Distributed Learning of Random Weights Fuzzy Neural Networks

Roberto Fierimonte, Marco Barbato, Antonello Rosato and Massimo Panella
Department of Information Engineering, Electronics and Telecommunications (DIET)
University of Rome "La Sapienza"
Via Eudossiana 18, 00184 Rome, Italy
Email: roberto.fierimonte@gmail.com; {marco.barbato, antonello.rosato, massimo.panella}@uniroma1.it

***Abstract*—In this paper, we propose a scalable, decentralized learning algorithm for Random Weights Fuzzy Neural Networks, when training data is distributed through a network of interconnected computing agents. In this scenario, the aim is for all the agents to converge to a single model, with the requirement that only local communications between the agents are permitted. In this work we assume that all the agents know the parameters of the antecedents, while the parameters of the consequents are estimated by using the Alternating Direction Method of Multipliers strategy. Experimental results show that the performance of the proposed algorithm is comparable to that of a centralized model, where all the data is collected by a single agent before the training process. To this date, this is the first publication that addressed the problem of training a fuzzy neural network over a fully decentralized infrastructure.**

I. Introduction

Fuzzy Neural Network (FNNs) are useful tools for the solution of supervised learning problems, as regression and classification, in many real-world applications relevant to physics, engineering, computer science, medicine, bioinformatics, econometrics, and so on. This paper deals with the learning of the Adaptive Neuro-Fuzzy Inference System (ANFIS) [1], which is based on a set of Sugeno first-order type rules. ANFIS networks are being widely used in many applications, such as rule-based process control, pattern recognition, function approximation, etc.

The classical training of ANFIS networks addresses two main problems: estimation of numerical parameters of antecedents and consequents for each fuzzy rule; regularization of the network complexity (i.e., estimation of the optimal number of rules) with the aim of maximizing the generalization capability of the FNN. Usually, ANFIS rules are determined by means of numerical datasets rather than linguistic rules provided by human experts. Such a process is commonly carried out by clustering the available data in the input or output space only or, more recently, by using techniques applied into the joint input-output space [2].

While the literature is plenty of training algorithms proposed in the last 25 years for FNNs, included ANFIS, regarding the classical case of datasets collected in a centralized computing node where the network architecture is implemented, no considerable contributions have been proposed so far regarding the case of distributed datasets. Precisely, we do not refer to the scaling of learning algorithms on a parallel/distributed computing architecture. Conversely, we are focusing on the problem of a set of agents (or nodes) distributed in the space, each with its own capability of computing and communicating, and having a local dataset acquired by a set associated sensors. Some examples in this regard are wireless sensor networks (WSN) [3], [4], swarm robots, trading systems on local financial markets and, more in general, mobile devices in pervasive computing applications [5].

This is an emerging scenario in the context of big data computing, where data cannot be collected in a centralized node or be transferred to all agents through the communication network. However, by means of a constrained amount of communication with the local neighborhood, each agent must be capable to solve the general regression problem represented by the complete dataset but using the local dataset only. For instance, a widely adopted approach to this problem is the well-known consensus strategy [6], by which all nodes try to converge to an average single model.

Several options are possible to the distributed learning of FNNs based on the ANFIS paradigm. A straightforward approach can be based on the application of the rules' synthesis, which should be based on distributed fuzzy clustering and distributed least-squares estimation/regularization. However, at this moment there are no consolidated approaches to distributed fuzzy clustering and this matter may be focused in future research works.

Alternatively, in this paper we propose a novel approach to solve the distributed learning problem by reformulating the ANFIS model as a Random Weights Fuzzy Neural Network (RWFNN). A RWFNN is a FNN based on the ANFIS architecture, in which the parameters of the membership functions are randomly selected instead of being estimated during the learning process [7]. It was proved that, if the parameters of the membership functions are kept fixed and only the consequent parameters are tuned, the resulting model is equivalent to a functional-link network, where the membership functions represent the functional expansion [1].

Following this approach, the learning of a RWFNN can be achieved by using well-known results of distributed learning methods, whose effectiveness has been already ascertained when applied to other types of neural networks and parametric models, as in the case of Random Vector Functional Links (RVFL) neural networks [8], [9]. Precisely, we will propose a novel approach where the RWFNN model is estimated through a network of interconnected computing agents and we will prove that the performance of the proposed algorithm is comparable to that of a centralized model, where all the data is collected by a single agent before of the training process.

The rest of the paper is organized as follows. In Sect. II we introduce the structure of RWFNNs. In Sect. III we formulate the problem of training a RWFNN over a network of agents, and we describe a fully decentralized learning algorithm for this purpose. In Sect. IV we present the experimental results obtained through numerical simulations on well-known datasets. Finally, in Sect. V we summarize our proposal and we discuss future works.

