AN ONLINE ALGORITHM FOR DISTRIBUTED DICTIONARY LEARNING

Symeon Chouvardas[1] *Yannis Kopsinis*[2] *Sergios Theodoridis*[2]

[1]Mathematical and Algorithmic Sciences Lab,
Huawei France R&D,
Paris, France.
Email: symeon.chouvardas@huawei.com

[2]University of Athens,
Dept. of Informatics and Telecommunications,
Athens 15784, Greece.
Emails: kopsinis@ieee.org, stheodor@di.uoa.gr

ABSTRACT

This paper proposes a novel algorithm for online distributed dictionary learning, where a set of nodes is requested to collectively estimate a common dictionary via sequentially received data vectors. At each time instance, in which a new datum becomes available, the sparse representation of the data with respect to the estimated dictionary is computed locally at each node by employing a sparsity promoting algorithm. In the sequel, the nodes cooperate in order to collaboratively update the dictionary via the distributed Recursive Least Squares (RLS) algorithm. Numerical examples, both with synthetic and real data, validate that the performance of the proposed algorithm is comparable to that of centralized state of the art algorithms.

***Index Terms*—** Distributed Dictionary Learning, Online Learning, Sparse Representation, Distributed RLS.

I. INTRODUCTION

In recent years, there is an ever increasing interest in sparse signal representations [1], [2] as the means to deal with various signal processing and machine learning tasks, such as data compression, denoising, restoration, classification [3], [4], [1], [5], [6], etc. Key ingredient in such techniques is to properly build a dictionary, i.e., a set of elementary signals/vectors, called atoms, which is representative for the data of interest. In particular, the objective is the specific signals to be accurately expressed or modeled as a linear combination of a small subset of the dictionary atoms. Although fixed and predefined dictionaries, such as overcomplete wavelet or Gabor dictionaries [7] have demonstrated notable performance in a number of tasks, the most effective dictionaries are those "trained" using a number of available signal exemplars, a practice referred to as *dictionary learning*, [3], [4], [1], [5], [6].

The present work deals with the problem of online dictionary learning in a distributed environment. The Big Data era, in which we live in, has led to several novel challenges associated with the large amount of data to be processed. For instance, data gathered by social networks translate to millions of training signals. Thus, it becomes clear that processing of such a large amount of data might be proved to be infeasible due to lack of processing power and/or of storage capabilities. These limitations can be overstepped if one resorts to the online learning philosophy, e.g., [6]. According to this, the data are processed sequentially, one per iteration step, until all data have been considered or a certain convergence criterion has been satisfied. Learning sparse representation and dictionaries in an online fashion is of paramount importance when a large number of high–dimensional observations vectors has to be processed.

Another approach to deal with large amounts of data is to split the full problem into subtasks and feed a number of processing units. In such a scenario, it is desirable to distribute the computations over the network instead of transmitting and processing all the data in a fusion center (FC). Following this philosophy, we propose a method for training the dictionary in a decentralized way as follows: the network nodes/agents are assumed to form an ad–hoc network and the dictionary is computed cooperatively; that is, each node computes the dictionary using locally obtained data as well as information, which is received by the nodes of the neighborhood. Besides the computational and storage ease per processing unit, which is attained by distributed processing, another important advantage is that of privacy. In particular, due to the fact that the processing of the data is performed locally and the nodes exchange (intermediate) training results instead of raw data, the need for sensitive information exchange is bypassed [8].

Related Works: Dictionary learning has been extensively studied over the past years, mainly in its batch form, e.g, [4], [9], [6]. In dictionary learning, it is required to compute, simultaneously, a dictionary as well as a set of sparse vectors; the latter are used to combine the columns of the dictionary in a way that sparsely represents the available training data. The most common approach, adopted in the majority of algorithms, is a two–step procedure, known as the *sparse coding step* and the *dictionary update step*. This procedure keeps one set of parameters fixed and minimizes the cost function with respect to the other, e.g., [4], [10]. An online dictionary learning algorithm has been proposed in the seminal work of [10], where the dictionary is computed via the block–coordinate descent method, whereas [11] employs the RLS algorithm for the dictionary computation.

