Cloud K-SVD: Computing data-adaptive representations in the cloud

Haroon Raja and Waheed U. Bajwa
Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854
Emails: haroon.raja@rutgers.edu, waheed.bajwa@rutgers.edu

***Abstract*—This paper studies the problem of data-adaptive representations for big, distributed data. It is assumed that a number of geographically-distributed, interconnected sites have massive local data and they are interested in collaboratively learning a low-dimensional geometric structure underlying these data. In contrast to some of the previous works on subspace representations, this paper focuses on the geometric structure of a union of subspaces (UoS). Specifically, it proposes a distributed algorithm, termed as cloud K-SVD, for learning a UoS structure underlying distributed data of interest. Cloud K-SVD accomplishes the goal of collaborative data-adaptive representations without requiring communication of individual data samples between different sites. The paper also provides a partial analysis of cloud K-SVD that gives insights into its convergence properties and deviations from a centralized solution in terms of properties of local data and topology of interconnections. Finally, it numerically illustrates the efficacy of cloud K-SVD.**

I. Introduction

We are witnessing a new era of information processing. Two defining characteristics of this era are *big data* and *distributed data*. First, ever-falling prices of semiconductor devices, consumer electronics, storage media, communications bandwidth, etc., are pushing individuals, corporations, and governments toward generation of massive quantities of data. Second, globalization, mobility, ubiquitous connectivity, and other extraneous factors are causing an increasing reliance on interconnected sets of geographically-distributed data collections for inference and decision making. The confluence of these two characteristics is inevitable, which will create both new challenges and opportunities. In this paper, we focus on one such challenge in the "big, distributed data" world of the future: *collaborative learning of compact data representations, suitable for a multitude of information processing tasks, using interconnected sets of geographically-distributed big data.*

A. Background

Modern information processing is based on the axiom that while real-world data may live in high-dimensional ambient spaces, relevant information within them almost always lies near low-dimensional geometric structures. In particular, knowledge of the (low-dimensional) geometric structure underlying data of interest is central to the success of a multitude of information processing tasks. But this knowledge is unavailable to us in an overwhelmingly large number of applications and a great deal of work has been done in the past to *learn* geometric structure of data from the data *themselves*. However, much of that work, often studied under rubrics such as *principal component analysis* (PCA) [1], *generalized PCA* (GPCA) [2], *hybrid linear modeling* (HLM) [3], and *dictionary learning* [4], [5], has been focused on methods in which the entire data are assumed available at a centralized location. In recent years, there has been some effort to extend these works to distributed settings; see, e.g., [6]–[11]. The setting studied in some of these works is that data is partitioned *horizontally*, with each distributed entity responsible for some dimensions of the data [6], [9]. Some of the other works in this direction focus on learning under the assumption of data lying near *(linear) subspaces* [7], [8], require extensive communications among the distributed entities [10], and ignore some of the technical details associated with processing among distributed entities having interconnections described by graphs of arbitrary topologies [7], [8], [10], [11].

In this paper, we are interested in a setting in which a number of geographically-distributed sites have massive local data and these sites are interested in collaboratively learning a geometric structure underlying their data by communicating among themselves over public and/or private networks. The key constraints in this problem, which we term *data-adaptive representations in the cloud*, that distinguish it from some of the prior related works are that (i) sites cannot communicate "raw" data among themselves and (ii) interconnections among sites cannot be described by a complete graph. Both these constraints are reflective of the future of big, distributed data in the world. In particular, the former constraint is justified because of the size of local data compilations as well as privacy concerns in the modern age. Similarly, the latter constraint is justified because linking geographically-distributed sites into a complete graph is cost prohibitive.

B. Our Contributions

One of the contributions of this paper is formulation of a distributed method, which we term as *cloud K-SVD*, that enables data-adaptive representations in the cloud. In contrast to works that assume a linear geometric structure for data [6]–[9], cloud K-SVD is based on the premise that data lie near a *union* of low-dimensional subspaces. The *union-of-subspaces* (UoS) model is a nonlinear generalization of the subspace model [12] and has received widespread acceptance in the community lately. The task of learning the UoS underlying data of interest from data themselves is often termed *dictionary learning* in the literature [4], [5]; in particular, dictionary learning—when compared to linear data-adaptive representations such as the PCA and the linear discriminant analysis [13]—has been shown to be highly effective for information processing tasks such as denoising [14], compression [4], object recognition [15], and inpainting [16]. Cloud K-SVD, as the name implies, is a distributed variant of the popular dictionary learning algorithm K-SVD [5] and relies on a classical iterative eigenvector algorithm [17, Ch. 8] and consensus averaging [18] for collaborative dictionary learning.

In addition, we also provide a partial analysis of cloud K-SVD that gives insights into its convergence properties and deviations from the centralized solution in terms of properties of local data and topology of interconnections. Finally, we carry out numerical experiments to demonstrate both the efficacy of cloud K-SVD and the usefulness of collaborative dictionary learning as opposed to local dictionary learning.

C. Relationship to Previous Work

Some of the earliest works in distributed information processing date back nearly three decades, such as distributed Kalman filtering [19] and consensus [20]. Since then a number of distributed methods have been proposed for myriad information processing tasks. Some recent examples of this that do not involve a centralized *fusion center* include methods for distributed classification [21]–[23], distributed linear regression [24], and distributed estimation [25]. However, relatively little attention has been paid to the problem of distributed learning of the geometric structure of data in a data-adaptive manner. Most notable exceptions to this include [7]–[11]. While our work as well as [7]–[11] rely on consensus averaging for computing the underlying geometric structure, we are explicit in our formulation that perfect consensus under arbitrary topologies cannot be achieved. In contrast, developments in [7]–[11] are carried out under the assumption of infinite-precision consensus averaging. Further, [7], [8] assume a subspace model for data, while [10] advocates the use of consensus averaging for computing sample covariance—an approach that requires extensive communications among the distributed entities.

