Learning Overcomplete Representations from Distributed Data: A Brief Review

Haroon Raja and Waheed U. Bajwa

Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854

ABSTRACT

Most of the research on dictionary learning has focused on developing algorithms under the assumption that data is available at a centralized location. But often the data is not available at a centralized location due to practical constraints like data aggregation costs, privacy concerns, etc. Using centralized dictionary learning algorithms may not be the optimal choice in such settings. This motivates the design of dictionary learning algorithms that consider distributed nature of data as one of the problem variables. Just like centralized settings, distributed dictionary learning problem can be posed in more than one way depending on the problem setup. Most notable distinguishing features are the online versus batch nature of data and the representative versus discriminative nature of the dictionaries. In this paper, several distributed dictionary learning algorithms that are designed to tackle different problem setups are reviewed. One of these algorithms is cloud K-SVD, which solves the dictionary learning problem for batch data in distributed settings. One distinguishing feature of cloud K-SVD is that it has been shown to converge to its centralized counterpart, namely, the K-SVD solution. On the other hand, no such guarantees are provided for other distributed dictionary learning algorithms. Convergence of cloud K-SVD to the centralized K-SVD solution means problems that are solvable by K-SVD in centralized settings can now be solved in distributed settings with similar performance. Finally, cloud K-SVD is used as an example to show the advantages that are attainable by deploying distributed dictionary algorithms for real world distributed datasets.

1. INTRODUCTION

Two main themes we can observe in data generated these days are: (i) there is *massive* amount of data and (ii) data are *distributed* across networked nodes/sites/agents*. Examples of such sources include datacenters, large-scale sensor networks, surveillance cameras, medical data in hospitals, etc. For solving information processing tasks like estimation, prediction, detection, etc., we require a model that can represent the observed data with high accuracy. Our focus in this paper is on learning a good model from observed, distributed data. Traditionally, subspace-based methods like principal component analysis have been used successfully for learning the model from observed data. Union-of-subspaces (UoS) model is a generalization of subspace model and has been shown to perform better for many information processing tasks (e.g., information processing tasks[1,2]). Data-adaptive methods for learning a UoS model from data of interest have been studied under the rubric of dictionary learning in literature.[3–5]

The goal in dictionary learning is to learn a dictionary $D \in \mathbb{R}^{n\times K}$ from a set of training data $\{y_i\}_{i=1}^{S}$ such that each data sample can be represented by a few columns/atoms of D. More precisely, the goal is to find a dictionary such that we can represent any sample using no more than T_0 columns of D, where $T_0 \ll n < K$. Dictionary learning has found applications in image processing,[1,2] hyperspectral imaging,[6] etc., and efforts have been made towards solving dictionary learning problems in an efficient manner; some examples of this include [3], [4], [7], [2], and [8].

Most of the work on dictionary learning assumes availability of training data at a centralized location.[3,4] Recently, contributions have been made towards solving the dictionary learning problem when data is distributed across a network of sites.[9–16] Distributed dictionary learning algorithms proposed in [9–13] deal with the case

Emails: haroon.raja@rutgers.edu, waheed.bajwa@rutgers.edu

*In the distributed processing literature, distributed entities with data are known with different names like node, site, agent, etc. In the following, we will use 'site' to refer to the entity with data.

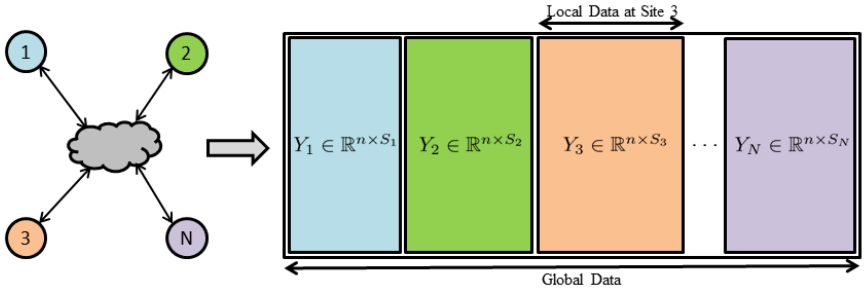

Figure 1. Batch data distributed across different sites in the network.

of batch data. Similar to [5] in centralized dictionary learning, [9] and [13] solve distributed dictionary learning problem by solving the underlying optimization program using gradient descent. On the other hand, [9] uses the diffusion-based strategy[17] to perform distributed optimization, while [13] proposes EXTRA-AO algorithm that extends the distributed optimization method called EXTRA,[18] which is a consensus-based[19] algorithm. Furthermore, [13] provides analysis that shows that EXTRA-AO algorithm converges to a stationary point of the optimization program. It is also shown empirically in [13] that under certain conditions, the algorithm proposed in [9] will not converge. Next, [12] and [11] also solve the distributed dictionary learning problem for batch data settings by providing distributed implementations of the centralized K-SVD algorithm.[3] While both the methods perform sparse coding steps locally at each site, they differ in the way they perform the dictionary update step in a distributed way. Specifically, [11] uses distributed version of power method[20] to perform the dictionary update step, while [12] uses ADMM-based distributed optimization approach[21] to achieve the goal. Due to this difference in the two approaches, [11] is able to provide convergence analysis , while no such guarantees are provided in [12].

Algorithms proposed in [14–16] solve the distributed dictionary learning problem for online settings. The authors in [15] pose dictionary learning problem as a recursive least squares problem and use D-RLS[22] algorithm to solve the resulting optimization problem. On the other hand, [16] learns a discriminative dictionary for online settings in a distributed manner using a variant of the Arrow-Hurwicz saddle point algorithm.[23] Finally, [14] learns only a segment of dictionary at each site in online settings.

To summarize, this paper reviews the algorithms developed to solve the dictionary learning problem when data is available only at different distributed sites. In the following we will first formally define the distributed dictionary learning problem for different settings in Section 2, distributed dictionary learning algorithms are reviewed in Section 3, and a detailed discussion of the *cloud K-SVD*[11] algorithm is provided in Section 4.

Notation. We use lower-case letters to represent scalars and vectors, while we use upper-case letters to represent matrices. Given a vector v, $\text{supp}(v)$ returns indices of the nonzero entries in v, $\|v\|_p$ denotes its ℓ_p norm, $\|v\|_0$ counts the number of its nonzero entries, and superscript $(\cdot)^\mathsf{T}$ denotes the transpose operation. We use $\mathbb{1}$ to denote a vector of all 1's. We define W to be a doubly stochastic matrix, i.e., $\mathbb{1}^\mathsf{T} W = \mathbb{1}^\mathsf{T}$ and $W\mathbb{1} = \mathbb{1}$. Given a set $\mathcal{I}$, $v_{|\mathcal{I}}$ and $A_{|\mathcal{I}}$ denote a subvector and a submatrix obtained by retaining entries of vector v and columns of matrix A corresponding to the indices in $\mathcal{I}$, respectively. Given matrices $\{A_i \in \mathbb{R}^{n_i \times m_i}\}_{i=1}^N$, the operation $\text{diag}\{A_1, \ldots, A_N\}$ returns a block-diagonal matrix $A \in \mathbb{R}^{\sum n_i \times \sum m_i}$ that has A_i's on its diagonal. Finally, given a matrix A, a_j and $a_{j,T}$ denote the j^{th} column and the j^{th} row of A, respectively.

