Sequential Nonlinear Learning for Distributed Multiagent Systems via Extreme Learning Machines

Nuri Denizcan Vanli, Muhammed O. Sayin, Ibrahim Delibalta, and Suleyman Serdar Kozat, *Senior Member, IEEE*

***Abstract*— We study online nonlinear learning over distributed multiagent systems, where each agent employs a single hidden layer feedforward neural network (SLFN) structure to sequentially minimize arbitrary loss functions. In particular, each agent trains its own SLFN using only the data that is revealed to itself. On the other hand, the aim of the multiagent system is to train the SLFN at each agent as well as the optimal centralized batch SLFN that has access to all the data, by exchanging information between neighboring agents. We address this problem by introducing a distributed subgradient-based extreme learning machine algorithm. The proposed algorithm provides guaranteed upper bounds on the performance of the SLFN at each agent and shows that each of these individual SLFNs asymptotically achieves the performance of the optimal centralized batch SLFN. Our performance guarantees explicitly distinguish the effects of data- and network-dependent parameters on the convergence rate of the proposed algorithm. The experimental results illustrate that the proposed algorithm achieves the oracle performance significantly faster than the state-of-the-art methods in the machine learning and signal processing literature. Hence, the proposed method is highly appealing for the applications involving big data.**

***Index Terms*— Distributed systems, extreme learning machine (ELM), multiagent optimization, sequential learning, single hidden layer feedforward neural networks (SLFNs).**

I. Introduction

A. Preliminaries

THE demand for neural network inspired learning structures is steadily growing owing to their superior nonlinear modeling power and low complexity [1]–[3]. Although several neural-adaptive learning methods [1]–[5] are used for processing data in a centralized manner, the steadily increasing growth of the data sizes (in terms of both dimensionality and length) prohibits centralized processing due to computational complexity, storage, and communication issues [6], [7]. To address this problem, several distributed learning algorithms are proposed in the machine learning and signal processing literature [8]–[13]. Although these algorithms are shown to achieve certain statistical and deterministic convergence rates, they are usually based on linear models, which significantly limit their performance in real-life applications [14].

In this paper, we resolve these issues by introducing a sequential nonlinear kernel-adaptive learning algorithm with guaranteed convergence bounds without any statistical assumptions. In particular, we propose a distributed single hidden layer feedforward neural network (SLFN) structure with strong theoretical convergence guarantees in a deterministic sense, i.e., without any statistical assumptions on the data. Our algorithm builds upon the existing centralized extreme learning machine (ELM)-based methods in a novel manner by: 1) providing a rigorous distributed formulation of the neural network-based optimization problem and 2) introducing a low-complexity algorithm to achieve the performance of the optimal centralized ELM-based method. Through extensive set of simulations and real-life experiments, we demonstrate significant performance gains achieved by the proposed algorithm with respect to the conventional distributed learning algorithms in the signal processing and machine learning literature.

B. Prior Art and Comparison

For applications involving high levels of complexity and nonlinearity, several SLFN-based algorithms are introduced for accurate nonlinear modeling [1], [15]–[18]. In order to train SLFNs, a popular method is to use the ELM algorithm [1]. In particular, Huang *et al.* [1] illustrate the sufficient initialization conditions to train the SLFNs, which significantly improve the convergence speed of the SLFN-based algorithms. Liang *et al.* [2] have extended these ideas to sequential learning problems. Recently, several variants of the ELM method are proposed for various learning tasks and cost functions. Specifically, Balasundaram *et al.* [5] considered the one-norm minimization and introduced a method to generalize the ELM algorithm for this cost measure. In [19], the ELM algorithm is used for ranking problems, and in [20], hierarchical structures are incorporated into the ELM framework to process high-dimensional data. Our work builds upon these existing works in the literature by providing a scalable distributed method, over which any such ELM-based algorithm can be applied. Furthermore, the computational complexity of the proposed algorithm is linear in the number

Manuscript received April 30, 2015; accepted February 26, 2016. Date of publication March 11, 2016; date of current version February 15, 2017. This work was supported in part by the Turkish Academy of Sciences Outstanding Researcher Programme within the Scientific and Technological Research Council of Turkey under Contract 113E517 and Contract 115E917 and in part by Türk Telekom Laboratories.

N. D. Vanli, M. O. Sayin, and S. S Kozat are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: vanli@ee.bilkent.edu.tr; sayin@ee.bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

I. Delibalta is with AveaLabs, Avea Communications Services Inc., Istanbul 34367, Turkey, and also with the Graduate School of Social Sciences and Humanities, Koç University, Istanbul 34450, Turkey (e-mail: ibrahim.delibalta@avea.com.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2536649

of hidden nodes for each agent, whereas it is quadratic for the original ELM method [2]. Therefore, the proposed algorithm is highly appealing for applications involving big data, since the introduced algorithm directly decreases: 1) the amount of data to be processed at each agent and 2) the computational complexity of the processing algorithms at each agent.

Distributed learning and optimization problems are extensively studied in the context of signal processing [8]–[10] and optimization [11]–[13], [21], [22] due to their natural modeling abilities of underlying phenomenon. In the distributed setting, the authors use linear models to capture the relationship between the input and output data [8]–[10]. However, the convergence results in these papers are only valid under certain statistical assumptions on the data, which do not usually hold in real-life applications. In this paper, we do not commit to linear models, but instead use an SLFN-based structure, which has shown to elegantly model the complex nonlinear relationships [1]. Furthermore, our analyses hold in an individual sequence manner without any statistical assumptions on the data, unlike [8]–[10].

We emphasize that the conventional distributed multiagent optimization algorithms [11]–[13], [21], [22] usually have linear structures in general, similar to [8]–[10]. On the other hand, these methods are mainly based on subgradient methods, and the theoretical analyses are carried out from a deterministic perspective. Yet, these subgradient methods may not provide satisfactory performance if the cost function is nondifferentiable at certain points. This holds, since these nondifferentiable points are usually the desired solutions of the optimization problems (e.g., consider the regularized one-norm loss). Therefore, in this paper, we do not directly use a subgradient-based approach, but instead use a splitting method [23]–[25] to ensure that our algorithm attains such nondifferentiable values. In this manner, our algorithm achieves a considerably smaller accumulated error and illustrates a better convergence performance as shown in our experiments.

C. Contributions

Our main contributions are as follows.

1) We introduce a sequential nonlinear optimization algorithm over distributed multiagent learning systems. Here, the multiagent structure optimizes the SFLNs for both additive and radial basis function kernels in a fully distributed manner. The proposed algorithm is truly sequential, such that it processes each new data pair and update the SLFN model without the knowledge of the time horizon.
2) We show that by our diffusion scheme, each agent can successfully and uniformly optimize the SFLN weights to minimize the overall network cost (over the entire data) with observing only a portion of the data. We demonstrate this result in a deterministic sense without any statistical assumptions on the data, such that our results are guaranteed to hold uniformly for all input and output sequences.
3) We achieve this performance with a computational complexity only linear in the data length. Thus, our algorithm can be efficiently used in applications involving big data.
4) We demonstrate the significant performance gains achieved by our algorithm over numerical examples and benchmark real data sets.

D. Organization of This Paper

The organization of this paper is as follows. In Section II, we introduce the multiagent learning problem and use the SLFNs as well as the forward–backward splitting method to derive a sequential distributed optimization algorithm. We then analyze the convergence performance of the introduced algorithm and provide guaranteed regret bounds in Sections III and IV. In Section V, we compare the performance of our algorithm with respect to the state-of-the-art algorithms over benchmark data sets. This paper concludes with several remarks in Section VI.

II. Problem Description

A. Notation

Throughout this paper, all vectors are column vectors and represented by boldface lowercase letters. Matrices are represented by boldface uppercase letters. For a matrix $\boldsymbol{H}$, $||\boldsymbol{H}||_F$ is the Frobenius norm. For a vector $\boldsymbol{x}$ (and matrix $\boldsymbol{H}$), $||\boldsymbol{x}||$ (and $||\boldsymbol{H}||$) is the ℓ^2-norm. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y}$ is the inner product. Here, $\mathbf{0}$ (and $\mathbf{1}$) denotes the vector with all zeros (and ones), and the dimensions can be understood from the context. For a matrix $\boldsymbol{H}$, $\boldsymbol{H}_{jk}$ represents its entry at the jth row and the kth column.

B. System Overview

We study the distributed sequential training of SLFN structures, which can be used in various applications, including nonlinear optimization, regression, and classification. As an example, in wireless sensor networks, a number of agents observe different data sequences related to (or generated by) a phenomenon of interest. In a centralized approach to this problem (see [2] and references therein), the entire data sequences are required to be processed by a single centralized agent. To this end, each agent transmits its observations into a centralized processor, where a sequential algorithm, e.g., online ELM [2] is applied. However, this centralized structure cannot process the data in a truly sequential manner, since the data are delayed due to transmission and then processed in chunks at the centralized processor. Furthermore, each agent may have the capability of generating huge volumes of data due to the recent developments in hardware technologies [8]. Therefore, centralized processing of such huge amounts of data may not be feasible or even possible due to the transmission, computational complexity, and storage issues [10].

To resolve these problems, we introduce a distributed framework, where each agent sequentially processes its own data and shares the extracted information with its neighbors, as shown in Fig. 1. Thus, the data are processed in a truly sequential and completely decentralized manner. Nevertheless, as rigorously shown in Section IV, the proposed distributed sequential method achieves asymptotically the same performance with its optimal centralized batch variant.

