Efficient and Distributed Generalized Canonical Correlation Analysis for Big Multiview Data

Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, Hyun Ah Song,
Partha Talukdar, Nicholas D. Sidiropoulos, Christos Faloutsos, and Tom Mitchell

Abstract— Generalized canonical correlation analysis (GCCA) integrates information from data samples that are acquired at multiple feature spaces (or 'views') to produce low-dimensional representations – which is an extension of classical two-view CCA. Since the 1960s, (G)CCA has attracted much attention in statistics, machine learning, and data mining because of its importance in data analytics. Despite these efforts, the existing GCCA algorithms have serious complexity issues. The memory and computational complexities of the existing algorithms usually grow as a quadratic and cubic function of the problem dimension (the number of samples / features), respectively – e.g., handling views with $\approx 1,000$ features using such algorithms already occupies $\approx 10^6$ memory and the per-iteration complexity is $\approx 10^9$ flops – which makes it hard to push these methods much further. To circumvent such difficulties, we first propose a GCCA algorithm whose memory and computational costs scale *linearly* in the problem dimension and the number of nonzero data elements, respectively. Consequently, the proposed algorithm can easily handle very large sparse views whose sample and feature dimensions both exceed $\approx 100,000$. Our second contribution lies in proposing two distributed algorithms for GCCA, which compute the canonical components of different views in parallel and thus can further reduce the runtime significantly if multiple computing agents are available. We provide detailed convergence analyses of the proposed algorithms and show that all the large-scale GCCA algorithms converge to a Karush-Kuhn-Tucker (KKT) point at least sublinearly. Judiciously designed synthetic and real-data experiments are employed to showcase the effectiveness of the proposed algorithms.

1 Introduction

Canonical Correlation Analysis (CCA) [1] has been an essential tool in data analytics and machine learning, which is used to extract low-dimensional representations from two views of the same set of entities in different "feature spaces" – e.g., areas of technical expertise and social network data for a set of authors. A view is a high-dimensional representation of an entity in a certain feature domain – e.g., the English word "car" and its translation in Spanish "coche" are two views of the same entity in different domains, and both car and coche can be represented as high-dimensional feature vectors, e.g., using co-occurrences with other English and Spanish terms, respectively. The applications of multiview analysis span a wide spectrum, including clustering [2], [3], regression [4], brain imaging [5], natural language processing [6], [7], and many other topics; also see [8], [9], [10], [11]. Classical CCA focuses on the two-view case. Generalized Canonical Correlation Analysis (GCCA) that aims at handling more than two views has also drawn much attention since in many applications GCCA arises naturally [12], [13], [14], [15], [16], [5], [17].

X. Fu is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA. email: xiao.fu@oregonstate.edu.

K. Huang is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, 55455 USA email: huang663@umn.edu.

N. D. Sidiropoulos is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA email: nikos@virginia.edu.

H. Song, C. Faloutsos and Tom Mitchell are with the Machine Learning Department at Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA. e-mail (hyunahs,christos+,tom.mitchell)@cs.cum.edu

E. E. Papalexakis is with the Computer Science & Engineering Department at the University of California Riverside, 900 University Ave, Riverside, CA 92521, USA. email epapalex@cs.ucr.edu

P. Partha is with Department of Computational and Data Sciences and Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India 560012, email ppt@cds.iisc.ac.in

1.1 Related Work and Challenges

Computationally, GCCA poses very challenging optimization problems. Starting from the 1960s, different formulations and algorithms of GCCA have been considered [12], [18], [19], [17], such as the sum-of-correlations (SUMCOR), sum-of-squared-correlations (SSQCOR) and maximal-variation (MAX-VAR) formulations, to name a few. In recent years, along with the increasing ability of collecting more and more multiview data and the demand for integrating and analyzing such data, the study of GCCA algorithms has been gaining renewed interest [13], [14], [20], [21], [15]. Horst proposed a power method-like algorithm to deal with the SUMCOR problem [19]. This method was further studied by Zhang *et al.* in [21], where some special cases were investigated and optimality conditions of the Horst algorithm in those cases were presented. Tenenhaus *et al.* proposed a block coordinate descent (BCD)-based algorithm combined with local linearization to deal with different formulations of GCCA [13], and later extended the framework to sparse GCCA [14] – which seeks sparse canonical components. Rupnik *et al.* proved that the SUMCOR problem is NP-hard, and proposed using semidefinite relaxation to approximate it [15]. Recently, Rastogi *et al.* [7] and Fu *et al.* [22] proposed efficient algorithms to handle the MAX-VAR problem.

Despite the long history of GCCA research, some old challenges remain, and new one arise in this era of big data. First, the aforementioned algorithms mostly consider

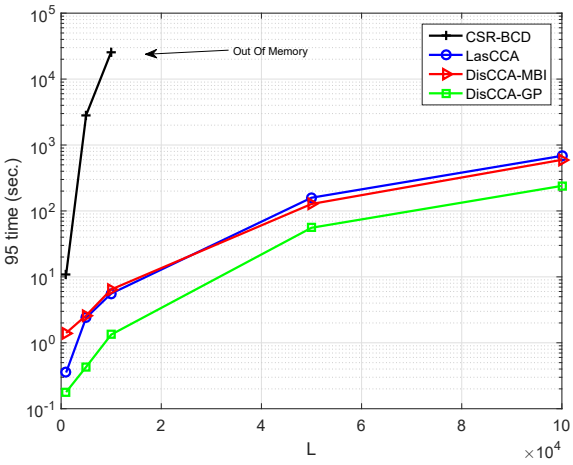


Fig. 1. `95time` (runtime for capturing 95% of the total sum of pairwise correlations among the views) versus size of the views. 5 components are sought; 5 views; each view is a $L \times 0.8L$ matrix; data density $\rho = 5 \times 10^{-3}$. **LasCCA**, **DisCCA-MBI**, and **DisCCA-GP** are the proposed algorithms; CSR-BCD is a competitor which uses whitening.

extracting the first canonical component of each view, and then use a deflation method to find the other ones (i.e., subtracting the previously found component from the data and repeating). Deflation is known to suffer from error propagation and can easily destroy sparsity, which is usually relied upon to process large-scale data. Second, many (G)CCA algorithms employ a whitening process (i.e., multiplying the square roots of the correlation matrices of the views to the data; cf. Eqs. (2)–(3)). This destroys sparsity, can easily exhaust memory resources, and also requires a large number of flops when the size of the views is even moderate (e.g., when both dimensions of the view matrices are $\sim 10,000$). The whitening process thus poses serious scalability problems in terms of both memory and computation. There has recently been interest in designing scalable algorithms for CCA in the machine learning community [23], [24], [25], [26], [27], but mostly focusing on the two-view case. One possible workaround is the algorithm in [7], which uses rank-truncated data to circumvent the scalability issues and can extract multiple canonical components simultaneously – albeit the truncation pre-processing also filters out potentially useful information, which degrades performance. The method in [22] makes use of data sparsity to handle large-scale MAX-VAR, but heavily relies on the special formulation of MAX-VAR GCCA, which is not easy to be generalized to other formulations of GCCA, e.g., SUMCOR. Third, to the best of our knowledge, no distributed algorithm for GCCA has appeared in the literature. Designing distributed algorithms for GCCA is well-motivated by a number of reasons, including privacy, confidentiality, and data security/integrity concerns, in addition to better scaling, faster computation, and suitability for cloud computing implementation.

1.2 Contributions

In this work, we consider designing scalable and distributed algorithms for GCCA. Specifically, our interest lies in the SUMCOR formulation where pairwise highly correlated reduced-dimension views are sought. Our design is under a setting where the views are large sparse matrices. Such a scenario is well-motivated in practice. For example, when applying GCCA to find common latent structure of multilingual views, the views are word-word co-occurrence matrices of different languages which are very large and very sparse. Our contributions are as follows:

• **Scalable GCCA**: We propose a SUMCOR GCCA algorithm that scales very gracefully with the problem dimension (i.e., the number of samples/features) when the data is sparse. We carefully reformulate the SUMCOR problem to an equivalent optimization problem. Working with this reformulation, the proposed algorithm does not need to explicitly instantiate the correlation matrices and whitening matrices, resulting in substantial memory savings. Computationally, the key components of the algorithms are sparse matrix-vector multiplications and singular value decomposition of (typically very) "thin" matrices, which are both lightweight and suitable for big data analytics. A sneak peek of the experimental results is provided in Fig. 1, where one can verify that our method (**LasCCA**) scales seamlessly to bigger views, while the competitor using whitening runs out of memory when the size of the views gets large. Besides scalability, the proposed algorithm can simultaneously extract any number of canonical components, which is preferred over deflation-based approaches.

• **Distributed GCCA**: Based on the same reformulation, we also propose two distributed GCCA algorithms for SUMCOR. To the best of our knowledge, these are the first distributed algorithms for SUMCOR GCCA. In the distributed framework, the views are stored in different nodes of a network, and only a small amount of information needs to be exchanged between the nodes and a coordinator. Most of the computations are carried out locally and in parallel at the nodes. We propose two different distributed algorithms under the same network settings, namely, `DisCCA-MBI` and `DisCCA-GP`. `DisCCA-MBI` requires less communication overhead but more computational resources, while `DisCCA-GP` uses twice as much overhead as that of `DisCCA-MBI` but utilizes the computational resources in a more efficient way. By exploring more nodes/cores, running time can be further reduced by the distributed algorithms by at least 30% compared to that of the proposed scalable algorithm in our experiments.

