Distributed Semi-supervised Fuzzy Regression with Interpolation Consistency Regularization

Ye Shi, *Member, IEEE*, Leijie Zhang, Zehong Cao, *Member, IEEE*, and M. Tanveer, *Senior Member, IEEE*, Chin-Teng Lin *Fellow, IEEE*

Abstract—Recently, distributed semi-supervised learning (DSSL) algorithms have shown their effectiveness in leveraging unlabeled samples over interconnected networks, where agents cannot share their original data with each other and can only communicate non-sensitive information with their neighbors. However, existing DSSL algorithms cannot cope with data uncertainties and may suffer from high computation and communication overhead problems. To handle these issues, we propose a distributed semi-supervised fuzzy regression (DSFR) model with fuzzy if-then rules and interpolation consistency regularization (ICR). The ICR, which was proposed recently for semi-supervised problem, can force decision boundaries to pass through sparse data areas, thus increasing model robustness. However, its application in distributed scenarios has not been considered yet. In this work, we proposed a distributed Fuzzy C-means (DFCM) method and a distributed interpolation consistency regularization (DICR) built on the well-known alternating direction method of multipliers to respectively locate parameters in antecedent and consequent components of DSFR. Notably, the DSFR model converges very fast since it does not involve back-propagation procedure and is scalable to large-scale datasets benefiting from the utilization of DFCM and DICR. Experiments results on both artificial and real-world datasets show that the proposed DSFR model can achieve much better performance than the state-of-the-art DSSL algorithm in terms of both loss value and computational cost. Our code is available online[1].

Index Terms—distributed semi-supervised learning, fuzzy regression model, fuzzy C-means method, interpolation consistency regularization, alternating direction method of multipliers.

I. INTRODUCTION

Over the past few decades, semi-supervised learning (SSL) algorithms have been thoroughly investigated for ways to use both labeled and unlabeled data to train better models [1]–[8]. The different categories of SSL algorithms generally include self-training [2]–[4], co-training [5], graph-based methods [6]–[8] and MixUp-based methods [9], [10], etc. Most of these methods work by adding regularization terms that deal with unlabeled samples in the loss functions. Further, most of them only work in centralized scenarios, where training data is processed at a central node. However, rising concerns over data privacy and security [11]–[14] has made distributed computing a far more popular paradigm. Consequently, in many real-world scenarios [15]–[17], the available training data is spread across interconnected networks comprising multiple agents. Typically, these agents are not allowed to share the data they have with others and can only communicate non-sensitive information to their neighbors. As such, the training data must be stored and processed on multiple local nodes instead of in one central place. To date, only a few researchers have investigated distributed semi-supervised learning (DSSL) [18]–[20], and none have considered data uncertainty in the training samples. Both properties are common in real-world datasets, so a framework that considers these factors would invariably increase model performance, especially with complex data.

Data uncertainty inherently exists during the process of data collection due to measurement errors, incomplete knowledge and subject difference. Generally, there are two types of data uncertainty: epistemic uncertainty and aleatoric uncertainty. The former usually comes from observations or linguistic ambiguity in knowledge while the latter results mainly from to inherent variability of physical systems. One of the best methods for dealing with incomplete or uncertain information is fuzzy inference systems [21]–[23] that rely on fuzzy logic. As learning machines that find the parameters of fuzzy systems (i.e., fuzzy sets, fuzzy rules), fuzzy neural networks (FNN) [24]–[26] comprise an antecedent and a consequent component that offers a specific architecture for tackling data uncertainty. Actually, fuzzy systems have been introduced to SSL methods for years and has found their way onto many directions [27]–[31]. Fuzzy SSL algorithms, however, are still in their infancy. They usually rely heavily on human knowledge and can only be processed in a centralized way.

In search of a fuzzy SSL method that can better utilize unlabeled samples, we were motivated by a recently developed technique called interpolation consistency regularization (ICR) [10]. Unlike other SSL methods, which essentially use unlabeled data to supplement the available training data, ICR expands the sample space to capture more extrinsic information. ICR not only helps to train a better estimator based on

Ye Shi is with the School of Information Science and Technology, ShanghaiTech University, Shanghai, 201210, China, and with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia. Email: shiye@shanghaitech.edu.cn.

Leijie Zhang and Chin-Teng Lin are with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia. Email: Leijie.Zhang@student.uts.edu.au, Chin-Teng.Lin@uts.edu.au.

Zehong Cao is with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia and the Department of Discipline of ICT, University of Tasmania, Hobart, TAS 7001, Australia. Email: zhcaonctu@gmail.com.

M. Tanveer is with the Discipline of Mathematics, Indian Institute of Technology Indore, Simrol, Indore, 453552, India. Email: mtanveer@iiti.ac.in.

[1]https://github.com/leijiezhang/DSFR

augmented training samples, but also encourages consistency between predictions based on the augmented samples, i.e., $f(\lambda x_i + (1 - \lambda)x_j)$, and interpolated predictions based on those samples, i.e., $\lambda f(x_i) + (1 - \lambda)f(x_j)$. Further, ICR can push the decision boundaries toward low-density areas, which increases model robustness and generalization performance. Currently, ICR methods have been widely used in semi-supervised classification tasks with backpropagation training schemes [9], [10], [32], but seldom with semi-supervised regression tasks.

In this paper, we designed a distributed semi-supervised fuzzy regression (DSFR) model with fuzzy if-then rules and ICR to handle data uncertainty and reduce computation and communication overheads in interconnected networks. A distributed Fuzzy C-means (DFCM) algorithm locates the parameters in an antecedent component and a distributed interpolation consistency regularization (DICR) algorithm obtains the parameters in a consequent component. Both the DFCM algorithm and the DICR algorithm are implemented following the well-known alternating direction method of multipliers (ADMM) [33] to guarantee consensus among all local agents. Data is only ever processed locally by the agent that owns the data, and very little information is passed between neighbors, all of which is non-sensitive. Notably, the DSFR model converges very quickly since it does not involve a backpropagation procedure. Benefiting from DFCM and DICR, it also scales well to large datasets. Thus, the main contributions of this paper include:

- A novel DSFR model with ICR that handles data uncertainty and has lower computation and communication overheads in interconnected networks than existing DSSL algorithms.
- A DFCM algorithm to locate parameters in the antecedent component of the DSFR model. The DFCM can be used directly with both labeled and unlabeled training data available over interconnected networks.
- A DICR algorithm to obtain parameters in the consequent component of the DSFR model. This is the first implementation to extend ICR to a distributed and semi-supervised scenario. In contrast to existing DDSL algorithms, such as graph-based DDSL [19], [20], DICR results in smaller loss values and enjoys much greater scalability.

The remainder of the paper is organized as follows: Section II describes related work; Sections III and IV are devoted to centralized and distributed semi-supervised fuzzy regression with ICR, respectively; and Section V presents the experiments conducted to evaluate and verify the proposed model.

II. RELATED WORK

A. Distributed Learning

Distributed learning algorithms [34]–[38] have been successfully applied to many real-world applications, including wireless sensor networks [35] and privacy-preserving [39]. Qin et al. [35] combined graph theory with the distributed consensus theory of multi-agent systems to design two decentralized algorithms for dealing with distributed problems in wireless sensor networks. One algorithm is based on K-means, the other on FCM. Targeting privacy preservation, Liu et al. [39] devised a novel shadow coding schema to save and recover privacy information over distributed networks. Two interesting works in [40] and [41] provided novel learning algorithms for distributed games. Deep learning solutions involving consensus models over distributed deep architectures have also been proposed with the aim of improving performance [42], [43]. For example, Forero et al. [34] proposed a fully distributed method based on support vector machine algorithms by applying an ADMM strategy to obtain optimal global parameters without the need to exchange and process information through a central communications unit. Ye et al. [37] later presented a decentralized ELM algorithm that combines the Jacobian and Gauss-Seidel Proximal ADMM methods. Obviously, all these distributed learning algorithms do well in alleviating privacy concerns and dealing with decentralized scenarios. However, none address uncertainty in the training data.