To the best of our knowledge, the work presented in here is the first one to investigate dictionary learning in a distributed setting. The basis for this work is the centralized dictionary learning algorithm of K-SVD [5], which involves computing the largest left- and right-singular vectors of certain matrices for a total of K times in each iteration of the algorithm. At the heart of cloud K-SVD then is a way of carrying out this step within the algorithm in a distributed manner. We accomplish this through the use of consensus averaging coupled with the classical *power method* for computing eigenvectors of a matrix. Superficially, this makes our approach appear similar to the ones taken in [7], [26] for computing eigenvectors of a matrix in a distributed manner using the power method. However, as noted earlier, we do not assume perfect consensus during iterations of the power method, which leaves open the question of convergence of the distributed variant of the power method—an issue that does not arise in [7], [26]. In fact, one of our main contributions is analytically characterizing the convergence behavior of this distributed step within the cloud K-SVD algorithm. The nature of this analysis is reminiscent of the one carried out in [27] in the context of convergence behavior of distributed eigenanalysis of a network using a power method-like iterative algorithm. However, there are differences in the analysis of [27] and our work because of the exact place where consensus averaging is carried out in the two works, which is dictated by the distinct nature of the two applications.

D. Notation and Organization

We use lower-case letters to represent scalars and vectors, while we use upper-case letters to represent matrices. Superscript $(\cdot)^\mathsf{T}$ denotes the transpose operation, $\|\cdot\|_0$ counts the number of nonzero entries in a vector, $\|v\|_p$ denotes the usual ℓ_p norm of vector v, while $\|A\|_F$ and $\|A\|_2$ denote the Frobenius norm and the operator norm of matrix A. Finally, given a matrix A, a_j and $a_{j,T}$ denote the j^{th} column and the j^{th} row of A, respectively.

The rest of this paper is organized as follows. In Section II, we formulate the problem of data-adaptive representations in the cloud under the UoS model. In Section III, we describe the cloud K-SVD algorithm. In Section IV, we provide preliminary analysis of some aspects of the cloud K-SVD algorithm. Finally, we provide some numerical results in Section V and concluding remarks in Section VI.

II. Problem Formulation

In this paper, we consider a collection of N geographically-distributed sites that are interconnected to each other according to a fixed topology. Here, we use "site" in the broadest possible sense of the term, with a site corresponding to a single computational system (e.g., sensor, drone, smartphone, tablet, server, database), a collection of co-located computational systems (e.g., data center, computer cluster, robot swarm), etc. Mathematically, we represent this collection and the interconnections through an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, \cdots, N\}$ denotes the sites and $\mathcal{E}$ denotes edges in the graph with $(i,i) \in \mathcal{E}$, while $(i,j) \in \mathcal{E}$ whenever there is a connection between site i and site j. The only assumption we make about the topology of $\mathcal{G}$ is that it is a connected graph.

Next, we assume that each site i has a massive collection of local data, expressed as a matrix $Y_i \in \mathbb{R}^{n \times S_i}$ with $S_i \gg n$ representing the number of data samples at the i^{th} site. We can express all of this distributed data into a single matrix $Y = \begin{bmatrix} Y_1 & Y_2 & \dots & Y_N \end{bmatrix} \in \mathbb{R}^{n \times S}$, where $S = \sum_{i=1}^{N} S_i$ denotes the total number of data samples distributed across the N sites; see Fig. 1 for a schematic representation of this. In this setting, the fundamental objective is for each site to collaboratively learn a low-dimensional geometric structure that underlies the global (distributed) data Y. The basic premises behind collaborative structure learning of global data, as opposed to local structure learning of local data, are manifold. First, since the number of global samples is much larger than the number of local samples, we expect that collaborative learning for data

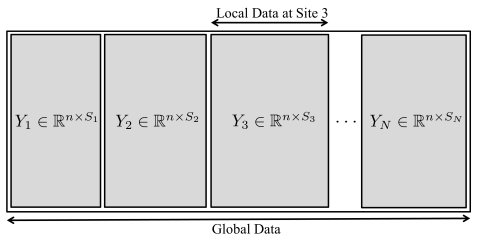
Local Data at Site 3
$Y_1 \in \mathbb{R}^{n\times S_1}$
$Y_2 \in \mathbb{R}^{n\times S_2}$
$Y_3 \in \mathbb{R}^{n\times S_3}$
$\cdots$
$Y_N \in \mathbb{R}^{n\times S_N}$
Global Data

Fig. 1. A schematic representing data Y distributed across N sites.

representations will outperform local learning. Second, local learning will be strictly suboptimal for some sites in cases where sampling density, noise level, fraction of outliers, etc., are not uniform across all sites. Collaborative learning, on the other hand, evens out such nonuniformities within local data.

The fundamental assumption in this paper is that the low-dimensional geometric structure underlying the global (and local) data corresponds to a union of T_0-dimensional subspaces in $\mathbb{R}^n$, where $T_0 \ll n$. One possible means of learning such a geometric structure is studied under the moniker dictionary learning, which learns an *overcomplete dictionary* D such that each data sample is well approximated by no more than T_0 columns (i.e., *atoms*) of D [4], [5]. Specifically, assuming that the global data Y is available at a centralized location, the problem of dictionary learning can be expressed as

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

where $D \in \mathbb{R}^{n\times K}$ is an overcomplete dictionary having unit-norm columns, $X \in \mathbb{R}^{K\times S}$ corresponds to representation coefficients of the data having no more than T_0 nonzero coefficients per sample, and x_s denotes the s^{th} column in the coefficient matrix X. The problem of dictionary learning, as expressed in (1), is non-convex in $\big(D, X\big)$, although it is convex in D alone. One of the most popular approaches to dictionary learning therefore involves alternate minimization in which one alternates between solving (1) for D using a fixed X and then solving (1) for X using a fixed D [5], [28].

Unlike the classical literature on dictionary learning, however, we do not have the global data Y available at a centralized location. In particular, data aggregation either at a centralized location or at any one of the individual sites is impractical due to communications and storage costs of big data. Instead, our goal is to have individual sites collaboratively learn dictionaries $\{\widehat{D}_i\}_{i\in\mathcal{N}}$ such that the performance of these *collaborative dictionaries* on global data Y is close to the performance of a dictionary D that could have been learned from Y in a centralized fashion. In addition, we are interested in accomplishing this goal without ever exchanging individual samples between sites because of privacy concerns. In the following, we present a distributed variant of a popular dictionary learning algorithm that accomplishes this goal.

III. Proposed Algorithm

A number of methods have been proposed for dictionary learning in the past. In this section, we focus on the well-studied K-SVD algorithm [5] as the basis for distributed dictionary learning. We have chosen to work with K-SVD because of its iterative nature and its reliance on the SVD, both of which enable its exploitation for distributed purposes. In the following, we first provide a brief overview of the K-SVD algorithm, followed by our proposed algorithm—termed cloud K-SVD—for distributed dictionary learning.