II. Random Weights Fuzzy Neural Networks

In this Section we introduce the RWFNN model and we describe its architecture. We show that the problem of training a RWFNN can be formulated as a Regularized Least Squares problem. A RWFNN is constituted by 5 feed forward layers, each layer is in turn constituted by a set of nodes and each node is associated with a fuzzy rule. Each node performs a particular operation on the signals coming from the previous layer, and sends the result of the calculation to the nodes in the next layer. There are no connections between nodes in the same layer.

Let us consider the problem of estimating a scalar output $y \in \mathbb{R}$ from a d-dimensional input $\mathbf{x} = [x_1, \dots, x_d]^{\mathrm{T}}$. Several alternatives are possible for the fuzzification of crisp inputs, the composition of input membership functions (MFs), and the way rule outputs are combined [10]. Let m be the predefined number of rules of the RWFNN network; usually, the structure of the fuzzy inference system can be summarized as follows.

- Layer 1. Every node i in this layer, $i = 1 \dots m$, is associated with an input MF $\mu_{(i,j)}(x_j, \boldsymbol{\alpha})$ operating on the jth dimension of the input vector $\mathbf{x}$ for the ith rule. The values for the parameters of the the antecedents $\boldsymbol{\alpha}$ are chosen are the beginning or the learning process from a fixed probability distribution, which is independent on the training data.
- Layer 2. Every node in the second layer corresponds to an *if-then* rule of the FIS. If the adopted operator for the logical AND is the algebraic product, then the output of the ith node is:
$$w_i(\mathbf{x}) = \prod_{j=1}^{d} \mu_{(i,j)}(x_j), \; i = 1 \dots m. \tag{1}$$
- Layer 3. Normalization:
$$\overline{w}_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{h=1}^{m} w_h(\mathbf{x})}, \; i = 1 \dots m. \tag{2}$$
- Layer 4. Local output of the ith rule:
$$\tilde{f}_i(\mathbf{x}) = \overline{w}_i(\mathbf{x})(\boldsymbol{\beta}_i^{\mathrm{T}} \mathbf{x}^+), \; i = 1 \dots m, \tag{3}$$
where $\boldsymbol{\beta}_i$ is the $(d+1)$-dimensional vector given by $\boldsymbol{\beta}_i = [\beta_{(i,0)}, \dots, \beta_{(i,d)}]^{\mathrm{T}}$ and $\mathbf{x}^+ = [1, \mathbf{x}]^{\mathrm{T}}$.
- Layer 5. This layer is constituted by a single node that computes overall output $\hat{y}$ as the sum of all the normalized firing strengths:
$$\hat{y} = \sum_{i=1}^{m} \tilde{f}_i \, . \tag{4}$$

Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ be the set of data available for the training phase, and denote with $x_{(r,j)}$ the jth component of the rth input vector. We define the hidden matrix $\mathbf{H} = \left[\mathbf{H}^1, \dots, \mathbf{H}^m\right]$, where

$$\mathbf{H}^i = \begin{bmatrix} \overline{w}_i & \overline{w}_i x_{(1,1)} & \cdots & \overline{w}_i x_{(1,d)} \\ \overline{w}_i & \overline{w}_i x_{(2,1)} & \cdots & \overline{w}_i x_{(2,d)} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{w}_i & \overline{w}_i x_{(n,1)} & \cdots & \overline{w}_i x_{(n,d)} \end{bmatrix}, \; i = 1 \dots m. \tag{5}$$

and the output vector $\mathbf{y} = [y_1, \dots, y_n]^{\mathrm{T}}$. Rearranging the parameters $\boldsymbol{\beta}$ in the form:

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_{(1,0)} \cdots \boldsymbol{\beta}_{(1,d)} \boldsymbol{\beta}_{(2,0)} \cdots \boldsymbol{\beta}_{(2,d)} \cdots \boldsymbol{\beta}_{(m,0)} \cdots \boldsymbol{\beta}_{(m,d)}]^{\mathrm{T}}, \tag{6}$$

the optimization problem for training a RWFNN can be reformulated as a Least-Squares (LS) problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \left\| \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \right\|_2^2 , \tag{7}$$

where $\|\cdot\|_2$ is the l^2 norm and $p = m(d+1)$. The formulation in (7) may suffer of numerical instability due to the possible small values of $\overline{w}_i$, additionally, the optimal solution is undetermined if $n < p$. For this reason, we modify the optimization problem by setting it in the form of a Regularized Least-Squares (RLS) problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \left\| \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|_2^2 \tag{8}$$

where $\lambda > 0$ is the *regularization factor*. The new problem (8) is strictly convex, and then it has a unique solution that can be obtained in closed form as:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda \mathbf{I}\right)^{-1} \mathbf{H}^{\mathrm{T}} \mathbf{y} \, . \tag{9}$$