2. PROBLEM FORMULATION

Consider a dataset $Y \in \mathbb{R}^{n \times S}$, where S is the total number of samples in the dataset and n is the dimensionality of each sample. Now consider that dataset Y is distributed across a number of geographically distributed and interconnected nodes/sites and any site i has only access to the local samples $Y_i \in \mathbb{R}^{n \times S_i}$ as shown in Figure 1. Here S_i is the number of samples at site i. Mathematically, we can represent the communication topology of these interconnected sites by an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Here, $\mathcal{N} = \{1, \ldots, N\}$ is the set of sites in the network and $\mathcal{E}$ is the set of edges in $\mathcal{G}$, where $(i, i) \in \mathcal{E}$ and $(i, j) \in \mathcal{E}$ whenever site i can communicate directly with

site j. Next we highlight different settings considered for data acquisition at any site i in distributed dictionary literature. Specifically, the following three cases have been considered for distributed dictionary learning.

1. In [9–13], batch data settings are assumed, i.e., complete batch data Y_i is available at any site i before start of the dictionary learning algorithm.
2. In [15] and [16], streaming data setup is considered and it is assumed that in each iteration t of dictionary learning, we observe a new data sample $y_i^{(t)}$ at each site i.
3. In [24] and [14], an online setup is once again assumed with the difference from [15] and [16] being that in each dictionary learning iteration t, a new sample $y_I^{(t)}$ is observed at some subset of sites $\mathcal{N}_I \subseteq \mathcal{N}$.

In all three of these cases, the goal of distributed dictionary learning is to learn a dictionary $D \in \mathbb{R}^{n\times K}$ at each site such that each sample in Y can be represented using a few columns (called atoms in dictionary learning literature) of D. We can pose this distributed dictionary learning problem as an optimization program. Depending on the problem setup and the sparse coding method used in the dictionary learning formulation, the form of optimization program varies in different cases. In the following, we provide a general description of these optimization programs.

2.1 Constrained optimization for batch data

In the case of a centralized dataset available in a batch setting, dictionary learning can be posed as the following optimization problem with a constraint on ℓ_0-norms of representation vectors:

$$\big(D, X\big) = \arg\min_{D,X} \|Y - DX\|_F^2 \text{ s.t. } \forall s, \|x_s\|_0 \leq T_0. \tag{1}$$

This problem can be efficiently solved using the K-SVD algorithm.[3] In the case of distributed data, we can write (1) as follows:

$$\big(D, \{X\}_{i=1}^N\big) = \arg\min_{D,\{X_i\}_{i=1}^N} \sum_{i=1}^N \|Y_i - DX_i\|_F^2 \text{ s.t. } \forall i, s, \|x_{i,s}\|_0 \leq T_0. \tag{2}$$

Methods proposed in [11] and [12] solve this optimization problem by developing distributed variants of the K-SVD algorithm.

2.2 Unconstrained optimization for batch data

The dictionary learning problem in the case of a centralized, batch dataset can also be solved as an unconstrained optimization problem as [5]:

$$\big(D, X\big) = \arg\min_{D,X} \frac{1}{2}\big(\|Y - DX\|_F^2 + \lambda\|X\|_1\big). \tag{3}$$

Distributed variants of the dictionary learning problem posed in (3) have been solved in [9] and [13]. Specifically, [13] poses this problem as the following unconstrained optimization problem:

$$\big(D, \{X\}_{i=1}^N\big) = \arg\min_{D,\{X_i\}_{i=1}^N} \frac{1}{2}\sum_{i=1}^N \big(\|Y_i - DX_i\|_F^2 + \lambda\|X_i\|_1\big) + \sum_{k=1}^K \gamma_k \|d_k\|_2^2. \tag{4}$$

Here, d_k is the k^{th} atom of dictionary D, the regularization term in (4) promotes sparsity in coefficients X, while the second regularization term ensures the columns, d_k, of dictionary are bounded. Further, the optimization problem solved in [9] has the same form as (4), with the exception of the last regularization term (regularization of d_k).

2.3 Online optimization for dictionary learning

In the case of streaming data, [4] provides a solution to the dictionary learning problem. Defining the convex set

$$\mathcal{D} := \{D \in \mathbb{R}^{n\times K} \text{s.t. } \forall j = i, \ldots, K,\ d_j^{\mathsf{T}} d_j \leq 1\},$$

[4] solves the following optimization problem in iteration t:

$$(D^{(t)}, X) = \arg \min_{D\in\mathcal{D}, \{x^{(m)}\}_{m=1}^{t}} \frac{1}{t} \sum_{m=1}^{t} \left(\frac{1}{2} \|y^{(m)} - Dx^{(m)}\|_2^2 + \lambda \|x^{(m)}\|_1 \right). \tag{5}$$

In the case of distributed, streaming data, [14–16] solve a similar online problem for distributed online data as follows:

$$(D^{(t)}, X) = \arg \min_{D\in\mathcal{D}, \{x^{(m)}\}_{m=1}^{t}} \frac{1}{t} \sum_{i=1}^{N} \sum_{m=1}^{t} \left(\frac{1}{2} \|y_i^{(m)} - Dx_i^{(m)}\|_2^2 + \lambda \|x_i^{(m)}\|_1 \right). \tag{6}$$

More details on how [14–16] solve (6) are provided in Section 3. It is important to point out here that although [14–16] solve the distributed dictionary learning problem for streaming data, the underlying dictionary learning problems they solve are different. In [15], the problem that is solved is more closely related to the one solved in [4] for the centralized setup, i.e., learning a dictionary for data representation. On the other hand, the goal in [16] is to learn a discriminative dictionary for classification purposes. It is therefore more closely related to the dictionary learning problem solved in [7] for the centralized setup involving streaming data. Finally, [14] differs from all the other distributed dictionary learning approaches since its goal is to learn only a portion of dictionary at any given site i, in contrast to learning a *common* dictionary at all sites.

3. ALGORITHMS FOR DISTRIBUTED DICTIONARY LEARNING

Recall that formulation of the distributed dictionary learning problem depends on the choice of sparse coding algorithm, underlying information processing task (i.e., representation or classification), and dataset properties (i.e., batch or online settings). Table 1 lists the works that have studied the distributed dictionary learning problem under these different settings. Notice also from different formulations of the dictionary learning problem as an optimization problem in Section 2 that the dictionary learning objective function is not convex in dictionary D and coefficients X simultaneously. Nonetheless, the objective function is convex in both the variables separately. Due to this reason, centralized as well as distributed algorithms employ alternate optimization (AO) to solve the problem. Another common theme in distributed dictionary learning algorithms, with the exception of [14], is that sites perform sparse coding locally without any interactions with other sites. The reason [14] differs from the other algorithms is because it learns a part of the dictionary at each site, as opposed to learning a full, common dictionary at each site. In the following, we review the solutions provided by the different works listed in Table 1 for different settings.

3.1 Batch distributed data

We first review the implementation details of distributed dictionary learning algorithms proposed for batch data that is distributed across different sites.

3.1.1 Dictionary learning using diffusion adaptation

In [9] a solution is provided for the optimization problem given in (4). The sparse coding step in [9] is solved locally at each site i using the basis pursuit algorithm [25, p. 161]. For distributed implementation of the dictionary update step, [9] uses the diffusion *adapt-then-combine* (ATC) strategy [17, Chap. 7, p. 459]. Specifically, the distributed dictionary update step at site i and iteration t of dictionary learning is given by following equations:

$$\begin{aligned} \psi_i^{(t)} &= D_i^{(t-1)} + \alpha_i \left(Y_i - D_i^{(t-1)} X_i^{(t)} \right) X_i^{(t)^{\mathsf{T}}}, \text{ and} \\ D_i^{(t)} &= \sum_{l\in\mathcal{N}_i} w_{i,l} \psi_l^{(t)}. \end{aligned} \tag{7}$$

	Common dictionary at sites	Partial/segments of a dictionary at each site
Batch Settings	[9–13]	–
Online Settings	[15], [16]	[24], [14]

Table 1. Different possible setups for dictionary learning problem and their proposed solutions.

