Distributed Music Classification Using Random Vector Functional-Link Nets

Simone Scardapane, Roberto Fierimonte, Dianhui Wang,
Massimo Panella and Aurelio Uncini

***Abstract*— In this paper, we investigate the problem of music classification when training data is distributed throughout a network of interconnected agents (e.g. computers, or mobile devices), and it is available in a sequential stream. Under the considered setting, the task is for all the nodes, after receiving any new chunk of training data, to agree on a single classifier in a decentralized fashion, without reliance on a master node. In particular, in this paper we propose a fully decentralized, sequential learning algorithm for a class of neural networks known as Random Vector Functional-Link nets. The proposed algorithm does not require the presence of a single coordinating agent, and it is formulated exclusively in term of local exchanges between neighboring nodes, thus making it useful in a wide range of realistic situations. Experimental simulations on four music classification benchmarks show that the algorithm has comparable performance with respect to a centralized solution, where a single agent collects all the local data from every node and subsequently updates the model.**

I. Introduction

Music classification is the task of automatically assigning a song to one (or more) classes, depending on its audio content [1]. It is a fundamental task in many music information retrieval (MIR) systems, whose broader scope is to efficiently retrieve songs from a vast database depending on the user's requirements [2]. Examples of labels that can be assigned to a song include its musical genre [3], [4], artist [5], induced mood [2] and leading instrument [6]. Classically, the interest in music classification is two-fold. First, being able to correctly assess the aforementioned characteristics can increase the efficiency of a generic MIR system (see survey [2] and references therein). Secondly, due to its properties, music classification can be considered as a fundamental benchmark for supervised learning algorithms [1]: apart from the intrinsic partial subjectivity of assigning labels, datasets tend to be relatively large, and a wide variety of features can be used to describe each song. These features can also be supplemented by meta-informations and social tags [7].

In multiple real world applications, a third characteristic of musical data concerns its distributed nature (e.g., songs distributed over multiple computers on a network) [8]. To this end, in this paper we consider the problem of music classification under the additional constraint that training data (i.e. songs) is distributed throughout a network of interconnected nodes. By exploiting their local datasets, and limited communication with their neighbors, nodes must agree on a single classifier, whose generalization capability should approximate sufficiently well that of a centralized model built by first collecting all the local datasets. For generality, no node in the network is allowed to coordinate the training process. Additionally, nodes can communicate only with their direct neighbors, but they are not permitted to exchange their data points. Finally, in this paper a sequential setting is considered, where new training data is arriving continuously in a streaming fashion. This is a highly general setting [9], which subsumes many possible applications, including decentralized music classification on peer-to-peer (P2P) systems [10], and over wireless sensor networks [8].

In particular, we propose a sequential, fully distributed learning algorithm for a class of neural network models known as Random Vector Functional-Link (RVFL) nets, also known as Random-Weights Neural Networks (RWNN) [9], [11], [12]. The algorithm is based on the use of the decentralized average consensus (DAC) method [13], an extremely efficient procedure for computing global averages on a network, starting from local measurement vectors. From a theoretical viewpoint, the proposed algorithm can be seen as an extension of the 'learning by consensus' (LBC) theory outlined in a recent paper by Georgopoulos and Hasler [14]. LBC is a general framework for turning any iterative learning algorithm (defined in the centralized case) into a fully distributed learning algorithm. This is achieved by a two-step procedure. First, each node applies a local update rule (e.g., a gradient descent step) using its own dataset. This results in L local updated classifiers $f_1, \ldots, f_L$, where L is the number of nodes in the network. Then, the final classifier is obtained by averaging the functions $f_1, \ldots, f_L$ in a decentralized fashion using a DAC algorithm.

In this paper, we extend the LBC idea to the sequential learning setting, and we apply the same two-step procedure for training a RVFL in a decentralized, online manner. As local update rule, a standard blockwise recursive least-square (BRLS) method is used [15]. Successively, the resulting algorithm is tested on four freely available music classification benchmarks, showing that it compares favorably with a RVFL trained in a centralized way using all the local datasets, in terms of accuracy, speed and efficiency. Although in this paper we focus on the application of the algorithm to music classification tasks, we note that it can be considered as a general decentralized procedure for training RVFL nets, thus

Simone Scardapane, Roberto Fierimonte, Massimo Panella and Aurelio Uncini are with the Department of Information Engineering, Electronics and Telecommunications (DIET), "Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy. Emails: {simone.scardapane, massimo.panella, aurelio.uncini}@uniroma1.it, roberto.fierimonte@gmail.com. Dianhui Wang is with the Department of Computer Science and Computer Engineering, La Trobe University, Melbourne, VIC 3086, Australia. Email: dh.wang@latrobe.edu.au.

making it applicable to a wider range of domains. We will come back on this point in the concluding remarks.

