A Comparison of Consensus Strategies for Distributed Learning of Random Vector Functional-Link Networks

Roberto Fierimonte, Simone Scardapane, Massimo Panella and Aurelio Uncini

Abstract Distributed machine learning is the problem of inferring a desired relation when the training data is distributed throughout a network of agents (e.g. robots in a robot swarm). Multiple families of distributed learning algorithms are based on the decentralized average consensus (DAC) protocol, an efficient algorithm for computing an average starting from local measurement vectors. The performance of DAC, however, is strongly dependent on the choice of a weighting matrix associated to the network. In this paper, we perform a comparative analysis of the relative performance of 4 different strategies for choosing the weighting matrix. As an applicative example, we consider the distributed sequential algorithm for Random Vector Functional-Link networks. As expected, our experimental simulations show that the training time required by the algorithm is drastically reduced when considering a proper initialization of the weights.

Keywords Consensus · Distributed machine learning · Random vector functional-link

R. Fierimonte · S. Scardapane (✉) · M. Panella · A. Uncini
Department of Information Engineering Electronics
and Telecommunications (DIET), "Sapienza" University of Rome,
Via Eudossiana 18, 00184 Rome, Italy
e-mail: simone.scardapane@uniroma1.it

R. Fierimonte
e-mail: roberto.fierimonte@gmail.com

M. Panella
e-mail: massimo.panella@uniroma1.it

A. Uncini
e-mail: aurel@ieee.org

S. Bassis et al. (eds.), *Advances in Neural Networks*, Smart Innovation,
Systems and Technologies 54, DOI 10.1007/978-3-319-33747-0_14

1 Introduction

In the last decade, the problem of distributed machine learning, i.e. the problem of learning by a set of data originated in a distributed system, has become an extensive reserched topic, since a large number of real-world applications can be modeled in the form of a distributed learning problem (e.g. inference in Wireless Sensor Networks [2], decentralized databases and datacenters [1], music classification over P2P networks [10], and several others).

Different algorithms were developed to deal with the problem, including works on distributed support vector machines [4], distributed neural networks [5, 11], and applications of distributed optimization techniques [2]. One recently developed strategy is the 'learning by consensus' (LBC) strategy [5]. LBC allows to transform any centralized iterative learning algorithm into a fully distributed protocol. It consists in the alternative application of local update rules and distributed averaging steps, implemented via the distributed average consensus (DAC) protocol [6, 7]. The use of DAC allows to obtain a fully decentralized algorithm, without the need for a fusion center, and with only local communication between neighboring nodes. In [10], we extended the LBC framework to the decentralized sequential setting, where data arrives continuously at every node. In particular, we presented a fully distributed, sequential algorithm for a particular class of neural networks known as Random Vector Functional-Links (RVFLs) [8]. RVFLs are composed of a fixed layer of non-linearities, followed by an adaptable linear layer. Due to this, the resulting training algorithms can be formulated in the form of linear regression problems, with strong capabilities of scaling to large training sets, as demonstrated in several previous works [11].

The DAC protocol computes the average in an iterative fashion, by locally weighting the estimations of each neighbor at every node. Hence, its convergence behavior is strongly dependent on the particular choice of mixing parameters. Among the many choices that guarantee global convergence, several strategies for choosing them have been proposed in the literature, each with its own asymptotic behavior and computational requirements [12]. Currently, a thorough analysis and comparison on this topic is missing. In this paper, we begin this investigation by comparing the performance of 4 different strategies for the DAC protocol, using the algorithm presented in [10] as a benchmark application. The strategies that we analyze vary from choosing a fixed value for every coefficient, to more complex choices satisfying strong optimality conditions. Our experimental results show that the performance of the DAC protocol, and by consequence the performance of any distributed training algorithm based on its application, can improve significantly with proper choices of the mixing parameters.

The rest of the paper is organized as follows. In Sect. 2 we present the DAC protocol, together with the 4 strategies that we investigate. Then, in Sect. 4 we briefly describe the distributed algorithm for RVFLs presented in [10]. Experimental results are then provided in Sect. 5, while Sect. 6 concludes the paper.

