithm in (2.6) is named as set membership normalized least mean square algorithm (SM-NLMS) and achieves better convergence speed and steady-state MSE with reduced computational load than NLMS algorithm [26]. We present a detailed explanation of the algorithm in Algorithm 1.

In the next section, we present several methods for adaptively constructing combination weights at the fusion center based on set-membership filtering in order to reduce computational load on the central node.

Algorithm 1 The Set-Membership NLMS Algorithm

Choose γ
$\mathbf{w}(0) \leftarrow Initialize$
$\alpha \leftarrow Constant$
for all $t \geq 0$ **do**
  $\hat{d}(t) = \mathbf{x}^T(t)\mathbf{w}(t)$
  $e(t) = d(t) - \hat{d}(t)$
  if $|e(t)| > \gamma$ **then**
    $\mu(t) = 1 - \frac{\gamma}{|e(t)|}$
    $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{x}(t)e(t)}{\alpha + \mathbf{x}^T(t)\mathbf{x}(t)}$
  end if
end for

2.3 Adaptive Combination Weights

We deploy SMF scheme at the fusion center for performing an adaptive combination of peripheral set-membership filters with different error bounds running on distributed nodes to estimate their corresponding desired signal $d_i(t)$. We emphasize that using SMF scheme provides lower computational complexity which offers a comparable performance suitable for big data applications than standard LMS algorithms. Also, we get benefits of fast converging and lower steady state MSE performance obtained by using different bounds on peripheral nodes.

We use a system where m SMF filter running on distributed nodes as in Fig.2.1, each one updates their parameter vector $\mathbf{w}_i(t) \in \mathbb{R}^n$ and produces estimate $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$ with respect to its bound γ_i. On the fusion center, we combine information coming from each node linearly through time variant weight vector $w(t)^{(i)} \in \mathbb{R}^m$ which is trained with combination SMF filter with bound $\bar{\gamma}$. We denote input to the combination stage as

$$\mathbf{y}(t) \triangleq col\{\hat{d}_1(t), ..., \hat{d}_m(t)\}$$

and the parameter vector of the combination stage is

$$\mathbf{w}(t) \triangleq col\{w^{(1)}(t), ..., w^m(t)\}.$$

The output of the combination stage is

$$\hat{d}(t) = \mathbf{y}^T(t)\mathbf{w}(t)$$

and the final estimation error is

$$e(t) \triangleq d_t - \hat{d}(t).$$

In the following subsections, we seek and train parameter vectors for the combination weights satisfying upper bound $\bar{\gamma}$ within different parameter spaces.

2.3.1 Unconstrained Adaptive Combination Weights

The first parameter space is for the unconstrained linear combination weights and defined as

$$\mathcal{W}_1 \triangleq \{\mathbf{w} \in \mathbb{R}^m\},$$

which is the Euclidean space. Therefore, within the SMF scheme, for the update of the combination weights we have

$$\mathbf{w}(t+1) = \arg \min_{\mathbf{w} \in \mathcal{H}_1(t)} ||\mathbf{w} - \mathbf{w}(t)||^2, \tag{2.7}$$

where

$$\mathcal{H}_1(t) \triangleq \{\mathbf{w} \in \mathcal{W}_1 : |d(t) - \mathbf{w}^T \mathbf{y}(t)| \leq \bar{\gamma}$$

is the constraint set for the update and the solution for the (2.7) as we did in (2.4) yields

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t) \frac{\mathbf{y}(t) e(t)}{\mathbf{y}^T(t) \mathbf{y}(t)} \tag{2.8}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(t)|} & \text{if } |e(t)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases}$$

We present a detailed explanation of the algorithm for the unconstrained combination method in Algorithm 2.

Algorithm 2 Unconstrained Combination Algorithm for the Fusion Center

Choose $\bar{\gamma}$
$\mathbf{w}(0) \leftarrow Initialize$
$\alpha \leftarrow Constant$
for $i = 1$ **to** m **do**
  $\mathbf{w}_i(0) \leftarrow Initialize$
  Choose γ_i
end for
for all $t \geq 0$ **do**
  for $i = 1$ **to** m **do**
    $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$
    $e_i(t) = d(t) - \hat{d}_i(t)$
    if $|e_i(t)| > \gamma_i$ **then**
      $\mu_i(t) = 1 - \frac{\gamma_i}{|e_i(t)|}$
      $\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \mu_i(t)\frac{\mathbf{x}(t)e_i(t)}{\alpha + \mathbf{x}^T(t)\mathbf{x}(t)}$
    end if
  end for
  $\mathbf{y}(t) = [\hat{d}_1(t)...\hat{d}_m(t)]^T$
  $\hat{d}(t) = \mathbf{y}^T(t)\mathbf{w}(t)$
  $e(t) = d(t) - \hat{d}(t)$
  if $|e(t)| > \bar{\gamma}$ **then**
    $\mu(t) = 1 - \frac{\bar{\gamma}}{|e(t)|}$
    $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{y}(t)e(t)}{\alpha + \mathbf{y}^T(t)\mathbf{y}(t)}$
  end if
end for

2.3.2 Affine Adaptive Combination Weights

Parameter space for the affine combination weights is defined as

$$\mathcal{W}_2 \triangleq \{\mathbf{w} \in \mathbb{R}^m : \mathbf{1}^T\mathbf{w} = 1\},$$

where $\mathbf{1} \in \mathbb{R}^m$ denotes a vector of ones such that sum of weights to be one, i.e. $\sum_{i=1}^m w^{(i)} = 1$. Therefore, the constraint set in this case is

$$\mathcal{H}_2(t) \triangleq \{\mathbf{w} \in \mathcal{W}_2 : |d(t) - \mathbf{w}^T\mathbf{y}(t)| \leq \bar{\gamma}\}.$$

We remove the affine constraint with the following re-parametrization. Define parameter vector $\mathbf{z}(t) \in \mathbb{R}^{n-1}$, where

$$\mathbf{z}^{(i)}(t) \triangleq \mathbf{w}^{(i)}(t),\ \forall i \in \{1, 2, ..., m-1\}$$

and

$$\mathbf{w}^{(m)}(t) = 1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t). \tag{2.9}$$

Therefore, the final estimation error is expressed with the use of unconstrained parameter vector as

$$\begin{aligned} e(t) &= d(t) - \underbrace{\begin{bmatrix} \mathbf{z}^{(1)}(t) \\ \vdots \\ \mathbf{z}^{(m-1)}(t) \\ 1 - \mathbf{1}^T\mathbf{z}(t) \end{bmatrix}^T}_{\mathbf{w}(t)} \underbrace{\begin{bmatrix} \hat{d}_1(t) \\ \vdots \\ \hat{d}_{m-1}(t) \\ \hat{d}_m(t) \end{bmatrix}}_{\mathbf{y}(t)}, \\ &= d(t) - \begin{bmatrix} \hat{d}_1(t) \\ \vdots \\ \hat{d}_{m-1}(t) \end{bmatrix}^T \mathbf{z}(t) - (1 - \mathbf{1}^T\mathbf{z}(t))\hat{d}_m(t), \\ &= \underbrace{d(t) - \hat{d}_m(t)}_{a(t)} - \underbrace{\begin{bmatrix} \hat{d}_1(t) - \hat{d}_m(t) \\ \vdots \\ \hat{d}_{m-1}(t) - \hat{d}_m(t) \end{bmatrix}^T}_{\mathbf{c}(t)} \mathbf{z}(t). \end{aligned} \tag{2.10}$$

Here in (2.10) we present $z(t)$ as the unconstrained parameter vector, $a(t)$ as the desired signal and $c(t)$ as the input to the unconstrained optimization problem which is given as

$$\mathbf{z}(t+1) = \arg \min_{\mathbf{z} \in \widetilde{\mathcal{H}}_2(t)} \|\mathbf{z} - \mathbf{z}(t)\|^2, \tag{2.11}$$

where the constraint set is defined as

$$\widetilde{\mathcal{H}}_2(t) \triangleq \{\mathbf{z} \in \mathbb{R}^{m-1} : |a(t) - \mathbf{z}^T\mathbf{c}(t)| \leq \bar{\gamma}\}.$$

Since now the optimization problem is same as in unconstrained case, as in (2.7) the solution yields

$$\mathbf{z}(t+1) = \mathbf{z}(t) + \mu(t)\frac{\mathbf{c}(t)e(t)}{\mathbf{c}(t)^T\mathbf{c}(t)} \tag{2.12}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(t)|} & \text{if } |e(t)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases}$$

The input vector $\mathbf{c}(t)$ to the re-parameterized unconstrained version of the optimization problem can be expressed in terms of initial input vector $y(t)$ as

$$\mathbf{c}(t) = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ 0 & 1 & \cdots & 0 & -1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix}}_{\widetilde{\mathbf{G}}} \mathbf{y}(t)$$

Therefore, we can express the each element of unconstrained parameter vector as

$$\mathbf{z}^{(i)}(t+1) = \mathbf{z}^{(i)}(t) + \mu(t) \frac{\widetilde{\mathbf{G}}^{(i)} \mathbf{y}(t) e(t)}{\mathbf{y}(t)^T \widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}} \mathbf{y}(t)} \tag{2.13}$$

which leads to

$$1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t+1) = 1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t) - \mu(t) \frac{\sum_{i=1}^{m-1} \widetilde{\mathbf{G}}^{(i)} \mathbf{y}(t) e(t)}{\mathbf{y}(t)^T \widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}} \mathbf{y}(t)} \tag{2.14}$$

and inserting (2.9) leads to

$$\mathbf{w}^{(m)}(t+1) = \mathbf{w}^{(m)}(t) + \mu(t) \underbrace{\begin{bmatrix} -1 \\ \vdots \\ -1 \\ m-1 \end{bmatrix}^T}_{\mathbf{g}} \frac{\mathbf{y}(t) e(t)}{\mathbf{y}(t)^T \widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}} \mathbf{y}(t)}. \tag{2.15}$$

Thus, by (2.13) and (2.15), we have

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t) \underbrace{\begin{bmatrix} \widetilde{\mathbf{G}} \\ \mathbf{g}^T \end{bmatrix}}_{\mathbf{G}} \frac{\mathbf{y}(t) e(t)}{\mathbf{y}(t)^T \widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}} \mathbf{y}(t)}. \tag{2.16}$$

Note that $\widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}} = \mathbf{G}$, therefore equation (2.15) yields to parameter vector update of

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t) \frac{\mathbf{G} \mathbf{y}(t) e(t)}{\mathbf{y}(t)^T \mathbf{G} \mathbf{y}(t)} \tag{2.17}$$

where

$$\mathbf{G} \triangleq \begin{bmatrix} \mathbf{I}_{m-1} & -\mathbf{1} \\ -\mathbf{1}^T & m-1 \end{bmatrix}$$

and

$$\mu(t) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(t)|} & \text{if } |e(t)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases}$$

and $-\mathbf{1} \in \mathbb{R}^{m-1}$ is a vector where all its elements is minus one. Note that, algorithm for constructing affine combination weights is easily obtained by introducing the matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{m-1} & -\mathbf{1} \\ -\mathbf{1}^T & m-1 \end{bmatrix}$$

and replacing the line 22 in Algorithm 2 with the update line $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{G\mathbf{y}(t)e(t)}{\alpha + \mathbf{y}(t)^T G\mathbf{y}(t)}$.

2.3.3 Convex Adaptive Combination Weights

Lastly, the parameter space for the convex combination weights is defined as

$$\mathcal{W}_3 = \{\mathbf{w} \in \mathbb{R}^m : \mathbf{1}^T\mathbf{w} = 1 \wedge \mathbf{w}^{(i)} \geq 0, \forall i \in \{1, ..., m\}\}.$$

Therefore, the constraint set in this case is

$$\mathcal{H}_3(t) \triangleq \{\mathbf{w} \in \mathcal{W}_3 : |d(t) - \mathbf{w}^T\mathbf{y}(t)| \leq \bar{\gamma}\}.$$

In order to get unconstrained optimization problem as we did above, we re-parameterize vector $\mathbf{w}(t)$ with the parameter vector $\mathbf{z}(t) \in \mathbb{R}^m$ by introducing a softmax layer to the central node such that

$$\mathbf{w}^{(i)}(t) = \frac{e^{-\mathbf{z}^{(i)}(t)}}{\sum_{k=1}^{m} e^{-\mathbf{z}^{(k)}(t)}}, \tag{2.18}$$

which maps the unconstrained combination weight to the probability simplex in order to obtain the convex combination weight on the fusion center.

