Consensus learning for distributed fuzzy neural network in big data environment

Ye Shi, *Member, IEEE*, Chin-Teng Lin, *Fellow, IEEE*, Yu-Cheng Chang, *Student Member, IEEE*, Weiping Ding, *Senior Member, IEEE*, Yuhui Shi, *Fellow, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*—Uncertainty and distributed nature inherently exist in big data environment. Distributed fuzzy neural network (D-FNN) that not only employs fuzzy logics to alleviate the uncertainty problem but also deal with data in a distributed manner, is effective and crucial for big data. Existing D-FNNs always avoided consensus for their antecedent layer due to computational difficulty. Hence such D-FNNs are not really distributed since a single model can not be agreed by multiple agents. This paper proposes a real D-FNN model to handle the uncertainty and distributed challenges in the big data environment. The proposed D-FNN model considers consensus for both the antecedent and consequent layers. A novel consensus learning, which involves a distributed structure learning and a distributed parameter learning, is proposed to handle the D-FNN model. The proposed consensus learning algorithm is built on the well-known alternating direction method of multipliers, which does not exchange local data among agents. The major contribution of this paper is to propose the real D-FNN model for the big data and the novel consensus learning algorithm for this D-FNN model. Simulation results on widespread datasets demonstrate the superiority and effectiveness of the proposed D-FNN model and consensus learning algorithm.**

***Index Terms*—big data, distributed fuzzy neural network, consensus learning, distributed structure learning, distributed parameter learning, alternating direction method of multipliers.**

I. Introduction

In the last decade, emerging technology and breakthroughs have driven the boom of big data in a wide variety of domains, such as social networks, commerce, astronomy, biology, medicine and so on [1]. Big data has been continuously generated at an unprecedented rate. It is expected that digital information will reach 40 trillion gigabytes by 2020, increasing from 0.13 trillion gigabytes in 2005 [2]. How to benefit from the massive amount of big data has become more significant than ever before.

Nowadays, machine learning has proved to be a ubiquitous and indispensable solution for data-driven problems in most sciences [1], [3]. On the other hand, uncertainty inherently exists in these data-driven problems during the process of data collection. Particularly, data collected from sensors, social media, financial and medical records are prone to confronting various uncertainties due to measurement errors, incomplete knowledge and subject difference [4]–[6]. Various forms of uncertainty may degrade the effectiveness and accuracy of intelligent agents, e.g. decision, prediction and control systems. Generally, there are two types of uncertainty that should be taken into consideration when designing intelligent agents — epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty mainly results from imprecision in data and observations or linguistic ambiguity in knowledge while aleatoric uncertainty refers to the inherent variability of subjects or physical systems [5], [6]. Worse, the big data environment is more likely to suffer from uncertainty due to the high dimensional, heterogeneous and unpredictable data characteristics. Therefore, alleviating the effect of uncertainty in the big data environment has become an urgent and inevitable task.

Fortunately, fuzzy inference systems [7], which take advantage of fuzzy logic to measure the value with incomplete or uncertain information, has been demonstrated to be powerful and effective to deal with uncertainties [8]. Therefore, fuzzy neural network (FNN) which exploits both fuzzy inference systems and neural networks can be a good solution for machine learning tasks in the big data environment with uncertainty. Several studies have demonstrated that FNN is capable of dealing with uncertainty benefiting from its fuzzification operation and if-then-rule architecture [6], [9], [10]. The Gaussian membership function used in the fuzzification operation can describe the input data in a specific range of distribution as membership degree, which provides a tolerance toward epistemic uncertainty, such as noise and variations in data [6], [10]. The fuzzy if-then-rule can model dependency between input and output variables. Each fuzzy rule will represent an inference logic for different observation situations associated with various data [6]. Apart from the fuzzy-based method, an extreme learning machine method was used in [11]

Ye Shi and Chin-Teng Lin contributed equally to this work. This work was supported in part by the Australian Research Council (ARC) under discovery grants DP180100670 and DP180100656. We also thank the NSW Defence Innovation Network and NSW State Government of Australia for financial support of this project through grant DINPP2019 S1-03/09. Research was also sponsored in part by the Office of Naval Research Global, US, and was accomplished under Cooperative Agreement Number ONRG-NICOP-N62909-19-1-2058. The work was also supported in part by the National Natural Science Foundation of China under Grant 61976120, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20191445, in part by the Six Talent Peaks Project of Jiangsu Province under Grant XYDXXJS-048, and in part by the Qing Lan Project of Jiangsu Province.

Ye Shi, Chin-Teng Lin, and Yu-Cheng Chang are with CIBCI lab, Centre for Artificial Intelligence, the School of Computer Science, University of Technology Sydney, Broadway, NSW 2007, Australia (email: Ye.Shi-1@uts.edu.au, Chin-Teng.Lin@uts.edu.au and Yu-Cheng.Chang@student.uts.edu.au)

Weiping Ding is with School of Information Science and Technology, Nantong University, Nantong 226019, China (e-mail: dwp9988@163.com)

Yuhui Shi is with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China (e-mail: shiyh@sustc.edu.cn)

Xin Yao is with Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: xiny@sustc.edu.cn)

to address the classification problem on uncertain XML documents. However, its application in the big data environment is not clear.

Another key issue in the big data environment is the distributed nature of the real world data. The massive amount of data may be not available on a single controller, but distributed throughout a network of interconnected agents [12]. The limitations of communication load and storage resources of a single agent also possibly restrict its implementation for large amount of data. Additionally, unlimited data transition between agents may lead to serious security and privacy issues, which have attracted increasing public attention [13]–[15]. Therefore, a centralized algorithm that must be implemented with all the data in a centralized agent is neither practical nor safe especially in the big data environment. A distributed algorithm relied on the local data and limited communication among agents is necessary for the big data environment. In what we refer to as ”consensus learning”, a single model must be agreed by all the agents based on some consensus protocols after the distributed learning process. Thus, consensus learning for distributed FNN (D-FNN) is quite expected in the big data environment.

Distributed machine learning algorithms, such as distributed extreme learning machines [16], [17], distributed support vector machines [18], [19], and distributed deep neural networks [20], [21] have been widely investigated. In [16], a distributed extreme learning machine with kernels based on MapReduce is proposed to realize its parallel computation. Very recently, a decentralized multi-task learning based on extreme Learning machines is proposed in [17], where the alternating direction method of multipliers (ADMM) [22] is employed in an alternating optimization procedure. It is clear that the computational complexity of this method increases dramatically due to the alternating procedure. The ADMM procedure is also employed in [18], [19] to achieve distributed solutions for support vector machines. In [20], [21], ADMM-based algorithms are proposed to train the deep neural networks to avoid gradient-based methods. It should be mentioned that, none of these distributed algorithms can deal with the uncertainty problem in the big data environment.

Recently, several distributed algorithms for FNN were proposed [23], [24]. The authors in [23] proposed a decentralized algorithm for random-weights FNN, where the parameters in the antecedent layer are randomly selected instead of being estimated. An online implementation for the same FNN structure in [23] is further proposed in [24]. There's no doubt that such a random method for parameter identification can result in very large deviations during the learning process. In addition, it suffers from the curse of dimensionality as the number of fuzzy rules increases exponentially with the increase of input space. Moreover, the proposed distributed algorithms can only assure the consensus on the consequent layer instead of both antecedent and consequent layers of the FNN. In other words, such distributed algorithms are not really distributed since a single model can not be agreed by multiple agents. Therefore, these distributed algorithms are neither practical nor efficient in the big data environment.