• **Analysis and Validation**: The convergence properties of the proposed algorithms are investigated. Specifically, we show that both the centralized and distributed algorithms produce solution sequences that converge to a Karush-Kuhn-Tucker (KKT) point of the SUMCOR problem. We also show that the algorithms shrink the gap from an initial solution to a KKT point (measured by a potential function) at a sublinear rate – i.e., the gap shrinks to ϵ after $\mathcal{O}(1/\epsilon)$ iterations. As for validation, we first employ judiciously designed synthetic-data experiments to showcase the effectiveness of the proposed algorithms. We then apply the algorithms to multilingual document data, namely, the RCV2 corpus, and a four-language word-word co-occurrence dataset to evaluate the performance.

Reproducibility: We open-source our code at the authors' websites, where a `MATLAB` demo is available.

A preliminary version of this work was presented at

IEEE International Conference on Data Mining (ICDM 2016) [28]. In this journal version, the original algorithms in [28] are modified to provide convergence guarantees; a new distributed algorithm that has significantly better runtime performance is proposed; detailed and comprehensive convergence analyses of the algorithms are presented; practical issues such as judicious centering of the data to avoid memory explosion and diagonal loading to circumvent the ill-conditioning problem of correlation matrices are taken into consideration, and more comprehensive synthetic and real-data experiments are provided.

Notation: $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ denote a real-valued n-dimensional vector and a real-valued $m \times n$ matrix, respectively. The Frobenious norm and the matrix 2-norm are denoted by $\|\boldsymbol{X}\|_F$ and $\|\boldsymbol{X}\|_2$, respectively. $\mathrm{Tr}(\boldsymbol{X})$ denotes the trace of the square matrix $\boldsymbol{X}$. $\mathrm{nnz}(\boldsymbol{X})$ counts the number of non-zero elements in $\boldsymbol{X}$. $\boldsymbol{I}_n$ denotes an identity matrix of size $n \times n$. $\mathcal{R}(\boldsymbol{X})$ denotes the range space of $\boldsymbol{X}$.

2 Problem Statement

Let $\boldsymbol{Y}_i \in \mathbb{R}^{L\times M_i}$ denote the ith (centered) view, where $\boldsymbol{Y}_i(\ell,:)$ corresponds to the ℓth data point (or entity) in the ith view, L is the number of entities, and M_i denotes the number of features of the ith view. The SUMCOR GCCA problem can be expressed as below [29]:

$$\max_{\{\boldsymbol{Q}_i\in\mathbb{R}^{M_i\times K}\}_{i=1}^I} \sum_{i=1}^I\sum_{j\neq i}^I \mathrm{Tr}(\boldsymbol{Q}_i^T\boldsymbol{X}_i^T\boldsymbol{X}_j\boldsymbol{Q}_j) \tag{1a}$$

$$\text{s.t. } \boldsymbol{Q}_i^T\left(\boldsymbol{X}_i^T\boldsymbol{X}_i\right)\boldsymbol{Q}_i = \boldsymbol{I}_K,\ \forall i; \tag{1b}$$

where $\boldsymbol{X}_i = (1/\sqrt{L})\boldsymbol{Y}_i \in \mathbb{R}^{L\times M_i}$ denotes the ith normalized view, $\boldsymbol{Q}_i \in \mathbb{R}^{M_i\times K}$, and K is the number of canonical components that we seek. By this normalization, $(\boldsymbol{X}_i^T\boldsymbol{X}_i)$ is an estimate of the auto-correlation matrix of the ith view and $(\boldsymbol{X}_i^T\boldsymbol{X}_j)$ an estimate of the cross-correlation matrix between the ith and the jth views; i.e., we seek K-dimensional representations of the views such that they are pair-wisely highly correlated, and we aim to extract K canonical components *simultaneously*. Our approach starts with the idea of block coordinate descent (BCD) – i.e., separately updating $\boldsymbol{Q}_i$ for $i = 1,\ldots,I$ when fixing other $\boldsymbol{Q}_j$'s for $j \neq i$. By doing so, the subproblem w.r.t. $\boldsymbol{Q}_i$ can be expressed as

$$\max_{\boldsymbol{Q}_i}\ \mathrm{Tr}\left(\boldsymbol{Q}_i^T\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j\boldsymbol{Q}_j\right) \tag{2a}$$

$$\text{s.t. } \boldsymbol{Q}_i^T\left(\boldsymbol{X}_i^T\boldsymbol{X}_i\right)\boldsymbol{Q}_i = \boldsymbol{I}_K. \tag{2b}$$

To solve the subproblem (2), it is tempting to do the following whitening-based reformulation: Assume that $\boldsymbol{X}_i^T\boldsymbol{X}_i$ has full rank (which is reasonable since views of real data usually contain noise) and thus admits a square root decomposition, i.e., $\boldsymbol{X}_i^T\boldsymbol{X}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{1/2}(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{1/2}$. Then, let $\boldsymbol{Q}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}\boldsymbol{Z}_i$. We can re-write (2) as

$$\max_{\boldsymbol{Z}_i:\ \boldsymbol{Z}_i^T\boldsymbol{Z}_i=\boldsymbol{I}}\ \mathrm{Tr}\left(\boldsymbol{Z}_i^T(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j\boldsymbol{Q}_j\right). \tag{3}$$

In some existing approaches such as those in [13], [14], [15], the key steps are essentially identical to the above change of variables – although they consider the simplest case where $K = 1$.

The upshot of this reformulation is that Problem (3) is essentially a Procrustes projection problem [30], i.e.,

$$\begin{aligned}\min_{\boldsymbol{Z}_i}\ &\|\boldsymbol{Z}_i - \boldsymbol{T}_i\|_F^2\\ \text{s.t. }&\boldsymbol{Z}_i^T\boldsymbol{Z}_i = \boldsymbol{I},\end{aligned} \tag{4}$$

where $\boldsymbol{T}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j\boldsymbol{Q}_j$, since the second-order terms in the objective function of (4) do not affect the solution and thus (4) and (3) are equivalent. The Procrustes projection problem can be solved to optimality by applying singular value decomposition (SVD) to $\boldsymbol{T}_i$. Such a method works well when M_i is small. However, instantiating the whitening matrix $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}$ poses serious complexity issues: first, the matrix $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{1/2}$ is very likely to be dense even when $\boldsymbol{X}_i$ is sparse. Consequently, computing its inverse requires $\mathcal{O}(M_i^3)$ flops (e.g., when $M_i = 100,000$, this step requires $\mathcal{O}(10^{15})$ flops). More importantly, storing a dense and large $M_i\times M_i$ matrix is almost prohibitive – when $M_i = 100,000$, storing $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{1/2}$ needs 75 GB memory. In this work, we will propose algorithms that completely avoid instantiating $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}$, and thus are highly scalable in terms of both memory and computational complexity.

3 Large-Scale GCCA (LASCCA)

Instead of applying the change of variables $\boldsymbol{Q}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}\boldsymbol{Z}_i$ and reformulating (2) to (3), we let

$$\boldsymbol{G}_i = \boldsymbol{X}_i\boldsymbol{Q}_i,\quad \forall i.$$

Note that $\boldsymbol{G}_i \in \mathbb{R}^{L\times K}$ is a very "thin" matrix since in practice $L \gg K$ holds and K, i.e., the number of canonical components sought, is usually small and can be controlled by the designer.

We wish to change the variables of Problem (2) from $\boldsymbol{Q}_i$ to $\boldsymbol{G}_i$, so that we can get rid of the difficult constraints, i.e., $\boldsymbol{Q}_i^T\boldsymbol{X}_i^T\boldsymbol{X}_i\boldsymbol{Q}_i = \boldsymbol{I}_K$. To this end, we will make use of the following lemma:

Lemma 1 *Consider $\boldsymbol{G}_i = \boldsymbol{X}_i\boldsymbol{Q}_i$. When $\mathrm{rank}(\boldsymbol{X}_i) = M_i$, we have*

$$\boldsymbol{G}_i = \boldsymbol{X}_i\boldsymbol{Q}_i \Leftrightarrow \boldsymbol{Q}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\boldsymbol{G}_i,\ \boldsymbol{G}_i(:,k)\in\mathcal{R}(\boldsymbol{X}_i),\ \forall k.$$

Proof: The "$\Rightarrow$" part is trivial to prove. To show the "$\Leftarrow$" part, first notice that $\boldsymbol{G}_i(:,k)\in\mathcal{R}(\boldsymbol{X}_i)$ for all k implies that $\boldsymbol{G}_i = \boldsymbol{X}_i\boldsymbol{\Theta}_i$ holds for a certain $\boldsymbol{\Theta}_i \in \mathbb{R}^{K\times K}$. Then, we have

$$\boldsymbol{Q}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T(\boldsymbol{X}_i\boldsymbol{\Theta}_i).$$

The above results in $\boldsymbol{Q}_i = \boldsymbol{\Theta}_i$ when $\mathrm{rank}(\boldsymbol{X}_i) = M_i$. □

Using Lemma 1, Problem (2) can be written as

$$\max_{\boldsymbol{G}_i}\ \mathrm{Tr}\left(\boldsymbol{G}_i^T\boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j\right)$$

$$\text{s.t. } \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}_K, \tag{5a}$$

$$\boldsymbol{G}_i(:,k)\in\mathcal{R}(\boldsymbol{X}_i),\ \forall k. \tag{5b}$$

The key difference between our reformulation in (5) and that in (3) is that we avoided using the square root decomposition of the correlation matrices, i.e., $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}$. As we will verify later, such a change will significantly reduce memory cost and computational complexity when the views are large and sparse.