There are also several examples of fuzzy algorithms for distributed learning [44]–[47]. For example, Fierimonte et al. [44] developed a decentralized FNN with random weights, where parameters in the fuzzy membership functions are chosen randomly as opposed to being trained. In subsequent work, they introduced an online implementation of the same FNN structure [45]. Notably, a random method of identifying parameters can result in very large deviations in accuracy during the learning process. Additionally, the algorithms are only applied in the consequent layers of the FNN [44], [45], which means that, strictly speaking, this decentralized FNN model is only partially distributed. A more recent proposition by Shi et al. [46] involves a distributed FNN with a consensus learning strategy. A novel method of distributed clustering optimize the parameters in the antecedent layer, while a similar method of distributed parameter learning does the same for the consequent layer. This solution successfully manages data uncertainty in a distributed setting, but it does not consider unlabeled samples. Dang et al. [47] proposed a transfer fuzzy clustering method to enable neighboring agents to learn from each other collaboratively. Similar to distributed learning, multi-view learning converges to an optimal estimator by collaboratively learning from multiple datasets. By applying a large margin learning mechanism, [48] proposed a two-view fuzzy model that collaboratively learning from each other. A multi-view fuzzy logic system is introduced in [49] by adopting nonnegative matrix factorization to build a hidden space in order to extract useful information shared among different datasets.

B. Semi-supervised learning

SSL algorithms [2], [3], [5] were developed to improve model performance with additional training data in the form of unlabeled samples, while still leveraging the accuracy afforded by labeled samples. The first tools to solve SSL problems were self-training [2], [3] methods, which uses unlabeled samples in an iterative two-part procedure. First, the model is trained on a set of only labeled samples. The trained model is then used to predict the labels of unlabeled data. High-confidence predictions are then added to the training set and

the model is retrained. An alternative strategy is co-training [5], which takes a similar iterative approach to self-training. The difference is that co-training algorithms maintain two separate estimators which work together on different training subsets and teach each other during the learning process. Subsequently, generative models [1], [50], [51] were proposed. These algorithms assume that the model can represent hybrid distributions as identified in a set of unlabeled samples.

More recently, an increasing number of researchers are incorporating regularization terms into the loss function of SSL algorithms as a way to extract more useful information from the unlabeled data. Generally, these regularization terms fall into one of three categories. These are: traditional regularization [52], which summarizes and transfers traditional SSL models into regularization terms; consistency regularization [53], which forces the predictions generated to be low-entropy so the decision boundary does not land in a dense sample area; and entropy minimization [54], [55], which guarantees that the distribution of the augmented dataset will be the same as the original.

To implement SSL in deep architecture, an interpolation consistency training procedure, Verma et al. [10] proposed a solution inspired by MixUp [9] that learns a decision boundary which ensures consistency between predictions based on the interpolated samples and interpolations based on the resulting predictions. Meanwhile, Berthelot et al. [32] was developing MixMatch – the current state-of-the-art in model performance. MixMatch basically combines many of the recent dominant SSL mechanisms, including entropy minimization and MixUp tools. However, all these algorithms are designed for centralized scenarios and cannot applied directly to decentralized problem solving.

Only a few DSSL methods have been proposed [18]–[20] Chang et al. [18] uses unlabeled data to reduce distribution errors and applied time-consuming kernel ridge regression on distributed nodes, after which the weighted average of those nodes' outputs is calculated to obtain a final estimator. Taking advantage of a wavelet neural network, Xie et al. [19] devised a new DSSL scheme that incorporates a graph-based regularization term into a distributed loss function. However, the process involves constructing a relationship graph of all the sample points, which is quite time-consuming, especially with large-scale datasets. In pursuit of better efficiency, the researchers later modified their information sharing strategy to use an event-triggered communication scheme [20], and in a later work still, they updated the objective function to reduce the complexity of loss function by avoiding the twice continuously differential in their previous two models [56]. While there are many ways reduce computing time, high computational overhead is an inherent problem with graph-based regularization that cannot be solved completely. Ultimately, the choice of DSSL strategy comes down to one of either less robustness or more time-consumption.

III. CENTRALIZED SEMI-SUPERVISED FUZZY REGRESSION

This section sets out the formulation for the centralized semi-supervised fuzzy regression (CSFR) model with ICR.

Notably, CSFR is sequentially trained by an unsupervised structure learning and a semi-supervised parameter learning.

A. Fuzzy inference system

Let us briefly describe the fuzzy inference system based on a first-order Takagi-Sugeno (T-S) method. Consider the estimation of a scalar output $y \in \mathbb{R}$ from a D-dimensional input $x = [x_1, x_2, \cdots, x_D]$. The k-th fuzzy rule can be represented as

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

where A_{kj} is a Gaussian fuzzy set with the following membership function:

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

where m_{kj} and σ_{kj} are respectively the mean value and standard deviation.

The firing strength for each fuzzy rule is

$$\bar{\phi}_k(x) = \frac{\prod_{j=1}^{D} \varphi_{kj}(x_j)}{\sum_{k=1}^{K} \prod_{j=1}^{D} \varphi_{kj}(x_j)}. \tag{2}$$

where K denotes the number of applied fuzzy rules.

The overall output is obtained by summing the outputs of all fuzzy rules multiplied with a weighted vector:

$$\hat{y} = \sum_{k=1}^{K} \bar{\phi}_k(x)(w_{k0} + \sum_{j=1}^{D} w_{kj} x_j), \tag{3}$$

The structure learning process therefore aims to optimize the parameters of the Gaussian membership functions in (1) for each fuzzy rule, i.e., m_{kj} and σ_{kj}, $j \in \{1, \cdots, D\}$, which is done through the FCM clustering method [57]. In turn, the goal of the parameter learning process is to identify the output weights $w_{k0}, \cdots, w_{kD}$ in (3), done with a least-squares algorithm [58]. A more detailed description of the structure learning procedure follows next.

B. Fuzzy C-Means for the structure learning

As mentioned, the structure learning process is governed by the FCM algorithm. The fuzzy rules are generated by clustering training data into several groups where each group corresponds to a fuzzy rule. This procedure follows.

Let $\mathcal{D} := \{(X_i, Y_i)|X_i \in \mathbb{R}^D, Y_i \in \mathbb{R}, \ i \in \{1, \cdots, N\}\}$ denote the training data set and x_{ij} denote the j-th feature of the i-th sample X_i. The aim of FCM algorithm is to partition N samples into K groups $\mathcal{C} := \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$, where K denotes the total number of fuzzy rules and is often assumed to be known as a priori. The most important task for the structure learning is to identifying the total number K of clusters, where the $k-$th center can be identified follows:

$$\mathbf{m}_k = \arg\min_{\mathbf{m}_k} \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{C}_k|} u_{ik}^{\alpha} ||X_i - \mathbf{m}_k||^2 \tag{4}$$

where $|\mathcal{C}_k|$ denotes the $k-$th cluster, and $\alpha \geq 1$ determines the level of cluster fuzziness. (α is commonly set to 2.) With an iterative technique, the FCM algorithm then assembles and refines the clusters. All the K centers at iteration $t = 0$ are initialized randomly, i.e., $\{\mathbf{m}_1(0), \cdots, \mathbf{m}_K(0)\}$, the procedures from $t + 1$ iterates as follows:

- Update the membership value:

$$u_{ik}^{\alpha}(t+1) = \frac{1}{\sum_{c=1}^{K} \left(\frac{||X_i - \mathbf{m}_k(t)||}{||X_i - \mathbf{m}_c(t)||} \right)^{\frac{2}{\alpha-1}}} \quad (5)$$

- Update the cluster center:

$$\mathbf{m}_k(t+1) = \frac{\sum_{i=1}^{N} u_{ik}^{\alpha}(t+1)X_i}{\sum_{i=1}^{N} u_{ij}^{\alpha}(t+1)} \quad (6)$$

The algorithm stops to optimize when the values of centers stay unchanged during the iteration, and the obtained centers become the centers of each fuzzy set. Meanwhile, the standard variance σ_{kj} of $k-$th fuzzy set can be written as

$$\sigma_{kj} = \sqrt{\sum_{i=1}^{N} u_{ik}^{\alpha}(X_{ij} - m_{kj})^2 / \sum_{i=1}^{N} u_{ik}^{\alpha}} \quad (7)$$

And voilà, we have all the parameters in the antecedent layer.