To overcome the aforementioned issues in distributed algorithm for FNN, this paper proposes a fully D-FNN model, which considers the consensus for its antecedent and consequent layers and hence is really distributed. A novel consensus learning algorithm, which consists of consensus-based structure learning and parameter learning, is proposed for the D-FNN model. Particularly, the consensus-based structure learning for the antecedent layer is built on a distributed clustering method, which can alleviate the dimensionality problem of the random-weights FNN. The consensus-based parameter learning for the consequent layer is realized by a distributed least square algorithm. The consensus-based structure learning and parameter learning are implemented sequentially with the latter employed after the former. Both of them are built on the well-known ADMM, which has been shown as an efficient solution for the consensus-based problems [22], [25]. Although many centralized algorithms for the FNN with clustering method for structure learning were studied in [26]–[28], to the authors' best knowledge, its distributed counterparts were not quite considered in the literature. The contribution of this paper is three-fold:

- This paper proposes a new D-FNN model to address the inherent issues in the big data environment, including the uncertainty and distributed challenges. Note that existing D-FNN models always avoided the consensus for its antecedent layer due to computational difficulty. The proposed real D-FNN exploits distributed structure learning and parameter learning sequentially for the antecedent layer and consequent layer, respectively.
- A novel consensus learning algorithm is proposed to address the proposed D-FNN model. The consensus learning algorithm that consists of consensus-based structure learning and parameter learning is built on the well-known ADMM. It's worth noting that the consensus learning algorithm is very scalable and does not suffer from slow training speed or gradient vanishing problems compared with back-propagation-based methods.
- This paper provides comprehensive simulations of various structures of the D-FNN. Simulation results of the proposed consensus learning algorithm outperform all existing D-FNN algorithms in terms of both generalization accuracy and training speed. Thus the superiority and effectiveness of the consensus learning algorithm are clear.

The rest of the paper is organized as follows. Section II is devoted to modelling of the structure learning and parameter learning for centralized FNN. Section III extends the centralized FNN to D-FNN, for which the consensus learning algorithm is proposed. A comprehensive simulation is conducted in Section IV to confirm the superiority and effectiveness of the proposed D-FNN model and consensus learning algorithm. Conclusions are drawn in Section V.

II. Structure Learning and Parameter Learning for Centralized FNN

Let us briefly recall the structure of existing FNN, which employs the first-order of the Takagi-Sugeno method of fuzzy inference system. Suppose estimating a scar output $y \in \mathbb{R}$

from a D-dimensional input $x = [x_1, x_2, \cdots, x_D]$, then the k-th fuzzy rule of the T-S system is

$$\begin{array}{c}\text{Rule } k\text{: IF } x_1 \text{ is } A_{k1} \text{ and } \cdots \text{ and } x_D \text{ is } A_{kd} \\ \text{Then } y = w_{k0} + \sum_{j=1}^{D} w_{kj} x_j\end{array}$$

where A_{kj} is a Gaussian membership fuzzy set whose membership function is described by,

$$\varphi_{kj}(x_j) = \exp\left[-\left(\frac{x_j - m_{kj}}{\sigma_{kj}}\right)^2\right] \tag{1}$$

where m_{kj} and σ_{kj} are the mean and standard variance of the Gaussian membership function, respectively. Usually, the FNN is constituted by five feed-forward layers, whose structure is provided in Fig.1.

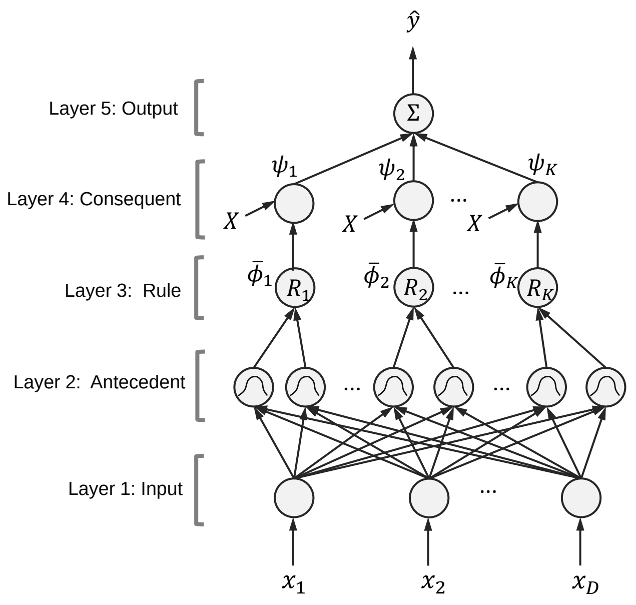


Fig. 1. The structure of FNN

Layer 1 is the input layer, where each node corresponds to one input variable, i.e. x_d, and transmits the scaled input value to the next layer.

Layer 2 is the antecedent layer, where each node corresponds to one fuzzy set and outputs a membership value according to (1).

Layer 3 is the rule layer, where each node represents one fuzzy logic rule and performs antecedent matching of this rule using the following AND operation:

$$\phi_k(x) = \prod_{j=1}^{D} \varphi_{kj}(x_j), \tag{2}$$

which is the firing strength of fuzzy rule k. The obtained firing strength is then normalized by

$$\bar{\phi}_k(x) = \frac{\phi_k(x)}{\sum_{k=1}^{K} \phi_k(x)}. \tag{3}$$

where K is the total number of fuzzy rules.

Layer 4 is called the consequent layer, where each node performs a defuzzification process for each fuzzy rule k using a weighted average operation as follows:

$$\psi_k(x) = \bar{\phi}_k(x)(w_{k0} + \sum_{j=1}^{D} w_{kj} x_j), \tag{4}$$

Layer 5 is the output layer, which calculates the overall output by summing the output of each fuzzy rules in Layer 4 as follows,

$$\hat{y} = \sum_{k=1}^{K} \psi_k(x), \tag{5}$$

Generally, FNN involves identification of structure and parameters. The structure learning is to identify the parameter of Gaussian membership function in the antecedent layer for each fuzzy rule k, i.e. m_{kj} and σ_{kj}, $j \in \{1, \cdots, D\}$; the parameter learning is to identify the output weights $w_{k0}, \cdots, w_{kD}$ in the consequent layer. Both of the structure learning and parameter learning can be addressed by the use of back propagation method [29]. Note that the back-propagation learning process often suffers from slow training speed and gradient vanishing issues. Thus, it is neither practical nor efficient especially in the big data environment, where the data possibly have very large samples and dimensions. To avoid the aforementioned issues, we considered employing the clustering method [26] for the structure learning and the least square algorithm [27] for the parameter learning.

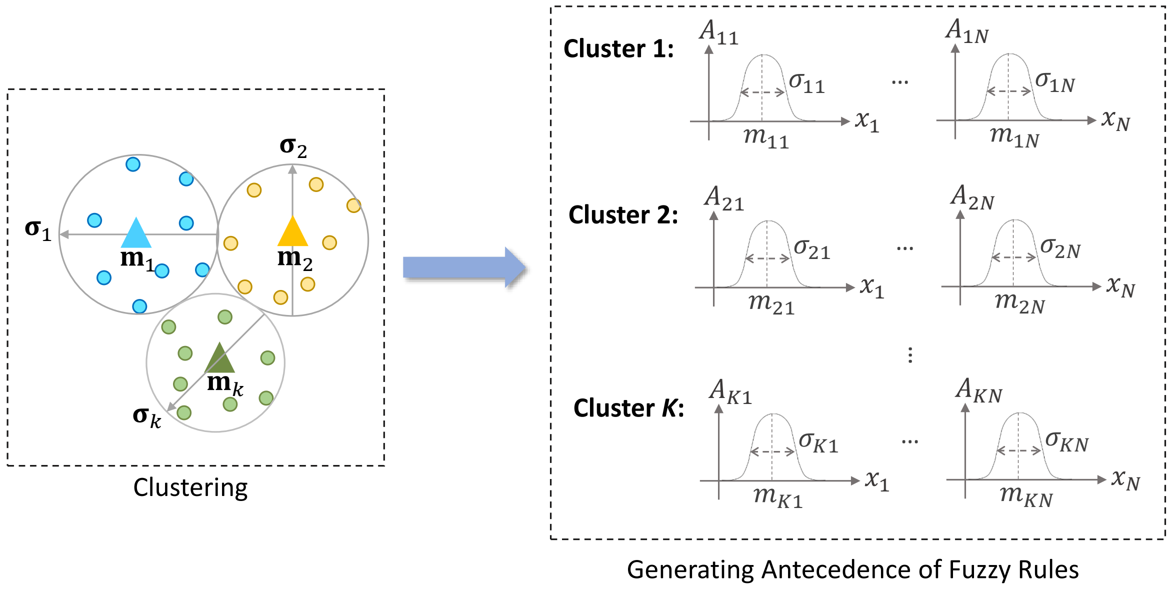
The fuzzy rules are determined by the input-output pairs of the training samples. A basic idea is to group the input-output pairs into clusters and use one rule for each cluster. In this paper, the K-means algorithm [30], which is one of the most popular and efficient clustering algorithm, is employed for structure learning to identify parameters in the antecedent layer of the FNN. As shown in Fig.2, the clustering centers obtained by the K-means algorithm will be the centers of Gaussian membership function in the antecedent layer of the FNN.