There are several difficulties for solving the subproblem in (5). First, let us denote

$$\boldsymbol{H}_i = \boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j.$$

Some components that constitute $\boldsymbol{H}_i$, e.g., $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}$, are difficult to compute and store because of the $(\cdot)^{-1}$ operator. Second, even if we have obtained $\boldsymbol{H}_i$, solving (5) seems nontrivial: At first glance, the orthogonality constraint in (5a) and the subspace constraint in (5b) together make the optimization problem w.r.t. $\boldsymbol{G}_i$ quite complicated. For now, let us assume that $\boldsymbol{H}_i$ can be somehow computed and first look at the problem of solving Problem (5). Here, we show that

Lemma 2 *The following $\boldsymbol{G}_i$ is a global optimal solution to Problem* (5)*:*

$$\boldsymbol{G}_i = \boldsymbol{U}_i\boldsymbol{V}_i^T,\ \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T = \mathrm{svd}(\boldsymbol{H}_i), \tag{6}$$

where $\mathrm{svd}(\boldsymbol{H}_i)$ *denotes the economy-size SVD of* $\boldsymbol{H}_i$*.*

Proof: First, let us consider Problem (5) without the constraint (5b). Then, the relaxed problem is equivalent to solving a Procrustes projection problem as follows

$$\min_{\boldsymbol{G}_i:\ \boldsymbol{G}_i^T\boldsymbol{G}_i=\boldsymbol{I}} \left\|\boldsymbol{G}_i - \boldsymbol{H}_i\right\|_F^2. \tag{7}$$

An optimal solution of the above is given in (6). It remains to show that the solution in (6) still stays in $\mathcal{R}(\boldsymbol{X}_i)$. Indeed, since $\boldsymbol{H}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i)$ by its definition, it follows that $\boldsymbol{U}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i)$ and thus the solution in (6) also resides in the range space of $\boldsymbol{X}_i$. Since the solution in (6) is an optimal solution of the constraint-relaxed version of Problem (5), it is also an optimal solution of Problem (5). □

Using Lemma 2, one natural thought is to update $\boldsymbol{G}_i$ using (6) when fixing $\boldsymbol{G}_j$ for $j \neq i$ and perform this cyclically w.r.t. each $\boldsymbol{G}_i$. This strategy falls into the category of the so-called Gauss-Seidel type block coordinate descent (BCD) [31]. The concern here is that, when dealing with non-convex constraints (e.g., $\boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}$), Gauss-Seidel BCD cannot guarantee convergence of the solution sequence in general [31]. To circumvent this situation, we propose to update $\boldsymbol{G}_i$ using the following simple modification:

$$\boldsymbol{G}_i^{(r+1)} = \boldsymbol{U}_i\boldsymbol{V}_i^T,\ \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T = \mathrm{svd}(\boldsymbol{H}_i + \sigma\boldsymbol{G}_i^{(r)}), \tag{8}$$

where $\boldsymbol{G}_i^{(r)}$ denotes $\boldsymbol{G}_i$ obtained at the rth iteration and $\sigma > 0$ is a real-valued positive scalar (and $\boldsymbol{G}^{(0)}$ denotes the initialization of the iterations which will be specified later). Note that adding the extra term $\sigma\boldsymbol{G}_i^{(r)}$ does not change the complexity of this step, since the SVD is still applied to a thin matrix. More interestingly, as will be seen later, this minor modification allows us to analyze convergence properties of the algorithm rigorously. As preparation for our forthcoming proof, we show that

Algorithm 1: `LasCCA`

input : $\{\boldsymbol{X}_i\}_{i=1}^I$; K
1 $\boldsymbol{Q}_i = \mathrm{randn}(M_i, K)$;
2 $\boldsymbol{U}_i\boldsymbol{\Sigma}\boldsymbol{V}_i^T = \mathrm{svd}(\boldsymbol{X}_i\boldsymbol{Q}_i, K, '\,\mathrm{econ}')$, $\boldsymbol{G}_i^{(0)} = \boldsymbol{U}_i$;
3 $r \leftarrow 1$;
4 **repeat**
5 **for** $i = 1 : I$ **do**
6 $\mathcal{G}_i^{(r)} = \{\boldsymbol{G}_1^{(r+1)}, \ldots, \boldsymbol{G}_{i-1}^{(r+1)}, \boldsymbol{G}_{i+1}^{(r)}, \ldots, \boldsymbol{G}_I^{(r)}\}$
7 $\boldsymbol{H}_i^{(r)} \leftarrow$ `H_Compute` $\left(\{\boldsymbol{X}_i\}_{i=1}^I, \mathcal{G}_i^{(r)}\right)$;
8 $\boldsymbol{U}_i^{(r)}\boldsymbol{\Sigma}_i^{(r)}\boldsymbol{V}_i^{(r)} \leftarrow \mathrm{svd}(\boldsymbol{H}_i^{(r)} + \sigma\boldsymbol{G}_i^{(r)})$; $\mathcal{O}(LK^2)$
9 $\boldsymbol{G}_i^{(r+1)} \leftarrow \boldsymbol{U}_i^{(r)}(\boldsymbol{V}_i^{(r)})^T$;
10 **end**
11 $r \leftarrow r + 1$;
12 **until** *some stopping criterion is reached*;
output: $\{\boldsymbol{G}_i\}$

Corollary 1 *The update of $\boldsymbol{G}_i^{(r+1)}$ in* (8) *is an optimal solution to*

$$\begin{aligned}\min_{\boldsymbol{G}_i}\ & -\mathrm{Tr}(\boldsymbol{G}_i^T\boldsymbol{H}_i) + \frac{\sigma}{2}\left\|\boldsymbol{G}_i^{(r)} - \boldsymbol{G}_i\right\|_F^2 \\ \text{s.t. }& \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}, \quad \boldsymbol{G}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i),\ \forall k,\end{aligned} \tag{9}$$

given that $\boldsymbol{G}_i^{(r)}(:,k) \in \mathcal{R}(\boldsymbol{X}_i)$ for all k holds.

Proof: By expanding the objective and discarding the constants, the optimization problem becomes

$$\begin{aligned}\min_{\boldsymbol{G}_i}\ & -\mathrm{Tr}\left(\left(\boldsymbol{H}_i + \sigma\boldsymbol{G}_i^{(r)}\right)^T\boldsymbol{G}_i\right) \\ \text{s.t. }& \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}, \quad \boldsymbol{G}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i),\ \forall k,\end{aligned} \tag{10}$$

Applying the result of Lemma 2, one can see that the solution in (8) is an optimal solution to the above if $\boldsymbol{H}_i + \boldsymbol{G}_i^{(r)} \in \mathcal{R}(\boldsymbol{X}_i)$. Using the assumption that $\boldsymbol{G}_i^{(r)}(:,k) \in \mathcal{R}(\boldsymbol{X}_i)$, we complete the proof. □

We should mention that the assumption $\boldsymbol{G}_i^{(r)}(:,k) \in \mathcal{R}(\boldsymbol{X}_i)$ for all k is not hard to fulfill in practice: By initializing $\boldsymbol{G}_i^{(0)} \in \mathcal{R}(\boldsymbol{X}_i)$ (e.g., using the truncated SVD which has linear complexity in $\mathrm{nnz}(\boldsymbol{X}_i)$), this assumption can always be satisfied in the subsequent iterations.

The overall BCD algorithm is presented in Algorithm 1, which we name *large-scale generalized CCA* (`LasCCA`). One can see that the SVD step in (8) is not difficult in terms of both computational and memory costs. In fact, $\boldsymbol{H}_i \in \mathbb{R}^{L\times K}$ is a very "thin" matrix (recall that K is usually much smaller than L) – meaning that the memory burden is very light. In addition, computing SVD of $\boldsymbol{H}_i$ costs $\mathcal{O}(LK^2)$ flops, which is merely linear in L, i.e., the number of data samples. At this point, we have not elaborated how to compute $\boldsymbol{H}_i$ but only put an operator called `H_Compute`$(\cdot,\cdot)$ there (cf. line 2 in Algorithm 1). In the next subsection, we will focus on computing $\boldsymbol{H}_i$.

3.1 Building Block: Iterative Least Squares

Let us now turn attention to the problem of computing $\boldsymbol{H}_i$. As we have mentioned, computing and storing $(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}$ may not be affordable when constructing $\boldsymbol{H}_i$ at each step. Fortunately, explicit computation of this term is not necessary when calculating $\boldsymbol{H}_i$. In fact, what has been used for constructing $\boldsymbol{H}_i$ is $(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ for different j's,

which is, again, a very thin matrix that only costs $\mathcal{O}(LK)$ memory. To obtain this thin matrix, consider the following unconstrained linear least squares (LS) problem

$$\min_{\boldsymbol{R}_j} \ \|\boldsymbol{X}_j\boldsymbol{R}_j - \boldsymbol{G}_j\|_F^2 . \tag{11}$$

Assume that $\mathrm{rank}(\boldsymbol{X}_j) = M_j$ and $L \geq M_j$. It is well-known that the optimal solution $\boldsymbol{R}_j^\star$ can be written in the following form:

$$\boldsymbol{R}_j^\star = (\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j. \tag{12}$$

Eq. (12) means that to obtain the key ingredient of building up $\boldsymbol{H}_i$, what one needs to do is to solve an associated LS problem. The merit of looking at the problem of finding $(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ from a LS point of view is that the LS problem, although admitting a closed-form solution as in (12), can also be solved iteratively using very lightweight iterations that can also take advantage of sparsity. When the problem size is small, this line of thinking is unnecessary and is more cumbersome compared to directly inverting $(\boldsymbol{X}_j^T\boldsymbol{X}_j)$. However, when L and M_j are large, inverting $(\boldsymbol{X}_j^T\boldsymbol{X}_j)$ is no longer feasible, as we mentioned before. On the other hand, the LS problem can be solved very efficiently using many iterative algorithms if $\boldsymbol{X}_j$ is sparse. All the first-order algorithms such as gradient descent [31], stochastic gradient [32] and accelerated gradient descent [33] can be applied to obtain $(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ without instantiating $(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}$. These algorithms all exploit the sparsity of $\boldsymbol{X}_j$ and thus the per-iteration complexity is very low. In this work, we propose to employ a powerful tool for unconstrained LS – the *conjugate gradient* (CG) algorithm. There is rich literature on CG since it has been the workhorse for solving large-scale LS problems for decades, and we refer the reader to [34], [35] for details; but we highlight some points that are highly relevant for our objectives, in the following remarks.