Here, $\alpha_i < \frac{2}{\|X_i^{(t)}\|_F}$ is the step size, $\mathcal{N}_i$ is the neighborhood of site i (i.e., set of sites with which i can communicate directly) and w_i is the i^{th} row of a doubly stochastic matrix W, which is defined based on the interconnection among sites in the network (i.e., $w_{i,j} = 0$ if sites i and j are not connected and $w_{i,j} > 0$ otherwise). Numerical simulations in [9] show the efficacy of the proposed algorithm by recovering a dictionary from synthetic image patches and showing that the recovered dictionary resembles the dictionary learned using the centralized version of algorithm.

3.1.2 Dictionary learning using consensus based optimization

The work in [13] provides another solution to the optimization problem given in (4), which is termed EXTRA-AO. In [13], the sparse coding step is performed locally using proximal method in [26]. Next, [13] uses the EXTRA[18] algorithm to update dictionary in a distributed manner. For step size $\alpha > 0$ and $f_i(D_i^{(t-1)}, X_i^{(t-1)}) = \|Y_i - D_i^{(t-1)} X_i^{(t-1)}\|_F^2$, dictionary update in iteration t of EXTRA-AO algorithm is given by

$$D_i^{(t)} = \begin{cases} \sum_{j\in\mathcal{N}_i} w_{i,j} D_j^{(t-1)} - \alpha \nabla_D f_i(D_i^{(t-1)}, X_i^{(t-1)}), \ t = 1, \\ D_i^{(t-1)} + \sum_{j\in\mathcal{N}_i} w_{i,j} D_j^{(t-1)} - \alpha \nabla_D f_i(D_i^{(t-1)}, X_i^{(t-1)}) - \sum_{j\in\mathcal{N}_i} \bar{w}_{i,j} D_j^{(t-2)} + \alpha \nabla_D f_i(D_i^{(t-2)}, X_i^{(t-2)}), \ \text{if } t > 1. \end{cases} \tag{8}$$

Here, $\alpha > 0$ is the step size, $\nabla_D f_i$ is the gradient of local function f_i with respect to D, $\bar{w}_{i,j}$ is the $(i,j)^{th}$ entry of matrix $\bar{W}$, which is defined as $\bar{W} = (W + I)/2$ for a doubly stochastic matrix W. Furthermore, [13] provides convergence analysis and shows that as $t \to \infty$, $D_i^{(t)}$ and $X_i^{(t)}$ converge to the stationary point of the objective function in (4) [13, Proposition 1].

3.1.3 Distribued K-SVD

Like centralized K-SVD,[3] the distributed dictionary learning algorithms proposed in [11] and [12] use orthogonal matching pursuit (OMP)[27] for the sparse coding stage. Since both the algorithms are learning a complete dictionary at each site i, we can perform sparse coding locally at each site. For dictionary update, K-SVD updates one dictionary atom at a time. In order to update one atom at a time, K-SVD solves the optimization problem in (1) by fixing $K-1$ atoms of the dictionary and optimizing over the remaining atom. For dictionary atom k, (1) can be rewritten as

$$(d_{i,k}^{(t)}, x_{i,k,T}^{(t)}) = \arg\min_{d_k, x_{k,T}} \sum_{i=1}^{N} \left\| \left(Y_i - \sum_{j=1, j\neq k}^{K} d_{i,j}^{(t)} x_{i,j,T}^{(t)} \right) - d_{i,k}^{(t)} x_{i,k,T}^{(t)} \right\|_F^2. \tag{9}$$

Cloud K-SVD[11] solves this problem using a distributed variant of power method,[20,28] which is a numerical method for computing the dominant eigenvector of a matrix. In contrast, [12] solves the objective function (9) using ADMM[21] for distributed optimization of (9). Due to the difference in methodology, [11] is able to prove the convergence of the cloud K-SVD solution to the centralized K-SVD solution, while no such analysis is provided by [12]. More details on cloud K-SVD algorithm will be provided in Section 4.

3.2 Online distributed data

The distributed dictionary learning problem for online settings is solved in [15] and [16]. In [15], dictionary learning is carried out for the representation problem, while the goal in [16] is to learn a discriminative dictionary for solving the classification problem.

3.2.1 Distributed online dictionary learning for data representation

In [15], (6) is recast to solve for one row of the dictionary at a time using the D-RLS algorithm.[22] In iteration T_d of dictionary learning, for updating j^{th} row $d_{i,j,T}^{(T_d)}$ of dictionary $D_i^{(T_d)}$ at site i, [15] solves the following optimization problem:

$$d_{i,j,T}^{(T_d)} = \arg \min_{d_i \in \mathbb{R}^K} \frac{1}{T_d} \sum_{i=1}^{N} \sum_{t=1}^{T_d} \zeta^{T_d - t} \frac{1}{2} |y_{i,j}^{(t)} - d_i^{\mathsf{T}} x_i^{(t)}|^2 \quad \text{s.t. } d_i = d_l, \ \forall l \in \mathcal{N}_i. \tag{10}$$

Here, $\zeta \in (0, 1]$ is the forgetting factor. This problem is solved in [15] using D-RLS[22] algorithm, which uses ADMM[21] to solve the distributed optimization problem. Specifically, in iteration T_d of dictionary learning, [15] computes the following two quantities:

$$\begin{aligned} \Phi_i^{(T_d)} &= \zeta \Phi_i^{(T_d - 1)} + x_i^{(T_d)} x_i^{(T_d)^{\mathsf{T}}}, \\ p_{i,j}^{(T_d)} &= \zeta p_{i,j}^{(T_d - 1)} + y_{i,j}^{(T_d)} x_i^{(T_d)}. \end{aligned} \tag{11}$$

In each dictionary learning iteration t, [15] performs R iterations of the RLS algorithm.[22] We can write Lagrangian function of the optimization problem (10) using Lagrange multiplier v. In iteration r of RLS at site i, for all the neighboring nodes $l \in \mathcal{N}_i$, the Lagrangian multipliers are updated as:

$$v_{i,l,j}^{(T_d,r)} = v_{i,l,j}^{(T_d,r-1)} + \frac{c}{2} (d_{i,j,T}^{(T_d,r-1)} - d_{l,j,T}^{(T_d,r-1)}). \tag{12}$$

Using (11) and (12), the update for row j of dictionary is given by

$$d_{i,j,T}^{(T_d,r)} = {\Phi_i^{(T_d)}}^{-1} p_{i,j}^{(T_d)} + \frac{c}{2} \Phi^{(T_d)^{-1}} \left(|\mathcal{N}_i| d_{i,j,T}^{(T_d,r-1)} + \sum_{l \in \mathcal{N}_i \setminus \{i\}} d_{l,j,T}^{(T_d,r-1)} \right) - \frac{1}{2} {\Phi_i^{(T_d)}}^{-1} \sum_{l \in \mathcal{N}_i} (v_{i,l,j}^{(T_d,r)} - v_{l,i,j}^{(T_d,r)}). \tag{13}$$

In order to establish the efficacy of the proposed approach, [15] provides numerical simulations using synthetic data.