Let $\mathcal{D} := \{(X_i, Y_i) | X_i \in \mathbb{R}^D, Y_i \in \mathbb{R},\ i \in \mathcal{N} : \{1, \cdots, N\}\}$ denote the set of training data, x_{ij} denote the j-th component of the i-th observation X_i. The K-means algorithm aims at partitioning the N observations into K sets $\mathcal{C} := \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$. It should be noted that the total number of clusters K, which is assumed to be known a priori, is also the total number of fuzzy rules. Let $\mathcal{K} := \{1, \cdots, K\}$, the clustering problem amounts to identifying the centers of each cluster $k \in \mathcal{K}$ by minimizing the squared error distortion, i.e.,

$$\mathbf{m}_k = \arg\min_{\mathbf{m}_k} \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{C}_k|} ||X_i - \mathbf{m}_k||^2 \tag{6}$$

where $\mathbf{m}_k$ is the center of cluster $\mathcal{C}_k$. The K-means algorithm uses an iterative refinement technique. Starting from an initial set of K centers, i.e. $\{\mathbf{m}_1(0), \cdots, \mathbf{m}_K(0)\}$, the algorithm alternates between an assignment step and an update step as follows.

- Assignment step: Assign each observation X_i to the cluster $\mathcal{C}_k^{(t)}$, whose center is closest to X_i.



σ_1
σ_2
$\mathbf{m}_1$
$\mathbf{m}_2$
$\mathbf{m}_k$
σ_k
Clustering
Cluster 1:
Cluster 2:
Cluster K:
A_{11} σ_{11} m_{11} x_1
A_{1N} σ_{1N} m_{1N} x_N
A_{21} σ_{21} m_{21} x_1
A_{2N} σ_{2N} m_{2N} x_N
A_{K1} σ_{K1} m_{K1} x_1
A_{KN} σ_{KN} m_{KN} x_N
Generating Antecedence of Fuzzy Rules

Fig. 2. The clustering method to identify Gaussian parameters in the antecedent layer of the FNN

- Update step: Update the center of each cluster by

$$\mathbf{m}_k^{(t)} = \frac{1}{|\mathcal{C}_k(t)|}\sum_{X_i\in\mathcal{C}_k(t)} X_i.$$

The K-means algorithm converges if the centers are unchanged during the iteration. Once the centers of each cluster are obtained, the center of each fuzzy set is thus obtained. Accordingly, the standard variance σ_{kj} of each fuzzy set can be obtained by

$$\sigma_{kj} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij}-m_{kj})^2}. \tag{7}$$

By now, the parameters of Gaussian membership function in the antecedent layer are determined. In the following, a closed-form solution to identify the output weights in the consequent layer will be introduced.

Define a hidden matrix $H := [H_1, \cdots, H_K]$, where

$$H_k = \begin{bmatrix} \bar{\phi}_k(X_1) & \bar{\phi}_k(X_1)x_{11} & \cdots & \bar{\phi}_k(X_1)x_{1D} \\ \bar{\phi}_k(X_2) & \bar{\phi}_k(X_2)x_{21} & \cdots & \bar{\phi}_k(X_2)x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\phi}_k(X_N) & \bar{\phi}_k(X_N)x_{N1} & \cdots & \bar{\phi}_k(X_N)x_{ND}, \end{bmatrix} \tag{8}$$

and the output vector $Y := [Y_1, \cdots, Y_N]$. The output weight $\mathbf{w}$ in the consequent layer of FNN can be learned by solving the following optimization problem,

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{2}||Y - H\mathbf{w}||^2 + \frac{\mu}{2}||\mathbf{w}||^2, \tag{9}$$

where $\mu > 0$ is a trade-off factor between the training error and regularization, the weight $\mathbf{w}$ has the following form:

$$\mathbf{w} = [w_{10}, \cdots, w_{1d}, \cdots, w_{K0}, \cdots, w_{KD}]^T, \tag{10}$$

It should be noted that selecting the value of μ appropriately can make the solution much more stable and have better generalization performance [31]. Since (9) is a standard least square optimization problem, its closed form solution can be easily obtained by:

$$\mathbf{w} = (H^TH + \mu I)^{-1}H^TY, \tag{11}$$

where I is the identity matrix with dimension of $K(D+1)$.

III. Consensus Learning for D-FNN

In the big data environment, large-scale data may exist in different locations and machines, referred as multiple agents in this paper. Note that the centralized FNN must implement all data in a single agent. It may not be possible to perform centralized FNN in the big data environment due to several reasons. First, the communication load and storage resources of a single agent may limit its implementation of large-scale data. In addition, transiting the data between multiple agents may result in serious data security and privacy issues, which have attracted increasing public attention recently. Moreover, the whole system will fail if the centralized agent loses or disconnects due to contingency. Therefore, there is a great demand for D-FNN, where the global training process can be performed in each individual agent with limited information exchange. Moreover, the D-FNN is more flexible and robust against the contingencies compared with the centralized FNN.

In this section, the centralized FNN is extended to its distributed version D-FNN to deal with the big data. A novel consensus learning is proposed for the D-FNN, which integrates multiple FNNs corresponding to multiple agents in the big data environment and agrees on a single FNN based on consensus protocols. The consensus learning algorithm consists of consensus-based structure learning and parameter learning. Both of them are built on ADMM, which is widely employed in consensus-based distributed problems [22].

Generally, the distributed algorithms can be classified into two types based on network topology. The first one is implemented in a star network, where a fusion center is required to communicate with all agents. The second one does not need such fusion center. The agents are only allowed to communicate with their neighbouring agents based on the underlying network topology. Although the former has better convergence performance, the latter is more preferred in the big data environment due to the practicability and security issues. The fusion center may not exist in the big data environment. In addition, the requirement that all agents must communicate with the fusion center instead of with their neighbouring ones, will increase the potential risk of data leakage. Therefore, the D-FNN proposed in this paper will employ the second type of distributed algorithm.

We consider a network with L nodes connected with E edges and each node is assumed as an agent. This network can be described as an undirected graph $\mathcal{G} = \{\mathcal{L}, \xi\}$, where $\mathcal{L}$ and ξ are the sets of vertexes and edges, respectively. A simple network consisting of five agents and six edges is shown in Fig.3 for illustration purpose. For the dataset $\mathcal{D} := \{(X_i, Y_i) | i \in \mathcal{N}\}$, Let $\{\mathcal{D}^1, \cdots, \mathcal{D}^L\}$ be a decomposition of the entire dataset $\mathcal{D}$. For ease of representation, each subset $\mathcal{D}^l$ is assumed to be the data located on each agent $l \in \mathcal{L}$. The set of neighbouring agents of agent l is defined as $\mathcal{N}_l$. Similarly, we introduce subset $\mathcal{C}^l$ to represent the observation subset of agent l. In each subset $\mathcal{C}^l$, let $\mathcal{C}^l_k$ denote the subset of cluster k in agent l such that $\bigcup_{k=1}^{K} \mathcal{C}^l_k = \mathcal{C}^l$. Based on