With respect to the state-of-the-art, the strengths of RVFL neural networks for music classification tasks have been explored previously in [1]. In particular, RVFL are linear-in-the-parameters models, making them extremely efficient to train even when confronted with large amounts of data (see the benchmark experiments in [1]). Two distributed, batch training algorithms for RVFLs were proposed in [9]. Under this respect, the algorithm presented here can be seen as a sequential extension of the DAC-based batch algorithm of [9]. Additionally, we underline that the algorithm proposed in this paper shares some similarities with the diffusion recursive least-square (DRLS) [16], belonging to the family of diffusion adaptive filters. Finally, since the algorithm is based on a distributed average of multiple predictors, it is connected to the parallelized, online ensemble literature [17]. More generally, learning with distributed training data is a highly active research topic in several scientific areas (see [9, Section 1] and references therein). In particular, significant work can be found for parallelizing gradient descent optimization procedures in the context of big data processing [18]. The DAC algorithm, instead, is fundamental in several approaches to distributed learning, particularly in the case of WSNs networks [13], [19] and with the use of the Alternating Direction Method of Multipliers algorithm (ADMM) [9], [20].

The rest of the paper is organized as follows. In Section II we introduce the basic theory of RVFL training in the centralized case, and the DAC method. Then, in Section III we formulate the distributed music classification problem, and describe the algorithm that we propose. To validate our proposal, in Section IV we show experimental results on several freely available music classification benchmarks. Finally, in Section V we summarize our contribution and outline future lines of research.

II. Preliminaries

In this section we introduce briefly the basic theory required for formulating our algorithm. In particular, we describe the RVFL model in Section II-A, followed by the BRLS training algorithm in Section II-B. Then, we describe the DAC protocol in Section II-C.

A. Random Vector Functional-Link Nets

A RVFL net is a function mapping a generic input $\mathbf{x} \in \mathbb{R}^d$ to a linear combination of B non-linear transformations of the input itself [11], [12]:

$$f(\mathbf{x}) = \sum_{m=1}^{B} \beta_m h_m(\mathbf{x}; \mathbf{w}_m) = \boldsymbol{\beta}^T \mathbf{h}(\mathbf{x}; \mathbf{w}_1, \dots, \mathbf{w}_B) \,. \quad (1)$$

Parameters $\mathbf{w}_1, \dots, \mathbf{w}_B$ in Eq. (1) are extracted from an uniform probability distribution at the beginning of the learning process. Thus, the model is linear in the set of free parameters $\boldsymbol{\beta}$. Despite this simplification, it can be shown that model (1) possesses universal approximation capability, provided a sufficiently large number of non-linear transformations is adopted in Eq. (1) [12]. More precisely, under moderate assumptions on the smoothness of the underlying function, the approximation error of a RVFL net is in the order $\mathcal{O}(\frac{C}{\sqrt{B}})$, for a given constant $C \in \mathbb{R}$ independent of B [12]. Clearly, the simplicity of model (1) is counter-balanced by the need of having a large expansion block (i.e., a large B), together with a possible increase in variance due to the stochastic choice of the parameters $\mathbf{w}_1, \dots, \mathbf{w}_B$.

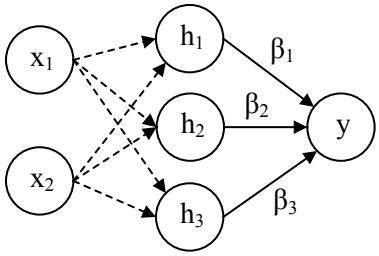


Fig. 1. Schematic depiction of a RVFL architecture with two inputs, three hidden nodes, and one output. Fixed connections are shown as dashed lines, whilst trainable connections as fixed lines.

For choosing the parameters $\boldsymbol{\beta}$, suppose we are given a set of N examples of the mapping that the model must satisfy, in the form of a training set S:

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \,. \quad (2)$$

Moreover, define the hidden matrix $\mathbf{H}$ and the output vector $\mathbf{y}$ as:

$$\mathbf{H} = [\mathbf{h}(\mathbf{x}_1) \dots \mathbf{h}(\mathbf{x}_N)]^T \,, \quad (3)$$

$$\mathbf{y} = [y_1 \dots y_N]^T \,, \quad (4)$$

where we have dropped the parameterization of $\mathbf{h}(\cdot)$ with respect to the weights $\mathbf{w}_1, \dots, \mathbf{w}_B$ for readability. Then, the optimal weight vector is obtained by solving the following regularized least-square problem:

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^B}{\arg\min} \frac{1}{2} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \,, \quad (5)$$

where $\lambda \in \mathbb{R}$ is a scalar known as *regularization factor*. Solution of problem (5) can be obtained in closed form as:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H}^T\mathbf{y} \,. \quad (6)$$

