



**HAL**  
open science

## Learning a common dictionary over a sensor network

Pierre Chainais, Cédric Richard

► **To cite this version:**

Pierre Chainais, Cédric Richard. Learning a common dictionary over a sensor network. CAMSAP 2013, Dec 2013, Saint-Martin, France. pp.1-4. hal-00923742

**HAL Id: hal-00923742**

**<https://hal.science/hal-00923742v1>**

Submitted on 4 Jan 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Learning a common dictionary over a sensor network

Pierre Chainais

LAGIS UMR CNRS 8219 - Ecole Centrale Lille  
INRIA-Lille Nord Europe, SequeL, France  
Email: pierre.chainais@ec-lille.fr

Cédric Richard

Laboratoire Lagrange UMR CNRS 7293,  
University of Nice Sophia-Antipolis, France  
Email: cedric.richard@unice.fr

**Abstract**—We consider the problem of distributed dictionary learning, where a set of nodes is required to collectively learn a common dictionary from noisy measurements. This approach may be useful in several contexts including sensor networks. Diffusion cooperation schemes have been proposed to solve the distributed linear regression problem. In this work we focus on a diffusion-based adaptive dictionary learning strategy: each node records independent observations and cooperates with its neighbors by sharing its local dictionary. The resulting algorithm corresponds to a distributed alternate optimization. Beyond dictionary learning, this strategy could be adapted to many matrix factorization problems in various settings. We illustrate its efficiency on some numerical experiments.

## I. INTRODUCTION

In a variety of contexts, huge amounts of high dimensional data are recorded from multiple sensors. When sensor networks are considered, it is desirable that computations be distributed over the network rather than centralized in some fusion unit. Indeed, centralizing all measurements lacks robustness - a failure of the central node is fatal - and scalability due to the needed energy and communication resources. In distributed computing, every node communicates with its neighbors only and processing is carried out by every node in the network. Another important remark is that relevant information from the data usually lives in a space of much reduced dimension compared to the physical space. The extraction of this relevant information calls for the identification of some adapted sparse representation of the data. Sparsity is an important property which favors the identification of the main components that are characteristic of the data. Each observation is then described by a sparse subset of atoms taken from a redundant dictionary. We study the problem of dictionary learning distributed over a sensor network in a setting where a set of nodes is required to collectively learn an adaptive sparse representation from independent observations. Learning an adaptive representation of the data is useful for many tasks such as storing, transmitting or analyzing the data to understand its content. A basic dictionary can be obtained by using a Principal Component Analysis (PCA) also known as Karhunen-Loève decomposition in signal processing. However the number of atoms of such a decomposition is limited to the dimension of the data space. Many recent works have shown the interest of learning a *redundant dictionary* allowing for a *sparse representation* of the data, see [1] for an up-to-date review. Furthermore, the problem of dictionary learning belongs to the more general family of matrix factorization problems that appears in a host of applications. In this paper, we consider the situation where a set of connected nodes independently record data from observations of the same kind of physical system: each obser-

vation is assumed to be described by a sparse representation using a common dictionary for all sensors. For instance, a set of cameras observe the same kind of scenes or a set of microphones records the same kind of sound environment. The dictionary learning and the matrix factorization problems are connected to the linear regression problem. Let us consider a set of observations described by a data matrix  $\mathbf{S}$  where each column corresponds to one observation. Assume that  $\mathbf{S} = \mathbf{D}\mathbf{X}$ . If either the coefficients  $\mathbf{X}$  (resp. the dictionary  $\mathbf{D}$ ) are known, the estimation of the dictionary (resp. the coefficients) knowing  $\mathbf{D}$  (resp.  $\mathbf{X}$ ) is a linear regression problem. Several recent works have proposed efficient solutions to the problem of least mean square (LMS) distributed linear regression, see [2] and references therein. The main idea is to use a so-called *diffusion strategy*: each node  $n$  carries out its own estimation  $\mathbf{D}_n$  of the same underlying linear regression vector  $\mathbf{D}$  but can communicate with its neighbors as well. The information provided to some node by its neighbors is taken into account according to weights interpreted as diffusion coefficients. Under some mild conditions, the performance of such an approach in terms of mean squared error is similar to that of a centralized approach [3]. Let  $\mathbf{D}_c$  the centralized estimate which uses all the observations at once. It can be shown that the error  $\mathbf{E}\|\mathbf{D}_n - \mathbf{D}\|_2$  of the distributed estimate is of the same order as  $\mathbf{E}\|\mathbf{D}_c - \mathbf{D}\|_2$ : diffusion networks match the performance of the centralized solution. Our work gives strong indication that the classical dictionary learning technique based on block coordinate descent on the dictionary  $\mathbf{D}$  and the coefficients  $\mathbf{X}$  can be adapted to the distributed framework by adapting the diffusion strategy mentioned above. Our numerical experiments also strongly support this idea. The theoretical analysis is the subject of ongoing work. Note that solving this type of matrix factorization problems is really at stake since it corresponds to many inverse problems: denoising, adaptive compression, recommendation systems... A distributed approach is highly desirable both for use in sensor networks and for parallelization of numerically expensive learning algorithms.