Remark 1 To solve Problem (11), CG only requires multiplications of $\boldsymbol{X}_j$ and a matrix with size $M_j \times K$ – if $\boldsymbol{X}_j$ is sparse, such multiplications can be easily carried out with a per-iteration complexity of $\mathcal{O}(\mathrm{nnz}(\boldsymbol{X}_j)K)$ flops (see Appendix A). Another advantage of using CG is that even in the worst case, CG is provably convergent to the desired solution within a finite number of iterations [34]; in practice, CG almost always converges to a very good accuracy level within 20 iterations even when the size of $\boldsymbol{X}_i$ reaches $100,000 \times 100,000$. We should mention that using an "intermediate" LS problem to find the (pseudo-)inverse of a large sparse matrix is a classic technique, whose usage in two-view CCA can be traced back thirty years ago [36], [37], although how to extend this idea to GCCA was not clear – and our proposed algorithm fills this gap by using the reformulation in (5).

Using CG, $\boldsymbol{H}_i$ can be computed efficiently. The corresponding algorithm is presented in Algorithm 2. In short, we first use CG to form $\boldsymbol{R}_j = (\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ as we explained. Then, $\boldsymbol{C}_j = \boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ can be formed via multiplying $\boldsymbol{X}_j$ by $\boldsymbol{R}_j$. If $\boldsymbol{X}_j$ is sparse, then this step is easy. Then, the matrix $\boldsymbol{P}_i = \sum_{j\neq i}\boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$ can be obtained by summation over the index set $\{1,\ldots,I\}/\{i\}$. By solving

$$\boldsymbol{E}_i = \arg\min_{\boldsymbol{E}} \ \|\boldsymbol{P}_i - \boldsymbol{X}_i\boldsymbol{E}\|_F^2$$

using CG we obtain $\boldsymbol{E}_i = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\boldsymbol{P}_i$. Finally, by another multiplication of a sparse matrix $\boldsymbol{X}_i$ and the thin matrix $\boldsymbol{E}_i$, we get $\boldsymbol{H}_i$.

Complexity: We factor out each iteration's processing details in the `H_Compute` method. One can see that the major computations all have the same complexity order – and it is linear in $\mathrm{nnz}(\boldsymbol{X}_i)$. Therefore, if the views are sufficiently sparse, the computation of $\boldsymbol{H}_i$ can be very efficient. Recall that in `LasCCA`, the other major operation is taking a thin SVD of $\boldsymbol{H}_i$ which costs $\mathcal{O}(LK^2)$. Combining, one can see that the `LasCCA` algorithm consists of operations that have computational complexity that is linear in L (number of samples) or $\mathrm{nnz}(\boldsymbol{X}_i)$, which is clearly a favorable property. In terms of memory complexity, only thin matrices of size $M_i \times K$ and $L \times K$ are involved. When K is small, the memory cost can be very low.

3.2 Convergence Properties

It is first noted that `LasCCA` monotonically increases the objective value of $\sum_{i=1}^T\sum_{j\neq i}^I \mathrm{Tr}(\boldsymbol{G}_i^T\boldsymbol{H}_i)$ if `CG` solves the associated LS problems. This is by the nature of BCD – each update improves the objective value [31], [38], [39]. Therefore, the objective sequence converges since it is bounded from above and monotonic.

It is also of great interest to study the convergence properties of the solution sequence $\{\boldsymbol{G}_i^{(r)}\}_r$ for all i. Note that since the constraint sets of SUMCOR are not convex, classical BCD convergence results such as those in [31], [38] do not apply to `LasCCA`. Here, we perform custom analysis for the `LasCCA` algorithm. As one will see, the augmentation parameter σ plays an essential role in showing convergence and iteration complexity of `LasCCA`.

As we have been dealing with the reformulated problem after the change of variables, i.e.,

$$\begin{aligned}\max_{\{\boldsymbol{G}_i\}} \ & \sum_{i=1}^I \mathrm{Tr}\left(\boldsymbol{G}_i^T\boldsymbol{H}_i\right) \\ \text{s.t. } & \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}_K, \ \boldsymbol{G}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i), \ \forall k,\end{aligned} \tag{13}$$

where $\boldsymbol{H}_i = \boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\sum_{j\neq i}\boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j$, instead of the original SUMCOR problem in (2), a natural question is that whether or not a KKT point of Problem (13) is also a KKT point of Problem (2). Note that the answer is not obvious: what we have relied upon to handle large-scale GCCA is the fact that the two problems are equivalent at the *global optimal solutions*, but this does not necessarily hold at any KKT point. To address this question, we first show that

Lemma 3 *Let $\{\boldsymbol{G}_i^*\}_{i=1}^I$ be a KKT point of Problem* (13)*. Then, $\{\boldsymbol{Q}_i^* = \boldsymbol{X}_i\boldsymbol{G}_i^*\}_{i=1}^I$ is a KKT point of Problem* (2)*.*

Proof: The proof is relegated to Appendix C. □

Given Lemma 3, it suffices to show that `LasCCA` reaches a KKT point of Problem (13). We show that:

Algorithm 2: H_Compute

input : $\{\boldsymbol{X}_i\}$; $\mathcal{G}_i$.
for $j = 1, \ldots, I$ **do**
  if $j \neq i$ **then**
    $\boldsymbol{R}_j \leftarrow \texttt{CG}(\boldsymbol{X}_j, \boldsymbol{G}_j)$; $\quad \mathcal{O}(\text{nnz}(\boldsymbol{X}_j)K)$
    $\boldsymbol{C}_j \leftarrow \boldsymbol{X}_j \boldsymbol{R}_j$; $\quad \mathcal{O}(\text{nnz}(\boldsymbol{X}_j)K)$
  end
end
$\boldsymbol{P}_i \leftarrow \sum_{j \neq i} \boldsymbol{C}_j$;
$\boldsymbol{E}_i \leftarrow \texttt{CG}(\boldsymbol{X}_i, \boldsymbol{P}_i)$; $\quad \mathcal{O}(\text{nnz}(\boldsymbol{X}_i)K)$
$\boldsymbol{H}_i \leftarrow \boldsymbol{X}_i \boldsymbol{E}_i$; $\quad \mathcal{O}(\text{nnz}(\boldsymbol{X}_i)K)$
output: $\boldsymbol{H}_i$

Proposition 1 *Assume that* $\text{rank}(\boldsymbol{X}_i) = M_i$ *and* $0 < \sigma < +\infty$. *Then, the* LasCCA *algorithm has the following convergence properties:*

a) *(Subsequence Convergence) Every limit point of the solution sequence produced by* LasCCA *is a KKT point of Problem* (2).
b) *(Global Convergence) The whole solution sequence produced by* LasCCA *converges, i.e,* $\boldsymbol{G}_i^{(r+1)} \to \boldsymbol{G}_i^{(r)}$ *for all* i; *in addition, the sequence converges to a set* $\mathcal{K}$ *which consists of all the KKT points of Problem* (2).
c) *(Iteration Complexity) We have*

$$\sum_{i=1}^{I} \|\boldsymbol{G}_i^{(r+1)} - \boldsymbol{G}_i^{(r)}\|_F^2 = \mathcal{O}(1/r);$$

i.e., LasCCA *converges to a KKT point at least sublinearly.*

Proof: Please see Appendix D. □

Remark 2 A practical issue here is how to choose σ? In proximal term-augmented BCD, a relatively large σ is sometimes employed to improve the conditioning of the associated subproblems. The trade-off here is that if σ is set to be too large, it means that $\boldsymbol{G}_i^{(r+1)}$ is confined to be close to $\boldsymbol{G}_i^{(r)}$, which may result in slow convergence; if σ is too small, the subproblems may be still ill-conditioned and hard to be solved efficiently. In our case, fortunately, we do not need a large σ to improve conditioning of the subproblem – our subproblem w.r.t. $\boldsymbol{G}_i$ can be easily solved. Therefore, one empirically effective choice is to use a small $\sigma > 0$, e.g., $\sigma = 10^{-8}$, which will not restrict the algorithm to small step sizes but can offer theoretical guarantees of convergence.