B. Sequential Learning for RVFL

In a sequential setting, the dataset S is not processed as a whole, but it is presented in a series of batches (or chunks) $S_1, \dots, S_T$ such that:

$$\bigcup_{i=1}^{T} S_i = S \,. \quad (7)$$

This encompasses situations where training data arrives in a streaming fashion, or the case where the dataset S is too large for the matrix inversion in Eq. (6) to be practical. RVFLs can be trained efficiently in the sequential setting by the use of the BRLS algorithm [15]. Denote by $\boldsymbol{\beta}[n]$ the estimate

of the optimal weight vector after having observed the first n chunks, and by $\mathbf{H}_{n+1}$ and $\mathbf{y}_{n+1}$ the matrices collecting the hidden nodes values and outputs of the $(n+1)$th chunk S_{n+1}. BRLS recursively computes Eq. (6) by the following two-step update:

$$\mathbf{P}[n+1] = \mathbf{P}[n] - \mathbf{P}[n]\mathbf{H}_{n+1}^T\mathbf{M}_{n+1}^{-1}\mathbf{H}_{n+1}\mathbf{P}[n]\,, \tag{8}$$

$$\boldsymbol{\beta}[n+1] = \boldsymbol{\beta}[n] + \mathbf{P}[n+1]\mathbf{H}_{n+1}^T\left(\mathbf{y}_{n+1} - \mathbf{H}_{n+1}\boldsymbol{\beta}[n]\right)\,, \tag{9}$$

where we have defined:

$$\mathbf{M}_{n+1} = \mathbf{I} + \mathbf{H}_{n+1}\mathbf{P}[n]\mathbf{H}_{n+1}^T\,. \tag{10}$$

The matrix $\mathbf{P}$ in Eq. (9) and Eq. (10) can be initialized as $\mathbf{P}[0] = \lambda^{-1}\mathbf{I}$, while the weights $\boldsymbol{\beta}[0]$ as the zero vector. For a derivation of the algorithm, based on the Sherman-Morrison formula, and an analysis of its convergence properties we refer the interested reader to [15].

C. Decentralized Average Consensus

For the rest of the paper, we will consider a network of L interconnected agents, whose connectivity is known *a-priori* and is fixed. Example of agents are computers in a P2P network, sensors in a WSN, or robots in a robot swarm. We can fully describe the connectivity between the nodes in the form of an $L \times L$ connectivity matrix $\mathbf{W}$, where $W_{ij} \neq 0$ if and only if nodes i and j are connected. For simplicity, we will assume that the network is connected (i.e., every node can be reached from another node with a finite number of steps), and undirected (i.e., $\mathbf{W}$ is symmetric).

DAC is a fundamental iterative network protocol to compute global averages with respect to local measurements of the nodes, requiring only local communications between them [13], [14], [19]. Its simplicity makes it suitable for implementation even in the most basic networks [14]. In particular, suppose that at the beginning of the DAC algorithm every node k has a measurement (column) vector denoted by $\boldsymbol{\theta}_k[0]$, $k = 1 \dots L$. Then, at a generic iteration $n+1$, every node computes the following update:

$$\boldsymbol{\theta}_k[n+1] = \sum_{j=1}^{L} W_{kj}\boldsymbol{\theta}_j[n]\,. \tag{11}$$

Defining the matrix $\boldsymbol{\theta}[n] = \left[\boldsymbol{\theta}_1[n] \dots \boldsymbol{\theta}_L[n]\right]$, Eq. (11) can be written compactly as:

$$\boldsymbol{\theta}[n+1] = \mathbf{W}\boldsymbol{\theta}[n]\,. \tag{12}$$

If the weights of the connectivity matrix $\mathbf{W}$ are chosen appropriately, this recursive procedure converges to the global average given by [13], [14]:

$$\lim_{n\to+\infty} \boldsymbol{\theta}_k[n] = \frac{1}{L}\sum_{k=1}^{L}\boldsymbol{\theta}_k[0],\ \forall k \in \{1, 2, \dots, L\}\,. \tag{13}$$

Practically, the procedure can be stopped after a certain number of iterations is reached, or when the norm of the update is smaller than a given user-defined threshold δ:

$$\left\|\boldsymbol{\theta}_k[n+1] - \boldsymbol{\theta}_k[n]\right\|_2^2 < \delta,\ \forall k \in \{1, 2, \dots, L\}\,. \tag{14}$$

In the case of undirected, connected networks, a simple way of ensuring convergence is given by choosing the so-called 'max-degree' weights for the connectivity matrix $\mathbf{W}$ [13]:

$$W_{kj} = \begin{cases} \frac{1}{d+1} & \text{if } k \text{ is connected to } j \\ 1 - \frac{d_k}{d+1} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}\,, \tag{15}$$

where d_k is the degree of node k, and d is the maximum degree of the network.[1]

III. Distributed Music Classification

In this section we describe the problem of distributed music classification in Section III-A, followed by the illustration and application of our distributed training algorithm for RVFL nets in Section III-B.