The paper is organized as follows. Section II formulates the problem of distributed dictionary learning. Section III recalls about dictionary learning techniques based on block coordinate descent approaches and their generalizations. Section IV presents the diffusion strategy for distributed dictionary learning. Section V shows some numerical experiments and results. Section VI points to main claims and prospects.

## II. PROBLEM FORMULATION

Consider  $N$  nodes over some region. In the following, boldfaced letters denote column vectors, and capital letters denote matrices. The node  $n$  takes  $q_n$  measurements  $\mathbf{y}_n(i)$ ,

$1 \leq i \leq q_n$  from some physical system. All the observations are assumed to originate from independent realizations  $\mathbf{s}_n(i)$  of the same underlying stochastic source process  $\mathbf{s}$ . Each measurement is a noisy measurement

$$\mathbf{y}_n(i) = \mathbf{s}_n(i) + \mathbf{z}_n(i) \quad (1)$$

where  $\mathbf{z}$  denotes the usual i.i.d. Gaussian noise with covariance matrix  $\Sigma_n = \sigma_n^2 \mathbb{I}$ . Our purpose is to learn a redundant dictionary  $\mathbf{D}$  which carries the characteristic properties of the data. This dictionary must yield a sparse representation of  $\mathbf{s}$  so that:

$$\forall n, \quad \mathbf{y}_n(i) = \underbrace{\mathbf{D}\mathbf{x}_n(i)}_{\mathbf{s}_n(i)} + \mathbf{z}_n(i) \quad (2)$$

where  $\mathbf{x}_n(i)$  features the coefficients  $x_{nk}(i)$  associated to the contribution of atom  $\mathbf{d}_k$ , the  $k$ -th column in the dictionary matrix  $\mathbf{D}$ , to  $\mathbf{s}_n(i)$ . The sparsity of  $\mathbf{x}_n(i)$  means that only few components of  $\mathbf{x}_n(i)$  are non zero.

We consider the situation where a unique dictionary  $\mathbf{D}$  generates the observations at all nodes. On the contrary, observations will not be shared between nodes (this is one potential generalization). Our purpose is to learn this dictionary in a distributed manner thanks to in-network computing only, see section IV. As a consequence, each node will locally estimate a local dictionary  $\mathbf{D}_n$  thanks to i) its observations  $\mathbf{y}_n$  and ii) communication with its neighbors. The neighborhood of node  $n$  will be denoted by  $\mathcal{N}_n$ , including node  $n$  itself. The number of nodes connected to node  $n$  is the degree  $\nu_n$ .

### III. DICTIONARY LEARNING STRATEGIES

#### A. Problem formulation

Various approaches to dictionary learning have been proposed [1]. Usually, in the centralized setting, the  $q$  observations are denoted by  $\mathbf{y}(i) \in \mathbb{R}^p$  and grouped in a matrix  $\mathbf{Y} = [\mathbf{y}(1), \dots, \mathbf{y}(q)]$ . As a consequence,  $\mathbf{Y} \in \mathbb{R}^{p \times q}$ . The dictionary (associated to some linear transform) is denoted by  $\mathbf{D} \in \mathbb{R}^{p \times K}$ : each column is one atom  $\mathbf{d}_k$  of the dictionary. We gather the coefficients associated to observations in a single matrix  $\mathbf{X} = [\mathbf{x}(1), \dots, \mathbf{x}(q)]$ . We will consider learning methods based on block coordinate descent or alternate optimization on  $\mathbf{D}$  and  $\mathbf{X}$  with a sparsity constraint on  $\mathbf{X}$  [1], [4], [5].