A. Dictionary Learning Using K-SVD

The K-SVD algorithm initializes with a random dictionary D and solves (1) by iterating between two stages: a *sparse coding stage* and a *dictionary update stage* [5]. Specifically, for a fixed estimate of the dictionary D, the sparse coding stage involves solving for X as follows:

$$\forall j, \quad x_j = \arg\min_{x_j} \|y_j - Dx_j\|_2^2 \quad \text{s.t.} \quad \|x_j\|_0 \leq T_0. \quad (2)$$

Here, x_j is the j^{th} column of X. Note that although sparse coding, as stated in (2), has combinatorial complexity, it can be easily solved by either convexifying (2) [29] or using greedy pursuit algorithms [30].

After the sparse-coding stage, K-SVD fixes X and moves to the dictionary update stage. The main novelty in K-SVD lies in the manner in which it carries out dictionary update. Specifically, it iterates through the K atoms of D, solving for the k^{th} atom as follows:

$$\begin{aligned} d_k &= \arg\min_{d_k\in\mathbb{R}^n} \left\| \left(Y - \sum_{j=1, j\neq k}^{K} d_j x_{j,T}\right) - d_k x_{k,T} \right\|_F^2 \\ &= \arg\min_{d_k\in\mathbb{R}^n} \left\|E_k - d_k x_{k,T}\right\|_F^2 . \end{aligned} \quad (3)$$

Here, E_k is the representation error for sample data Y using all atoms of D except the k^{th} atom. Next, K-SVD, as proposed in [5], defines an ordered set $\omega_k = \{i : 1 \leq i \leq K, x_{k,T}(i) \neq 0\}$ and an $S \times |\omega_k|$ binary matrix Ω_k that has ones in $(\omega_k(i), i)^{th}$ entry and zero everywhere else. It is then easy to see from (3) and these definitions that

$$d_k = \arg\min_{d_k} \left\|E_{k,R} - d_k x_{k,R}\right\|_F^2 , \quad (4)$$

where $E_{k,R} = E_k\Omega_k$ and $x_{k,R} = x_{k,T}\Omega_k$.

Solving (4) is equivalent to finding the best rank-1 approximation of $E_{k,R}$, given by the Eckart–Young theorem as:

$$d_k x_{k,R} = \sigma_1 u_1 v_1^{\mathsf{T}}, \quad (5)$$

where u_1 and v_1 correspond to the largest left- and right-singular vectors of $E_{k,R}$, while σ_1 denotes the largest singular value of $E_{k,R}$. It then follows that the k^{th} atom of the dictionary can simply be set equal to $d_k = u_1$. In K-SVD, it is further advocated that the k^{th} row of the "reduced" coefficient matrix, $x_{k,R}$, should be simultaneously updated to $x_{k,R} = \sigma_1 v_1^{\mathsf{T}}$.

The dictionary update stage in K-SVD involves K such applications of the Eckart–Young theorem to update the K atoms of D and the K "reduced" rows of X. The algorithm then moves to the sparse coding stage and continues alternating between the two stages till a stopping criterion (e.g., a prescribed representation error) is reached. We conclude our discussion of the K-SVD algorithm by noting that it is guaranteed to converge to a local minimum [5].

B. Distributed Dictionary Learning Using Cloud K-SVD

In this section, we propose a distributed dictionary learning algorithm based on the K-SVD method. The key to distributing K-SVD is understanding ways in which both the sparse coding and the dictionary update stages can be distributed across the N sites. To this end, we assume distributed dictionary learning is in iteration $t+1$ and each site in this iteration has a local estimate $\widehat{D}_i^{(t)}$ of the desired dictionary. In order for the sparse coding stage to proceed, we propose that each site computes the representation coefficients of its local data without collaborating with other sites by locally solving

$$\forall s, x_{s,i} = \arg\min_{x_{s,i}} \|Y_i - \widehat{D}_i^{(t)} x_{s,i}\|_F^2 \quad \text{s.t.} \quad \|x_{s,i}\|_0 \leq T_0, \tag{6}$$

where $\{x_{s,i}\}_{i=1}^{S_i}$ is the coefficient vector for the s^{th} sample at site i. Note that this "local" sparse coding for distributed dictionary learning greatly simplifies the sparse coding stage and is justified as long as the local dictionary estimates $\widehat{D}_i^{(t)}$ remain reasonably close to each other.

The next challenge in distributed dictionary learning based on the K-SVD algorithm arises during the dictionary update stage. Recall from the previous section that the dictionary update stage involves computing the largest left- and right-singular vectors of the error matrix $E_{k,R} = E_k\Omega_k$ for $k = 1, \ldots, K$. However, unless the local dictionary estimates $\widehat{D}_i^{(t)}$ happen to be identical, we end up with N such error matrices in a distributed setting due to N different local dictionary estimates. In order to resolve this, we propose to use the following definition of the error matrix in a distributed setting:[1] $E_{k,R} := \begin{bmatrix} E_{1,k,R} & E_{2,k,R} & \ldots & E_{N,k,R}\end{bmatrix}$, where

$$E_{i,k,R} := Y_i\Omega_{i,k} - \sum_{j\neq k} \widehat{d}_{i,j} x_{i,j,T}\Omega_{i,k}. \tag{7}$$

Here, $x_{i,j,T}$ is the j^{th} row of coefficient matrix X_i available at site i, $\Omega_{i,k}$ is similar to Ω_k defined for K-SVD except that it is defined for only local data at site i, and $\widehat{d}_{i,j}$ is the estimate of dictionary's j^{th} atom after t iterations of dictionary learning.

Next, in keeping with the development in the centralized K-SVD algorithm, we propose that each of the N sites updates the kth atom of its respective local dictionary and the k^{th} row of its respective "reduced" coefficient matrix, $x_{i,k,R} := x_{i,k,T}\Omega_{i,k}$ by collaboratively computing the dominant left- and right-singular vectors of the *distributed error matrix* $E_{k,R}$. In fact, it turns out that we need only worry about the dominant left-singular vector, u_1, of $E_{k,R}$ in this case. Indeed, since $u_1^\mathsf{T} E_{k,R} = \sigma_1 v_1$ with v_1 being the dominant right-singular vector of $E_{k,R}$, each site can update the k^{th} row of its respective "reduced" coefficient matrix by setting $\widehat{d}_{i,k} = u_1$ and setting $x_{i,k,R} = \widehat{d}_{i,k}^\mathsf{T} E_{i,k,R}$. Now define $M = E_{k,R}E_{k,R}^\mathsf{T}$ and notice that u_1 corresponds to the dominant eigenvector of M. Further, we can express this matrix M as $M := \sum_{i=1}^N M_i$, where each $M_i = E_{i,k,R}E_{i,k,R}^\mathsf{T}$ is a matrix that is readily computable at each local site. Our goal now is computing the dominant eigenvector of $M = \sum_{i=1}^N M_i$ in a distributed manner at each site. In order for this, we make use of distributed power method, which has been invoked previously in [7], [8], [27] and which corresponds to a distributed variant of the classical power method for eigenanalysis [17].

