2014 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2014, REIMS, FRANCE

# DICTIONARY LEARNING OVER LARGE DISTRIBUTED MODELS VIA DUAL-ADMM STRATEGIES

*Zaid J. Towfic, Jianshu Chen, and Ali H. Sayed*

Electrical Engineering Department
University of California, Los Angeles

## ABSTRACT

We consider the problem of dictionary learning over large scale models, where the model parameters are distributed over a multi-agent network. We demonstrate that the dual optimization problem for inference is better conditioned than the primal problem and that the dual cost function is an aggregate of individual costs associated with different network agents. We also establish that the dual cost function is smooth, strongly-convex, and possesses Lipschitz continuous gradients. These properties allow us to formulate efficient distributed ADMM algorithms for the dual inference problem. In particular, we show that the proximal operators utilized in the ADMM algorithm can be characterized in closed-form with linear complexity for certain useful dictionary learning scenarios.

***Index Terms***— dictionary learning, augmented Lagrangian, ADMM, dual decomposition, consensus strategy, diffusion strategy

## 1. INTRODUCTION

Dictionary learning is a procedure that enables the representation of input feature vectors as sparse linear combinations of dictionary elements called atoms. The technique can be leveraged to solve some useful problems such as image denoising [1, 2], dimensionality-reduction [3, 4], bi-clustering [5], feature-extraction and classification [6], and novel document detection [7]. The solution usually involves two steps: (i) an inference step, where the coefficients used to mix the dictionary elements are computed via a sparse coding calculation and (ii) a dictionary update step, where the dictionary elements are updated in order to improve future reconstructions.

In many dictionary learning tasks, and especially those involving big data applications, the dictionary size has been increasing steadily. As a result, it is becoming prohibitive to execute the dictionary learning task on a single processor or at a single physical location since the data may be stored in a distributed manner. This difficulty motivates us to develop a *model*-distributed dictionary learning problem in which different agents in a connected network maintain separate parts of the dictionary and they do not need to share their dictionary elements or private information. In addition, we consider a broad scenario where we will not assume that the feature vectors are available at each agent, but only to a subset of the agents [8].

In comparison to our previous work in [8, 9], where strategies of the consensus and diffusion type were considered, we shall instead employ in this article the class of ADMM strategies (see, e.g., [10–12] and the references therein) to solve the inference step. The motivation for doing so is the following. We will explain in Sec. 4 that for some special penalty functions, but not always, the minimization component of the inference step (corresponding to future expression (32)) can be solved in *closed-form*. This observation will imply that the computational complexity of the ADMM implementation for these special cases will become lower than the complexity of solution techniques that rely on repeated iterations, which would be needed for scenarios involving more general penalty functions (see, e.g., [8]). However, as explained after (35) further ahead, for these more general scenarios and when no closed-form solutions are possible for (32), an ADMM implementation will end up involving three separate time-scales, whereas diffusion or consensus strategies will involve only two time-scales. Depending on the complexity of the iterations in the third time-scale of the solution, this implementation can end up being more or less complex than consensus or diffusion implementations. Regardless, it is important to note that having a third time-scale necessarily requires faster processing units at the agents for ADMM than would be required by consensus or diffusion strategies.

**Notation**: We use boldface letters to represent random quantities. The random variable $\boldsymbol{x}$ may have a realization $x$, which is denoted in plain font.

## 2. PROBLEM FORMULATION

Consider the following dictionary learning problem over a connected network of $N$ agents:

$$\min_{W} \quad \mathbb{E}\Big[f\left(\boldsymbol{x}_t - W\boldsymbol{y}_t^o\right) + h_y(y_t^o)\Big] + h_W(W) \tag{1}$$

$$\text{s.t.} \quad W \in \mathbb{W} \tag{2}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator, $\boldsymbol{x}_t \in \mathbb{R}^M$ is a random input data vector available at time $t$, and $W \in \mathbb{R}^{M\times N}$ is a dictionary matrix. The set $\mathbb{W}$ denotes a constraint set on $W$, and $\boldsymbol{y}_t^o$ is the solution to the following inference problem for an input data realization $x_t$ and for a given dictionary matrix $W$:

$$y_t^o \triangleq \arg\min_{y} \left[f(x_t - Wy) + h_y(y)\right] \tag{3}$$

We assume that the penalty functions $f(\cdot)$ and $h_W(\cdot)$ are convex while the function $h_y(y)$ is strongly-convex. Observe that the dictionary learning task consists of two steps: (i) the inference step involving sparse coding for a given $x_t$ and $W$ in (3), and (ii) the dictionary update step in (1)–(2).

The $q-$th column of $W$ is called an *atom*, and the constraint set $\mathbb{W}$ may, for example, constrain the norm of each atom in the dictionary, say, as [2]:

$$\mathbb{W} = \{W : \ \|[W]_{:,q}\|_2 \leq 1\} \tag{4}$$

This work was supported in part by NSF grants CCF-1011918 and ECCS-1407712.

978-1-4799-3694-6/14/$31.00 

Alternatively, if we are interested in the non-negative factorization [2] and topic modeling [7] problems, the constraint set may instead be defined as

$$\mathbb{W} = \{W : \ \|[W]_{:,q}\|_2 \leq 1, \ W \succeq 0\} \tag{5}$$

where the notation $W \succeq 0$ indicates that each entry of the matrix $W$ is nonnegative.

