Consensus-based Cooperative Algorithms for Training over Distributed Datasets Using Stochastic Gradients

DOI:
10.1109/TNNLS.2021.3071058

Document Version
Accepted author manuscript

Consensus-based Cooperative Algorithms for Training over Distributed Datasets Using Stochastic Gradients

Zhongguo Li. *Student Member, IEEE*, Bo Liu and Zhengtao Ding, *Senior Member, IEEE*

***Abstract*—In this paper, distributed algorithms are proposed for training a group of neural networks with private datasets. Stochastic gradients are utilised in order to eliminate the requirement for true gradients. To obtain a universal model of the distributed neural networks trained using local datasets only, consensus tools are introduced to derive the model towards the optimum. Most of the existing works employ diminishing learning rates, which are often slow and impracticable for online learning, while constant learning rates are studied in some recent works, but the principle for choosing the rates is not well established. In this paper, constant learning rates are adopted to empower the proposed algorithms with tracking ability. Under mild conditions, the convergence of the proposed algorithms is established by exploring the error dynamics of the connected agents, which provides an upper bound for selecting the constant learning rates. Performances of the proposed algorithms are analysed with and without gradient noises, in the sense of mean-square-error (MSE). It is proved that the MSE converges with bounded errors determined by the gradient noises, and the MSE converges to zero if the gradient noises are absent. Simulation results are provided to validate the effectiveness of the proposed algorithms.**

***Index Terms*—Consensus, optimisation, distributed training, neural networks, convergence analysis, multi-agent systems.**

I. Introduction

OPTIMISATION and learning over distributed networks have been widely studied in recent years, owing to their significant potentials in many biological, engineering, and social applications [1–6]. Several critical limitations of the centralised methods can be addressed by the distributed algorithms: first, communicational requirement is relieved as information exchanges are confined to adjacent neighbours; second, local datasets can be kept private and do not need to be revealed to remote fusion centres; third, computational burdens are distributed into a set of agents, where each of them only needs to process its local datasets.

Recent advances in distributed optimisation have been reported in a number of works, e.g., [2, 3, 7–11], where some of them are discrete-time algorithms [2, 7, 9] and some utilise continuous-time methods [3, 8, 10, 12]. Distributed subgradient-based methods have been proposed in [2], and projection-based dual algorithms are studied in [7]. Other branches also include the alternating direction method of multipliers (ADMM) [9] and the fixed-time consensus-based algorithms [11], where the combination matrices of the communication graphs are required to be doubly stochastic [2, 7].

More recently, considerable interests have been concentrated on training neural networks using distributed datasets [4, 13–15]. One of the most important reasons is that we are now faced with the era of big data, and massive amounts of information are generated everyday and everywhere. Learning from big data using a single model is challenging and impractical [16]. Moreover, critical measures have been widely adopted by many countries and companies for purposes of data protection. It is unlikely that a central server can be trusted to collect private information from all participants in the networks.

To deal with those challenges, significant efforts have been dedicated to distributed learning [4, 13, 17, 18]. Various approaches have been proposed in the existing literature, for example, the incremental strategies [19–21], and the consensus-based strategies [4, 6, 13]. In the incremental strategies, a cyclic path that traverses all the agents in the network is required at every iteration, which is computationally intensive and also sensitive to link failures. On the other hand, the consensus-based methods can update the local models simultaneously using local interactions only. In addition, consensus-based methods can provide a fast convergence speed with superior robustness to single end-point failures. In [13], a consensus-based algorithm for distributed machine learning is proposed, where restrictions on the consensus steps are imposed, and various extensions have been studied in recent works [4, 15]. Distributed support vector machine (SVM) learning for wireless sensor networks is considered in [17]. Stochastic variance reduction methods are developed in [6] by using batch gradient descent algorithms. In general, strong assumptions, e.g., convergent learning parameters and basin of attraction, have been utilised to establish the convergence of those algorithms in the existing studies. It is required that the weights of the neural networks move towards the optimum after each iteration (see, for example, Assumption 1 in [4] and Case 1 in [13]), which however is quite different from the actual learning behaviours of the distributed agents. Due to the use of stochastic gradients for training problems, the parameter errors may fluctuate around the true optimum, instead of monotonically decrease. In the worst scenario, the parameters may even diverge without properly designed learning rates. Two critical issues have been overlooked in the existing studies: how the learning rates should be designed for the cooperative algorithms, and how the designs will affect the

The authors are with Department of Electrical and Electronic Engineering, University of Manchester, Sackville Street Building, Manchester, M13 9PL, UK (e-mails: zhongguo.li@manchester.ac.uk; bo.liu-2@manchester.ac.uk; zhengtao.ding@manchester.ac.uk).

* Corresponding author: Zhengtao Ding.

performance of the algorithms, which motivates our study in this paper.

In the aforementioned literature, we observe that exact gradients are often applied for distributed optimisation, while stochastic gradient algorithms are commonly used for machine learning problems. It should be noted that the cost functions and the true gradients are rarely available, and approximations are needed for training [15]. In this paper, we consider distributed training problems for a group of neural networks, where each of them is equipped with local dataset that cannot be shared with any others. We will first examine the distributed algorithms for deterministic optimisation problems. Consensus algorithms are adopted to drive the parameters of local neural networks towards a weighted average. The optimal solution can then be achieved by combining with the local training algorithms. To gain a deep understanding of the learning behaviour, we regard the stochastic gradients as the exact gradients disturbed by noises. In this way, distributed optimisation can be analysed as a special training without noises. Two different training methods, learning-then-consensus (LTC) and consensus-then-learning (CTL) algorithms, are proposed with stochastic gradient noises. A compact expression for both algorithms is established, based on which the convergence and performance of both algorithms can be analysed in a unified manner.

We notice that most of the studies have been contributed to the algorithm development using diminishing learning rates [2, 22, 23]. However, the algorithms with decaying sequences are very slower [2], and when the step sizes decay to zero, the algorithms are no longer able to track the changing solution. This prevents those algorithms from being deployed for online machine learning problems with streaming data samples. In our algorithms, constant step sizes are employed, which is useful for continuous learning at a relatively faster convergence speed. Although the algorithms with constant learning rates demonstrate great performance in tracking capability and learning speed, they also suffer from disturbances caused by the gradient noises, and optimality errors cannot be completely removed in the presence of gradient noises. Therefore, the performance of algorithms with constant learning rates should be carefully analysed using approximated gradients. However, effective criteria for designing the learning rates have not been studied for the distributed algorithms, as having been discussed above. In this paper, we further establish a link between the learning rates and the convergence of the consensus-based algorithms. To facilitate the performance analysis, the error dynamics of the networks is derived, by which a closed-loop error recursion can be developed. It is a very challenging task to analyse the performance of networked agents, as the error of the parameters and gradient noises of one agent will be diffused across the network via the communication graphs. It is guaranteed that the algorithms converge to a bounded region by choosing proper step sizes determined by the gradient and the noise properties. It is also proved that in the absence of gradient noises, the worst MSE will converge to zero.

The significance of our study is to provide training algorithms for distributed neural networks, where datasets are large and the data access is restricted to local agents only. The key contributions have been summarised as follows.

1) We provide a guideline for choosing the constant learning rates, by which the convergence of the algorithms can be guaranteed.
2) The properties of the gradient noises and their impact on the algorithm performance are explored, which provides a deeper insight into the learning behaviours of distributed agents.
3) We have developed a unified framework for distributed training problems, which includes the existing algorithms in [4, 15] and [13] as special cases.

The remainder of this paper is organised as follows. In Section II, the problem of training neural networks is formulated, and distributed algorithms are proposed to solve the problem. Section III presents the convergence and performance analysis of the proposed algorithms. Simulations and discussions are elaborated in Section IV. Finally, Section V concludes this paper.

Notation: Let $\mathbb{R}$, $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ denote the real numbers, real vectors of n dimensions and real matrices of dimension $n\times m$. For a series of vectors, $a_1,\ldots,a_N$, let $\mathrm{col}(a_1,\ldots,a_N)$ be a column vector by stacking $(a_1,\ldots,a_N)$ on top of each other. The identity matrix of dimension $n\times n$ is represented by I_n. $\mathbf{0}_n$ and $\mathbf{1}_n$ represent a column vector of dimension n with all entries being zero and one, respectively. A^T denotes the transpose of a matrix $A\in\mathbb{R}^{n\times m}$. The Kronecker product is denoted by $\otimes$. For a vector $x\in\mathbb{R}^n$, $\|x\|=\sqrt{x^Tx}$ denotes the Euclidean norm; $\|x\|_\infty=\max_i|x_i|$ denotes the infinity norm; and $\|x\|_A^2=x^TAx$. For $x\in\mathbb{R}^n$ and $f(x):\mathbb{R}^n\to\mathbb{R}$, we denote $\nabla_x f(p)=[\frac{\partial f}{\partial x_1}(p),\ldots,\frac{\partial f}{\partial x_n}(p)]^T$ as the gradient of function $f(x)$ at p, and $\nabla_x^2 f(p)=[\frac{\partial^2 f(p)}{\partial x_i\partial x_j}]_{n\times n}$ as the Hessian matrix of $f(x)$ at p. For a square matrix $A=[a_{ij}]_{n\times n}$, the infinity norm $\|A\|_\infty=\max_{1\le i\le n}\sum_{j=1}^n|a_{ij}|$ is the maximum absolute row sum. In this paper, the superscript index k denotes the kth iteration, and the subscript i represents the ith agent in the network, unless explicitly stated otherwise. Boldface letters denote random variables, and normal letters represent deterministic variables.