A. Problem Statement

In music classification, we suppose that the input $\mathbf{x} \in \mathbb{R}^d$ to the model is given by a suitable d-dimensional representation of a song. Examples of features that can be used in this sense include temporal features such as the zero-crossing count, compact statistics in the frequency and cepstral domain [2], [6], higher-order descriptors (e.g. timbre [5]), meta-information on the track (e.g., author), and social tags extracted from the web [7]. Features can also be learned from the musical data itself [21]. The output is instead given by one of M predefined classes, where each class represents a particular categorization of the song, such as its musical genre. In our experiments, we consider the standard M bit encoding for the output, associating to an input $\mathbf{x}_i$ a single output vector $\mathbf{y}_i$ of M bits, where if its elements are $y_{ij} = 1$ and $y_{ik} = 0$, $k \neq j$, then the corresponding pattern is of class j. We can retrieve the actual class from the M-dimensional RVFL output as:

$$\text{Class of } \mathbf{x} = \arg\max_{j=1\dots M} f_j(\mathbf{x})\,, \tag{16}$$

where $f_j(\mathbf{x})$ is the jth element of the M-dimensional output $f(\mathbf{x})$. The derivation in Section II-A extends trivially to the situation of multiple outputs. In this case, $\boldsymbol{\beta}$ becomes a $B \times M$ matrix and the output vector $\mathbf{y}$ becomes an $N \times M$ matrix, where the ith row corresponds to the M-dimensional output $\mathbf{y}_i^T$ of the training set. Additionally, we replace the L_2-norm on vectors in (5) with a suitable matrix norm.

We will assume that the training data is distributed throughout a network of nodes, and each node k receives a sequence of chunks $S_{k,1}, \dots, S_{k,T}$. The task if for all the nodes, after receiving any new chunk of data, to agree on a single RVFL model, whose generalization capability should approximate reasonably well that of a single RVFL model trained by first collecting all the local batches. For generality

[1]The degree of a node is the number of nodes in the network to which it is directly connected.

purposes, we assume that exchange of the local datasets is forbidden, and that the nodes are allowed to communicate only with their direct neighbors. In the following, we present a decentralized, sequential learning algorithm for RVFL models fulfilling all these properties.

B. Consensus-based Sequential RVFL

The proposed training algorithm is based on alternating local update steps, through the use of the BRLS algorithm described in Section II-B, and global averaging steps, using the DAC protocol presented in Section II-C. Practically, we consider the following algorithm:

1) **Initialization**: the nodes agree on parameters $\mathbf{w}_1, \dots, \mathbf{w}_B$ in Eq. (1). In particular a single node, chosen with a leader election strategy [9], can extract them at random and broadcast them to the rest of the network. Moreover, all the nodes initialize their own local estimate of the $\mathbf{P}$ matrix in Eq. (8) and Eq. (10) as $\mathbf{P}_k[0] = \lambda^{-1}\mathbf{I}$, and their estimate of the output weight vector as $\boldsymbol{\beta}_k[0] = \mathbf{0}$.
2) At every iteration $n+1$, each node k receives a new batch $S_{k,n+1}$. The following steps are performed:
   2.1) **Local update**: every node computes (locally) its estimate $\boldsymbol{\beta}_k[n+1]$ using Eqs. (8)-(9) and local data $S_{k,n+1}$.
   2.2) **Global average**: the nodes agree on a single parameter vector by averaging their local estimates with a DAC protocol. The final weight vector at iteration $n+1$ is then given by:

$$\boldsymbol{\beta}[n+1] = \frac{1}{L}\sum_{k=1}^{L}\boldsymbol{\beta}_k[n+1]\,. \tag{17}$$

As we stated in Section I, this algorithm can be justified as an extension of the 'learning by consensus' theory outlined in [14], or as a decentralized bagging procedure over linear predictors [17].

IV. Experimental results

A. Experiment Setup

We tested the proposed algorithm on four freely available music classification benchmarks. A schematic description of their characteristics is given in Table I. Below we provide more information on each of them.