3.2.2 Distributed online dictionary learning for data classification

For classification problem let variable z denotes the classifier and label $u_i^{(t)}$ at site i in dictionary learning iteration t, [16] solves the following optimization problem

$$\left\{ D_i^{(T_d)}, z_i^{(T_d)} \right\}_{i=1}^{N} = \arg \min_{D_i \in \mathcal{D}, z_i \in \mathcal{Z}} \frac{1}{T_d} \sum_{t=1}^{T_d} \sum_{i=1}^{N} h_i(D_i, z_i; (y_i^{(t)}, u_i^{(t)})) \quad \text{s.t.} \quad D_i = D_l, \ z_i = z_l, \ l \in \mathcal{N}_i. \tag{14}$$

Here, h is logistic loss, defined as

$$h(D_i, z_i; (y_i^{(t)}, u_i^{(t)})) := \frac{1}{1 + \exp\left(-u_i^{(t)} z_i^{\mathsf{T}} x_i^{(t)}(D_i; y_i^{(t)})\right)}, \tag{15}$$

where, $x_i^{(t)}(D_i; y_i^{(t)})$ is the sparse code of sample $y_i^{(t)}$ using dictionary D_i. We can write Lagrangian function of the optimization problem (14) using Lagrange multipliers Λ and ν. For some step size $\epsilon_t > 0$ chosen as $O(1/t)$, [16, Algorithm 1] provides following update for the primal variables

$$\begin{aligned} D_i^{(t+1)} &= D_i^{(t)} - \epsilon_t (\nabla_{D_i} h_i(D_i^{(t)}, z_i^{(t)}; (y_i^{(t)}, u_i^{(t)}))) + \sum_{l \in \mathcal{N}_i} (\Lambda_{i,l}^{(t)} - \Lambda_{l,i}^{(t)}), \text{ and} \\ z_i^{(t+1)} &= z_i^{(t)} - \epsilon_t (\nabla_{z_i} h_i(D_i^{(t)}, z_i^{(t)}; (y_i^{(t)}, u_i^{(t)}))) + \sum_{l \in \mathcal{N}_i} (\nu_{i,l}^{(t)} - \nu_{l,i}^{(t)}). \end{aligned} \tag{16}$$

The dual variables are updated as[16]

$$\Lambda_{i,l}^{(t+1)} = \Lambda_{i,l}^{(t)} + \epsilon_t(D_i^{(t+1)} - D_l^{(t+1)}), \text{ and}$$
$$\nu_{i,l}^{(t+1)} = \nu_{i,l}^{(t)} + \epsilon_t(z_i^{(t+1)} - z_l^{(t+1)}). \tag{17}$$

Convergence analysis provided in [16] shows that the proposed method asymptotically achieves, in expectation, the first order stationary conditions for the primal variables (i.e., gradient of Lagrangian with respect to primal variables D and z will be zero). Furthermore, feasibility conditions are shown to be satisfied in expectation asymptotically (i.e., gradient of Lagrangian with respect to dual variables Λ and ν will be zero). Finally, [16] provides simulations on real world data to prove the usefulness of proposed method.

3.3 Learning a partial dictionary

All the algorithms we have studied so far were learning the complete dictionary D at each site i in the network. In contrast, [14] learns only a segment of a dictionary at each site, i.e., D is distributed across the network such that $D = [D_1 \cdots D_N]$, where D_i is the segment of dictionary at site i. Another difference in the setup considered in [14] is that every new data point $y_I^{(t)}$ is only observed by some set of sites, i.e., $\mathcal{N}_I \in \mathcal{N}$. In contrast, other methods assume every site i observes a unique data point $y_i^{(t)}$ at time t. Due to these difference in the problem setup, the optimization problem (6) looks different for this scenario. Specifically, the problem is different in following two aspects:

1. In all the other methods, since each site has access to the full dictionary after each dictionary update step, each site is able to carry out sparse coding locally. In the case of partial dictionary learning, each site only has access to a segment of dictionary, because of which we need to perform the sparse coding step in a distributed manner.

2. For the case of complete dictionary learning, one can write the objective function as a 'sum-of-costs' (cf. Sections 2.1, 2.2, and 2.3), while in the case of partial dictionary learning, we are dealing with the case of 'cost-of-sum'.

In terms of the details, the objective function for sparse coding step in [14] is given as follows:

$$\{x^*\}_{i\in\mathcal{N}} = \arg\min_{\{x_i\}_{i\in\mathcal{N}}} f(y_I^{(t)} - \sum_{i=1}^{N} D_i x_i) + \sum_{i=1}^{N} h_{x_i}(x_i). \tag{18}$$

Here, $h_{x_i}(.)$ is a regularization term which is given as $h_{x_i}(x_i) = \|x_i\|_1$ for sparse coding problem (formulations for some other problems are given in [14, Table. I]). Since (18) is not amenable to distributed computations in its present form, [14] considers the dual form of (18). The dual problem for (18) at site i is given by

$$J_i(\nu; y_I^{(t)}) = \begin{cases} \frac{-\nu^\mathsf{T} y_I^{(t)}}{|\mathcal{N}_I|} + \frac{1}{N} f^*(\nu) + h_{x_i}^*(D_i^\mathsf{T}), \ i \in \mathcal{N}_I, \\ \\ \frac{1}{N} f^*(\nu) + h_{x_i}^*(D_i^\mathsf{T}\nu), \ i \notin \mathcal{N}_I. \end{cases} \tag{19}$$

Here, f^* and $h_{x_i}^*$ are conjugate functions of f and h_{x_i}, respectively. Next, in any iteration t the objective is to learn a dual variable $\nu^{(t)}$ that minimizes the dual problem across all the sites, i.e.,

$$\nu^{(t)} = \arg\min_{\nu} \sum_{i=1}^{N} J_i(\nu; y_I^{(t)}) \text{ s.t. } \nu \in \mathcal{V}_f. \tag{20}$$

Where $\mathcal{V}_f$ is the domain of $f(.)$. This distributed optimization problem is solved in [14] using the diffusion strategy.[17] By performing enough diffusion iterations for each sample $y_I^{(t)}$, each site i converges to an estimate

$\nu_i^{(t)}$ that is close to the optimal value $\nu^{(t)}$. Once we have an estimate for $\nu^{(t)}$, and if f and h_{x_i} are strongly convex, we can compute primal values as follows:

$$x_i^{(t)} = \arg\max_{x_i} (D_i^{\mathsf{T}} \nu_i^{(t)})^{\mathsf{T}} x_i - h_{x_i}(x_i). \tag{21}$$

Now using an estimate of the sparse code $x_i^{(t)}$ at site i, we can update dictionary D_i locally as follows:

$$D_i^{(t)} = \Pi_{\mathcal{D}_i}(\text{prox}_{\mu_i . h_{D_i}}(D_i^{(t-1)} + \mu_i \nu_i^{(t)} x_i^{(t)^{\mathsf{T}}})), \tag{22}$$

where, $\text{prox}_{\mu_i . h_{D_i}}(.)$ is the proximal operator and is defined as,

$$\text{prox}_{\mu_i . h_{D_i}}(x) := \arg\min_u \left(\mu_i h_{D_i}(u) + \frac{1}{2}\|u - x\|_2^2 \right).$$

Finally, the effectiveness of the algorithm in [14] is shown by performing simulations over real world datasets.