Note that SM-NLMS algorithm can also be constructed through gradient descent method with stochastic cost function defined as

$$F(e(t)) \triangleq \begin{cases} \left(\frac{|e(t)| - \bar{\gamma}}{\|\mathbf{y}(t)\|}\right)^2 & |e(t)| > \bar{\gamma} \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, for the unconstrained parameter vector update, stochastic gradient algorithm is then given by

$$\mathbf{z}(t+1) = \mathbf{z}(t) - \frac{1}{2}\nabla_{\mathbf{z}} F(e(t)), \tag{2.19}$$

which by the chain rule yields to

$$\mathbf{z}(t+1) = \mathbf{z}(t) - \frac{1}{2}[\nabla_{\mathbf{z}}\mathbf{w}(t)]^T \nabla_{\mathbf{w}} F(e(t)). \tag{2.20}$$

Note that $\nabla_{\mathbf{z}}\mathbf{w}(t) = \mathbf{w}(t)\mathbf{w}(t)^T - \text{diag}\{\mathbf{w}(t)\}$ [17] and by this we obtain

$$\mathbf{z}(t+1) = \mathbf{z}(t) + \mu(t)[\mathbf{w}(t)\mathbf{w}(t)^T - \text{diag}\{\mathbf{w}(t)\}]\frac{\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\mathbf{y}(t)} \tag{2.21}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(t)|} & \text{if } |e(t)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$\mathbf{w}(t) = \frac{e^{-\mathbf{z}(t)}}{\|e^{-\mathbf{z}(t)}\|_1}.$$

Finally, we easily obtain the algorithm for constructing the convex combination weights by defining unconstrained parameter vector as in (2.18) and by replacing line 22 in Algorithm 2 with the update line in (2.20).

In the next section, with the algorithms defined above, we evaluate the MSE performance of the algorithms within different schemes.

2.4 Simulations and Results

In this section, through series of simulations, we demonstrate the performance of the proposed SMF centralized distributed estimation algorithms and compare the steady-state and convergence performances with various methods, i.e. NLMS centralized method and variable step size NLMS, as well as its superior computation and communication efficiency [18, 38].

We first considered the performance for the non-stationary data case where statistics of source data has an abrupt change in the middle of simulations. We also analyzed how predetermined error bounds on the central node effects the performance of the overall system. Then we demonstrate simulations with real and synthetic benchmark data sets such as Elevators, Kinematics, Pumadyn and California housing data [39]. In the final part, we compare computation and communication load of the proposed algorithms with respect to NLMS centralized distributed estimation algorithm and single variable step-size NLMS algorithm to demonstrate the efficiency of our solutions.

Through this section, we refer set-membership normalized least mean square algorithm as "SM-NLMS" and centralized systems that use unconstrained, affine and convex combination methods on central nodes as "SM-UNC", "SM-AFF" and "SM-CONV" respectively. We also introduce variable step size NLMS algorithm as "VSS-NLMS" and centralized NLMS algorithm as "NLMS"[18, 38].

2.4.1 Non-Stationary Data

In this part, we study our algorithms in a non-stationary environment where data source statistics have an abrupt change in the middle of time horizon of the simulations. We use static noise statistics over all distributed nodes and central node as well, which is all the nodes in the network experience same level of noise disturbance to their observations. We create a sequence for $T = 10000$ time step considering a linear-in-parameter model such that

$$d_i(t) = \mathbf{w}_o^T \mathbf{x}(t) + n_i(t),$$

where $\mathbf{w}_o \in \mathbb{R}^7$ denotes the parameter of interest, $\mathbf{x}(t) \in \mathbb{R}^7$ is the input regressor vector and $n_i(t)$ is the corresponding additive white Gaussian noise signal with fixed variance σ_n^2 .

We sample the parameter of interest $\mathbf{w}_o$ from normal distribution with zero mean and unit variance and normalize it to $||\mathbf{w}_o|| = 1$. In the middle of time horizon, we sample the parameter of interest again to create a non-stationary

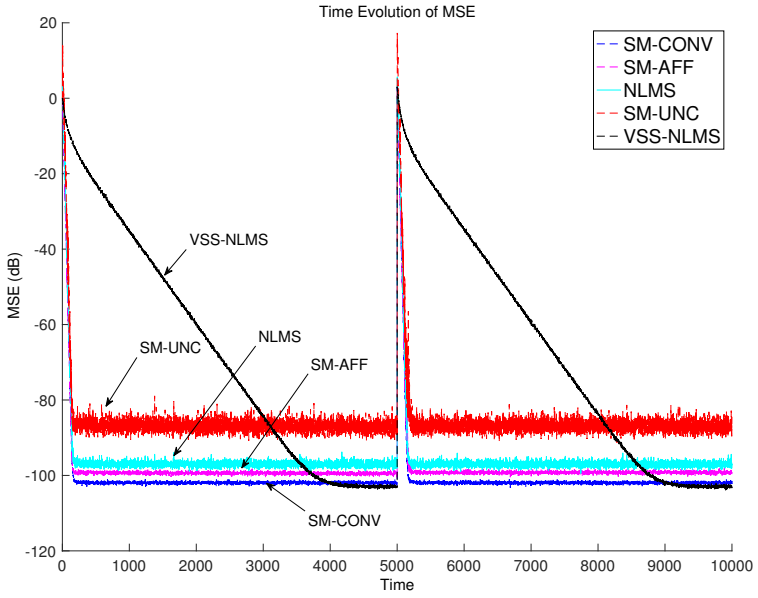

Figure 2.3: Time evolution of MSE performance of the proposed algorithm compared with others over non-stationary data having 20dB SNR and input vector eigenvalue spread of 1. Note that drastic increase in the middle of figure corresponds to the time re-sample the parameter of interest to create non-stationary environment and measure the tracking performance of the compared algorithms.

environment. We sample the input vectors from normal distribution so that eigenvalue spread is 1 and we select the additive white noise so that we have 20dB observation signal. We use 10 distributed nodes, where we deploy SMF algorithms with different error bounds set varying around $\sqrt{5\sigma_n^2}$ and we select the final error bound on the central node as $\sqrt{5\sigma_n^2}$. We select the learning rate for the NLMS algorithms as $\mu = 0.2$ and we select the learning rate interval for the VSS-NLMS algorithm as $(\mu_{max}, \mu_{min}) = (0.2, 0.02)$. We averaged the results over 200 trials to obtain the ensemble average of the MSE measure.

In Fig.2.3, we present the performance comparison of the algorithms under non-stationary data. We observe that the introduced algorithms have better learning-rate performance while VSS-NLMS algorithm performs the worst in this measure due to non-collaborative structure of the algorithm. We also note that SM-CONV and VSS-NLMS algorithms have the best steady-state error performance compared to the others. We also emphasize that the proposed algorithms

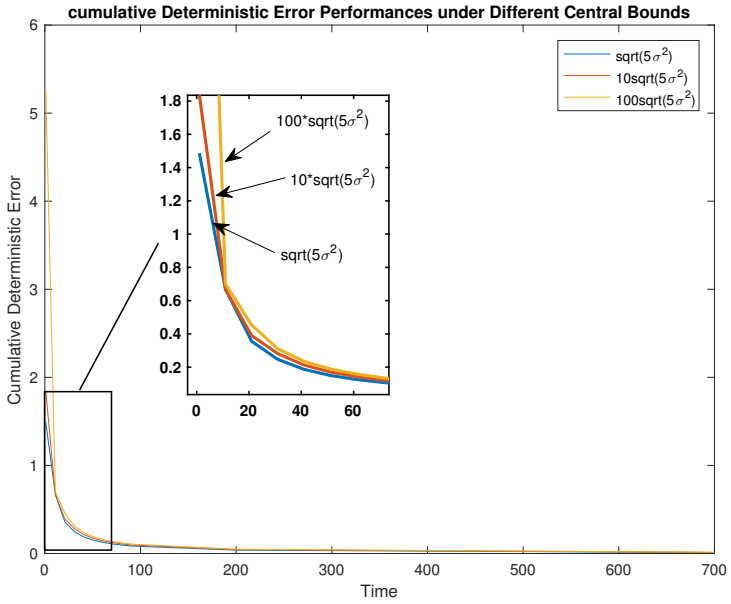

Figure 2.4: Cumulative deterministic error performance of algorithms with different central error bounds.

reach the presented performance with much less computation and communication load than the compared algorithms.

In addition, error bound selection is indeed a problem for set-membership filtering (SMF), especially when the power of the noise of the environment is unknown. One of our main motivation for using the centralized distributed estimation approach with SMF is to resolve this problem by collecting information from distributed nodes with different SMFs with a wide range of representative error bounds. Hence, on the peripheral nodes, we use diverse range of error bounds to cover nearly every important realistic case.

However, we emphasize that the selection of the error bound on the central node is important. The error bound of the fusion center determines the trade-off between low residual error and low computational complexity. Therefore, it should be selected based on the application specifications. For instance, if we seek a low residual error and computational load is not a concern, then we set a tight bound and system updates itself until reaching the desired bound. For another

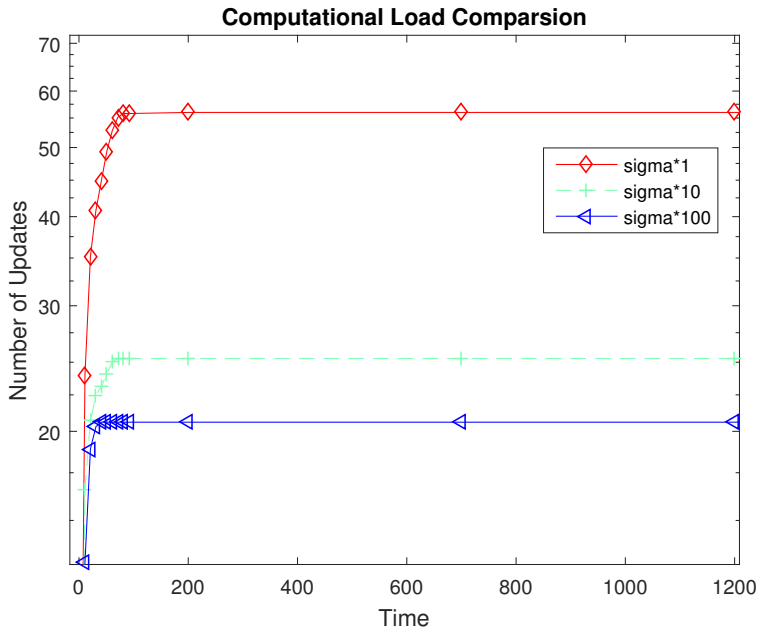

Figure 2.5: Number of updates that each algorithm with different central error bounds require. Number of updates are presented in a semi-log scale.

case, if we seek for convergence with a low computational complexity, then we set a loose bound and system stops updating after converging to the bound.

To this end, in this part, we also study the selection of the final stage error bound in a stationary environment. We use convex algorithm to construct the combination weights. For comparison, we set the error bound of the central node as $\sqrt{5\sigma_n^2}$, $10\sqrt{5\sigma_n^2}$ and $100\sqrt{5\sigma_n^2}$ for different cases. We present the evolution of cumulative deterministic squared error for different selection of final error bound in Fig.2.4 and evolution of the number of updates they require in Fig.2.5. We observe that algorithms having tighter final error bounds converge faster compared to loose bounded algorithms. However, we also observe that these tight bounded algorithms make much more update than other, which results in higher computational load over the system.