We study the case when the atoms are distributed across the network. Specifically, we partition the dictionary matrix $W$ into block columns and the vector $y$ into row entries according to:

$$W = \begin{bmatrix} W_1 & \cdots & W_N \end{bmatrix}, \qquad y = \mathrm{col}\{y_1, \ \ldots, \ y_N\} \tag{6}$$

where each agent $k$ in the network is assumed to have access only to $W_k \in \mathbb{R}^{M \times N_k}$ and $y_k \in \mathbb{R}^{N_k \times 1}$. Observe that

$$\sum_{k=1}^{N} N_k = N \tag{7}$$

It is also assumed that the feasible set $\mathbb{W}$ can be decomposed as the intersection of $N$ independent convex sets, $\mathbb{W} = \mathbb{W}_1 \cap \ldots \cap \mathbb{W}_N$, where $\mathbb{W}_k$ places constraints on the submatrix $W_k$. The regularization functions $h_y(y)$ and $h_W(W)$ are assumed to be decomposable as well:

$$h_y(y) = \sum_{k=1}^{N} h_{y_k}(y_k), \quad h_W(W) = \sum_{k=1}^{N} h_{W_k}(W_k) \tag{8}$$

## 3. DISTRIBUTED DICTIONARY LEARNING

### 3.1. A well conditioned optimization problem

In our earlier works [8, 9], and following duality arguments, we explained that problem (3) can be transformed into another form that is more amenable to distributed implementations. Specifically, we showed that the solution of (3) can be obtained from the minimization of the following sum-of-cost form over an auxiliary vector variable $\nu \in \mathbb{R}^{M \times 1}$:

$$\min_{\nu} \quad \sum_{k=1}^{N} J_k(\nu; x_t) \tag{9a}$$

$$\text{s.t.} \quad \nu \in \mathbb{V}_f \tag{9b}$$

The individual costs that appear in (9a) are given by:

$$J_k(\nu; x_t) \triangleq \begin{cases} -\dfrac{\nu^T x_t}{|\mathcal{N}_I|} + \dfrac{1}{N} f^\star(\nu) + h_{y_k}^\star(W_k^T \nu), & k \in \mathcal{N}_I \\ \dfrac{1}{N} f^\star(\nu) + h_{y_k}^\star(W_k^T \nu), & k \notin \mathcal{N}_I \end{cases} \tag{10}$$

where $f^\star(\cdot)$ and $h_{y_k}^\star(\cdot)$ denote the convex conjugate functions of $f(\cdot)$ and $h_{y_k}(\cdot)$, respectively, $\mathbb{V}_f$ is the domain of the conjugate function $f^\star(\cdot)$, and the notation $\mathcal{N}_I$ denotes the set of "informed" agents (i.e., $k \in \mathcal{N}_I$ only if agent $k$ in the network has access to $x_t$). Recall that the convex conjugate function of a function $g(x) : \mathbb{R}^{M \times 1} \mapsto \mathbb{R}$ (not necessarily convex) is defined by [13, pp.90-95]

$$g^\star(u) \triangleq \sup_x \Big[ u^T x - g(x) \Big], \quad u \in \mathbb{V}_g \tag{11}$$

If the function $g(x)$ happens to be strongly-convex, then the domain of its conjugate function becomes $\mathbb{V}_g = \mathbb{R}^{M \times 1}$.

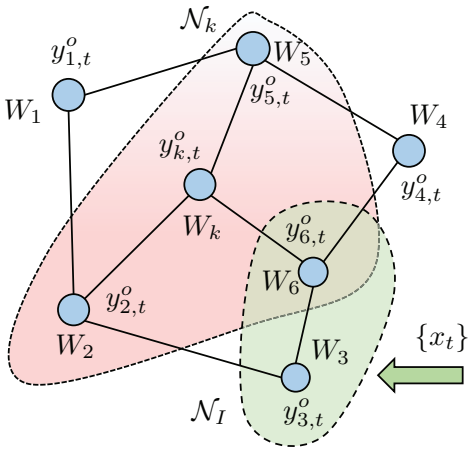


**Fig. 1**. The data sample $x_t$ at each time $t$ is available to a subset $\mathcal{N}_I$ of agents in the network (e.g., agents 3 and 6 in the figure), and each agent $k$ is in charge of one sub-dictionary, $W_k$, and the corresponding optimal sub-vector of coefficients estimated at time $t$, $y^o_{k,t}$. We use $\mathcal{N}_k$ to denote the set of neighbors of agent $k$.

Problem (9a)–(9b) is better conditioned than problem (3). For example, while the original objective function (3) is not necessarily differentiable, the new aggregate cost function (9a) is strongly-convex with Lipschitz gradient (and, hence, differentiable) when $f(\cdot)$ is strongly-convex with Lipschitz gradients (reference [8] also examines a case where $f(\cdot)$ is not strongly-convex, but the aggregate cost in (9a) still possesses these desirable properties). This means that we do not need to utilize subgradient techniques to optimize (3), which are known to be slower than gradient-based techniques for minimizing differentiable functions as in (9a)–(9b).