II. Problem Formulation and Algorithm Development

In this section, we present distributed training methods for neural networks by consensus-based approaches. We will first examine the structure of two commonly used consensus-based learning algorithms to deal with deterministic objective functions (see (2)-(5) further ahead). Considering that stochastic objectives and gradients are often adopted in training neural networks, we thus introduce the approximated gradients to replace the unavailable true gradients. Then, we reformulate the two algorithms in a unified expression (as in (13)-(15) further ahead) to ease the performance analysis in the subsequent section. Two assumptions on the empirical risks and the gradient noises are introduced to facilitate the theoretical analysis.

Before presenting the problem formulation, we first introduce some definitions related to graph theory. The connection among a group of N agents is described by a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$,

which consists of N vertices, $\mathcal{V} = \{1, \dots N\}$, denoting the agents in the network, and a set of edges, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, representing the communication links among the agents. In this paper, the graphs are assumed to be connected. If an edge pair $(j, i) \in \mathcal{E}$, then agent j is said to be a neighbour of agent i. We denote all the neighbours of the ith agent as $\mathcal{N}_i \triangleq \{j | (j, i) \in \mathcal{E}, \forall j \in \mathcal{V}\} \cup \{i\}$. The combination matrix is defined as $\mathcal{C} = [c_{ij}]_{N \times N}$, where $c_{ij} > 0$ if edge pair $(j, i) \in \mathcal{E}$, and $c_{ij} = 0$ otherwise. It is assumed that self-loop exists in the graph, that is, $c_{ii} > 0$. Moreover, each row of $\mathcal{C}$ sums to one, i.e., $\mathcal{C}\,\mathbf{1}_N = \mathbf{1}_N$. Note that, for connected graphs, $\mathcal{C}$ has a single eigenvalue at one, and all other eigenvalues are less than one in magnitude [24]. For more graph definitions and properties, we refer the reader to [25] and [26]. The combination matrix $\mathcal{C}$ is assumed to be row stochastic only, which is different from the doubly stochastic matrices used in [2, 7, 9, 20]. As a result, the communication topology can be unbalanced directed graphs.

Our framework consists of a group of N neural networks distributed over a connected graph, where each agent possesses a local dataset that cannot be revealed to global fusion centres. The objective of the network is to minimise the empirical risk over the entire dataset, given by

$$\min_{\theta \in \mathbb{R}^n} \mathcal{R}(\mathcal{D}, \theta) = \sum_{i=1}^{N} \mathcal{R}_i(\mathcal{D}_i, \theta), \tag{1}$$

where $\mathcal{D}$ and $\mathcal{D}_i$ denote the entire dataset and the sub-dataset of agent i, respectively; $\theta \in \mathbb{R}^n$ is the weight of the neural networks. In this paper, we assume the local neural networks have the same optimal parameters, denoted by θ^*. This is one of the most common cases in the machine learning problems, for example, identifying an underlying statistical distribution [27], tracking the same target [28, 29], and learning a common model with similar data distributions [4, 13].

Existing works on parallel learning algorithms have been mostly focused on the master-slave graphs [30], which may cause communication jam over the master node, and consequently lead to the collapse of all networked machines. Therefore, we will adopt the full distributed learning networks without any central node to avoid possible failures and communication latency. Detailed advantages and comparisons of the two types of networks have been well documented in recent studies [4, 6].

For each training machine $i \in \mathcal{V}$, the gradient-based algorithm is designed as

$$\psi_i^k = \theta_i^{k-1} - \eta_i \nabla_\theta \mathcal{R}_i(\mathcal{D}_i, \theta_i^{k-1}), \tag{2}$$

$$\theta_i^k = \sum_{j \in \mathcal{N}_i} c_{ij} \psi_j^k, \tag{3}$$

where ψ_i^k is an intermediate variable, and $\eta_i > 0$ is the learning rate. The updates of coefficients include two procedures. In the first stage, the agents train their neural networks independently via local learning process as in (2), where only local information is used, including the latest coefficients θ_i^{k-1} and the gradient of the empirical risk $\nabla_\theta \mathcal{R}_i(\mathcal{D}_i, \theta_i^{k-1})$. In the second stage (3), the consensus of the locally trained parameters is performed, where neighbouring information ψ_j^k for all $j \in \mathcal{N}_i$ is exchanged. Therefore, we call this design as learning-then-consensus algorithm. With this setting, all the agents are able to obtain the same neural network as if all sub-datasets are available for every local agent, which will be demonstrated by the convergence analysis. Similarly, we can swap the steps in (2) and (3), yielding the consensus-then-learning algorithm

$$\psi_i^k = \sum_{j \in \mathcal{N}_i} c_{ij} \theta_j^{k-1}, \tag{4}$$

$$\theta_i^k = \psi_i^k - \eta_i \nabla_\theta \mathcal{R}_i(\mathcal{D}_i, \psi_i^k). \tag{5}$$

In many existing studies, decaying learning rates have been utilised, e.g., [2, 9, 22], where the following two conditions are usually required

$$\sum_{k=0}^{\infty} \eta_i(k) = \infty, \quad \sum_{k=0}^{\infty} \eta_i^2(k) < \infty. \tag{6}$$

Though, in general, algorithms with decaying learning rates can reach the optimal solution almost surely, the convergence speed is very slow (see, e.g., [2, 22, 31] for detailed discussions). More importantly, when the learning rates decay to zero, the machine will stop learning even if the optimal solution has evolved. For algorithms with fixed learning rates, they will converge to the optimal solution at an exponential rate under mild conditions (see [2, 32, 33]). In addition, they are capable of tracking the change of the solution, due to their continuously learning ability, which is useful in many practical scenarios with drifting optimum θ^* or with streaming datasets $\mathcal{D}_i$ in real time. Therefore, in this paper, we will concentrate on analysing the algorithms with constant learning rates as employed in the proposed LTC and CTL strategies. In the sequel, we will combine constant learning rates with approximated gradients, and analyse the resulting MSE performance.

Now, we examine the properties of the empirical risks, $\mathcal{R}_i(\mathcal{D}_i, \theta)$. For the i th agent with m_i samples in $\mathcal{D}_i$, the gradient can be calculated by

$$\nabla_\theta \mathcal{R}_i(\mathcal{D}_i, \theta) = \frac{1}{m_i} \sum_{k=1}^{m_i} \nabla_\theta L(s_i^k, \theta), \tag{7}$$

where s_i^k represents the kth data sample in $\mathcal{D}_i$, and $L(s_i^k, \theta)$ is the empirical risk of sample s_i^k. In many cases, the empirical risks and their gradients are not available or cannot be expressed by explicit functions (usually defined as the expectations of loss functions). Furthermore, it should be noted that the expressions for the gradients are dependent on the entire datasets, which are not available when we train the neural networks using the instant sample s_i^k. Nevertheless, we can replace the true gradients by approximations with random noises, also termed as stochastic gradients. In stochastic learning problems, the gradients are usually computed by the realisations of real data samples used at the kth iteration. We denote the estimated gradient $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}_i}(\boldsymbol{\theta})$ as

$$\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}_i}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{R}_i(\boldsymbol{\theta}) + \boldsymbol{d}_i(\boldsymbol{\theta}), \tag{8}$$

where $\nabla_{\boldsymbol{\theta}} \mathcal{R}_i(\boldsymbol{\theta})$ is the exact gradient, and $\boldsymbol{d}_i(\boldsymbol{\theta})$ is the gradient noise. It can be understood as the real gradients disturbed

by noises, due to the realisations of instant data samples. For notational conveniences, $\mathcal{D}_i$ has been dropped from our expressions. Note that we are now using boldface letters to reflect their stochastic nature.

To this end, we can reformulate the proposed LTC and CTL algorithms as

$$\boldsymbol{\psi}_i^k = \boldsymbol{\theta}_i^{k-1} - \eta_i \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}}_i(\boldsymbol{\theta}_i^{k-1}), \tag{9}$$

$$\boldsymbol{\theta}_i^k = \sum_{j \in \mathcal{N}_i} c_{ij} \boldsymbol{\psi}_j^k, \tag{10}$$

and

$$\boldsymbol{\psi}_i^k = \sum_{j \in \mathcal{N}_i} c_{ij} \boldsymbol{\theta}_j^{k-1}, \tag{11}$$

$$\boldsymbol{\theta}_i^k = \boldsymbol{\psi}_i^k - \eta_i \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}}_i(\boldsymbol{\psi}_i^k). \tag{12}$$

To facilitate the performance analysis of both the algorithms, we capture the structure of them in a unified expression as

$$\boldsymbol{\psi}_i^{k-1} = \sum_{j \in \mathcal{N}_i} a_{1,ij} \boldsymbol{\theta}_j^{k-1}, \tag{13}$$

$$\boldsymbol{\phi}_i^k = \boldsymbol{\psi}_i^{k-1} - \eta_i \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}}_i(\boldsymbol{\psi}_i^{k-1}), \tag{14}$$

$$\boldsymbol{\theta}_i^k = \sum_{j \in \mathcal{N}_i} a_{2,ij} \boldsymbol{\phi}_j^k, \tag{15}$$

where $[A_1]_{ij} = a_{1,ij}$ and $[A_2]_{ij} = a_{2,ij}$.