III. Distributed learning for RWFNN

A. Formulation of the problem

When training a RWFNN in a distributed setting, we consider the dataset $\mathcal{T}$ distributed over a network of L interconnected agents [3], [8], [11]. The network of agents can be modeled as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, L\}$ is the set of the agents, and $\mathcal{E}$ is the set of the edges. The connectivity of the graph is fixed and known a priori and can be formalized as a $L \times L$ weights matrix $\mathbf{W}$, where $W_{ij} \neq 0$ if and only if agents i and j are connected. A possible strategy to choose these weights will be discussed in the following. In this paper, we focus on the simplest case that represents connected and undirected topologies. We impose three additional constraints in the design of our algorithm:

- no agent is allowed to coordinate the training process;
- agents are not allowed to exchange any data pattern;
- only local communication between connected agents is possible.

These constraints are not restrictive and make our proposal suitable to be employed in different applications. Let $\mathcal{T}_k = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_k}, y_{n_k})\}$ the training set available at the kth agent, such that $\cup_{k=1}^{L} \mathcal{T}_k = \mathcal{T}$. In this case, the optimization problem in (8) can be reformulated in a distributed fashion as:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \left(\sum_{k=1}^{L} \left\| \mathbf{y}_k - \mathbf{H}_k \boldsymbol{\beta}_k \right\|_2^2 \right) + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|_2^2 , \tag{10}$$

where $\mathbf{H}_k$ and $\mathbf{y}_k$ are the hidden matrix and the output vector computed over the training set $\mathcal{T}_k$.

B. ADMM-based training of RWFNN

The proposed algorithm to train a RWFNN over a network of agents is based on the well-known Alternating Direction Method of Multipliers (ADMM) [12]. In particular, we follow the derivation for a distributed Least-Squares problem. As the first step, we rewrite the problem in (10) introducing a global variable $\mathbf{z} \in \mathbb{R}^p$, and forcing the local variables $\boldsymbol{\beta}_k$ to be equal at convergence. The new problem has the form of:

$$\begin{aligned} \boldsymbol{\beta}^* = \underset{\mathbf{z},\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_L \in \mathbb{R}^p}{\arg\min} \; & \frac{1}{2}\left(\sum_{k=1}^{L} \|\mathbf{y}_k - \mathbf{H}_k\boldsymbol{\beta}_k\|_2^2\right) + \frac{\lambda}{2}\|\mathbf{z}\|_2^2 \\ & \text{subject to}\, \boldsymbol{\beta}_k = \mathbf{z},\ \forall k = 1,\ldots L\,. \end{aligned} \tag{11}$$

Then we construct the augmented Lagrangian for the new problem:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}\left(\sum_{k=1}^{L} \|\mathbf{y_k} - \mathbf{H}_k\boldsymbol{\beta}_k\|_2^2\right) + \frac{\lambda}{2}\|\mathbf{z}\|_2^2 \\ & + \sum_{k=1}^{L} \mathbf{t}_k^{\mathrm{T}}\left(\boldsymbol{\beta}_k - \mathbf{z}\right) + \frac{\rho}{2}\sum_{k=1}^{L}\|\boldsymbol{\beta}_k - \mathbf{z}\|_2^2\,, \end{aligned} \tag{12}$$

where $\mathbf{t}_k$, $k = 1,\ldots,L$ are the Lagrange multipliers and $\rho > 0$ is a penalty parameter. ADMM solves the problem in (11) iteratively. Each iteration of the algorithm consists in alternating a minimization step for the variables $\boldsymbol{\beta}_k$ and $\mathbf{z}$ with an update step for the Lagrange multipliers $\mathbf{t}_k$. In our application, this steps can be obtained in closed form:

$$\boldsymbol{\beta}_k[n+1] = \left(\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k + \rho\mathbf{I}\right)^{-1}\left(\mathbf{H}_k^{\mathrm{T}}\mathbf{y}_k - \mathbf{t}_k[n] + \rho\mathbf{z}[n]\right)\,, \tag{13}$$

$$\mathbf{z}[n+1] = \frac{\rho\overline{\boldsymbol{\beta}} + \overline{\mathbf{t}}}{\lambda/L + \rho}\,, \tag{14}$$

$$\mathbf{t}[n+1] = \mathbf{t}[n] + \rho\left(\boldsymbol{\beta}_k[n+1] - \mathbf{z}[n+1]\right)\,, \tag{15}$$

where $\overline{\boldsymbol{\beta}}$ and $\overline{\mathbf{t}}$ are the averages computed over $\boldsymbol{\beta}_k[n+1]$ and $\mathbf{t}_k[n+1]$, respectively. These averages can be computed over the network using the Distributed Average Consensus (DAC) protocol [4] or a push-sum protocol [13] or a number of alternative techniques dependent on the network's architecture.