The optimizer of (3), $y_t^o$, can be recovered from the optimizer of (9a)–(9b), $\nu_t^o$, in a distributed manner (see (34) further ahead). We require that at least one agent has access to the data vector $x_t$ at time $t$ so that $|\mathcal{N}_I| \geq 1$. Figure 1 illustrates the data setup and $\mathcal{N}_I$. In [8, 9], the inference step (9a)–(9b) was solved by using a distributed implementation of the consensus or diffusion type. For the reasons explained before, we now examine an alternative solution method that is based on using instead an ADMM procedure. As alluded to earlier, our main motivation is to show in Sec. 4 that the resulting ADMM algorithm can be made efficient by solving its minimization step in closed-form for some special penalty functions $f(\cdot)$ and $h_y(\cdot)$ of the form shown further ahead in (36).

### 3.2. Related Works

Some related work appears in the studies [14,15]. However, there are some key differences in problem formulation, generality, and technique. For example, these references do not deal with dictionary learning problems and focus only on the solution of special cases of the inference problem (3), as illustrated below. Their problem formulations focus on determining sparse solutions to linear systems of equations, and they can be interpreted as corresponding to scenarios with dictionaries that are static and not learned from data and where it is further assumed that all agents have access to the same data $x_t$. In comparison, in this article, we show how to *jointly* address the inference *and* learning problems (1)–(3), allow the dictionary elements to be updated dynamically, and require only a subset of the agents to have access to the data.

For instance, one of the problems studied in [14] is the following inference problem (compare with (3)):

$$y_t^o \triangleq \arg\min_{y} \quad \sum_{k=1}^{N}\left[\gamma\|y_k\|_1 + \frac{\delta}{2}\|y_k\|^2\right] \tag{12a}$$

$$\text{s.t.} \quad \sum_{k=1}^{N} W_k y_k = x_t \tag{12b}$$

This formulation can be recast as a special case of (3) by selecting:

$$h_{y_k}(y_k) = \gamma\|y_k\|_1 + \frac{\delta}{2}\|y_k\|^2 \tag{13a}$$

$$f(x_t) = I_{\mathbb{B}}\left(x_t - \sum_{k=1}^{N} W_k y_k\right) \tag{13b}$$

where $I_{\mathbb{B}}(\cdot)$ is the indicator function defined by:

$$I_{\mathbb{B}}(u) = \begin{cases} 0, & u \in \mathbb{B} \\ \infty, & u \notin \mathbb{B} \end{cases} \tag{14}$$

and the set $\mathbb{B}$ is defined as

$$\mathbb{B} \triangleq \{\mathbb{0}_M\} \tag{15}$$

Constraints of the form (12b), or a residual function of the form (13b), are not generally meaningful for problems involving *both* learning and inference tasks since the measurements $x_t$ tend to be subject to noise and, moreover, the dictionary matrix $W$ will be continually adjusted over time rather than remain static. In [15], the authors relax condition (12b) and consider instead:

$$y_t^o \triangleq \arg\min_{y} \quad \sum_{k=1}^{N}\left[\gamma\|y_k\|_1 + \frac{\delta}{2}\|y_k\|^2\right] \tag{16a}$$

$$\text{s.t.} \quad \left\|\sum_{k=1}^{N} W_k y_k - x_t\right\| \leq \sigma \tag{16b}$$

for some $\sigma \geq 0$, which again can be viewed as a special case of problem (3) for the same $h_{y_k}(\cdot)$ from (13a) and with the indicator function in (13b) replaced by $I_{\mathbb{C}}(u)$ relative to the set

$$\mathbb{C} \triangleq \left\{u \in \mathbb{R}^{M\times 1} : \|u\|_2 \leq \sigma\right\} \tag{17}$$

An alternative problem formulation that removes the indicator functions is considered in [15, 16], namely,

$$y_t^o \triangleq \arg\min_{y}\left[\frac{1}{2}\|x_t - Wy\|^2 + \gamma\|y\|_1\right] \tag{18}$$

Here, we now have

$$h_y(y) = \gamma\|y\|_1, \qquad f(u) = \frac{1}{2}\|u\|^2 \tag{19}$$

However, the dictionary elements, as well as the entries of $x_t$, were partitioned by *rows* across the network as opposed to our column-wise partitioning in (6):

$$W = [U_1^T, \ldots, U_N^T]^T \tag{20}$$

In this case, it is straightforward to rewrite problem (18) in the form

$$y_t^o \triangleq \arg\min_{y} \sum_{k=1}^{N}\left[\frac{1}{2}\|x_{k,t} - U_k y\|^2 + \frac{\gamma}{N}\|y\|_1\right] \tag{21}$$

which is naturally in the sum-of-costs form (9a)–(9b) in the original domain and does not require transformation under duality, as was the case to arrive at (9a)–(9b) from (3). The more challenging problem where the matrix $W$ is partitioned column-wise, as we are considering in (6), was not examined in [15] where it was stated that this case cannot be trivially recast into their sum-of-costs formulation. Using the same dual argument that led to (9a)–(9b), we are able to transform this column-wise partitioned problem into a sum-of-costs problem, as already shown in [8, 9], even for more general functions $f(\cdot)$ and $h_y(\cdot)$. For instance, expression (10) shows how the individual costs should be constructed in this case in terms of the conjugate functions of the penalties.