1) Distributed Power Method: Power method is an iterative method for computing eigenvectors of a matrix. It is simple to implement and, assuming that the largest eigenvalue λ_1 is strictly greater than the second-largest eigenvalue λ_2, it converges to the subspace spanned by the dominant eigenvector at an exponential rate. In this paper, we are interested in a distributed variant of the power method to compute the dominant eigenvector of $M = \sum_{i=1}^N M_i$, where the M_i's are distributed across N sites. To this end, we proceed as follows.

First, all sites initialize to the same (unit-norm) estimate of the eigenvector $\widehat{q}_i^{(0)} = q^{init}$.[2] Next, assuming that the sites are carrying out iteration t_m of the distributed power method, each site computes $M_i\widehat{q}_i^{(t_m-1)}$ locally, where $\widehat{q}_i^{(t_m-1)}$ denotes an estimate of the dominant eigenvector of M at the ith site after $t_m - 1$ iterations. In the next step, the sites collaboratively attempt to compute an approximation $\widehat{v}_i^{(t_m)}$ of $\sum_i M_i\widehat{q}_i^{(t_m-1)}$ at each site. In the final step of the t_m^{th} iteration of the distributed power method, each site normalizes its estimate of the dominant eigenvector of M locally, $\widehat{q}_i^{(t_m)} := \widehat{v}_i^{(t_m)}[\widehat{r}_i^{(t_m)}]^{-1}$, where $\widehat{r}_i^{(t_m)} := \|\widehat{v}_i^{(t_m)}\|_2$.

It is clear from the preceding discussion that the key in the distributed power method is the ability of the sites to collaboratively compute an approximation of $\sum_i M_i\widehat{q}_i^{(t_m-1)}$ in each iteration. In order for this, we make use of the popular consensus averaging method [31]. To perform consensus averaging, we first design a doubly-stochastic weight matrix W that adheres to the topology of the underlying graph $\mathcal{G}$. In particular, we have that $w_{i,j} = 0$ whenever $(i,j) \notin \mathcal{E}$. We refer the reader to [31]–[33] for designing appropriate weight matrices in a distributed manner. Next, assuming that we are in the t_m^{th} iteration of the distributed power method, define $z_i^{(0)} := M_i\widehat{q}_i^{(t_m-1)}$ and $Z^{(0)^\mathsf{T}} := \begin{bmatrix} z_1^{(0)} & z_2^{(0)} & \ldots & z_N^{(0)}\end{bmatrix}$. Further, let $\mathcal{N}_i := \{j : (i,j) \in \mathcal{E}\}$ be the neighborhood of site i. Consensus works by having each site carry out the following updates through communications with its neighbors:

$$z_i^{(t_c)} = \sum_{j\in\mathcal{N}_i} w_{i,j} z_j^{(t_c-1)}, \tag{8}$$

where t_c denotes an index for consensus iterations. Note that the dynamics of the overall system in this case evolve as

$$Z^{(t_c)} = W^{t_c} Z^{(0)}. \tag{9}$$

It then follows that $Z_{i,T}^{(t_c)} \xrightarrow{t_c} \mathbf{1}^\mathsf{T} Z^{(0)}/N$, where $Z_{i,T}^{(t_c)}$ denotes the i^{th} row of $Z^{(t_c)}$ and $\mathbf{1} \in \mathbb{R}^N$ denotes a (column) vector of

[1]We are dropping the iteration count t in the following to simplify notation.

[2]This can be accomplished, for example, through the use of random number generators initialized with the same seed.

all ones [31]. This in particular implies that each site achieves perfect consensus averaging as $t_c \to \infty$ and obtains

$$Z_{i,T}^{(\infty)^\mathsf{T}} = \frac{1}{N}\sum_{j=1}^{N} z_j^{(0)}. \tag{10}$$

In practice, we can not perform infinite consensus iterations within each iteration of the distributed power method. We therefore make use of the modification of the standard consensus averaging proposed in [27] and define $\widehat{v}_i^{(t_m)} := Z_{i,T}^{(t_c)^\mathsf{T}}/[W^{t_c}\mathrm{e}_1]_i$, where $\mathrm{e}_1 := \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^\mathsf{T}$ and $[\cdot]_i$ denotes the i^{th} entry of a vector. In particular, we will have an error $\epsilon_{i,c}$ within $\widehat{v}_i^{(t_m)}$ at each site for any finite number of consensus iterations performed, i.e.,

$$\widehat{v}_i^{(t_m)} := Z_{i,T}^{(t_c)^\mathsf{T}}/[W^{t_c}\mathrm{e}_1]_i = \sum_{j=1}^{N} M_j \widehat{q}_j^{(t_m-1)} + \epsilon_{i,c}. \tag{11}$$

After finishing consensus iterations, each site i in iteration t_m has vector $\widehat{v}_i^{(t_m)}$ that is normalized to get an estimate of the eigenvector of M at site. Finally, we carry out enough iterations of the distributed power method at each site that the error between successive estimates of the eigenvector falls below a prescribed threshold. We have now motivated and described all the pieces of the proposed algorithm and are ready to state the full distributed dictionary learning algorithm, termed *cloud K-SVD*, which is detailed in Algorithm 1.