C. Early stopping

The convergence behavior of the algorithm at the nth iteration can be studied for all the agents by analyzing the 'primal residual' $\mathbf{r}_k[n]$ and the 'dual residual' $\mathbf{s}[n]$, defined by:

$$\mathbf{r}_k[n] = \boldsymbol{\beta}_k[n] - \mathbf{z}[n]\,, \tag{16}$$

$$\mathbf{s}[n] = -\rho\left(\mathbf{z}[n] - \mathbf{z}[n-1]\right)\,. \tag{17}$$

A stopping criterion for the algorithm consists in verifying if for all the agents, the norms of the residuals are less than two thresholds:

$$\|\mathbf{r}_k[n]\|_2 < \epsilon_{\text{primal}}(k)\,, \tag{18}$$

$$\|\mathbf{s}[n]\|_2 < \epsilon_{\text{dual}}\,. \tag{19}$$

A possible strategy for choosing the values of the thresholds is given by [12]:

$$\epsilon_{\text{primal}}(k) = \sqrt{L}\epsilon_{\text{abs}} + \epsilon_{\text{rel}}\max\{\|\boldsymbol{\beta}_k[n]\|_2\,, \mathbf{z}[n]\} \tag{20}$$

$$\epsilon_{\text{dual}} = \sqrt{L}\epsilon_{\text{abs}} + \epsilon_{\text{rel}}\max_k\{\|\mathbf{t}_k\|_2\} \tag{21}$$

where ϵ_{abs} and ϵ_{rel} are the values for the absolute and relative tolerance respectively. If the stopping criterion is not satisfied after a maximum number of iterations $N_{\max}$, the algorithm is terminated. The pseudocode for the proposed algorithm at agent k is given in Table I.

TABLE I. PSEUDOCODE OF THE PROPOSED DISTRIBUTED ALGORITHM AT THE kTH AGENT

```
Input: Training set $\mathcal{T}_k$, number of nodes $L$ (global), regularization parameters $\lambda$, $\rho$ (global), absolute $\epsilon_{\text{abs}}$ and relative $\epsilon_{\text{rel}}$ tolerances, maximum number of iterations $N_{\max}$
Output: Optimal vector $\boldsymbol{\beta}_k^*$
 1: Generate the parameters of the membership functions, in accordance with the other $L-1$ agents.
 2: Initialize $\mathbf{t}_k[0] = 0, \mathbf{z}[0] = 0$.
 3: for $n = 0$ to $N_{\max}$ do
 4:     Compute $\boldsymbol{\beta}_k[n]$ according to Eq. (13).
 5:     Compute $\overline{\boldsymbol{\beta}}$ and $\overline{\mathbf{t}}$ using the DAC protocol.
 6:     Compute $\mathbf{z}[n+1]$ according to Eq. (14).
 7:     Update $\mathbf{t}_k[n+1]$ according to Eq. (15).
 8:     Check the stopping criterion.
 9: end for
10: return $\mathbf{z}[n+1]$
```

IV. EXPERIMENTAL RESULTS

A. Description of datasets

To validate our proposal we employ three widespread datasets extensively used for regression purposes, available from the UCI Machine Learning Repository[1]. The datasets were carefully chosen to represent different applicative domains and, at the same time, they reveal a varied composition in terms of number of instances and features. A brief summary of the datasets is given in Table II; below we provide a more detailed description for each of them.

- *Airfoil* - This is a NASA dataset [14] composed of different size NACA 0012 airfoils subject to aero-dynamical and acoustical measurements. The attributes represent the stream velocity, the angle of attack and the features of the airfoil. The desired output is the sound pressure level.
- *Critical Assessment of protein Structure Prediction (CASP)* - This dataset describes the physicochemical properties of a Protein Tertiary Structure, using values taken from the CASP experiments[2]. The number of

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

TABLE II. GENERAL DESCRIPTION OF THE DATASETS.

Dataset name	Features	Instances	Desired Output
Airfoil	5	1503	Sound pressure level
CASP	9	45730	RMSD
CCPP	4	9568	Electrical energy

instances corresponds to the amount of measured decoys, while the attributes pertain to their inner structures and exposed areas. The output is the Root Mean Square Deviation (RMSD) of the residuals.

- *Combined Cycle Power Plant (CCPP)* - This dataset consists of a number of measurements collected from a power plant over a six-years time interval [15]. The attributes refer to four different ambient variables, while the output is the predicted electrical energy output of the power plant. Interestingly enough, the measures are collected using a sensor network.