4. CLOUD K-SVD

In Section 3, we briefly reviewed the algorithms proposed for distributed dictionary learning. Among these methods, cloud K-SVD[11] is the only method that has been shown to converge to one of the centralized algorithms (K-SVD[3]). Over the past few years, usefulness of K-SVD to different data processing tasks has been studied extensively. Since cloud K-SVD converges to the same dictionary as the centralized K-SVD (see Section 4.2), it can be used to solve similar problems in distributed settings. In this section we provide a detailed description of the cloud K-SVD algorithm for distributed dictionary learning in batch data settings.

4.1 Algorithm

Cloud K-SVD solves the dictionary learning problem posed in (2) by means of alternating minimization, i.e., fixing one variable at a time and optimizing over the other variable. We initialize cloud K-SVD with same dictionary D^{init} at each site. In iteration t of cloud K-SVD, we first fix dictionary $D_i^{(t)}$ and update sparse code $X_i^{(t)}$. It is clear from (4) that we can compute sparse code $x_{i,s}^{(t)}$ for each sample $y_{i,s}$ at site i if we know the dictionary at site i. Next we fix $X_i^{(t)}$ and update dictionary $D_i^{(t)}$. In order to update dictionary, we do need access to complete dataset and also the coefficients corresponding to all the samples. Since we cannot aggregate data at a centralized location, we need to develop an algorithm to perform dictionary update step in a distributed manner. Cloud K-SVD updates dictionary $D_i^{(t)}$ by updating one dictionary atom at a time. Furthermore, in addition to updating d_k, K-SVD[3] proposes simultaneously updating coefficients corresponding to atom d_k, denoted by $x_{k,T}$. Here, $x_{k,T}$ is a row vector comprising k^{th} row of matrix X. For updating dictionary atom $d_{i,k}$ and coefficients, $x_{i,k,T}$ we can rewrite (2) as follows:

$$(d_{i,k}^{(t)}, x_{i,k,T}^{(t)}) = \arg\min_{d_k, x_{k,T}} \sum_{i=1}^{N} \| \underbrace{Y_i - \sum_{j=1, j\neq k}^{K} d_{i,j}^{(t)} x_{i,j,T}^{(t)}}_{E_{i,k}^{(t)}} - d_{i,k}^{(t)} x_{i,k,T}^{(t)} \|_F^2. \tag{23}$$

In order to simplify computations, K-SVD[3] further defines an ordered set $\omega_k^{(t)} = \{s : 1 \leq s \leq S, x_{k,T}^{(t)}(s) \neq 0\}$, where $x_{k,T}^{(t)}(s)$ denotes the s^{th} element of $x_{k,T}^{(t)}$, and an $S \times |\omega_k^{(t)}|$ binary matrix $\Omega_k^{(t)}$ that has ones in $(\omega_k^{(t)}(s), s)$ locations and zeros everywhere else. For distributed setup we define $\Omega_{i,k}^{(t)}$ such that $\Omega_k^{(t)} = \text{diag}\{\Omega_{1,k}^{(t)}, \ldots, \Omega_{N,k}^{(t)}\}$. Then, defining $E_{i,k,R}^{(t)} = E_{i,k}^{(t)} \Omega_{i,k}^{(t)}$ and $x_{i,k,T,R}^{(t)} = x_{i,k,T}^{(t)} \Omega_{i,k}^{(t)}$, we can rewrite (23) as

$$(d_{i,k}^{(t)}, x_{i,k,R}^{(t)}) = \arg \min_{d_k, x_{i,k,T}} \sum_{i=1}^{N} \|E_{i,k}\Omega_{i,k}^{(t)} - d_k x_{i,k,T}\Omega_{i,k}^{(t)}\|_F^2$$
$$= \arg \min_{d_k, x_{k,T}} \sum_{i=1}^{N} \|E_{i,k,R}^{(t)} - d_k x_{i,k,T,R}\|_F^2. \quad (24)$$

Solving this equation is equivalent to finding the best rank-one approximation of $E_{k,R}^{(t)} := [E_{1,k,R}^{(t)}, \ldots, E_{N,k,R}^{(t)}]$, which is given by the Eckart-Young theorem as $d_{i,k}^{(t)} x_{k,R}^{(t)} = \sigma_1 u_1 v_1^{\mathsf{T}}$, where u_1 and v_1 denote the largest left- and right-singular vectors of $E_{k,R}^{(t)}$, respectively, while σ_1 denotes the largest singular value of $E_{k,R}^{(t)}$. Since the dominant left singular vector of $E_{k,R}^{(t)}$ is actually the principal eigenvector of $E_{k,R}^{(t)} E_{k,R}^{(t)^{\mathsf{T}}} := \sum_{i=1}^{N} E_{i,k,R}^{(t)} E_{i,k,R}^{(t)^{\mathsf{T}}}$, we can use distributed power method[20] to compute $d_{i,k}^{(t)}$ as shown in Steps 7–18 of Algorithm 1. The distributed power method used to compute $d_{i,k}^{(t)}$ in Algorithm 1 relies on *consensus averaging* to obtain an estimate of eigenvector at each site i (Steps 10–14 in Algorithm 1). Consensus averaging works under the assumption that we have a connected graph $\mathcal{G}$. Based on the topology of $\mathcal{G}$, we define a doubly stochastic matrix W (i.e., $\mathbb{1}^{\mathsf{T}} W = \mathbb{1}^{\mathsf{T}}$ and $W\mathbb{1} = \mathbb{1}$). More specifically, matrix W is designed such that, $(i,j)^{th}$-entry of W, $w_{i,j} > 0$ if sites i and j are connected and $w_{i,j} = 0$ if the sites are not connected. Using this matrix W, each site computes weighted sum of values of its neighboring sites in each iteration of consensus algorithm as shown in Step 13 of Algorithm 1. Once we learn $d_{i,k}^{(t)}$, we can compute $x_{i,k,R}^{(t)}$, the segment of $x_{k,R}^{(t)}$ at site i, using $x_{i,k,R}^{(t)} = d_{i,k}^{(t)^{\mathsf{T}}} E_{i,k,R}^{(t)}$ (Step 18 in Algorithm 1). We keep on repeating this process until a stopping criterion is reached.

4.2 Convergence results

In this section we discuss convergence behavior of cloud K-SVD algorithm. Since in practice we cannot perform infinite consensus iterations and power method iterations, this will result in numerical errors in the dictionary update part (Steps 4–18 in Algorithm 1). Due to these errors it is not clear whether dictionaries, $D_i^{(t)}$, learned using cloud K-SVD will be close to the dictionary learned using centralized K-SVD. Results in [11] show that under some assumptions on centralized K-SVD and network connectivity, if we perform enough consensus iterations T_c and power method iterations T_p, the difference between the outer-product of columns of dictionaries returned by cloud K-SVD $\{D_i^{(t)}\}$ and the centralized dictionary $D^{(t)}$ is small in ℓ_2-norm. In the following we will summarize the convergence results proved in [11].

4.2.1 Algorithmic assumptions

[A1] (*Lasso for sparse coding*) For the purposes of analysis, [11] assumes both cloud K-SVD and centralized K-SVD use the *lasso*[29] method for computing sparse codes (Step 3 in Algorithm 1). Specifically, it is assumed sparse coding is carried out by solving

$$x_{i,s} = \arg \min_{x \in \mathbb{R}^K} \tfrac{1}{2}\|y_{i,s} - Dx\|_2^2 + \tau\|x\|_1 \quad (25)$$

with the regularization parameter $\tau > 0$ selected in a way that $\|x_{i,s}\|_0 \leq T_0 \ll n$. This can be accomplished, for example, by making use of the *least angle regression* algorithm.[30] It is important to note here that while [11] uses *lasso* for the purpose of analysis, other methods like orthogonal matching pursuit (OMP) can be used for sparse coding step and similar kind of analysis can be performed for these methods as well.