To calculate the output weights in the consequent layer, parameter learning is described next with a closed-form solution.

C. Closed-form solution for the parameter learning

Parameter learning begins with a hidden matrix defined as $H(X) := [H_1, \cdots, H_K] \in \mathbb{R}^{N \times K(D+1)}$, where

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

and the output vector $Y := [Y_1, \cdots, Y_N]$.

The output weight matrix $\mathbf{w} \in \mathbb{R}^{K(D+1)}$ can be written as:

$$\mathbf{w} = [w_{10}, \cdots, w_{1d}, \cdots, w_{K0}, \cdots, w_{KD}]^T, \quad (9)$$

it can be identified by calculating the optimization problem follows,

$$\min \mathcal{F}_s(\mathbf{w}; X) = \min_{\mathbf{w}} \frac{1}{2}||Y - f(\mathbf{w}; X)||^2 + \frac{\mu}{2}||\mathbf{w}||^2, \quad (10)$$

where $f(\mathbf{w}; X) = H(X)\mathbf{w}$, $\mu > 0$ trades off the model performance between training error and model generalization. (10) is a standard least-squares optimization problem, which can be obtained with a closed form solution follows:

$$\mathbf{w} = (H^T H + \mu I)^{-1} H^T Y, \quad (11)$$

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

D. Semi-supervised fuzzy regression with ICR

As mentioned before, using ICR to solve semi-supervised fuzzy regression problems can push decision boundaries into low-density areas, leading to better generalization performance in semi-supervised scenarios [9], [10]. We followed the data augmentation techniques in [9], [10] to generate interpolation consistency loss and involved it into the objective function (10).

The training set consists of a labeled data set $X \in \mathcal{C}_s$ and an unlabeled data set $U \in \mathcal{C}_u$, denoted as $\mathcal{C} = \mathcal{C}_s \cup \mathcal{C}_u$. Accordingly, the number of the training samples can be expressed as $N = N_s + N_u$. ICR augments this training set with virtual samples constructed from the unlabeled data set $\mathcal{C}_u$ as follows:

$$\tilde{U} = \lambda U_1 + (1-\lambda)U_2, \ U_1, U_2 \in \mathcal{C}_u, \quad (12)$$
$$f(\mathbf{w}; \tilde{U}) = \lambda f(\mathbf{w}; U_1) + (1-\lambda)f(\mathbf{w}; U_2) \quad (13)$$

where λ is randomly sampled from a beta distribution.

The objective function of semi-supervised fuzzy regression (SFR) with ICR is

$$\min \mathcal{F}(\mathbf{w}; X, U) = \min_{\mathbf{w}} \mathcal{F}_s(\mathbf{w}; X) + \gamma \mathcal{F}_u(\mathbf{w}; U), \quad (14)$$

where

$$\begin{aligned} \mathcal{F}_u(\mathbf{w}; U) &= ||f(\mathbf{w}; \tilde{U}) - (\lambda f(\mathbf{w}; U_1) + (1-\lambda)f(\mathbf{w}; U_2))||^2 \\ &= ||H(\tilde{U})\mathbf{w} - (\lambda H(U_1)\mathbf{w} + (1-\lambda)H(U_2)\mathbf{w})||^2, \\ &\triangleq ||B(U)\mathbf{w}||^2, \end{aligned} \quad (15)$$

where $B(U) = H(\tilde{U}) - \lambda H(U_1) - (1-\lambda)H(U_2)$. The closed-form solution of (14) is:

$$\mathbf{w} = (\gamma B^T(U)B(U) + \mu I + H^T(X)H(X))^{-1} H^T(X)Y. \quad (16)$$

IV. DISTRIBUTED SEMI-SUPERVISED FUZZY REGRESSION WITH ICR

In this section, we extend the CSFR model to its distributed version and design distributed training algorithms for it. Since the CSFR model is sequentially trained by structure learning and parameter learning, we will develop a distributed structure learning and a distributed parameter learning sequentially.

Our distributed computing scenario is conceived as an undirected graph $\mathcal{G} = \{\mathcal{L}, \xi\}$, of L agents (nodes) connected by E edges, where $\mathcal{L}$ and ξ denotes the nodes set and the edges set, respectively. Agent l is the target agent, and $\mathcal{N}_l$ is the set of agents neighboring agent l. Given a dataset $\mathcal{D} := \{(X_i, Y_i)|i \in \mathcal{N}\}$, let $\{\mathcal{D}^1, \cdots, \mathcal{D}^L\}$ be its decomposition in entirety. The subset $\mathcal{D}^l$ is the data housed on the $l-$th node, where $l \in \mathcal{L}$. The subset of samples located on the $l-$th node can be then denoted as $\mathcal{C}^l$. Within each subset $\mathcal{C}^l$, $\mathcal{C}_k^l$ denotes the subset of smples in the $k-th$ cluster of agent l such that $\bigcup_{k=1}^{K} \mathcal{C}_k^l = \mathcal{C}^l$. An illustration of this architecture is shown in Fig.1.

The corresponding structure learning problem is solved through a consensus strategy, formulated as

$$\min_{\mathbf{m}_k^l} \frac{1}{2} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{X_i^l \in \mathcal{C}_k^l} (u_{ik}^q)^{\alpha} ||X_i^l - \mathbf{m}_k^l||^2 \quad (17a)$$

$$\text{s.t.} \quad \mathbf{m}_k^l = \mathbf{r}_k, \ l \in \mathcal{L}, \ k \in \mathcal{K}, \quad (17b)$$

where $\mathbf{m}_k^l$ represents the local center of the $k-h$ fuzzy set on agent l, $\mathbf{r}_k$ denotes the global center, which integrates all the
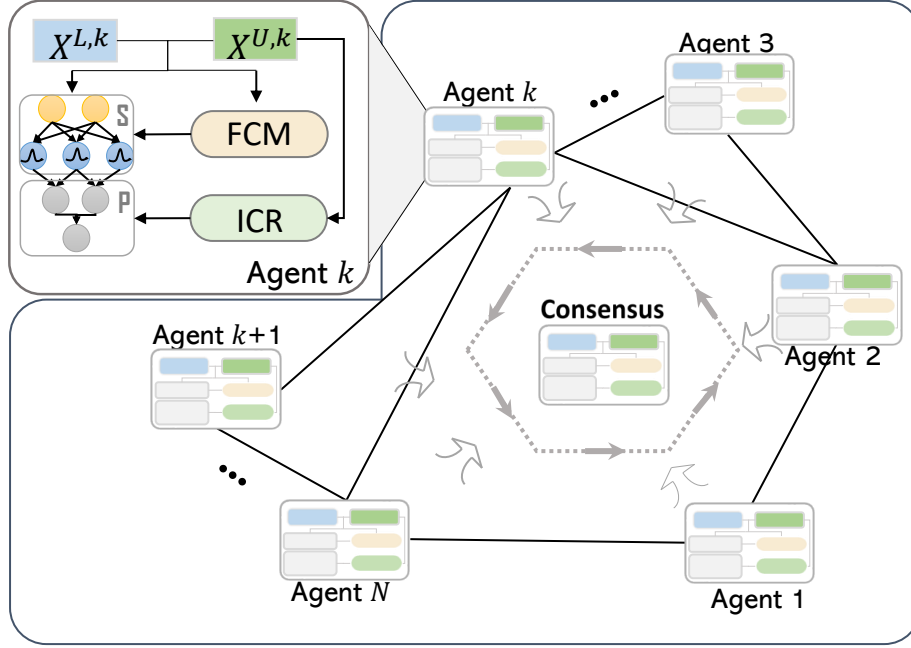
Fig. 1. Architecture of the DSFR model. The upper-left part depicts detailed structures of each local model, which is presented in the bottom-right part. Different colors are used to distinguish different types of data and methods.

local centers, and $|\mathcal{C}_k^l|$ is the cardinality operation for a local subset. Moreover, (17b) is the constraint that assures all local centers coincide at one global center. Notably, all the local variables among different agents can be parallelly computed, thus improving the computing speed.