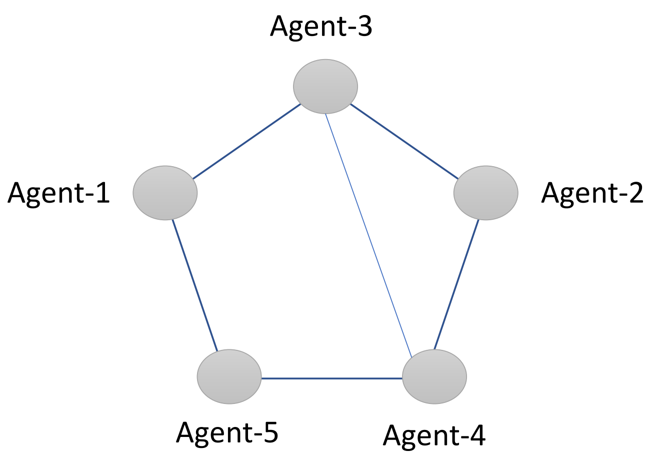


Fig. 3. A simple network with five agents

the consensus strategy, the structure learning problem for the D-FNN can be formulated as follows,

$$\min_{\mathbf{m}^l_k} \frac{1}{2} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{X^l_i \in \mathcal{C}^l_k} ||X^l_i - \mathbf{m}^l_k||^2 \tag{12a}$$

$$\text{s.t.} \quad \mathbf{m}^l_k = \mathbf{r}_k, \ l \in \mathcal{L}, \ k \in \mathcal{K}, \tag{12b}$$

where $\mathbf{m}^l_k$ is a local variable, representing the center of fuzzy set k of agent l, $\mathbf{r}_k$ is a global variable to integrate all local centers and $|\mathcal{C}^l_k|$ represents the cardinality operation for a set. The constraint (12b) employs the consensus strategy, which assures all the local centers coincide at one global vector of center. It is worth noting that the local variable $\mathbf{m}^l_k$ can be computed in parallel for each agent l.

Once the global center $\mathbf{r}_k$ of each fuzzy rule k is determined, the global standard variance can be easily calculated by:

$$\bar{\sigma}_{kj} = \sqrt{\frac{1}{N} \sum_{l=1}^{L} |\mathcal{C}^l| (\sigma^l_{kj})^2} \tag{13}$$

where $\bar{\sigma}_{kj}$ is the j-th component of the global standard variance of the k-th fuzzy rule, σ^l_{kj} is the corresponding standard variance of local agent l, $|\mathcal{C}^l|$ represents the cardinality of subset $\mathcal{C}^l$. It should be noted that the standard variance σ^l_{kj} of rule k dimension j is calculated in a component-wise manner, i.e.

$$\sigma^l_{kj} = \sqrt{\frac{1}{|\mathcal{C}^l_k|} \sum_{X^l_i \in \mathcal{C}^l_k} (X^l_{ij} - \mathbf{m}^l_{kj})^2} \tag{14}$$

where X^l_{ij} and $\mathbf{m}^l_{kj}$ are the j-th component of X^l_i and $\mathbf{m}^l_k$, respectively.

The above distributed clustering by consensus learning for antecedent layer identification of the D-FNN is shown in Fig.4

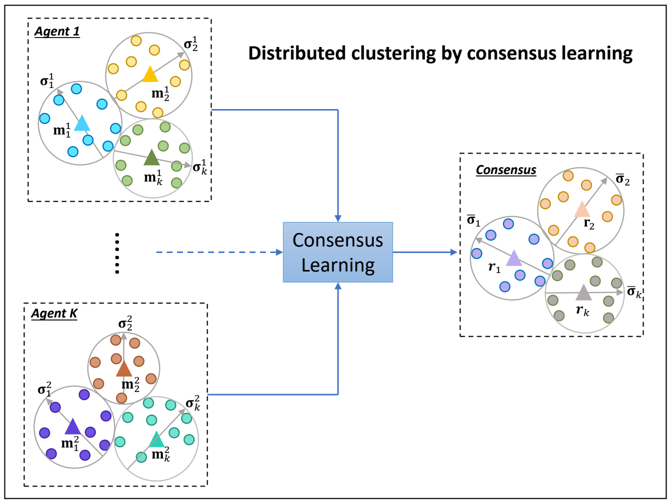


Fig. 4. The procedure of distributed clustering by consensus learning for antecedent layer identification of the D-FNN

Similarly, the parameter learning problem for the D-FNN based on the consensus strategy is formulated as follows,

$$\min_{\mathbf{w}^l} \frac{1}{2} \sum_{l=1}^{L} (||Y^l - H\mathbf{w}^l||^2 + \mu ||\mathbf{w}^l||^2), \tag{15a}$$

$$\text{s.t.} \quad \mathbf{w}^l = \mathbf{z}, \ l \in \mathcal{L}, \ q \in \mathcal{N}_l, \tag{15b}$$

where $\mathbf{w}^l$ is a local variable, representing the output weight of agent l, $\mathbf{z}$ is a common global weight to integrate all local weights.

The procedure of the proposed consensus learning algorithm, which consists of distributed structure learning for the antecedent layer and distributed parameter learning for the consequent layer, is described in Fig.5.

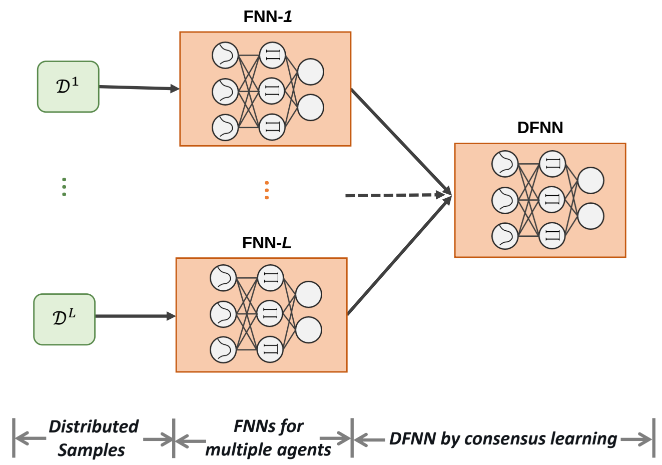


Fig. 5. The procedure of consensus learning for the D-FNN

A. Overview of ADMM

In this section, we first introduce the preliminary knowledge of ADMM [22], which will be used to solve the proposed optimization problem (12) and (15) for the consensus learning. The standard form of ADMM solves the following problem:

$$\min f(\mathbf{x}) + g(\mathbf{y}) \tag{16}$$

$$\text{s.t.} \quad A\mathbf{x} + B\mathbf{y} = c, \ \ \mathbf{x} \in \mathcal{C}_f, \ \ \mathbf{y} \in \mathcal{C}_g \tag{17}$$

Then the following augmented Lagrangian is constructed,

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \quad & f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^T(A\mathbf{x} + B\mathbf{y} - c) \\ & + \frac{\rho}{2}||A\mathbf{x} + B\mathbf{y} - c||^2 \end{aligned}\tag{18}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier and ρ is a positive constant to trade-off the convergence rate and numerical accuracy. The ADMM solves (18) iteratively via an alternating procedure, which starts from arbitrary initial points $\mathbf{y}(0)$ and $\boldsymbol{\lambda}(0)$ and then updates the variable $\mathbf{x}$ and $\mathbf{y}$ and multipliers $\boldsymbol{\lambda}$ sequentially by the following procedure:

$$\mathbf{x}(t+1) = \arg \min_{\mathbf{x} \in \mathcal{C}_f} L(\mathbf{x}, \mathbf{y}(t), \boldsymbol{\lambda}(t)), \tag{19}$$

$$\mathbf{y}(t+1) = \arg \min_{\mathbf{y} \in \mathcal{C}_g} L(\mathbf{x}(t+1), \mathbf{y}, \boldsymbol{\lambda}(t)), \tag{20}$$

$$\boldsymbol{\lambda}(t+1) = \boldsymbol{\lambda}(t) + \rho(A\mathbf{x}(t+1) + B\mathbf{y}(t+1) - c), \tag{21}$$

until convergence. The convergence behavior at iteration t can be inspected by analyzing the primal residual and dual residual with given tolerances as follows,

$$||A\mathbf{x}(t) + B\mathbf{y}(t) - c||^2 \leq \epsilon_1, \tag{22}$$

$$||\boldsymbol{\lambda}(t) - \boldsymbol{\lambda}(t-1)||^2 \leq \epsilon_2, \tag{23}$$

where ϵ_1 and ϵ_2 are the values of convergence tolerance of the ADMM procedure.