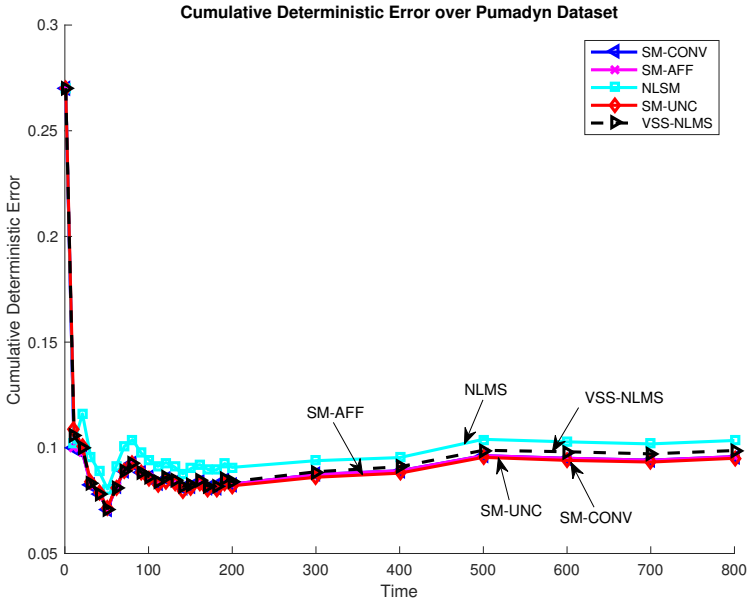

Figure 2.6: Cumulative deterministic error performance of the proposed algorithms compared to NLMS and VSS-NLMS over Pumadyn dataset.

2.4.2 Benchmark Real Data

Here, we apply our algorithms to the learning of the benchmark real-life problems [39]. In real-life dataset experiments, we use 10 distributed nodes and since this time we do not know the power of the additive noise, we set the error bounds of the SMF filters running on distributed nodes in a wide range spread around 0.15 and again we choose the error bound for the central node as 0.15. For NLMS algorithm we choose step size $\mu_{NLMS} = 0.2$ and for VSS-NLMS algorithm we set the step size range as $(\mu_{max}, \mu_{min}) = (0.2, 0.02)$.

For the first experiment, we use Pumadyn data with regressor dimension $n = 32$ which is a dataset obtained from a realistic simulations of the dynamics of Unimation Puma 560 robot arm [39]. We present the deterministic cumulative squared error results in Fig.2.6. Note that in Fig.2.6, mixture approaches show superior performance over other filters. Here, we observe that the introduced methods present superior performance compared to the others regarding both transient and steady-state performances. We also note that VSS-NLMS algorithm

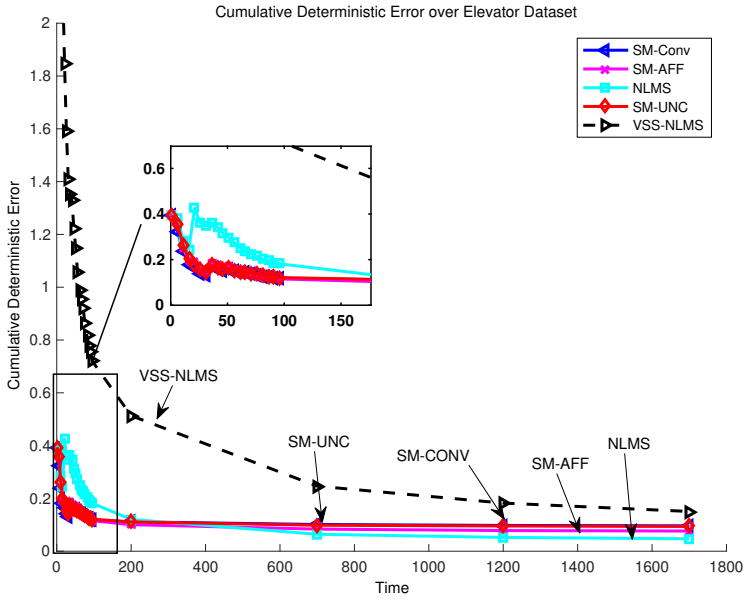

Figure 2.7: Cumulative deterministic error performance of the proposed algorithms compared to NLMS and VSS-NLMS over Elevator dataset.

presents comparable performance in transient state with the proposed algorithm. However, the proposed algorithms performs better in steady-state error.

For the second benchmark test, we use Elevator dataset, which is obtained from the task of controlling F16 aircraft and the desired data is related to an action taken on the elevators of the aircraft [39]. The regressor dimension for Elevator data is 18. We present the results of the simulations regarding this dataset in Fig.2.7. We observe in this case that the proposed algorithms perform superior compared to VSS-NLMS regarding both learning-rate and steady-state error performance. Although NLMS algorithm reaches a lower steady-state error, it has a lower learning-rate performance compared to the introduced algorithms.

Besides Pumadyn and Elevator experiments, we also use California Housing and Kinematics datasets to perform real-life simulations [39]. California housing dataset is based on house prices in California having different features, e.g. square-footage. Kinematics dataset is also obtained from forward kinematics of

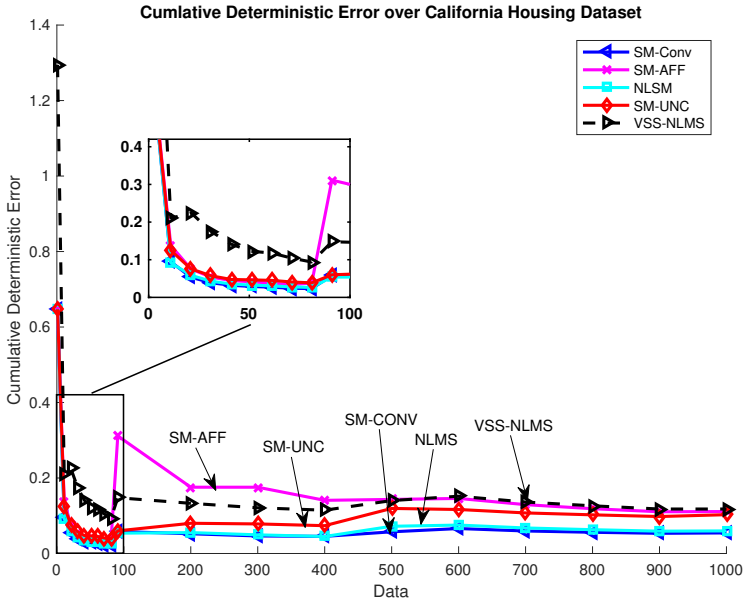

Figure 2.8: Cumulative deterministic error performance of the proposed algorithms compared to NLMS and VSS-NLMS over California Housing dataset.

an8 link robot arm [39]. We present the simulation results over California housing and Kinematics datasets in Fig. 2.8 and Fig.2.9 respectively. We observe that over California Housing data, VSS-NLMS algorithm performs inferior compared to the other algorithms. This is mainly because we do not have any prior information about the dataset and non-collaborative VSS-NLMS algorithm cannot perform well due to the poor selection of predefined parameters. On the other hand, distributed algorithms performs better even if we do not have any prior information about the data since we cover a selection of cases over distributed nodes running different configuration of learning algorithms. We also note that SM-CONV algorithm performs superior compared to other algorithms regarding both learning-rate and steady-state error performance. In simulations over Kinematics data, we observe that the introduced algorithms perform superior compared to distributed NLMS algorithm and single VSS-NLMS algorithm regarding both transient and steady-state error performance.

We note that during simulations over benchmark real-data, the compared algorithms show different performances, due to lack of prior information on the

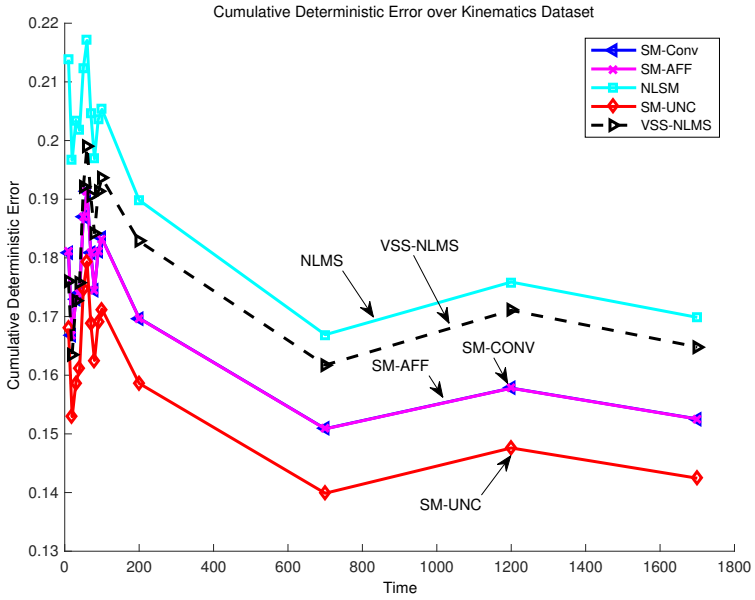

Figure 2.9: Cumulative deterministic error performance of the proposed algorithms compared to NLMS and VSS-NLMS over Kinematics dataset.

behavior of the datasets. However, the introduced algorithms managed to perform better than the standard LMS based algorithms in every case. We also emphasize that the proposed algorithms present these performances with significantly reduced communication and computation load. We present detailed results for this aspect of our methods in the following subsection.

2.4.3 Communication and Computation Load

One of the critical aspects of the proposed algorithms is the reduced communication and computation load regarding lessened update of weights compared to the standard LMS algorithm and distributed methods. To present that, we calculated the total number of addition and multiplication operation that each algorithm required during a simulation over a stationary data for 8000 instances. In Table 2.1, we demonstrate results for addition and multiplication operation that each algorithm made and show that proposed algorithms computationally more efficient than other conventional LMS algorithms and centralized approaches.

Table 2.1: Total Number of Addition and Multiplication Operations Each Algorithm Require over 8000 Stationary Data Instances

	SM-UNC	SM-AFF	SM-CONV	NLMS	VSS-NLMS
Addition	680	1020	775	240600	480000
Multiplication	680	1020	775	160000	400000

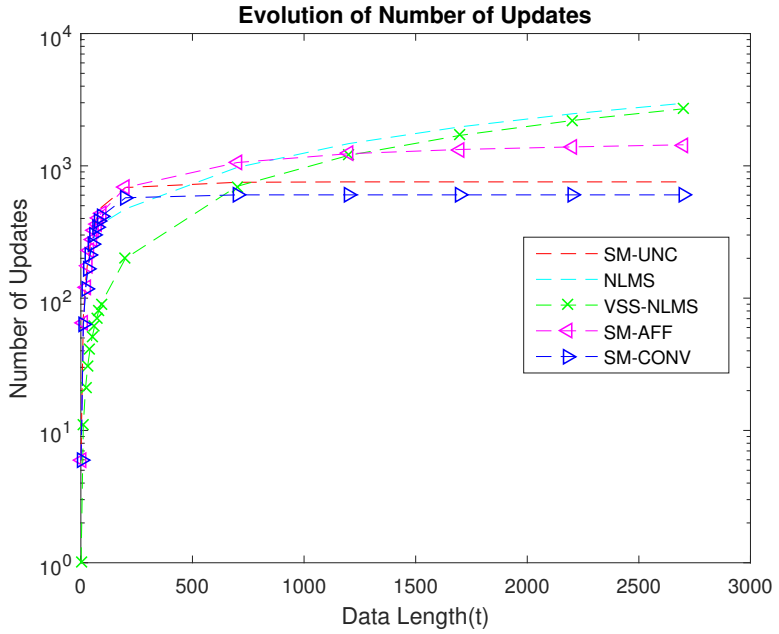

Figure 2.10: Number of updates that each algorithm made on their estimations and combination weights over stationary data. Only 2500 instances are presented since SMF based algorithms stop updating after 200 instances.

Although, the computational cost among the proposed algorithms do not differ much, we emphasize that the unconstrained mixture is the most computationally efficient one. In Fig.2.10, we also presented number of updates that each algorithm made both on their distributed nodes and on central node. We note that the set-membership based distributed algorithms stop updating after satisfying their error bound requirement, which means also the distributed nodes stop sending information to the fusion center. On the other hand, the conventional LMS based algorithms keep updating their estimations and combination weights. Note that the distributed NLMS algorithm requires more updates than the single VSS-NLMS algorithm. Therefore, we conclude that the proposed methods reduce both communication and computation load over the network.