Remark 1. *During each iteration of the cloud K-SVD algorithm, each site exchanges only an n-dimensional vector with its neighbor sites and the total number of such exchanges for each site is given by $KT_dT_mT_c$, where T_d denotes the number of dictionary updates, T_m denotes the number of power method iterations, and T_c denotes the number of consensus iterations. The most important thing to note here in this regard is that the amount of data exchanges among the distributed sites is independent of the number of data samples available at each site, making cloud K-SVD a particularly useful algorithm for big data problems.*

Remark 2. *A careful reading of Algorithm 1 reveals that normalization by $[W^{t_c}\mathrm{e}_1]_i$ in Step 16 is redundant due to the normalization in Step 18. We retain the current form of Step 16 however to facilitate the forthcoming convergence analysis.*

Algorithm 1: Cloud K-SVD

```
1:  Input: Local data, $Y_1,\dots,Y_N$, and weight matrix $W$
2:  Initialize: Set $t \leftarrow 0$ and $\widehat{D}_i^{(t)} \leftarrow D$, $i = 1,2,\dots,N$,
    for a randomly generated dictionary $D$
3:  while $stopping\ rule$ do
4:    $t \leftarrow t+1$
5:    (Sparse Coding) The $i^{th}$ site solves $\forall s = 1,\dots,S_i$,
        $x_{s,i} = \arg\min_{x_{s,i}} \|y_{s,i} - \widehat{D}_i^{(t)} x_{s,i}\|_2^2$ s.t. $\|x_{s,i}\|_0 \le T_0$.
6:    for $k = 1$ to $K$ (Dictionary Update) do
7:      $M_i \leftarrow E_{i,k,R} E_{i,k,R}^\mathsf{T}$
8:      Initialize Power Method: Set $t_m \leftarrow 0$ and
        $\widehat{q}_i^{(t_m)} \leftarrow q^{init}$, $i = 1,2,\dots,N$, for a randomly
        generated vector $q^{init}$ with $\|q^{init}\|_2 = 1$
9:      while $stopping\ rule$ do
10:       $t_m \leftarrow t_m + 1$
11:       Initialize Consensus: Set $t_c \leftarrow 0$ and
          $z_i^{(t_c)} \leftarrow M_i \widehat{q}_i^{(t_m-1)}$, $i = 1,2,\dots,N$
12:       while $stopping\ rule$ do
13:         $t_c \leftarrow t_c + 1$
14:         $z_i^{(t_c)} = \sum_{j\in\mathcal{N}_i} w_{i,j} z_j^{(t_c-1)}$
15:       end while
16:       $\widehat{v}_i^{(t_m)} \leftarrow z_i^{(t_c)}/[W^{t_c}\mathrm{e}_1]_i$
17:       $\widehat{r}_i^{(t_m)} \leftarrow \sqrt{\widehat{v}_i^{(t_m)^\mathsf{T}} \widehat{v}_i^{(t_m)}}$
18:       $\widehat{q}_i^{(t_m)} \leftarrow \widehat{v}_i^{(t_m)} [\widehat{r}_i^{(t_m)}]^{-1}$
19:     end while
20:     $\widehat{d}_{i,k}^{(t)} \leftarrow \widehat{q}_i^{(t_m)}$
21:     $x_{i,k,R} \leftarrow \widehat{d}_{i,k}^{(t)\mathsf{T}} E_{i,k,R}$
22:   end for
23: end while
24: Return $\widehat{D}_i^{(t)}$, $i = 1,2,\dots,N$
```

IV. Convergence Analysis

In this section, we are interested in understanding whether cloud K-SVD results in dictionaries $\{\widehat{D}_i\}$ that are close in some sense to the centralized solution, given that power method and consensus averaging cannot be performed for an infinite number of iterations in practice. A positive answer to this is dependent on the convergence behavior of individual steps involved in cloud K-SVD. There are three main steps in cloud K-SVD: sparse coding, dictionary update, and the distributed power method used to compute dominant singular vectors of truncated error matrices as part of the dictionary update. In this paper, we make a partial progress toward convergence analysis of cloud K-SVD by providing a convergence analysis of the distributed power method within cloud K-SVD. Extension of this initial analysis to convergence behavior of the entire algorithm will be focus of future work.

Convergence behavior of distributed power method proposed in Algorithm 1 is dependent on errors introduced due to two iterative procedures: power method and consensus averaging. The main analytical contribution of this paper is demonstrating that both these errors are well behaved as long as enough consensus iterations are performed within each power method iteration. The following theorem summarizes this analytical contribution.

Theorem 1. *Consider the symmetric matrix $M = \sum_{i=1}^{N} M_i$ with eigenvalues $|\lambda_1| > |\lambda_2| \ge \dots \ge |\lambda_n|$, and define $\alpha := \sum_{i=1}^{N} \|M_i\|_2$ and $\gamma = \sqrt{\sum_{i=1}^{N} \|M_i\|_F^2}$. Let q denote the principal eigenvector of M, $\widehat{q}_i$ denote an estimate of q obtained at site i using the distributed power method (Steps 8–19 in Algorithm 1), and τ_{mix} denote the mixing time of a Markov chain associated with the weight matrix W. Suppose that the distributed power method is performed for t iterations* *and assume that if a centralized power method were also initialized with $q_c^{(0)} = q^{init}$ then $\max_{t_m=1,\dots,t} \frac{1}{\|Mq_c^{(t_m)}\|_2} \leq \beta$. Then, as long as $|\langle q, q^{init}\rangle| > 0$, $\epsilon \in \left(0, (10\alpha^2\beta^2)^{-1/3t}\right)$, and consensus averaging within each iteration of the distributed power method (Steps 11–15 in Algorithm 1) is performed for $O\left(t \cdot \tau_{mix} \cdot \log\left(2\alpha\beta\epsilon^{-1}\right) + \tau_{mix}\log\left(\frac{\gamma\sqrt{N}}{\alpha}\right)\right)$ iterations, we have that*

$$\forall i, \quad \left\|qq^T - \widehat{q}_i\widehat{q}_i^T\right\|_2 \leq c\left|\frac{\lambda_2}{\lambda_1}\right|^t + 4\epsilon^{3t}, \qquad (12)$$

where c is a positive numerical constant.

Theorem 1 states that $\widehat{q}_i \xrightarrow{t} \pm q\ \forall i$ at an exponential rate. In particular, the first term in (12) is due to errors in the power method, while the second term in there is due to errors in consensus averaging. In order to prove this theorem, we need a lemma that states that if the errors between the estimates provided by the distributed power method and the estimate provided by the centralized power method are bounded at the start of an iteration then these errors remain bounded at the end of that iteration. In order to state and prove such a lemma, we need a result from the literature on consensus averaging that specifies approximation errors in *vector* consensus averaging as a function of the number of iterations. The following theorem is adapted from [27, Theorem 5] in this regard.