4 Distributed GCCA (DisCCA)

In practice, there are many scenarios in which distributed computing is desirable. For example, sometimes the views (i.e., $\{\boldsymbol{X}_i\}_{i=1,\ldots,I}$) may be collected and stored at different nodes within a network, and directly exchanging the views $\boldsymbol{X}_i$ may not be allowed for security and/or legal reasons. Another reason is multi-core parallel computing-based acceleration: instead of computing $\boldsymbol{G}_i$ sequentially as in Algorithm 1, it may be more appealing to compute $\boldsymbol{G}_i$ for all i in parallel in a distributed fashion – which may well expedite the whole process when I is large. In this section, we propose a distributed implementation that is

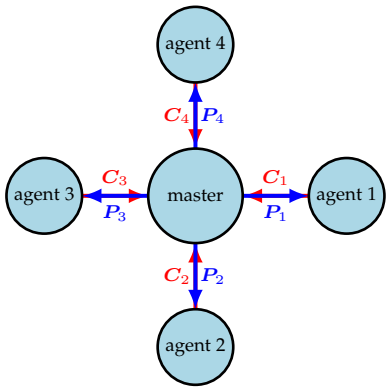


Fig. 2. Architecture of the considered network for DisCCA and information exchange among distributed nodes in DisCCA.

a greedy variant of Algorithm 1. The proposed approach uses a master to coordinate the nodes or cores (cf. Fig. 2), and most computations happen locally at the nodes; in addition, the information exchanges only involve small-size thin matrices. Convergence analysis will be used to back this implementation in the next section.

Let us consider a network structure as in Fig. 2. There, the nodes can be machines that are located in different places or cores within one machine. The views are stored at the nodes and are not allowed to be exchanged. The master is a coordinator who collects and distributes some derivative information, and we wish to leave the heavy computations to the nodes. To see how we approach this problem, let us consider the following modification to LasCCA. Specifically, instead of updating $\boldsymbol{G}_i$ sequentially and cyclically as in Algorithm 1, the nodes update locally $\mathbf{G}_i$ simultaneously based on the previous iterate, which results in the following update strategy:

$$\boldsymbol{H}_i^{(r)} \leftarrow \boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T \sum_{j \neq i} \boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j^{(r)} \tag{14a}$$

$$\boldsymbol{U}_i^{(r)}\boldsymbol{\Sigma}_i^{(r)}\boldsymbol{V}_i^{(r)} \leftarrow \text{svd}\left(\boldsymbol{H}_i^{(r)} + \sigma\boldsymbol{G}_i^{(r)}\right) \tag{14b}$$

$$\boldsymbol{G}_i^{(r+1)} \leftarrow \boldsymbol{U}_i^{(r)}(\boldsymbol{V}_i^{(r)})^T. \tag{14c}$$

Note that the update in (14) is different from that in Algorithm 1: in LasCCA (Algorithm 1), the update of $\boldsymbol{G}_i^{(r)}$ uses the information of $\boldsymbol{G}_j^{(r+1)}$ for $j < i$ and $\boldsymbol{G}_j^{(r)}$ for $j > i$, while this update in (14) only uses $\boldsymbol{G}_j^{(r)}$ for all $j \neq i$. By this slight modification, a significant part of (14) can be computed in parallel at different nodes.

After all $\boldsymbol{G}_i^{(r+1)}$'s are computed by different nodes, a natural question is which one to update. A simple and straightforward way is to update the $\boldsymbol{G}_i$ that brings maximal improvement to the objective function of (5). Using this idea, we obtain the distributed CCA (DisCCA) algorithm that is presented in Algorithm 3. One can see that the lines 25-30 implement the idea of selecting a block to update. In optimization theory, adding these lines makes the algorithm fall into the *maximum block improvement* (MBI) framework [40] which is a greedy variant of the Gauss-Seidel (GS)-type BCD. To distinguish MBI-based distributed GCCA with the one that will be proposed later, we call Algorithm 3 DisCCA-MBI.

Compared to GS-type BCD (cf. LasCCA), every update of DisCCA-MBI gives a larger improvement of the objective, which is clearly favorable. Another motivation of using MBI is that it has very low communication overhead between the nodes and the master. Specifically, after every iteration (indexed by r), only the node who has updated its associated $\boldsymbol{G}_i^{(r)}$ needs to send out $\boldsymbol{C}_i^{(r)}$ (cf. line 16 in Algorithm 3), and the other nodes do not have to send out anything. In the broadcasting stage, the master node sends $\boldsymbol{P}_i$'s to the agents, which are also thin matrices. When communication is expensive, such a strategy is quite economical— since sending $\boldsymbol{C}_i$ and $\boldsymbol{P}_i$ only costs $\mathcal{O}(KL)$ overhead and K is usually quite small. On the other hand, DisCCA-MBI "wastes" most of the computations since it computes all the blocks but updates only one. Hence, there is clearly an interesting resource-time trade-off between choosing LasCCA and DisCCA-MBI in practice.

Complexity: The computational and memory complexities of DisCCA-MBI are the same as those of LasCCA, since all the basic operations are the same, except that DisCCA-MBI employs multiple computing agents. The communication overhead that arises in the distributed algorithm is in the order of $\mathcal{O}(LK)$ in each iteration, as mentioned. Since K is usually much smaller than L, this amount of overhead is easily affordable in most cases.

4.1 Convergence Properties of DisCCA-MBI

Unlike LasCCA that uses a GS-type BCD updating rule for the blocks, DisCCA-MBI employs the so-called Gauss-Southwell rule. As a consequence, DisCCA-MBI needs slightly weaker conditions to establish subsequence convergence and global convergence (i.e., the a) and b) parts in Theorem 1). To be specific, to show subsequence and global convergence of DisCCA-MBI, we do not need to assume $\sigma > 0$ – any value of $\sigma \geq 0$ suffices to establish these results. In fact, these results can be directly obtained by applying the convergence analysis of the maximum block improvement strategy as in [40]. To show iteration complexity of DisCCA, on the other hand, $\sigma > 0$ will be needed. To be specific, we have the following properties of DisCCA-MBI:

Proposition 2 *Assume that* $\mathrm{rank}(\boldsymbol{X}_i) = M_i$. *Then, the DisCCA algorithm has the following convergence properties:*

a) *(Subsequence Convergence) When* $\sigma \geq 0$, *every limit point of the solution sequence produced by DisCCA-MBI is a KKT point of Problem* (2).

b) *(Global Convergence) When* $\sigma \geq 0$, *the whole solution sequence produced by DisCCA-MBI converges, i.e,* $\boldsymbol{G}_i^{(r+1)} \to \boldsymbol{G}_i^{(r)}$ *for all* i; *in addition, the sequence converges to a set* $\mathcal{K}$ *which consists of all the KKT points of Problem* (2).

c) *(Iteration Complexity) Assume* $\sigma > 0$, *we have*

$$\sum_{i=1}^{I} \left\| \boldsymbol{G}_i^{(r+1)} - \boldsymbol{G}_i^{(r)} \right\|_F^2 = \mathcal{O}(1/r);$$

i.e., DisCCA-MBI converges to a KKT point at least sublinearly.

Proof: Please see Appendix E. □

Algorithm 3: DisCCA-MBI

input : $\{\boldsymbol{X}_i\}_{i=1}^I$; K

1 $\boldsymbol{Q}_i = \mathrm{randn}(M_i, K)$;

2 $\boldsymbol{U}_i \boldsymbol{\Sigma} \boldsymbol{V}_i^T = \mathrm{svd}(\boldsymbol{X}_i \boldsymbol{Q}_i, K, '\,\mathrm{econ}')$, $\boldsymbol{G}_i^{(0)} = \boldsymbol{U}_i$;

3 $r \leftarrow 0$;

4 **repeat**

5 **if** $r = 0$ **then**

6 $\boldsymbol{R}_i^{(r)} \leftarrow \mathtt{CG}(\boldsymbol{X}_i, \boldsymbol{G}_i^{(r)})$; (node)

7 $\boldsymbol{C}_i^{(r)} \leftarrow \boldsymbol{X}_i \boldsymbol{R}_i^{(r)}$; (node)

8 node i for all i sends $\boldsymbol{C}_i$ to master;

9 $\boldsymbol{P}_i^{(r)} \leftarrow \sum_{j \neq i} \boldsymbol{C}_j^{(r)}$; (master)

10 master sends $\boldsymbol{P}_i$ to node i;

11 $\boldsymbol{E}_i^{(r)} \leftarrow \mathtt{CG}(\boldsymbol{X}_i, \boldsymbol{P}_i^{(r)})$; (node)

12 $\boldsymbol{H}_i^{(r)} \leftarrow \boldsymbol{X}_i \boldsymbol{E}_i^{(r)}$; (node)

13 **else**

14 $\boldsymbol{R}_{i^{(r-1)}}^{(r)} \leftarrow \mathtt{CG}\left(\boldsymbol{X}_i, \boldsymbol{G}_{i^{(r)}}^{(r)}\right)$; (node $i^{(r)}$)

15 $\boldsymbol{C}_{i^{(r-1)}}^{(r)} \leftarrow \boldsymbol{X}_i \boldsymbol{R}_{i^{(r-1)}}^{(r)}$; (node $i^{(r)}$)

16 node $i^{(r)}$ sends $\boldsymbol{C}_{i^{(r-1)}}^{(r)}$ to master;

17 $\boldsymbol{P}_i^{(r)} \leftarrow \boldsymbol{P}_i^{(r)} - \boldsymbol{C}_{i^{(r-1)}}^{(r-1)} + \boldsymbol{C}_{i^{(r-1)}}^{(r)}$ for $i \neq i^{(r-1)}$; (master)

18 master sends $\boldsymbol{P}_i^{(r)}$ to node i for $i \neq i^{(r-1)}$;