By choosing different matrices of A_1 and A_2, we can switch between different training strategies. Selecting $A_1 = I_N$ and $A_2 = \mathcal{C}$ leads to the LTC algorithm, and choosing $A_1 = \mathcal{C}$ and $A_2 = I_N$ yields the CTL algorithm. In addition, if both both A_1 and A_2 are set to be $\mathcal{C}$, then we obtain a new learning algorithm with two consensus steps at each iteration. Using the unified expression, we can analyse the learning behaviours of different algorithms together.

Before proceeding to the convergence analysis of the unified algorithms in (13)-(15), two underlying assumptions for the performance analysis are needed, which are similar to those used in existing studies, as we will discuss.

Assumption 1: The Hessian matrix of each empirical risk, $\nabla^2_{\boldsymbol{\theta}} \mathcal{R}_i(\boldsymbol{\theta})$, is bounded, i.e.,

$$\underline{\alpha}_i I_n \leq \nabla^2_{\boldsymbol{\theta}} \mathcal{R}_i(\boldsymbol{\theta}) \leq \bar{\alpha}_i I_n, \tag{16}$$

where $0 \leq \underline{\alpha}_i \leq \bar{\alpha}_i$ are non-negative constants, with at least one $\underline{\alpha}_i > 0$. ■

Assumption 2: The gradient noise, $\boldsymbol{d}_i(\boldsymbol{\theta})$, satisfies the following properties:

$$\mathbb{E}\left[\boldsymbol{d}_i(\boldsymbol{\theta}) | \boldsymbol{\mathcal{F}}^{k-1}\right] = 0, \tag{17}$$

$$\mathbb{E}\left[\|\boldsymbol{d}_i(\boldsymbol{\theta})\|^2 | \boldsymbol{\mathcal{F}}^{k-1}\right] \leq \zeta^2 \|\boldsymbol{\theta}\|^2 + \xi^2, \tag{18}$$

where ζ and ξ are non-negative constant, and $\boldsymbol{\mathcal{F}}^{k-1}$ represents the filtration (past history in σ-field) of the random process $\boldsymbol{\theta}_i^l$, for all $l \leq k-1$ and $i \in \mathcal{V}$. ■

By Assumption 1, the network objective, $\sum_{i=1}^N \mathcal{R}_i(\boldsymbol{\theta})$, is guaranteed to be strongly convex, and consequently there exists a unique optimal parameter θ^*. In many problems of practical interest, the objective functions are usually convex or can be reformulated as convex functions by means of regularisation, e.g., least-mean-square (LMS) learning [34] and distributed estimation [35] (see [36] and references therein for more examples). Moreover, strong convexity can be used to establish exponential convergence, which is helpful for optimisation and learning over networks [2, 3, 7, 11]. In distributed optimisation problems, some strong assumptions have been considered. For example, bounded Hessian matrices have been commonly used to facilitate the convergence analysis [1, 2, 7, 9]. In some works [2, 10, 37], bounded gradients are utilised with set constraints, which is unfeasible for unconstrained problems. In this sense, our assumption is less stringent because problems with unbounded gradients can be solved.

Expression (17) in Assumption 2 indicates that the gradient noise, $\boldsymbol{d}_i(\boldsymbol{\theta})$, is unbiased, which is a reasonable assumption since, for most of learning applications, the gradient expectation of a particular sample is equal to the exact gradient. The second expression in (18) assumes that the conditional variance of the gradient noise is bounded by the sum of the parameter norm $\|\boldsymbol{\theta}\|^2$ and a constant ξ^2. Uniformly bounded variance (stronger assumption) has been considered in many studies, for example [15, 38], which is simply bounded by a constant ξ^2. Equation (18) allows the conditional variance to have a wide range when the parameter $\boldsymbol{\theta}$ is far from the optimum θ^*. To see this, we can rewrite (18) as

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{d}_i(\boldsymbol{\theta})\|^2 | \boldsymbol{\mathcal{F}}^{k-1}\right] &\leq \zeta^2 \|\boldsymbol{\theta} - \theta^* + \theta^*\|^2 + \xi^2 \\ &\leq 2\zeta^2 \|\boldsymbol{\theta} - \theta^*\|^2 + 2\|\theta^*\|^2 + \xi^2.\end{aligned} \tag{19}$$

Defining a new set of variables, $\bar{\zeta}^2 = 2\zeta^2$ and $\bar{\xi}^2 = 2\|\theta^*\|^2 + \xi^2$, we have

$$\mathbb{E}\left[\|\boldsymbol{d}_i(\boldsymbol{\theta})\|^2 | \boldsymbol{\mathcal{F}}^{k-1}\right] \leq \bar{\zeta}^2 \|\tilde{\boldsymbol{\theta}}\|^2 + \bar{\xi}^2, \tag{20}$$

where $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \theta^*$ denotes the error of the parameters.

Remark 1: The distributed learning problem studied in this work is different from the traditional consensus problem that aims to achieve an agreement on a weighted average of the initial states [39]. Those consensus problems do not learn from injected data, and they can be understood as a special case by setting the learning step η_i in the proposed algorithms to zero. On the other hand, the training problem considered in this work enables the distributed agents to learn from the streaming data, and cooperatively search for the global optimal solution. In this case, the initial conditions of $\boldsymbol{\theta}_i$ will not change the optimal solution or affect the convergence of the proposed algorithms.

III. Performance Analysis

In this section, we will analyse the MSE performance of the proposed LTC and CTL algorithms in a unified manner. Since the training machines are coupled together by the network, the errors may diffuse across the agents. We will first derive the compact error dynamics by collecting all agents' error vectors in extended formats, based on which we will establish the error recursion dynamics by exploring the relations among different error vectors, as in (27)-(29) further ahead. Using the mean value theorem and conditions in Assumptions 1 and 2, we are

able to obtain the closed-loop MSE dynamics later in (49). Then, in Theorem 1, we provide a guideline for designing the learning rates, based on which we prove the convergence of the MSE to a bounded region with gradient noise. In addition, we further give the results in the absence of gradient noises as in Corollary 1.

First, we rewrite the learning process of the network in a compact form by collecting all the parameters of the local agents into the extended vectors as

$$\boldsymbol{\Psi}^k = \mathrm{col}(\boldsymbol{\psi}_1^k, \ldots, \boldsymbol{\psi}_N^k) \in \mathbb{R}^{nN}, \tag{21}$$

$$\boldsymbol{\Phi}^k = \mathrm{col}(\boldsymbol{\phi}_1^k, \ldots, \boldsymbol{\phi}_N^k) \in \mathbb{R}^{nN}, \tag{22}$$

$$\boldsymbol{\Theta}^k = \mathrm{col}(\boldsymbol{\theta}_1^k, \ldots, \boldsymbol{\theta}_N^k) \in \mathbb{R}^{nN}. \tag{23}$$

It follows from (13)-(15) that

$$\boldsymbol{\Psi}^{k-1} = (A_1 \otimes I_n)\boldsymbol{\Theta}^{k-1}, \tag{24}$$

$$\boldsymbol{\Phi}^k = \boldsymbol{\Psi}^{k-1} - (\Delta \otimes I_n)\left[\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1}) + \boldsymbol{D}(\boldsymbol{\Psi}^{k-1})\right], \tag{25}$$

$$\boldsymbol{\Theta}^k = (A_2 \otimes I_n)\boldsymbol{\Phi}^k, \tag{26}$$

where $\Delta = \mathrm{diag}(\eta_1, \ldots, \eta_N)$, $\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1}) = \mathrm{col}(\nabla_{\boldsymbol{\theta}}\mathcal{R}_1(\boldsymbol{\psi}_1^{k-1}), \ldots, \nabla_{\boldsymbol{\theta}}\mathcal{R}_N(\boldsymbol{\psi}_N^{k-1}))$, and $\boldsymbol{D}(\boldsymbol{\Psi}^{k-1}) = \mathrm{col}(\boldsymbol{d}_1(\boldsymbol{\psi}_1^{k-1}), \ldots, \boldsymbol{d}_N(\boldsymbol{\psi}_N^{k-1}))$. Note that we have represented the approximated gradient vector in (25) as the sum of the true gradient, $\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1})$, and the gradient noise, $\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})$, since we will explore their properties separately in the following analysis. Now, because we are interested in evaluating the MSE performance of the algorithms, we further derive the error recursion by a state transformation

$$\tilde{\boldsymbol{\Psi}}^{k-1} = (A_1 \otimes I_n)\tilde{\boldsymbol{\Theta}}^{k-1}, \tag{27}$$

$$\tilde{\boldsymbol{\Phi}}^k = \tilde{\boldsymbol{\Psi}}^{k-1} - (\Delta \otimes I_n)\left[\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1}) + \boldsymbol{D}(\boldsymbol{\Psi}^{k-1})\right], \tag{28}$$

$$\tilde{\boldsymbol{\Theta}}^k = (A_2 \otimes I_n)\tilde{\boldsymbol{\Phi}}^k, \tag{29}$$

where the extended error vectors are defined as $\tilde{\boldsymbol{\Psi}}^k = \boldsymbol{\Psi} - \theta^* \mathbf{1}_N$, $\tilde{\boldsymbol{\Phi}}^k = \boldsymbol{\Phi} - \theta^* \mathbf{1}_N$, and $\tilde{\boldsymbol{\Theta}}^k = \boldsymbol{\Theta} - \theta^* \mathbf{1}_N$.