After Identifying all the global centers, the global standard variance can be calculated via

$$\bar{\sigma}_{kj} = \sqrt{\frac{1}{N}\sum_{l=1}^{L}|\mathcal{C}^l|(\sigma_{kj}^l)^2} \tag{18}$$

where $|\mathcal{C}^l|$ is the cardinality of local subset $\mathcal{C}^l$ on agent l, σ_{kj}^l denotes the j-th element of fuzzy rule k's standard variance on subset $\mathcal{C}^l$, and $\bar{\sigma}_{kj}$ is the corresponding to global standard variance for all agents. Note that the standard variance σ_{kj}^l corresponding to the k-th rule and j-th dimension is calculated in the following element-wise method:

$$\sigma_{kj}^l = \sqrt{\sum_{X_i^l \in \mathcal{C}_k^l}(u_{ik}^l)^\alpha(X_{ij}^l - \mathbf{m}_{kj}^l)^2 / \sum_{X_i^l \in \mathcal{C}_k^l}(u_{ik}^l)^\alpha} \tag{19}$$

where X_{ij}^l and $\mathbf{m}_{kj}^l$ are the j-th components of X_i^l and $\mathbf{m}_k^l$, respectively.

The parameter learning process can be similarly modeled as follows:

$$\min_{\mathbf{w}^l} \quad \frac{1}{2}\sum_{l=1}^{L}(||Y^l - H(X^l)\mathbf{w}^l||^2$$
$$+\gamma||B(U^l)\mathbf{w}^l||^2) + \frac{\mu}{2}||\mathbf{z}||^2, \tag{20a}$$
$$\text{s.t.} \quad \mathbf{w}^l = \mathbf{z},\ l \in \mathcal{L}, \tag{20b}$$

where $\mathbf{w}^l$ represents $l-$th agent's the local output weight, and $\mathbf{z}$ is a the global weight that integrates local weights for all agents.

A. Distributed FCM

The optimization problem (17) is nonconvex, so using an exhaustive search method to solve the problem would not be efficient. Further, centralized clustering methods, such as the FCM algorithm used for structural learning, have several shortcomings including high communications overheads, no attention to privacy, and poor scalability when coping with large-scale data. Hence, a distributed variant of the centralized FCM algorithm is needed to address these issues. Accordingly, our formulation of a DFCM algorithm follows.

The first step is to construct the following augmented Lagrangian for (17):

$$\mathcal{L}_s(\mathbf{m}, \mathbf{r}, \boldsymbol{\lambda}_s) = \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{X_i^l \in \mathcal{C}_k^l}(u_{ik}^q)^\alpha||X_i^l - \mathbf{m}_k^l||^2$$
$$+\sum_{l=1}^{L}\sum_{k=1}^{K}\boldsymbol{\lambda}_{s,kl}^T(\mathbf{m}_k^l - \mathbf{r}_k)$$
$$+\frac{1}{2}\rho_s\sum_{l=1}^{L}\sum_{k=1}^{K}||\mathbf{m}_k^l - \mathbf{r}_k||^2 \tag{21}$$

where $\boldsymbol{\lambda}_{s,kl}$ denotes the Lagrange multiplier and ρ_s is a positive penalty factor. Denoting these parameters at iteration t as $\mathbf{r}(t)$ and $\boldsymbol{\lambda}_s(t)$, the variables are obtained iteratively using the following ADMM-based updating steps:

$$(u_{ik}^l)^\alpha(t+1) = \frac{1}{\sum_{c=1}^{K}(\frac{||X_i^l - \mathbf{m}_k^l(t)||}{||X_i^l - \mathbf{m}_c^l(t)||})^{\frac{2}{\alpha-1}}}, \tag{22}$$

$$\mathbf{m}^l(t+1) = \arg\min_{\mathbf{m}}\mathcal{L}(\mathbf{m}^l, \mathbf{r}(t), \boldsymbol{\lambda}_s(t)), \tag{23}$$

$$\mathbf{r}(t+1) = \arg\min_{\mathbf{r}}\mathcal{L}(\mathbf{m}^l(t+1), \mathbf{r}, \boldsymbol{\lambda}_s(t)), \tag{24}$$

$$\boldsymbol{\lambda}_{s,kl}(t+1) = \boldsymbol{\lambda}_{s,kl}(t) + \rho_s(\mathbf{m}_k^l(t+1) - \mathbf{r}_k(t+1)). \tag{25}$$

It is worth noting that (23) can be solved in parallel among different agents.

The cluster centers $\mathbf{m}^l(t+1)$ for each agent l are updated through the same assignment and update steps from the centralized FCM algorithm. But, it's easy to know that the solutions (23) and (24) can be calculated by setting the partial derivative of the corresponding variable to zero. Thus, the closed-form solution of (23) and (24) are as follows:

$$\mathbf{m}_k^l(t+1) = \frac{\sum_{X_i^l \in \mathcal{C}_k^l}(u_{ik}^l(t))^\alpha X_i - \boldsymbol{\lambda}_{s,kl}(t) + \rho_s \mathbf{r}_k(t)}{\sum_{X_i^l \in \mathcal{C}_k^l}(u_{ik}^l(t))^\alpha + \rho_s} \tag{26}$$

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

where

$$\bar{\mathbf{m}}_k^l(t+1) = \frac{1}{L}\sum_{l=1}^L \mathbf{m}_k^l(t+1), \tag{28}$$

$$\bar{\boldsymbol{\lambda}}_{s,kl}(t) = \frac{1}{L}\sum_{l=1}^L \boldsymbol{\lambda}_{s,kl}(t). \tag{29}$$

Algorithm 1 summarizes the above approach. The convergence behavior is examined by checking the norms of the following two residuals:

$$||\mathbf{m}_k^l(t) - \mathbf{m}_k^q(t)||^2 \le \epsilon_1, \tag{30}$$
$$||\boldsymbol{\lambda}_{s,k}^l(t) - \boldsymbol{\lambda}_{s,k}^l(t-1)||^2 \le \epsilon_2. \tag{31}$$

Algorithm 1 Distributed FCM (17)

Initialization: Set $t = 0$ and the Lagrange multipliers $\boldsymbol{\lambda}_{s,kl}(t) = \mathbf{0}$. Initialize cluster centers and assign them to each agent l.
  for $t = 0, 1, 2, \cdots$, **do**
    Update the membership value $(u_{ik}^l)^\alpha$ by (22) for each agent l.
    Update the local variables $\mathbf{m}^l(t+1)$ by (26) and broadcast it to each agent l.
    Update the global variables $\mathbf{r}(t+1)$ by (27) and broadcast them to each agent l.
    Update the dual variables $\boldsymbol{\lambda}_s(t+1)$ by (25) and broadcast them to each agent l
  end for

The computation complexity of DFCM relies on the local variables update and global variables update. The computation complexity for DFCM is $\mathcal{O}(NDK^2T_1)$, where T_1 is the required iteration of DFCM. Note that the DFCM is built on the well-known ADMM algorithm, which generally converges to modest accuracy (such as $\epsilon < 10^{-3}$) within a few tens of iterations. Therefore, the total computational complexity of DFCM is quite limited.