In all cases, we normalize the input variables between -1 and $+1$ before the experiments. We evaluate the accuracy of all the models by applying a 10-fold cross validation procedure on the datasets. The procedure is repeated 10 times by varying the initialization of the weights of the models and the topology of the network. The final values for the prediction error and the training time are averaged over the total 100 repetitions.

B. Implementation

We have implemented the proposed algorithm in an open source MATLAB toolbox[3]. In this paper we are not concerned about investigating the effects of the communication process over a real network, hence we provide a serial version of the algorithm. All the simulations were performed using MATLAB 2015a running on an Intel i7 @ 3.40 GHz processor and 16 GB of memory. In our experiments we compare the following models:

- **ANFIS**: this is an ANFIS network trained using the MATLAB Fuzzy Logic Toolbox, according to the well-known hybrid algorithm described in [1]. It provides a benchmark to assess the performance of the RWFNN model in the classical single agent case. In our experiments we set a maximum of 100 epochs for the training algorithm.
- **C-RWFNN**: this is the algorithm depicted in Sect. II. It corresponds to a single agent collecting all the data before the training process, and it can be interpreted as a baseline for the decentralized approach.
- **ADMM-RWFNN**: in this case, the training set is distributed across the agents, and the agents run the algorithm described in Sect. III. In all the simulations we set $N_{\text{max}} = 600, \epsilon_{\text{rel}} = \epsilon_{\text{abs}} = 10^{-3}$ and $\rho = 1$.
- **L-RWFNN**: as before, the training data is distributed through the network. Every agent trains a RWFNN using its own data points, but no information is exchanged. The value for the test error is averaged over the agents.

[3]https://bitbucket.org/robertofierimonte/code-distributed-rwfnn

TABLE III. OPTIMAL VALUES FOR THE REGULARIZATION PARAMETER λ.

Dataset name	λ
Airfoil	10^{-1}
CASP	10^{-3}
CCPP	10^{-2}

For all the datasets we construct the FIS by defining two linguistic labels for each feature and we use Gaussian membership functions given by:

$$\mu(x; c, \sigma) = \exp\left\{\frac{-(x-c)^2}{2\sigma^2}\right\}, \tag{22}$$

where the mean c is extracted from the uniform distribution over the interval $[-1, +1]$ and the standard deviation σ is extracted from the uniform distribution over $[0, 2]$. The same FIS is used as the starting point to train the ANFIS model. The optimal value for the regularization parameter λ is obtained by running 5 runs of 5-fold inner cross validation of C-RWFNN on the training data only. We search the value for the parameter in the discrete exponential interval $10^j, j \in \{-3, -2, -1, 0, 1, 2\}$, and we share the optimal value with ADMM-RWFNN and L-RWFNN. The chosen values of λ are showed in Table III.

C. Results and discussion

The first set of experiments consists in assessing the performance of C-RWFNN when compared to ANFIS. The aim of this comparison is to show that the considered architecture is able to match the performance of well-known and assessed fuzzy systems. We evaluate the accuracy of all the models using the Normalized Root Mean-Squared Error (NRMSE), defined by:

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n\hat{\sigma}_y}}, \tag{23}$$

where $\hat{\sigma}_y$ is the estimated variance of $\{y_1, \ldots, y_n\}$. The results of this set of experiments are shown in Table IV.

We can see that in all cases, C-RWFNN achieves only a slightly higher error when compared to ANFIS, ranging from an additional 0.03 for the CASP dataset to 0.04 for the Airfoil dataset, and resulting in approximately the same values for the CCPP dataset. On the opposite, a comparison of the computational time required by the two models shows a high disproportion in favor of C-RWFNN, which requires an amount of time approximately 4 orders of magnitude lower with respect to ANFIS. An analysis of this results suggests that when the structure of the FIS is fixed, a fine tuning of the membership functions' parameters leads to an extremely high growth in the computational complexity of the model, which is not adequately compensated by the improvement in the performance.

TABLE IV. AVERAGE VALUES FOR TEST ERROR AND COMPUTATIONAL TIME, TOGETHER WITH STANDARD DEVIATION, FOR THE CENTRALIZED MODELS. RESULTS FOR C-RWFNN ARE HIGHLIGHTED IN BOLD.