- *Garageband* [22] is a genre classification dataset, considering 1856 songs and 9 different genres (alternative, blues, electronic, folkcountry, funksoulrnb, jazz, pop, raphiphop and rock). The input is given by 49 features extracted according to the procedure detailed in [22].
- *LMD* is another genre classification task, of higher difficulty [3]. In this case, we have 3160 different songs categorized in 10 Latin American genres (tango, bolero, batchata, salsa, merengue, ax, forr, sertaneja, gacha and pagode). The input is a 30-dimensional feature vector, extracted from the middle 30 seconds of every song.
- *Artist20* is an artist recognition task comprising 1413 songs distributed between 20 different artists [5]. The 30-dimensional input vector comprises both Mel Frequency Cepstral Coefficients (MFCC) and chroma features (see [5] for additional details).
- *YearPredictionMSD* is a year recognition task derived from the subset of the million song dataset [23] available on the UCI machine learning repository.[2] It is a dataset of 500000 songs categorized by year. In our experiment, we consider a simplified version comprising only the initial 200000 songs, and the following binary classification output: a song is of class (a) if it was written previously than 2000, and of class (b) otherwise. This is a meaningful task due to the unbalance of the original dataset with respect to the decade $2001-2010$.

In all cases, input features were normalized between -1 and $+1$ before the experiments. Testing accuracy is computed over a 10-fold cross-validation of the data, and every experiment is repeated 50 times to average out randomness effects due to the initialization of the parameters. Additionally, to increase the dataset size, we artificially replicate twice the training data for all datasets, excluding YearDatasetMSD.

We consider networks of 8 nodes, whose topology is constructed according to the so-called 'Erdős−Rényi model'. In particular, every pair of nodes in the network has a 20% probability of being connected, with the only constraint that the overall network is connected. Training data is distributed evenly across the nodes, and chunks are constructed such that every batch is composed of approximately 20 examples (100 for the YearPredictionMSD dataset). We compare the following algorithms:

- **Consensus-based RVFL** (CONS-RVFL): this is trained according to the DAC-based algorithm detailed in Section III-B. For the DAC procedure, we set the maximum number of iterations to 300, and $\delta = 10^{-4}$.
- **Centralized RVFL** (C-RVFL): this is a RVFL trained by first collecting all the local chunks and aggregating them in a single batch. It can be considered as an upper bound on the performance of CONS-RVFL.
- **Local RVFL** (L-RVFL): in this case, nodes update their estimate using their local batch, but no communication is performed. Final misclassification error is averaged across the nodes. This can be considered as a worst-case baseline for the performance of any distributed algorithm.

In all cases, we use sigmoid hidden functions given by:

$$h(\mathbf{x};\mathbf{w},b) = \frac{1}{1+\exp\left\{-\mathbf{w}^T\mathbf{x}+b\right\}}\,. \tag{18}$$

where parameters $\mathbf{w}$ and b in (18) are assigned randomly from an uniform distribution over the interval $[-1,+1]$. Optimal parameters for C-RVFL are found by executing an inner 3-fold cross-validation on the training data. In particular, we search the uniform interval $\{50,100,150,\dots,1000\}$ for the number of hidden nodes, and the exponential interval 2^j,

[2] https://archive.ics.uci.edu/ml/

TABLE I

GENERAL DESCRIPTION OF THE DATASETS. ADDITIONAL INFORMATION ON EACH OF THEM IS PROVIDED IN SECTION IV-A.

Dataset name	**Features**	**Instances**	**Task**	**Classes**	**Reference**
Garageband	49	1856	Genre recognition	9	[22]
Latin Music Database (LMD)	30	3160	Genre recognition	10	[3]
Artist20	30	1413	Artist recognition	20	[5]
YearPredictionMSD	90	200000	Decade identification	2	[23]

TABLE II

OPTIMAL PARAMETERS FOUND BY THE GRID-SEARCH PROCEDURE.

Dataset	**Hidden nodes**	λ
Garageband	300	2^{-3}
LMD	400	2^{-2}
Artist20	200	2^{-4}
YearPredictionMSD	300	1

TABLE III

FINAL MISCLASSIFICATION ERROR AND TRAINING TIME FOR THE THREE MODELS, TOGETHER WITH STANDARD DEVIATION. THE PROPOSED ALGORITHM IS HIGHLIGHTED IN BOLD. TRAINING TIME FOR CONS-RVFL AND L-RVFL IS AVERAGED OVER THE NODES.

Dataset	**Algorithm**	**Error**	**Time** [secs]
Garageband	C-RVFL	0.40 ± 0.02	0.24 ± 0.09
	L-RVFL	0.45 ± 0.03	0.13 ± 0.03
	CONS-RVFL	$\mathbf{0.40 \pm 0.02}$	$\mathbf{0.15 \pm 0.04}$
LMD	C-RVFL	0.25 ± 0.02	0.70 ± 0.17
	L-RVFL	0.31 ± 0.03	0.46 ± 0.08
	CONS-RVFL	$\mathbf{0.26 \pm 0.02}$	$\mathbf{0.49 \pm 0.10}$
Artist20	C-RVFL	0.37 ± 0.04	0.13 ± 0.07
	L-RVFL	0.47 ± 0.04	0.06 ± 0.01
	CONS-RVFL	$\mathbf{0.37 \pm 0.04}$	$\mathbf{0.09 \pm 0.02}$
YearPredictionMSD	C-RVFL	0.27 ± 0.01	8.66 ± 0.93
	L-RVFL	0.27 ± 0.01	2.35 ± 0.48
	CONS-RVFL	$\mathbf{0.27 \pm 0.01}$	$\mathbf{2.46 \pm 0.62}$