19 $\boldsymbol{P}_{i^{(r-1)}}^{(r)} \leftarrow \boldsymbol{P}_{i^{(r-1)}}^{(r-1)}$; (node)

20 $\boldsymbol{E}_i^{(r)} \leftarrow \mathtt{CG}(\boldsymbol{X}_i, \boldsymbol{P}_i^{(r)})$; (node)

21 $\boldsymbol{H}_i^{(r)} \leftarrow \boldsymbol{X}_i \boldsymbol{E}_i^{(r)}$; (node)

22 **end**

23 $\boldsymbol{U}_i^{(r)} \boldsymbol{\Sigma}_i^{(r)} \boldsymbol{V}_i^{(r)} \leftarrow \mathrm{svd}\left(\boldsymbol{H}_i^{(r)} + \sigma \boldsymbol{G}_i^{(r)}\right)$. (node)

24 $\boldsymbol{G}_i^{(r+1)} \leftarrow \boldsymbol{U}_i^{(r)} (\boldsymbol{V}_i^{(r)})^T$. (node)

25 $v_i^{(r)} \leftarrow \mathrm{Tr}((\boldsymbol{G}_i^{(r+1)})^T \boldsymbol{H}_i^{(r)})$; (node)

26 node i sends $v_i^{(r)}$ to master;

27 $i^{(r)} \leftarrow \arg\max_{i \in \{1,\ldots,I\}} v_i^{(r)}$; (master)

28 master sends an update command to node $i^{(r)}$;

29 $\boldsymbol{G}_{i^{(r)}}^{(r+1)} \leftarrow \boldsymbol{G}_{i^{(r)}}^{(r+1)}$; (node)

30 $\boldsymbol{G}_i^{(r+1)} \leftarrow \boldsymbol{G}_i^{(r)}$ for $i \neq i^{(r)}$; (node)

31 $r \leftarrow r + 1$;

32 **until** *some stopping criterion is reached*;

output: $\{\boldsymbol{G}_i\}$

Remark 3 Like LasCCA, DisCCA-MBI offers monotonically increasing objective values of Problem (13). This is by the nature of MBI and straightforward to show. This is important in practice, because it ensures that every iteration makes progress towards our goal of capturing more sum-of-correlations. Note that Propositions 1–2 summarizes the convergence properties of LasCCA and DisCCA-MBI under any positive integer $I \geq 2$. One interesting question is that, when $I = 2$, where the SUMCOR CCA problem is solvable, do the large-scale algorithms lose optimality? The answer is no—and the algorithms converge to a global optimal solution at a *linear rate* (which is in general much faster compared to sublinear rate convergence). This is reminiscent of the *orthogonal iteration* algorithm [35] for computing leading eigenvectors of a symmetric matrix; see the supplementary materials (Appendix F) for details.

5 DisCCA via Gradient Projection

As mentioned, DisCCA-MBI tends to "waste" most of the computational resources since it computes all potential updates of the $\boldsymbol{G}_i$'s but only implements one block. This

raises a natural question: is there a way to update $\boldsymbol{G}_i$ for $i = 1, \ldots, I$ in parallel? The answer is affirmative – if one is willing to afford more communication overhead.

To explain the approach, let us re-write the GCCA problem in (13) as the following:

$$\begin{aligned} &\min_{\boldsymbol{G}} \ \mathrm{Tr}\left(\boldsymbol{G}^T \boldsymbol{B} \boldsymbol{G}\right) \\ &\text{s.t. } \boldsymbol{G}_i^T \boldsymbol{G}_i = \boldsymbol{I}_K, \ \boldsymbol{G}_i \in \mathcal{R}(\boldsymbol{X}_i), \ \forall i, \end{aligned} \tag{15}$$

where $\boldsymbol{G} = [\boldsymbol{G}_1^T, \ldots, \boldsymbol{G}_I^T]^T$, and

$$\boldsymbol{B} = -\begin{bmatrix} \boldsymbol{0}, & \boldsymbol{S}_1\boldsymbol{S}_2, & \ldots, & \boldsymbol{S}_1\boldsymbol{S}_I \\ \boldsymbol{S}_2\boldsymbol{S}_1, & \boldsymbol{0}, & \ldots, & \boldsymbol{S}_2\boldsymbol{S}_I \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{S}_I\boldsymbol{S}_1, & \boldsymbol{S}_I\boldsymbol{S}_2 & \ldots, & \boldsymbol{0} \end{bmatrix}, \tag{16}$$

and $\boldsymbol{S}_i = \boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T$. Instead of resorting to block coordinate descent, let us consider optimizing Problem (15) w.r.t. all the variables at once, i.e., $\boldsymbol{G}$. We propose the following gradient projection (GP) procedure:

$$\boldsymbol{G}^{(r+1)} \leftarrow \mathcal{P}_{\mathcal{G}}\left(\boldsymbol{G}^{(r)} - \alpha \nabla f(\boldsymbol{G}^{(r)})\right), \tag{17}$$

where $f(\boldsymbol{G})$ denotes the objective function of (15), $\mathcal{P}_{\mathcal{G}}(\cdot)$ denotes the operator of projecting to the set $\mathcal{G}$, where $\mathcal{G} = \{\boldsymbol{G} \mid \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}, \boldsymbol{G}_i \in \mathcal{R}(\boldsymbol{X}_i), \ \forall i\}$, i.e., the constraint set of (15), and $\alpha > 0$ is the step size of GP. The motivation of employing GP is distributed implementation. To explain, one can show that

$$\nabla f(\boldsymbol{G}^{(r)}) = \left[\left(\nabla f_1(\boldsymbol{G}_1^{(r)})\right)^T, \ldots, \left(\nabla f_I(\boldsymbol{G}_I^{(r)})\right)^T\right]^T,$$

and $\nabla f_i(\boldsymbol{G}_i^{(r)}) = -\boldsymbol{H}_i^{(r)}$, where

$$\boldsymbol{H}_i^{(r)} = \boldsymbol{X}_i(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T \sum_{j\neq i} \boldsymbol{X}_j(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1}\boldsymbol{X}_j^T\boldsymbol{G}_j^{(r)}$$

is defined as before. Therefore, the update rule in (17) is completely separable w.r.t. each of the $\boldsymbol{G}_i$'s. In fact, the updates w.r.t. $\boldsymbol{G}_i$'s are very similar to those in `LasCCA` and DisCCA. We have the following

Lemma 4 *Assume that $\boldsymbol{G}_i^{(r)} \in \mathcal{R}(\boldsymbol{X}_i)$. Then, the following solution*

$$\begin{aligned} \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T &\leftarrow \mathrm{svd}\left(\boldsymbol{G}_i^{(r)} + \alpha\boldsymbol{H}_i^{(r)}\right) \\ \boldsymbol{G}_i^{(r+1)} &\leftarrow \boldsymbol{U}_i\boldsymbol{V}_i^T, \quad \forall i, \end{aligned} \tag{18}$$

is optimal for (17).

Proof: The projection problem in (17) can be written as

$$\begin{aligned} &\min_{\boldsymbol{G}} \ \left\|\boldsymbol{G} - \left(\boldsymbol{G}^{(r)} - \alpha\nabla f(\boldsymbol{G}^{(r)})\right)\right\|_F^2 \\ &\text{s.t. } \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}, \ \boldsymbol{G}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i), \ \forall k, i. \end{aligned}$$

By noting that the above is separable w.r.t. $\boldsymbol{G}_i$, solving the above amounts to solving

$$\begin{aligned} &\min_{\boldsymbol{G}_i} \ \left\|\boldsymbol{G}_i - \left(\boldsymbol{G}_i^{(r)} + \alpha\boldsymbol{H}_i^{(r)}\right)\right\|_F^2 \\ &\text{s.t. } \boldsymbol{G}_i^T\boldsymbol{G}_i = \boldsymbol{I}, \ \boldsymbol{G}_i(:,k) \in \mathcal{R}(\boldsymbol{X}_i), \ \forall k, i. \end{aligned}$$

Algorithm 4: `DisCCA-GP`

input : $\{\boldsymbol{X}_i\}_{i=1}^I$; K

$\boldsymbol{U}_i\boldsymbol{\Sigma}\boldsymbol{V}_i^T = \mathrm{svds}(\boldsymbol{X}_i, K)$, $\boldsymbol{G}_i^{(0)} = \boldsymbol{U}_i$; (node)

$r \leftarrow 0$;

repeat

- $\boldsymbol{R}_i^{(r)} \leftarrow \mathtt{CG}(\boldsymbol{X}_i, \boldsymbol{G}_i^{(r)})$; (node)
- $\boldsymbol{C}_i^{(r)} \leftarrow \boldsymbol{X}_i\boldsymbol{R}_i^{(r)}$; (node)
- node i for all i sends $\boldsymbol{C}_i$ to master;
- $\boldsymbol{P}_i^{(r)} \leftarrow \sum_{j\neq i}\boldsymbol{C}_j^{(r)}$; (master)
- master sends $\boldsymbol{P}_i$ to node i;
- $\boldsymbol{E}_i^{(r)} \leftarrow \mathtt{CG}(\boldsymbol{X}_i, \boldsymbol{P}_i^{(r)})$; (node)
- $\boldsymbol{H}_i^{(r)} \leftarrow \boldsymbol{X}_i\boldsymbol{E}_i^{(r)}$; (node)
- $\boldsymbol{U}_i^{(r)}\boldsymbol{\Sigma}_i^{(r)}(\boldsymbol{V}_i^{(r)})^T \leftarrow \mathrm{svd}\left(\alpha\boldsymbol{H}_i^{(r)} + \boldsymbol{G}_i^{(r)}\right)$. (node)
- $\boldsymbol{G}_i^{(r+1)} \leftarrow \boldsymbol{U}_i^{(r)}(\boldsymbol{V}_i^{(r)})^T$. (node)
- $r \leftarrow r + 1$;

until *some stopping criterion is reached*;

output: $\{\boldsymbol{G}_i\}$

for all i. Now, using the assumption that $\boldsymbol{G}_i^{(r)} \in \mathcal{R}(\boldsymbol{X}_i)$ and the same arguments as in Proposition 2 and Corollary 1, one can see that the solution in (18) solves the above problem to optimality. □