Dataset	Algorithm	Test NRMSE	Training Time [s]
Airfoil	**C-RWFNN**	$\mathbf{0.253 \pm 0.021}$	$\mathbf{0.04 \pm 0.01}$
	ANFIS	0.215 ± 0.01	70.67 ± 26.72
CASP	**C-RWFNN**	$\mathbf{0.147 \pm 0.040}$	$\mathbf{85.41 \pm 29.19}$
	ANFIS	0.117 ± 0.003	2742.13 ± 956.22
CCPP	**C-RWFNN**	$\mathbf{0.071 \pm 0.013}$	$\mathbf{0.06 \pm 0.02}$
	ANFIS	0.071 ± 0.01	70.83 ± 5.75

A second set of experiments is to show the accuracy and the computational time of the proposed algorithm when varying the number of agents in the network from $L = 5$ to $L = 25$ by steps of 5. In this experiment we construct the network according to the so-called 'Erdős−Rényi model' [16], such that every pair of agents has a 25% probability to be connected. The only constraint is that the overall network is connected. Additionally, we choose the weights $\mathbf{W}$ using the *Metropolis-Hastings* strategy [17]:

$$W_{kj} = \begin{cases} \dfrac{1}{\max\{d_k, d_j\} + 1} & k \neq j,\ \{k,j\} \in \mathcal{E} \\ 1 - \sum_{j \in \mathcal{N}_k} \dfrac{1}{\max\{d_k, d_j\} + 1} & k = j \\ 0 & k \neq j,\ \{k,j\} \notin \mathcal{E} \end{cases} \tag{24}$$

where $\mathcal{N}_k$ is the set of agents' indexes directly connected to agent k and $d_k = |\mathcal{N}_k \backslash k|$, with $|\cdot|$ denoting the cardinality of a set. This choice of the weights ensures convergence for the DAC protocol [18], and therefore for ADMM-RWFNN. The results for this experiment are showed in Fig. 1, while a summary of the results for $L = 25$ is given in Table V. The panels on the left show the trend of the test error, while the panels on the right show the trend of the training time. As mentioned before, the test error for L-RWFNN is averaged over the agents.

We start our discussion observing that in all cases, L-RWFNN fails in tracking the performance of the centralized model as the number of agents increases. Particularly poor results are obtained on Airfoil (Fig. 1a) and CASP (Fig. 1c) datasets, with an additional error of 0.4 and 0.07, respectively. It is interesting to note that the evolution of the error, when varying the number of agents, results in different trends for the three datasets: in fact for the CASP dataset the gap between L-RWFNN and C-RWFNN remains approximately constant for any size of the network. Additionally we highlight that the local model suffers of high variance, due to the different results achieved by the agents.

The second important aspect to highlight is that ADMM-RWFNN is able to track the error of C-RWFNN regardless of the number of agents in the network. In particular the two models have the same performance on the Airfoil and CCPP datasets, while the distributed algorithm achieves a slightly lower error on the CASP dataset. The unusually high variance achieved by ADMM-RWFNN on the CCPP dataset (Fig. 1e) is

TABLE V. AVERAGE VALUES FOR THE TEST ERROR AND COMPUTATIONAL TIME, TOGETHER WITH STANDARD DEVIATION, FOR C-RWFNN, L-RWFNN AND ADMM-RWFNN ON A 25-AGENT NETWORK. TRAINING TIME FOR L-RWFN AND ADMM-RWFNN IS AVERAGED OVER THE NODES. RESULTS FOR THE PROPOSED ALGORITHM ARE HIGHLIGHTED IN BOLD.

Dataset	Algorithm	Test NRMSE	Training time [s]
Airfoil	C-RWFNN	0.253 ± 0.021	0.04 ± 0.01
	L-RWFNN (25 nodes)	0.645 ± 0.158	0.02 ± 0.01
	ADMM-RWFNN (25 nodes)	$\mathbf{0.254 \pm 0.021}$	$\mathbf{0.45 \pm 0.10}$
CASP	C-RWFNN	0.147 ± 0.040	85.41 ± 29.19
	L-RWFNN (25 nodes)	0.219 ± 0.093	7.56 ± 1.86
	ADMM-RWFNN (25 nodes)	$\mathbf{0.127 \pm 0.02}$	$\mathbf{26.96 \pm 6.99}$
CCPP	C-RWFNN	0.071 ± 0.013	0.06 ± 0.02
	L-RWFNN (25 nodes)	0.088 ± 0.024	0.05 ± 0.01
	ADMM-RWFNN (25 nodes)	$\mathbf{0.072 \pm 0.014}$	$\mathbf{0.1 \pm 0.03}$

a consequence to the intrinsic variability of C-RWFNN model, rather than being due to a poor performance of the distributed algorithm.

When considering the computational time required by the three models, there are few considerations to be made. We start with noticing that for all the datasets, the training time required by L-RWFNN is lower than the training time required by C-RWFNN, and it remains approximately constant for any size of the network. This an obvious consequence of the fact that a smaller dataset is used for the training phase. On the opposite the training time required by ADMM-RWFNN is strongly dependent on the size of the network and, in general, it shows a decreasing trend as the number of agents increases. It is also interesting to notice how, for a large dataset like CASP, ADMM-RWFNN is always faster than C-RWFNN, resulting in a reduction in training time of 60% when $L = 25$.