$j \in \{-10, -9, \ldots, 9, 10\}$ for λ. These parameters are then shared with L-RVFL and CONS-RVFL. Resulting parameters from the grid search procedure are listed in Table II. All experiments are performed using MATLAB R2013b on an Intel Core2 Duo E7300, @2.66 GHz and 2 GB of RAM.

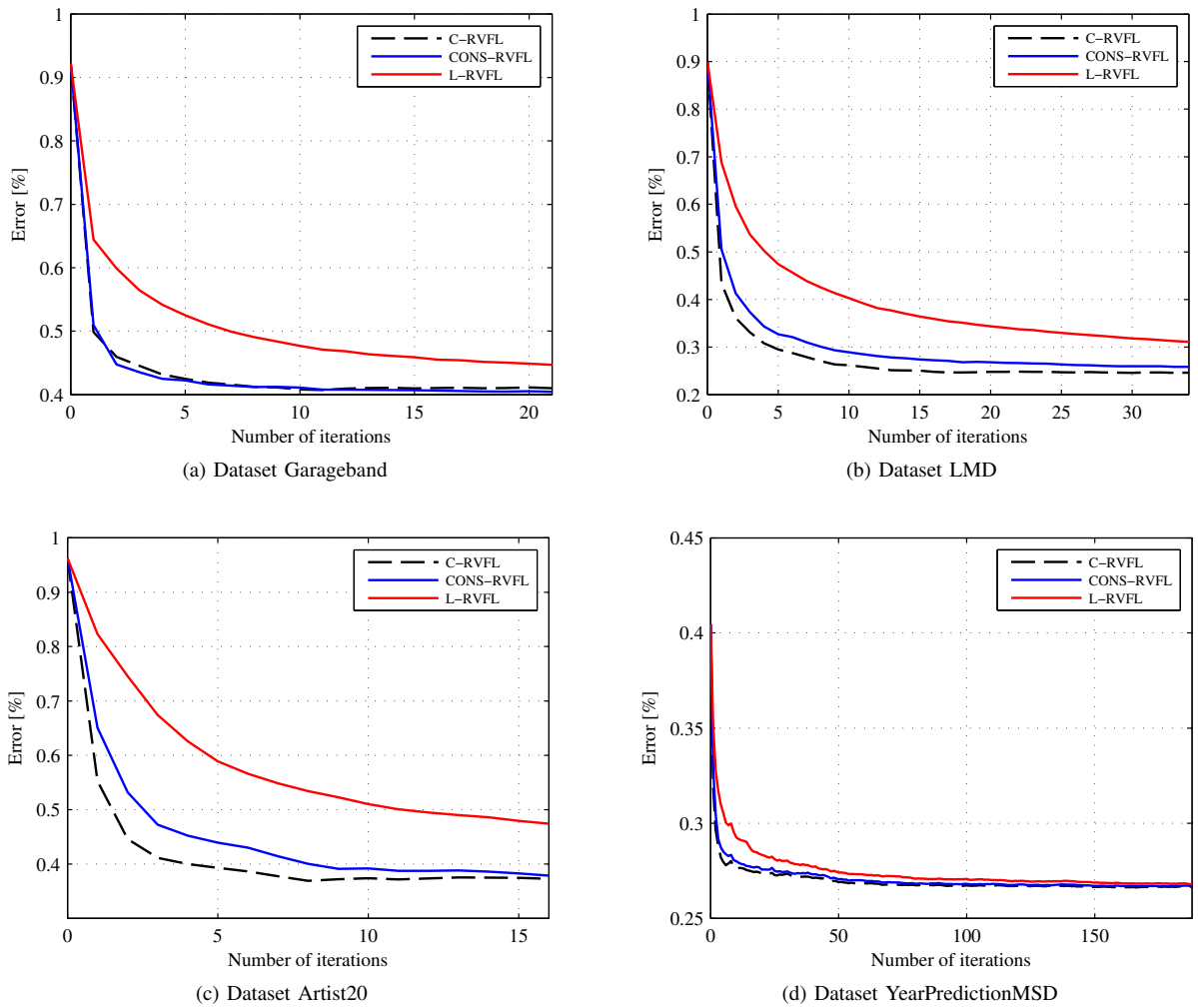
B. Results and Discussion

We start our discussion of the results by analyzing the final misclassification error and training time for the three models, reported in Table III. Results of the proposed algorithm, CONS-RVFL, are highlighted in bold. Clearly, whenever we consider medium-sized datasets, the performance of L-RVFL is strictly worse than the performance of C-RVFL, ranging from an additional 5% misclassification error for Garageband and LMD, up to an additional 10% for Artist20. Although this is a trivial result, it stresses the importance of leveraging all available data to obtain good performances. The most important fact highlighted in Table III, however, is that CONS-RVFL is able to efficiently match the performance of C-RVFL in all situations, except for a small decrease in the LMD dataset. From a computational perspective, this performance is achieved with a very small overhead in term of training time with respect to L-RVFL in all cases (as evidenced by the fourth column in Table III).

In a sequential setting, the evolution of the testing error after every batch is equally as important as the final accuracy obtained. We report it in Fig. 2(a)-(d) for the four datasets. Performance of C-RVFL, L-RVFL and CONS-RVFL are shown with dashed black, solid red and solid blue lines respectively. Moreover, performance of L-RVFL is averaged across the nodes. Once again, we see that CONS-RVFL is able to track very efficiently the accuracy obtained by C-RVFL. The performance is practically equivalent in the Garageband and YearPredictionMSD datasets (Fig. 2(a) and Fig. 2(d)), while convergence speed is slightly slower in the LMD and Artist20 case (Fig. 2(b) and Fig. 2(c)), although by a small amount.

Next, we investigate the behavior of CONS-RVFL when varying the size of the network. In fact, due to its parallel nature, we expect that, the higher the number of nodes, the lower the training time (apart from communication bottlenecks, depending on the real channel of the network). The following experiments show that the increase in time required by the DAC procedure for bigger networks is more than compensated by the gain in time obtained by processing a lower number of samples per node. To this end, we consider the training time required by CONS-RVFL when varying the number of nodes of the network from 2 to 14 by steps of 2, keeping the same topology model as before. Results