Chapter 3

Efficient and Optimal Decentralized Online Learning of Dynamic Parameters

In this part of the thesis, we study the problem of online learning over a decentralized distributed network, where nodes in the network collaboratively estimate a dynamically evolving parameter using noisy observations. Nodes in the network are equipped with processing and communication capabilities and can share their observations or local estimates to their connected neighbors. The conventional distributed estimation algorithms only diffuse their current observations or estimations and in this regard, these algorithms cannot perform optimal online learning in mean-square error sense (MSE). To this end, we present an optimal distributed learning algorithm through disclosure of local estimate for tracking the underlying dynamic parameter. We first show that optimal estimation can be achieved through diffusion of all the time stamped observations for any arbitrary network and we prove that optimal estimation through disclosure of local estimates is possible for certain network topologies. We then derive an iterative algorithm to recursively calculate the combination weights and construct the optimal estimation for each time step. Through series of simulations, we demonstrate the superior performance of the proposed algorithm compared

to state-of-the-art diffusion distributed estimation algorithms regarding convergence rate and steady-state error levels. We also show that while conventional distributed estimation schemes cannot track highly dynamic parameters, through optimal weight and estimation construction, the proposed algorithm presents stable MMSE performance.

3.1 Optimal Distributed Estimation Framework

In this framework, we consider a distributed network with m agents equipped with processing and communication capabilities. In Fig. 3.1, we illustrate such a network as an undirected graph, where vertices and edges represents the agents and communication links respectively. For each agent i, we denote the set of other agents, whose information is available to the agent i after k-hops over communication links as $\mathbf{N}_i^{(k)}$. We define $\mathbf{N}_i^{(k)}$ as

$$\mathbf{N}_i^{(k)} = \{j_i, \cdots, j_{\pi_i}\}, \tag{3.1}$$

where $\pi_i^{(k)} = |\mathbf{N}_i^{(k)}|$ is the cardinality of $\mathbf{N}_i^{(k)}$ and $\mathbf{N}_i^{(0)} = \{i\}$ and $\mathbf{N}_i^{(k)} = \emptyset$ for $k < 0$. Each agent observes a noisy version of an underlying dynamic state. The underlying state $x_t \in \mathbb{R}$ evolves according to a random walk model such that [1]

$$x_{t+1} = \gamma x_t + w_t, \tag{3.2}$$

where $\gamma \in \mathbb{R}$ is the expected rate of change. The term $w_t \in \mathbb{R}$ is the state noise and driven by a white Gaussian random process $\{\mathcal{W}_t\}$ with variance σ_w^2. The initial state is sampled from a Gaussian random variable such that $\mathcal{X}_0 \sim N(0, \sigma_0^2)$. Observation of the agent i at time t is then given by

$$y_{i,t} = x_t + n_{i,t} \tag{3.3}$$

for $i = 1, \cdots, m$ and $n_{i,t} \in \mathbb{R}$ is also driven by a white Gaussian process $\{\mathcal{N}_{i,t}\}$ with variance $\sigma_{n_i}^2$. We assume that the observation noise is spatially independent

[1] In this paper, all random variables are presented as uppercase calligraphic letters, i.e. $\mathcal{X}$ and all realizations of these variables are presented as their lowercase characters, i.e. x. All vectors are column vectors and denoted by boldface letters.

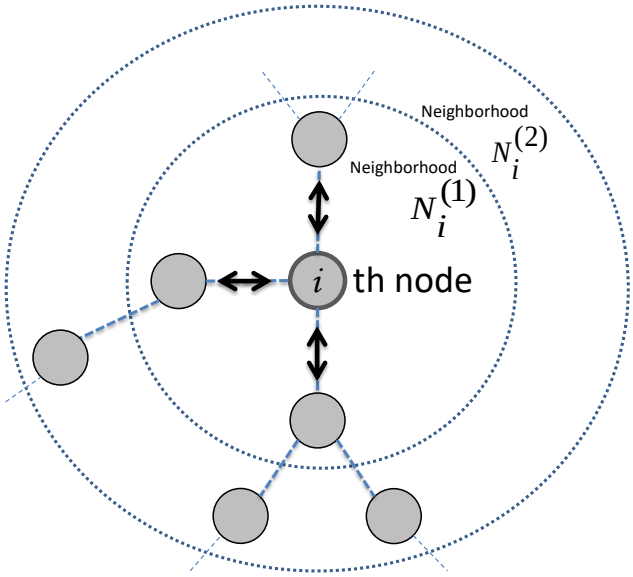

Figure 3.1: Neighborhoods of agent i over a distributed network.

and the variance of the noise signals are known to each agent. Note that even if we do not use this assumption, the noise variance can be estimated from data as well [40]. Correspondingly, $y_{i,t}$ becomes a realization of a random process $\{\mathcal{Y}_{i,t}\}$, where $\mathcal{Y}_{i,t} = \mathcal{X}_t + \mathcal{N}_{i,t}$. At each instant, an agent receives a local observation and diffused information from neighboring agents, while it diffuses information to its neighboring agent as well, as we illustrate in Fig.3.1.

Obviously, the agents can track the underlying state in MMSE sense under certain regulatory conditions[41]. However, with distributed cooperation, our intention is to enhance the learning rate of the system [40]. To this end, we aim to find an optimal learning strategy regarding MSE performance for distributed networks.

We first consider a simple case to achieve the optimal learning strategy. For this case, we restrict the agents in the network to diffuse time and agent stamped versions of their observations and the received information. Thus, each agent has access to the observations of the other agents. However, we note that the observations from non-neighboring agents can only be accessed after certain hops

over communication links, i.e. information of an agent $j \in \mathbf{N}_i^{(2)}$ can be accessed by agent i after 2 hops as seen in Fig.3.1.

We also emphasize that due to the connected structure of the network, each agent will have access to all the observations in the network, although with delay of certain hops. Therefore, we denote the information aggregated at agent i at time t as

$$D_{i,t} = \left\{ \{y_{i,\tau}\}^{\tau \leq t}, \{y_{j,\tau}\}_{j \in \mathbf{N}_i}^{\tau \leq t-1}, \{y_{j,\tau}\}_{j \in \mathbf{N}_i^{(2)}}^{\tau \leq t-2}, \cdots, \{y_{j,\tau}\}_{j \in \mathbf{N}_i^{(\kappa_i)}} \right\}, \tag{3.4}$$

where κ_i denotes the communication link delay for the furthest node. Note that $\{y_{j,\tau}\}_{j \in \mathbf{N}^{(t_i)}}^{\tau \leq t-t_i}$ is the set of observations received from t_i hop away neighborhood of agent i, which is explicitly defined as

$$\{y_{j,\tau}\}_{j \in \mathbf{N}^{(t_i)}}^{\tau \leq t-t_i} \triangleq \{y_{j_1,t-t_i}, \ldots, y_{j_1,0}, \cdots, y_{j_{\pi_i^{(t_i)}},t-t_i}, \ldots, y_{j_{\pi_i^{(t_i)}},0}\}. \tag{3.5}$$

With this aggregated information, we construct the optimization problem for distributed framework in MSE sense as

$$\begin{aligned} \hat{x}_{i,t} = \arg\min_x \mathbb{E}\Big[||\mathcal{X}_t - x||^2 \Big| \{\mathcal{Y}_{i,\tau} = y_{i,\tau}\}^{\tau \leq t}, \\ \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in \mathbf{N}_i}^{\tau \leq t-1}, \cdots, \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in \mathbf{N}_i^{\kappa_i}}^{\tau \leq t-\kappa_i} \Big]. \end{aligned} \tag{3.6}$$

We find the solution to optimization problem in (3.6) as MMSE estimate for the agent i, which is given by the expectation of x conditioned on all the accessed information such that

$$\begin{aligned} \hat{x}_{i,t} = \mathbb{E}\Big[\mathcal{X}_t \Big| \{\mathcal{Y}_{i,\tau} = y_{i,\tau}\}^{\tau \leq t}, \\ \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in \mathbf{N}_i}^{\tau \leq t-1}, \cdots, \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in \mathbf{N}_i^{\kappa_i}}^{\tau \leq t-\kappa_i} \Big]. \end{aligned} \tag{3.7}$$

Therefore, estimate in (3.7) is found to be optimal in MSE sense for the case where agents diffuse time stamped observations.

Remark 3.1.1 *Although the presented case can achieve the optimal estimation in MSE sense through the disclosure of time stamped observations, this scheme requires excessive amount of storage on nodes and communication load for the*

network, especially for larger networks. Note that reduced storage and communication load are essential for the applicability of the distributed networks [42, 43]. Therefore, we develop optimal learning strategies for distributed networks, where nodes only store and diffuse their current local estimations. However, in the next section, we show that optimal estimation with disclosure of local estimates can only be achieved over certain network topologies.

3.2 Optimal Estimation with Disclosure of Local Estimates

In this section, we show whether optimal estimation can be achieved through disclosure of local estimations for certain network topologies. We first prove that disclosure of local estimates is not enough to achieve optimal estimation over cyclic networks. Then we study the case on tree-networks to prove that optimal estimation is achievable over such networks.

3.2.1 Estimation over Cyclic Networks

In this part, we show that optimal estimation cannot be constructed through disclosure of local estimates over cyclic networks with a basic counter example [35]. For the sake of simplicity, we assume that the underlying state is static, i.e. $x_t = x_o \in \mathbb{R}$ for all t and is sampled from a Gaussian distribution with mean $\bar{x}$ and variance σ_x^2. We also assume that the noise levels are the same for all agents, i.e. $\sigma_{n_i}^2 = \sigma_n^2$. For this case, we consider a cyclic network with 5 agents as in Fig.3.2.

In Section.3.1, we proved that the optimal estimation is achievable with full disclosure of information over arbitrary networks. Using the results in (3.7), we calculate the optimal estimates on agents 2, 3 and 5 at time $t = 2$ with an abuse

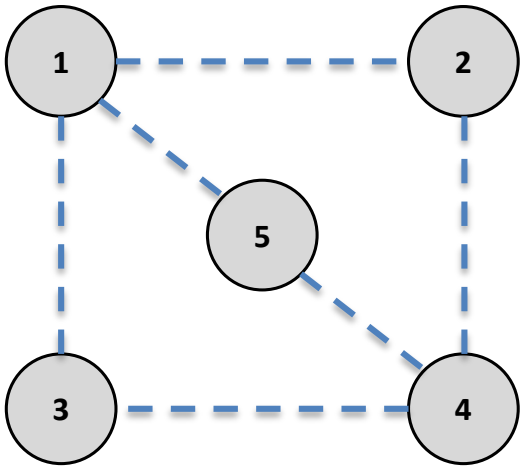

Figure 3.2: Cyclic network of 5 agents.

of notation as

$$\begin{aligned}
\hat{x}_{2,2} &= \mathbb{E}\left[\mathcal{X}_2 \middle| y_{1,1}, y_{2,2}, y_{2,1}, y_{4,1}\right] \\
&= \frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1}+y_{2,2}+y_{2,1}+y_{4,1}), \\
\hat{x}_{3,2} &= \mathbb{E}\left[\mathcal{X}_2 \middle| y_{1,1}, y_{3,2}, y_{3,1}, y_{4,1}\right] \\
&= \frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1}+y_{3,2}+y_{3,1}+y_{4,1}), \\
\hat{x}_{5,2} &= \mathbb{E}\left[\mathcal{X}_2 \middle| y_{1,1}, y_{5,2}, y_{5,1}, y_{4,1}\right] \\
&= \frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1}+y_{5,2}+y_{5,1}+y_{4,1}).
\end{aligned}$$

Again in the same scheme, we calculate the optimal estimate on agent 1 at time $t = 3$ as

$$\begin{aligned}
\hat{x}_{1,3} &= \mathbb{E}\left[\mathcal{X}_3 \middle| y_{1,3}, y_{1,2}, y_{1,1}, y_{2,2}, y_{2,1}, y_{3,2}, y_{3,1}, y_{5,2}, y_{5,1}, y_{4,1}\right] \\
&= \frac{\sigma_x^2}{10(\sigma_x^2+\sigma_n^2)}(y_{1,3}+y_{1,2}+y_{1,1}+y_{2,2}+y_{2,1}+y_{3,2}+y_{3,1} \\
&\qquad + y_{5,2}+y_{5,1}+y_{4,1}).
\end{aligned} \tag{3.8}$$

However, with the disclosure of the local estimates only, we construct our estimate for agent 1 at time $t = 3$ as

$$\tilde{x}_{1,3} = \mathbb{E}\left[\mathcal{X}_3 \middle| y_{1,3}, y_{1,2}, y_{1,1}, \hat{x}_{2,2}, \hat{x}_{2,1}, \hat{x}_{3,2}, \hat{x}_{3,1}, \hat{x}_{5,2}, \hat{x}_{5,1}\right]. \tag{3.9}$$

Note that all the parameters in calculation of the expectation in (3.9) are jointly Gaussian. Thus, the estimate $\tilde{x}_{1,3}$ is linear in $\hat{x}_{2,2}$, $\hat{x}_{3,2}$ and $\hat{x}_{5,2}$ such that

$$\begin{aligned}
\tilde{x}_{1,3} &= \cdots + a\hat{x}_{2,2} + b\hat{x}_{3,2} + c\hat{x}_{5,2} \\
&= \cdots + a\frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1} + y_{2,2} + y_{2,1} + y_{4,1}) \\
&\quad + b\frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1} + y_{3,2} + y_{3,1} + y_{4,1}) \\
&\quad + c\frac{\sigma_x^2}{4(\sigma_x^2+\sigma_n^2)}(y_{1,1} + y_{5,2} + y_{5,1} + y_{4,1}),
\end{aligned}$$

where $\cdots$ represents the other terms in the calculation. We point out that the optimal estimation $\hat{x}_{1,3}$ in (3.8) implies $a+b+c = 4/10$ due to terms $y_{2,2}$, $y_{3,2}$ and $y_{5,2}$. However, at the same time it implies $a + b + c = 4/20$ due to term $y_{4,1}$ that exists in $\hat{x}_{2,2}$, $\hat{x}_{3,2}$ and $\hat{x}_{5,2}$. Therefore, this contradiction proves that the optimal estimation $\hat{x}_{1,3}$ cannot be constructed through the disclosure of local estimations.