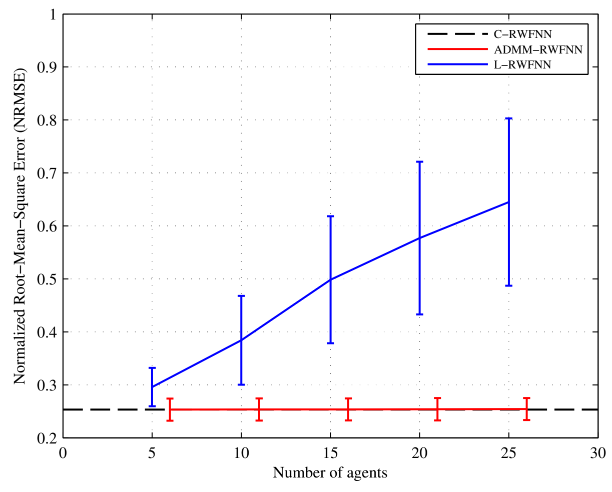
Overall, the experimental results show that the effectiveness of ADMM-RWFNN over C-RWFNN increases according to the size of the network, alongside with the enlargement of the dataset, making our proposal particularly interesting for big data and large scale applications.

V. CONCLUSIONS

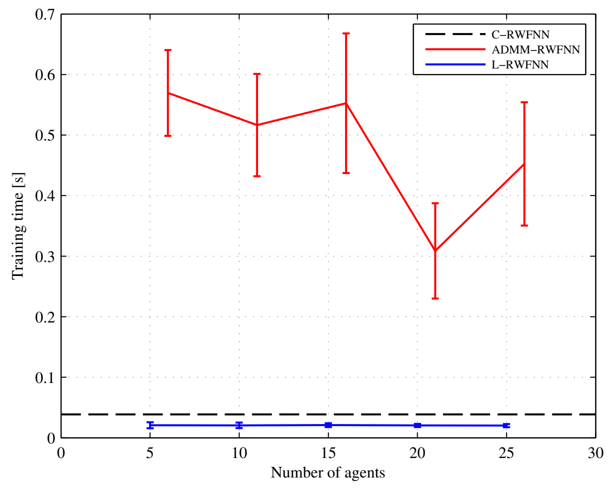
In this paper, we have proposed a completely decentralized algorithm for training a FNN architecture when the training data is distributed over a network of computing agents. To tackle this problem, we have considered RWFNNs as a particular instance of ANFIS networks and, by using an ADMM-based distributed learning technique, we have proposed a novel algorithm to estimate the RWFNN model through a network of interconnected agents.

The numerical results obtained by experimental simulations on well-known datasets prove that the performance of the proposed algorithm is comparable to that of a centralized model, where all the data is collected by a single agent before the training process.

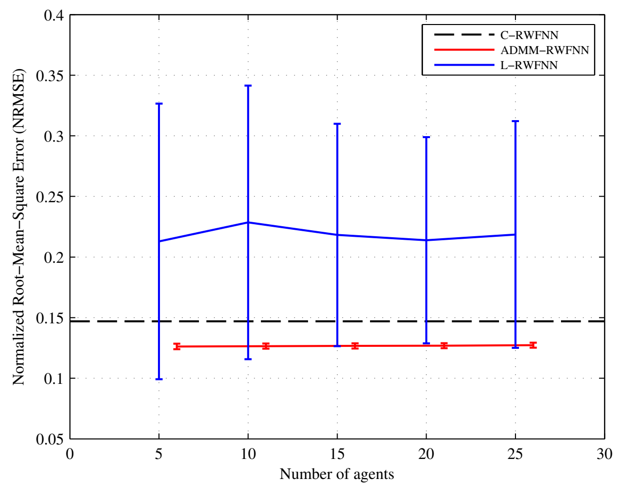
In future works, we aim at extending the distributed approach to different and more complex FNN architectures,



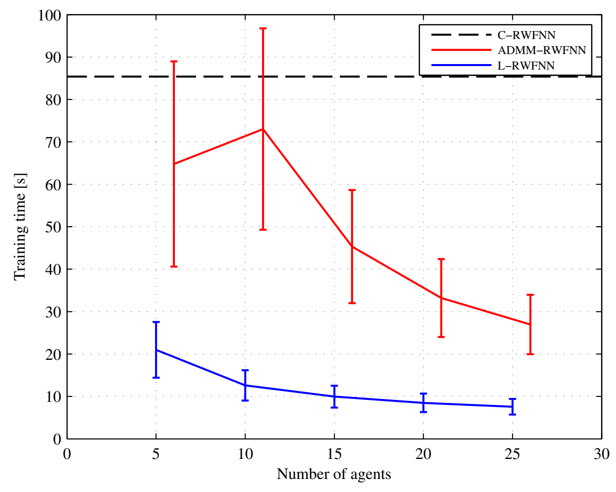
(a) Test error (Dataset Airfoil)