2 Decentralized Average Consensus

Distributed average consensus, or simply Consensus, is a totally distributed iterative protocol designed to compute the average of a measurements vector within a network. We assume the network in the form of a graph $G(V, E)$ with nodes V and edges E, wherein connectivity is known a priori and can be expressed in the form of an adjacency matrix $\mathbf{A}$ whose elements are such that:

$$A_{ij} = \begin{cases} 1 & \{i,j\} \in E \\ 0 & \{i,j\} \notin E \end{cases} . \tag{1}$$

For the sake of simplicity, we consider only connected graphs, i.e. graphs where exists a directed path between each pair of nodes $u, v \in V$. Let $\boldsymbol{\beta}_i(t)$ be the vector of measurements associated with the ith node of the network at instant t, and $N = |V|$, the task of the protocol is for all the nodes to converge to the average of the initial values of the measurements:

$$\hat{\boldsymbol{\beta}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\beta}_i(0) . \tag{2}$$

For discrete-time distributed systems the DAC protocol is defined by a set of linear updating equations in the form of:

$$\boldsymbol{\beta}_i(t+1) = \sum_{j=1}^{N} w_{ij} \boldsymbol{\beta}_j(t) , \tag{3}$$

which can be reformulated compactly as a linear system:

$$\boldsymbol{\beta}(t+1) = \mathbf{W}\boldsymbol{\beta}(t) . \tag{4}$$

The matrix $\mathbf{W}$ is named weights matrix, and the value of its generic element w_{ij} denotes the strength of the connection between nodes i and j, or alternatively, the confidence that node i assigns to the information coming from node j. We denote with $\mathscr{W}$ the set of the admissible weights matrices:

$$\mathscr{W} = \{\mathbf{W} \in \mathbb{R}^{N \times N} : w_{ij} = 0 \quad \text{if} \quad i \neq j, \{i,j\} \notin E\} . \tag{5}$$

For suitable choices of the weights matrix $\mathbf{W} \in \mathscr{W}$, the iterations defined in Eq. (4) converge locally to the average (2). A large number of modifications for the basic DAC protocol detailed here were proposed in literature, including the case where the agreement is a weighted average of the initial values [2], and applications to problems with dynamic topologies and time delays [7]. In this paper, we focus on the simplest case, which represents fixed, undirected network topologies.

3 Consensus Strategies

Different strategies for the DAC protocol correspond to different choices of the weights matrix. Clearly, the choice of a particular weight matrix depends on the available information at every node about the network topology, and on their specific computational requirements.

3.1 Max-Degree

The first strategy that we consider is the max-degree weights matrix, which is a common choice in real-world applications, and is defined entry-wise by:

$$w_{ij} = \begin{cases} 1/(d+1) & i \neq j, \{i,j\} \in E \\ 1 - d_i/(d+1) & i = j \\ 0 & i \neq j, \{i,j\} \notin E \end{cases}, \tag{6}$$

where d_i is the degree of the ith node, and d is the maximum degree of the network.

3.2 Metropolis-Hastings

An even simpler choice is the Metropolis-Hastings weights matrix:

$$w_{ij} = \begin{cases} 1/(\max\{d_i, d_j\} + 1) & i \neq j, \{i,j\} \in E \\ 1 - \sum_{j \in \mathcal{N}_i} 1/(\max\{d_i, d_j\} + 1) & i = j \\ 0 & i \neq j, \{i,j\} \notin E \end{cases}, \tag{7}$$

where $\mathcal{N}_i$ is the set of nodes' indexes directly connected to node i. Differently from the max-degree strategy, the Metropolis-Hastings strategy does not require the knowledge of global information (the maximum degree) about the network topology, but requires that each node knows the degrees of all its neighbors.

3.3 Minimum Asymptotic

The third matrix strategy considered here corresponds to the optimal strategy introduced in [12], wherein the weights matrix is constructed to minimize the asymptotic

convergence factor $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/N)$, where $\rho(\cdot)$ denotes the spectral radius operator. This is achieved by solving the constrained optimization problem:

$$\begin{aligned} &\text{minimize} \quad \rho(\mathbf{W} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/N) \\ &\text{subject to} \quad \mathbf{W} \in \mathscr{W}, \quad \mathbf{1}^{\mathrm{T}}\mathbf{W} = \mathbf{1}^{\mathrm{T}}, \quad \mathbf{W}\mathbf{1} = \mathbf{1} \end{aligned} \tag{8}$$

Problem (8) is non-convex, but it can be shown to be equivalent to a semidefinite programming (SDP) problem [12], solvable using efficient ad-hoc algorithms.