The data is represented as the sum of a linear combination of atoms and a noise term  $\mathbf{Z} \in \mathbb{R}^{p \times q}$ :

$$\mathbf{Y} = \mathbf{D}\mathbf{X} + \mathbf{Z} \quad (3)$$

Dictionary learning is an ill-posed *matrix factorization* problem. The dictionary is potentially redundant with  $K \gg p$ . Some modeling is necessary to constrain the set of possible solutions. Various conditions can be considered (non-negativity, orthogonality, ...). In general, a dictionary is considered as *adapted* to the data if each observation  $\mathbf{y}(i)$  can be described by a small number of coefficients  $\mathbf{x}(i)$ . One usually searches for a *sparse representation* and imposes the sparsity of  $\mathbf{X}$ .

#### B. Learning a redundant dictionary for sparse representation

The redundancy of the dictionary and the sparsity of the coefficients are complementary. The extreme case would be the one where the dictionary contains each one of the

true data underlying the noisy observation so that only one non zero coefficient in  $\mathbf{x}(i)$  would be sufficient to describe the observation  $\mathbf{y}(i)$ . Then we would have  $K = q$  and  $\mathbf{X}$  would be maximally sparse (only 1 non zero coefficient per observation). This dictionary would not be very interesting since its generalization power would be very limited. A good dictionary offers a compromise between fidelity to the training set and its ability to generalize. The size of the dictionary is often chosen a priori so that  $K > p$  to ensure some redundancy and  $K < q$  to ensure it captures some common features shared by the data. For instance, when working on image patches of size  $8 \times 8$  (the data lives in dimension  $p = 64$ ), it is typically proposed to learn dictionaries of size 256 or 512 [1], [5].

In the classical setting, the noise is usually assumed to be Gaussian i.i.d. so that the reconstruction error is measured by the L2-norm. Sparsity of the coefficient matrix is imposed through a L0 relaxed to L1-penalization in the mixed optimization problem:

$$(\mathbf{D}, \mathbf{X}) = \underset{(\mathbf{D}, \mathbf{X})}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_2^2 + \lambda \|\mathbf{X}\|_1 \quad (4)$$

Under some mild conditions, this problem is known to provide a solution to the L0-penalized problem (ideally we would prefer to directly solve the L0-penalized problem) [6].

#### C. Block coordinate descent

One way to solve problem (4) is to use block coordinate descent [4], that is alternate optimization on  $\mathbf{X}$  and  $\mathbf{D}$ . There are several possibilities to do this [5]. For instance, after some initialization, one may use gradient descents on  $\mathbf{X}$  and  $\mathbf{D}$  [7]. Such approaches are attractive since we know that linear regression by gradient descent can be translated in the distributed framework [2]. One possible choice is the Basis Pursuit algorithm. At each step, Iterated Soft Thresholding [6] estimates  $\mathbf{X}$  by iterating the following gradient descent and soft thresholding steps over  $s$  and  $t$ :

$$1) \quad \mathbf{X}^{(s,t+1/2)} = \mathbf{X}^{(s,t)} + \lambda \mu \mathbf{D}^{(s,t)T} [\mathbf{Y} - \mathbf{D}^{(s)} \mathbf{X}^{(s,t)}]$$

$$2) \quad \mathbf{X}^{(s,t+1)} = \operatorname{SoftThreshold}_{\lambda \mu}(\mathbf{X}^{(s,t+1/2)})$$