B. Distributed ICR

Likewise, a distributed variant of ICR is needed to tackle the regression problem. In contrast to existing DDSL algorithms [19], [20], which are typically based on graph operations, DICR is more faster and more accurate with dramatically reduced computing times, especially with large-scale datasets. The formulation follows.

The augmented Lagrangian for (20) is

$$\mathcal{L}(\mathbf{w}^l, \mathbf{z}, \boldsymbol{\beta_q}) = \frac{1}{2}\sum_{l=1}^L (||Y^l - H(X^l)\mathbf{w}^l||^2 + \gamma||B(U^l)\mathbf{w}^l||^2)$$
$$+ \frac{\mu}{2}||\mathbf{z}||^2 + \sum_{l=1}^L \boldsymbol{\beta}_l^T(\mathbf{w}^l - z)$$
$$+ \frac{1}{2}\rho_p \sum_{l=1}^L ||\mathbf{w}^l - z||^2 \tag{32}$$

where $\boldsymbol{\beta}_l$ is the Lagrange multiplier, and ρ_p is a small positive penalty parameter. This augment Lagrangian is then solved using ADMM method by

$$\mathbf{w}^l(t+1) = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}(t), \boldsymbol{\beta}(t)), \tag{33}$$
$$\mathbf{z}^l(t+1) = \arg\min_{\mathbf{z}} \mathcal{L}(\mathbf{w}(t+1), \mathbf{z}, \boldsymbol{\beta}(t)), \tag{34}$$
$$\boldsymbol{\beta}_l(t+1) = \boldsymbol{\beta}_l(t) + \rho_p(\mathbf{w}^l(t+1) - \mathbf{z}(t+1)). \tag{35}$$

The closed-form solution of (33) is clear; w on the l-th agent is updated by

$$\mathbf{w}^l(t+1) = Q((H^l(X))^T Y^l + \rho_p \mathbf{z}(t) - \boldsymbol{\beta_l}(t)), \tag{36}$$

where I is the identity matrix with dimension $K(D+1)$ and Q can be obtained by

$$Q = ((H^l(X))^T H^l(X) + \gamma B^T(U)B(U) + \mu I)^{-1}. \tag{37}$$

The solution for (34) is similarly found by

$$\mathbf{z}(t+1) = \frac{\sum_{l=1}^L (\beta_l + \rho_p \mathbf{w}^l(t+1))}{\mu + \rho_p L}, \tag{38}$$

Algorithm 2 Distributed ICR

Initialization: set $t = 0$ and initialize the global weight $\mathbf{z}(t)$ and Lagrange multipliers $\beta(t)$ for each agent l.
  for $t = 0, 1, 2, \cdots$, **do**
    Augmented data generation:
    Randomly select M samples from the unlabeled data set $\mathcal{C}_u$ as U_1 and another M samples as U_2. Then generate M number of λ using a beta distribution, and generate M augmented samples using the interpolation technique in (12).
    Update the local variables $w^l(t+1)$ via (36) for each agent l.
    Update the global variable $z(t+1)$ via (38) and broadcast it to the other agents.
    Update the dual variables $\beta(t+1)$ by (35) and broadcast them to the other agents.
  end for

The computation complexity of DICR is $\mathcal{O}(KDLT_2)$ if we ignore the cost of data augmentation, where T_2 is the iteration number required by ADMM procedure. As we analyzed before, T_2 is generally about a few tens. Therefore, the total computation complexity of DICR is small.

V. EXPERIMENTS

To verify the effectiveness of the DSFR framework, we conducted extensive experiments on six different types of datasets. Descriptive statistics are provided in Table I; brief descriptions follow:

- The Airfoil dataset [59]. Assembled by NASA, this dataset contains readings of noise tests with different-sized NACA 0012 airfoils.
- The WarCraft Master Skill Level (WMSL) dataset [60]. A set of playing data drawn from the WarCraft real-time strategy game, this set contains the playing data of 3360 gamers. After removing broken records, the final dataset numbered data on 3338 players.
- The Combined Cycle Power Plant (CCPP) dataset [61]. This set contains 9568 samples of gas and steam turbine loads. We chose the full electrical power load as the target label and used the remaining data to train the model.
- The California Housing (CH) dataset [62]. A data set of information about real estate in California, we discarded all non-numeric features and treated the median house value of a district as the target.
- The King County Housing (KCH) dataset [63]. This is a collection of houses sold in King County USA. After removing time-related and restriction-related variables, we used house price as the target and the rest of the features for training.
- The artificial dataset. We built this dataset to investigate the performance of the DSFR framework. 5,000 samples $X \in \mathbb{R}^8$ were generated from a Gaussian distribution with a mean of $[0, 0.5, \cdots, 3.5]$ and a standard deviation of $[0.2, 0.4, \cdots, 1.6]$. The corresponding label Y was given by: $Y = 0.3 \sum_{i=1}^{8}(X_i)^2 + 0.7 \sum_{i=1}^{8}(\cos(X_i))$. We denoted the range of labels as $R = max(X) - min(X)$ and generated two types of noise from Gaussian distributions $\mathcal{N}(0.1R, (0.1R)^2)$ and $\mathcal{N}(R, (0.1R)^2)$, respectively. Then, 5% level of type-1 noise and 10% type-2 noise were randomly added to labels.

It should be noted that all feature values of these datasets are normalized between -1 and 1. As listed in Table I, a 5-fold cross-validation is used in our experiment to randomly generate 5 test sets and 5 training sets, among which we randomly choose 50 samples as labeled data and the rest as unlabeled data. Then we re-conduct the experiment 10 times and get the average results and its standard variance. For fairness, we arrange the training sets on an interconnected networks with five fully connected agents, and we set rule number as 5 in all agents for all these datasets. All the experiments are conducted on computing nodes with 26 cores of 2.11 GHz and 95 GB memory.

All feature values in all datasets were normalized between -1 and 1. As noted in Table I, we randomly generated 5 test sets

TABLE I
DATASET INFORMATION

Dataset	Features	Samples	Labeled	Unlabeled	Test
Airfoil	5	1503	50	1153	300
WMSL	18	3338	50	2621	667
CCPP	4	9568	50	7605	1913
CH	8	20640	50	16462	4128
KCH	15	21613	50	17191	4322

and 5 training sets using 5-fold cross-validation, from which we randomly chose 50 samples to serve as the labeled data; the remaining samples were unlabeled data. All experiments were conducted on a computing node comprising 26 CPU cores of 2.11 GHz and a memory of 95 GB. The evaluation metric was normalized root mean square error (NRMSE), calculated by

$$\text{NRMSE} = \sqrt{\frac{1}{N\hat{\sigma}_Y^2} \sum_{i=1}^{N} (\hat{Y}_i - Y_i)^2}, \tag{39}$$

To analyze the convergence of our proposed model and the influence of the rule number and the sample number that is chosen for ICR regularization, we fix the Laplacian parameter ρ_p and ρ_s at 0.1 and 0.1, and the parameters μ, γ, and η, which is corresponding to the L_2norm regularizer, the mix-up regularizer, and the graph-based regularizer, at 0.1, 0.1, and 0.0001, respectively.

To obtain the best results of our model, all parameters are chosen from an interval of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Besides, we set the FCM parameter α as 1.1 to increase the diversity of fuzzy rules in all experiments.

A. Performance on Different Datasets

All parameters in this section were chosen from an interval of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Further, the FCM parameter α was set to 1.1 for all experiments to increase the diversity of fuzzy rules. We repeated each experiment 10 times, reporting the average NRMSE and its standard variance as the results. For a fair comparison, we arranged the training sets on interconnected networks with five fully connected agents, setting the number of rules to 5 for all agents with all datasets. We used 50 samples in the centralized scenario and 10 samples for each agent in the distributed setting.