The distributed dictionary learning problem has been studied in [12], [13], [14]. In particular, the work in [12] considers a distributed scenario in which each node/agent is in charge of a portion of the dictionary elements. This problem is effectively solved by exploiting the form of the dual function and resorting to the diffusion rationale. Moreover, in [14] the authors propose a distributed K–SVD algorithm and employ the Alternating Directions Method of Multipliers (ADMM) to solve the problem. Finally, an online distributed algorithm, considering a common dictionary, has been studied in [13] and a diffusion–based gradient descent algorithm is employed.

Contributions: In this paper, a novel algorithm for online distributed dictionary learning, in the setup where the nodes seek for a common dictionary, is presented. Compared to the effort [13], which also studies the online distributed problem, for the first time, here, we propose an RLS–based algorithm, for the dictionary estimation, the performance of which is enhanced and it is almost similar to that of centralized algorithms. For the sparse coding step we employ the Least Angle Regression (LARS) algorithm at each mode. Afterwards, a distributed RLS algorithm is used so that the nodes compute, in a collaborative way, the dictionary. Comparative experiments against other state of the art dictionary learning algorithms verify that the algorithm demonstrates competitive performance.

Notation: Lowercase (uppercase) boldfaced letters stand for vectors (matrices). The set of real numbers is denoted by $\mathbb{R}$.

Symeon Chouvardas was with the University of Athens at the time this research was conducted. The project HANDiCAMS acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 323944 by the FET HANDICAM FP7. This work is partly supported by Marie Curie IEF, "SOL", 302898.

Moreover, $\|\cdot\|$, $\|\cdot\|_1$ are the Euclidean and ℓ_1 norm respectively. Finally, $|\mathcal{S}|$ stands for the cardinality of the set $\mathcal{S}$.

II. PROBLEM FORMULATION

We consider a network consisting of N nodes/agents. Each node, $k \in \mathcal{N} := \{1, 2, \ldots, N\}$, has access to n_k data vectors, $\boldsymbol{x}_k(n) \in \mathbb{R}^M$, $n = 1, \ldots, n_k$. Our goal is to compute an overcomplete dictionary $\boldsymbol{A} \in \mathbb{R}^{M\times K}$, $K > M$, which is common to the nodes of the network, so that:

$$\boldsymbol{x}_k(n) = \boldsymbol{A}\boldsymbol{b}_k(n) \tag{1}$$

where $\boldsymbol{b}_k(n) \in \mathbb{R}^K$ is a sparse vector, i.e., it comprises a small number of non–zero coefficients. The physical reasoning of the model presented in (1) is that each observation vector is a linear combination of few columns of the dictionary $\boldsymbol{A}$.

Assuming that the observations are centrally available at a fusion center for processing, the optimization problem can be written as:

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{B}_k} &\sum_{k\in\mathcal{N}} \|\boldsymbol{X}_k - \boldsymbol{A}\boldsymbol{B}_k\|_F^2,\\ &s.t. \quad \forall k, n \quad \|\boldsymbol{b}_k(n)\|_0 \le T_0,\end{aligned}$$

where $\boldsymbol{X}_k = [\boldsymbol{x}_k(1), \ldots, \boldsymbol{x}_k(n_k)] \in \mathbb{R}^{M\times n_k}$, $\boldsymbol{B}_k = [\boldsymbol{b}_k(1), \ldots, \boldsymbol{b}_k(n_k)] \in \mathbb{R}^{K\times n_k}$, $\|\cdot\|_0$ denotes the ℓ_0 pseudo–norm and T_0 is a positive integer. This problem, which is NP–hard due to the ℓ_0 norm-based sparsity enforcing constraint, is usually relaxed by substituting the ℓ_0 norm constraint by an ℓ_1–norm constraint, e.g., [10]. Alternatively, greedy techniques are employed for the estimation of the sparse coding vector, e.g., [4].

In the sequel, we will discuss how this problem can be tackled in a distributed way. Moreover, the problem will be properly recast so as to be solved in an online fashion.