3.2.2 Estimation over Tree Networks

In this part, we study optimal learning strategies over tree-based networks. We define the tree-networks as a graph structures that the vertices are connected with undirected edges without any cycles. We also note that for any arbitrary network topology, a minimum spanning tree of the network can be constructed for eliminating the cycles [44, 45, 46, 47]. In the following, we show that the optimal estimation can be constructed through the disclosure of local estimates over tree-based networks.

At any time t, we defined the information aggregated on agent i as $D_{i,t}$ in (3.4) provided a time stamped full information disclosure. Using $D_{i,t}$, we partition the set coming from a particular neighborhood using the tree structure of the network.

Note that for the tree networks, a neighboring set for agent i can be expressed as

$$\mathbf{N}_i^{(k)} = \bigcup_{j \in \mathbf{N}_i} (\mathbf{N}_i^{(k)} \cap \mathbf{N}_j^{(k-1)})$$

and again due to the network structure, the intersecting sets are disjoint such that

$$(\mathbf{N}_i^{(k)} \cap \mathbf{N}_{j_1}^{(k-1)}) \bigcap (\mathbf{N}_i^{(k)} \cap \mathbf{N}_{j_2}^{(k-1)}) = \emptyset$$

for all $j_1, j_2 \in \mathbf{N}_i$ and $j_1 \neq j_2$. Therefore, we can partition the information received at agent i after k-hops as

$$\{y_{j,\tau}\}_{j \in \mathbf{N}_i^{(k)}} = \{\{y_{j,\tau}\}_{j \in \mathbf{N}_i^{(k)} \cap \mathbf{N}_{j_1}^{(k-1)}}, \cdots, \{y_{j,\tau}\}_{j \in \mathbf{N}_i^{(k)} \cap \mathbf{N}_{j_{\pi_i}}^{(k-1)}}\}.$$

Using this partitioning method, we define the set of new measurements coming from agent j to i at time $t = 2$ as

$$z_{j \to i,2} \triangleq \left\{ \{y_{k,\tau}\}_{k \in \mathbf{N}_i^{(1)} \cap \mathbf{N}_j^{(0)}}, \{y_{k,\tau}\}_{k \in \mathbf{N}_i^{(2)} \cap \mathbf{N}_j^{(1)}} \right\}. \tag{3.10}$$

Note that the expression in (3.10) can also be written as

$$z_{j \to i,2} = D_{j,2} / \{y_{j,1}, y_{i,1}\},$$

where $y_{j,1} = D_{j,1}$ and $y_{i,1} = D_{i,1} = z_{i \to j,1}$. Thus we can generalize the new information expression for any time t as

$$z_{j \to i,t} = D_{j,t} / \{D_{j,t-1} \cup z_{i \to j,t-1}\}, \tag{3.11}$$

Using (3.11), we write all the information aggregated at agent i as

$$D_{i,t} = \{y_{i,t}, z_{j_1 \to i,t-1}, \cdots, z_{j_{\pi_i} \to i,t-1}, D_{i,t-1}\}, \tag{3.12}$$

where $z_{j \to i,t}$ is constructible from $D_{i,\tau}$ and $D_{j,\tau}$ for $\tau \leq t$ as we show in (3.11). Therefore, using (3.12), we construct the optimal estimation again with an abuse of notation as

$$\hat{x}_{i,t} = \mathbb{E}\left[\mathcal{X}_t \middle| y_{i,t}, z_{j_1 \to i,t-1}, \cdots, z_{j_{\pi_i} \to i,t-1}, D_{i,t-1} \right]. \tag{3.13}$$

Considering $z_{j\to i,t}$ is constructible from $D_{i,\tau}$ and $D_{j,\tau}$ for $\tau \leq t$, we can write the optimal estimation in (3.13) such that

$$\begin{aligned}\hat{x}_{i,t} &= \mathbb{E}\left[\mathcal{X}_t \middle| \{y_i,\tau\}^{\tau\leq t}, \{D_{j,\tau}\}_{j\in\mathbf{N}_i}^{\tau\leq t-1}\right] \\ &= \mathbb{E}\left[\mathcal{X}_t \middle| y_{i,t}, D_{i,t-1}, \{D_{j,t-1}\}_{j\in\mathbf{N}_i}\right]\end{aligned}$$

and since $\hat{x}_{j,t-1} = \mathbb{E}[\mathcal{X}_{t-1}|D_{j,t-1}]$, then we obtain

$$\begin{aligned}\hat{x}_{i,t} &= \mathbb{E}\left[\mathcal{X}_t \middle| y_{i,t}, \mathbb{E}[\mathcal{X}|D_{i,t-1}], \{\mathbb{E}[\mathcal{X}|D_{j,t-1}]\}_{j\in\mathbf{N}_i}\right] \\ &= \mathbb{E}\left[\mathcal{X}_t \middle| y_{i,t}, \hat{x}_{i,t-1}, \{\hat{x}_{j,t-1}\}_{j\in\mathbf{N}_i}\right],\end{aligned} \tag{3.14}$$

which concludes our proof of obtaining optimal estimation through disclosure of local estimates over tree-networks.

Note that we made the derivations in this section for a generalized case such that we proved all the information to be aggregated in an agent can be reconstructed from disclosure of local estimations over tree-networks.

In the following, we introduce efficient and optimal distributed online learning algorithm for dynamic state estimation. We propose a method that iteratively calculates the underlying state at each time instant and achieves MMSE performance.

3.3 Efficient and Optimal Distributed Online Learning

In Section 3.2, we proved that over a tree network, the optimal estimation can be achieved using the disclosure of local estimates as

$$\hat{x}_{i,t} = \mathbb{E}\left[\mathcal{X}_t \middle| y_{i,t}, \hat{x}_{i,t-1}, \{\hat{x}_{j,t-1}\}_{j\in\mathbf{N}_i}\right].$$

Note that each local estimate $\hat{x}_{i,t}$ is linear in old estimates $\hat{x}_{i,\tau}$ and $\{\hat{x}_{j,t-1}\}_{j\in\mathbf{N}_i}$. Therefore, instead of disclosing the local estimates, we constraint each agent to

disclose the information that do not included in old estimations. Then each receiving agent extracts the innovation terms, new information in the disclosed data that agent does not have. Although, this operation imposes more computational load on agents, it reduces the communication load on the network, which is more essential in distributed systems [42].

We denote these innovations extracted from data disclosed by agent j for agent i at time t as $z_{j\to i,t-1}$. Therefore, we define the random vector containing the previous estimate and the aggregated information on agent i at time t

$$\boldsymbol{d}_{i,t} = \begin{bmatrix} \mathcal{Y}_{i,t} & \hat{\mathcal{X}}_{i,t-1} & \mathcal{Z}_{j_1\to i,t-1} & \cdots & \mathcal{Z}_{j_{\pi_i}\to i,t-1} \end{bmatrix}^T, \tag{3.15}$$

so that we find the optimal estimation for the state with realizations of the elements in $\boldsymbol{d}_{i,t}$ is

$$\hat{x}_{i,t} = \mathbb{E}\left[\mathcal{X}_t \middle| \mathcal{Y}_{i,t} = y_{i,t}, \hat{\mathcal{X}}_{i,t-1} = \hat{x}_{i,t-1}, \{\mathcal{Z}_{j\to i,t-1} = z_{j\to i,t-1}\}_{j\in\mathbf{N}_i}\right].$$

Due to the state-space model defined in (3.2) and (3.3), all the parameters in (3.15) are jointly Gaussian. Hence, for the estimation of next state at agent i we have

$$\hat{x}_{i,t} = \alpha_{i,t}\hat{x}_{i,t-1} + \beta_{i,t}y_{i,t} + \sum_{j\in\mathbf{N}_i} c_{i,j}^{(j)} z_{j\to i,t-1}. \tag{3.16}$$

Using estimation model in (3.16), the information disclosed by agent j at time t is given by

$$\begin{aligned} z_{j,t} &= \hat{x}_{j,t} - \alpha_{j,t}\hat{x}_{j,t-1} \\ &= \beta_{j,t}y_{j,t} + \sum_{k\in\mathbf{N}_j} c_{j,t}^{(k)} z_{k\to j,t-1}. \end{aligned} \tag{3.17}$$

Then we extract the innovation from the disclosed information on agent i as

$$\begin{aligned} z_{j\to i,t} &= z_{j,t} - c_{j,t}^{(i)} z_{i\to j,t-1} \\ &= z_{j,t} - c_{j,t}^{(i)} z_{i,t-1} + c_{j,t}^{(i)} c_{i,t-1}^{(j)} z_{j\to i,t-2}. \end{aligned} \tag{3.18}$$

Remark 3.3.1 *Note that there are diffused information that are received after certain delays over the network. Therefore, some of the received information will*

be the noisy versions of the previous instances of the underlying state. Due to the random walk model in (3.2)*, the state noise of these previous instances becomes correlated with the conditioned parameters, which requires a different approach from the existing methods [35].*

In order to calculate the parameters in the estimation recursion (3.16), first we need to calculate the auto-correlation matrix of the information collecting random vector $\boldsymbol{d}_{i,t}$ and the cross-correlation vector with the underlying state $\mathcal{X}_t$, where we define them as $\boldsymbol{\Sigma}_{dd_i,t}$ and $\boldsymbol{\Sigma}_{xd_i,t}$ respectively.