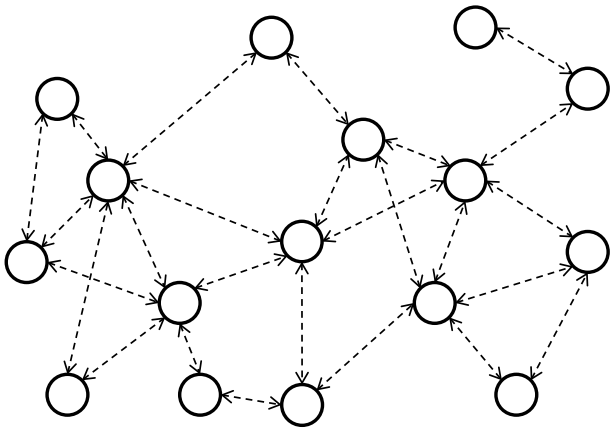

Fig. 1. Example multiagent network. Each agent is connected to and communicates with a set of other agents, which form its neighborhood. The network is irreducible and aperiodic, such that each node is achievable from any other node at irregular times.

More formally, each agent $k \in \{1, \ldots, K\}$ observes a pair of input vector and target data synchronously at each time t, i.e., $(\boldsymbol{x}_{t,k}, y_{t,k})$, where $\boldsymbol{x}_{t,k} \in \mathbb{R}^m$ and $y_{t,k} \in \mathbb{R}$. In real-life applications, the network may be subject to several sources of uncertainties, such as asynchronous data arrivals, topology changes, and link or node failures. For the brevity of this paper, we do not incorporate such uncertainties into our framework and refer to the relevant papers in the literature [26]–[31], where the effects of such uncertainties to the performance of the first-order methods are thoroughly analyzed. In this framework, the aim of each agent is to learn the relationship between these input vector and target data sequences. This learning process is performed in a sequential manner, such that after $y_{t,k}$ is predicted and its true value is revealed, each agent updates its prediction model in order to more accurately predict the next sample $y_{t+1,k}$. In many real-life applications, this relationship between the input vectors and the target data is highly nonlinear [14]. Hence, in order to effectively capture such nonlinear relationships, at each agent k, we use an SLFN with N hidden nodes and estimate the target data at each agent $k \in \{1, \ldots, K\}$ as

$$\hat{y}_{t,k} = \sum_{n=1}^{N} w_{t,n,k}\, z(\boldsymbol{a}_{n,k}, b_{n,k}; \boldsymbol{x}_{t,k}) \tag{1}$$

where for the kth agent, $w_{t,n,k}$ is the weight connecting the nth hidden node to the output node, $z(\boldsymbol{a}_{n,k}, b_{n,k}; \boldsymbol{x}_{t,k})$ is the output of the nth hidden node for the input $\boldsymbol{x}_{t,k}$, $\boldsymbol{a}_{n,k} \in \mathbb{R}^m$ is the weight vector connecting the input layer to the nth hidden node, and $b_{n,k} \in \mathbb{R}$ is the bias parameter of the nth hidden node. As shown in [2], if the number of hidden nodes is sufficiently large, we can randomly pick the weight vector $\boldsymbol{a}_{n,k}$ and the bias parameter $b_{n,k}$ without any performance degradation for infinitely differentiable activation functions $z(\cdot, \cdot; \cdot)$. Thus, we randomly generate a weight vector $\boldsymbol{a}_n$ and a bias parameter b_n, then use these parameters at each agent, i.e., we set $\boldsymbol{a}_{n,k} = \boldsymbol{a}_n$ and $b_{n,k} = b_n$, $\forall k \in \{1, \ldots, K\}$.[1] Then, we can rewrite (1) in the vector form as

$$\hat{y}_{t,k} = \langle \boldsymbol{w}_{t,k}, \boldsymbol{z}_{t,k} \rangle \tag{2}$$

where $\boldsymbol{z}_{t,k} \triangleq [z(\boldsymbol{a}_{1,k}, b_{1,k}; \boldsymbol{x}_{t,k}), \ldots, z(\boldsymbol{a}_{N,k}, b_{N,k}; \boldsymbol{x}_{t,k})]^T$ and $\boldsymbol{w}_{t,k} \triangleq [w_{t,1,k}, \ldots, w_{t,N,k}]^T$.

In the distributed setting, the aim of each agent is to minimize a cost function $f_{t,k}(\cdot) + r_k(\cdot)$ at time t over a convex set $\mathcal{W}$, where $f_{t,k}(\cdot)$ and $r_k(\cdot)$ are the generic convex functions. Here, for the kth agent, the function $f_{t,k}(\cdot)$ is an empirical loss, and the function $r_k(\cdot)$ is a regularization term. As an example, by setting $f_{t,k}(\boldsymbol{w}) = (y_t - \langle \boldsymbol{w}, \boldsymbol{z}_{t,k} \rangle)^2$ and $r_k(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_p$, we obtain the regularized least squares cost function. We emphasize that we consider this generic cost function, since we can train the ELMs using different cost functions instead of the squared error loss depending on the application. Our approach covers these different cases and models a broad range of practical cost functions that are widely used in optimization [32], machine learning [33], and signal processing literature [34]. In a more formal manner, the aim of each agent is to sequentially solve the following optimization problem:

$$\min \sum_{t=1}^{T} [f_{t,k}(\boldsymbol{w}_{t,k}) + r_k(\boldsymbol{w}_{t,k})] \tag{3}$$

$$\text{s.t. } \boldsymbol{w}_{t,k} \in \mathcal{W} \tag{4}$$

for any T, where T is the length of the data, which is arbitrary and unknown, i.e., the agents perform optimization in a truly sequential manner without the knowledge of the time horizon.

Although the objective of each agent is to minimize the cost function in (3), the aim in the distributed framework is to propose an algorithm that achieves asymptotically the same performance as the optimal batch centralized processor. To this end, we define the following global loss functions:

$$f_t(\boldsymbol{w}_{t,k}) \triangleq \sum_{j=1}^{K} f_{t,j}(\boldsymbol{w}_{t,k}) \tag{5}$$

$$r(\boldsymbol{w}_{t,k}) \triangleq \sum_{j=1}^{K} r_j(\boldsymbol{w}_{t,k}) \tag{6}$$

which represent the loss of the SLFN weight vector of the kth agent on the data observed by all agents on the distributed network at time t. In this way, we aim to construct a distributed learning algorithm that is consistent among all the agents, i.e., each agent should be able to acquire the information from every input and output data pairs observed by every other agent. In particular, our aim is to solve the following optimization problem:

$$\min_{\boldsymbol{w}_{t,k} \in \mathcal{W}} \sum_{t=1}^{T} [f_t(\boldsymbol{w}_{t,k}) + r(\boldsymbol{w}_{t,k})] \tag{7}$$

over all $k \in \{1, \ldots, K\}$. Here, we emphasize that we avoid any statistical assumptions on the input vector and the target data. Instead, our aim is to introduce a sequential algorithm that provides guaranteed performance in a strong deterministic sense. To this end, we next introduce an algorithm, which is used by each agent to minimize the cost function of that

[1] This procedure can be performed before the processing starts by exchanging either the corresponding parameters or the seed of the random number generator.

agent (5), (6). Then, SLFN weight vectors trained by this algorithm are diffused among the agents to solve the global SLFN-based optimization problem in (7).

C. Forward–Backward Splitting Method

For the applications involving big data and/or requiring sequential processing, the dimensionality of the input data can be relatively large [14]. Hence, we aim to design highly efficient algorithms with low complexity to solve the optimization problem in (7). To this end, we use subgradient methods [11] due to their computational efficiency and generalization capabilities [12]. The set of subgradients of a function i at $\boldsymbol{w}$ is defined as follows:

$$\partial i(\boldsymbol{w}) = \{\boldsymbol{g} \in \mathbb{R}^m : i(\boldsymbol{v}) \geq i(\boldsymbol{w}) + \langle \boldsymbol{g}, \boldsymbol{v} - \boldsymbol{w} \rangle \quad \forall \boldsymbol{v} \in \mathbb{R}^m\}. \tag{8}$$

By using the subgradient method to minimize (5) at each individual node k, we obtain iterates of the form $\boldsymbol{w}_{t+1,k} = \boldsymbol{w}_{t,k} - \mu_t \boldsymbol{g}_{t,k} - \mu_t \boldsymbol{h}_{t,k}$, where μ_t is the learning rate of the subgradient method, $\boldsymbol{g}_{t,k} \in \partial f_{t,k}(\boldsymbol{w}_t)$, and $\boldsymbol{h}_{t,k} \in \partial r_k(\boldsymbol{w}_t)$. However, if the cost functions $f_{t,k}(\cdot)$ or $r_k(\cdot)$ are nondifferentiable at certain points, then the iterates $\boldsymbol{w}_{t,k}$ cannot usually attain these values [32], which are usually the desired minima of the cost functions. As an example, for $r_k(\boldsymbol{w}) = \lambda \, ||\boldsymbol{w}||_1$, the desired minima is rarely achievable by the aforementioned update rule. We emphasize that such nondifferentiable cost functions are usually studied in the context of optimization and learning theory via the SLFNs and ELM methods [2].