III. STRATEGIES FOR ONLINE DICTIONARY LEARNING

First of all, in order to help the reader grasp the basic concepts of the online dictionary learning problem, let us describe the methodology of [10]. Since we deal here with a decentralized problem the originally proposed cost function is properly reformulated; the reasoning, however, remains the same. At iteration step n, in which a new datum per node becomes available, estimates are produced by minimizing the function

$$f_n(\boldsymbol{A}) = \frac{1}{n}\sum_{k\in\mathcal{N}}\sum_{i=1}^{n} l(\boldsymbol{x}_k(i), \boldsymbol{A}), \tag{2}$$

where

$$l(\boldsymbol{x}, \boldsymbol{A}) := \min_{\boldsymbol{b}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{b}\|^2 + \lambda\|\boldsymbol{b}\|_1, \tag{3}$$

with λ being a regularization parameter. Furthermore, as it is usually the case in most distributed learning problems, in order to prevent the dictionary from taking arbitrarily large values, we constraint the columns to have an ℓ_2 norm less than or equal to 1 or, equivalently, to belong to the following convex set: $\mathcal{C} := \{\boldsymbol{A} \in \mathbb{R}^{M\times K} : \forall j = 1, \ldots, K, \quad \boldsymbol{a}^{(j)T}\boldsymbol{a}^{(j)} \le 1\}$, where $\boldsymbol{a}^{(j)}$ is the j–th column of $\boldsymbol{A}$. It is worth pointing out that the problem of minimizing (2) is not convex. However, it is convex in each one of its arguments, $\boldsymbol{A}$, $\boldsymbol{b}$, while keeping the other one fixed. A commonly adopted method, which will also be followed here, is a two–step approach, which alternates between the two unknown quantities, minimizing, at each step, one of them, while keeping the other one fixed, e.g., [10], [15], [16].

III-A. Online Dictionary Update via RLS

To keep the discussion simple, let us consider for the time being a centralized mode of operation. The subscripts denoting the node index will be omitted; the decentralized problem will be revisited short after. Recall the previous discussion around the two–step approach. Let us consider the iteration step n, in which the sparse vector, denoted by $\widehat{\boldsymbol{b}}_k(n)$, has been estimated by minimizing $l(\boldsymbol{x}(n), \widehat{\boldsymbol{A}}(n-1))$, where $\widehat{\boldsymbol{A}}(n-1)$ is the most recently computed dictionary. Then, in the dictionary update step, an estimate of the unknown dictionary can be computed by minimizing the following cost function:

$$\widehat{f}_n(\boldsymbol{A}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\zeta^{n-i}\|\boldsymbol{x}(i) - \boldsymbol{A}\widehat{\boldsymbol{b}}(i)\|^2,$$

where $\zeta \in (0, 1]$ is the so called forgeting factor employed so as to diminish the effect of past measurements. This cost function can be effectively minimized by adopting the RLS approach. It is worth pointing out that, the RLS algorithm lends itself nicely in the online dictionary learning problem, since for the dictionary update only the most recently received data vector and the most recently estimated sparse vector are employed without being forced to store the entire history of measurements.

Let us now turn our focus on the decentralized problem. Ideally, we would like the dictionary estimate at step n to be computed as follows:

$$\widehat{\boldsymbol{A}}_k(n) = \arg\min_{\boldsymbol{A}} \frac{1}{n}\sum_{k\in\mathcal{N}}\sum_{i=1}^{n}\zeta^{n-i}\frac{1}{2}\|\boldsymbol{x}_k(i) - \boldsymbol{A}\widehat{\boldsymbol{b}}_k(i)\|^2, \ \forall k \in \mathcal{N}. \tag{4}$$

Notice that (4) decouples over the rows of the dictionary, so the minimization can be equivalently written for each one of the rows $\widehat{\boldsymbol{a}}_k^{(r)}(n)$ of $\widehat{\boldsymbol{A}}_k(n)$, $r = 1, \ldots, M$:

$$\widehat{\boldsymbol{a}}_k^{(r)}(n) = \arg\min_{\overline{\boldsymbol{a}}} \frac{1}{n}\sum_{k\in\mathcal{N}}\sum_{i=1}^{n}\zeta^{n-i}\frac{1}{2}\|x_k^{(r)}(i) - \overline{\boldsymbol{a}}^T\widehat{\boldsymbol{b}}_k(i)\|^2, \tag{5}$$

where $x_k^{(r)}(i)$ is the r–th coefficient of the observation vector.