Under the convexity condition of function $f(\cdot)$ and $g(\cdot)$ and domain set $\mathcal{C}_f$ and $\mathcal{C}_g$, ADMM is known to converge to a unique stable point [22]. A recent study also proves the convergence of ADMM for a variety of nonconvex and possibly nonsmooth functions given some sufficient conditions [25].

B. Consensus-based distributed structure learning

The optimization problem (12) is nonconvex, which can not be efficiently solved by any exhaustive search methods. Meanwhile, the centralized clustering methods such as the K-means algorithm still suffer from the issues of communication load, privacy concerns and contingencies in the big data environment. Thus, a distributed clustering method is crucial to address the optimization problem (12). In this paper, a distributed K-means algorithm is proposed to address the structure learning optimization problem (12). This method is motivated by [32], where a distributed clustering scheme is developed for wireless sensor networks.

First, we construct the following augmented Lagrangian for (12):

$$\begin{aligned}\mathcal{L}_s(\mathbf{m}, \mathbf{r}, \boldsymbol{\lambda}_s) = \quad & \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{X_i^l \in \mathcal{C}_k^l} ||X_i^l - \mathbf{m}_k^l||^2 \\ & + \sum_{l=1}^{L}\sum_{k=1}^{K} \boldsymbol{\lambda}_{s,kl}^T(\mathbf{m}_k^l - \mathbf{r}_k) \\ & + \frac{1}{2}\rho_s \sum_{l=1}^{L}\sum_{k=1}^{K} ||\mathbf{m}_k^l - \mathbf{r}_k||^2 \end{aligned}\tag{24}$$

where $\boldsymbol{\lambda}_{s,kl}$ is the Lagrange multiplier and $\rho_s > 0$ is a penalty parameter. Starting from the initial points $\mathbf{r}(0)$ and $\boldsymbol{\lambda}_s(0)$, the variables at each iteration t are updated iteratively by the following procedures based on the ADMM:

$$\mathbf{m}^l(t+1) = \arg \min_{\mathbf{m}} \mathcal{L}(\mathbf{m}^l, \mathbf{r}(t), \boldsymbol{\lambda}_s(t)), \tag{25}$$

$$\mathbf{r}(t+1) = \arg \min_{\mathbf{r}} \mathcal{L}(\mathbf{m}^l(t+1), \mathbf{r}, \boldsymbol{\lambda}_s(t)), \tag{26}$$

$$\boldsymbol{\lambda}_{s,kl}(t+1) = \boldsymbol{\lambda}_{s,kl}(t) + \rho_s(\mathbf{m}_k^l(t+1) - \mathbf{r}_k(t+1)). \tag{27}$$

It should be noted that (25) can be solved in parallel for each agent l. Similarly, we can also employ the assignment step and update step in the K-means algorithm to update the cluster centers $\mathbf{m}^l(t+1)$ for each agent l. On the other hand, since the optimization problem (26) is linear on $\mathbf{r}$, the closed-form solution can be easily solved by setting the partial derivatives with respect to $\mathbf{r}$ to zero. The closed-form solution of (26) is given directly as follows,

$$\mathbf{r}_k(t+1) = \frac{1}{\rho_s}\bar{\boldsymbol{\lambda}}_{s,kl}(t) + \bar{\mathbf{m}}_k^l(t+1), \tag{28}$$

where

$$\bar{\mathbf{m}}_k^l(t+1) = \frac{1}{L}\sum_{l=1}^{L} \mathbf{m}_k^l(t+1), \tag{29}$$

$$\bar{\boldsymbol{\lambda}}_{s,kl}(t) = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\lambda}_{s,kl}(t). \tag{30}$$

From (28), we can see updating $\mathbf{r}_k(t+1)$ requires the communication between all the agents. However, it may not possible in the big data environment. Therefore, a more practical updating procedure is more preferred here. Similarly with the idea in [33], this paper employs the well-known distributed average consensus (DAC) strategy [34] to update $\mathbf{r}_k(t+1)$. DAC is an iterative strategy to compute the global average requiring data only exchanged in a local neighborhood. At iteration t, the local DAC update is given by:

$$\alpha^l(t+1) = \sum_{q\in\mathcal{L}_l} W_{lq}\alpha^l(t), \tag{31}$$

where α^l is the local variable corresponding to each agent l, W_{lq} is a weighted connectivity matrix. Suppose d_l is the number of neighboring agents of agent l and $d = \max_{l\in\mathcal{L}} d_l$. According to [34], given the following matrix-degree weight,

$$W_{lq} = \begin{cases} \frac{1}{d+1}, & \text{if} \quad q \in \mathcal{L}_l; \\ 1 - \frac{d_l}{d+1}, & \text{if} \quad q = l; \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

the following procedure converges to the global average:

$$\lim_{t\to+\infty} \alpha^l(t) = \frac{1}{L}\sum_{l=1}^{L}\alpha^l(0), \ \forall l \in \mathcal{L}. \tag{33}$$

Therefor, $\bar{\mathbf{m}}_k^l(t+1)$ and $\bar{\boldsymbol{\lambda}}_{s,kl}(t)$ can be easily obtained by using the DAC iteration (31) with the matrix-degree weight (32). It is worth noting that it is not necessary to match the cluster ordering of each agent before the consensus procedure. The consensus algorithm will still converge with random initial cluster ordering.

Based on the analysis above, we can summarize the algorithm for distributed structure learning in Algorithm 1. The convergence behavior of Algorithm 1 can be inspected by checking the norms of the following two residuals:

$$||\mathbf{m}_k^l(t) - \mathbf{m}_k^q(t)||^2 \le \epsilon_1, \tag{34}$$

$$||\boldsymbol{\lambda}_{s,k}^l(t) - \boldsymbol{\lambda}_{s,k}^l(t-1)||^2 \le \epsilon_2. \tag{35}$$

Algorithm 1 ADMM-based distributed structure learning (12)

```
Initialization: Set $t = 0$ and the Lagrange multipliers $\boldsymbol{\lambda}_{s,kl}(t) = \mathbf{0}$. Initialize the cluster centers $\mathbf{m}_k^l(t)$ by using K-means algorithm for each agent $l$.
for $t = 0, 1, 2, \cdots$, do
    Update the local variables $\mathbf{m}^l(t+1)$:
    Assignment step: Each agent $l$ assigns each $X_i^l$ to the cluster $\mathcal{C}_k^l(t)$, whose center $\mathbf{r}_k^l(t)$ is closest to $X_i^l$.
    Update step: Each agent $l$ updates the center of each cluster $\mathcal{C}_k^l(t)$ by
        $\mathbf{m}_k^l(t+1) = \frac{1}{|\mathcal{C}_k^l(t)|}\sum_{X_i^l\in\mathcal{C}_k^l(t)} X_i^l$    (36)
    Update the global variables $\mathbf{r}(t+1)$ by (28) and broadcast it to each agent $l$.
    Update the dual variables $\boldsymbol{\lambda}_s(t+1)$ by (27) and broadcast it to each agent $l$
end for
```

C. Consensus-based distributed parameter learning

Similarly, we solve the optimization problem (15) in a distributed manner by the use of ADMM. The augmented Lagrangian for (15) is as follow,