We first calculate the terms of $\boldsymbol{\Sigma}_{xd_i,t}$ starting with

$$\begin{aligned}\mathbb{E}[\mathcal{X}_t\mathcal{Y}_{i,t}] &= \mathbb{E}[\mathcal{X}_t(\mathcal{X}_t+\mathcal{N}_{i,t})] = \mathbb{E}[\mathcal{X}_t^2] \\ &= \gamma^2\mathbb{E}[\mathcal{X}_{t-1}^2] + \sigma_w^2.\end{aligned} \tag{3.19}$$

Then we calculate

$$\begin{aligned}\mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{i,t-1}] &= \mathbb{E}[\mathcal{X}_t(\alpha_{i,t-1}\hat{\mathcal{X}}_{i,t-2} + \beta_{i,t-1}\mathcal{Y}_{i,t-1} \\ &\qquad + \sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathcal{Z}_{j\to i,t-2})] \\ &= \alpha_{i,t-1}\mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{i,t-2} + \beta_{i,t-1}\mathbb{E}[\mathcal{X}_t\mathcal{Y}_{i,t-1}]] \\ &\qquad + \sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j\to i,t-2})] \\ &= \gamma^2\alpha_{i,t-1}\mathbb{E}[\mathcal{X}_{t-2}\hat{\mathcal{X}}_{i,t-2}] + \gamma\beta_{i,t-1}\mathbb{E}[\mathcal{X}_{t-1}^2] \\ &\qquad + \sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j\to i,t-2}]\end{aligned} \tag{3.20}$$

In order to complete (3.20), we need to calculate the term $\mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j\to i,t-2}]$. For that, we first introduce

$$\boldsymbol{h}_{i,0} = \gamma \begin{bmatrix}\beta_{j_1,0}\\ \vdots \\ \beta_{j_{\pi_i},0}\end{bmatrix}\mathbb{E}[\mathcal{X}_0^2].$$

Then, with this initialization, for any time t we find

$$\boldsymbol{h}_{i,t} = \gamma\begin{bmatrix}\beta_{j_1,t}\mathbb{E}[\mathcal{X}_t^2] + \boldsymbol{c}_{j_1,t}^T\boldsymbol{h}_{j_1,t-1}\\ \vdots \\ \beta_{j_{\pi_i},t}\mathbb{E}[\mathcal{X}_t^2] + \boldsymbol{c}_{j_{\pi_i},t}^T\boldsymbol{h}_{j_{\pi_i},t-1}\end{bmatrix} - \gamma\begin{bmatrix}c_{j_1,t}^{(i)}\\ \vdots \\ c_{j_{\pi_i},t}^{(i)}\end{bmatrix}\odot\begin{bmatrix}h_{j_1,t-1}^{(i)}\\ \vdots \\ h_{j_{\pi_i},t-1}^{(i)}\end{bmatrix},$$

where

$$\boldsymbol{c}_{j_1,t} = \left[c_{j_1,1}^{(k_1)} \dots c_{j_1,1}^{(k_{\pi_{j_1}})}\right]^T, \quad k \in \mathbf{N}_{j_1}.$$

Note that $\boldsymbol{h}_{i,t}$ can also be expressed as

$$\boldsymbol{h}_{i,t-1} = \begin{bmatrix} E[\mathcal{Z}_{j_1 \to i,t-1}\mathcal{X}_t] \\ \vdots \\ E[\mathcal{Z}_{j_{\pi_i} \to i,t-1}\mathcal{X}_t] \end{bmatrix}.$$

Using this introduced notation, we obtain

$$\begin{aligned} \mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j\to i,t-2}] &= \gamma\mathbb{E}[\mathcal{X}_{t-1}\mathcal{Z}_{j\to i,t-2}] \\ &= \gamma h_{i,t-1}^{(j)}. \end{aligned}$$

Therefore, we can finalize the calculation of term $\mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{t-1}]$ as

$$\begin{aligned} \mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{t-1}] = \gamma^2\alpha_{i,t-1}&\mathbb{E}[\mathcal{X}_{t-2}\hat{\mathcal{X}}_{t-2}] \\ &+ \gamma\beta_{i,t-1}\mathbb{E}[\mathcal{X}_{t-1}^2] + \gamma\boldsymbol{c}_{i,t-1}^T\boldsymbol{h}_{i,t-1} \end{aligned}$$

Additionally, we define the cross correlation term between the state and the estimation as

$$\begin{aligned} \tilde{\sigma}_{i,t}^2 &\triangleq \mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{i,t}] \\ &= \gamma\alpha_{i,t}\tilde{\sigma}_{i,t-1}^2 + \beta_{i,t}\mathbb{E}[\mathcal{X}_t^2] + \boldsymbol{c}_{i,t}^T\boldsymbol{h}_{i,t} \end{aligned}$$

and the variance for the underlying state as

$$\begin{aligned} \sigma_t^2 &\triangleq \mathbb{E}[\mathcal{X}_t^2] \\ &= \gamma^2\sigma_{t-1}^2 + \sigma_w^2, \end{aligned}$$

which concludes our calculation for the terms in $\boldsymbol{\Sigma}_{xd_i,t}$ such that

$$\begin{aligned} \boldsymbol{\Sigma}_{xd_i,t} &= \Big[\mathbb{E}[\mathcal{X}_t\mathcal{Y}_{i,t}]\ \mathbb{E}[\mathcal{X}_t\hat{\mathcal{X}}_{i,t-1}] \\ &\qquad \mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j_1\to i,t-1}] \cdots \mathbb{E}[\mathcal{X}_t\mathcal{Z}_{j_{\pi_i}\to i,t-1}]\Big]^T \\ &= \begin{bmatrix} \gamma^2\sigma_{t-1}^2 + \sigma_w^2 & \gamma\tilde{\sigma}_{i,t-1}^2 & \boldsymbol{h}_{i,t-1}^T \end{bmatrix}^T. \end{aligned}$$

Next, we calculate the terms of $\mathbf{\Sigma}_{dd_i,t}$. First we have

$$\begin{aligned}\mathbb{E}[\mathcal{Y}_{i,t}^2] &= \mathbb{E}[(\mathcal{X}_t + \mathcal{N}_{i,t})^2] \\ &= \sigma_t^2 + \sigma_{n_i}^2.\end{aligned} \tag{3.21}$$

then, for the term $\mathbb{E}[\hat{\mathcal{X}}_{i,t-1}\mathcal{Y}_{i,t}]$ we get

$$\begin{aligned}\mathbb{E}[\hat{\mathcal{X}}_{i,t-1}\mathcal{Y}_{i,t}] &= \mathbb{E}[\hat{\mathcal{X}}_{i,t-1}\mathcal{X}_t] \\ &= \gamma\tilde{\sigma}_{i,t-1}^2\end{aligned} \tag{3.22}$$

and note that we already found that $\mathbb{E}[\mathcal{Y}_{i,t}\mathcal{Z}_{j\to i,t-1}] = \boldsymbol{h}_{i,t-1}^{(j)}$.

We then calculate the terms that includes the estimation variable. We begin with defining

$$\begin{aligned}\hat{\sigma}_{i,t-1}^2 &\triangleq \mathbb{E}[\hat{\mathcal{X}}_{i,t-1}^2] \\ &= \mathbb{E}\Bigg[\Big(\alpha_{i,t-}\hat{\mathcal{X}}_{i,t-2} + \beta_{i,t-2}\mathcal{Y}_{i,t-1} \\ &\qquad + \sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathcal{Z}_{j\to i,t-2}\Big)^2\Bigg] \\ &= \alpha_{i,t-1}^2\underbrace{\mathbb{E}[\hat{\mathcal{X}}_{i,t-2}^2]}_{\hat{\sigma}_{i,t-2}^2} + 2\alpha_{i,t-1}\beta_{i,t-2}\underbrace{\mathbb{E}[\hat{\mathcal{X}}_{i,t-2}y_{i,t-1}]}_{\gamma\tilde{\sigma}_{i,t-2}^2} \\ &\quad + 2\alpha_{i,t-1}\sum_{j\in N_i} c_{i,t-1}^{(j)}E[\hat{\mathcal{X}}_{i,t-2}\mathcal{Z}_{j\to i,t-2}] \\ &\quad + \beta_{i,t-2}^2\underbrace{\mathbb{E}[\mathcal{Y}_{i,t-1}^2]}_{\sigma_{t-1}^2+\sigma_{n_i}^2} \\ &\quad + 2\beta_{i,t-2}\underbrace{\sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathbb{E}[\mathcal{Y}_{i,t-1}\mathcal{Z}_{j\to i,t-2}]}_{\gamma\boldsymbol{c}_{i,t-1}^T\boldsymbol{h}_{i,t-2}} \\ &\quad + \mathbb{E}\Bigg[\Big(\sum_{j\in\mathbf{N}_i} c_{i,t-1}^{(j)}\mathcal{Z}_{j\to i,t-2}\Big)^2\Bigg]\end{aligned} \tag{3.23}$$

and $\hat{\sigma}_{i,0}^2 = \beta_{i,0}^2(\sigma_0^2+\sigma_{n_i}^2)$. Note that we need to calculate the term $\mathbb{E}[\hat{\mathcal{X}}_{i,t-2}\mathcal{Z}_{j\to i,t-2}]$ in order to complete (3.23). For that, we first introduce a more compact form of

the term $\mathcal{Z}_{j\to i,t}$ such that

$$\begin{aligned}
\mathcal{Z}_{j\to i,t} &= \beta_{j,t}\mathcal{Y}_{j,t} + \sum_{k\in\mathbf{N}_j,k\neq i} c_{j,t}^{(k)}\Big(\beta_{k,t-1}\mathcal{Y}_{k,t-1} + \sum_{l\in\mathbf{N}_k,l\neq j} c_{k,t-1}^{(l)}\Big(\beta_{l,t-2}\mathcal{Y}_{l,t-2} + \sum\cdots\Big)\Big) \\
&= \Bigg[\overbrace{\beta_{j,t} + \frac{1}{\gamma}\sum_{k\in\mathbf{N}_j,k\neq i} c_{j,t}^{(k)}\Big(\beta_{k,t-1} + \frac{1}{\gamma}\sum_{l\in\mathbf{N}_k,l\neq j} c_{k,t-1}^{(l)}\Big(\beta_{l,t-2} + \frac{1}{\gamma}\sum\cdots\Big)\Big)}^{g_{i,t}^{(j)}}\Bigg]\mathcal{X}_t \\
&- \Bigg[\frac{1}{\gamma}\sum_{k\in\mathbf{N}_j,k\neq i} c_{j,t}^{(k)}\Big(\beta_{k,t-1} + \frac{1}{\gamma}\sum_{l\in\mathbf{N}_k,l\neq j} c_{k,t-1}^{(l)}\Big(\beta_{l,t-2} + \frac{1}{\gamma}\sum\cdots\Big)\Big)\Bigg]\mathcal{W}_{t-1} \\
&- \Bigg[\frac{1}{\gamma}\sum_{k\in N_j,k\neq i}\sum_{l\in N_k,l\neq j} c_{j,t}^{(k)}c_{k,t-1}^{(l)}\Big(\beta_{l,t-2} + \frac{1}{\gamma}\sum\cdots\Big)\Bigg]\mathcal{W}_{t-2} - \cdots - \big[\cdots\big]\mathcal{W}_{t-\kappa_i+1} + (\text{i.n.t.}),
\end{aligned} \tag{3.24}$$

where κ_i is the number of hops from the furthest agent and i.n.t. is the abbreviation of *independent noise terms*. Note that, in (3.24) we define the term $g_{i,t}^{(j)}$, which can be calculated in a recursive form. Using (3.24), we can write the term $\mathcal{Z}_{j\to i,t}$ as

$$\begin{aligned}
\mathcal{Z}_{j\to i,t} &= g_{i,t}^{(j)}\mathcal{X}_t - (g_{i,t}^{(j)} - \beta_{j,t})\mathcal{W}_{t-1} - \gamma\Big(g_{i,t}^{(j)} - \beta_{j,t} - \frac{1}{\gamma}\sum_{k\in N_j,k\neq i} c_{j,t}^{(k)}\beta_{k,t-1}\Big)\mathcal{W}_{t-2} \\
&\quad - \cdots - \big(\cdots\big)\mathcal{W}_{t-\kappa_i+1} + (\text{i.n.t.})
\end{aligned} \tag{3.25}$$

and we obtain

$$\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{Z}_{j\to i,t}] = g_{i,t}^{(j)}\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{X}_t] - (g_{i,t}^{(j)} - \beta_{j,t})\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{W}_{t-1}] - \cdots - \big(\cdots\big)\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{W}_{t-\kappa_i+1}].$$

We note that state noise $\mathcal{W}_t$ is independent from previous states and we can express the term $\mathcal{X}_{i,t}$ as

$$\begin{aligned}
\hat{\mathcal{X}}_{i,t} &= \beta_{i,t}\mathcal{W}_{t-1} + \Big(\alpha_{i,t}\beta_{i,t-1} + \sum_{j\in N_i} c_{i,t}^{(j)}\beta_{j,t-1}\Big)\mathcal{W}_{t-2} \\
&\quad + \Big(\alpha_{i,t}\alpha_{i,t-1}\beta_{i,t-2} + \alpha_{i,t}\sum_{j\in N_i} c_{i,t}^{(j)}\beta_{j,t-2} + \sum_{j\in N_i}\sum_{k\in N_j,k\neq i} c_{i,t}^{(j)}c_{j,t-1}^{(k)}\beta_{k,t-2}\Big)\mathcal{W}_{t-3} \\
&\quad + \cdots + \big(\cdots\big)\mathcal{W}_{t-\kappa_i+1} + \text{i.n.t.}.
\end{aligned} \tag{3.26}$$