It can be readily seen that the optimization (5) cannot be minimized distributively, since the summation terms are coupled over the global variable $\overline{\boldsymbol{a}}$. Nevertheless, following a similar philosophy as in [17], [18], this problem can be efficiently solved in a decentralized way. This can be achieved by reformulating the optimization problem as follows, $\forall r = 1, \ldots, M$:

$$\begin{aligned}\widehat{\boldsymbol{a}}_k^{(r)}(n) = \arg\min_{\overline{\boldsymbol{a}}_k} \frac{1}{n}\sum_{k\in\mathcal{N}}\sum_{i=1}^{N}\zeta^{n-i}\frac{1}{2}\|x_k^{(r)}(i) - \overline{\boldsymbol{a}}_k^T\widehat{\boldsymbol{b}}_k(i)\|^2\\ \text{s.t.} \quad \overline{\boldsymbol{a}}_k = \overline{\boldsymbol{a}}_l, \ \forall l \in \mathcal{N}_k,\end{aligned} \tag{6}$$

where $\mathcal{N}_k$ denotes the neighborhood of node k. Considering a fully connected network, i.e., a network in which there exists a path (possibly multihop) connecting any two nodes of the network, then the problems (5), (6) are equivalent, e.g., [19], [18]. On top of that, the formulation of (6) can be solved in a decentralized fashion, via the ADMM, e.g., [20]. The algorithm, which solves (6) distributively, is the so–called Distributed RLS (DRLS) algorithm, originally proposed in [18] and will be employed here, for the dictionary estimation.

Let us now describe in brief the steps of the DRLS algorithm. Each node, k, at each iteration step n, computes the matrix $\boldsymbol{\Phi}_k(n) = \sum_{i=1}^{n}\zeta^{n-i}\widehat{\boldsymbol{b}}_k(i)\widehat{\boldsymbol{b}}_k^T(i) = \zeta\boldsymbol{\Phi}_k(n-1) + \widehat{\boldsymbol{b}}_k(n)\widehat{\boldsymbol{b}}_k^T(n)$, as well as the vector $\boldsymbol{p}_k^{(r)}(n) = \sum_{i=1}^{n}\zeta^{n-i}x_k^{(r)}(i)\widehat{\boldsymbol{b}}_k(i) = \zeta\boldsymbol{p}_k^{(r)}(n-1) + x_k^{(r)}(n)\widehat{\boldsymbol{b}}_k(n)$, $r = 1, \ldots, M$. Before computing the estimates, the algorithm updates the Lagrangian multipliers, which

Table I. Online Dictionary Learning Distributed RLS (OnDiRLS)

Initialize: $\widehat{\boldsymbol{A}}_k(0), \boldsymbol{\Phi}_k(0),\ \boldsymbol{p}_k^{(r)}(0),\ \forall k \in \mathcal{N},\ r = 1, \ldots, M$
FOR n=1,2,...
  FOR k=1,...,N
Compute $\widehat{\boldsymbol{b}}_k(n)$ by minimizing $l(\boldsymbol{x}_k(n), \widehat{\boldsymbol{A}}_k(n-1))$
Update $\boldsymbol{\Phi}_k(n)$.
    FOR r=1,...,M
    Update $\boldsymbol{p}_k^{(r)}(n)$.
    Compute $\widehat{\boldsymbol{a}}_k^{(r)}(n)$ via the DRLS algorithm using $x_k^{(r)}(n), \widehat{\boldsymbol{b}}_k(n)$
    ENDFOR
Normalize the colums of $\widehat{\boldsymbol{A}}_k(n)$ to one.
  ENDFOR
ENDFOR

Table II. The DRLS algorithm

Initialize: $\widehat{\boldsymbol{a}}_k^{(r)}(n,0) = \widehat{\boldsymbol{a}}_k^{(r)}(n-1)$, $\boldsymbol{v}_{k,l}^{(r)}(n,0) = \boldsymbol{v}_{k,l}^{(r)}(n,0)$
FOR t=1,...,R
  Compute $\boldsymbol{v}_{k,l}^{(r)}(n,t)$ via (7).
  Compute $\widehat{\boldsymbol{a}}_k^{(r)}(n,t)$ via (8).
ENDFOR
RETURN $\widehat{\boldsymbol{a}}_k^{(r)}(n) := \widehat{\boldsymbol{a}}_k^{(r)}(n,R)$

are associated to the equality constraints of (6). More specifically, node k stores and updates a vector $\forall l \in \mathcal{N}_k$ as follows:

$$\boldsymbol{v}_{k,l}^{(r)}(n,t) = \boldsymbol{v}_{k,l}^{(r)}(n,t-1) + \frac{c}{2}(\widehat{\boldsymbol{a}}_k^{(r)}(n,t-1) - \widehat{\boldsymbol{a}}_l^{(r)}(n,t-1)), \tag{7}$$

where c is a positive regularization parameter. The index t in (7) is an index denoting the number of iterations the RLS algorithm runs for each new datum and takes values $t = 1, \ldots, R$; the larger the R the better the algorithm performs, at the expense of an increased complexity.