### 3.3. ADMM-based inference over a network

We return to the general formulation (9a)–(9b) and proceed to solve it via an ADMM scheme. We start by rewriting

$$\min_{\nu} \quad \sum_{k=1}^{N}\left[J_k(\nu_k; x_t) + I_{\mathbb{V}_f}(\nu_k)\right] \tag{22a}$$

$$\text{s.t.} \quad \nu_1 = \nu_2 = \cdots = \nu_N \tag{22b}$$

where we introduced the variables $\{\nu_k\}$ and constrained them to be the same through (22b), and where $I_{\mathbb{V}_f}(\nu)$ is the indicator function (14). We assume that the graph of the network is connected, i.e., there exists a path from any agent to any other agent possibly through other agents in the network. We recall the following definitions for the incidence and Laplacian matrices of a graph [17, 18].

**Definition 1** (Incidence matrix of an undirected graph)**.** *Given a graph $G$, the incidence matrix $C$ is an $E \times N$ matrix, where $E$ is the total number of edges in the graph and $N$ is the total number of nodes, with entries defined as follows:*

$$[C]_{ek} = \begin{cases} +1, & k \text{ is the lower indexed node connected to } e \\ -1, & k \text{ is the higher indexed node connected to } e \\ 0, & k \text{ is not connected to edge } e \end{cases} \tag{23}$$

*Thus, $C\mathbb{1}_N = \mathbb{0}_E$. Self-loops are excluded.* □

**Definition 2** (Laplacian matrix of a graph)**.** *Given a graph $G$, the Laplacian matrix $L$ is an $N \times N$ matrix whose entries are defined as follows:*

$$[L]_{k\ell} = \begin{cases} |\mathcal{N}_k| - 1, & k = \ell \\ -1, & k \neq \ell, \ell \in \mathcal{N}_k \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

*where $|\mathcal{N}_k|$ denotes the number of neighbors of agent $k$ (including the agent itself). It holds that $L = C^T C$, where $C$ is the incidence matrix of the graph.* □

Let $C$ and $L$ respectively denote the incidence and Laplacian matrices of the graph. Each agent $k$ has access to the $k$-th row and column of the Laplacian matrix since they are aware of their local network connections. In addition, each node $k$ has access to each row $e$ of the incidence matrix where $c_{ek} \neq 0$. It is then possible to rewrite (22a)-(22b) as

$$\min_{w} \quad \sum_{k=1}^{N}\left[J_k(\nu_k; x_t) + I_{\mathbb{V}_f}(\nu_k)\right] \tag{25a}$$

$$\text{s.t.} \quad \mathcal{C}\nu = \mathbb{0}_{MN} \tag{25b}$$

where we introduced the extended quantities:

$$\mathcal{C} \triangleq C \otimes I_M, \qquad \nu \triangleq \text{col}\{\nu_1, \ldots, \nu_N\} \tag{26}$$

and where $\otimes$ denotes the Kronecker product operation [19]. The augmented Lagrangian of the constrained problem (25a)–(25b) is:

$$L_2(\{\nu_k\}, \lambda) = \sum_{k=1}^{N} \big[J_k(\nu_k; x_t) + I_{\mathbb{V}_f}(\nu_k)\big] + \lambda^T \mathcal{C}\nu + \frac{\eta}{2}\|\nu\|_{\mathcal{L}}^2 \tag{27}$$

where $\lambda \in \mathbb{R}^{EM}$ is the Lagrange multiplier associated with the consensus constraint $\mathcal{C}\nu = \mathbb{0}_{MN}$, $\mathcal{L} = \mathcal{C}^T\mathcal{C}$, and $\eta$ is a non-negative regularization parameter. The augmentation of the Lagrangian introduces a penalty term that encourages nodes to reach agreement. The variable $\nu$ that minimizes (27) is in the consensus space due to (25b). The dual function, $g_2(\lambda)$, is found by minimizing the Lagrangian $L_2(\nu, \lambda)$ over the primal variable $\nu$, say,

$$g_2(\lambda) = \min_{\{\nu_\ell\}} L_2(\{\nu_\ell\}, \lambda) \tag{28}$$

The goal of the optimization procedure is to find the optimal Lagrange multiplier, $\lambda_t^o$, by maximizing the dual function:

$$\lambda_t^o \triangleq \arg\max_{\lambda} g_2(\lambda) = L_2(\{\nu_{\ell,t}^o\}, \lambda) \tag{29}$$

This task can be achieved by means of a gradient ascent recursion. For example, once $k$ obtains $\nu_{k,t}^o$, the algorithm can adjust the Lagrange multiplier $\lambda$ according to the recursion (where we are using the subscript $i$ to denote the iteration index):

$$\begin{aligned}
\lambda_{e_{\ell k},i} &= \lambda_{e_{\ell k},i-1} + \eta \nabla_{\lambda_{e_{\ell k}}} g_2(\lambda_{i-1}) \\
&= \lambda_{e_{\ell k},i-1} + \eta \nabla_{\lambda_{e_{\ell k}}} L_2(\{\nu_{\ell,t}^o\}, \lambda_{i-1}) \\
&= \lambda_{e_{\ell k},i-1} + \eta \sum_{k=1}^{N} \sum_{\ell=k+1}^{N} g_{\ell k}(\nu_{k,t}^o - \nu_{\ell,t}^o) \\
&= \lambda_{e_{\ell k},i-1} + \eta(\nu_{k,t}^o - \nu_{\ell,t}^o), \quad [\ell \in \mathcal{N}_k, \ell > k]
\end{aligned} \tag{30}$$