Note however that the gradient term, $\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1})$, and the noise term, $\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})$, in (28) remain coupled with the original state, $\boldsymbol{\Psi}^{k-1}$. Expression (20) provides a link that can be used to relate the gradient noise, $\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})$, with the error state, $\tilde{\boldsymbol{\Psi}}^{k-1}$, due to Assumption 2. To relate the gradient, $\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1})$, with the error term $\tilde{\boldsymbol{\Psi}}^{k-1}$, we apply the mean value theorem [40], which states that, for a twice-differentiable function $g(z) : \mathbb{R}^m \to \mathbb{R}$, the following relation holds

$$\nabla_z g(y) = \nabla_z g(x) + \left[\int_0^1 \nabla_z^2 g[x + \tau(y - x)]d\tau\right](y - x). \tag{30}$$

Hence, substituting $\boldsymbol{\Psi}^{k-1}$ and $\theta^* \mathbf{1}_N$ into (30), we can obtain

$$\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1}) = \boldsymbol{\Gamma}(\theta^* \mathbf{1}_N) + \left[\int_0^1 \nabla_\theta \boldsymbol{\Gamma}(\theta^* \mathbf{1}_N + \tau\tilde{\boldsymbol{\Psi}}^{k-1}]d\tau\right]\tilde{\boldsymbol{\Psi}}^{k-1}. \tag{31}$$

Then, we have

$$\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1}) = \boldsymbol{P}^{k-1}\tilde{\boldsymbol{\Psi}}^{k-1}, \tag{32}$$

by substituting $\boldsymbol{\Gamma}(\theta^* \mathbf{1}_N) = \mathbf{0}_{nN}$ into (31) and denoting

$$\boldsymbol{P}^{k-1} = \int_0^1 \nabla_\theta \boldsymbol{\Gamma}(\theta^* \mathbf{1}_N + \tau\tilde{\boldsymbol{\Psi}}^{k-1})d\tau. \tag{33}$$

The matrix $\boldsymbol{P}^{k-1}$ plays an important role in the performance analysis, of which the properties will be further elaborated later. It should be noted from the definition of $\boldsymbol{\Gamma}(\boldsymbol{\Psi}^{k-1})$ that $\boldsymbol{P}^{k-1}$ is a block diagonal matrix, where each block is symmetrical and of the form

$$\boldsymbol{P}_i^{k-1} = \int_0^1 \nabla_{\boldsymbol{\theta}}^2 \mathcal{R}_i(\theta^* \mathbf{1}_N + \tau\tilde{\boldsymbol{\Psi}}^{k-1})d\tau. \tag{34}$$

Thus, we can denote $\boldsymbol{P}^{k-1} = \mathrm{diag}(\boldsymbol{P}_1^{k-1}, \ldots, \boldsymbol{P}_N^{k-1})$. Now, applying (32) to (28) yields

$$\begin{aligned}\tilde{\boldsymbol{\Phi}}^k &= \tilde{\boldsymbol{\Psi}}^{k-1} - (\Delta \otimes I_n)\left[\boldsymbol{P}^{k-1}\tilde{\boldsymbol{\Psi}}^{k-1} + \boldsymbol{D}(\boldsymbol{\Psi}^{k-1})\right] \\ &= \left[I_{nN} - (\Delta \otimes I_n)\boldsymbol{P}^{k-1}\right]\tilde{\boldsymbol{\Psi}}^{k-1} - (\Delta \otimes I_n)\boldsymbol{D}(\boldsymbol{\Psi}^{k-1}).\end{aligned} \tag{35}$$

Combining (27), (29) and (35) together yields

$$\begin{aligned}\tilde{\boldsymbol{\Theta}}^k =& (A_2 \otimes I_n)\left[I_{nN} - (\Delta \otimes I_n)\boldsymbol{P}^{k-1}\right](A_1 \otimes I_n)\tilde{\boldsymbol{\Theta}}^{k-1} \\ &- (A_2\Delta \otimes I_n)\boldsymbol{D}(\boldsymbol{\Psi}^{k-1}).\end{aligned} \tag{36}$$

We define the MSE vectors of the network as

$$\mathbb{M}^k = \mathrm{col}(\mathbb{E}\,\|\tilde{\boldsymbol{\psi}}_1^k\|^2, \ldots, \mathbb{E}\,\|\tilde{\boldsymbol{\psi}}_N^k\|^2), \tag{37}$$

$$\mathbb{N}^k = \mathrm{col}(\mathbb{E}\,\|\tilde{\boldsymbol{\phi}}_1^k\|^2, \ldots, \mathbb{E}\,\|\tilde{\boldsymbol{\phi}}_N^k\|^2), \tag{38}$$

$$\mathbb{H}^k = \mathrm{col}(\mathbb{E}\,\|\tilde{\boldsymbol{\theta}}_1^k\|^2, \ldots, \mathbb{E}\,\|\tilde{\boldsymbol{\theta}}_N^k\|^2). \tag{39}$$

Note that the MSE vectors have been defined in an agent-wise manner. Taking the Euclidean norm for each agent's error dynamics in (27) and (29), we have

$$\left\|\tilde{\boldsymbol{\psi}}_i^{k-1}\right\|^2 = \left\|\sum_{j=1}^N a_{1,ij}\tilde{\boldsymbol{\theta}}_j^{k-1}\right\|^2. \tag{40}$$

Since $\|\boldsymbol{\theta}_i^{\tilde{k-1}}\|$ is a convex function of $\tilde{\boldsymbol{\theta}}_i^{k-1}$, and $\sum_{j=1}^N a_{1,ij}\tilde{\boldsymbol{\theta}}_j^{k-1}$ is a convex combination of $\tilde{\boldsymbol{\theta}}_i^{k-1}$, applying Jensen's inequality to (40) leads to

$$\left\|\tilde{\boldsymbol{\psi}}_i^{k-1}\right\|^2 \le \sum_{j=1}^N a_{1,ij}\left\|\tilde{\boldsymbol{\theta}}_j^{k-1}\right\|^2. \tag{41}$$

Taking the expectation of both sides of (41), the augmented MSE vector of $\mathbb{M}^{k-1}$ can be obtained as

$$\mathbb{M}^{k-1} \preceq A_1\mathbb{H}^{k-1}, \tag{42}$$

where the curled symbol $\preceq$ denotes component-wise inequality. Following similar lines, we have

$$\mathbb{H}^k \preceq A_2\mathbb{N}^k. \tag{43}$$

Now, the remaining task is to construct the relation between $\mathbb{N}^k$ and $\mathbb{M}^{k-1}$ by using expression (35) and Assumptions 1 and 2. We will further explore the structure of (35) by analysing the properties of each agent's error vector $\tilde{\boldsymbol{\Phi}}_i^k$. Taking the Euclidean norm and expectation of $\tilde{\boldsymbol{\Phi}}_i^k$ in (35) leads to

$$\begin{aligned}\mathbb{N}_i^k =& \mathbb{E}\,\|\left[I_n - (\eta_i \otimes I_n)\boldsymbol{P}_i^{k-1}\right]\tilde{\boldsymbol{\psi}}_i^{k-1}\|^2 \\ &+ \mathbb{E}\,\|(\eta_i \otimes I_n)\boldsymbol{d}_i(\boldsymbol{\psi}_i^{k-1})\|^2,\end{aligned} \tag{44}$$

where we have used the first condition of the gradient noises in (17). Due to Assumption 1, it follows from (34) that the symmetrical matrix $\boldsymbol{P}_i^{k-1}$ satisfies

$$\underline{\alpha}_i I_n \leq \boldsymbol{P}_i^{k-1} \leq \bar{\alpha}_i I_n. \tag{45}$$

To bound the first term in the right-hand-side of (44), we can apply (45) to derive

$$0 \leq \left[I_n - (\eta_i \otimes I_n)\boldsymbol{P}_i^{k-1}\right]^2 \leq \bar{\lambda}_i^2 I_n, \tag{46}$$

where $\bar{\lambda}_i^2 = \max\{(1-\eta_i\underline{\alpha}_i)^2, (1-\eta_i\bar{\alpha}_i)^2\}$. We can appeal to Assumption 2 to bound the second term in (44) by using the relation in (20), which yields

$$\mathbb{E}\,\|(\eta_i \otimes I_n)\boldsymbol{d}_i(\boldsymbol{\psi}_i^{k-1})\|^2 \leq \eta_i^2\bar{\zeta}^2\|\tilde{\boldsymbol{\psi}}_i^{k-1}\|^2 + \eta_i^2\bar{\xi}^2. \tag{47}$$

Finally, applying the results in (46) and (47) to (44), the relation between $\mathbb{N}^k$ and $\mathbb{M}^{k-1}$ is obtained as

$$\mathbb{N}^k \preceq \Xi\mathbb{M}^{k-1} + \Pi, \tag{48}$$

where $\Xi = \mathrm{diag}(\bar{\lambda}_1^2 + \eta_1^2\bar{\zeta}^2, \ldots, \bar{\lambda}_N^2 + \eta_N^2\bar{\zeta}^2)$, and $\Pi = \mathrm{col}(\eta_1^2\bar{\xi}^2, \ldots, \eta_N^2\bar{\xi}^2)$. Combining the results in (42), (43) and (48), we derive the closed-loop MSE error $\mathbb{H}^k$ as

$$\mathbb{H}^k \preceq A_2\Xi A_1\mathbb{H}^{k-1} + A_2\Pi. \tag{49}$$

We are now ready to present the convergence of the MSE to a bounded region of the optimal weights.