To address this problem, we use the forward–backward splitting method [25]. In particular, without the loss of generality, we assume that the cost function $f_{t,k}(\cdot)$ is differentiable, and our aim is to ensure that the iterates $\boldsymbol{w}_{t,k}$ attain the nondifferentiable points of the function $r_k(\cdot)$ (note that if $f_{t,k}(\cdot)$ is nondifferentiable at some points, whereas $r_k(\cdot)$ is differentiable, one can easily switch the labels of the cost functions and apply the below procedure). To solve the optimization problem in (5), we perform the following updates:

$$\boldsymbol{w}'_{t+1,k} = \boldsymbol{w}_{t,k} - \mu_t \boldsymbol{g}_{t,k} \tag{9}$$

$$\boldsymbol{w}_{t+1,k} = \arg\min_{\boldsymbol{w}} \left\{ \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}'_{t+1,k}\|^2 + \mu_t r_k(\boldsymbol{w}) \right\} \tag{10}$$

where (9) updates the parameter vector using a subgradient of the first function $f_{t,k}(\cdot)$. Then, in (10), we seek to minimize the regularization cost $r_k(\cdot)$ while not getting too far from the intermediate parameter vector calculated in (9). Note that by using the updates in (9) and (10), we make sure that the iterates attain the aforementioned nondifferentiable points due to the following observation. The zero vector should be an element of the set of subgradients of (10). Therefore, $\boldsymbol{0} \in \boldsymbol{w}_{t+1,k} - \boldsymbol{w}'_{t+1,k} + \mu_t \partial r_k(\boldsymbol{w}_{t+1,k})$. This indicates that there always exists a subgradient vector $\boldsymbol{h}_{t+1,k} \in \partial r_k(\boldsymbol{w}_{t+1,k})$, such that

$$\boldsymbol{w}_{t+1,k} = \boldsymbol{w}_{t,k} - \mu_t \boldsymbol{g}_{t,k} - \mu_t \boldsymbol{h}_{t+1,k}. \tag{11}$$

That is, we can reach any nondifferentiable points.

In the following, we introduce a distributed algorithm that uses the subgradient method described in this section to train the SFLN models in a truly sequential manner over distributed architectures and derive the corresponding guaranteed performance bounds.

Algorithm 1 Distributed Sequential Splitting ELM

for $t = 1$ **to** T **do**
  for $k = 1$ **to** K **do**
    $\hat{y}_{t,k} = \langle \boldsymbol{w}_{t,k}, \boldsymbol{z}_{t,k} \rangle$
    $\boldsymbol{\phi}_{t+1,k} = \sum_{j=1}^{K} \boldsymbol{C}_{jk} \boldsymbol{w}_{t,j}$
    $\boldsymbol{w}'_{t+1,k} = \boldsymbol{\phi}_{t+1,k} - \mu_t \boldsymbol{g}_{t,k}$
    $\boldsymbol{w}_{t+1,k} = \arg\min_{\boldsymbol{w}} \frac{1}{2} \left|\left| \boldsymbol{w} - \boldsymbol{w}'_{t+1,k} \right|\right|^2 + \mu_t r_k(\boldsymbol{w})$
  end for
end for

III. Distributed Sequential Splitting Extreme Learning Machine

In this section, we introduce a sequential distributed SLFN-based algorithm, which uses the forward–backward splitting method for optimization (the complete description of the algorithm is given in Algorithm 1). At each time t, each agent k calculates its estimate of the target data as in (2) and then diffuses the weights of the SFLN, i.e., $\boldsymbol{w}_{t,k}$, to its neighbors, as shown in Fig. 1. Each agent k, after receiving the weight vectors from the neighboring nodes, performs a weighted averaging

$$\boldsymbol{\phi}_{t+1,k} = \sum_{j=1}^{K} \boldsymbol{C}_{jk} \boldsymbol{w}_{t,j} \tag{12}$$

to construct an intermediate weight vector, where $\boldsymbol{C}$ is the communication matrix of the graph, such that $\boldsymbol{C}_{jk}$s are the combination weights. After diffusion of the weights of the SLFN, each agent then performs its update using the forward–backward splitting method based on the combined weight vector, as described in Section II-C.

We analyze the performance of this sequential distributed kernel-adaptive learning algorithm using the following standard assumptions.

1) The convex set $\mathcal{W}$ has a diameter of D, i.e., $||\boldsymbol{w} - \boldsymbol{v}|| \leq D$, $\forall \boldsymbol{w}, \boldsymbol{v} \in \mathcal{W}$.
2) The norms of the subgradients are bounded by A, i.e., $\|\boldsymbol{g}_{t,k}\|, \|\boldsymbol{h}_{t,k}\| \leq A$, $\forall \boldsymbol{g}_{t,k} \in \partial f_{t,k}(\boldsymbol{w}_{t,k})$ and $\forall \boldsymbol{h}_{t,k} \in \partial r_k(\boldsymbol{w}_{t,k})$.
3) The communication graph $\boldsymbol{C}$ forms a doubly stochastic matrix, such that $\boldsymbol{C}$ is irreducible and aperiodic, i.e., $\boldsymbol{C}\boldsymbol{1} = \boldsymbol{C}^T\boldsymbol{1} = \boldsymbol{1}$.
4) Initial weights are equal at each node, i.e., $\boldsymbol{w}_{1,k} = \boldsymbol{w}_{1,j}$, $\forall k, j \in \{1, \ldots, K\}$.

Here, the first assumption enforces a bounded parameter vector, which is clearly realistic, since our cost function has regularization penalties; thus, any parameter vector of interest should be bounded. Similar diameter boundedness assumptions are also used in a number of different papers in the literature [22], [35]. The second assumption holds for a variety of loss functions, since our optimization space is bounded, and consequently, the outputs of the hidden nodes are also bounded. As an example, any cost function whose gradient is

Lipschitz (e.g., squared error loss, logistic loss, and hinge loss) satisfies the second assumption. For a more detailed discussion of this assumption, we refer to [25], [33], [35], and [36]. The third assumption enforces that the contributions of each agent to the overall multiagent framework are equal. This assumption is widely used to analyze the mixing properties of Markov chains [37]. Using similar arguments, the rate of the information diffusion in distributed networks is analyzed under the same assumption in [11]–[13] and [21]. The last assumption is an unbiasedness condition (that also appear in numerous works in the literature [11]–[13], [21]), which is used only for presentation purposes, and our bounds hold [with $\mathcal{O}(1)$ excess factors] even when this assumption is not satisfied.

For Algorithm 1, we have Theorem 1, which states that the introduced algorithm minimizes a sum of convex objective functions with a regret of $\mathcal{O}(K^{1.25}\sqrt{T})$, uniformly at each agent. We note that the convergence result in Theorem 1 uniformly holds for all agents, without any statistical assumptions on the data and without any knowledge of the time horizon, i.e., our algorithm does not need T in hindsight. Hence, the proposed algorithm is truly sequential and consequently highly suitable for applications involving high-dimensional data.

Theorem 1: Under Assumptions 1–4, Algorithm 1, when applied to any input and output data sequences, achieves the following convergence guarantee:

$$\sum_{t=1}^{T}[f_t(\boldsymbol{w}_{t,k})+r(\boldsymbol{w}_{t,k})]-\min_{\boldsymbol{w}\in\mathcal{W}}\sum_{t=1}^{T}[f_t(\boldsymbol{w})+r(\boldsymbol{w})] \leq \mathcal{O}\left(\frac{AD}{\sqrt{\alpha}}K^{1.25}\sqrt{T}\right)$$

for all T and $\forall k \in \{1,\ldots,K\}$, with a suitable learning rate. Here, α is the spectral gap of the communication matrix $\boldsymbol{C}$, D is the diameter of the convex set of the optimization problem, and A is an upper bound on the norms of the subgradients.

This theorem states that the weights of the SLFNs at every agent uniformly yield the same asymptotical normalized accumulated error as the optimal weights that can be chosen only in hindsight, i.e., after observing all the data in a batch mode and centralized manner. Here, we define the normalized accumulated error of the proposed algorithm as follows:

$$\frac{1}{T}\sum_{t=1}^{T}[f_t(\boldsymbol{w}_{t,k})+r(\boldsymbol{w}_{t,k})]$$

and the normalized accumulated error of the competitor algorithm can be defined by replacing $\boldsymbol{w}_{t,k}$ with $\boldsymbol{w}$. Therefore, as $T\to\infty$, the error per instance of our algorithm converges to the error per instance of the optimal batch variant. Thus, although each agent observes a data of length T, all agents can still achieve the normalized performance of the centralized batch algorithm (which processes the entire data of length KT) through the communication between agents for any given network (with a finite size). Furthermore, if the cost function $f_t(\cdot)$ is F-strongly convex and $r(\cdot)$ is R-strongly convex, where $S \triangleq F+R>0$, then Algorithm 1 achieves a convergence guarantee of $\mathcal{O}(K^{1.5}\log T)$, as presented in Corollary 1.

Corollary 1: Under Assumptions 1–4 and when the sum of the cost functions are S-strongly convex for some $S>0$, Algorithm 1, when applied to any input and output data sequences, achieves the following convergence guarantee:

$$\sum_{t=1}^{T}[f_t(\boldsymbol{w}_{t,k})+r(\boldsymbol{w}_{t,k})]-\min_{\boldsymbol{w}\in\mathcal{W}}\sum_{t=1}^{T}[f_t(\boldsymbol{w})+r(\boldsymbol{w})] \leq \mathcal{O}\left(\frac{A^2}{\alpha}K^{1.5}\log T\right)$$

for all T and $\forall k \in \{1,\ldots,K\}$, with a suitable learning rate. Here, α is the spectral gap of the communication matrix $\boldsymbol{C}$ and A is an upper bound on the norms of the subgradients.