[A2] (*Network connectivity*) We assume that the set of nodes $\mathcal{N}$ form a connected network, i.e., any node in the network can reach any other node.

[A3] (*Identical initialization*) Cloud K-SVD and centralized K-SVD are identically initialized, i.e., $D_i^{(0)} = D^{(0)}, i = 1, \ldots, N$, where $D^{(t)}, t \geq 0$, in the following denotes the centralized K-SVD dictionary estimate in the t^{th} iteration.

Algorithm 1: Cloud K-SVD for distributed dictionary learning using batch data

Input: Local data $Y_1, Y_2, \ldots, Y_N$, problem parameters K and T_0, and doubly-stochastic matrix W.

Initialize: Generate $d^{ref} \in \mathbb{R}^n$ and $D^{init} \in \mathbb{R}^{n \times K}$ randomly, set $t \leftarrow 0$ and $D_i^{(t)} \leftarrow D^{init}, i = 1, \ldots, N$.

while *stopping rule* **do**
  $t \leftarrow t + 1$
  (*Sparse Coding*) The i^{th} site solves $\forall s, x_{i,s}^{(t)} \leftarrow \arg\min_{x \in \mathbb{R}^K} \|y_{i,s} - D_i^{(t-1)} x\|_2^2$ s.t. $\|x\|_0 \leq T_0$
  for $k = 1$ **to** K (*Dictionary Update*) **do**
    $E_{i,k,R}^{(t)} \leftarrow Y_i \Omega_{i,k}^{(t)} \; - \sum_{j=1}^{k-1} d_{i,j}^{(t)} x_{i,j,T}^{(t)} \Omega_{i,k}^{(t)} - \sum_{j=k+1}^{K} d_{i,j}^{(t-1)} x_{i,j,T}^{(t)} \Omega_{i,k}^{(t)}$
    $M_i \leftarrow E_{i,k,R}^{(t)} {E_{i,k,R}^{(t)}}^{\mathsf{T}}$
    (***Initialize Distributed Power Method***) Generate q^{init} randomly, set $t_p \leftarrow 0$ and $\widehat{q}_i^{(t_p)} \leftarrow q^{init}$
    while *stopping rule* **do**
      $t_p \leftarrow t_p + 1$
      (***Initialize Consensus Averaging***) Set $t_c \leftarrow 0$ and $z_i^{(t_c)} \leftarrow M_i \widehat{q}_i^{(t_p - 1)}$
      while *stopping rule* **do**
        $t_c \leftarrow t_c + 1$
        $z_i^{(t_c)} \leftarrow \sum_{j \in \mathcal{N}_i} w_{i,j} z_i^{(t_c - 1)}$
      end while
      $\widehat{v}_i^{(t_p)} \leftarrow z_i^{(t_c)} / [W_1^{t_c}]_i$
      $\widehat{q}_i^{(t_p)} \leftarrow \widehat{v}_i^{(t_p)} / \|\widehat{v}_i^{(t_p)}\|_2$
    end while
    $d_{i,k}^{(t)} \leftarrow \operatorname{sgn}\left(\langle d^{ref}, \widehat{q}_i^{(t_p)} \rangle\right) \widehat{q}_i^{(t_p)}$
    $x_{i,k,R}^{(t)} \leftarrow {d_{i,k}^{(t)}}^{\mathsf{T}} E_{i,k,R}^{(t)}$
  end for
end while

Return: $D_i^{(t)}, i = 1, 2, \ldots, N$.

4.2.2 Properties of the K-SVD solution

In addition to Assumptions A1–A3, we also assume that the K-SVD solution satisfies the properties mentioned below.

[P1] Let $x_{i,s}^{(t)}$ denote the solution of the lasso (i.e., (25)) for $D = D^{(t-1)}$ and $\tau = \tau^{(t)}, t = 1, \ldots, T_d$. Then there exists some $C_1 > 0$ such that the following holds:

$$\min_{t,i,s,j \notin \operatorname{supp}(x_{i,s}^{(t)})} \tau^{(t)} - \left|\langle d_j^{(t)}, y_{i,s} - D^{(t-1)} x_{i,s}^{(t)} \rangle\right| > C_1.$$

[P2] Define $\Sigma_{T_0} = \left\{\mathcal{I} \subset \{1, \ldots, K\} : |\mathcal{I}| = T_0\right\}$. Then there exists some $C_2 > \frac{C_1^4 \tau_{\min}^2}{1936}$ such that the following holds:

$$\min_{t=1,\ldots,T_d, \mathcal{I} \in \Sigma_{T_0}} \sigma_{T_0}\left(D_{|\mathcal{I}}^{(t-1)}\right) \geq \sqrt{C_2},$$

where $\sigma_{T_0}(\cdot)$ denotes the T_0^{th} (ordered) singular value of a matrix.

[P3] Let $\lambda_{1,k}^{(t)} > \lambda_{2,k}^{(t)} \geq \ldots \lambda_{n,k}^{(t)} \geq 0$ denote the eigenvalues of the centralized "reduced" matrix $E_{k,R}^{(t)} {E_{k,R}^{(t)}}^{\mathsf{T}}, k \in \{1, \ldots, K\}$, in the t^{th} iteration, $t \in \{1, \ldots, T_d\}$. Then there exists some $C_3 < 1$ such that the following holds:

$$\max_{t,k} \frac{\lambda_{2,k}^{(t)}}{\lambda_{1,k}^{(t)}} \leq C_3.$$

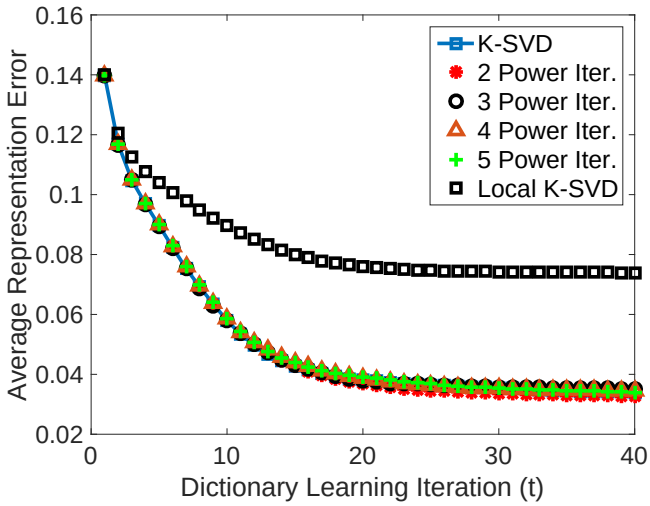

Figure 2. Representation errors for dictionaries learned using centralized K-SVD, cloud K-SVD, and local K-SVD. Here, local K-SVD means we are applying K-SVD only for data that is available at any given site locally.