Theorem 2. *[27] Let $v_i^{(t_c)}$ be the $n \times 1$ vector held by site i after t_c consensus iterations, $v := \sum_{j=1}^N v_j^{(0)}$ be the desired vector, and $z := \sum_{j=1}^N |v_j^{(0)}|$ be a vector whose entries are the sum of absolute values of the initial vectors $v_i^{(0)}$. Then, for any $\delta > 0$, we have after $t_c = O(\tau_{mix}\log\delta^{-1})$ consensus iterations that $\left\|\frac{v_i^{(t_c)}}{[W^{t_c}e_1]_i} - v\right\|_2 \leq \delta\|z\|_2\ \forall i$.*

Lemma 1. *Let q denote the output of centralized power method and $\widehat{q}_i$ denote the output of the distributed power method at i^{th} site after t_m iterations. Similarly, let q' and $\widehat{q}'_i$ denote the outputs of centralized and distributed power methods after $(t_m+1) \leq t$ iterations, respectively. Fix an $\epsilon \in (0,1)$, define $\delta := \frac{\alpha}{\gamma\sqrt{N}}\left(\frac{\epsilon}{2\alpha\beta}\right)^{3t}$, and assume that $\forall i$, $\|q - \widehat{q}_i\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha} \leq \frac{1}{2\alpha\beta^2(2\alpha+\delta\gamma\sqrt{N})}$. Then, assuming $O(\tau_{mix}\log\delta^{-1})$ consensus iterations, we have that*

$$\forall i,\ \|q' - \widehat{q}'_i\|_2 \leq (2\alpha\beta)^3\left(\max_{i=1,\dots,N}\|q - \widehat{q}_i\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha}\right).$$

Proof: Fix an $i \in \{1,\dots,N\}$ and define $v := Mq$, $r := \|v\|_2$, $\widehat{v}_i := M\widehat{q}_i$, and $\widehat{r}_i := \|\widehat{v}_i\|_2$. Next, note that $q' - \widehat{q}'_i = v(r^{-1} - \widehat{r}_i^{-1}) + (v - \widehat{v}_i)\widehat{r}_i^{-1}$. Using the triangle inequality, we therefore obtain

$$\|q' - \widehat{q}'_i\|_2 \leq \|v\|_2|r^{-1} - \widehat{r}_i^{-1}| + \|v - \widehat{v}_i\|_2\widehat{r}_i^{-1}. \qquad (13)$$

We now need to bound $\|v\|_2$, $|r^{-1} - \widehat{r}_i^{-1}|$, $\|v - \widehat{v}_i\|_2$, and $\widehat{r}_i^{-1}$. To this end, we define $\widehat{v} := \sum_{i=1}^N M_i\widehat{q}_i$ and write $\widehat{v}_i = \widehat{v} + \epsilon_{i,c}$, where $\epsilon_{i,c}$ is the consensus error. It then follows that $v - \widehat{v}_i = \left[\sum_{i=1}^N M_i(q - \widehat{q}_i)\right] - \epsilon_{i,c}$. It can further be shown using Theorem 2 and some algebraic manipulations that $\|\epsilon_{i,c}\|_2 \leq \delta\gamma\sqrt{N}$. We therefore obtain

$$\|v - \widehat{v}_i\|_2 \leq \sum_{i=1}^N \|M_i\|_2\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}. \qquad (14)$$

Next, notice that $|r^{-1} - \widehat{r}_i^{-1}| = |r - \widehat{r}_i|r^{-1}\widehat{r}_i^{-1}$ and further it is straightforward to show that $|r - \widehat{r}_i| \leq r^{-1}|\widehat{r}_i^2 - r^2|$. Now,

$$\begin{aligned}|\widehat{r}_i^2 - r^2| &= |\widehat{v}_i^{\mathsf{T}}\widehat{v}_i - v^{\mathsf{T}}v| = |\widehat{v}_i^{\mathsf{T}}\widehat{v}_i - v^{\mathsf{T}}v - \widehat{v}_i^{\mathsf{T}}v + \widehat{v}_i^{\mathsf{T}}v| \\ &\leq \|\widehat{v}_i - v\|_2(\|\widehat{v}_i\|_2 + \|v\|_2). \qquad (15)\end{aligned}$$

Since $\widehat{v}_i = \widehat{v} + \epsilon_{i,c}$, it can be shown that $\|\widehat{v}_i\|_2 \leq \alpha + \delta\gamma\sqrt{N}$. In addition, it can also be shown that $\|v\|_2 \leq \alpha$. It then follows from (14) and the above discussion that

$$\begin{aligned}|\widehat{r}_i^2 - r^2| &\leq (2\alpha + \delta\gamma\sqrt{N})\left(\sum_{i=1}^N \|M_i\|_2\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}\right) \\ &\leq (2\alpha + \delta\gamma\sqrt{N})\left(\alpha\max_i\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}\right). \quad (16)\end{aligned}$$

We can now use this inequality to upperbound $|r^{-1} - \widehat{r}_i^{-1}|$ as

$$\begin{aligned}&|r^{-1} - \widehat{r}_i^{-1}| \\ &\leq \widehat{r}_i^{-1}\beta^2(2\alpha + \delta\gamma\sqrt{N})(\alpha\max_i\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}). \quad (17)\end{aligned}$$

The only remaining quantity we need to bound is $\widehat{r}_i^{-1}$. To this end, notice that $|r - \widehat{r}_i| \geq (r^{-1})^{-1} - (\widehat{r}_i^{-1})^{-1}$. Since $|r - \widehat{r}_i| \leq r^{-1}|\widehat{r}_i^2 - r^2|$, we obtain from (15) and (16) that

$$\begin{aligned}&(r^{-1})^{-1} - (\widehat{r}_i^{-1})^{-1} \\ &\leq \alpha r^{-1}(2\alpha + \delta\gamma\sqrt{N})\left(\max_i\|q - \widehat{q}_i\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha}\right). \quad (18)\end{aligned}$$

It then follows from the lemma's assumption along with some algebraic manipulations that $\widehat{r}_i^{-1} \leq 2\beta$. Finally, plugging the bounds on $\widehat{r}_i^{-1}$, $|r^{-1} - \widehat{r}_i^{-1}|$, $\|v\|_2$, and $\|v - \widehat{v}_i\|_2$ in (13), we obtain

$$\begin{aligned}&\|q' - \widehat{q}'_i\|_2 \\ &\quad\leq 2\alpha\beta^3\left(\alpha\max_i\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}\right)(2\alpha + \delta\gamma\sqrt{N}) \\ &\qquad + 2\beta\left(\alpha\max_i\|q - \widehat{q}_i\|_2 + \delta\gamma\sqrt{N}\right) \\ &\quad= \left(4\alpha^3\beta^3 + 2\alpha^3\beta^3\frac{\delta\gamma\sqrt{N}}{\alpha} + 2\alpha\beta\right) \times \\ &\qquad\left(\max_i\|q - \widehat{q}_i\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha}\right). \quad (19)\end{aligned}$$