IV. Proofs

A. Proof of Theorem 1

In order to relate the performance of the weights of an individual agent to the optimal weights, we define an average weight vector. We first compare the performance of this weight vector with respect to the optimal weights and then relate this average weight to the weights at each agent. To this end, let

$$\bar{\boldsymbol{w}}_t=\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_{t,k} \tag{13}$$

denote the average weight vector. According to this average weight, we also define the following subgradients:

$$\bar{\boldsymbol{g}}_{t,k} \in \partial f_{t,k}(\bar{\boldsymbol{w}}_t) \tag{14}$$

$$\bar{\boldsymbol{h}}_{t,k} \in \partial r_k(\bar{\boldsymbol{w}}_t). \tag{15}$$

We then consider the average estimation parameter $\bar{\boldsymbol{w}}_t$ in (13) and obtain the following recursion:

$$\begin{aligned}\bar{\boldsymbol{w}}_{t+1}&=\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_{t+1,k}\\&=\frac{1}{K}\sum_{k=1}^{K}\left[\sum_{j=1}^{K}\boldsymbol{C}_{jk}\boldsymbol{w}_{t,j}-\mu_t\boldsymbol{g}_{t,k}-\mu_t\boldsymbol{h}_{t+1,k}\right]\\&=\bar{\boldsymbol{w}}_t-\frac{\mu_t}{K}\sum_{k=1}^{K}(\boldsymbol{g}_{t,k}+\boldsymbol{h}_{t+1,k})\end{aligned} \tag{16}$$

where the last line follows from Assumption 3, i.e., $\sum_{k=1}^{K}\boldsymbol{C}_{jk}=1$. By subtracting an arbitrary $\boldsymbol{w}\in\mathcal{W}$ from both sides of (16) and taking the squared norm of both sides, we obtain

$$||\bar{\boldsymbol{w}}_{t+1}-\boldsymbol{w}||^2=||\bar{\boldsymbol{w}}_t-\boldsymbol{w}||^2+\frac{\mu_t^2}{K^2}\left\|\sum_{k=1}^{K}\boldsymbol{g}_{t,k}+\boldsymbol{h}_{t+1,k}\right\|^2-\frac{2\mu_t}{K}\sum_{k=1}^{K}\langle\boldsymbol{g}_{t,k}+\boldsymbol{h}_{t+1,k},\bar{\boldsymbol{w}}_t-\boldsymbol{w}\rangle. \tag{17}$$

The second term of (17) can be bounded as follows:

$$\left\|\sum_{k=1}^{K}\boldsymbol{g}_{t,k}+\boldsymbol{h}_{t+1,k}\right\|^2 \leq \left(\sum_{k=1}^{K}\|\boldsymbol{g}_{t,k}+\|\boldsymbol{h}_{t+1,k}\|\right)^2 \leq 4A^2K^2. \tag{18}$$

We then consider the third term of (17) and write the first inner product $-\langle \boldsymbol{g}_{t,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle$ as follows:

$$-\langle \boldsymbol{g}_{t,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle = -\langle \boldsymbol{g}_{t,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\rangle - \langle \boldsymbol{g}_{t,k}, \boldsymbol{w}_{t,k} - \boldsymbol{w}\rangle. \tag{19}$$

The last term in (19) can be upper bounded using the convexity of $f_{t,k}(\cdot)$ as follows:

$$\begin{aligned}\langle \boldsymbol{g}_{t,k}, \boldsymbol{w} - \boldsymbol{w}_{t,k}\rangle &\le f_{t,k}(\boldsymbol{w}) - f_{t,k}(\boldsymbol{w}_{t,k}) - \frac{F}{2}\|\boldsymbol{w} - \boldsymbol{w}_{t,k}\|^2 \\ &\le f_{t,k}(\boldsymbol{w}) - f_{t,k}(\bar{\boldsymbol{w}}_t) + \langle \bar{\boldsymbol{g}}_{t,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\rangle \\ &\quad - \frac{F}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t,k}\|^2 + \|\boldsymbol{w}_{t,k} - \bar{\boldsymbol{w}}_t\|^2).\end{aligned} \tag{20}$$

Putting (20) back in (19) and noting that the norm of the gradients is upper bounded by A, we obtain

$$\begin{aligned}-\langle \boldsymbol{g}_{t,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle &\le f_{t,k}(\boldsymbol{w}) - f_{t,k}(\bar{\boldsymbol{w}}_t) + 2A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| \\ &\quad - \frac{F}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t,k}\|^2 + \|\boldsymbol{w}_{t,k} - \bar{\boldsymbol{w}}_t\|^2).\end{aligned} \tag{21}$$

We next consider the second inner product in the third term of (17) and denote it as follows:

$$\begin{aligned}-\langle \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle &= -\left\langle \boldsymbol{h}_{t+1,k}, \sum_{j=1}^{K} \boldsymbol{C}_{jk}\boldsymbol{w}_{t,j} - \boldsymbol{w}_{t+1,k}\right\rangle \\ &\quad - \left\langle \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \sum_{j=1}^{K} \boldsymbol{C}_{jk}\boldsymbol{w}_{t,j}\right\rangle \\ &\quad - \langle \boldsymbol{h}_{t+1,k}, \boldsymbol{w}_{t+1,k} - \boldsymbol{w}\rangle.\end{aligned} \tag{22}$$

Here, we observe that the first term of (22) can be upper bounded as

$$\begin{aligned}&\left\langle \boldsymbol{h}_{t+1,k}, \boldsymbol{w}_{t+1,k} - \sum_{j=1}^{K} \boldsymbol{C}_{jk}\boldsymbol{w}_{t,j}\right\rangle \\ &\qquad = \langle \boldsymbol{h}_{t+1,k}, -\mu_t \boldsymbol{g}_{t,k} - \mu_t \boldsymbol{h}_{t+1,k}\rangle \le 2A^2\mu_t\end{aligned} \tag{23}$$

where the second line follows from our update rule. We then consider the second term of (22) and upper bound it as

$$\begin{aligned}&-\left\langle \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \sum_{j=1}^{K} \boldsymbol{C}_{jk}\boldsymbol{w}_{t,j}\right\rangle \\ &\qquad = -\left\langle \boldsymbol{h}_{t+1,k}, \sum_{j=1}^{K} \boldsymbol{C}_{jk}(\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,j})\right\rangle \\ &\qquad \le \sum_{j=1}^{K} \boldsymbol{C}_{jk} A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,j}\|\end{aligned} \tag{24}$$

where the second line follows due to Assumption 3, since $\sum_{j=1}^{K} \boldsymbol{C}_{jk} = 1$. Finally, we consider the third term of (22) and upper bound it using the convexity of $r_k(\cdot)$ [by using a similar trick that we have done in (20)] and obtain

$$\begin{aligned}&\langle \boldsymbol{h}_{t+1,k}, \boldsymbol{w} - \boldsymbol{w}_{t+1,k}\rangle \\ &\quad \le r_k(\boldsymbol{w}) - r_k(\bar{\boldsymbol{w}}_{t+1}) + \langle \bar{\boldsymbol{h}}_{t+1,k}, \bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_{t+1,k}\rangle \\ &\qquad - \frac{R}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t+1,k}\|^2 + \|\boldsymbol{w}_{t+1,k} - \bar{\boldsymbol{w}}_{t+1}\|^2).\end{aligned} \tag{25}$$

Putting (23)–(25) back in (22), we obtain

$$\begin{aligned}&-\langle \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle \\ &\quad \le r_k(\boldsymbol{w}) - r_k(\bar{\boldsymbol{w}}_{t+1}) + 2A^2\mu_t \\ &\qquad + A\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_{t+1,k}\| + \sum_{j=1}^{K} \boldsymbol{C}_{jk} A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,j}\| \\ &\qquad - \frac{R}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t+1,k}\|^2 + \|\boldsymbol{w}_{t+1,k} - \bar{\boldsymbol{w}}_{t+1}\|^2).\end{aligned} \tag{26}$$

Adding (21) and (26), and summing from $k = 1$ to K, we obtain the third term of (17) as follows:

$$\begin{aligned}&-\sum_{k=1}^{K}\langle \boldsymbol{g}_{t,k} + \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle \\ &\quad \le \sum_{k=1}^{K}\Bigg\{ f_{t,k}(\boldsymbol{w}) - f_{t,k}(\bar{\boldsymbol{w}}_t) \\ &\qquad + r_k(\boldsymbol{w}) - r_k(\bar{\boldsymbol{w}}_{t+1}) + \sum_{j=1}^{K} \boldsymbol{C}_{jk} A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,j}\| \\ &\qquad + 2A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| + A\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_{t+1,k}\| \\ &\qquad + 2A^2\mu_t - \frac{F}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t,k}\|^2 + \|\boldsymbol{w}_{t,k} - \bar{\boldsymbol{w}}_t\|^2) \\ &\qquad - \frac{R}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t+1,k}\|^2 + \|\boldsymbol{w}_{t+1,k} - \bar{\boldsymbol{w}}_{t+1}\|^2)\Bigg\}.\end{aligned} \tag{27}$$

From the definition of the subgradient, we have $f_{t,k}(\bar{\boldsymbol{w}}_t) \ge f_{t,k}(\boldsymbol{w}_{t,k}) + \langle \boldsymbol{g}_{t,k}, \boldsymbol{w}_{t,k} - \bar{\boldsymbol{w}}_t\rangle$ and $r_k(\bar{\boldsymbol{w}}_{t+1}) \ge r_k(\boldsymbol{w}_{t+1,k}) + \langle \boldsymbol{h}_{t+1,k}, \boldsymbol{w}_{t+1,k} - \bar{\boldsymbol{w}}_{t+1}\rangle$. Using these inequalities, we can write (27) as follows:

$$\begin{aligned}&-\sum_{k=1}^{K}\langle \boldsymbol{g}_{t,k} + \boldsymbol{h}_{t+1,k}, \bar{\boldsymbol{w}}_t - \boldsymbol{w}\rangle \\ &\quad \le \sum_{k=1}^{K}\Bigg\{ f_{t,k}(\boldsymbol{w}) - f_{t,k}(\boldsymbol{w}_{t,k}) \\ &\qquad + r_k(\boldsymbol{w}) - r_k(\boldsymbol{w}_{t+1,k}) + \sum_{j=1}^{K} \boldsymbol{C}_{jk} A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,j}\| \\ &\qquad + 3A\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| + 2A\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_{t+1,k}\| \\ &\qquad + 2A^2\mu_t - \frac{F}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t,k}\|^2 + \|\boldsymbol{w}_{t,k} - \bar{\boldsymbol{w}}_t\|^2) \\ &\qquad - \frac{R}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{t+1,k}\|^2 + \|\boldsymbol{w}_{t+1,k} - \bar{\boldsymbol{w}}_{t+1}\|^2)\Bigg\}.\end{aligned} \tag{28}$$

To simplify (28), we present Lemma 1, whose proof is given in Section IV-C.