where $g_{\ell k}$ denotes the $(\ell, k)$-th element of the adjacency matrix $G$ of the topology graph, which is defined as $g_{\ell k} = 1$ when $\ell \in \mathcal{N}_k$, and is zero otherwise. Moreover, the vector $\lambda_{e_{\ell k},i-1} \in \mathbb{R}^M$ denotes the block in $\lambda_{i-1}$ that is the dual variable associated with edge $e$ that connects nodes $\ell$ and $k$ (these nodes are the only agents with access to this block). Observe that in (30), the dual step-size is chosen to be the augmentation parameter $\eta$. Either of the two neighboring nodes $k$ or $\ell$ can update $\lambda_{e_{\ell k}}$ at each time-step (for example, the lower-indexed node).

We still need to address the minimization step (28) since its solution is needed in (30). This task can be pursued via the Jacobian ADMM approach [11, p.356] [12] [20, p.219]. First, each node $k$ exchanges its variables $\{\nu_{k,i-1}, \lambda_{e_{\ell k},i-1}\}$ with its neighbors, and then each node holds its neighbor variables fixed, and optimizes over its own variable:

$$\nu_{k,i} = \arg\min_{\nu_k} L_2(\{\nu_k, \nu_{\ell,i-1}\}, \lambda_{i-1}), \quad [\ell \in \mathcal{N}_k \backslash \{k\}] \tag{31}$$

The agents in the network will repeat steps (30)–(31) until they converge, i.e., the agents execute:

$$\begin{cases}
\nu_{k,i} = \arg\min_{\nu_k} L_2(\{\nu_k, \nu_{\ell,i-1}\}, \lambda_{i-1}), & [\ell \in \mathcal{N}_k \backslash \{k\}] \quad (32) \\
\lambda_{e_{\ell k},i} = \lambda_{e_{\ell k},i-1} + \eta(\nu_{k,i} - \nu_{\ell,i}), & [\ell \in \mathcal{N}_k, \ell > k] \quad (33)
\end{cases}$$

The main difficulty associated with (32)–(33) is solving the first step (32). In general, the optimization in (32) will be constrained when $\mathbb{V}_f \neq \mathbb{R}^{M\times 1}$ and agents may need to use an iterative scheme to solve (32). We show later in App. A that step (32) can actually be solved in closed-form for the important case $f(\cdot) = \frac{1}{2}\|\cdot\|^2$ and $h_{y_k}(\cdot) = \gamma\|\cdot\|_1 + \frac{\delta}{2}\|\cdot\|^2$.

Now, once the algorithm converges, each node will have access to $\nu_t^o$ since all nodes are expected to converge to the same optimizer in the consensus space due to (25b). Once this is achieved, the nodes can recover $y_{k,t}^o$ uniquely (since $h_{y_k}(y_k)$ is assumed to be strongly-convex) from $\nu_t^o$ as follows [8, 9]:

$$y_{k,t}^o = \arg\max_{y_k} \big[(W_k^T \nu_t^o)^T y_k - h_{y_k}(y_k)\big] \tag{34}$$

where $y_{k,t}^o$ is the $k-$th entry of the solution $y_t^o$ to the inference step (3). Observe that each node can recover $y_{k,t}^o$ using its own information $W_k$ and $\nu_t^o$ without any further communication. As observed in [8, 9], and as we will see in Sec. 4, the optimization in (34) can also be computed in closed-form in some important cases.

### 3.4. Distributed dictionary update

Once $y_{k,t}^o$ is found at all nodes, it is still necessary to update the dictionary elements. It was shown in [8, 9] that this task can be accomplished efficiently by using the following stochastic incremental proximal gradient algorithm [21]:

$$W_{k,t} = \Pi_{\mathbb{W}_k}\Big\{\text{prox}_{\mu_w \cdot h_{W_k}}\big(W_{k,t-1} + \mu_w \nu_t^o (y_{k,t}^o)^T\big)\Big\} \tag{35}$$

where $W_{t-1}$ is sub-divided according to (6), $\Pi_{\mathbb{W}_k}(X)$ is the projection of the matrix $X$ onto the set $\mathbb{W}_k$. This leads to the dictionary learning algorithm listed in Algorithm 1. Observe that the ADMM implementation in Alg. 1 requires three time-scales: (1) iteration over $t$, (2) iteration over $i$ between two successive time instants $t$, and (3) iterations between two successive indices $i$ to solve (32) by means of some iterative procedure. In contrast, the diffusion-based dictionary learning algorithm proposed in [8, 9] requires only two time-scales corresponding to the $t$ and $i$ domains. We will see in the next section that (32) can be solved in closed-form for the special penalty functions (36), thus eliminating the need for the last time-scale in the ADMM implementation in this particular case.