The estimate is then computed by:

$$\begin{aligned}\widehat{\boldsymbol{a}}_k^{(r)}(n,t) &= \boldsymbol{\Phi}_k^{-1}(n)\boldsymbol{p}_k^{(r)}(n) \\ &+ \frac{c}{2}\boldsymbol{\Phi}_k^{-1}(n)\left(|\mathcal{N}_k|\widehat{\boldsymbol{a}}_k^{(r)}(n,t-1) + \sum_{l\in\mathcal{N}_k\setminus\{k\}} \widehat{\boldsymbol{a}}_l^{(r)}(n,t-1)\right) \\ &- \frac{1}{2}\boldsymbol{\Phi}_k^{-1}(n)\sum_{l\in\mathcal{N}_k}(\boldsymbol{v}_{k,l}^{(r)}(n,t) - \boldsymbol{v}_{l,k}^{(r)}(n,t)).\end{aligned} \tag{8}$$

again performing R internal iterations.

IV. THE PROPOSED ALGORITHM

The core steps of the proposed algorithm are summarized in Table I. At some time instance, n, node k estimates the sparse coding vector $\widehat{\boldsymbol{b}}_k(n)$ by minimizing (3) using the most recently estimated dictionary, denoted by $\widehat{\boldsymbol{A}}_k(n-1)$. Put in mathematical terms, $\widehat{\boldsymbol{b}}_k(n)$ is the result of minimizing $l(\boldsymbol{x}_k(n), \widehat{\boldsymbol{A}}_k(n-1))$ with respect to $\boldsymbol{b}$. Here, we employ the LARS algorithm, e.g., [21], for the estimation of $\boldsymbol{b}_k(n)$. However, it should be pointed out that this choice is not restrictive and one can employ any sparse coding algorithm, e.g., [6], [22]. In the sequel, the nodes cooperatively minimize (4) by employing the distributed RLS, the steps of which are given in Table II.

Theorem 1. *The sequence of dictionaries, which will be the the same from node to node due to the equality constraints, converges to a stationary point of the objective function* $f_n(\cdot)$.

Proof. The proof, which follows a similar philosophy as the ones in [10], [23], is omitted due to lack of space and will be presented elsewhere.□

Note that the problem, which is dealt with here, is nonconvex. Thus, convergence to the global optimum cannot be guaranteed. However, according to Theorem 1, the algorithm converges to a stationary point of $f_n(\cdot)$. As it is argued in [10], such points often lead to a good performance in practical applications, such as image denoising [1].

It should be highlighted that the convergence proof is possible due to the two time scales incorporated in the dictionary update phase. This is a notable difference to the online distributed algorithm of [13], which operates in a single time scale, and the theoretical properties of the algorithm are not studied.

Complexity: The most computationally expensive step of the algorithm, is the inversion of the matrix $\boldsymbol{\Phi}_k(n)$. However, it has been shown in [18], that if $\zeta = 1$, which is usually the case, then a closed form expression for the inverse matrix is provided via the matrix inversion lemma. In such case, the complexity of the proposed algorithm amounts to $O(RK^2)$ coming from (8). Finally, it should be stressed that the complexity of the LARS algorithm is of order $O(Mrs^3)$, where s is the number of non-zero coefficients of the sparse coding vectors.

V. NUMERICAL EXAMPLES

In this section, we present numerical experiments, using both synthetic and real data, in order to validate the efficiency of the proposed approach.