Lemma 1: Under Assumptions 2–4, the deviation of the parameter vector of each agent from the average parameter vector is upper bounded as follows:

$$\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| \le 2A\sqrt{K}\sum_{s=1}^{t-1} \mu_{t-s}\sigma^{s-1} \tag{29}$$

where σ is the second largest singular value of the communication matrix $\boldsymbol{C}$.

Considering the definitions of the cost functions, noting that

$$\frac{1}{K}\sum_{k=1}^{K}(\|\boldsymbol{w}-\boldsymbol{w}_{t,k}\|^2+\|\boldsymbol{w}_{t,k}-\bar{\boldsymbol{w}}_t\|^2)\leq\|\boldsymbol{w}-\bar{\boldsymbol{w}}_t\|^2 \quad (30)$$

from Jensen's inequality, and using Lemma 1, we can rewrite (28) as follows:

$$\begin{aligned}-\sum_{k=1}^{K}&\langle \boldsymbol{g}_{t,k}+\boldsymbol{h}_{t+1,k},\bar{\boldsymbol{w}}_t-\boldsymbol{w}\rangle\\ &\leq f_t(\boldsymbol{w})-f_t(\boldsymbol{w}_{t,k})\\ &\quad+r(\boldsymbol{w})-r(\boldsymbol{w}_{t+1,k})+8A^2K\sqrt{K}\sum_{s=1}^{t-1}\mu_{t-s}\sigma^{s-1}\\ &\quad+2A^2K\mu_t+4A^2K\sqrt{K}\sum_{s=1}^{t}\mu_{t-s+1}\sigma^{s-1}\\ &\quad-\frac{KF}{2}(\|\boldsymbol{w}-\bar{\boldsymbol{w}}_t\|^2)-\frac{KR}{2}(|\|\boldsymbol{w}-\bar{\boldsymbol{w}}_{t+1}\||^2). \quad (31)\end{aligned}$$

In order to obtain the desired upper bound, we put (18) and (31) back in (17) and summing from $t=1$ to T, we obtain

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t+1,k})-r(\boldsymbol{w})\\ &\leq\sum_{t=1}^{T}\left\{\frac{K}{2}\left(\frac{1}{\mu_t}-F\right)\|\boldsymbol{w}-\bar{\boldsymbol{w}}_t\|^2\right.\\ &\quad-\frac{K}{2}\left(\frac{1}{\mu_t}+R\right)\|\boldsymbol{w}-\bar{\boldsymbol{w}}_{t+1}\|^2\\ &\quad+8A^2K\sqrt{K}\sum_{s=1}^{t-1}\mu_{t-s}\sigma^{s-1}\\ &\quad\left.+4A^2K\mu_t+4A^2K\sqrt{K}\sum_{s=1}^{t}\mu_{t-s+1}\sigma^{s-1}\right\}. \quad (32)\end{aligned}$$

Here, we make the following observation:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{s=1}^{t-1}\mu_{t-s}\sigma^{s-1}&=\sum_{s=1}^{T}\sigma^{s-1}\sum_{t=s+1}^{T}\mu_{t-s}\\ &\leq\frac{1}{1-\sigma}\sum_{t=1}^{T}\mu_t \quad (33)\end{aligned}$$

which yields [by adding $r(\bar{\boldsymbol{w}}_1)$ to the both sides of the inequality] the following regret bound:

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t,k})-r(\boldsymbol{w})\\ &\leq\frac{K}{2}\sum_{t=2}^{T}\left(\frac{1}{\mu_t}-\frac{1}{\mu_{t-1}}-F-R\right)\|\boldsymbol{w}-\bar{\boldsymbol{w}}_t\|^2\\ &\quad+AD+\frac{K}{2}\left(\frac{1}{\mu_1}-F\right)\|\boldsymbol{w}-\bar{\boldsymbol{w}}_1\|^2\\ &\quad+4A^2K\left(1+\frac{3\sqrt{K}}{1-\sigma}\right)\sum_{t=1}^{T}\mu_t. \quad (34)\end{aligned}$$

If we have $F=R=0$, i.e., if neither of the cost functions are strongly convex, we obtain the following upper bound:

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t,k})-r(\boldsymbol{w})\\ &\leq\frac{D^2K}{2}\left[\frac{1}{\mu_1}+\sum_{t=2}^{T}\left(\frac{1}{\mu_t}-\frac{1}{\mu_{t-1}}\right)\right]\\ &\quad+AD+4A^2K\left(1+\frac{3\sqrt{K}}{1-\sigma}\right)\sum_{t=1}^{T}\mu_t\\ &\leq AD+\frac{D^2K}{2\mu_T}+4A^2K\left(1+\frac{3\sqrt{K}}{1-\sigma}\right)\sum_{t=1}^{T}\mu_t. \quad (35)\end{aligned}$$

Defining the function $\beta(K,\sigma)\triangleq 1+(3\sqrt{K}/1-\sigma)$ and letting $\mu_t=(D/4A(\beta(K,\sigma)t))^{1/2}$, we can rewrite the upper bound in (35) as follows:

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t,k})-r(\boldsymbol{w})\\ &\qquad\leq AD(1+4K\sqrt{\beta(K,\sigma)}) \quad (36)\end{aligned}$$

since

$$\sum_{t=1}^{T}\frac{D}{4A\sqrt{\beta(K,\sigma)t}}\leq\frac{D\sqrt{T}}{2A\sqrt{\beta(K,\sigma)}}. \quad (37)$$

This concludes the proof of Theorem 1. □

B. Proof of Corollary 1

Here, we consider the case $S=F+R>0$, where we do not require both F and R to be strictly positive, instead either $F>0$ or $R>0$ is sufficient. Then, (34) yields the following upper bound:

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t,k})-r(\boldsymbol{w})\\ &\leq\frac{D^2K}{2}\left[\frac{1}{\mu_1}+\sum_{t=2}^{T}\left(\frac{1}{\mu_t}-\frac{1}{\mu_{t-1}}-S\right)\right]\\ &\quad+AD+4A^2K\left(1+\frac{3\sqrt{K}}{1-\sigma}\right)\sum_{t=1}^{T}\mu_t. \quad (38)\end{aligned}$$

Picking $\mu_t=(1/St)$, we obtain $(1/\mu_t)-(1/\mu_{t-1})-S=0$, which yields

$$\begin{aligned}&\sum_{t=1}^{T}f_t(\boldsymbol{w}_{t,k})-f_t(\boldsymbol{w})+r(\boldsymbol{w}_{t,k})-r(\boldsymbol{w})\\ &\leq AD+\frac{SD^2K}{2}+4A^2K\left(1+\frac{3\sqrt{K}}{1-\sigma}\right)\frac{1+\log T}{S} \quad (39)\end{aligned}$$

since

$$\sum_{t=1}^{T}\frac{1}{St}\leq\frac{1+\log T}{S}. \quad (40)$$

This concludes the proof of Corollary 1. □

C. Proof of Lemma 1

Let $\boldsymbol{W}_t \triangleq [\boldsymbol{w}_{t,1}, \ldots, \boldsymbol{w}_{t,K}]$, $\boldsymbol{G}_t \triangleq [\boldsymbol{g}_{t,1}, \ldots, \boldsymbol{g}_{t,K}]$, and $\boldsymbol{H}_t \triangleq [\boldsymbol{h}_{t,1}, \ldots, \boldsymbol{h}_{t,K}]$. Then, by letting $\mathbf{1}_k$ denotes the vector of all zeros but only a single one at its kth entry, we can write the desired norm as

$$\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| = \left\| \boldsymbol{W}_t \left(\frac{1}{K}\mathbf{1} - \mathbf{1}_k \right) \right\|. \tag{41}$$

According to our update rule, we have the following recursion:

$$\boldsymbol{W}_t = \boldsymbol{W}_1 \boldsymbol{C}^{t-1} - \sum_{s=1}^{t-1} \mu_{t-s} (\boldsymbol{G}_{t-s} + \boldsymbol{H}_{t-s+1}) \boldsymbol{C}^{s-1}. \tag{42}$$

Putting (42) back in (41) and noting that $\boldsymbol{C}\mathbf{1} = \mathbf{1}$ due to Assumption 3 and $\bar{\boldsymbol{w}}_1 = \boldsymbol{w}_{1,k}$, $\forall k \in \{1, \ldots, K\}$ due to Assumption 4, we obtain

$$\begin{aligned} &\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| \\ &\quad = \sum_{s=1}^{t-1} \mu_{t-s} \left\| (\boldsymbol{G}_{t-s} + \boldsymbol{H}_{t-s+1}) \left(\frac{1}{K}\mathbf{1} - \boldsymbol{C}^{s-1}\mathbf{1}_k \right) \right\| \\ &\quad \le \sum_{s=1}^{t-1} \mu_{t-s} \|\boldsymbol{G}_{t-s} + \boldsymbol{H}_{t-s+1}\|_F \left\| \frac{1}{K}\mathbf{1} - \boldsymbol{C}^{s-1}\mathbf{1}_k \right\|. \end{aligned} \tag{43}$$

Here, the first term of (43) can be upper bounded as follows:

$$||\boldsymbol{G}_{t-s} + \boldsymbol{H}_{t-s+1}||_F \le 2A\sqrt{K} \tag{44}$$

since the norms of the subgradients are upper bounded by A due to Assumption 2.