**Algorithm 1** Distributed Dictionary Learning via Dual-Jacobian-ADMM: General Algorithm

**for** each input data sample $x_t$, each node $k$ **do**
  **for** Repeat until convergence: $i = 0, 1, \ldots$ **do**
    Solve for $\{\nu_{k,i}\}$:

$$\nu_{k,i} = \arg\min_{\nu_k} L_2(\{\nu_k, \nu_{\ell,i-1}\}, \lambda_{i-1}), \quad [\ell \in \mathcal{N}_k \backslash \{k\}]$$

    Update Lagrange Multipliers:

$$\lambda_{e_{\ell k},i} = \lambda_{e_{\ell k},i-1} + \eta(\nu_{k,i} - \nu_{\ell,i}), \quad [\ell \in \mathcal{N}_k, \ell > k]$$

  **end for**
  Set $\nu_t^o = \nu_{k,i}$.
  Compute optimal local primal variable:

$$y_{k,t}^o = \arg\max_{y_k} \big[(W_k^T \nu_t^o)^T y_k - h_{y_k}(y_k)\big]$$

  Update the dictionary using:

$$W_{k,t} = \Pi_{\mathbb{W}_k}\Big\{\text{prox}_{\mu_w \cdot h_{W_k}}\big(W_{k,t-1} + \mu_w \nu_t^o (y_{k,t}^o)^T\big)\Big\}$$

**end for**

## 4. EXPERIMENTAL RESULTS

In this section, we consider an image denoising application to illustrate the dictionary learning task. Let

$$f(u)=\frac{1}{2}\|u\|^2,\quad h_y(y)=\gamma\|y\|_1+\frac{\delta}{2}\|y\|^2,\quad h_W(W)=0 \tag{36}$$

Let $N_k=1$ for each node and let all agents in the network have access to $x_t$ so that $|\mathcal{N}_I|=N$ and each agent in the network is responsible for maintaining a single column of the dictionary $W$. While Alg. 1 does not require these simplifications, we will see that with this setup, step (32) can be solved in closed-form at every agent. Specifically, in this case, equation (10) becomes:

$$J_k(\nu;x_t)\ \triangleq -\frac{\nu^T x_t}{N}+\frac{1}{2N}\|\nu\|^2+\mathcal{S}_{\frac{\gamma}{\delta}}\left(\frac{w_k^T\nu}{\delta}\right) \tag{37}$$

where

$$\mathcal{S}_{\frac{\gamma}{\delta}}(x)\triangleq-\frac{\delta}{2}\cdot\left\|\mathcal{T}_{\frac{\gamma}{\delta}}(x)\right\|_2^2-\gamma\cdot\left\|\mathcal{T}_{\frac{\gamma}{\delta}}(x)\right\|_1+\delta\cdot x^T\mathcal{T}_{\frac{\gamma}{\delta}}(x) \tag{38}$$

and $\mathcal{T}_\lambda(x)$ denotes the entry-wise soft-thresholding operator on the vector $x$:

$$[\mathcal{T}_\lambda(x)]_n\triangleq(|[x]_n|-\lambda)_+\mathrm{sgn}([x]_n) \tag{39}$$

where $(x)_+\triangleq\max(x,0)$. Substituting (37) into Alg. 1, we obtain the listing of Algorithm 2 shown in the table (see App. A), where $\mathcal{N}_k$ denotes the neighborhood set of node $k$, including node $k$ itself.

**Algorithm 2** Distributed Dictionary Learning via Dual-ADMM

**for** each input data sample $x_t$, each node $k$ **do**

**for** Repeat until convergence: $i=0,1,\ldots$ **do**

Solve (32) as follows:

$$c_{k,i}=\sum_{\ell=1}^{k-1}g_{k\ell}(\lambda_{e_{k\ell},i-1}+\eta\nu_{\ell,i-1})-\sum_{\ell=k+1}^{N}g_{\ell k}(\lambda_{e_{\ell k},i-1}-\eta\nu_{\ell,i-1}) \tag{40}$$

$$\nu_{k,i}^1=\left(\left(\frac{1}{N}+\eta(|\mathcal{N}_k|-1)\right)I_M+\frac{w_{k,t-1}w_{k,t-1}^T}{\delta}\right)^{-1}\times\left(\frac{x_t}{N}+\frac{\gamma}{\delta}w_{k,t-1}+c_{k,i}\right) \tag{41}$$

$$\nu_{k,i}^2=\left(\frac{1}{N}+\eta(|\mathcal{N}_k|-1)\right)^{-1}\left(\frac{x_t}{N}+c_{k,i}\right) \tag{42}$$

$$\nu_{k,i}^3=\left(\left(\frac{1}{N}+\eta(|\mathcal{N}_k|-1)\right)I_M+\frac{w_{k,t-1}w_{k,t-1}^T}{\delta}\right)^{-1}\times\left(\frac{x_t}{N}-\frac{\gamma}{\delta}w_{k,t-1}+c_{k,i}\right) \tag{43}$$

$$\nu_{k,i}=\begin{cases}\nu_{k,i}^1, & w_k^T\nu_{k,i}^1>\gamma\\ \nu_{k,i}^2, & |w_k^T\nu_{k,i}^2|\le\gamma\\ \nu_{k,i}^3, & w_k^T\nu_{k,i}^3<-\gamma\end{cases} \tag{44}$$

Update Lagrange Multipliers:

$$\lambda_{e_{k\ell},i}=\lambda_{e_{k\ell},i-1}+\eta(\nu_{\ell,i}-\nu_{k,i}),\quad[\ell\in\mathcal{N}_k,\ell>k] \tag{45}$$

**end for**

Set $\nu_t^o=\nu_{k,i}$.