We compared several alternative methods within the semi-supervised fuzzy regression (SFR) framework that using different regularization, fully-supervised regression (FR) model, and one SSL model. All methods are compared in both centralized and distributed ways and described follows:

- s-SFR: A variant of SFR, this method apply FCM method in the structure learning process to leverage both labeled and unlabeled samples. A ℓ_2 regularization is used in the parameter learning process to enhance model generalization performance.
- **SFR-ICR**: Our proposed SFR model, based on s-SFR, this fuzzy regression model add an ICR term in the parameter learning process to further utilize unlabeled data.

TABLE II
PERFORMANCE COMPARISON ON EACH DATASET

Datasets	Algorithms	Performance					
		Centralized Methods			Distributed Methods		
		Test NRMSE	T-test	Time(s)	Test NRMSE	T-test	Time(s)
Airfoil	FR	0.6657/0.0188	0.0011	0.0581	0.6658/0.0345	0.0053	0.3058
	LapWNN	1.3967/0.2478	0.0002	0.2151	1.4428/0.3934	0.0000	7.1624
	s-SFR	2.4290/1.5583	0.0419	0.1183	1.6763/0.6645	0.0134	0.2809
	SFR-G	1.3124/0.3607	0.0078	0.2066	1.2663/0.3048	0.0047	0.4250
	SFR-ICR	**0.7423/0.0288**	-	0.1363	**0.7361/0.0230**	-	0.8179
WMSL	FR	0.6702/0.0176	0.0047	0.2811	0.6768/0.0244	0.0017	0.5923
	LapWNN	**0.7475/0.0865**	0.0323	27.741	0.8734/0.0576	0.0110	10.678
	s-SFR	0.9871/0.0788	0.0044	0.2953	0.8710/0.0415	0.0013	0.5864
	SFR-G	0.9199/0.0474	0.0468	1.0250	0.8761/0.0458	0.0015	0.9353
	SFR-ICR	0.7648/0.0311	-	0.5166	**0.7588/0.0312**	-	1.7486
CCPP	FR	0.2472/0.0075	0.0003	0.1173	0.2472/0.0075	0.0003	0.2711
	LapWNN	0.7475/0.0865	0.0001	27.741	0.8734/0.0576	0.0001	10.678
	s-SFR	0.3342/0.0528	0.0571	0.0878	0.3258/0.0234	0.0026	0.2297
	SFR-G	0.3187/0.0412	0.0811	7.6291	0.3556/0.0825	0.0095	1.2010
	SFR-ICR	**0.2809/0.0100**	-	0.1227	**0.2780/0.0083**	-	0.6057
KCH	FR	0.5631/0.0080	0.0019	0.2485	0.5640/0.0047	0.0001	0.8764
	LapWNN	0.7262/0.0266	0.0969	93.464	0.9348/0.0291	0.0256	24.184
	s-SFR	0.8063/0.0433	0.0041	0.1947	0.8075/0.0800	0.0103	0.9030
	SFR-G	0.7728/0.0432	0.0123	89.677	0.7925/0.0706	0.0011	9.8767
	SFR-ICR	**0.6952/0.0270**	-	0.5664	**0.6673/0.0211**	-	2.5265
CH	FR	0.5426/0.0180	0.0000	0.8325	0.5393/0.0220	0.0000	2.7241
	LapWNN	1.0023/0.0650	0.0223	94.025	1.0382/0.0294	0.0000	24.801
	s-SFR	0.9387/0.1030	0.0012	0.6695	0.9503/0.1207	0.0451	2.4122
	SFR-G	0.9886/0.0113	0.0093	53.773	0.9313/0.0586	0.0341	9.6628
	SFR-ICR	**0.6875/0.0406**	-	1.1965	**0.7703/0.1217**	-	6.1851
Artificial	FR	1.0067/0.0022	0.0435	0.3371	1.0070/0.0063	0.0019	0.6782
	LapWNN	1.2384/0.0824	0.0052	2.5700	1.0891/0.0462	0.0034	8.2684
	s-SFR	1.1590/0.1202	0.0027	0.2177	1.2529/0.1662	0.0016	0.7128
	SFR-G	1.0715/0.0478	0.0261	1.9241	1.0819/0.0745	0.0007	1.0998
	SFR-ICR	**1.0409/0.0339**	-	0.5266	**1.0121/0.0094**	-	1.7868

- SFR-G: Another variant of SFR, different from SFR-ICR, this model involves a graph-based regularization term. We designed this method to show the efficiency and effectiveness of SFR-ICR under the same settings.
- FR: This is a fully-supervised fuzzy regression model. FR can provide a lower bound of the loss for the SFR-ICR.
- LapWNN: This method involves a graph-based SSL based on a wavelet neural network (WNN) [19]. Since LapWNN was developed for distributed semi-supervised scenarios, thus it is a quite relevant baseline method.

The results shown in Table II, place SFR-ICR as the clear outstanding performer in both the centralized and distributed

settings, providing a strong endorsement of the effectiveness of semi-supervised fuzzy regression. T-tests were implemented to show statistical significance between SFR-ICR and other methods. As shown in Table II, all the p-values are smaller than 0.05, thus the difference between our SFR-ICR and other methods are significant.

Notably, SFR-G outperformed LapWNN on all datasets in the distributed scenarios, indicating that FNN-based SSL outperforms WNN-based SSL methods. Particularly, SFR-ICR reduced 31.96% NRMSE values compared with the distributed results by LapWNN on average of all datasets. Within the SFR scheme, the NRMSE is also smaller by SFR-
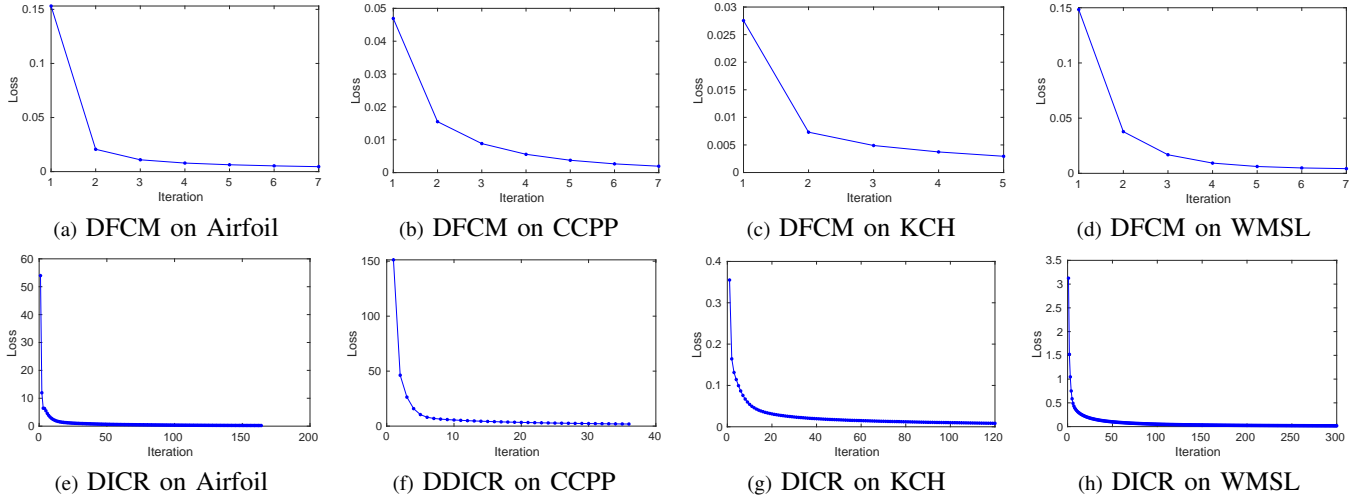
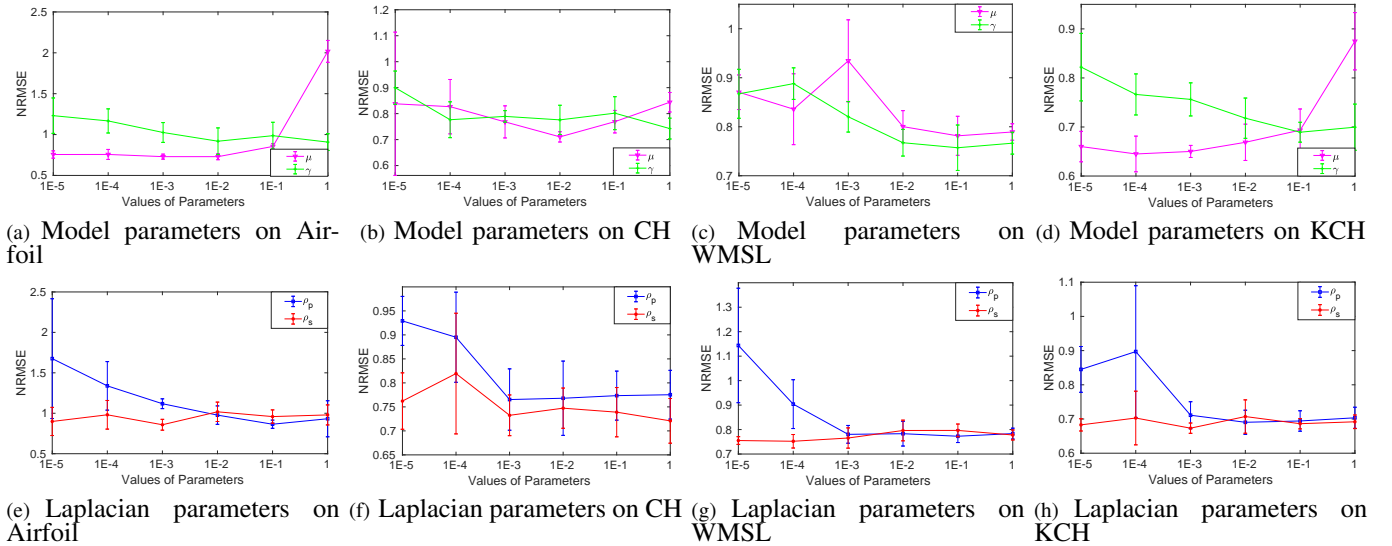Fig. 2. Convergence analysis of the DFCM method (top) and DICR method (bottom).