Therefore, we conclude that

$$\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{Z}_{j\to i,t}] = g_{i,t}^{(j)}\mathbb{E}[\hat{\mathcal{X}}_{i,t}\mathcal{X}_t] + A\sigma_w^2,$$

where the term A is calculated according to recursions in (3.25) and (3.26). Finally, we calculate the term remaining terms of $\mathbf{\Sigma}_{dd_i,t}$ as

$$\begin{aligned}
\mathbb{E}\left[\left(\mathcal{Z}_{j\to i,t-1}\right)^2\right] =& (g_{i,t-1}^{(j)^2})\sigma_{t-1}^2 + \Bigg[\Big((g_{i,t-1}^{(j)} - \beta_{j,t-1})^2 \\
&- \big(g_{i,t-1}^{(j)} - \beta_{j,t-1} - \frac{1}{\gamma}\sum_{k\in N_j, k\neq i} c_{j,t-1}^{(k)}\beta_{k,t-2}\big)\Big)^2 - B_{j,4}^2 - B_{j,5}^2 \cdots - B_{j,\kappa_i}^2\Bigg]\sigma_w^2 \\
&+ (\beta_{j,t-1})^2\sigma_{n_j} + \sum_{k\in N_j, k\neq i}(c_{j,t-1}^{(k)}\beta_{k,t-2})\sigma_{n_k}^2 \\
&+ \sum_{k\in N_j, k\neq i}\sum_{l\in N_k, l\neq j}(c_{j,t-1}^{(k)}c_{k,t-2}^{(l)}\beta_{k,t-2}\beta_{l,t-3})^2\sigma_{n_l}^2 + \cdots \\
&+ \sum_{k\in N_j, k\neq i}\sum_{l\in N_k, l\neq j}\cdots\sum_{r\in N_i^{(\kappa_i-3)}}\sum_{s\in N_i^{(\kappa_i-2)}}\sum_{m\in N_i^{(\kappa_i-1)}, m\neq r}(c_{j,t-1}^{(k)}c_{k,t-2}^{(l)}\cdots c_{s,t-\kappa_i}^{(m)} \\
&\times \beta_{k,t-2}\beta_{l,t-3}\cdots\beta_{m,t-\kappa_i})^2\sigma_m^2
\end{aligned} \tag{3.27}$$

and

$$\begin{aligned}
\mathbb{E}\Big[\mathcal{Z}_{j_o\to i,t-1}\mathcal{Z}_{j_p\to i,t-1}\Big] =& g_{i,t-1}^{j_o}g_{i,t-1}^{j_p}\sigma_{t-1}^2 + \Big[(g_{i,t-1}^{(j_o)} - \beta_{j_o,t-1})(g_{i,t-1}^{(j_p)} - \beta_{j_p,t-1}) \\
&- \big(g_{i,t-1}^{(j_o)} - \beta_{j_o,t-1} - \frac{1}{\gamma}\sum_{k\in N_{j_o}, k\neq i} c_{j_o,t-1}^{(k)}\beta_{k,t-2}\big) \\
&\times \big(g_{i,t-1}^{(j_p)} - \beta_{j_p,t-1} - \frac{1}{\gamma}\sum_{k\in N_{j_p}, k\neq i} c_{j_p,t-1}^{(k)}\beta_{k,t-2}\big) \\
&- B_{j_o,4}B_{j_p,4} - \cdots - B_{j_o,\kappa_i}B_{j_p,\kappa_i}\Big]\sigma_w^2,
\end{aligned} \tag{3.28}$$

where $o, p \in \{1, \ldots, \pi_i\}, o \neq p$ and $B_{j,t}$ for $t = 4, \ldots, \kappa - i$, represents the remaining recursive terms derived in (3.25).

Next, we find the optimal information combination weights in (3.16). We define the vector containing the parameters such that

$$\boldsymbol{P} \triangleq \left[\tilde{\alpha}_{i,t} \;\; \beta_{i,t} \;\; c_{i,t}^{(j_1)} \;\; \cdots \;\; c_{i,t}^{(j_{\pi_i})}\right]^T.$$

Algorithm 3 The Efficient and Optimal Distributed Online Learning Algorithm (EODL)

for $i = 1$ **to** m **do**
  $\hat{x}_{i,0} = \bar{x}$
  $\hat{\sigma}^2_{i,0} = \sigma^2_0$
end for
for $t \geq 1$ **do**
  for $i = 1$ **to** m **do**
    Receive $\{z_{j,t-1}\}_{j \in \mathbf{N}_i}$
    Extract Innovation
    $z_{j \to i,t-1} = z_{j,t-1} - c^{(i)}_{j,t-1} + c^{(i)}_{j,t-1} c^{(j)}_{i,t-2} z_{j \to i,t-3}$
    Calculate $\boldsymbol{\Sigma}_{xd_i,t}$,$\boldsymbol{\Sigma}_{dd_i,t}$
    Find Parameters:
    $\boldsymbol{P} \triangleq [\tilde{\alpha}_{i,t}\ \beta_{i,t}\ c^{(j_1)}_{i,t}\ \cdots\ c^{(j_{\pi_i})}_{i,t}]^T$
    $\boldsymbol{P} \leftarrow \boldsymbol{\Sigma}^{-1}_{dd_i,t} \boldsymbol{\Sigma}_{xd_i,t}$
    $\alpha_{i,t} = \gamma - \gamma\beta_{i,t} - \sum_{j \in N_i} c^{(j)}_{i,t} g^{(j)}_{i,t}$
    Update:
      $\hat{x}_{i,t} = \alpha_{i,t}\hat{x}_{i,t-1} + \beta_{i,t} y_{i,t}$
        $+ \sum_{j \in N_i} c^{(j)}_{i,t} z_{j \to i,t-1}$
      $\hat{\sigma}^2_{i,t} = \gamma^2 \hat{\sigma}^2_{i,t-1} + \sigma^2_w - \boldsymbol{\Sigma}^T_{xd_i,t} \boldsymbol{\Sigma}^{-1}_{dd_i,t} \boldsymbol{\Sigma}_{xd_i,t}$
  end for
end for

Note that in (3.13), all the conditioned parameters are jointly Gaussian with the state. Therefore, we calculate the parameter vector as $\boldsymbol{P} = \boldsymbol{\Sigma}^{-1}_{dd_i,t} \boldsymbol{\Sigma}_{xd_i,t}$ and the estimation and variance recursions are then given by

$$\hat{x}_{i,t} = \gamma \hat{x}_{i,t-1} + \beta_{i,t}\big(y_{i,t} - \gamma \hat{x}_{i,t-1}\big) + \sum_{j \in N_i} c^{(j)}_{i,t} \Big(z_{j \to i,t-1} - g^{(j)}_{i,t} \hat{x}_{i,t-1} \Big),$$

$$\hat{\sigma}^2_{i,t} = \gamma^2 \hat{\sigma}^2_{i,t-1} + \sigma^2_w - \boldsymbol{\Sigma}^T_{xd_i,t} \boldsymbol{\Sigma}^{-1}_{dd_i,t} \boldsymbol{\Sigma}_{xd_i,t}.$$

Hence, for the parameter $\alpha_{i,t}$ in (3.16), we have

$$\alpha_{i,t} = \gamma - \gamma\beta_{i,t} - \sum_{j \in N_i} c^{(j)}_{i,t} g^{(j)}_{i,t},$$

which finalizes the efficient and optimal distributed online learning algorithm. We give the detailed explanation of the overall algorithm in Algorithm 3.

In the sequel, we provide numerical examples to evaluate the performance of our algorithm against several other distributed learning algorithms under different schemes.

3.4 Simulations

In this section, we study the performance of the proposed algorithm under different scenarios. For the network structure, we consider a depth 2 tree-network. Each agent i observes a noise corrupted version $y_t \in \mathbb{R}$ of an underlying state $x_t \in \mathbb{R}$, where it evolves according to the random walk model in (3.2) with $\gamma = 0.98$. The state noise w_t is driven by a Gaussian process, with zero mean and variance $\sigma_w^2 = 0.025$. The observation noise is also zero-mean white Gaussian random process. We define MSE performance measure for the overall network as $MSE = \sum_{i=1}^{m} MSE(i)$, where $MSE(i) = \sum_{t=1}^{T} \mathbb{E}[(\mathcal{X}_t - \hat{x}_{i,t})^2]$.

We use ensemble average over 200 experiments in order to approximate the global MSE measure and select time horizon $T = 4000$ for the all experiments. We compare the performance of the proposed algorithm with diffusion least mean squares (D-LMS) and diffusion recursive least squares (D-RLS) algorithm and diffusion implementation of the Kalman filtering algorithm (D-Kalman) under different settings. We also use consensus distributed algorithm in our comparison framework [14, 13]. We implement the diffusion distributed algorithms with adapt-then-combine (ATC) technique, where each agent first makes an estimate based on their local observation and discloses this estimates [13]. Then, agents decides on their final estimate by combining the local and received estimations. For the combination step, we use the Metropolis rule, where the combination weight $\lambda_{i,j}$ for the estimate coming to agent i from agent j is calculated as

$$\lambda_{i,j} = \begin{cases} \frac{1}{\max(\mathbf{N}_i, \mathbf{N}_j)} & \text{if } i \neq j \text{ are linked,} \\ 0 & \text{for } i \text{ and } j \text{ not linked,} \\ 1 - \sum_{j \in \mathcal{N}_i \setminus i} \lambda_{i,j} & \text{for } i = j. \end{cases}$$

We set the learning rates of diffusion LMS and consensus algorithm to $\mu = 0.2$ and

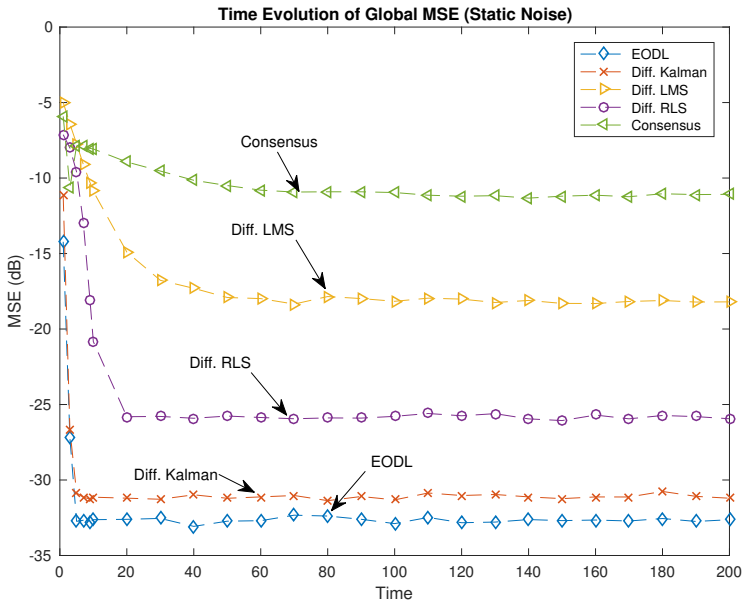

Figure 3.3: Comparison of global MSE of algorithms under space-invariant noise with $\gamma = 0.98$.

also we set the memory parameter of diffusion RLS algorithm to $\eta = 0.3$. Note that we set the memory parameter of RLS algorithm low in order to put more emphasis on the recent observations and estimations. We choose this parameter so that the diffusion RLS algorithm converges fast, but still be able to track the underlying dynamic parameter.

In Fig.3.3, we compare the algorithms under space-invariant noise over the network, where each agent experiences the same level of disturbance. We select the standard deviation of the observation noise $\sigma_{n_i} = 0.0467$ for all $i = 1, \ldots, m$. We observe that the proposed algorithm (EODL) achieves a superior performance regarding the global MSE measure and the convergence rate compared to other distributed estimation algorithms. Note that the consensus algorithm performs the worst since it has a decaying learning rate as the nodes reach to consensus with time and the network loses its adaptation capabilities against a dynamic parameter.