By Lemma 4, we can implement a gradient projection algorithm for Problem (15) in a distributed manner very easily – see the details in Algorithm 4. We name this algorithm `DisCCA-GP`. The algorithm looks very similar to `DisCCA-MBI`, with some slight differences: first, all the nodes update their own $\boldsymbol{G}_i$'s in each iteration, which results in high efficiency in terms of utilizing the computational resources. Second, all the nodes send out $\boldsymbol{C}_i$ and receive an updated $\boldsymbol{P}_i$ after each iteration, which leads to as much as twice the communication overhead compared to that of `DisCCA-MBI`. In cases where communication overhead is not a serious concern, `DisCCA-GP` is clearly a good choice.

Complexity: As `DisCCA-MBI`, `DisCCA-GP` does not increase the computational and memory complexities compared to `LasCCA`. The communication overhead, however, is $\mathcal{O}(2LK)$ in each iteration which is twice as needed in `DisCCA-MBI`. The reason is that each node needs to update its own $\boldsymbol{C}_i$ to the master in the GP algorithm, while only the node updated $\boldsymbol{G}_i$ needs to report to the master in the MBI algorithm.

5.1 Convergence Properties of DisCCA-GP

When using gradient projection over a nonconvex set, a critical consideration is convergence. In general, GP is not guaranteed to converge when the constraint is nonconvex. In fact, it is not even guaranteed to decrease the objective function under such cases. Fortunately, our objective function in (15) has special structure that one can exploit to ensure convergence and monotonicity. To explain, we first show that

Lemma 5 *Assume that* $\mathrm{rank}(\boldsymbol{X}_i) = M_i$ *and* α *is chosen such that* $\alpha < 1$. *Then,* `DisCCA-GP` *ensures the following holds:*

$$f\left(\boldsymbol{G}^{(r)}\right) - f\left(\boldsymbol{G}^{(r+1)}\right) \geq \left(\frac{1}{2\alpha} - \frac{1}{2}\right)\|\boldsymbol{G}^{(r)} - \boldsymbol{G}^{(r+1)}\|_F^2.$$

Proof: The key of the proof is to show that the objective function in (15) has a 1-Lipschitz continuous gradient, or, largest eigenvalue of the matrix $\boldsymbol{B}$ can be bounded by 1. To this end, let us re-write $\boldsymbol{B}$ as

$$\boldsymbol{B} = -\begin{bmatrix}\boldsymbol{S}_1\\ \boldsymbol{S}_2\\ \vdots\\ \boldsymbol{S}_I\end{bmatrix}[\boldsymbol{S}_1,\ldots,\boldsymbol{S}_I] + \begin{bmatrix}\boldsymbol{S}_1, & \boldsymbol{0}, & \ldots, & \boldsymbol{0}\\ \boldsymbol{0}, & \boldsymbol{S}_2, & \ldots, & \boldsymbol{0}\\ \vdots & \vdots & \vdots & \vdots\\ \boldsymbol{0}, & \boldsymbol{0} & \ldots, & \boldsymbol{S}_I\end{bmatrix}, \quad (19)$$

where we have used the fact that $\boldsymbol{S}_i\boldsymbol{S}_i = \boldsymbol{S}_i$ since $\boldsymbol{S}_i$ is an orthogonal projector. Another property of $\boldsymbol{S}_i$ is that all the eigenvalues are 1. Indeed, Let $\boldsymbol{X}_i = \boldsymbol{U}_{\boldsymbol{X}_i}\boldsymbol{\Sigma}_{\boldsymbol{X}_i}\boldsymbol{V}_{\boldsymbol{X}_i}^T$. One can see that $\boldsymbol{S}_i = \boldsymbol{U}_{\boldsymbol{X}_i}\boldsymbol{U}_{\boldsymbol{X}_i}^T$. Consequently, the later matrix is congruent with the identity matrix $\boldsymbol{I}$. Notice that the first term in the right hand side of (19) is negative semidefinite since its a Gram matrix with a minus sign. Therefore, the matrix $\boldsymbol{B}$ satisfies

$$\lambda_{\max}(\boldsymbol{B}) \leq 1.$$

Relying on the above, now we are ready to show that $f(\boldsymbol{G})$ has sufficient decrease at each iteration. We first notice that

$$f(\boldsymbol{G}^{(r+1)}) \leq f(\boldsymbol{G}^{(r)}) + \left\langle \boldsymbol{B}\boldsymbol{G}^{(r)}, \boldsymbol{G}^{(r+1)} - \boldsymbol{G}^{(r)}\right\rangle + \frac{1}{2}\|\boldsymbol{G}^{(r)} - \boldsymbol{G}^{(r+1)}\|_F^2, \quad (20)$$

by the Lipschitz continuity of the gradient of the objective [38]. In addition, we have

$$\left\langle \boldsymbol{B}\boldsymbol{G}^{(r)}, \boldsymbol{G}^{(r+1)} - \boldsymbol{G}^{(r)}\right\rangle + \frac{1}{2\alpha}\|\boldsymbol{G}^{(r)} - \boldsymbol{G}^{(r+1)}\|_F^2 \leq \left\langle \boldsymbol{B}\boldsymbol{G}^{(r)}, \boldsymbol{G}^{(r)} - \boldsymbol{G}^{(r)}\right\rangle + \frac{1}{2\alpha}\|\boldsymbol{G}^{(r)} - \boldsymbol{G}^{(r)}\|_F^2, \quad (21)$$

since $\boldsymbol{G}^{(r+1)}$ satisfies

$$\boldsymbol{G}^{(r+1)} = \arg\min_{\boldsymbol{G}^T\boldsymbol{G}=\boldsymbol{I}} f(\boldsymbol{G}^{(r)}) + \left\langle \boldsymbol{B}\boldsymbol{G}^{(r)}, \boldsymbol{G} - \boldsymbol{G}^{(r)}\right\rangle + \frac{1}{2\alpha}\|\boldsymbol{G} - \boldsymbol{G}^{(r)}\|_F^2 \quad (22)$$

following the upper-bound interpretation of gradient projection [38]. Combining (20)-(22), the proof is completed. □

Leveraging Lemma 5, we show the following:

Proposition 3 *Assume that* $\mathrm{rank}(\boldsymbol{X}_i) = M_i$ *and* α *is chosen such that* $\alpha < 1$. *Then, the* `DisCCA-GP` *algorithm has the following convergence properties:*

a) *(Subsequence Convergence) Every limit point of the solution sequence produced by* `DisCCA-GP` *is a KKT point of Problem* (2).
b) *(Global Convergence) The whole solution sequence produced by* `DisCCA-GP` *converges to a set* $\mathcal{K}$ *which consists of all the KKT points of Problem* (2).
c) *(Iteration Complexity) We have*

$$\sum_{i=1}^{I}\left\|\boldsymbol{G}_i^{(r+1)} - \boldsymbol{G}_i^{(r)}\right\|_F^2 = \mathcal{O}(1/r);$$

i.e., `DisCCA-GP` *converges to a KKT point at least sublinearly.*

Proof: The algorithm is gradient projection over nonconvex smooth manifolds. The proof follows the recent work in [41]. The key of the proof in [41] is that the objective function has a Lipschitz continuous gradient and every iteration yields sufficient decrease of the objective function – which we have shown in Lemma 5. The other steps are the same as those in [41] and thus is omitted. □

In Table 1, we summarize properties of the proposed three algorithms, where we consider a single-core implementation of `LasCCA` and an I-core implementation of `DisCCA-MBI` and `DisCCA-GP`, respectively. The difference between `DisCCA-MBI` and `DisCCA-GP` is that `DisCCA-MBI` has lower communication overhead and also lower "block update efficiency" – it only updates one $\boldsymbol{G}_i$ in iteration r; `DisCCA-GP`, on the other hand, has twice as much communication overhead relative to `DisCCA-MBI`, but it updates all $\boldsymbol{G}_i$'s in iteration r – which means higher efficiency of computational resource usage. Practitioners should to choose from `LasCCA`, `DisCCA-MBI`, and `DisCCA-GP` according to the resources that they have.

TABLE 1
Summary of properties of proposed algorithms.

Algorithm	Distributed	No. of Cores	Overhead	Updated Blocks/Iter.
`LasCCA`	✗	one	–	I (No. of views)
`DisCCA-MBI`	✓	I (No. of views)	$\mathcal{O}(KL)$	one
`DisCCA-GP`	✓	I (No. of views)	$\mathcal{O}(2KL)$	I (No. of views)

Remark 4 We presented `LasCCA`, `DisCCA-MBI`, and `DisCCA-GP` under several assumptions, e.g., $\boldsymbol{X}_i$ is already centered and $\mathrm{rank}(\boldsymbol{X}_i) = M_i$ for all i. In practice, these are necessarily automatically satisfied and directly applying the proposed algorithms may be problematic. For example, if the uncentered data $\boldsymbol{X}_i$ is sparse, subtracting the mean of the data samples from all rows of $\boldsymbol{X}_i$ (i.e., centering) will make the centered data very dense – which is disastrous for the subsequent processing. In addition, if $\boldsymbol{X}_i$ is rank-deficient, computing $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}$ may lead to numerical problems. Nevertheless, these problems can be circumvented by carefully modifying the proposed algorithms – please see Appendix B for details.