Properties P1 and P2 correspond to sufficient conditions for $x_{i,s}^{(t)}$ to be a unique solution of (25)[31] and guarantee that the centralized K-SVD generates a unique collection of sparse codes in each iteration. Property P3, on the other hand, ensures that algorithms such as the power method can be used to compute the dominant eigenvector of $E_{k,R}^{(t)}E_{k,R}^{(t)^\mathsf{T}}$ in each dictionary learning iteration.[28] In particular, P3 is a statement about the worst-case spectral gap of $E_{k,R}^{(t)}E_{k,R}^{(t)^\mathsf{T}}$.

4.2.3 Main result

The main result for cloud K-SVD is given in terms of the $\|\cdot\|_2$ norm mixing time, T_{mix}, of the Markov chain associated with the doubly-stochastic weight matrix W, defined as

$$T_{mix} = \max_{i=1,\ldots,N} \inf_{t\in\mathbb{N}} \left\{ t : \|\mathrm{e}_i^\mathsf{T} W^t - \tfrac{1}{N}\mathbf{1}^\mathsf{T}\|_2 \leq \frac{1}{2} \right\}. \tag{26}$$

Here, $\mathrm{e}_i \in \mathbb{R}^N$ denotes the i^{th} column of the identity matrix I_N. Note that T_{mix} can be upper bounded in terms of inverse of the absolute spectral gap of W, defined as $1-|\lambda_2(W)|$ with $\lambda_2(W)$ denoting the second largest (in modulus) eigenvalue of W.[32] As a general rule, better-connected networks can be made to have smaller mixing times compared to sparsely connected networks [32, Chap. 15],[33].

Theorem 4.1 (Stability of cloud K-SVD Dictionaries). *Under assumptions A1–A3 and assuming that K-SVD algorithm satisfies properties P1–P3, suppose we perform* $T_p = \Omega(T_d K\log(N) + T_d\log(KST_0) + \log(nN\delta_d^{-1}))$ *power method iterations and* $T_c = \Omega\big(T_p T_{mix} + T_{mix}\log N\big)$ *consensus iterations. Then at the end of* T_d *dictionary learning iterations, we will have*

$$\max_{\substack{i=1,\ldots,N\\ k=1,\ldots,K}} \left\| d_{i,k}^{(T_d)} d_{i,k}^{(T_d)^\mathsf{T}} - d_k^{(T_d)} d_k^{(T_d)^\mathsf{T}} \right\|_2 \leq \delta_d. \tag{27}$$

This result shows that we need to perform consensus iterations whose number scale linearly with that of power method iteration T_p and mixing time of graph T_{mix}, while this scaling is logarithmic in the number of nodes. Another important point to note here is that the number of power method iterations needed in this case scale only logarithmically with the number of samples S, which is a desirable thing in the context of big data problems.

4.3 Numerical results

In this section we provide numerical results to demonstrate the efficacy of cloud K-SVD for distributed data processing tasks. In our first example we will use synthetic data to establish effectiveness of cloud K-SVD for data representation problems. Our second example uses cloud K-SVD to solve a classification problem for MNIST dataset.[34]

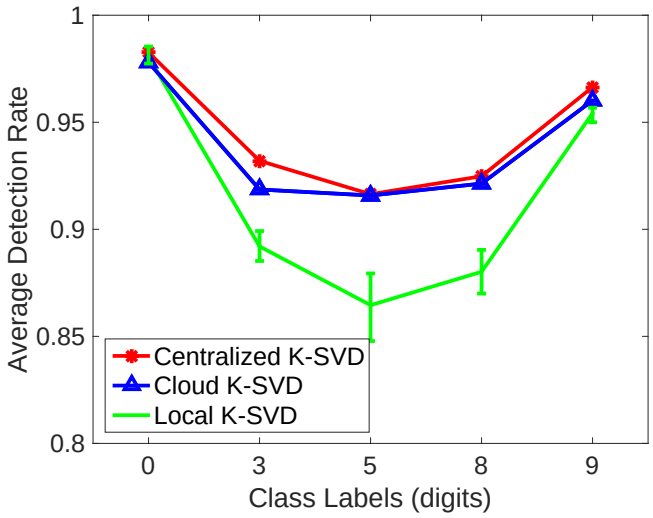

Figure 3. Classification for MNIST dataset using centralized K-SVD, cloud K-SVD, and local K-SVD.

4.3.1 Representation error for synthetic data

Our first experiment deals with the data representation problem and demonstrates the effectiveness of collaboratively learning a dictionary using cloud K-SVD, as opposed to each site learning a *local dictionary* from its local data using the canonical K-SVD algorithm (referred to as *local K-SVD* in the following). The experimental setup consists of a total of $N = 100$ sites, with each site i having local data $Y_i \in \mathbb{R}^{20\times 500}$, i.e., at each site i we have $S_i = 500$ local samples of dimension $n = 20$. Interconnections between the sites are randomly generated using an Erdős–Rényi graph with parameter $p = 0.5$. In order to generate synthetic data at individual sites, we first generate a dictionary with $K = 50$ atoms, $D \in \mathbb{R}^{20\times 50}$, with columns uniformly distributed on the unit sphere in $\mathbb{R}^{20}$. Next, we randomly select a 45-column subdictionary of D for each site and then generate samples for that site using a linear combination of $T_0 = 3$ randomly selected atoms of this subdictionary, followed by addition of white Gaussian noise with variance $\sigma^2 = 0.01$. All data samples in our experiments are also normalized to have unit ℓ_2 norms. Sparse coding in these experiments is performed using an implementation of OMP provided in [35]. Finally, in order to carry out distributed consensus averaging, a doubly-stochastic weight matrix W is generated according to the local-degree weights method described in [19, Sec. 4.2].

We also carry out experiments to demonstrate variations in cloud K-SVD results when we change the number of power method iterations (T_p). In Figure 2, we plot average representation error, which is defined as $\frac{1}{nS}\sum_{i=1}^{N}\sum_{j=1}^{S_i}\|y_{i,j} - Dx_{i,j}\|_2$, as a function of the number of dictionary learning iterations for three dictionary learning methods, namely, centralized (canonical) K-SVD, cloud K-SVD, and local K-SVD. It can be seen from this figure, which corresponds to an average of 100 Monte-Carlo trials, that cloud K-SVD and centralized K-SVD have similar performance and both of them perform better than local K-SVD. In particular, the local K-SVD error is ≈ 0.06 after 40 dictionary learning iterations (i.e., $T_d = 40$), while it is ≈ 0.03 for cloud K-SVD and centralized K-SVD. Notice that changes in the number of power method iterations induce relatively minor changes in the representation error of cloud K-SVD.

4.3.2 Classification of MNIST dataset

For evaluation of cloud K-SVD on real dataset, we perform classification of digits $\{0, 3, 5, 8, 9\}$ from MNIST dataset.[34] We use 6000 samples for each digit, where 5000 samples are used for training and remaining 1000 for testing. The data are five-times randomly split into training and test samples. For cloud K-SVD, Erdős–Rényi graph with parameter $p = 0.5$ is used to generate a network with 10 sites and data is equally distributed among them. Before performing dictionary learning, data is down sampled from $\mathbb{R}^{784}$ to $\mathbb{R}^{256}$. Next, a separate dictionary is learned for each digit using centralized K-SVD, cloud K-SVD, and local K-SVD. Each dictionary has dimensions $\mathbb{R}^{256\times 400}$, i.e., $K = 400$, and sparsity level of $T_0 = 10$ is used. Minimum residue-based rule [6, Sec.II-A] is used for classification, more details on which are given below.