We then consider the second term of (43). In order to derive an upper bound, we first let $\sigma_1(\boldsymbol{A}) \ge \sigma_2(\boldsymbol{A}) \ge \cdots \ge \sigma_K(\boldsymbol{A})$ denote the singular values of a matrix $\boldsymbol{A}$ and $\lambda_1(\boldsymbol{A}) \ge \lambda_2(\boldsymbol{A}) \ge \cdots \ge \lambda_K(\boldsymbol{A})$ denote the eigenvalues of a symmetric matrix $\boldsymbol{A}$. Since the communication graph $\boldsymbol{C}$ is a doubly stochastic matrix (by Assumption 3), we have $\sigma_1(\boldsymbol{C}) = 1$ and $\lambda_1(\boldsymbol{C}^T\boldsymbol{C}) = 1$. Then, we let $\boldsymbol{B} \triangleq (1/K)\mathbf{1}\mathbf{1}^T$ and consider the following norm:

$$\begin{aligned} \left\| \frac{1}{K}\mathbf{1} - \boldsymbol{C}^{s-1}\mathbf{1}_k \right\| &= \|\boldsymbol{B}\mathbf{1}_k - \boldsymbol{C}^{s-1}\mathbf{1}_k\| \\ &= \|(\boldsymbol{B} - \boldsymbol{C})^{s-1}\mathbf{1}_k\| \end{aligned} \tag{45}$$

where

$$\|(\boldsymbol{B} - \boldsymbol{C})^t \mathbf{1}_k\| \le \sigma_1(\boldsymbol{B} - \boldsymbol{C}) \|(\boldsymbol{B} - \boldsymbol{C})^{t-1}\mathbf{1}_k\| \tag{46}$$

$\forall t \ge 1$. Thus, applying (46) $s-1$ times to (45), we obtain

$$\left\| \frac{1}{K}\mathbf{1} - \boldsymbol{C}^{s-1}\mathbf{1}_k \right\| \le \sigma_1^{s-1}(\boldsymbol{B} - \boldsymbol{C}) \|\mathbf{1}_k\|. \tag{47}$$

Here, we observe that $\boldsymbol{C}$ is a doubly stochastic matrix by Assumption 3, hence, $\lambda_1(\boldsymbol{C}) = 1$. Therefore, the eigenspectrums of $\boldsymbol{B} - \boldsymbol{C}$ and $(\boldsymbol{B} - \boldsymbol{C})^T(\boldsymbol{B} - \boldsymbol{C})$ are equal to the eigenspectrums of $\boldsymbol{C}$ and $\boldsymbol{C}^T\boldsymbol{C}$, respectively, except the largest eigenvalues, i.e., $\lambda_1(\boldsymbol{C}) = \lambda_1(\boldsymbol{C}^T\boldsymbol{C}) = 1$, are removed. Thus, from (47), we obtain

$$\left\| \frac{1}{K}\mathbf{1} - \boldsymbol{C}^{s-1}\mathbf{1}_k \right\| \le \sigma_2^{s-1}(\boldsymbol{C}). \tag{48}$$

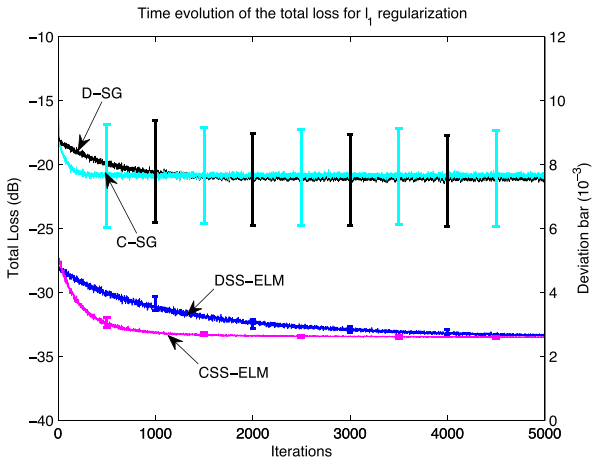

Fig. 2. Comparison of the algorithms for the linear regression model in (50) with mean square error loss and ℓ_1 norm regularization.

Putting (44) and (48) back in (43), we obtain

$$\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}_{t,k}\| \le 2A\sqrt{K} \sum_{s=1}^{t-1} \mu_{t-s} \sigma_2^{s-1}(\boldsymbol{C}) \tag{49}$$

which is the desired result. This concludes the proof of Lemma 1. □

V. Simulations

In this section, we evaluate the performance of distributed sequential splitting ELM (DSS-ELM) algorithm (presented in Algorithm 1) for various multiagent networks with different regularization factors. We illustrate the superior performance of our algorithm for different regression applications involving both real and synthetic data.

A. Stationary Scenario

In this section, we consider a multiagent network of size $K = 10$, where each agent observes a desired signal through a linear model. In particular, the kth agent observes

$$y_{t,k} = \boldsymbol{v}^T \boldsymbol{x}_{t,k} + n_{t,k} \tag{50}$$

where $\boldsymbol{v} \in \mathbb{R}^{10}$ is chosen from a normal distribution, while 50% of the entries are randomly set to zero, $\boldsymbol{x}_{t,k} \in \mathbb{R}^{10}$ is a realization of a spatially and temporally independent normal random vector process, and $n_{t,k} \in \mathbb{R}$ is a zero-mean white Gaussian noise with variance $\sigma_{n_k}^2 = 10^{-3}$. We normalize the input and output attributes to the interval $[0, 1]$ as recommended in [2].

The empirical loss function is chosen as the square error loss, i.e., $f_{t,k} = (y_{t,k} - \hat{y}_{t,k})^2$, whereas the regularization loss function is chosen as ℓ_1-norm regularization in Fig. 2 and ℓ_2^2-norm regularization in Fig. 2, where we set $\lambda = 10^{-4}$. We point out that for these regularization factors, the update rule (i.e., the sixth line in Algorithm 1) is given by

$$\boldsymbol{w}_{t+1,k} = \begin{cases} \operatorname{sign}(\boldsymbol{w}'_{t+1,k}) \odot \left[|\boldsymbol{w}'_{t+1,k}| - \mu_t \lambda \right]_+, & \text{for } \ell_1 \\ \dfrac{1}{1 + \mu_t \lambda} \boldsymbol{w}'_{t+1,k}, & \text{for } \ell_2^2. \end{cases} \tag{51}$$

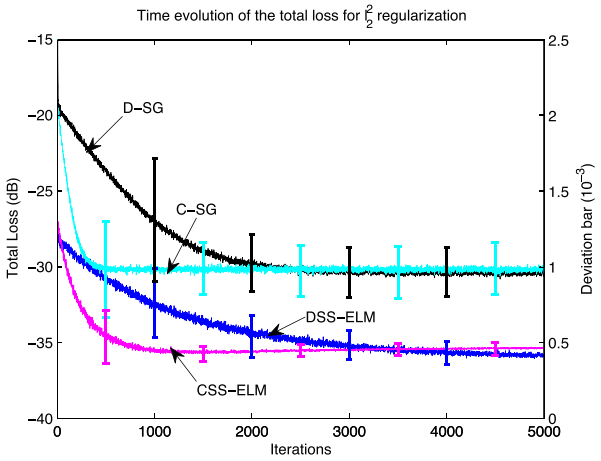

Fig. 3. Comparison of the algorithms for the linear regression model in (50) with mean square error loss and ℓ_2^2 norm regularization.

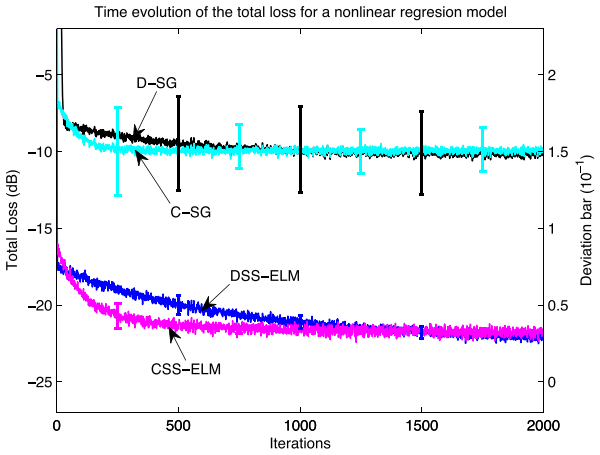

Fig. 4. Comparison of the algorithms for the nonlinear regression model in (52) with mean square error loss and ℓ_2^2 norm regularization.

In Figs. 2 and 3, we compare the performance of the DSS-ELM algorithm with the distributed stochastic gradient (D-SG) algorithm [8], and their centralized variants, i.e., the centralized stochastic gradient (C-SG) and the centralized sequential splitting ELM (CSS-ELM) algorithms. In the distributed multiagent framework, each agent exchanges information with its neighboring agents over a communication network. We have generated this network randomly. Agents employ the uniform combination rule [38], where $C_{j,k} = 1/\pi_k$ with π_k representing the number of neighboring agents of the kth agent. In the centralized framework, all the data observed by every agent are collected at a central processing unit and processed through the conventional processing techniques. In the experiments, the step sizes of the D-SG and C-SG algorithms are set to 0.05, whereas it is 0.1 for the DSS-ELM and CSS-ELM algorithms (these learning rates are chosen for a fair performance comparison, since the number of hidden nodes is approximately twice of the length of the linear parameter vector). We set the number of hidden neurons to 24 in order to achieve a reasonable steady-state loss and use a sigmoidal additive activation function. We have performed 250 independent trials (with random hidden neuron initializations) and presented the averaged results in Figs. 2 and 3. The offset and bias parameters of the hidden neurons are chosen from a uniform distribution $\mathcal{U}[-1, 1]$.