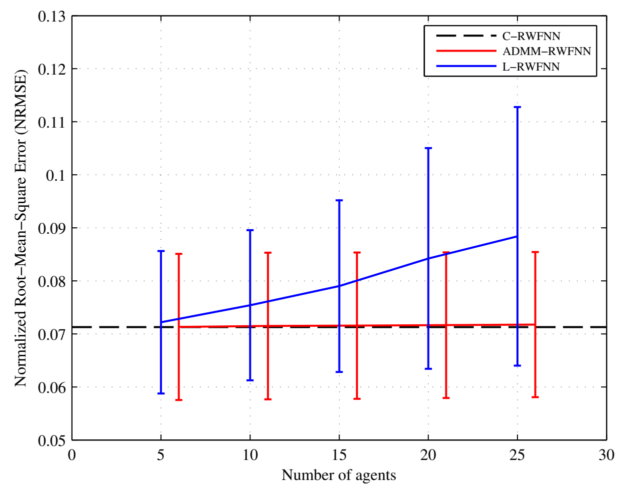
(b) Training time (Dataset Airfoil)



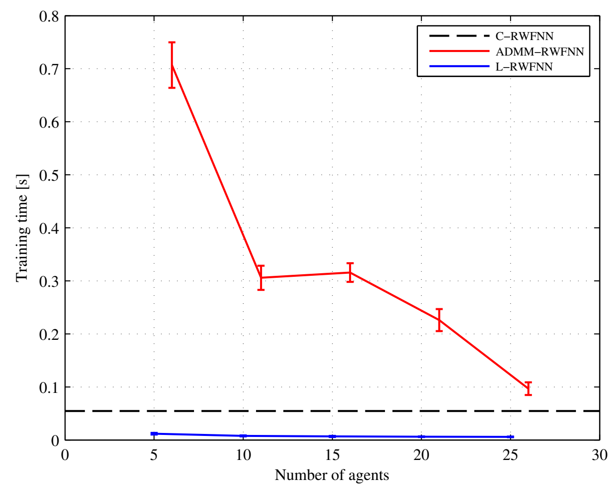
(c) Test error (Dataset CASP)



(d) Training time (Dataset CASP)



(e) Test error (Dataset CCPP)



(f) Training time (Dataset CCPP)

Fig. 1. Performance of ADMM-RWFNN and L-RWFNN, compared to C-RWFNN, for different number of agents. The panels on the left show the test error, while the panels on the right show the training time. Vertical bars represent the standard deviation from the mean value. Lines for ADMM-RWFNN are slightly shifted to increase readability.

supported by the latest advances in non-convex optimization frameworks over networks [19], [20] and distributed clustering techniques for rules' synthesis by numerical datasets [21].

REFERENCES

[1] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[2] M. Panella and A. S. Gallo, "An input-output clustering approach to the synthesis of ANFIS networks," *IEEE Trans. Fuzzy Syst.*, vol. 13, pp. 69–81, 2005.

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

[4] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Distributed detection and estimation in wireless sensor networks," in *E-Reference Signal Processing*, R. Chellapa and S. Theodoridis, Eds. Elsevier, 2013, pp. 329–408.

[5] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[6] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[7] Y.-L. He, X.-Z. Wang, and J. Z. Huang, "Fuzzy nonlinear regression analysis using a random weight network," *Information Sciences*, 2016, in press.

[8] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed Learning with Random Vector Functional-Link Networks," *Information Sciences*, vol. 301, pp. 271–284, 2015.

[9] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, "Distributed music classification using random vector functional-link nets," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[10] J.-S. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[11] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, 2015.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "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.

[13] C. Hensel and H. Dutta, "Gadget svm: a gossip-based sub-gradient svm solver," in *International Conference on Machine Learning (ICML), Numerical Mathematics in Machine Learning Workshop*, 2009.

[14] T. F. Brooks, D. S. Pope, and M. A. Marcolini, *Airfoil self-noise and prediction*. National Aeronautics and Space Administration, Office of Management, Scientific and Technical Information Division, 1989, vol. 1218.

[15] 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.

[16] M. Newman, *Networks: an introduction*. Oxford University Press, 2010.

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

[18] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[19] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *Automatic Control, IEEE Transactions on*, vol. 58, no. 2, pp. 391–405, 2013.

[20] P. Di Lorenzo and G. Scutari, "NEXT: In-Network Nonconvex Optimization," *IEEE Transactions on Signal and Information Processing over Networks*, 2016, in press.

[21] A. Rosato, R. Altilio, and M. Panella, *Recent Advances on Distributed Unsupervised Learning*, ser. Smart Innovation, Systems and Technologies. Springer International Publishing, 2016, in press.