Fig. 3. NRMSE and its standard deviation with different model parameters (top) and Laplacian parameters (bottom).

ICR than by s-SFR and SFR-G, closely outpacing FR, which indicates that ICR has a much greater ability to extract useful information from unlabeled samples than other regularization methods. Particularly, SFR-ICR respectively reduced 23.19% and 19.44% NRMSE values compared with the distributed results by s-SFR and SFR-G on average of all datasets.

The SFR-ICR is also superior in efficiency. SFR-ICR costs far less time than other methods in centralized ways. For distributed methods, the reason computing speeds of some small-scale dataset is slower is that communication overhead among agents offsets the computing superiority of SFR-ICR.

We also assessed distributed SFR-ICR (DSFR-ICR) against its centralized counterpart SFR-ICR (CSFR-ICR). DSFR-ICR performed better on all datasets, except for California Housing, demonstrating that a distributed approach can further increase the model performance by spreading computing resources across different agents.

B. Convergence Analysis

To investigate the convergence behaviors of proposed algorithms, we plotted the loss values of DFCM and DICR with each iteration of the optimization procedure, as shown in Fig. 2. In this and following subsections, we initialized the Laplacian parameter ρ_p and ρ_s at 0.1 and 0.1, and the parameters μ, γ, and η, which is corresponding to the ℓ_2 regularizer, the mix-up regularizer, and the graph-based regularizer, at 0.1, 0.1, and 0.0001, respectively. Due to space limitations, only the results for the Airfoil, CCPP, and KCH datasets are shown. Both methods converge quickly within 7 iterations for DFCM and 100 iterations or fewer for DICR.

C. Effects of regularization and ADMM parameters

As is convention, we assessed the influence of each parameter by fixing all other parameters using the initialization mentioned in Section V.B and varying the parameter in appropriate intervals: $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. The results of the parameter analysis is presented in Fig. 3.
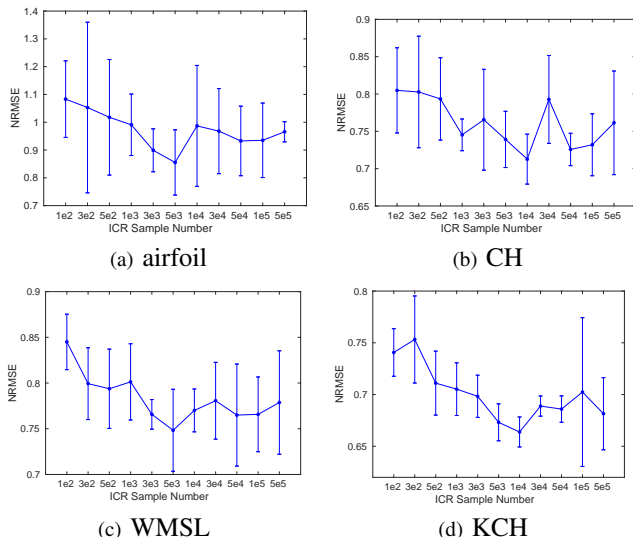
The variance in the ADMM parameter ρ_s at different values was very small. ρ_p performs better with values greater than 10^{-3}. Thus, we can conclude that ρ_s is robust at a wide range of values, but ρ_s performs better when choosing a relatively large value.

D. Effects of the number of interpolated unlabel samples

We also investigate how the number of interpolated unlabel samples affect the performance of DICR. After fixing the other parameters, we set the interpolated sample number within the interval $\{100, 300, 500, 1K, 3K, 5K, 10K, 30K, 50K, 100K, 500K\}$. The results are plotted in Fig. 4. All curves show a trend that declines in the beginning then climbs after reaching a minimum NRMSE. The best performance with the Airfoils and WMSL datasets was with 5K ICR MixUp samples. With CH and KCH, the best performance was at 10K samples. Hence, it is best to set the number of MixUp ICR samples near to the training sample numbers for each dataset.

E. Effects of the rule and agent number

The number of rules was varied in the interval of $\{2, 3, \cdots, 20\}$. As shown in Fig. 5, the more rules, the better the performance. Here, we randomly selected 500 labeled training samples and distributed them relatively evenly over different numbers of agents. As shown in Fig. 6, DSFR was able to stabilize with a range of agent numbers.

VI. CONCLUSION

In this paper, we proposed a novel DSFR model with fuzzy if-then rules and ICR. The result is a framework that handles uncertainty and also dramatically reduces computation and communication overheads in interconnected networks with multiple agents. Adopting the ADMM strategy, DSFR involves a DFCM for structure learning and a DICR for parameter learning. Notably, both algorithms can extract useful information from not only labeled samples but also unlabeled ones. The DICR algorithm increases model performance and robustness by restricting the decision boundaries to low-density regions. Comprehensive experiments on both artificial and real-world datasets show that DSFR excels in efficiency and effectiveness compared to the state-of-the-art algorithms. A deep variant of DSFR will be considered in future work to handle high-dimensional unlabeled data. This method will modify fuzzy logics to make them more suitable to high-dimensional applications. In addition, we will extend the current distributed learning methods to suit for more complex distributed scenarios.

Fig. 4. NRSME and its standard deviation when varying the number of ICR samples.