Compute the primal local primal variable $y_{k,t}^o=\frac{1}{\delta}\mathcal{T}_\gamma(w_{k,t-1}^T\nu_t^o)$.

Update the dictionary using:

$$w_{k,t}=\Pi_{\mathbb{W}_k}\left\{w_{k,t-1}+\mu_w\nu_t^o(y_{k,t}^o)^T\right\} \tag{46}$$

**end for**

The computation of $\nu_{k,i}^1$, $\nu_{k,i}^2$, and $\nu_{k,i}^3$ in (41)–(43) can be done efficiently by using the matrix-inversion lemma [19, p.48]:

$$\left(\alpha I_M+\frac{1}{\delta}w_kw_k^T\right)^{-1}b=\frac{b}{\alpha}-\frac{(w_k^Tb)\cdot w_k}{\alpha^2\cdot(\delta+\frac{1}{\alpha}\|w_k\|^2)} \tag{47}$$

We let $N=196$ so that the dictionary $W\in\mathbb{R}^{M\times N}$ is distributed over 196 agents. The network is generated randomly where the probability that any two nodes are connected is 0.5, and the network is ensured to be connected. We extract a total of one million $10\times10$ patches from images 101-200 of the the non-calibrated natural image dataset [22]. The border two pixels were discarded around each image and the top-left $1019\times1019$ pixels were then used for patch extraction. With each data sample being a $10\times10$ patch from a certain image, the dimension of the input data sample is $M=100$ (vertically stacked columns). We initialize each entry of the dictionary matrix $W$ with a zero-mean unit-variance Gaussian random variable. The columns are then scaled to guarantee that the sub-unit-norm constraint (2) is satisfied. The patch extraction, preprocessing, and image reconstruction code utilized (excluding dictionary learning and patch inference steps) is borrowed from [23].

Given the one million data samples $\{x_t\}$, each of which satisfies $x_t\in\mathbb{R}^M$, the network wishes to solve (1)–(3), where $\mathbb{W}=\mathbb{W}_1\cap\ldots\cap\mathbb{W}_N$ and

$$\mathbb{W}_k=\{w_k:\ \|w_k\|_2\le1\} \tag{48}$$

for each atom $q$ in dictionary $w_k$. We use Alg. 2 and let the lower-indexed node update (45). We utilize $\gamma=45$, $\delta=0.1$, and $\eta=0.002$. A step-size of $\mu_w=5\times10^{-5}$ was utilized for the adaptation of the dictionary elements $\{w_k\}$. We chose to run Alg. 2 for 50 iterations (as opposed to the 300 iterations required for the gradient-descent type algorithm in [8, 9]). The data were presented in minibatches [24] of size four samples/minibatch and the update gradients $\nu_t^o y_{k,t}^o$ were averaged over the four samples per step.

In the far right of Fig. 2, we show the dictionary learned over the 196 agents in the network (f) as well as the one learned by using the centralized method in [2] as a benchmark (c). The original image without corruption is listed in pane (a), while the corrupted image — the input to the algorithms—is listed in (d). The parameters $\eta$ and iterations used for the denoising are the same as the training. The corrupted image's PSNR[1] is 15.8dB, while the PSNR for the recovered image using the centralized solution of [2] was found to be 24.5dB. The average denoising PSNR performance across the distributed network is found to be 25.4dB with a standard deviation of 0.0024dB. Thus, the performance is uniform across the network.

## A. CLOSED-FORM SOLUTION OF (32)

First, observe that the indicator function $I_{\mathbb{V}_f}(\nu_k)=0$ in (27) since $f(u)=\frac{1}{2}\|u\|^2$ is strongly-convex and thus $\mathbb{V}_f=\mathbb{R}^M$. Therefore, we have that $L_2(\{\nu_k\},\lambda_{i-1})$ becomes:

$$L_2(\{\nu_k\},\lambda_{i-1})=\sum_{n=1}^{N}J_n(\nu_n;x_t)+\sum_{n=1}^{N}\sum_{\ell=n+1}^{N}g_{\ell n}\lambda_{e_{\ell n}}^T(\nu_n-\nu_\ell)+\frac{\eta}{2}\sum_{n=1}^{N}\sum_{\ell=n+1}^{N}g_{\ell n}\|\nu_n-\nu_\ell\|^2 \tag{49}$$

[1] PSNR is the peak-signal-to-noise ratio defined as $\mathrm{PSNR}\triangleq10\log_{10}(I_{\max}^2/\mathrm{MSE})$, where $I_{\max}$ is the maximum pixel intensity in the image and MSE is the mean-square-error over all image pixels.

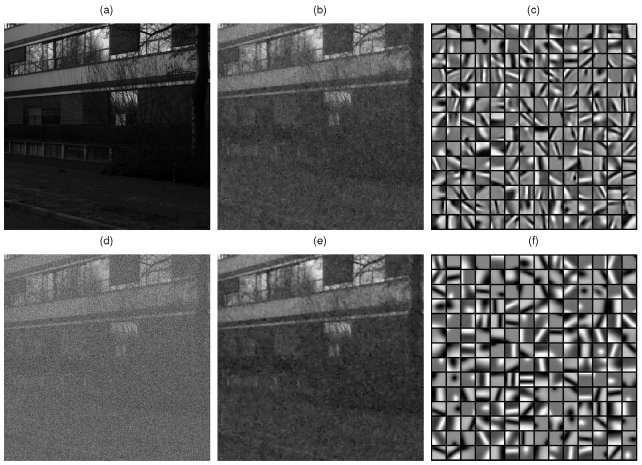


**Fig. 2**. Application of dictionary learning to image denoising: (a) original image; (b) denoised image by using the centralized method from [2]; (c) dictionary obtained by the centralized method from [2]; (d) image corrupted by additive white Gaussian noise; (e) denoised image by our proposed distributed method at agent 1; (f) dictionary obtained by our proposed distributed implementation.