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \mathbf{z}, \boldsymbol{\lambda}_p) = \quad & \frac{1}{2}\sum_{l=1}^{L}||Y^l - H^l\mathbf{w}^l||^2 + \frac{\mu}{2}||\mathbf{z}||^2 \\ & + \sum_{l=1}^{L}\boldsymbol{\lambda}_{p,l}^T(\mathbf{w}^l - \mathbf{z}) \\ & + \frac{1}{2}\rho_p\sum_{l=1}^{L}||\mathbf{w}^l - \mathbf{z}||^2 \end{aligned} \tag{37}$$

where $\boldsymbol{\lambda}_{p,l}$ is the Lagrange multiplier and $\rho_p > 0$ is a penalty parameter. The ADMM-based procedures for distributed parameter learning are as follows,

$$\mathbf{w}^l(t+1) = \arg\min_{\mathbf{m}} \mathcal{L}(\mathbf{w}^l, \mathbf{z}(t), \boldsymbol{\lambda}_p(t)), \tag{38}$$

$$\mathbf{z}(t+1) = \arg\min_{\mathbf{r}} \mathcal{L}(\mathbf{w}^l(t+1), \mathbf{z}, \boldsymbol{\lambda}_p(t)), \tag{39}$$

$$\boldsymbol{\lambda}_p(t+1) = \boldsymbol{\lambda}_p(t) + \rho_s(\mathbf{w}^l(t+1) - \mathbf{z}(t+1)). \tag{40}$$

Since (38) is a standard least square problem, its closed form solution can be easily obtained by:

$$\mathbf{w}^l(t+1) = ((H^l)^TH^l + \mu I)^{-1}((H^l)^TY^l - \boldsymbol{\lambda}_p(t) + \rho_p\mathbf{z}(t)), \tag{41}$$

where I is the identity matrix with dimension of $K(D+1)$. The closed-form solution of (39) can be obtained by:

$$\mathbf{z}(t+1) = \frac{\bar{\boldsymbol{\lambda}}_{p,l}(t) + \rho_p\bar{\mathbf{w}}^l(t+1)}{\mu/L + \rho_p}, \tag{42}$$

where

$$\bar{\mathbf{w}}^l(t+1) = \frac{1}{L}\sum_{l=1}^{L}\mathbf{w}^l(t+1),$$

$$\bar{\boldsymbol{\lambda}}_{p,l}(t) = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\lambda}_{p,l}(t).$$

Similarly with the procedure in the distributed structure learning, we also employ the DAC strategy in order to avoid communications among all agents. The $\bar{\mathbf{w}}^l(t+1)$ and $\bar{\boldsymbol{\lambda}}_{p,l}(t)$ are obtained using the DAC iteration (31) with the matrix-degree weight (32).

Similarly, we summarize the algorithm for distributed parameter learning in Algorithm 2. The convergence behavior of the Algorithm 2 can be inspected by checking the norms of the following two residuals:

$$||\mathbf{w}^l(t) - \mathbf{z}(t)||^2 \le \epsilon_1, \tag{43}$$

$$||\boldsymbol{\lambda}_p(t) - \boldsymbol{\lambda}_p(t-1)||^2 \le \epsilon_2. \tag{44}$$

Remark: One may notice that, during the consensus procedure of the D-FNN, the cluster number of agents are required to be the same. However, this may be not always true in some specific scenarios. To make our algorithms more general, the scenario that some clusters are missing in some agents should be considered. Here we provide a simple strategy to address the cluster mismatching problem. Suppose the cluster number

Algorithm 2 ADMM-based distributed parameter learning (15)

```
Initialization: Set $t = 0$ and initialize global weight $\mathbf{z}(t)$ and Lagrange multipliers $\boldsymbol{\lambda}_p(t)$ for each agent.
for $t = 0, 1, 2, \cdots$, do
    Update the local variables $\mathbf{w}^l(t+1)$ by (41)
    Update the global variables $\mathbf{r}(t+1)$ by (42) and broadcast it to each agent $l$.
    Update the dual variables $\boldsymbol{\lambda}_p(t+1)$ by (40) and broadcast it to each agent $l$
end for
```

of each agent are $K_1, K_2 \cdots, K_L$, respectively. Without loss of generality, we set $K_1 \leq K_2 \leq K_L$. As shown in Fig. 6, if a cluster is missing, the corresponding position is filled with zero-vector $\mathbf{0}$. After the filling procedure, the cluster number of each agent becomes the same. Then the Algorithm 1 can be used directly for the distributed structure learning of the D-FNN. If the data distribution of each agent are totally different, then the knowledge of domain adaptation [35] should be considered. However, this goes beyond the scope of this paper.

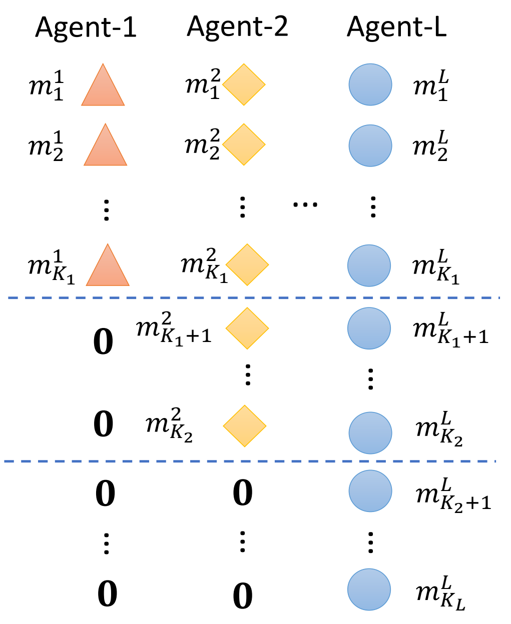


Fig. 6. The cluster centers of each agent in the mismatching scenario

IV. Simulations

In this section, the performance of the proposed consensus learning algorithm for D-FNN in the big data environment is evaluated by numerical simulations on several widespread datasets, which are available on UCI Machine Learning Repository [1] or Kaggle Datasets [2]. The simulation results based on the proposed consensus learning algorithm are compared with several state-of-the-art FNN algorithms.

[1]http://archive.ics.uci.edu/ml

[2]https://www.kaggle.com/datasets

TABLE I
Numerical Information of the Tested Datasets

Dataset	Samples	Dimensions	Desired Output
CCPP [36]	9,568	4	Electrical energy
KC-house [37]	21,613	15	House price
CASP [38]	45,730	9	Deviation
Motor [39]	998,070	7	Motor temperature

The datasets are selected to represent various domains in the big data environment though not all of them have very large-scale samples and dimensions. Here we briefly summarize the input and output information for these datasets in Table I. In all these cases, input variables are normalized between $[-1, 1]$ before the experiments.

To stimulate the distributed nature of the big data, a network of agents is randomly generated using a random topology model [33] with each agent's connectivity probability as 25%. Accordingly, each dataset is randomly decomposed for each agent. To evaluate the accuracy of all the models, we perform a 10-fold cross-validation for each dataset. In each fold, the following FNN algorithms are compared:

- Random-weight-FNN (R-FNN): This is the distributed FNN algorithm reported in [23], which randomly generates the Gaussian parameters in the antecedent layer and then employs the least square algorithm to identify the parameters in the consequent layer. As we mentioned in the Introduction, such a random method for parameter identification could result in very large deviations during the learning process. In addition, it suffered from the curse of dimensionality as the number of fuzzy rules increases exponentially with the increase of input space. Moreover, the distributed algorithm assured consensus only for the consequent layer instead of both the antecedent and consequent layers. Thus it is not really distributed and thus not practical in the big data environment.
- Centralized K-means-FNN (C-FNN): This is the centralized FNN algorithm provided in this paper. It employs the K-means algorithm to identify the Gaussian parameters in the antecedent layer and least square algorithm to identify the parameters in the consequent layer. Specifically, C-FNN solves the optimization problem (6) by the centralized K-means algorithm to obtain the parameters $\mathbf{m}_k$ and σ_{kj} and then fix them to solve the optimization problem (9) by the close-form solution (11). Note that by the use of the K-means algorithm for identifying parameters in the antecedent layer, it does not suffer from the curse of dimensionality as R-FNN does.
- Half-consensus learning D-FNN (H-FNN): This is the distributed algorithm for the same structure of C-FNN but employing the consensus protocol only for its consequent layer. Thus, the term "half-consensus learning" is used for the H-FNN, Note that by H-FNN, agents can not agree on a single FNN model after the learning procedure. Specifically, H-FNN solves the optimization problem