In Figs. 2 and 3, we observe that the distributed processing algorithms achieve a comparable performance with their corresponding centralized versions. We also point out that the DSS-ELM algorithm significantly outperforms the D-SG algorithm for both ℓ_1 and ℓ_2^2 regularization costs. For the ℓ_1-regularization case, the performance of the D-SG significantly deteriorates with respect to the ℓ_2^2-regularization case. On the other hand, the performance of the proposed DSS-ELM algorithm is not significantly affected from these regularization factors, since we use the forward–backward splitting method. Furthermore, since the underlying parameter vector $\boldsymbol{v}$ is sparse, the linear models fail to achieve low error rates, whereas the proposed SLFN-based algorithm yields a considerably smaller error.

We next consider the following nonlinear regression model:

$$y_{t,k} = \text{sign}(\boldsymbol{v}^T \boldsymbol{x}_{t,k}) \tag{52}$$

where $\boldsymbol{v}$ and $\boldsymbol{x}_{t,k}$ are chosen as in the previous setup, and the labels of 10% of the desired signals, i.e., $y_{t,k}$'s, are flipped in order to introduce label noise to the model. In Fig. 4, we present the performances of the DSS-ELM, CSS-ELM, D-SG, and C-SG algorithms for the ℓ_2^2-regularization loss. We observe that the proposed DSS-ELM algorithm significantly outperforms the linear gradient descent-based algorithms. This follows, since the SLFN-based structures can elegantly capture the nonlinear relationships between the desired data and the regressor vectors, whereas linear models cannot learn these relationships satisfactorily. Furthermore, the proposed DSS-ELM algorithm achieves the performance of the centralized CSS-ELM algorithm, which verifies the asymptotical convergence results presented in Theorem 1 and Corollary 1.

B. Nonstationary Scenario

In this example, we evaluate the performance of the DSS-ELM, CSS-ELM, D-SG, and C-SG algorithms in nonstationary environments. We consider the regression setup in (50), where the unknown parameter vector $\boldsymbol{v}$ evolves in time through a random walk model (we emphasize that this setup is extensively used in the machine learning and signal processing literature [39]) as follows:

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \boldsymbol{q}_t \tag{53}$$

where $\boldsymbol{q}_t \in \mathbb{R}^{10}$ is a realization of independent identically distributed zero-mean Gaussian random vector process with autocovariance matrix $10^{-6}\boldsymbol{I}_{10}$.

In Fig. 5, we present the time evolution of the ℓ_2^2-regularized square error loss function of the proposed algorithms. Comparing Fig. 5 with the stationary setup in Fig. 3, we observe that our algorithm suffers approximately 10-dB performance loss, whereas the linear learning models suffer approximately 20-dB performance loss. Therefore, the proposed DSS-ELM algorithm is considerably more robust against the nonstationarity compared with the linear learning algorithms. This follows

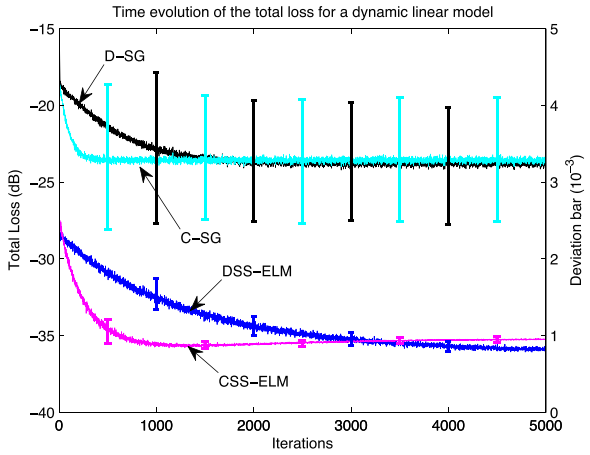

Fig. 5. Comparison of the algorithms for the dynamic linear regression model in (50) and (53) with mean square error loss and ℓ_2^2 norm regularization.

from the efficient SLFN-based nonlinear formulation of our regression model.

In Figs. 2–5, we also provide the sample standard deviations of the algorithms with the corresponding error bars. For presentation purposes, we used different scales for the variance bars and the mean error curves. The values of the mean errors can be read using the scale on the left, whereas the values of the variance bars can be read using the scale on the right. As can be observed from Figs. 2–5, the D-SG and C-SG algorithms have higher variances compared with the DSS-ELM and CSS-ELM algorithms. This is natural, since the mean errors of the D-SG and C-SG algorithms are also higher. As these algorithms cannot attain low mean error rates, consequently, they cannot attain low variance values either. On the other hand, the DSS-ELM and CSS-ELM algorithms achieve a significantly lower variance values. Furthermore, the difference in the variances of the centralized and distributed variants of the algorithms is not significantly different from one another. Specifically, the variances in the steady state are almost identical for centralized and distributed variants of the algorithms. On the other hand, the distributed algorithms yield a higher variance in the early iterations, since the diffusion of the information among agents requires a certain amount of time, which is governed by the size and the structure of the network, as shown in Theorem 1 and Corollary 1.

C. Real Data Sets

In this section, we illustrate the performance of our algorithm for the benchmark data sets, i.e., California housing, protein tertiary structure, and household electrical power consumption [40], [41]. In these data sets, we compare the proposed DSS-ELM algorithm with the Online Sequential (OS)-ELM algorithm of [2]. Throughout this section, we use the sigmoidal additive activation function to generate SLFNs, and we use squared error loss function (with different regularization terms for the DSS-ELM algorithm). We normalize the input and output attributes of the data set to the interval [0, 1] as recommended in [2]. We use 100 data samples (which is more than the recommended value in [2]) at the initialization phase of the OS-ELM algorithm. For the DSS-ELM algorithm, we randomly generate a distributed multiagent network with the specified size, and we set the step size $\mu_t = 10^{-2}$ and the regularization constant $\lambda = 10^{-4}$.

1) *California Housing* data set involves the prediction of the median housing value based on the information collected on several parameters through all the block groups in California from the 1990 census. The number of hidden nodes is set to 90 for the DSS-ELM algorithm and 50 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to four for this experiment.
2) *Protein Tertiary Structure* data set considers the learning the size of the residue based on several other parameters, e.g., the molecular mass weighted exposed area. The number of hidden nodes is set to 50 for the DSS-ELM algorithm and 70 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to 8 for this experiment.
3) *Household Electrical Power Consumption* data set contains 2 075 259 measurements of individual households gathered between December 2006 and November 2010, where the aim is to predict the global active power. In our experiments, we used the initial $\sim$25% of the data corresponding to the data between December 2006 and November 2007. The number of hidden nodes is set to 70 for the DSS-ELM algorithm and 50 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to ten for this experiment.

We emphasize that the number of hidden nodes for the DSS-ELM and OS-ELM algorithms is carefully chosen to minimize the performance of the corresponding algorithms for a fair comparison between these methods. The number of agents is chosen according to the data length to prevent undertraining issues. As an example, for short data sequences, it may not be possible to train the SLFNs for huge number of agents, since each agent observes only a portion of the underlying data.

For each data set, the number of attributes as well as the length of the data that is used for training and testing can be found in Table I. We randomly choose the training and testing data for each trial and evaluate the average performance of the proposed algorithms over 50 independent trials (with random hidden neuron initializations). Note that the OS-ELM algorithm operates on a single centralized agent, whereas the DSS-ELM algorithm works for multiagent systems. Therefore, for the OS-ELM algorithm, the centralized agent is trained using all the training data, whereas for the DSS-ELM algorithm, each agent is trained using only a randomly chosen portion of the training data. As an example, for California housing data set, we have $K = 4$ agents and a training data of length 12 000. In this scenario, the OS-ELM algorithm is trained using the entire training data, whereas for the DSS-ELM algorithm, this entire training data are randomly separated into four different training data of length 3000 and assigned to different agents.