of this experiment are presented in Fig. 3(a) for datasets Garageband and Artist20, and in Fig. 3(b) for datasets LMD and YearPredictionMSD. The decrease in training time is extremely pronounced for Garageband, with a five-fold decrease going from 2 to 14 nodes, and for YearPredictionMSD, with a seven-fold decrease. This result is especially important, showing that CONS-RVFL can be efficiently used in large-scale situations. It is also consistent with the analysis of the batch DAC-based RVFL [9].

Similarly, the number of consensus iterations needed to reach the desired accuracy is shown in Fig. 4. Although the required number of iterations grows approximately linearly



(a) Dataset Garageband

(b) Dataset LMD

(c) Dataset Artist20

(d) Dataset YearPredictionMSD

Fig. 2. Evolution of the testing error after every iteration. Performance of L-RVFL is averaged across the nodes.

with respect to the size of the network, a low number of iterations is generally enough to reach convergence to a very good accuracy. In fact, no experiment in this section required more than 35 iterations in total. Additionally, the consensus procedure is extremely robust to a change in the network topology, as shown in [9, Section 5.2]. The same considerations apply here.

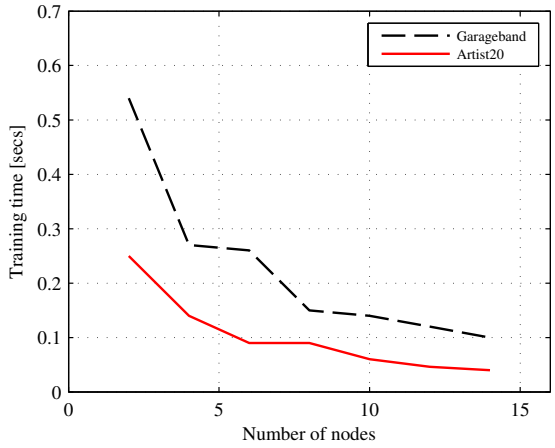
V. Conclusions

In this paper, we have detailed an algorithm for sequential, distributed learning of a RVFL net, based on alternating local update and global averaging steps. We have focused on its application to multiple distributed music classification tasks, including genre and artist recognition. These problems arise frequently in real-world scenarios, including P2P and mobile networks. Our experimental results show that the proposed algorithm can be efficiently applied in these situations, and compares favorably with a centralized solution in terms of accuracy and speed. Clearly, the algorithm can be successfully applied to distributed learning problems laying outside the applicative domain of this paper, particularly in real-world big data applications. Moreover, although in this paper we have focused on local updates based on the BRLS algorithm, nothing prevents the framework from being used with different rules, including efficient stochastic gradient descent updates. Currently, the main limitation of the algorithm is the need for a global synchronization method over the network for the DAC procedure, and the assumption of a fixed topology. Although these are standard assumptions in the distributed learning literature, they can be a limitation in extremely variable settings such as particular P2P networks [10]. Future work will consider the use of asynchronous consensus strategies [24] to this end.