Then we update  $\mathbf{X}^{(s+1)} = \mathbf{X}^{(s,T)}$  after  $T$  (typically 30 or 40) iterated soft thresholding. Note that one must have  $\mu \in \left(0, \frac{2}{\|\|\mathbf{D}\|\|^2}\right)$  where  $\|\|\cdot\|\|$  denotes the spectral norm. One may alternatively turn to Orthogonal Matching Pursuit for instance [5]. Then the dictionary  $\mathbf{D}^{(s+1)}$  can be updated knowing  $\mathbf{X}^{(s+1)}$ , using a simple gradient descent:

$$\tilde{\mathbf{D}}^{(s+1)} = \mathbf{D}^{(s)} + \eta \left[ \mathbf{Y} - \mathbf{D}^{(s)} \mathbf{X}^{(s+1)} \right] \mathbf{X}^{(s+1)T} \quad (5)$$

which tends to minimize  $\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2$  with respect to  $\mathbf{D}$  for  $0 < \eta < 2/\|\|\mathbf{X}\|\|^2$ . One can also use a pseudo-inverse [8]:

$$\begin{aligned} \tilde{\mathbf{D}}^{(s+1)} &= \underset{\mathbf{D}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\mathbf{X}^{(s+1)}\|_2^2 \\ &= \mathbf{Y}\mathbf{X}^{(s+1)T} \cdot \left( \mathbf{X}^{(s+1)} \mathbf{X}^{(s+1)T} \right)^{-1} \end{aligned} \quad (6)$$

The dictionary is then normalized:

$$\forall \leq k \leq K, \mathbf{d}_k = \frac{1}{\|\tilde{\mathbf{d}}_k\|_2} \tilde{\mathbf{d}}_k. \quad (7)$$

Various methods like FOCUS [9], K-SVD [5] or the majorization method [10] are not discussed here for sake of

brevity. At first, gradient descent methods are easier to adapt to the distributed diffusion strategy. The adaptation of K-SVD is also rather straightforward. The comparison of performances of various methods is under study.

#### IV. DISTRIBUTED DICTIONARY LEARNING

##### A. Diffusion strategies for distributed estimation

This section presents one particular effective diffusion strategy to solve LMS distributed estimation problems [2], [3]. We focus on the Adapt-Then-Combine (ATC) strategy. The Adapt-Then-Combine (ATC) strategy aims at solving the problem of a *scalar* least mean squares linear regression over a sensor network. Observations  $y_n(i)$  are assumed to arrive sequentially at consecutive instants  $i$ . In the usual setting [2], each sensor records both a noisy *scalar* measurement  $y_n(i) \in \mathbb{R}$  and a set of coefficients  $\mathbf{x}_n(i)$  under the assumption

$$y_n(i) = \mathbf{w}_o^T \mathbf{x}_{n,i} + z_n(i). \quad (8)$$

The objective is to collectively estimate  $\mathbf{w}_o$ . The purpose of ATC is that the sensors  $\{n : 1 \dots N\}$  yield estimates  $\mathbf{w}_n$  of the common underlying regression vector  $\mathbf{w}_o$  from observations  $\{y_n(i); \mathbf{x}_n(i)\}$  at time  $i$ . The cost function under the assumption of Gaussian noise is:

$$J(\mathbf{x}, \mathbf{w}) = \sum_{n=1}^N \underbrace{\mathbf{E}|y_n(i) - \mathbf{w}^T \mathbf{x}_{n,i}|^2}_{J_{loc}(\mathbf{x}_{n,i}, \mathbf{w})} \quad (9)$$

Let  $\mathbf{A}, \mathbf{C} \in (\mathbb{R}^+)^{N \times N}$  two matrices such that:

$$\begin{cases} c_{\ell,n} = a_{\ell,n} = 0 \text{ if } \ell \notin \mathcal{N}_n, \\ \mathbf{1}^T \mathbf{C} = \mathbf{1}^T, \mathbf{C} \mathbf{1} = \mathbf{1}, \mathbf{1}^T \mathbf{A} = \mathbf{1}^T \end{cases} \quad (10)$$

where  $\mathbf{1}$  is a column vector of ones. The ATC algorithm consists of 2 steps:

$$\psi_{n,i} = \mathbf{w}_{n,i-1} + \quad (\text{Adapt}) \quad (11)$$

$$\mu_n^w \sum_{\ell \in \mathcal{N}_n} c_{\ell,n}^w \underbrace{\mathbf{x}_{\ell,i-1} [y_\ell(i) - \mathbf{w}_{n,i-1}^T \mathbf{x}_{\ell,i-1}]}_{\nabla_w J_{loc}(\mathbf{x}_{\ell,i-1}, \mathbf{w}_{n,i-1})}$$

$$\mathbf{w}_{n,i} = \sum_{\ell \in \mathcal{N}_n} a_{\ell,n}^w \psi_{\ell,i} \quad (\text{Combine}) \quad (12)$$