In other scenario, we evaluate the MSE performance of the algorithms under

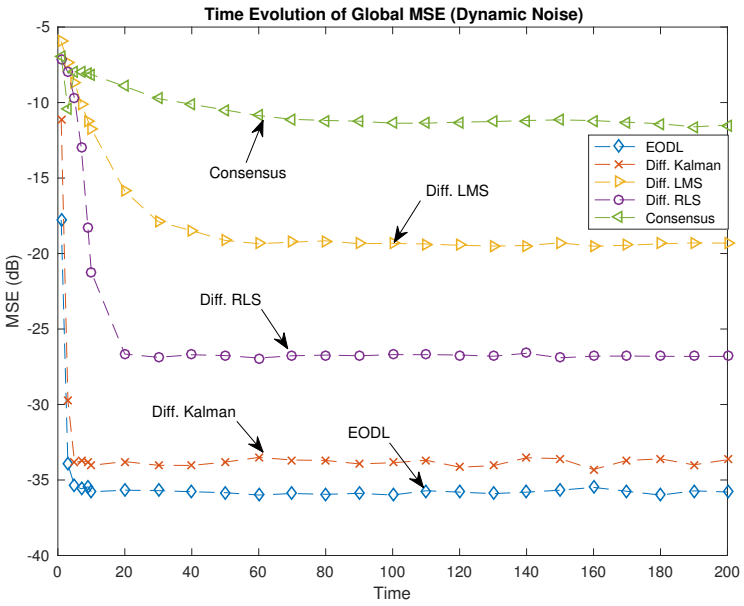

Figure 3.4: Comparison of global MSE of algorithms under space-variant noise with $\gamma = 0.98$.

space-variant noise statistics over the network. For this case, we generate the the observation noise of different agents randomly from a normal distribution with 0.01 mean and 0.02 variance. In Fig.3.4, we compare the algorithms for the space-variant noise case. Note that the algorithms perform better than the space-invariant noise statistics case. This is because, in this case some of the agents may experience smaller noise levels while some may experience higher, but through the communication between agents, they all can benefit from the estimations of the agents having better observation channels. We also emphasize that EODL algorithm even performs better in this case since it can utilize the information disclosure and estimation construction better than the other algorithms. Furthermore, we observe a similar performance between the compared algorithms, where EODL algorithm achieves a superior performance regarding the global MSE measure and the convergence rate compared to other distributed estimation schemes.

We also investigate the effect of the random walk parameter γ with a simulation under the space-variant noise framework. In this case, we select $\gamma = 1$. In

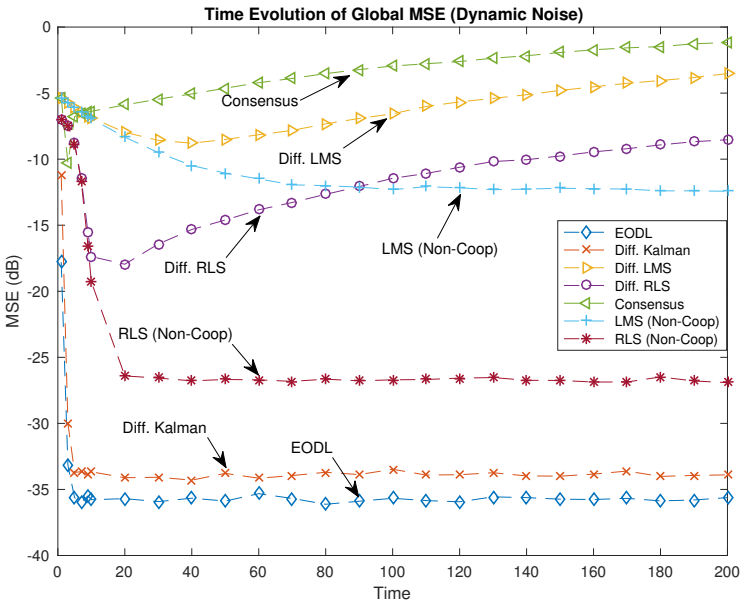

Figure 3.5: Comparison of global MSE of algorithms under space-variant noise with $\gamma = 1$. Even if D-LMS and D-RLS are stable when there is no cooperation among nodes, the networks are diverging.

Fig.3.5, we presented both the behavior of individual nodes and the network itself for RLS and LMS algorithms. We observe that D-LMS, D-RLS and consensus networks become unstable, even if individual nodes are stable and able to track the underlying parameter. Note that only D-Kalman and EODL networks are able to achieve good convergence and steady-state performance for $\gamma = 1$ case. We also emphasize that, with EODL algorithm, we produce the optimal parameters and the combination weights for the network in contrast to D-LMS, D-RLS and consensus algorithms, where we need to select their parameters beforehand. Therefore, the proposed algorithm overcomes the issue of parameter selection and provides more stable solution for the problem presented here.

Chapter 4

Conclusion

In this thesis, we study online learning over distributed networks. In this framework, we consider a network agents that collaboratively try to estimate an underlying parameter or event. However, the cooperation among the nodes can inflict high communication load on the network and processing a high amount of fast streaming data over the agent may require high computational power, where these resources are scarce on such systems. Therefore, for the applicability of these systems, both computation and communication wise efficient structures are required while maintaining a comparable performance. To this end, in this thesis, we introduced powerful distributed online learning algorithms that performs superior compared to the state-of-the-art distributed structures while efficient using the resources of the network.

In the first part of the thesis, we introduce a novel distributed learning algorithm for centralized networks in order to reduce the communication and computation demand of such approaches. Since the ordinary centralized learning systems send significant amount of information about their estimation from peripheral nodes to the central node, they are impractical in applications involving big data due to complexity issues. To this end, by using the SMF, we significantly reduce the communication and computation complexity of these approaches while providing superior performance. We provide unconstrained, affine and convex

combination weight configurations at the central node using set membership filtering framework.

Through numerical experiments in stationary and non-stationary environments and through regression of a benchmark real life problem, we investigate the steady-state mean square error and convergence rate performance of these algorithms compared with other centralized methods and ordinary learning systems. In these experiments we demonstrate that proposed algorithms can reach faster convergence rate and lower steady state error. Finally, we show that our set membership filtering based approaches require less update operations both on the peripheral nodes and the central node hence less communication and computation load on the network than the compared algorithms.

In the second part of the thesis, we introduce a novel approach for distributed estimation framework to produce optimal and efficient estimation schemes. Although the distributed learning algorithms have extensively studied and applied from modeling of highly complex structures to effectively processing of fast streaming big data, there are still unresolved issues for disclosure and utilization of the information among agents in the network.

To this end, we introduce a distributed online learning algorithm that achieves MMSE performance in the estimation of dynamic parameters by exploiting optimal information disclosure and estimation construction. We prove that achieving MMSE performance via aggregation of information is possible over an arbitrary network topology. Also, we show that using the aggregation of information imposes huge communication load on the network and requires excessive storage on agents. Therefore, we propose an efficient method, where agents only disclose the new information and construct estimations using innovation sequences. We derive an iterative algorithm to extract innovations from disclosed information and construct new estimates based on accessed information for each agent. We also prove that such method is only applicable over certain network topologies, i.e. tree-networks.

Finally, through series of simulations, we show that the proposed method

achieves superior performance compared to other state-of-the-art distributed estimation algorithms in terms of convergence rate and global MSE. We observe that under spatially variant noise statistics, the proposed method performs better due to efficient information disclosing and utilization capabilities of the algorithm. We also show that while the most of the distributed estimation networks diverge in case of tracking highly non-stationary parameter, the EODL algorithm can still achieve stable solutions and superior performance regarding global MSE and convergence rate compared to other state-of-the-art distributed estimation schemes.

Bibliography

[1] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, 2013.

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[3] S. Shahrampour, S. Rakhlin, and A. Jadbabaie, "Online learning of dynamic parameters in social networks," in *Advances in Neural Information Processing Systems*, 2013.

[4] J. N. Tsitsiklis, "Problems in decentralized decision making and computation.," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND DECISION SYSTEMS, 1984.

[5] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[6] D. Chaudhary, S. Nayse, and L. Waghmare, "Application of wireless sensor networks for greenhouse parameter control in precision agriculture," *International Journal of Wireless & Mobile Networks (IJWMN) Vol*, vol. 3, no. 1, pp. 140–149, 2011.

[7] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing,*

2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on, vol. 4, pp. 2033–2036, IEEE, 2001.

[8] F. S. Cattivelli and A. H. Sayed, "Diffusion lms strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.

[9] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[10] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4795–4810, 2010.

[11] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, 2008.

[12] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, 2012.

[13] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 3, pp. III–917, IEEE, 2007.

[14] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed kalman filtering: formulation and performance analysis," *Proc. Cognitive Information Processing*, pp. 36–41, 2008.

[15] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[16] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed lms for consensus-based in-network adaptive processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, 2009.

[17] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed, "Steady-state MSE performance analysis of mixture approaches to adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4050–4063, 2010.

[18] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NJ: John Wiley & Sons, 2003.

[19] M. O. Sayin, N. D. Vanli, and S. S. Kozat, "A transient analysis of affine mixtures," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 6227–6232, 2011.

[20] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed, "A transient analysis of affine mixtures," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 6227–6232, 2011.

[21] M. A. Donmez and S. S. Kozat, "Steady-state MSE analysis of convexly constrained mixture methods," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 3314–3321, 2012.

[22] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Transactions on Signal Processing*, vol. 54, no. 3, pp. 1078–1090, 2006.

[23] J. Arenas-Garcia, V. Gomez-Verdejo, and A. R. Figueiras-Vidal, "New algorithms for improved adaptive convex combination of LMS transversal filters," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 6, pp. 2239–2249, 2005.

[24] J. Arenas-Garcia, M. Martinez-Ramon, V. Gomez-Verdejo, and A. R. Figueiras-Vidal, "Multiple plant identifier via adaptive LMS convex combination," in *2003 IEEE International Symposium on Intelligent Signal Processing*, pp. 137–142, 2003.

[25] A. C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2685–2699, 1999.

[26] S. Gollamudi, S. Nagaraj, S. Kapoor, and Y. F. Huang, "Set-membership filtering and a set-membership normalized LMS algorithm with an adaptive step size," *IEEE Signal Processing Letters*, vol. 5, no. 5, 1998.

[27] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *The Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[28] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 168–176, ACM, 2006.

[29] S. Barbarossa and G. Scutari, "Bio-inspired sensor network design," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 26–35, 2007.

[30] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4185–4190, IEEE, 2008.

[31] D. Shah *et al.*, "Gossip algorithms," *Foundations and Trends® in Networking*, vol. 3, no. 1, pp. 1–125, 2009.

[32] S. Kar and J. M. Moura, "Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 674–690, 2011.

[33] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 8, pp. 1504–1510, 2007.

[34] A. H. Sayed and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*, pp. 233–237, IEEE, 2006.

[35] M. O. Sayin, N. D. Vanli, I. Delibalta, and S. S. Kozat, "Optimal and efficient distributed online learning for big data," in *Big Data (BigData Congress), 2015 IEEE International Congress on*, pp. 126–133, IEEE, 2015.

[36] D. Acemoglu, A. Nedic, and A. Ozdaglar, "Convergence of rule-of-thumb learning rules in social networks," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 1714–1720, IEEE, 2008.

[37] R. Frongillo, G. Schoenebeck, and O. Tamuz, "Social learning in a changing world," *Internet and Network Economics*, pp. 146–157, 2011.

[38] H. Zhao and Y. Yu, "Novel adaptive vss-nlms algorithm for system identification," in *Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference on*, pp. 760–764, June 2013.

[39] J. Alcala-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. Garca, L. Snchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[40] A. H. Sayed, *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[41] B. D. Anderson and J. B. Moore, "Optimal filtering," *Englewood Cliffs*, vol. 21, pp. 22–95, 1979.

[42] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5308–5323, 2014.

[43] M. O. Sayin and S. S. Kozat, "Single bit and reduced dimension diffusion strategies over distributed networks," *IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 976–979, 2013.

[44] P. Humblet, "A distributed algorithm for minimum weight directed spanning trees," *IEEE Transactions on Communications*, vol. 31, no. 6, pp. 756–762, 1983.

[45] M. Khan, G. Pandurangan, and V. A. Kumar, "Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 124–139, 2009.

[46] D. Peleg, *Distributed computing: a locality-sensitive approach*. SIAM, 2000.

[47] M. Elkin, "An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem," *SIAM Journal on Computing*, vol. 36, no. 2, pp. 433–456, 2006.