3.4 Laplacian Heuristic

The fourth and last matrix considered in this work is an heuristic approach [12] based on constant edge weights matrix:

$$\mathbf{W} = \mathbf{I} - \alpha\mathbf{L}\,, \tag{9}$$

where $\alpha \in \mathbb{R}$ is a user-defined parameter, and $\mathbf{L}$ is the Laplacian matrix associated to the network [2]. For weights matrices in the form of (9), the asymptotic convergence factor satisfies:

$$\begin{aligned} \rho(\mathbf{W} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/N) &= \max\{\lambda_2(\mathbf{W}), -\lambda_n(\mathbf{W})\} \\ &= \max\{1 - \alpha\lambda_{n-1}(\mathbf{L}), \alpha\lambda_1(\mathbf{L}) - 1\}\,, \end{aligned} \tag{10}$$

where $\lambda_i(\mathbf{W})$ denotes the ith eigenvalue associated to $\mathbf{W}$. The value of α that minimizes (10) is given by:

$$\alpha^* = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{N-1}(\mathbf{L})}\,. \tag{11}$$

4 Data-Distributed RVFL Network

Let us consider an artificial neural network with a single hidden layer and a single output node, obtained as a weighted sum of B nonlinear transformations of the input:

$$f_{\omega}(\mathbf{x}) = \sum_{m=1}^{B} \beta_m h_m(x; \omega_m) = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}(\mathbf{x}; \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m)\,, \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^d$, and each function $h_i(\cdot)$ is parameterized by a real-valued vector $\boldsymbol{\omega}_i$. The resulting model is named Functional-Link Artificial Neural Network (FLANN) and the functions $h_i(\cdot)$ are named basis functions, or functional links [3]. A partic-

ular instance of FLANN is the Random Vector Functional-Link (RVFL) network, in which the internal parameters of the B basis functions $\{\boldsymbol{\omega}_i\}_{i=1\dots B}$ are randomly generated from a fixed probability distribution before the learning process [11]. In [8] it was proven that, if the dimensionality of the functional expansion is adequately high, RVFLs possess universal approximation capabilities for a wide range of basis functions $\mathbf{h}(\cdot)$.

Let $T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)\}$ be the set of data available for the training, define the hidden matrix as

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_m(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_L) & \cdots & h_m(\mathbf{x}_L) \end{bmatrix} \tag{13}$$

and let $\mathbf{y} = \left[y_1, y_2, \dots, y_l\right]^{\mathrm{T}}$ be the output vector. Notice that we omitted the parameterization with respect to the parameters $\boldsymbol{\omega}_i$ for better readability. The optimal weights $\boldsymbol{\beta}^*$ of the RVFL are obtained as the solution of the regularized least-squares problem:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 , \tag{14}$$

where $\lambda > 0$ is a regularization factor. Solution to Eq. (14) can be expressed in closed form as:

$$\boldsymbol{\beta}^* = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y} . \tag{15}$$

In a sequential scenario, at each time step we receive a portion of the overall dataset T, i.e. $T = \bigcup_{i=1}^{P} T_i$, for a given number P of batches. Let us define $\mathbf{H}_i$ and $\mathbf{y}_i$ as the hidden matrix and output vector computed over the ith batch, respectively. The optimal solution in Eq. (15) can be computed recursively by means of the recursive least-square (RLS) algorithm [9]:

$$\mathbf{P}(n+1) = \mathbf{P}(n) - \mathbf{P}(n)\mathbf{H}_{n+1}^{T}\mathbf{M}_{n+1}^{-1}\mathbf{P}(n) , \tag{16}$$

$$\boldsymbol{\beta}(n+1) = \boldsymbol{\beta} + \mathbf{P}(n+1)\mathbf{H}_{n+1}^{T}\left[\mathbf{y}_{n+1} - \mathbf{H}_{n+1}\boldsymbol{\beta}(n)\right] , \tag{17}$$

where $\mathbf{P}(n)$ is an auxiliary state matrix, and we defined:

$$\mathbf{M}_{n+1} = \mathbf{I} + \mathbf{H}_{n+1}\mathbf{P}(n)\mathbf{H}_{n+1}^{T} . \tag{18}$$

When considering a distributed scenario, we suppose that training data is partitioned throughout a network of agents, such that at every time step each agent receives a new batch, and the local batches are mutually independent. A distributed algorithm for RVFL for the case of a single batch is presented in [11], and later extended to the more general setting in [10]. The algorithm presented in [10] is based on alternating local

Table 1 Description of the datasets

Name	Features	Instances	Task
G50C	50	550	Gaussian of origin (classification)
CCPP	4	9568	Plant output (regression)

updates in Eqs. (16)–(17), with global averaging steps over the output weights, based on the DAC protocol. Although we will choose this algorithm as an experimental benchmark for the 4 consensus strategies herein considered, a detailed analysis of it goes beyond the scope of the paper. For the interested reader, we refer to [10, 11] and references therein.