The ATC algorithm can be seen as a distributed gradient descent where each sensor tries to estimate  $\mathbf{w}_o$  as  $\mathbf{w}_{n,i}$  by exploiting its own measurement  $y_n(i)$  as well as information shared with its neighbors. Eq. (11) is the *Adapt* or *incremental* step, eq. (12) is the *Combine* or *diffusion* step which averages estimates from neighbors of node  $n$ . As a consequence, a local (possibly averaged if  $\mathbf{C} \neq \mathbb{I}$ ) gradient with respect to  $\mathbf{w}$  is computed at each node. An intermediate updated version of the local estimate of  $\mathbf{w}_o$  denoted by  $\psi_{n,i}$  is then obtained. The final estimate at each node is a local average of neighboring intermediate estimates weighted by matrix  $\mathbf{A}$ .

In the sequel, we will focus on the case where observations are not shared between nodes so that  $\mathbf{C} = \mathbb{I}$ . Various choices can be considered for  $\mathbf{A}$ . In the numerical experiments below we typically work with either some a priori fixed matrix  $\mathbf{A}$  or with the relative degree variance ( $\nu_\ell =$  degree of node  $\ell$ ):

$$a_{\ell,n} = \frac{\nu_\ell \sigma_\ell^2}{\sum_{m \in \mathcal{N}_n} \nu_m \sigma_m^2} \quad (13)$$

---

Initialize  $\mathbf{D}_{n,0}, \forall n$  (random subset of  $K$  observations  $y_n(i)$ ).  
Given a matrix  $\mathbf{A}$  satisfying (10),  $i = 0$ ,

**Repeat until convergence of**  $(\mathbf{D}_{n,i}, \mathbf{X}_{n,i})_{n=1:N}$

**For each node**  $n$  **repeat:**

1) Optimization w.r.t.  $\mathbf{X}_{n,i}$  (sparse coding):  
Given the dictionary  $\mathbf{D}_{n,i}$ , the coefficients  $\mathbf{X}_{n,i}$  are estimated using a sparse coding method (Basis Pursuit, OMP, FOCUSS,...)

2) Optimization w.r.t.  $\mathbf{D}_{n,i}$  (dictionary) e.g. by gradient descent:

$$\begin{cases} \psi_{n,i+1} = \mathbf{D}_{n,i} + \mu_n^D (\mathbf{Y}_n - \mathbf{D}_{n,i} \mathbf{X}_{n,i}) \mathbf{X}_{n,i}^T \\ \mathbf{D}_{n,i+1} = \sum_{\ell \in \mathcal{N}_k} a_{\ell,n}^D \psi_{\ell,i} \text{ (diffusion)} \end{cases}$$

and  $\forall 1 \leq k \leq K, \mathbf{d}_k \leftarrow \frac{\mathbf{d}_k}{\|\mathbf{d}_k\|_2}$  (normalization)

$\psi_{n,i+1}$  can also be updated by MOD or K-SVD,...at node  $n$ .

**EndFor** ( $n$ )

$i \leftarrow i + 1$

**EndRepeat**

---

TABLE I. ALGORITHM 1: ATC FOR SPARSE DICTIONARY LEARNING.

The performance analysis of this ATC diffusion strategy and some other variants can be found in [3]. The mean-square error of the ATC estimate of  $\mathbf{w}_o$  is similar to that of the centralized version (which would see all the observations at once). This diffusion strategy is very powerful to deal with a distributed solution to a linear regression problem. Let us emphasize that in this setting each observation is made of a couple  $(y_n, \mathbf{x}_n)$  where  $y_n$  is a scalar. In the dictionary learning problem, only the *vector*  $\mathbf{y}_n$  will be observed and both the dictionary (therefore  $\mathbf{D}$  in place of  $\mathbf{w}_o$ ) and the coefficient  $\mathbf{x}_n$  are to be jointly estimated: this is a factorization problem.