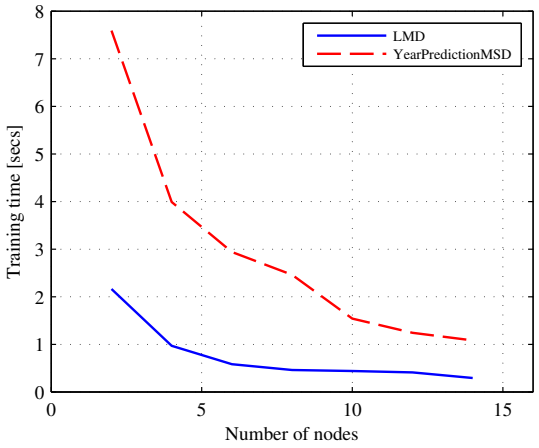
References

[1] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "Music classification using extreme learning machines," in *2013 8th International Symposium on Image and Signal Processing and Analysis (ISPA)*. IEEE, 2013, pp. 377–381.

[2] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 303–319, 2011.



(a) Datasets Garageband and Artist20



(b) Dataset LMD and YearPredictionMSD

Fig. 3. Training time required by CONS-RVFL, for varying sizes of the network, from 2 to 14 by steps of 2.

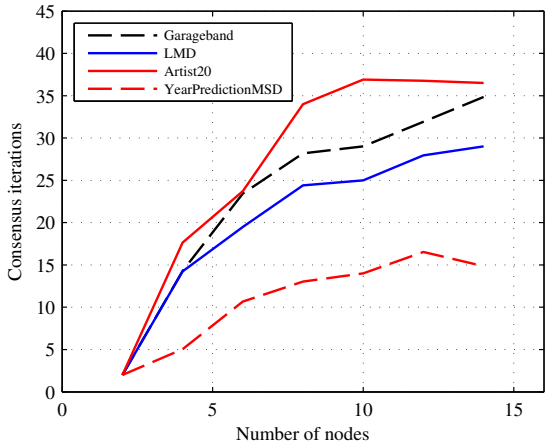


Fig. 4. Number of consensus iterations required to reach convergence, when varying the number of nodes in the network from 2 to 14.

[3] C. N. Silla, C. A. A. Kaestner, and A. L. Koerich, "Automatic music genre classification using ensemble of classifiers," in *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2007, pp. 1687–1692.

[4] A. Rizzi, N. M. Buccino, M. Panella, and A. Uncini, "Genre classification of compressed audio data," in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, Oct 2008, pp. 654–659.

[5] D. P. W. Ellis, "Classifying music audio with timbral and chroma features," in *Proceedings of the 8th International Conference on Music Information Retrieval*. Austrian Computer Society, 2007, pp. 339–340.

[6] A. Eronen, "Comparison of features for musical instrument recognition," in *2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*. IEEE, 2001, pp. 19–22.

[7] L. Chen, P. Wright, and W. Nejdl, "Improving music genre classification using collaborative tagging data," in *Proceedings of the second ACM international conference on web search and data mining*. ACM, 2009, pp. 84–93.

[8] S. Ravindran, D. Anderson, and M. Slaney, "Low-power audio classification for ubiquitous sensor networks," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, vol. 4. IEEE, 2004, pp. iv–337–iv–340.

[9] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed Learning with Random Vector Functional-Link Networks," *Information Sciences (under review)*, 2015.

[10] P. Han, B. Xie, F. Yang, and R. Shen, "A scalable p2p recommender system based on distributed collaborative filtering," *Expert systems with applications*, vol. 27, no. 2, pp. 203–210, 2004.

[11] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

[12] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[13] 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.

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

[15] A. Uncini, *Fundamentals of adaptive signal processing*. Springer, 2014.

[16] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, 2008.

[17] A. Fern and R. Givan, "Online ensemble learning: An empirical study," *Machine Learning*, vol. 53, no. 1-2, pp. 71–109, 2003.

[18] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *The Journal of Machine Learning Research*, vol. 13, pp. 165–202, 2012.

[19] 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.

[20] 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.

[21] S. Sigtia and S. Dixon, "Improved music feature learning with deep neural networks." in *2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'14)*, vol. 1. IEEE, 2014, pp. 6959–6963.

[22] I. Mierswa and K. Morik, "Automatic feature extraction for classifying audio data," *Machine learning*, vol. 58, no. 2-3, pp. 127–149, 2005.

[23] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*. University of Miami, 2011, pp. 591–596.

[24] F. Xiao and L. Wang, "Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays," *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1804–1816, 2008.