V-A. Dictionary Learning using Synthetic Data

In the first experiment, we consider a network comprising $N = 5$ nodes. The data vectors are of dimension $M = 20$, the dimension of the dictionary atoms equals $K = 40$ and the number of nonzero coefficients of the sparse coding vectors is set to 3. We also assume that the observation vectors are contaminated with additive noise, following Gaussian distribution with zero mean and variance equal to 0.01. All the entries of $\boldsymbol{A}$ are drawn independently at random from a uniform distribution in the $[-0.5, 0.5]$ and then the columns of the matrix are normalized. We compare the performance of the proposed algorithm with two variants of the algorithmic scheme in [10] and with the algorithm of [13], called here Online Distributed Gradient Descent (OnDiGrad). It is worth pointing out that, the algorithm of [10] does not operate in a distributed fashion. The previously mentioned variants, which will be explained right after, are proper reformulations in order the scheme to be able to work distributively. In the first one, called here by Online Dictionary Learning Centralized (OnDiLeCe), we assume that the data are centrally available and the algorithm runs over the whole amount of data. In the second variant, which is denoted by Online Dictionary Learning Individual (OnDiLeIn), we assume that the nodes compute the dictionary without cooperation and act as individual learners, instead. The reasoning of this experiment is to compare the proposed algorithm with another scheme, of similar complexity, in the case where the whole information is exploited (OnDiLeCe) and in the case where the nodes use only their local data (OnDiLeIn). Furthermore, the proposed scheme is compared to the OnDiGrad, which is suitable for online dictionary learning in distributed environments. The proposed scheme, OnDiLeCe and OnDiLeIn run $R = 10$ iterations, for each received datum, so as to compute the dictionary and for the proposed algorithm $c = 0.7$ (see (7), (8)). The parameters of the OnDiGrad are chosen so that the algorithm converges in the same error floor as the OnDiLeIn. The performance metric adopted is given by $1 - (1/N)\sum_{k\in\mathcal{N}} |\boldsymbol{a}^{(j)T}\boldsymbol{a}_k^{(j)}(n)|$ where $\boldsymbol{a}^{(j)}$ ($\boldsymbol{a}_k^{(j)}(n)$) stands for the j–th column of the true (estimated) dictionary, taking care for the permutation ambiguity in the dictionary atoms. Finally, the

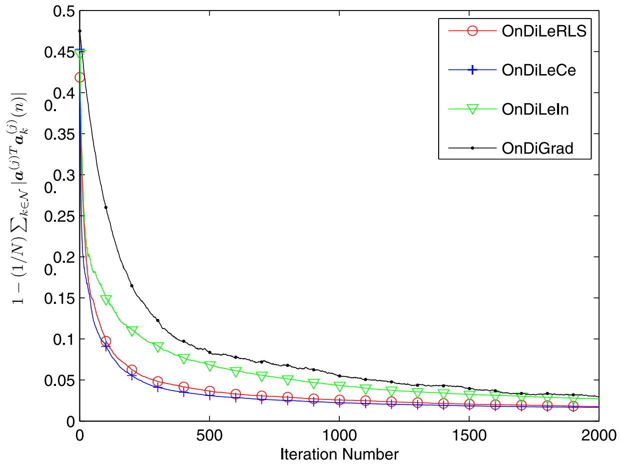

Fig. 1. Performance evaluation using synthetic data.

performance curves result from an ensemble average of 100 independent experiments. Fig. 1 illustrates that the proposed scheme outperforms significantly the OnDiLeIn scheme, since it computes in a fewer number of iterations a dictionary providing a lower error. Moreover, all algorithms outperform OnDiGrad, which is expected since the latter algorithm is of lower complexity. Interestingly, the OnDiLeCe slightly outperforms OnDiRLS; this verifies that the cooperation among the nodes leads us close to the centralized performance.

V-B. Image Denoising via Dictionary Learning

In this subsection, we validate the performance of the OnDiRLS in a real world application and specifically in an image denoising problem. The denoising performance of the proposed algorithm will be evaluated against the K-SVD algorithm [4] using the 256×256 test image commonly known as "boats". The aim here is a vis-a-vis comparison with a widely acclaimed batch algorithm, which is "allowed" to operate in an advantageous set up, compared to our proposed algorithm; that is, all the data are are provided collectively and also the algorithm is allowed to pass many times through the whole batch of data during the dictionary learning phase. On the contrary, in the case of our proposed algorithm, the data are distributed in $N = 5$ nodes without any node having access to the data of the rest of the nodes and any kind of data reuse (that has been previously processed), is not allowed (due to its online nature).