Finally, $\frac{\delta\gamma\sqrt{N}}{\alpha} \leq \left(\frac{\epsilon}{2}\right)^{3t} < 1$ since (i) $\delta = \frac{\alpha}{\gamma\sqrt{N}}\left(\frac{\epsilon}{2\alpha\beta}\right)^{3t}$, (ii) $\epsilon < 1$, and (iii) $\alpha r^{-1} \geq 1$, which implies $\alpha\beta \geq 1$. Plugging this into the above expression and noting that $\alpha\beta \leq \alpha^3\beta^3$, we obtain the claimed result. ■

Lemma 1 provides an understanding of the error accumulation in the distributed power method. While the factor of $(2\alpha\beta)^3$ in the lemma might seem discouraging, the fact that the distributed power method starts with a zero error keeps the total error in control. We now formally argue this in the proof of Theorem 1 below.

Proof of Theorem 1: Fix an $i \in \{1, \dots, N\}$, define q_{c} as the estimate of q obtained using the centralized power method that is initialized with the same q^{init} as the distributed power method, and notice that

$$\left\| qq^{\mathsf{T}} - \widehat{q}_i \widehat{q}_i^{\mathsf{T}} \right\|_2 \leq \|qq^{\mathsf{T}} - q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}}\|_2 + \|q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}} - \widehat{q}_i \widehat{q}_i^{\mathsf{T}}\|_2. \quad (20)$$

The convergence rate of the centralized power method is well studied and can be expressed as [17]

$$\|qq^{\mathsf{T}} - q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}}\|_2 \leq c \left| \frac{\lambda_2}{\lambda_1} \right|^t .$$

In order to bound $\|q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}} - \widehat{q}_i \widehat{q}_i^{\mathsf{T}}\|_2$, we make use of Lemma 1. In order to invoke this lemma, we first need to show that the assumption of the lemma holds for all iterations $t_m \leq (t-1)$. We start with $t_m = 0$ for this purpose and note that $q_{\mathsf{c}}^{(0)} = \widehat{q}_i^{(0)} = q^{init}$, which implies

$$\|q_{\mathsf{c}}^{(0)} - \widehat{q}_i^{(0)}\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha} \leq \left(\frac{\epsilon}{2}\right)^{3t}. \quad (21)$$

Further, under the assumptions of the theorem, it can be shown through elementary algebra that $\left(\frac{\epsilon}{2}\right)^{3t} \leq \frac{1}{2\alpha\beta^2(2\alpha+\delta\gamma\sqrt{N})}$. We now invoke mathematical induction and claim that the assumption in the lemma is satisfied for all $t_m \leq m$. Then we obtain from a recursive application of the statement of the lemma that for $t_m = (m+1)$, we have

$$\begin{aligned} &\|q_{\mathsf{c}}^{(m+1)} - \widehat{q}_i^{(m+1)}\|_2 + \frac{\delta\gamma\sqrt{N}}{\alpha} \\ &\quad \leq \frac{\delta\gamma\sqrt{N}}{\alpha} \sum_{i=0}^{m} (2\alpha\beta)^{3i} \overset{(a)}{\leq} 2 \cdot \frac{\delta\gamma\sqrt{N}}{\alpha} (2\alpha\beta)^{3m} \\ &\quad = 2 \cdot \epsilon^{3t} \frac{(2\alpha\beta)^{3m}}{(2\alpha\beta)^{3t}} \overset{(b)}{\leq} \frac{1}{2\alpha\beta^2(2\alpha + \delta\gamma\sqrt{N})}, \end{aligned} \quad (22)$$

where (a) follows from the geometric sum and the fact that $(2\alpha\beta)^3 > 2$, while (b) follows from the theorem assumptions and the fact that $m < t$.

We have now proved that the assumption of Lemma 1 holds for all $t_m \leq (t-1)$. In order to compute $\|q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}} - \widehat{q}_i \widehat{q}_i^{\mathsf{T}}\|_2$, therefore, we can recursively apply the result of this lemma up to the t^{th} iteration to obtain

$$\|q_{\mathsf{c}} - \widehat{q}_i\|_2 \leq \frac{\delta\gamma\sqrt{N}}{\alpha} \sum_{i=0}^{t} (2\alpha\beta)^{3i} \overset{(c)}{\leq} 2\epsilon^{3t}, \quad (23)$$

where (c) follows from the same arguments as in (22). The proof of the theorem now follows by noting the fact that $\|q_{\mathsf{c}} q_{\mathsf{c}}^{\mathsf{T}} - \widehat{q}_i \widehat{q}_i^{\mathsf{T}}\|_2 \leq (\|q_{\mathsf{c}}\|_2 + \|\widehat{q}_i\|_2)\|q_{\mathsf{c}} - \widehat{q}_i\|_2 \leq 4\epsilon^{3t}$. ∎

V. Numerical Experiments

In this section, we carry out numerical experiments using synthetic data to demonstrate the effectiveness of the cloud K-SVD algorithm. The numerical experiments correspond to a total of 100 sites, with each site having 500 samples in $\mathbb{R}^{20}$ within its local data and interconnections between the sites modeled through an Erdős–Rényi random graph with parameter $p = 0.5$. In order to generate data at sites, we first generate a dictionary $D \in \mathbb{R}^{20\times 50}$ with columns uniformly distributed on the unit sphere in $\mathbb{R}^{20}$, then randomly select a 45-column subdictionary of D for each site, and finally generate samples for that site using a linear combination of $T_0 = 3$ randomly selected atoms of the subdictionary. The data samples are normalized to have unit ℓ_2 norms and finally white Gaussian noise with variance $\sigma^2 = 0.01$ is added to the data samples. In order to carry out distributed consensus averaging, we generate a weight matrix W according to the local-degree weights method described in [31, Sec. 4.2].