5 Experimental Results

We compare the performance of the 4 different strategies illustrated in Sect. 3 in terms of number of iterations required to converge to the average and speed of convergence. In order to avoid that a particular network topology compromises the statistical significance of the experiments, we perform 25 rounds of simulation. In each round, we generate a random topology for an 8-nodes network, according to the so-called Erdős-Rényi model, such that every pair of nodes is connected with a fixed probability p. The only global requirement is that the overall topology is connected. In the experiments, we set $p = 0.5$. We consider 2 public available datasets, whose overview is given in Table 1. The G50C is an artificial classification dataset (see for example [11]), while the CCPP is a regression dataset taken from the UCI repository.[1] At each round, datasets are subdivided in batches following the procedure detailed in [10]. Since in real applications the value of the average is not available to the nodes, in order to evaluate the number of iterations, we consider that all the nodes reached consensus when $\|\boldsymbol{\beta}_i(t) - \boldsymbol{\beta}_i(t-1)\|^2 \leq 10^{-6}$ for any value of i.

In Fig. 1 we show the average number of iterations required by the DAC protocol, averaged over the rounds. The x-axis in Fig. 1 shows the index of the processed batch. As expected, the number of DAC iterations shows a decreasing trend as the number of processed training batches grows, since the nodes are slowly converging to a single RVFL model (see [10]). The main result in Fig. 1, however, is that a suitable choice of the mixing strategy can significantly improve the convergence time (and hence the training time) required by the algorithm. In particular, the optimal strategy defined by Eq. (8) achieves the best performance, with a reduction of the required number of iterations up to 35 and 28 % when compared with max-degree and Metropolis-Hasting strategies respectively. On the other side, the strategy based on constant edge matrix in Eq. (9) shows different behaviors for the 2 datasets, probably due to the heuristic nature of this strategy.

[1] https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant.

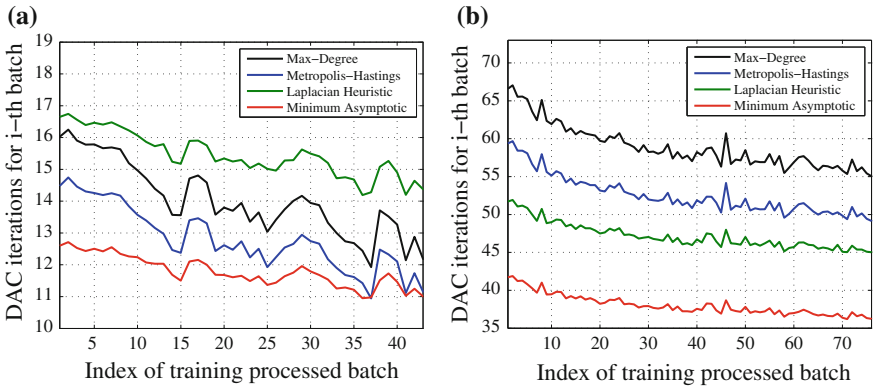

Fig. 1 Evolution of the DAC iterations required by the considered strategies to converge to the average, when processing successive amounts of training batches. **a** Dataset: G50C **b** Dataset: CCPP

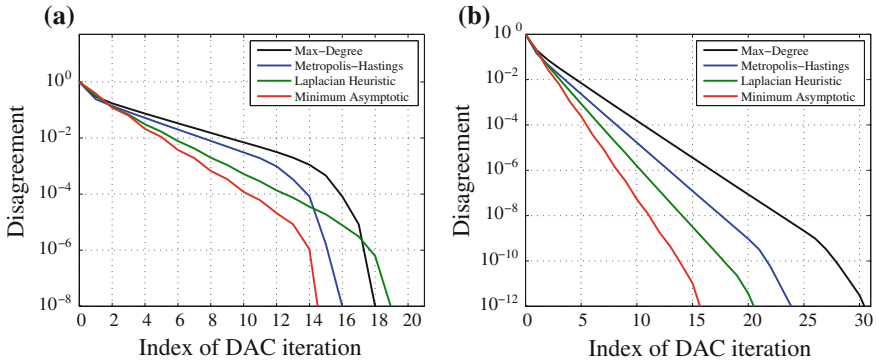

Fig. 2 Evolution of the relative network disagreement for the considered strategies as the number of DAC iterations increases. The *y*-axis is shown with a logarithmic scale. **a** Dataset: G50C **b** Dataset: CCPP