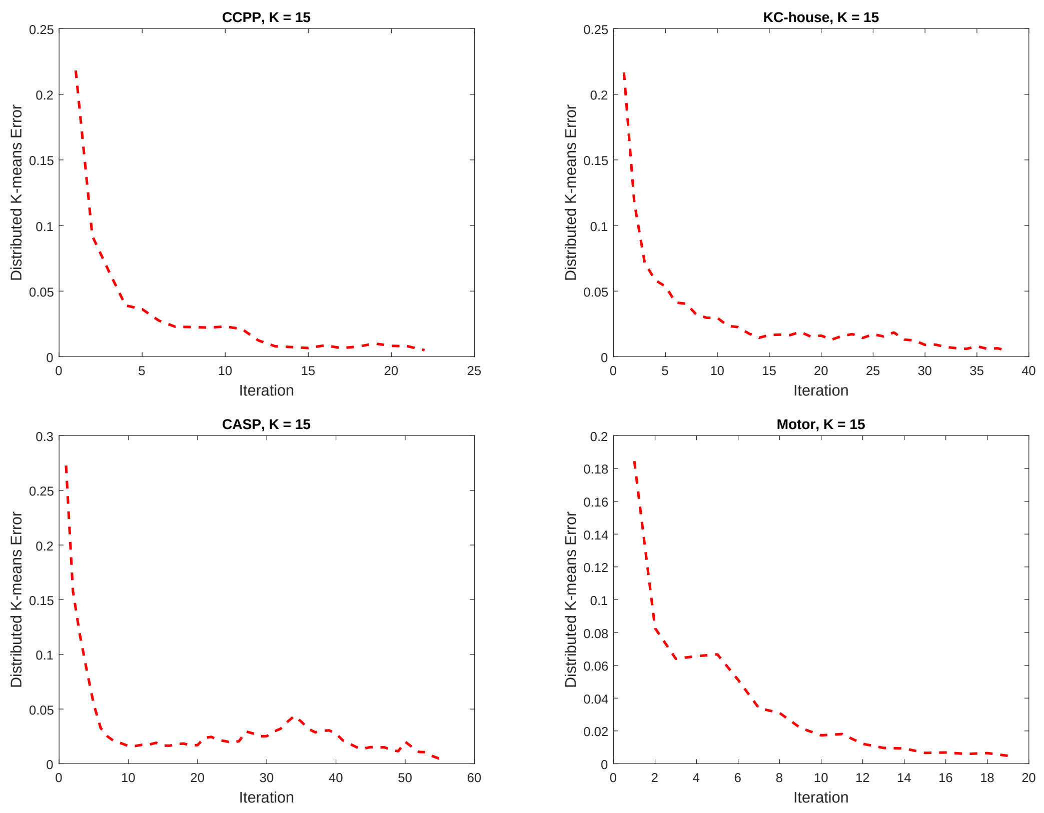


Fig. 7. Convergence bahavior of the distributed K-means algorithm for various datasets

(6) by the centralized K-means algorithm to obtain the parameters $\mathbf{m}_k$ and σ_{kj}. Then H-FNN broadcasts them to each agent and solves optimization problem (15) in a distributed manner by Algorithm 2.

- Consensus learning D-FNN (D-FNN): This is the consensus learning algorithm proposed by this paper, which employs the consensus protocol for both the antecedent and consequent layers. It's worth noting that this algorithm is the really distributed and practical one in the big data environment. The agreement among various agents on a single FNN model can be obtained after the consensus learning procedure. Particularly, D-FNN employs Algorithm 1 and Algorithm 2 sequentially to respectively solve the optimization problem (12) and optimization problem (15) in a fully distributed manner.

In this paper, all simulations are implemented using Matlab R2019b on a laptop with Intel i7 @ 4.0 GHz processor and 16 GB of memory. The convergence criteria of the ADMM procedure in Algorithm 1 and Algorithm 2 are set as $\epsilon_1 = \epsilon_2 = 10^{-3}$. The normalized root mean square error (NRMSE) defined by

$$\text{NRMSE} = \sqrt{\frac{1}{N\hat{\sigma}_Y^2}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}, \tag{45}$$

is used to evaluate the performance of the models.

The convergence behavior of distributed K-means algorithm and distributed parameter learning are provide by Fig.7 and Fig.8, respectively. It can be seen, both of the two algorithms converge quite fast for these datasets.

Table II summarizes the simulation results for the tested datasets by implementing R-FNN, C-FNN, H-FNN and D-FNN, respectively. The first column of Table II is the dataset name, the second column gives the total number of agents, the third column provides the trade-off factor in (15a). The fourth, fifth and sixth column of Table II present the simulation results of R-FNN including total number of rules, obtained NRMSE value and training time, respectively. The seventh column of Table II gives the K value, which is also the total number of clustering and fuzzy rules for implementing the C-FNN, H-FNN and D-FNN. Their simulation results are provided in the remaining columns. It should be noted that the NRMSE value obtained by the C-FNN is a lower bound of the one obtained by H-FNN. It can be seen from Table II, R-FNN can not handle the datasets KC-house and Motor due to the large-scale rule numbers and samples. The NRMSE value of CCPP obtained by C-FNN is much worse than the ones obtained by C-FNN, H-FNN and D-FNN. As for CASP, the NRMSE value obtained by R-FNN is a bit better than other three algorithms since R-FNN uses much more rules (512 vs 15). Clearly, R-FNN is not scalable and can not be used in the big data environment. We also test the D-FNN by using various values of K for CASP. Figure 9 provides the NRMSE of CASP by setting various K for the D-FNN. Generally, larger value of K will lead to smaller value of NRMSE. However, we still suggest to select a moderate value of K. Surprisingly, the NRMSE value of Motor by D-FNN is much better than the

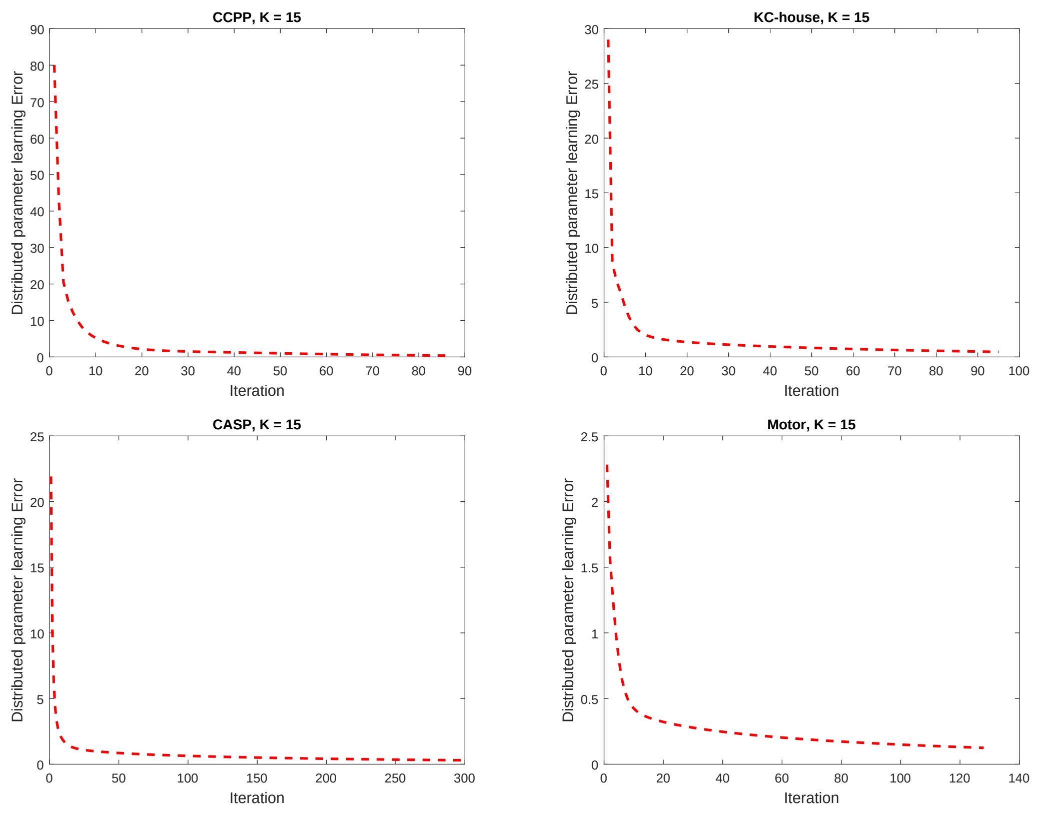


Fig. 8. Convergence bahavior of the distributed parameter learning algorithm for various datasets