##### B. Distributed alternate optimization for dictionary learning

The ATC diffusion strategy for distributed estimation described above originates the following approach to distributed alternate optimization for dictionary learning. Diffusion is ensured by the communication between nodes sharing their dictionary estimate with neighbors in  $\mathcal{N}_n$ . We emphasize that, in the present setting, observations will be the *vectors* (not only scalar)  $\mathbf{y}_n(i), i = 1 \dots q_n$  at node  $n$ . Moreover, observations are taken simultaneously at each node, not sequentially, so that a whole data matrix  $\mathbf{Y}_n$  is assumed to be available at node  $n$ . Here index  $i$  stands for iterations. The case where data arrive sequentially at each node can also be dealt with at the price of a natural adaptation of the present approach. Note that the  $\mathbf{x}_{n,i}$  are not known anymore: each node must estimate both its local dictionary  $\mathbf{D}_n$  and the coefficients  $\mathbf{X}_n$  which describe observations  $\mathbf{Y}_n = \mathbf{D}_n \mathbf{X}_n + \mathbf{Z}_n$ . At each iteration  $i$ , only the local dictionary estimates  $\mathbf{D}_{n,i}$  are assumed to be shared between neighbors, not observations, so that  $\mathbf{C} = \mathbb{I}$  in eq. (10). The algorithm is summarized in Tab. I. In summary, sparse representations are computed locally. Then each node updates its dictionary as a function of its local observations  $\mathbf{Y}_n$  (Adapt step) and its neighbors' dictionaries (Combine step). Based on known results for the ATC strategy in its usual setting, we expect that Algorithm 1 above converges to an accurate estimate of the common underlying dictionary  $\mathbf{D}$ . Next section supports this intuition thanks to numerical experiments on images.

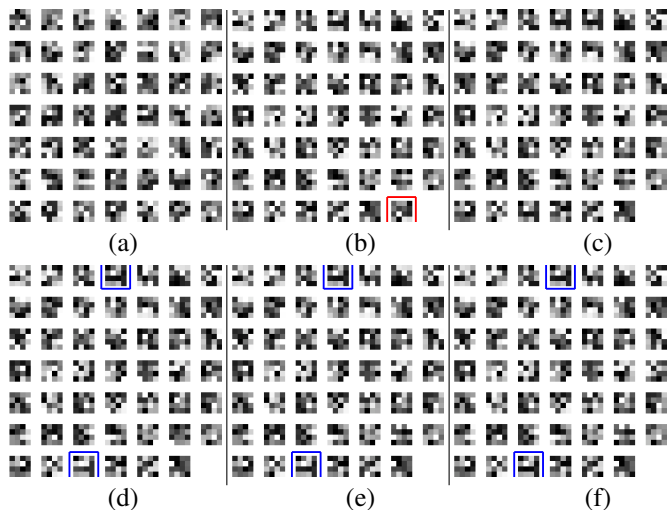


Fig. 1. (a) Examples of patches; (b) dictionary of 48 atoms for synthesis (i.i.d. random pixels), the red square identifies the atom which has not been learnt by the network; (c) dictionary learnt from the usual centralized learning (42 original atoms retrieved at 0.99 level); (d-f) dictionaries learnt at 3 different nodes (45 atoms retrieved at 0.99 level) where blue squares identify duplicates. In (b) & (c), atoms have been suitably reordered to make comparisons easier.

## V. NUMERICAL EXPERIMENTS & RESULTS

We present some numerical experiments to illustrate the relevance and efficiency of our approach. Below we show the results obtained on a dataset built from a redundant dictionary of 48 atoms of dimension 16 corresponding to image patches of size  $4 \times 4$ . This dictionary was obtained from i.i.d. uniform random variables over  $[-0.5, 0.5]$ . Atoms have been centered so that  $\forall k, \langle \mathbf{d}_k \rangle = 0$  and normalized so that  $\|\mathbf{d}_j\|_2 = 1$ , where  $\mathbf{d}_k$  is the  $k$ -th column (atom) of  $D$ . We emphasize that this dictionary is very redundant and cannot be orthogonal. Each data  $y_n(i)$  is the linear combination of 3 atoms with i.i.d. coefficients uniformly distributed over  $[-0.5, 0.5]$ , fig. 1(a). We have also generated similar datasets with additive Gaussian noise with SNR levels of 30dB down to 10dB (not shown for sake of brevity). We consider the simple situation of a set of 4 nodes in a symmetrically connected network. Thus we used a symmetric matrix  $\mathbf{A}$  such that:

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.2 & 0 & 0.2 \\ 0.2 & 0.6 & 0.2 & 0 \\ 0 & 0.2 & 0.6 & 0.2 \\ 0.2 & 0 & 0.2 & 0.6 \end{bmatrix} \quad (14)$$

Note that nodes are not even directly connected one to all the others. Fig. 1(d-f) show that all the nodes have consistently learnt the same dictionary of  $4 \times 4$  patches. Let us emphasize that these dictionaries are consistent, in the sense that no local reordering was necessary at any step. All the nodal dictionaries  $\mathbf{D}_n$  are close to the same common dictionary  $\mathbf{D}$ . It appears that the mean-square error over all estimates is similar to that obtained from the centralized dictionary learning procedure of section III. Therefore even though each node locally solves a matrix factorization problem from a particular disjoint subset of observations, the same common dictionary is (approximately) identified. This is made possible by the diffusion principle which relies on a simple communication between neighbors only. In this example, 45 atoms of the initial dictionary (48 atoms) were recovered with  $\langle \mathbf{d}_j, \mathbf{d}_j^{(o)} \rangle \geq 0.99$ .

Only 1 atom is learnt twice (duplicate) and 1 was not recovered at a level of 0.7, fig. 1(b). For noisy data, results remain similar for  $\text{SNR} \geq 15\text{dB}$ . We repeated this estimation procedure for many initializations which show that the performance of distributed learning is at least similar to centralized learning.

## VI. CONCLUSION & PROSPECTS

As a conclusion, we have presented an original algorithm which solves the problem of distributed dictionary learning over a sensor network. This is made possible thanks to a diffusion strategy which permits local communication between neighbors. Connected nodes exchange their local dictionaries estimated from disjoint subsets of data. This algorithm adapts usual dictionary learning techniques for sparse representation to the context of in-network computing. Numerical experiments illustrate the relevance of our approach to very efficiently learn a redundant dictionary for sparse coding in a distributed framework. Several improvements and generalizations can be considered. The choice of one of the many available sparse coding methods is crucial to get better dictionary estimates. The optimization of communication coefficients between nodes may be of some help as well. This approach to the general problem of distributed matrix factorization opens the way towards many prospects and applications. Moreover, as far as computational complexity is concerned, distributed parallel implementations are a potentially interesting alternative to online learning techniques [11]. We may even consider a dynamical context where observations arrive over time on a sensor network so that the dictionary would be learnt dynamically at each node in a collaborative manner.

## REFERENCES

- [1] I. Tomic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, march 2011.
- [2] F. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. on Sig. Pro.*, vol. 58, no. 3, pp. 1035–1048, 2010.
- [3] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Trans. on Sig. Pro.*, vol. 60, no. 10, pp. 5107–5124, 2012.
- [4] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. of Opt. Th. and App.*, vol. 109, pp. 475–494, 2001.
- [5] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Sig. Pro.*, vol. 54, no. 11, pp. 4311–4322, nov. 2006.
- [6] J.-L. Starck, F. Murtagh, and J. Fadili, *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity*. Cambridge University Press, 2010.
- [7] B. Olshausen and D. Field, "Emergence of simple-cell receptive properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.
- [8] K. Engan, S. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *Proc. of IEEE ICASSP*, vol. 5, pp. 2443–2446, 1999.
- [9] J. Murray and K. Kreutz-Delgado, "An improved focuss-based learning algorithm for solving sparse linear inverse problems," in *Proc. of 35th Asilomar Conf. on Sig. Sys. & Comp.*, vol. 1, nov. 2001, pp. 347–351.
- [10] M. Yaghoobi, T. Blumensath, and M. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Trans. on Sig. Pro.*, vol. 57, no. 6, pp. 2178–2191, 2009.
- [11] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.