Theorem 1: Let Assumptions 1 and 2 hold. If the constant learning rates satisfy

$$0 < \eta_i < \min\left\{\frac{2\underline{\alpha}_i}{\underline{\alpha}_i^2 + \bar{\zeta}^2}, \frac{2\bar{\alpha}_i}{\bar{\alpha}_i^2 + \bar{\zeta}^2}\right\}, \tag{50}$$

the worst MSE, $\|\mathbb{H}^k\|_\infty$, in the network converges to

$$\lim_{k\to\infty}\|\mathbb{H}^k\|_\infty \leq \frac{\max_{i\in\mathcal{V}}\{\eta_i^2\bar{\xi}^2\}}{1 - \max_{i\in\mathcal{V}}\{\bar{\lambda}_i^2 + \eta_i^2\bar{\zeta}^2\}}. \tag{51}$$

Proof: From (49), we can obtain the expression of the worst MSE in the network as

$$\begin{aligned}\|\mathbb{H}^k\|_\infty &\leq \|A_2\Xi A_1\|_\infty\|\mathbb{H}^{k-1}\|_\infty + \|A_2\Pi\|_\infty \\ &\leq \|\Xi\|_\infty\|\mathbb{H}^{k-1}\|_\infty + \|\Pi\|_\infty,\end{aligned} \tag{52}$$

where $\|A_1\|_\infty = \|A_2\|_\infty = 1$ has been used. To guarantee the convergence of (52), it should be satisfied that $\|\Xi\|_\infty < 1$. Thus, for all $i \in \mathcal{V}$, we have

$$\bar{\lambda}_i^2 + \eta_i^2\bar{\zeta}^2 < 1, \tag{53}$$

which means

$$(1-\eta_i\underline{\alpha}_i)^2 + \eta_i^2\bar{\zeta}^2 < 1, \tag{54}$$

$$(1-\eta_i\bar{\alpha}_i)^2 + \eta_i^2\bar{\zeta}^2 < 1. \tag{55}$$

Solving (54) and (55) results in

$$0 < \eta_i < \frac{2\underline{\alpha}_i}{\underline{\alpha}_i^2 + \bar{\zeta}^2},\ 0 < \eta_i < \frac{2\bar{\alpha}_i}{\bar{\alpha}_i^2 + \bar{\zeta}^2}. \tag{56}$$

We can recurrently iterate (52) to establish the relation between $\|\mathbb{H}^k\|_\infty$ and $\|\mathbb{H}^0\|_\infty$

$$\|\mathbb{H}^k\|_\infty \leq (\|\Xi\|_\infty)^k\|\mathbb{H}^0\|_\infty + \sum_{j=0}^{k-1}(\|\Xi\|_\infty)^j\|\Pi\|_\infty, \tag{57}$$

where $(\cdot)^k$ denotes the kth power of its argument. Observe that

$$\lim_{k\to\infty}\sum_{j=0}^{k-1}(\|\Xi\|_\infty)^j\|\Pi\|_\infty = \frac{\|\Pi\|_\infty}{1-\|\Xi\|_\infty}. \tag{58}$$

Therefore, substituting (58) and $\lim_{k\to\infty}\|\Xi\|_\infty^k = 0$ into expression (57), we can get

$$\lim_{k\to\infty}\|\mathbb{H}^k\|_\infty \leq \frac{\|\Pi\|_\infty}{1-\|\Xi\|_\infty} \tag{59}$$

$$= \frac{\max_{i\in\mathcal{V}}\{\eta_i^2\bar{\xi}^2\}}{1 - \max_{i\in\mathcal{V}}\{\bar{\lambda}_i^2 + \eta_i^2\bar{\zeta}^2\}}. \tag{60}$$

This completes the proof. ■

Remark 2: In Theorem 1, expression (50) provides an upper bound for designing the learning rates, based on which we can prove the convergence of the proposed algorithms. It should be noted that the learning rates can vary among the agents, which is more flexible than the homogeneous step sizes used in [2, 4, 13, 15]. Relation (51) reveals how the learning rates, the bounds of the Hessian matrix and the noise variance will affect the performance of the algorithms. It also indicates that the optimality error can be reduced by choosing small enough learning rates. The designer is able to balance between the convergence speed and the optimality error according to the real applications by adjusting the heterogeneous learning rates.

From (50), we observe that three parameters, $\underline{\alpha}_i, \bar{\alpha}_i$ and $\bar{\zeta}^2$, should be known when designing the learning rates. It is worth mentioning that the bounds of the empirical risk, $\underline{\alpha}_i$ and $\bar{\alpha}_i$, can be easily acquired from the loss functions, while one of coefficients in the gradient noises, $\bar{\zeta}^2$, should be approximated according to the practical applications. Fortunately, for most problems of practical interest [15, 38], the variances of the gradient noises can be modelled by a constant value, $\bar{\xi}^2$, that is, the term $\bar{\zeta}^2\|\tilde{\boldsymbol{\theta}}\|^2$ in (20) is equal to zero, in which case the learning rates can be chosen by setting $\bar{\zeta}^2 = 0$. If the gradient vectors are free of noises, i.e., $\bar{\zeta}^2 = 0$ and $\bar{\xi}^2 = 0$, we can establish the convergence results for the true gradient case, as stated in the following corollary. It is worth noting that our results for the noise-free case coincide with the classic results in [9, 19].

Corollary 1: Under Assumption 1 and $\bar{\zeta}^2 = 0$ and $\bar{\xi}^2 = 0$, if the learning rates satisfy

$$0 < \eta_i < \frac{2}{\bar{\alpha}_i}, \tag{61}$$

then the deterministic error vector $\mathbb{H}^k = \mathrm{col}(\|\tilde{\theta}_1^k\|^2, \ldots, \|\tilde{\theta}_N^k\|^2)$ converges to zero as $k \to \infty$.

Proof: The result follows directly from Theorem 1. ■

Now, we will examine how fast the proposed algorithm converges to the steady-state value. Instead of deriving the upper bound for the MSE, we need to explore the recursion of $\mathbb{E}\,\|\tilde{\boldsymbol{\Theta}}^k\|^2$. Retuning to (36), we can get

$$\mathbb{E}\,\|\tilde{\boldsymbol{\Theta}}^k\|^2 = \mathbb{E}\,\|\tilde{\boldsymbol{\Theta}}^{k-1}\|_{\boldsymbol{B}}^2 + \mathbb{E}\,\|(A_2\Delta\otimes I_n)\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})\|^2, \tag{62}$$

where $\boldsymbol{B} = (A_1\otimes I_n)\left[I_{nN} - (\Delta\otimes I_n)\boldsymbol{P}^{k-1}\right](A_2\otimes I_n)(A_2\otimes I_n)^T\left[I_{nN} - (\Delta\otimes I_n)\boldsymbol{P}^{k-1}\right](A_1\otimes I_n)^T$. According to Theorem 1, we have the worst MSE converges to a small neigh-

bourhood of zero, when the step size η_i is small enough. It then follows from (33) that $\boldsymbol{P}_i^{k-1}$ can be approximated by

$$\boldsymbol{P}_i^{k-1} \approx \nabla^2_{\boldsymbol{\theta}} \mathcal{R}_i(\theta^* \mathbf{1}_N). \tag{63}$$

Applying (63), we can approximate $\boldsymbol{B}$ by a deterministic quantity

$$\begin{aligned} B =&(A_1 \otimes I_n)\left[I_{nN} - (\Delta \otimes I_n)D\right](A_2 \otimes I_n) \\ &\times (A_2 \otimes I_n)^T \left[I_{nN} - (\Delta \otimes I_n)D\right](A_1 \otimes I_n)^T \end{aligned} \tag{64}$$

where $D \triangleq \operatorname{diag}(\nabla^2_{\boldsymbol{\theta}} \mathcal{R}_1(\theta^* \mathbf{1}_N), \ldots, \nabla^2_{\boldsymbol{\theta}} \mathcal{R}_N(\theta^* \mathbf{1}_N))$.