TABLE II
SIMULATION RESULTS FOR THE DATASETS

Dataset	L	u	R-FNN			K	C-FNN		H-FNN		D-FNN	
			Rules	NRMSE	Time(s)		NRMSE	Time (s)	NRMSE	Time (s)	NRMSE	Time (s)
CCPP	5	0.01	16	4.178	0.23	15	0.2313	0.15	0.2320	0.26	0.2331	0.61
KC-house	5	0.01	32768	-	-	15	0.5299	0.45	0.5304	0.51	0.5251	2.31
CASP	5	0.001	512	0.7682	457.4	15	0.7745	0.94	0.7748	0.83	0.7827	1.43
Motor	25	0.001	128	-	-	15	0.6230	13.31	0.6248	6.0	0.6079	9.82

other algorithms since the clustering results (fuzzy rules) of D-FNN is different from the others'. For such a large-scale dataset, the distributed clustering results can outperform the centralized clustering results. This phenomenon also verifies the effectiveness of the proposed consensus learning algorithm. In addition, we would like to re-emphasize that the superiority of D-FNN compared with other three methods are as follows: 1) The D-FNN is able to handle data in multiple agents. This capability becomes more significant in big data environment as the big data may exist in different locations and machines. 2) The D-FNN can alleviate the burden of communication load and storage resources in a single agent. 3) The D-FNN can preserve the users' data privacy by limiting the data transiting between multiple agents.

To verify the proposed distributed K-means method can work well in the case that each agent has different cluster numbers, we implemented it on the CASP dataset with different sample number and different cluster number of each agent, specifically, the cluster number are respectively $K_1 = 11, K_2 = 12, K_3 = 13, K_4 = 14, K_5 = 15$. The distributed K-means method converge within 50 iterations and achieves the NRMSE value 0.7887, which is consistent with the NRMSE value 0.7827 obtained by the distributed K-means method but with the same cluster number $K = 15$ of each agent.

V. CONCLUSIONS

Emerging technology and breakthroughs have driven the boom of big data in various domains. This paper has proposed a D-FNN model to deal with the inherent issues of the big data environment including the uncertainty and distributed challenge. The proposed D-FNN employed a sentential manner to exploit distributed structure learning and parameter learning based on distributed optimization methods. It's worth noting that the D-FNN is very scalable and does not suffer from slow training speed and gradient vanishing problems compared with back-propagation-based methods. The consensus learning algorithm has been proposed for the D-FNN in the big data environment. The consensus learning algorithm, which consists of consensus-based distributed structure learning and parameter learning is built on the well-known ADMM. Simulation results have verified the superiority and effectiveness

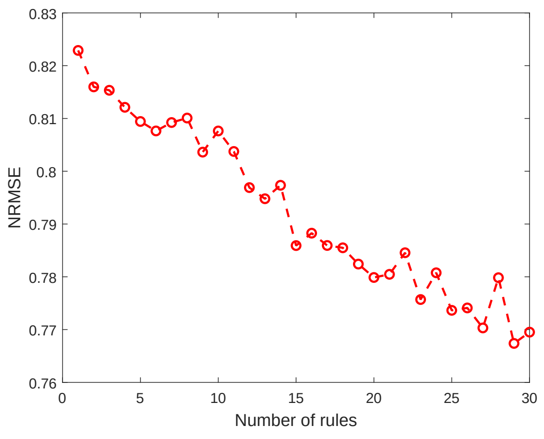


Fig. 9. NRMSE of CASP by setting various K

of the proposed consensus learning algorithm for D-FNN. The proposed consensus learning algorithm can be easily generalized to realize various functions and tasks in the area of machine learning. A new D-FNN model with hierarchy structure based on the proposed consensus learning algorithm is under consideration for future works.

References

[1] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[2] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.

[3] Z. Obermeyer and E. J. Emanuel, "Predicting the future—big data, machine learning, and clinical medicine," *The New England journal of medicine*, vol. 375, no. 13, pp. 1216–1219, 2016.

[4] R. Ak, V. Vitelli, and E. Zio, "An interval-valued neural network approach for uncertainty quantification in short-term wind speed prediction," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2787–2800, 2015.

[5] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: survey, opportunities, and challenges," *Journal of Big Data*, vol. 6, no. 1, pp. 1–16, 2019.

[6] I. Couso, C. Borgelt, E. Hullermeier, and R. Kruse, "Fuzzy sets in data analysis: From statistical foundations to machine learning," *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 31–44, 2019.

[7] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[8] C. El Hatri and J. Boumhidi, "Fuzzy deep learning based urban traffic incident detection," *Cognitive Systems Research*, vol. 50, pp. 206–213, 2018.

[9] G. Fu and Z. Kapelan, "Fuzzy probabilistic design of water distribution networks," *Water Resources Research*, vol. 47, no. 5, pp. 1–12, 2011.

[10] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[11] X. Zhao, X. Bi, G. Wang, Z. Zhang, and H. Yang, "Uncertain XML documents classification using extreme learning machine," *Neurocomputing*, vol. 174, pp. 375–382, 2016.

[12] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2–12, 2014.

[13] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *ACM Sigmod Record*, vol. 33, no. 1, pp. 50–57, 2004.

[14] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 1, pp. 647–651, IEEE, 2012.

[15] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless communications*, vol. 17, no. 1, pp. 51–58, 2010.

[16] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on mapreduce," *Neurocomputing*, vol. 149, pp. 456–463, 2015.

[17] Y. Ye, M. Xiao, and M. Skoglund, "Decentralized multi-task learning based on extreme learning machines," *arXiv preprint arXiv:1904.11366*, pp. 1–11, 2019.

[18] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[19] Y.-Q. Bai, Y.-J. Shen, and K.-J. Shen, "Consensus proximal support vector machine for classification problems with sparse solutions," *Journal of the Operations Research Society of China*, vol. 2, no. 1, pp. 57–74, 2014.

[20] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable ADMM approach," in *International conference on machine learning*, pp. 2722–2731, 2016.

[21] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3466–3473, 2018.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[23] R. Fierimonte, M. Barbato, A. Rosato, and M. Panella, "Distributed learning of random weights fuzzy neural networks," in *2016 IEEE International Conference on Fuzzy Systems*, pp. 2287–2294, IEEE, 2016.

[24] R. Fierimonte, R. Altilio, and M. Panella, "Distributed on-line learning for random-weight fuzzy neural networks," in *2017 IEEE International Conference on Fuzzy Systems*, pp. 1–6, IEEE, 2017.

[25] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[26] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent & fuzzy systems*, vol. 2, no. 3, pp. 267–278, 1994.

[27] J. de Jesús Rubio, "SOFMLS: online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[28] J. A. M. Hernández, F. G. Castañeda, and J. A. M. Cadenas, "An evolving fuzzy neural network based on the mapping of similarities," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1379–1396, 2009.

[29] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.

[30] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[31] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[32] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.

[33] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65–74, 2016.

[34] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[35] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.

[36] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.

[37] "Predict house price in king county using regression." https://www.kaggle.com/harlfoxem/housesalesprediction. Accessed: 2019-08-18.

[38] "Physicochemical properties of protein tertiary structure data set." https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure. Accessed: 2019-07-20.

[39] W. Kirchgässner, O. Wallscheid, and J. Böcker, “Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet synchronous motors,” in *2019 IEEE International Electric Machines & Drives Conference (IEMDC)*, pp. 1439–1446, IEEE, 2019.