The second experiment, whose results are shown in Fig. 2, is to show the speed of convergence for the considered strategies. This is made by evaluating the trend of the relative network disagreement:

$$RND(t) = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\boldsymbol{\beta}_i(t) - \hat{\boldsymbol{\beta}}\|^2}{\|\boldsymbol{\beta}_i(0) - \hat{\boldsymbol{\beta}}\|^2}, \tag{19}$$

as the number of DAC iterations increases. The value of $\hat{\boldsymbol{\beta}}$ in Eq. (19) is the true average given in Eq. (2). The *y*-axis in Fig. 2 is shown with a logarithmic scale. Results show that the "optimal" strategy has the fastest speed of convergence, as expected, while it is interesting to notice how, when compared to max-degree and Metropolis-Hastings weights, the heuristic strategy achieves a rapid decrease in disagreement in

the initial iterations, while its speed tends to become slower in the end (this is noticeable in Fig. 2a). This may help to explain the lower performance of this strategy in Fig. 1a.

6 Conclusions

In this paper we have conducted an empirical comparison on the performance of different strategies for the DAC protocol. In particular, we compared 4 time-invariant strategies for undirected topologies. Moreover, we focused on the application of the DAC protocol to a recently proposed distributed training algorithm for RVFL networks.

Experimental results show how an appropriate choice of the weights matrix can lead to considerable improvements both in the number of iterations required by the protocol to converge to the average, and in the speed of convergence. In particular, when compared to other strategies, an "optimal" choice of the weights matrix can save up to 30 % in time.

This work can be set in the development of efficient distributed machine learning algorithms. Although we have focused on a specific learning model, nodes can be trained locally using different models than RVFL networks. Moreover, we made the assumption of a fixed, undirected topology. This assumption is common in literature, but limits the applicability of the model to more complex problems and real-world applications [10]. Future works will extend the analysis to time-varying topologies [7] and strategies involving communication constraints (e.g. on the time and energy required for data exchange).

References

1. Baccarelli, E., Cordeschi, N., Mei, A., Panella, M., Shojafar, M., Stefa, J.: Energy-efficient dynamic traffic offloading and reconfiguration of networked datacenters for big data stream mobile computing: review, challenges, and a case study. IEEE Netw. Mag. (2015)
2. Barbarossa, S., Sardellitti, S., Di Lorenzo, P.: Distributed detection and estimation in wireless sensor networks. In: Chellapa, R., Theodoridis, S. (eds.) E-Reference Signal Processing, pp. 329–408. Elsevier (2013)
3. Comminiello, D., Scarpiniti, M., Azpicueta-Ruiz, L., Arenas-Garcia, J., Uncini, A.: Functional link adaptive filters for nonlinear acoustic echo cancellation. IEEE Trans. Audio, Speech, Lang. Process. **21**(7), 1502–1512 (2013)
4. Forero, P.A., Cano, A., Giannakis, G.B.: Consensus-based distributed support vector machines. J. Mach. Learn. Res. **11**, 1663–1707 (2010)
5. Georgopoulos, L., Hasler, M.: Distributed machine learning in networks by consensus. Neurocomputing **124**, 2–12 (2014)
6. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
7. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control **49**(9), 1520–1533 (2004)

8. Pao, Y.H., Park, G.H., Sobajic, D.J.: Learning and generalization characteristics of the random vector functional-link net. Neurocomputing **6**(2), 163–180 (1994)
9. Scardapane, S., Comminiello, D., Scarpiniti, M., Uncini, A.: Online sequential extreme learning machine with kernels. IEEE Trans. Neural Netw. Learn. Syst. (2015)
10. Scardapane, S., Fierimonte, R., Wang, D., Panella, M., Uncini, A.: Distributed music classification using random vector functional-link nets. In: Accepted for presentation at 2015 IEEE/INNS International Joint Conference on Neural Networks (IJCNN'15) (2015)
11. Scardapene, S., Wang, D., Panella, M., Uncini, A.: Distributed learning for random vector functional-link networks. Inf. Sci. **301**, 271–284 (2015)
12. Xiao, L., Boyd, S.: Fast linear iterations for distributed averaging. Syst. Control Lett. **53**(1), 65–78 (2004)