To deal with the second term in (62), the covariance matrix of the noise $\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})$ should be evaluated, given by

$$\bar{C} = \mathbb{E}\{\boldsymbol{D}(\theta^* \mathbf{1}_N)\boldsymbol{D}^T(\theta^* \mathbf{1}_N)\}. \tag{65}$$

Thus,

$$\begin{aligned} &\mathbb{E}\,\|(A_2\Delta \otimes I_n)\boldsymbol{D}(\boldsymbol{\Psi}^{k-1})\|^2 \\ &= \mathbb{E}\{\boldsymbol{D}^T(\boldsymbol{\Psi}^{k-1})(A_2\Delta \otimes I_n)^T(A_2\Delta \otimes I_n)\boldsymbol{D}(\boldsymbol{\Psi}^{k-1}) \\ &= \operatorname{tr}\{(A_2\Delta \otimes I_n)^T \bar{C}(A_2 \otimes I_n)\}. \end{aligned} \tag{66}$$

Substituting (64) and (66) into (62) yields

$$\mathbb{E}\,\|\tilde{\boldsymbol{\Theta}}^k\|^2 \approx \mathbb{E}\,\|\tilde{\boldsymbol{\Theta}}^{k-1}\|^2_B + \operatorname{tr}((A_2\Delta \otimes I_n)^T \bar{C}(A_2 \otimes I_n)), \tag{67}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of its argument. Under small step size η_i, we can regard $\operatorname{tr}\{(A_2\Delta \otimes I_n)^T \bar{C}(A_2 \otimes I_n)$ as a small perturbation imposed on the error recursion. Hence, the convergence rate can be approximated by $\rho(B)$ i.e., the spectral radius of B. Therefore, we need to demonstrate the stability of B, which is equivalent to the stability of $(A_1 \otimes I_n)\left[I_{nN} - (\Delta \otimes I_n)D\right](A_2 \otimes I_n)$. Noting that the maximum norms of A_1 and A_2 are one, then we have the spectral radius of B,

$$\rho(B) = [\rho(I_{nN} - (\Delta \otimes I_n)D)]^2. \tag{68}$$

According to the conditions of the learning rate in (50) and the definition of D, it is straightforward to obtain that all the eigenvalues of the B are within the unit circle. Thus, the convergence of the error dynamics is guaranteed with a convergence rate $\rho(B)$ that depends on the selection of the learning rate η_i.

Remark 3: Under connected graphs, the proposed algorithms will converge to a small neighbourhood of the optimal solution, upper bounded by (51). If the initial parameters of the neural networks are specified at different values, the algorithms will first make $\boldsymbol{\theta}_i$ move closer to each other, and the transient speed of this process is determined by the communication structure [41]. This transient process is much faster than the learning speed, and can be ignored in real implementation.

Remark 4: To reduce the communication burden, local stochastic gradient descent (LocalSGD) [42] and federated averaging algorithms [43] have been proposed. The agents can perform multiple local learning steps between two consecutive communication instances. It is worth mentioning that those works do not consider the communication graphs. They assume that all local weights are available while computing the average of them, usually performed by a central server. In our work, we do not require a central server to communicate with all agents. Instead, the local agents can only communicate with a limited number of neighbours, by which the communication requirement is reduced at each iteration.

IV. Simulation and Discussion

In this section, we will examine the convergence and performance of the proposed algorithms by simulation examples. In the first example, we consider an application of the proposed algorithms for distributed optimisation, where true gradients are used without noises. Limitations of this deterministic implementation are then discussed. In the second example, stochastic gradients will be used to train a set of distributed neural networks for handwriting recognition, by which the performance of the proposed algorithms will be discussed comparing with the local training algorithms. Furthermore, we will demonstrate how the learning rates affect the error dynamics of the parameters.

Now, we summarise the implementation structure of our distributed solution to the distributed learning problems. Algorithm 1 illustrates the detailed procedures of the proposed framework. Note that either Step 5 or Step 7 should be implemented, which corresponds to the CTL and LTC algorithms, respectively.

Algorithm 1 The Implementation Structure.

Initialisation:

for each agent $i \in \mathcal{V}$

1. initialise the parameters of the local neural network, θ_i^0;
2. establish connections with its adjacent neighbours, $\mathcal{N}_i$, in the network;
3. allocate the weights $c_{ij}, \forall j \in \mathcal{V}$, of its combination matrices;
4. design the learning rates η_i according to Theorem 1.

Iteration:

set $k := k + 1$, for $i \in \mathcal{V}$

5. update the intermediate variable $\boldsymbol{\psi}_i^{k-1}$ in (13) by communicating with its neighbours $\mathcal{N}_i$;
6. learn the intermediate variable $\boldsymbol{\phi}_i^k$ in (14) using local data samples in $\mathcal{D}_i$;
7. update parameters of neural networks $\boldsymbol{\theta}_i^k$ in (15) by communicating with its neighbours $\mathcal{N}_i$.

End if termination condition is satisfied or iteration budget is approached.

A. Example 1: Deterministic Optimisation

In this subsection, we implement the proposed framework for a logistic regression problem [27, 44]. This type of problem covers a number of important applications, for example, spam email classification and medical diagnosis [44]. Consider a network of N agents, where each of them possesses a number of m_i data samples in $\mathcal{D}_i$ and each sample has an h-dimensional feature space, $s_i^k \in \mathbb{R}^h$, and a binary label $l_i^k \in \{\pm 1\}$. The network objective is to solve the convex problem

$$\min_{\theta} \sum_{i=1}^{N} \sum_{k=1}^{m_i} \log\left[1 + e^{-l_i^k \theta^T s_i^k}\right] + \frac{\rho}{2}\|\theta\|^2, \tag{69}$$

where ρ denotes the regularisation parameter. Since data samples are collected by different agents who are not willing to share their private datasets, it is of practical significance

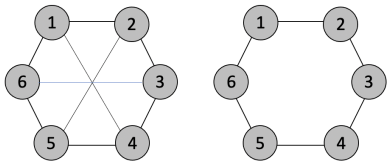


Fig. 1: The communication topologies.

to deploy the distributed framework. In this application, we assume the network contains six agents and each of them has very limited data samples such that true gradients can be calculated at each iteration, i.e., $\boldsymbol{d}_i = 0$. Bidimensional samples are used, which are randomly generated by a multivariate Gaussian distribution. We label the samples as $l_i^k = 1$ for those generated with a covariance matrix equal to the identity and mean equal to $(0, 0)$, and as $l_i^k = -1$ for mean equal to $(5, 3)$.

The communication topologies used in this simulation are displayed in Fig. 1, where the first graph possesses higher connectivity than the second one. For comparison purposes, we have included two commonly-used optimisation algorithms: the alternating direction method of multipliers (ADMM) [9] and distributed proximal gradient (DPG) [45]. The learning steps are kept the same as $\eta_i = 0.01$. The LTC and CTL algorithms show similar learning profiles, and therefore we only present the results using LTC algorithm to avoid redundancy. Because this problem is deterministic, we can obtain the exact optimal parameters by solving it using many effective optimisation toolboxes in centralised settings, and therefore, we will compare the obtained results using the proposed algorithms with the true optimal solutions. Figs. 2a and 2b demonstrate the errors of the weights and the cost values, respectively. It is clear that the change of communication graphs does not significantly influence the learning behaviours of the proposed LTC algorithms. As long as the communication graph is connected, the convergence speed is mainly dominated by the learning rate, instead of the graph structures. However, it should be noted that the communication topology will impact the consensus dynamics at the first few iterations, as shown in Fig. 3. Strong connectivity will accelerate the reduction of the discrepancy among the states, caused by different initial conditions.

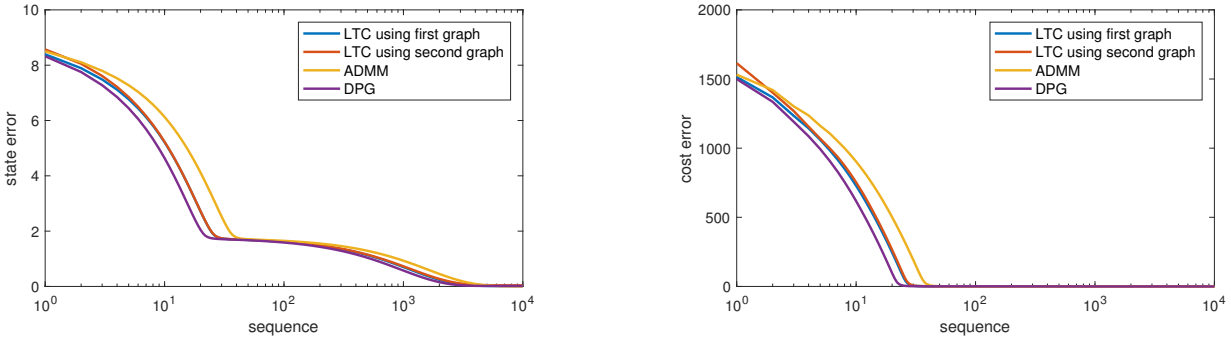
In Fig. 2, the ADMM algorithm exhibits the slowest convergence speed compared with the other three methods. Moreover, the ADMM-based approach utilises more auxiliary variables, consisting of two primal-variable updates and one dual-variable update, performed in an alternating manner. This complicates the algorithm deployment, and aggravates the computational burden. On the other hand, the DPG-based method shows a slightly fast convergence speed. To implement such an algorithm, additional sub-optimisation problems have to be solved, since it uses proximal operators. In comparison, the proposed methods in this paper are easy to be implemented, and they also show comparable convergence performance.