Let $\{D_c\}_{c=1}^5$ be the set of dictionaries for 5 classes and let $D = \begin{bmatrix} D_1 & D_2 & D_3 & D_4 & D_5 \end{bmatrix} \in \mathbb{R}^{256\times 2000}$ be the complete dictionary. For any test sample y_s, we perform sparse coding using D with sparsity constraint of $T_0 = 10$ to get coefficients $x_s \in \mathbb{R}^{2000}$. Then we partition x_s into five segment $\{x_{s,c}\}_{c=1}^5$, where $x_{s,c}$ are the coefficients corresponding to dictionary D_c of class c. Next we define residue for class c as

$$r_c = \|y_s - D_c x_{s,c}\|_2. \tag{28}$$

Finally, the detected class is given by $c^* = \arg\min_c r_c$. Performance of each method (centralized, cloud, and local K-SVD) is measured in terms of average detection rate on the test samples, defined as

$$R_c = \frac{\text{Number of samples in class } c \text{ detected correctly}}{\text{Total number of samples of class } c}, \tag{29}$$

with results given in Figure 3. We see that centralized and cloud K-SVD have comparable performance. But in the case of local K-SVD, classification rate deteriorates considerably. The bars in local K-SVD show the highest and lowest detection rates achieved among the 10 sites, which highlights the variation in effectiveness of models learned across different sites when using only the local data.

5. CONCLUSION

In this paper we reviewed existing methods for solving dictionary learning problem in a distributed setup. These methods target different problem setups that arise in practical situations. Specifically, (*i*) streaming and batch data settings, (*ii*) learnig dictionary for representation and classification tasks, and (*iii*) learning a full common dictionary versus partial dictionary at sites were examined. Furthermore, we provided detailed description of the cloud K-SVD solution for learning distributed dictionary in batch data settings. We also provided the convergence results and numerical simulations to show the efficacy of cloud K-SVD.

ACKNOWLEDGMENTS

This work is supported in part by the ARO under grant W911NF-14-1-0295 and by the NSF under grants CCF-1453073 and CCF-1525276.

REFERENCES

[1] Zhang, Q. and Li, B., "Discriminative K-SVD for dictionary learning in face recognition," in [*Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2010)*], 2691–2698 (2010).
[2] Jiang, Z., Lin, Z., and Davis, L. S., "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in [*Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2011)*], 1697–1704 (2011).
[3] Aharon, M., Elad, M., and Bruckstein, A., "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.* **54**(11), 4311–4322 (2006).
[4] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research* **11**, 19–60 (2010).
[5] Olshausen, B. A. and Field, D. J., "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision research* **37**(23), 3311–3325 (1997).
[6] Chen, Y., Nasrabadi, N. M., and Tran, T. D., "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.* **49**(10), 3973–3985 (2011).
[7] Mairal, J., Bach, F., and Ponce, J., "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. and Mach. Intell.,* **34**(4), 791–804 (2012).
[8] Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T., and Sejnowski, T. J., "Dictionary learning algorithms for sparse representation," *Neural computation* **15**(2), 349–396 (2003).
[9] Chainais, P. and Richard, C., "Learning a common dictionary over a sensor network," in [*Proc. IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*], 133–136, IEEE (2013).
[10] Raja, H. and Bajwa, W. U., "Cloud K-SVD: Computing data-adaptive representations in the cloud," in [*Proc. 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*], 1474–1481, IEEE (2013).
[11] Raja, H. and Bajwa, W. U., "Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data," *IEEE Trans. Signal Process.* **64**(1), 173–188 (2016).
[12] Liang, J., Zhang, M., Zeng, X., and Yu, G., "Distributed dictionary learning for sparse representation in sensor networks," *IEEE Trans. Image Process.* **23**(6), 2528–2541 (2014).

[13] Wai, H.-T., Chang, T.-H., and Scaglione, A., "A consensus-based decentralized algorithm for non-convex optimization with application to dictionary learning," in [*Proc. IEEE International Conference on Acoustics, Speech and Signal Process. (ICASSP)*], 3546–3550, IEEE (2015).

[14] Chen, J., Towfic, Z. J., and Sayed, A. H., "Dictionary learning over distributed models," *IEEE Trans. Signal Process.* **63**(4), 1001–1016 (2015).

[15] Chouvardas, S., Kopsinis, Y., and Theodoridis, S., "An online algorithm for distributed dictionary learning," in [*Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 3292–3296, IEEE (2015).

[16] Koppel, A., Warnell, G., Stump, E., and Ribeiro, A., "D4l: Decentralized dynamic discriminative dictionary learning," in [*Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 2966–2973, IEEE (2015).

[17] Sayed, A. H., "Diffusion adaptation over networks," *Academic Press Library in Signal Processing* **3**, 323–454 (2013).

[18] Shi, W., Ling, Q., Wu, G., and Yin, W., "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization* **25**(2), 944–966 (2015).

[19] Xiao, L. and Boyd, S., "Fast linear iterations for distributed averaging," *Systems & Control Letters* **53**(1), 65–78 (2004).

[20] Kempe, D. and McSherry, F., "A decentralized algorithm for spectral analysis," *J. Comput. and Syst. Sci.* **74**(1), 70–83 (2008).

[21] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning* **3**(1), 1–122 (2011).

[22] Mateos, G., Schizas, I. D., and Giannakis, G. B., "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Trans. Signal Process.* **57**(11), 4583–4588 (2009).

[23] Koppel, A., Jakubiec, F. Y., and Ribeiro, A., "A saddle point algorithm for networked online convex optimization," *IEEE Trans. Signal Process.* **63**(19), 5149–5164 (2015).

[24] Chen, J., Towfic, Z. J., and Sayed, A. H., "Online dictionary learning over distributed models," in [*Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 3874–3878, IEEE (2014).

[25] Starck, J.-L., Murtagh, F., and Fadili, J. M., [*Sparse image and signal processing: wavelets, curvelets, morphological diversity*], Cambridge university press (2010).

[26] Beck, A. and Teboulle, M., "Gradient-based algorithms with applications to signal recovery," *Convex Optimization in Signal Processing and Communications* , 42–88 (2009).

[27] Tropp, J. A. and Gilbert, A. C., "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inform. Theory* **53**(12), 4655–4666 (2007).

[28] Golub, G. H. and Van Loan, C. F., [*Matrix computations*], Johns Hopkins University Press, Baltimore, MD, third ed. (2012).

[29] Tibshirani, R., "Regression shrinkage and selection via the lasso," *J. Royal Stat. Soc. Series B (Methodological)* **58**(1), 267–288 (1996).

[30] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R., "Least angle regression," *Ann. Stat.* **32**(2), 407–451 (2004).

[31] Fuchs, J.-J., "On sparse representations in arbitrary redundant bases," *IEEE Trans. Inform. Theory* **50**, 1341–1344 (June 2004).

[32] Levin, D. A., Peres, Y., and Wilmer, E. L., [*Markov chains and mixing times*], American Math. Soc. (2009).

[33] Boyd, S. P., Ghosh, A., Prabhakar, B., and Shah, D., "Mixing times for random walks on geometric random graphs.," in [*Proc. The Algorithm Engineering and Experiments (ALENEX) and Analytic Algorithmics and Combinatorics (ANALCO)*], 240–249, SIAM (2005).

[34] LeCun, Y. and Cortes, C., "The MNIST database of handwritten digits." http://yann.lecun.com/exdb/mnist/ (1998).

[35] Rubinstein, R., Zibulevsky, M., and Elad, M., "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS Dept. Technion - Israel Institute of Technology, Haifa 32000 Israel* **40**(8), 1–15 (2008).