Taking the gradient of (49) with respect to $\nu_k$ and setting it to zero, we obtain:

$$\begin{aligned}\nabla_{\nu_k} L_2(\{\nu_k\}, \lambda_{i-1}) = \nabla_{\nu_k} J_k(\nu_k; x_t) + \sum_{\ell=k+1}^{N} g_{\ell k}\lambda_{e_{\ell k}} - \sum_{\ell=1}^{k-1} g_{\ell k}\lambda_{e_{\ell k}} + \\ \eta \sum_{\ell=k+1}^{N} g_{\ell k}(\nu_k - \nu_\ell) + \eta \sum_{\ell=1}^{k-1} g_{\ell k}(\nu_k - \nu_\ell) \\ = 0 \end{aligned} \tag{50}$$

Now, observe that $J_k(\nu_k; x_t)$ is piece-wise quadratic due to (38) and the fact that $\|\cdot\|^2$ is quadratic. We can obtain its gradient as

$$\nabla_{\nu_k} J_k(\nu_k; x_t) = \begin{cases} \frac{\nu_k - x_t}{N} + \frac{1}{\delta} w_k w_k^T \nu_k - \frac{\gamma}{\delta} w_k, & w_k^T \nu_k > \gamma \\ \frac{\nu_k - x_t}{N}, & |w_k^T \nu_k| \leq \gamma \\ \frac{\nu_k - x_t}{N} + \frac{1}{\delta} w_k w_k^T \nu_k + \frac{\gamma}{\delta} w_k, & w_k^T \nu_k < -\gamma \end{cases} \tag{51}$$

Substituting (51) into (50) and solving for $\nu_k$, the solution reduces to (40)–(44). Observe that this minimization only depends on variables from agent $k$'s immediate neighbors, and is done simultaneously across the nodes.

## B. REFERENCES


[1] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[2] 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, Mar. 2010.

[3] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, Jan. 2006.

[4] H. Shen and J. Z. Huang, "Sparse principal component analysis via regularized low rank matrix approximation," *Journal of Multivariate Analysis*, vol. 99, no. 6, pp. 1015–1034, Jul. 2008.

[5] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, "Biclustering via sparse singular value decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, Dec. 2010.

[6] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," in *Proc. Neural Information Processing Systems* (NIPS), Lake Tahoe, Nevada, Dec. 2008, pp. 1033–1040.

[7] S. P. Kasiviswanathan, H. Wangy, A. Banerjeey, and P. Melville, "Online $\ell_1$-dictionary learning with application to novel document detection," in *Proc. Neural Information Processing Systems* (NIPS), Lake Tahoe, Nevada, Dec. 2012, pp. 2267–2275.

[8] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," available as *arXiv:1402.1515*, Feb. 2014.

[9] J. Chen, Z. J. Towfic, and A. H. Sayed, "Online dictionary learning over distributed models," in *Proc. IEEE ICASSP*, Florence, Italy, May 2014, pp. 3874–3878.

[10] S. P. 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, Jul. 2011.

[11] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Distributed detection and estimation in wireless sensor networks," *in* Academic Press Library in Signal Processing, vol. 2, R. Chellapa and S. Theodoridis, *editors,* pp. 329–408, Elsevier, 2014.

[12] B.-S. He, "Parallel splitting augmented Lagrangian methods for monotone structured variational inequalities," *Computational Optimization and Applications*, vol. 42, no. 2, pp. 195–212, Mar. 2009.

[13] S. P. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[14] J. F. C. Mota, J. Xavier, P. M. Q. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 1942–1956, Apr. 2012.

[15] J.F.C. Mota, J.M.F. Xavier, P.M.Q. Aguiar, and M. Puschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[16] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[17] R. B. Bapat, *Graphs and Matrices*, Springer, NY, 2010.

[18] A. H. Sayed, "Diffusion adaptation over networks," *in Academic Press Library in Signal Processing,* vol. 3, R. Chellapa and S. Theodoridis, *editors,* pp. 323–454, Elsevier, 2014.

[19] A. J. Laub, *Matrix Analysis for Scientists and Engineers*, SIAM, PA, 2005.

[20] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, 1997.

[21] D. P. Bertsekas, "Incremental gradient, subgradient, and proximal methods for convex optimization: A survey," *LIDS Technical Report,* MIT, , no. 2848, 2010.

[22] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proceedings: Biological Sciences*, vol. 265, no. 1394, pp. 359–366, Mar. 1998.

[23] G. Peyré, "The numerical tours of signal processing - Advanced computational signal and image processing," *IEEE Computing in Science and Engineering*, vol. 13, no. 4, pp. 94–97, Jul. 2011.

[24] 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, Jan. 2012.