B. Example 2: Training using Stochastic Gradient

The previous example assumes that the datasets are small, and exact gradients can be calculated by traversing all the samples in the datasets. There are at least three reasons that discourage us from using exact gradients for training distributed neural networks: 1). calculating the exact gradients for all samples in the datasets is time-consuming and impractical, as the datasets are usually very large; 2). gradient differences are always present among distributed machines, and will be diffused across all participants via the communication network, since the data samples cannot be shared due to privacy concerns; 3). data samples are not available beforehand, and instead, they are generated during real-time training. In this example, we implement the proposed algorithms to a handwritten digit recognition problem. Six training agents are considered with a total number of 5000 samples that are evenly distributed to the agents. Each sample has a 400-dimension feature space, and a 10-class label.

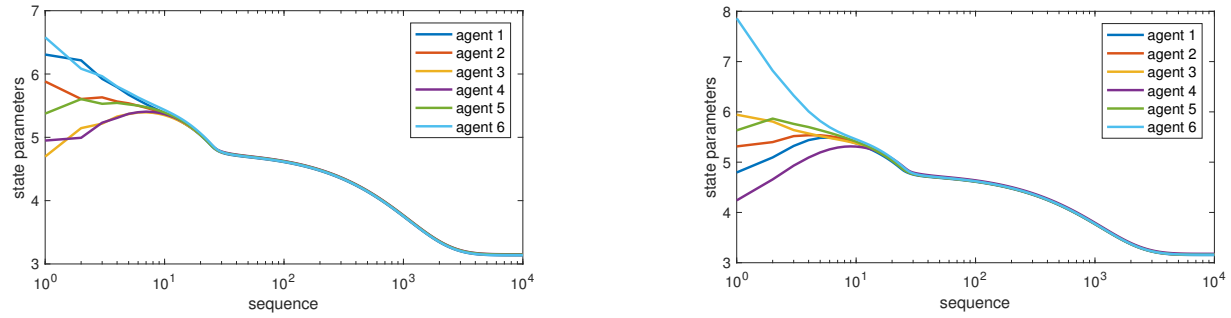
We first implement the consensus-based training algorithms in this paper with constant learning rates. For comparison purposes, we deploy other learning algorithms in existing literature, including the LocalSGD [42] and the local training algorithms without exchanging the parameters of neural networks. All implementations use the same local datasets, test datasets and learning rates. For the consensus-based learning algorithms, we can design the learning rates according to the results in Theorem 1. In the LocalSGD, the communication intervals have been set as $b = 5$ and 10. The profiles of the prediction accuracy are shown in Fig. 4. In general, the prediction accuracy obtained from the consensus algorithms exhibits less variance among the agents, that is, the parameters of the distributed neural networks have similar values, since the weighted average of the local parameters is employed at every consensus step. On the other hand, the performance of different agents using the local training method demonstrates larger differences, because the agents train their neural networks independently without any information exchange. Noticeably, the LocalSGD algorithm (with less communication requirement) converges slower than the consensus-based algorithm. Initially, the prediction accuracy is deteriorated at every communication interval, since averaging the parameters causes large differences among the agents. After some iterations, the performance is significantly improved.

To illustrate the advantages of using the proposed consensus-based learning algorithms, we present averages of the prediction accuracy in Fig. 5. It can be noticed that, after some iterations, the average prediction accuracy stabilises to a value of 90% by using the local training algorithms. In comparison, the parallel algorithms achieve a fast learning speed, since the distributed neural networks are trained simultaneously. The consensus-based learning algorithms still demonstrate an increasing trend (up to 95% by 1000 iterations) in the later period of learning, as shown in the enlarged part of Fig. 5. Although the global dataset does not increase, the performance of all agents in the network are improved by means of cooperation, which is of significant importance in learning over networks. The LocalSGD algorithm shows fluctuations



(a) (b)

Fig. 2: Performance of deterministic optimisation with different learning algorithms: (a) error of parameter θ_i, (b) error of the cost values.



(a) (b)

Fig. 3: Iterations of one state variable using LTC with different communication graphs: (a) the first graph, (b) the second graph.

at the early stage of learning, but gradually outperforms the local learning algorithms after some iterations. Overall, the LocalSGD converges in a slower speed compared with the algorithms proposed in this paper.

In order to verify the performance results in (51), we conduct a comparison study with three different learning rates $\eta_i = 0.01, 0.05$ and 0.1. The empirical risks of the network are designed according to [46], and we choose the range of the Hessian coefficients, i.e., the regularisation factors, for each agent from $(1, 10)$. In this case, the upper and lower bounds of the Hessian coefficients are $\bar{\alpha}_i = 10$ and $\underline{\alpha}_i = 1$. To approximate the gradient noises, we stimulate the neural networks with a set of random samples for different parameters θ, by which the variance of gradient noises is approximately bounded by a constant value $\bar{\xi}^2 = \mathbb{E}[(\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}}_i(\boldsymbol{\theta}) - \mathbb{E}[\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{R}}_i(\boldsymbol{\theta})])^2] = 1625.3$. It is worth mentioning that the parameters should be approximated before training such that desirable learning rates can be specified. Substituting the experimental results into (51), the worst MSE, $\|\mathbb{H}^k\|_\infty$, can be bounded theoretically as 0.27, 16.3 and 64.2 for the learning rates $\eta_i = 0.01, 0.05$ and 0.1, respectively. The simulation results of the worst MSE are presented in Fig. 6, where the theoretical bounds are shown in the dotted lines. It is clear that the simulation results are indeed confined to the theoretical bounds, and the relation (51) is conservative as the worst scenario has been considered. It is worth mentioning that, as the learning rate increases, the fluctuations of the parameters become larger, since the instant samples with gradient noises impose more influence on the learning behaviours of agents. In practice, the designer should balance between the convergence speed and the optimality error of the solution. The nominal performance analysis is conducive to the selection of the learning rates.

V. Conclusion

In this paper, distributed training using consensus-based algorithms has been studied. Due to the ubiquitous gradient noises in learning, we have proposed two learning algorithms that apply approximated gradients and constant learning rates. The introduction of gradient noises reveals the learning behaviours of the distributed agents in real applications, and the constant learning rates enable the proposed algorithms to be employed in online optimisation and learning problems. A unified expression for the two algorithms has been presented to facilitate the analysis of convergence and performance. Two assumptions are introduced, which are in general less restrictive than the existing studies. Based on those conditions, we have proved theoretically the convergence and performance of the MSE, by which the learning rates can be designed accordingly. Two simulation examples have been carried out to show the effectiveness and importance of our study.

References

[1] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, 2019.