Table II shows the training times and the Root Mean Square Error (RMSE) performances for training as well as

TABLE I
SPECIFICATIONS OF BENCHMARK DATA SETS

Data Set	# attributes	# training data	# test data
California Housing	8	12000	8620
Protein Tertiary	9	36000	9730
Household Power Consumption	7	450000	46067

TABLE II
COMPARISON OF THE RMSE PERFORMANCE OF THE ALGORITHMS

Data Set	Algorithms	Time (s)	RMSE (training)		RMSE (testing)	
			Mean	SD	Mean	SD
California Housing	DSS-ELM (ℓ_1)	0.0749	0.1776	0.0011	0.1779	0.0011
	DSS-ELM (ℓ_2^2)	**0.0705**	**0.1745**	**0.0008**	**0.1746**	**0.0007**
	OS-ELM	1.3219	0.1304	0.0014	0.1312	0.0021
Protein Tertiary	DSS-ELM (ℓ_1)	**0.0995**	**0.0635**	**0.0005**	**0.0638**	**0.0005**
	DSS-ELM (ℓ_2^2)	0.1204	**0.0633**	**0.0005**	**0.0638**	**0.0005**
	OS-ELM	5.1350	0.2308	0.0012	0.2314	0.0016
Household Power Consumption	DSS-ELM (ℓ_1)	1.3709	0.0164	0.0011	0.0165	0.0010
	DSS-ELM (ℓ_2^2)	**1.3127**	**0.0105**	**0.0009**	**0.0106**	**0.0009**
	OS-ELM	123.5087	0.0176	0.0323	0.0177	0.0318

TABLE III
RMSE PERFORMANCE OF THE DSS-ELM ALGORITHM FOR DIFFERENT NETWORK SIZES FOR THE PROTEIN TERTIARY DATA WITH ℓ_2^2-REGULARIZATION

# agents	# samples per agent	Time (s)	RMSE (training)		RMSE (testing)	
			Mean	SD	Mean	SD
4	9000	0.2619	0.0627	4.5e-4	0.0628	4.4e-4
5	7200	0.1959	0.0621	2.7e-4	0.0622	2.4e-4
6	6000	0.1634	0.0633	5.9e-4	0.0634	5.8e-4
8	4500	0.1211	0.0626	4.6e-4	0.0627	3.9e-4
9	4000	0.1072	0.0640	7.0e-4	0.0642	6.7e-4
10	3600	0.0977	0.0647	8.0e-4	0.0648	7.1e-4
12	3000	0.0827	0.0657	8.9e-4	0.0658	7.1e-4

testing for the DSS-ELM and OS-ELM algorithms. Table II shows that the DSS-ELM algorithm can train the SLFNs at each agent significantly faster than the OS-ELM algorithm. As an example, the training time for the DSS-ELM algorithm (that employs ℓ_2^2-regularization) is $\sim$1.33 s at each agent, where we have ten agents in the distributed multiagent network. On the other hand, for the OS-ELM algorithm, the training time is $\sim$124.80 s. Therefore, our algorithm presents an order of magnitude improvement in terms of the training time. Furthermore, even though the computational complexity of the proposed DSS-ELM algorithm is much lower than the OS-ELM algorithm, it yields a significantly better RMSE performance than the OS-ELM algorithm for protein tertiary and household power consumption data sets. This follows, since the DSS-ELM algorithm can adapt nonstationary environments in a faster manner compared with the OS-ELM algorithm. The proposed DSS-ELM algorithm uses a gradient descent-based learning algorithm that can effectively track nonstationarity (as also shown in Section V-B), whereas the OS-ELM algorithm uses a recursive least squares-based approach, which is less resilient to nonstationarity.

In Table II, we also observe that the variance of the DSS-ELM algorithm is usually smaller than the OS-ELM algorithm. The main reason behind this difference in the variances is due to the regularization constraints in the formulation of the DSS-ELM algorithm. The DSS-ELM algorithm presents a more robust performance compared with the OS-ELM algorithm due to the regularization constraints. Furthermore, the ℓ_2^2-regularization variant of the DSS-ELM has a marginally smaller variance compared with the ℓ_1-regularization variant. That is because, the gradient of the ℓ_1-norm is bigger than the gradient of the ℓ_2^2-norm when the parameter vector is close to its optimal value.

Finally, in Table III, we analyze the effects of the network size (i.e., the number of agents) to the performance of the overall system. As can be seen from Table III, as the number of agent increases, the training time of the SLFNs at each agent decreases (as the number of training data per agent decreases). On the other hand, the RMSE and the error variance at each agent increase as the number of agents increases, which is expected from the convergence upper bounds presented in Theorem 1 and Corollary 1.

VI. Conclusion

We study the sequential training of SLFNs over distributed multiagent networks. The aim of each agent is to minimize arbitrary cost functions by training an individual SLFN-based regressor (or classifier) using the data that is revealed only to this particular agent. On the other hand, the goal of the multiagent network is to train these individual SLFNs at each agent as well as the optimal centralized batch SLFN that has access to all the data revealed to all agents. We solve this problem by introducing a distributed sequential ELM-based algorithm. In particular, we show that our algorithm guarantees that the normalized accumulated error of these individual SLFNs at each agent is asymptotically the same as the normalized accumulated error of the optimal centralized batch SLFN that is trained by an ELM-based method. The proposed algorithm works in a truly sequential manner, such that it can be used without any training or initialization phase. Furthermore, the computational complexity of the algorithm is only linear in the number of hidden nodes; hence, it is highly appealing for applications involving big data.

References

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[2] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[3] J. R. McDonnell and D. Waagen, "Evolving recurrent perceptrons for time-series modeling," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 24–38, Jan. 1994.

[4] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, 2007.

[5] S. Balasundaram, D. Gupta, and N. Kapil, "1-norm extreme learning machine for regression and multiclass classification using Newton method," *Neurocomputing*, vol. 128, pp. 4–14, Mar. 2014.

[6] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[7] R. W. Parks *et al.*, "Parallel distributed processing and neural networks: Origins, methodology and cognitive functions," *Int. J. Neurosci.*, vol. 60, nos. 3–4, pp. 195–214, 1991.

[8] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[9] 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 Process. Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.

[10] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, Jul. 2006.

[11] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optim. Theory Appl.*, vol. 147, no. 3, pp. 516–545, Dec. 2010.

[12] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[13] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.

[14] N. D. Vanli and S. S. Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5471–5486, Oct. 2014.

[15] T. Matias, F. Souza, R. Araújo, and C. H. Antunes, "Learning of a single-hidden layer feedforward neural network using an optimized extreme learning machine," *Neurocomputing*, vol. 129, pp. 428–436, Apr. 2014.

[16] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 24–38, Jan. 2005.

[17] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. Neural Netw.*, vol. 9, no. 1, pp. 224–229, Jan. 1998.

[18] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 799–801, May 2000.

[19] H. Chen, J. Peng, Y. Zhou, L. Li, and Z. Pan, "Extreme learning machine for ranking: Generalization analysis and applications," *Neural Netw.*, vol. 53, pp. 119–126, May 2014.

[20] H.-G. Han, L.-D. Wang, and J.-F. Qiao, "Hierarchical extreme learning machine for feedforward neural network," *Neurocomputing*, vol. 128, pp. 128–135, Mar. 2014.

[21] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.

[22] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[23] P. L. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM J. Numer. Anal.*, vol. 16, no. 6, pp. 964–979, 1979.

[24] G. H.-G. Chen and R. T. Rockafellar, "Convergence rates in forward-backward splittingg," *SIAM J. Numer. Anal.*, vol. 7, no. 2, pp. 421–444, 1997.

[25] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Mach. Learn. Res.*, vol. 10, pp. 2899–2934, Dec. 2009.

[26] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part I: Modeling and stability analysis," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 811–826, Feb. 2015.

[27] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part II: Performance analysis," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 827–842, Feb. 2015.

[28] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part III: Comparison analysis," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 843–858, Feb. 2015.

[29] J. Keuper and F.-J. Pfreundt, "Asynchronous parallel stochastic gradient descent: A numeric core for scalable distributed machine learning algorithms," in *Proc. Workshop Mach. Learn. High-Perform. Comput. Environ. (MLHPC)*, 2015, Art. no. 1.

[30] S. Sra, A. W. Yu, M. Li, and A. J. Smola. (2015). "AdaDelay: Delay adaptive distributed stochastic convex optimization." [Online]. Available: http://arxiv.org/pdf/1508.05003v1.pdf.

[31] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27. 2014, pp. 2915–2923.

[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[33] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, nos. 2–3, pp. 169–192, 2007.

[34] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: Upper and lower bounds," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2354–2362, Aug. 2002.

[35] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2483–2493, Nov. 2013.

[36] E. Hazan and S. Kale, "Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization," *J. Mach. Learn. Res.*, vol. 15, pp. 2489–2512, Jul. 2014.
[37] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2003.
[38] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. 44th IEEE Conf. Decision Control, Eur. Control Conf. (CDC-ECC)*, Dec. 2005, pp. 2996–3000.
[39] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
[40] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statist. Probab. Lett.*, vol. 33, pp. 291–297, May 1997.
[41] M. Lichman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml/citation_policy.html.

N. Denizcan Vanli received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in electrical engineering and computer science with the Massachusetts Institute of Technology, Cambridge, MA, USA.

His current research interests include convex optimization, online learning, and distributed optimization.

Muhammed O. Sayin received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Illinois at Urbana–Champaign, Champaign, IL, USA.

His current research interests include distributed networks, optimal control, and dynamic games.

Ibrahim Delibalta received the B.S. degree in electrical engineering from Boğaziçi University, Istanbul, Turkey, in 1993, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1995. He is currently pursuing the Ph.D. degree in design, technology, and society, an interdisciplinary program, with Koç University, Istanbul, with a focus on social sciences and machine learning.

He was with Intel, Santa Clara, CA, USA, Silicon Graphics, Milpitas, CA, USA, and Cisco Systems, San Jose, CA, USA, where he was involved in the field of high performance microprocessor and network processor chip design from 1995 to 2010. He has held various positions with the Türk Telekom Group, Ankara, Turkey, since 2011, where he is currently leading the Emerging Services and Research and Development Team.

Suleyman Serdar Kozat (A'10–M'11–SM'11) received the B.S. (Hons.) degree from Bilkent University, Ankara, Turkey, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA.

He joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, as a Research Staff Member and later became a Project Leader with the Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He was a Research Associate with the Cryptography and Anti-Piracy Group, Microsoft Research, Redmond, WA, USA. He is currently an Associate Professor with the Electrical and Electronics Engineering Department, Bilkent University. He has co-authored over 100 papers in refereed high impact journals and conference proceedings and holds several patent inventions (currently used in several different Microsoft and IBM products, such as MSN and ViaVoice). He holds several patent inventions due to his research accomplishments with the IBM Thomas J. Watson Research Center and Microsoft Research. His current research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering, and machine learning algorithms for signal processing.

Dr. Kozat received many international and national awards. He is the Elected President of the IEEE Signal Processing Society, Turkey Chapter.