Fig. 5. NRSME and its standard deviation when varying the number of rules.
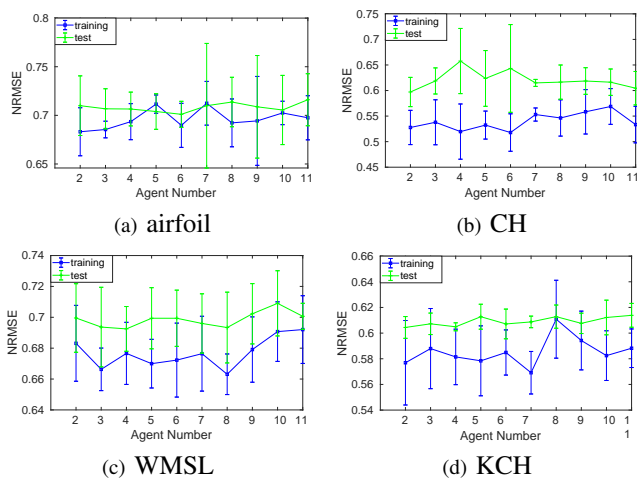

Fig. 6. NRSME and its standard deviation when varying the number of agents

In terms of the regularization parameters γ and μ, the NRMSE drops as parameter values increases, reaching its nadir at $\mu = 1$ with all datasets. This suggests that ICR plays a critical role in the model. The best result for μ, however, differs for every dataset. On Airfoil, CH and KCH datasets, μ decreases before reaching its bottom, then starts to climb again, which indicates that the proportion of the ℓ_2 term needs to be within a reasonable range.

REFERENCES

[1] X. J. Zhu, "Semi-supervised learning literature survey," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2005.
[2] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

[3] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 25–32, Association for Computational Linguistics, 2003.

[4] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models.," *WACV/MOTION*, vol. 2, 2005.

[5] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

[6] A. Anis, A. El Gamal, A. S. Avestimehr, and A. Ortega, "A sampling theory perspective of graph-based semi-supervised learning," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2322–2342, 2018.

[7] Y. Xu, H. Chen, J. Luo, Q. Zhang, S. Jiao, and X. Zhang, "Enhanced moth-flame optimizer with mutation strategy for global optimization," *Information Sciences*, vol. 492, pp. 181–203, 2019.

[8] J.-W. Jin and C. P. Chen, "Regularized robust broad learning system for uncertain data modeling," *Neurocomputing*, vol. 322, pp. 58–69, 2018.

[9] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[10] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," *arXiv preprint arXiv:1903.03825*, 2019.

[11] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[12] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 452–468, 2011.

[13] M. A. Pathak and B. Raj, "Privacy-preserving speaker verification and identification using gaussian mixture models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 397–406, 2012.

[14] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with hidden markov models," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5868–5871, IEEE, 2011.

[15] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, IEEE, 2017.

[16] H. Tuttle, "Facebook scandal raises data privacy concerns," *Risk Management*, vol. 65, no. 5, pp. 6–9, 2018.

[17] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2012.

[18] X. Chang, S.-B. Lin, and D.-X. Zhou, "Distributed semi-supervised learning with kernel ridge regression," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1493–1514, 2017.

[19] J. Xie, S. Liu, and H. Dai, "A distributed semi-supervised learning algorithm based on manifold regularization using wavelet neural network," *Neural Networks*, vol. 118, pp. 300–309, 2019.

[20] J. Xie, S. Liu, and H. Dai, "Distributed semi-supervised learning algorithm based on extreme learning machine over networks using event-triggered communication scheme," *Neural Networks*, vol. 119, pp. 261–272, 2019.

[21] 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.

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

[23] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, K.-S. Choi, and S. Wang, "Transfer prototype-based fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1210–1232, 2015.

[24] 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.

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

[26] 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*, 2019.

[27] Y. Yan, L. Chen, and W.-C. Tjhi, "Fuzzy semi-supervised co-clustering for text documents," *Fuzzy Sets and Systems*, vol. 215, pp. 74–89, 2013.

[28] S. Poria, A. Gelbukh, D. Das, and S. Bandyopadhyay, "Fuzzy clustering for semi-supervised learning–case study: Construction of an emotion lexicon," in *Mexican International Conference on Artificial Intelligence*, pp. 73–86, Springer, 2012.

[29] S. Zhou, Q. Chen, and X. Wang, "Fuzzy deep belief networks for semi-supervised sentiment classification," *Neurocomputing*, vol. 131, pp. 312–322, 2014.

[30] T. Zhang, Z. Deng, H. Ishibuchi, and L. M. Pang, "Robust tsk fuzzy system based on semi-supervised learning for label noise data," *IEEE Transactions on Fuzzy Systems*, 2020.

[31] Y. Jiang, D. Wu, Z. Deng, P. Qian, J. Wang, G. Wang, F.-L. Chung, K.-S. Choi, and S. Wang, "Seizure classification from EEG signals using transfer learning, semi-supervised learning and tsk fuzzy system," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2270–2284, 2017.

[32] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, pp. 5049–5059, 2019.

[33] S. P. Boyd, "Convex optimization: from embedded real-time to large-scale distributed.," in *KDD*, p. 1, 2011.

[34] 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.

[35] J. Qin, W. Fu, H. Gao, and W. X. Zheng, "Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory," *IEEE transactions on cybernetics*, vol. 47, no. 3, pp. 772–783, 2016.

[36] 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.

[37] 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.

[38] 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.

[39] S. Liu, Q. Qu, L. Chen, and L. M. Ni, "Smc: A practical schema for privacy-preserved data sharing over distributed data streams," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 68–81, 2015.

[40] M. Ye, L. Yin, G. Wen, and Y. Zheng, "On distributed nash equilibrium computation: hybrid games and a novel consensus-tracking perspective," *IEEE Transactions on Cybernetics*, 2020.

[41] M. Ye, "Distributed nash equilibrium seeking for games in systems with bounded control inputs," *IEEE Transactions on Automatic Control*, 2020.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] Y. Shi, C.-T. Lin, Y.-C. Chang, W. Ding, Y. Shi, and X. Yao, "Consensus learning for distributed fuzzy neural network in big data environment," *IEEE Transactions on Emerging Topics in Computational Intelligence*, p. to appear, 2020.

[47] B. Dang, Y. Wang, J. Zhou, R. Wang, L. Chen, C. P. Chen, T. Zhang, S. Han, L. Wang, and Y. Chen, "Transfer collaborative fuzzy clustering in distributed peer-to-peer networks," *IEEE Transactions on Fuzzy Systems*, 2020.

[48] Y. Jiang, Z. Deng, F.-L. Chung, and S. Wang, "Realizing two-view TSK fuzzy classification system by using collaborative learning," *IEEE transactions on systems, man, and cybernetics: systems*, vol. 47, no. 1, pp. 145–160, 2016.

[49] T. Zhang, Z. Deng, D. Wu, and S. Wang, "Multiview fuzzy logic system with the cooperation between visible and hidden views," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 6, pp. 1162–1173, 2018.

[50] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[51] S. Baluja, "Using labeled and unlabeled data for probabilistic modeling of face orientation," *International journal of pattern recognition and artificial intelligence*, vol. 14, no. 08, pp. 1097–1107, 2000.

[52] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," *arXiv preprint arXiv:1810.12281*, 2018.

[53] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[54] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, pp. 529–536, 2005.

[55] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, 2013.

[56] J. Xie, S. Liu, H. Dai, and Y. Rong, "Distributed semi-supervised learning algorithms for random vector functional-link networks with distributed data splitting across samples and features," *Knowledge-Based Systems*, p. 105577, 2020.

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

[58] 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.

[59] R. Lopez, E. Balsa-Canto, and E. Oñate, "Neural networks for variational problems in engineering," *International Journal for Numerical Methods in Engineering*, vol. 75, no. 11, pp. 1341–1360, 2008.

[60] J. J. Thompson, M. R. Blair, L. Chen, and A. J. Henrey, "Video game telemetry as a critical tool in the study of complex skill learning," *PloS one*, vol. 8, no. 9, p. e75129, 2013.

[61] 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.

[62] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[63] Y. Hu, Y. Zhao, Y. Cai, Z. Fu, and S. Ding, "Comparison of statistical learning and predictive models on breast cancer data and king county housing data," 2017.