The first set of experiments that we perform demonstrate 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). In Fig. 2, we plot representation errors, defined as $\frac{1}{nS}\sum_{i=1}^{N}\sum_{j=1}^{S_i}\|y_{i,j} - Dx_{i,j}\|_2$, of three dictionary learning methods, namely, centralized (canonical) K-SVD, cloud K-SVD, and local K-SVD, as a function of the number of dictionary learning iterations. It can be seen from this figure, which corresponds to an average of 100 Monte-Carlo trials, that cloud K-SVD performs much better than local K-SVD and that its performance comes very close to the centralized K-SVD algorithm. In particular, the error for local K-SVD is ~ 0.06 after 50 iterations, while it is ~ 0.03 for cloud K-SVD, which is within 0.01 of the error of centralized K-SVD.

The second set of experiments that we perform illustrate the convergence behavior of the distributed power method within cloud K-SVD (Steps 8–19 in Algorithm 1) as a function of the number of consensus iterations. The results of these experiments, which are reported in Fig. 3, correspond to five different values of consensus iterations (3, 4, 5, 10, 15) within each iteration of the distributed power method. Specifically, denote by $q^{(t_m)}$ the principal eigenvector of the matrix M in Algorithm 1 computed using Matlab and denote by $\widehat{q}_1^{(t_m)}$ an estimate of $q^{(t_m)}$ obtained at site 1 using the distributed power method of Algorithm 1 in iteration t_m. Then Fig. 3 plots $\|q^{(t_m)}q^{(t_m)\mathsf{T}} - \widehat{q}_1^{(t_m)}\widehat{q}_1^{(t_m)\mathsf{T}}\|_2$ averaged over all distributed power method iterations, dictionary learning iterations, and 100 Monte-Carlo trials as a function of the number of distributed power method iterations. It can be seen from this figure that the distributed power method hits an *error floor* with increasing distributed power method iterations, where the floor is fundamentally determined by the number of consensus iterations, as predicted by Theorem 1.

VI. Conclusion

In this paper, we have proposed a new dictionary learning algorithm, termed cloud K-SVD, that facilitates collaborative learning of a dictionary that best approximates massive data distributed across geographical regions. The efficacy of the proposed algorithm is demonstrated through extensive simulations, while a partial analysis of the convergency behavior of the proposed algorithm is also carried out. In the future, we

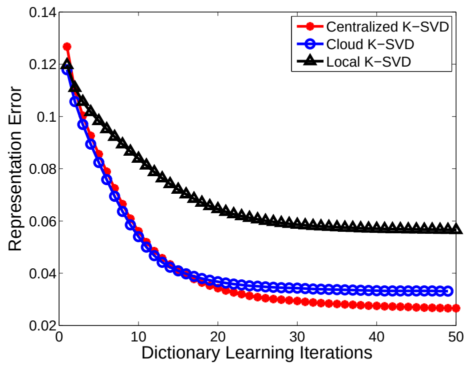

Fig. 2. Representation errors as a function of the number of dictionary learning iterations for centralized K-SVD, cloud K-SVD, and local K-SVD.

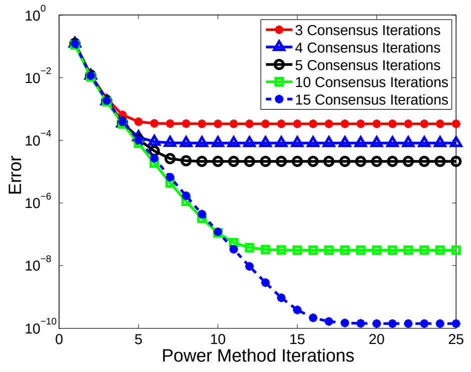

Fig. 3. Convergence behavior of the distributed power method as a function of varying number of consensus averaging iterations.

will analyze the convergence behavior of the entire algorithm and will apply it to real-world, big-data problems.

Acknowledgement

The authors would like to thank Tong Wu for his careful reading of the manuscript and many helpful suggestions.

References

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Edu. Psych.*, vol. 6, no. 24, pp. 417–441, Sep. 1933.

[2] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 27, no. 12, pp. 1945–1959, Dec. 2005.

[3] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," *Intl. J. Computer Vision*, vol. 100, no. 3, pp. 217–240, Dec. 2012.

[4] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, Feb. 2003.

[5] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[6] M. Gastpar, P.-L. Dragotti, and M. Vetterli, "The distributed Karhunen–Loève transform," *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 5177–5196, Dec. 2006.

[7] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. IEEE 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 1722–1726.

[8] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *Proc. IEEE Eleventh International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2010)*, 2010, pp. 1–5.

[9] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 725–738, 2011.

[10] R. Tron and R. Vidal, "Distributed computer vision algorithms through distributed averaging," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, 2011, pp. 57–63.

[11] S. Yoon and V. Pavlovic, "Distributed Probabilistic Learning for Camera Networks with Missing Data," in *Neural Information Processing Systems (NIPS)*, 2012.

[12] Y. Lu and M. Do, "A theory for sampling signals from a union of subspaces," *IEEE Trans. Signal Processing*, vol. 56, no. 6, pp. 2334–2345, Jun. 2008.

[13] D. L. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 18, no. 8, pp. 831–836, Aug. 1996.

[14] 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, 2006.

[15] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. ACM of the 26th Annual International Conference on Machine Learning*, 2009, pp. 689–696.

[17] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 2012.

[18] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[19] J. Speyer, "Computation and transmission requirements for a decentralized linear-quadratic-Gaussian control problem," *IEEE Trans. Automat. Control*, vol. 24, no. 2, pp. 266–269, Apr. 1979.

[20] J. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Trans. Autom. Control*, vol. 29, no. 1, pp. 42–50, Jan. 1984.

[21] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *The Journal of Machine Learning Research*, vol. 99, pp. 1663–1707, 2010.

[22] E. Kokiopoulou and P. Frossard, "Distributed classification of multiple observation sets by consensus," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 104–114, 2011.

[23] S. Lee and A. Nedić, "Distributed random projection algorithm for convex optimization," *IEEE J. Select. Topics Signal Processing*, vol. 7, no. 2, pp. 221–229, Apr. 2013.

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

[25] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. Signal Processing*, vol. 60, no. 12, pp. 6217–6234, Dec. 2012.

[26] M. E. Yildiz, F. Ciaramello, and A. Scaglione, "Distributed distance estimation for manifold learning and dimensionality reduction," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP 2009)*. IEEE, 2009, pp. 3353–3356.

[27] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 70–83, 2008.

[28] K. Engan, S. O. Aase, and J. H. Husøy, "Multi-frame compression: Theory and design," *Signal Processing*, vol. 80, no. 10, pp. 2121–2140, 2000.

[29] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[30] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[31] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[32] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[33] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, 2004.