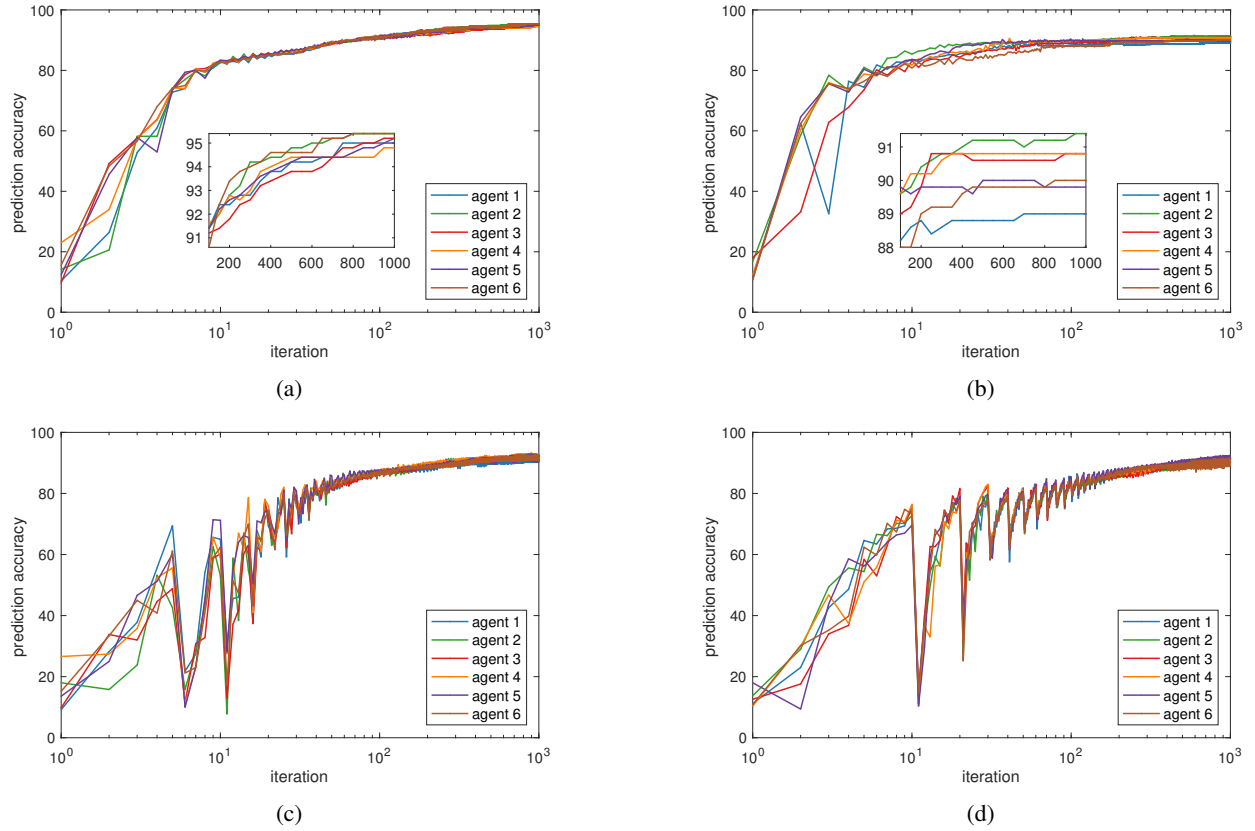


Fig. 4: Prediction accuracy using different training strategies: (a) consensus-based algorithm, (b) local training algorithm, (c) LocalSGD with $b = 5$, (d) LocalSGD with $b = 10$.

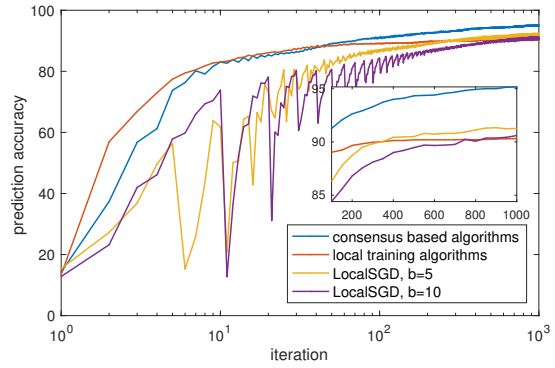


Fig. 5: Performance comparison using different algorithms.

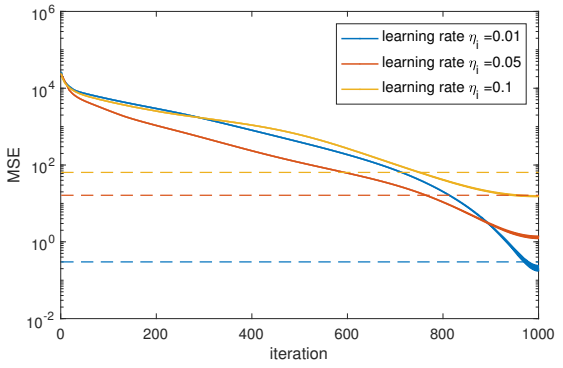


Fig. 6: The worst MSE in the network using different learning rates.

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.
[3] Z. Li, Z. Ding, J. Sun, and Z. Li, "Distributed adaptive convex optimization on directed graphs via continuous-time algorithms," *IEEE Trans. Autom. Control*, vol. 63, no. 5, pp. 1434–1441, 2018.
[4] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1771–1778, 2020.
[5] Z. Li and Z. Ding, "Distributed multiobjective optimization for network resource allocation of multiagent systems," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2019.2961475.
[6] B. Liu and Z. Ding, "Distributed heuristic adaptive neural networks with variance reduction in switching graphs," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2019.2956291.
[7] T.-H. Chang, A. Nedi, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 1524–1538, 2014.
[8] B. Gharesifard and J. Corts, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 781–786, 2014.
[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[10] Z. Deng, S. Liang, and Y. Hong, "Distributed continuous-time algorithms for resource allocation problems over weight-balanced digraphs," *IEEE Trans. Cybern.*, vol. 48, no. 11, pp. 3116–3125, 2018.
[11] B. Ning, Q. Han, and Z. Zuo, "Distributed optimization for multiagent systems: An edge-based fixed-time consensus approach," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 122–132, 2019.
[12] Z. Li, Z. Dong, Z. Liang, and Z. Ding, "Surrogate-based distributed optimisation for expensive black-box functions," *Automatica*, vol. 125, 2021.

[13] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2–12, 2014.
[14] R. Zhang and Q. Zhu, "A game-theoretic approach to design secure and resilient distributed support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5512–5527, 2018.
[15] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Int. Conf. Machine Learn.*, pp. 4848–4856, PMLR, 2018.
[16] X. Wu, J. Zhang, and F. Y. Wang, "Stability-based generalization analysis of distributed learning algorithms for big data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 801–812, 2020.
[17] W. Kim, M. S. Stankovi, K. H. Johansson, and H. J. Kim, "A distributed support vector machine learning over wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2599–2611, 2015.
[18] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012.
[19] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.
[20] A. Nedic and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.
[21] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.
[22] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, 1986.
[23] F. Salehisadaghiani and L. Pavel, "Distributed Nash equilibrium seeking: A gossip-based algorithm," *Automatica*, vol. 72, pp. 209–216, 2016.
[24] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2012.
[25] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*. Springer, 2008.
[26] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York: Springer-Verlag, 2001.
[27] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[28] Z. Li, Z. Wu, Z. Li, and Z. Ding, "Distributed optimal coordination for heterogeneous linear multi-agent systems with event-triggered mechanisms," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1763–1770, 2020.
[29] C. Wang, Z. Zuo, Q. Gong, and Z. Ding, "Formation control with disturbance rejection for a class of lipschitz nonlinear systems," *Sci. China Inf. Sci.*, vol. 60, no. 7, 2017.
[30] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th Symp. Oper. Syst. Des. Implementation*, 2014, pp. 583–598.
[31] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.
[32] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1229–1262, 2014.
[33] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, 2020.
[34] Y. Liu, C. Li, and Z. Zhang, "Diffusion sparse least-mean squares over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4480–4485, 2012.
[35] X. Shi, J. Cao, and W. Huang, "Distributed parametric consensus optimization with an application to model predictive consensus problem," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2024–2035, 2017.
[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Uni. Press, 2004.
[37] Y. Zhu, W. Yu, G. Wen, G. Chen, and W. Ren, "Continuous-time distributed subgradient algorithm for convex optimization with general constraints," *IEEE Trans. Autom. Control*, vol. 64, no. 4, pp. 1694–1701, 2019.
[38] 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, 2012.
[39] Z. Li, Z. Duan, G. Chen, and L. Huang, "Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 57, no. 1, pp. 213–224, 2010.
[40] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York, NY, USA: McGraw-Hill New York, 1976.
[41] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networkspart i: Transient analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, 2015.
[42] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.
[43] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artif. Intell. Stat.*, PMLR, Conference Proceedings, pp. 1273–1282, 2017.
[44] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013, vol. 398.
[45] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6013–6023, 2015.
[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

Zhongguo Li (Student Member, IEEE) received the B.Eng. degree in electrical and electronic engineering from the University of Manchester, Manchester, UK, and the joint B.Eng. degree in communication engineering from Jilin University, Jilin, China, in 2017. He is currently a Ph.D. candidate in electrical and electronic engineering at the University of Manchester.

From November 2020, he has been a Research Associate with the Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, UK. His research interests include distributed optimisation, game theory, decision-making and their applications in autonomous systems.

Bo Liu received the B.Eng. degree and M.Eng. degree from Wuhan University of Technology, Wuhan, China, in 2014 and 2017, respectively. He is currently pursing the Ph.D. degree in Electrical and Electronic Engineering at the University of Manchester, Manchester, U.K.

His current research interests include distributed optimisation, distributed machine learning, and their applications.

Zhengtao Ding (Senior Member, IEEE) received the B.Eng. degree from Tsinghua University, Beijing, China, in 1984, and the M.Sc. degree in systems and control, and the Ph.D. degree in control systems from the University of Manchester Institute of Science and Technology, Manchester, U.K. in 1986 and 1989, respectively. After working as a Lecturer with Ngee Ann Polytechnic, Singapore, for ten years, he joined the University of Manchester in 2003, where he is currently Professor of Control Systems with the Department of Electrical and Electronic Engineering. He is the author of the book: Nonlinear and Adaptive Control Systems (IET, 2013) and has published over 260 research articles.

His research interests include nonlinear and adaptive control theory and their applications, more recently network-based control, distributed optimisation and distributed machine learning, with applications to power systems and robotics. Prof. Ding has served as an Associate Editor for the IEEE Transactions on Automatic Control, IEEE Control Systems Letters, and several other journals. He is a member of IEEE Technical Committee on Nonlinear Systems and Control, IEEE Technical Committee on Intelligent Control, and IFAC Technical Committee on Adaptive and Learning Systems.