With respect to the denoising procedure, there are several approaches one could follow, [1]. Since our task here is not to optimize such an approach but to offer the means for a fair comparison, we adopted a simplified version of the denoising method proposed in [1], [24], [6]. In particular, all possible overlapped patches of size 8×8 are considered, counting to $(256 - 8 + 1)^2 = 62001$ patches in total, which are going to be the data used for the learning of the dictionary. Specifically, the patches are extracted from the noisy image, then each one of them is vectorized in lexicographic order and the resulted data vectors are gathered in a (64×62001) matrix for the K–SVD case or are randomly distributed to the N nodes for the proposed algorithm case. In the sequel, a dictionary for each one the competing methods is trained. Finally, each one of the patches is denoised separately based on its sparse representation on the estimated dictionaries. Note that for the whole image each one of its pixels apparently belongs to more than one patch (due to the fact that they are overlapped). Accordingly, the final denoised

Fig. 2. The original image, the noisy one, and the denoised image.

Table III. Image Denoising

PSNR (noisy Image)	KSVD	OnDiRLS
16	22.9	24
18	25.5	26.2
22	28.8	29.1
28	33.8	32.5

value of each of the image pixels is computed as the average of the corresponding pixel of all the associated patches [1].

The original picture, the noisy one, and the one resulting after the denoising process of the proposed algorithm can be seen in Fig. 2. Furthermore, the PSNR for the noisy and the recovered images is given in Table III. There, it can be observed that the performance of the OnDiRLS is slightly better compared to that of the the K–SVD in the lower PSNR scenarios and that the K–SVD leads to better denoising in the high PSNR regime. It appears that the distributed operation of the proposed algorithm renters it relatively robust to the noise.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a novel algorithm for online distributed dictionary learning is proposed. At each iteration step, in which a datum is received at each node of the network, the LARS algorithm computes the sparse coding vector, exploiting the most recent dictionary estimate and the new observation. In the sequel, the nodes of the network cooperate, via the DRLS algorithm, in order to estimate the common dictionary. The performance of the proposed scheme is validated both with synthetic and real data.

Future research focuses on the task of developing low–complexity versions of the algorithm for online distributed dictionary learning.

VII. REFERENCES

[1] M. Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer, 2010.

[2] S. Theodoridis, Y. Kopsinis, and K. Slavakis, *Sparsity-aware learning and compressed sensing: An overview*. Academic press, 2014.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[4] M. Aharon, M. Elad, and A. Bruckstein, "K–svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[5] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.
[6] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 2015.
[7] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.
[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, “Tools for privacy preserving distributed data mining,” *ACM SIGKDD Explorations*, vol. 4, no. 2, pp. 28–34, 2002.
[9] I. Tosic and P. Frossard, “Dictionary learning,” *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.
[10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.
[11] K. Skretting and K. Engan, “Recursive least squares dictionary learning algorithm,” *Signal Processing, IEEE Transactions on*, vol. 58, no. 4, pp. 2121–2130, 2010.
[12] J. Chen, Z. J. Towfic, and A. H. Sayed, “Dictionary learning over distributed models,” *arXiv preprint arXiv:1402.1515*, 2014.
[13] P. Chainais and C. Richard, “Distributed dictionary learning over a sensor network,” *arXiv preprint arXiv:1304.3568*, 2013.
[14] J. Liang, M. Zhang, X. Zeng, and G. Yu, “Distributed dictionary learning for sparse representation in sensor networks,” *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2528–2541, June 2014.
[15] K. Slavakis and G. B. Giannakis, “Online dictionary learning from big data using accelerated stochastic approximation algorithms,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 16–20.
[16] H. Lee, A. Battle, R. Raina, and A. Y. Ng, “Efficient sparse coding algorithms,” in *Advances in neural information processing systems*, 2006, pp. 801–808.
[17] G. Mateos and G. B. Giannakis, “Distributed recursive least-squares: Stability and performance analysis,” *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3740–3754, 2012.
[18] G. Mateos, I. D. Schizas, and G. B. Giannakis, “Distributed recursive least-squares for consensus-based in-network adaptive estimation,” *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.
[19] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, “Consensus in ad hoc wsns with noisy linkspart i: Distributed estimation of deterministic signals,” *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.
[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Belmont, MA: Athena–Scientific, 1999.
[21] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, “Least angle regression,” *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[22] E. J. Candès and M. B. Wakin, “An introduction to compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.
[23] M. Mardani, G. Mateos, and G. B. Giannakis, “Subspace learning and imputation for streaming big data matrices and tensors,” *arXiv preprint arXiv:1404.4667*, 2014.
[24] J. Mairal, M. Elad, and G. Sapiro, “Sparse representation for color image restoration,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.