6 Experiments

In this section, we use synthetic and real-data experiments to showcase the effectiveness of `LasCCA` and `DisCCA`. All the algorithms are implemented in `MATLAB` and are tested on a Linux server with multiple 2GHz cores and 128 GB RAM.

• **Baselines.** We employ the GCCA algorithms in the literature that can deal with large-scale multiview data and extract multiple canonical components simultaneously. To serve these purposes, we first employ the algorithms that use a strategy for change of variables as proposed in [13], [14], [15]. The algorithm follows a BCD strategy where each subproblem is as in (3). The difficulty of applying the exact algorithms in [13], [14] here is that those algorithms were not designed to handle the $K > 1$ case. But this can be fixed by solving each subproblem in (3) using a Procrustes projection. This baseline will be referred to as *correlation square root-based block coordinate descent* (`CSR-BCD`). In addition, we also use the *multiview latent semantic analysis* (`MVLSA`) algorithm that was proposed by Rastogi *et al.* [7]. We should

mention that MVLSA aims at solving the MAX-VAR GCCA problem, which has a different objective function:

$$\min_{\boldsymbol{G}^T\boldsymbol{G}=\boldsymbol{I}_K,\{\boldsymbol{Q}_i\}} \sum_{i=1}^{I} \|\boldsymbol{X}_i\boldsymbol{Q}_i - \boldsymbol{G}\|_F^2 \tag{23}$$

The formulation in (23) enforces a common latent structure on the different views, which may lose generality if SUMCOR is sought since pairwise similarity does not imply an overall similarity. Nevertheless, MAX-VAR and SUMCOR conceptually share the same goal – which is to seek highly correlated $\boldsymbol{X}_i\boldsymbol{Q}_i$'s – and thus using it as a baseline is reasonable. More importantly, the MVLSA algorithm scales very well. We run DisCCA-MBI and DisCCA-GP using a multi-core implementation – i.e., we use I cores if we have I views. Through this section, the maximal number of iterations of CG is fixed to be 20 in our algorithms. For LasCCA and DisCCA-MBI, we set $\sigma = 10^{-8}$; and we let $\alpha = 0.99$ for DisCCA-GP.

6.1 Synthetic-Data Experiments

• **Data Generation.** We generate the synthetic data as follows: First, we let $\boldsymbol{Z} \in \mathbb{R}^{L\times M}$ be a common latent factor of different views, where $\boldsymbol{Z}$ is a randomly generated sparse matrix whose non-zero entries follow the i.i.d. zero-mean unit-variance Gaussian distribution. Then, a sparse matrix $\boldsymbol{A}_i \in \mathbb{R}^{M\times M}$ is multiplied to $\boldsymbol{Z}$, resulting in $\boldsymbol{X}_i = \boldsymbol{Z}\boldsymbol{A}_i$. The overall sparsity level of $\boldsymbol{X}_i$ is controlled to satisfy experiment specifications. This way, the views have a perfectly correlated common latent factor, i.e., $\boldsymbol{Z}$, and this fact can be used to benchmark the performance of the algorithms, as will be seen shortly. In the synthetic-data simulations, we fix $M_1 = \ldots = M_I = M$, and test the algorithms under different sizes and sparsity levels of $\boldsymbol{X}_i$, where the sparsity level ρ_i is defined as $\rho_i = {}^{\text{nnz}(\boldsymbol{X}_i)}/_{LM}$, and we let $\rho_1 = \ldots = \rho_I = \rho$ in this section.

• **Evaluation.** We evaluate the synthetic data experiment by observing the captured sum of pairwise correlations ("correlation captured" in short) between the views. We generate data that are perfectly correlated in some latent domain, and thus the optimal correlation value is known. Specifically, the optimal value of Problem (5) is $v_{\text{opt}} = I(I-1)K$ when $\boldsymbol{A}_i\boldsymbol{Q}_i$ for all i's are all aligned to a same right singular vector space of $\boldsymbol{Z}$ – under such a scenario $\boldsymbol{X}_i\boldsymbol{Q}_i$'s are all perfectly correlated with each other. Therefore, this value can be used to benchmark all the algorithms. We also define a metric called "95time", which records the time that is needed for an algorithm to capture 95% of the optimal value of the total sum of correlations, i.e., $I(I-1)K$.

• **Results.** Table 2 shows the performance of the algorithms of different L's (and M_i's). Here, we fix $\rho = 5\times 10^{-3}$ and change L and M, where M is set to be $M = 0.8\times L$. We seek $K = 5$ canonical components for each of the $I = 5$ views. MVLSA requires first truncating the rank of the views to a certain number, and thus we use the first 100 singular values and vectors to approximate each view. Note that under the above settings, the optimal value of sum of correlations is $v_{\text{opt}} = 100$. We let all the iterative algorithms run for 20 iterations in this experiment, and the results are averaged over 10 random trials where in each trial $\boldsymbol{Z}$ and $\boldsymbol{A}_i$ are randomly generated. We initialize CSR-BCD, LasCCA and DisCCA using the same initialization which is randomly generated within $\mathcal{R}(\boldsymbol{X}_i)$ – i.e., we use $\boldsymbol{G}_i^{(0)} = \boldsymbol{U}_i$ where $\boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T = \text{svd}(\boldsymbol{X}_i\boldsymbol{Q}_i)$ and $\boldsymbol{Q}_i$ is randomly generated.

TABLE 2
Average correlations captured of the algorithms. $M = 0.8\times L$; $K = 5$; $I = 5$; $\rho = 5\times 10^{-3}$; "†" means "out-of-memory"; $v_{\text{opt}} = 100$.

Algorithm	measure	$L(\times 10^3)$				
		1	5	10	50	100
LasCCA	corr. cap.	99.86	99.30	99.07	**99.04**	**99.07**
	95time (sec.)	0.35	2.40	5.58	158.68	684.62
DisCCA-MBI	corr. cap.	99.66	98.26	97.88	97.80	97.85
	95time (sec.)	1.41	2.60	6.44	127.50	596.18
DisCCA-GP	corr. cap.	99.73	98.73	98.37	98.31	98.30
	95time (sec.)	0.18	**0.43**	**1.33**	55.55	**239.86**
CSR-BCD	corr. cap.	**100**	**100**	**100**	†	†
	95time (sec.)	10.83	2841.81	25828.17	†	†
MVLSA (50)	corr. cap.	94.23	51.92	28.27	6.47	3.31
	time (sec.)	**0.09**	0.72	1.89	**55.01**	312.26
MVLSA (250)	corr. cap.	100.00	81.75	57.56	13.33	6.60
	time (sec.)	1.17	12.16	60.82	170.57	800.46

As one can see from the table, in terms of captured correlations, the CSR-BCD algorithm, which uses the change of variables suggested in [14], [13], [15], gives the best performance when $L \leq 10,000$ – it always reaches the optimal objective value, although the problem is non-convex and NP-hard. However, when $L = 50,000$, the algorithm exhausts the memory of our machine, since it needs to calculate I dense matrices with a size of $M_i \times M_i$ and store them. To be precise, this method needs $\mathcal{O}(2.5\times 10^9)$ memory space for instantiating each $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}$ when $L = 50,000$. The proposed algorithms, i.e., LasCCA, DisCCA-MBI and DisCCA-GP, give slightly lower objective values compared to that of CSR-BCD, which are already very good – they both capture around 98% of the total correlations, even in the cases where CSR-BCD cannot run, i.e., when $L = 50,000$ and $L = 100,000$. Another observation is that when $L = 1,000$, MVLSA works very well, i.e., most correlations are captured when truncating the ranks of $\boldsymbol{X}_i$'s to 50 and 250, respectively (cf. the results of MVLSA (50) and MVLSA (250), respectively). However, when L increases, the performance degrades rapidly since the approximation of the views becomes much coarser.

In terms of 95time, as one can see from the same table (also see Fig. 1), the runtime of CSR-BCD grows rapidly with L increasing. The reason is now clear: besides the difficulty of calculating $(\boldsymbol{X}^T\boldsymbol{X})^{-1/2}$ (a full eigen-decomposition of a dense matrix which costs $\mathcal{O}(1.25\times 10^{14})$ flops), many operations in the algorithm, e.g., $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1/2}\boldsymbol{Z}_i$, are large dense matrix-matrix multiplications, which are very difficult to compute. On the other hand, the proposed algorithms scale well. In particular, DisCCA-GP gives the best 95time performance using a multi-core implementation, which is 240 seconds when $L = 100,000$. DisCCA-MBI uses slightly more than twice of the time used by DisCCA-GP, i.e., 596 seconds, under the same settings to capture 95% of the sum-of-correlations, but also only uses half of the communication overhead that DisCCA-GP uses. LasCCA is slower compared to the multi-core algorithms when $L \geq 50,000$ since it only uses one core, but it captures more sum-of-correlations compared to that of DisCCA-GP and DisCCA-MBI.

Table 3 shows the captured correlations of a larger problem (i.e., $L = 120,000$ and $M_i = 100,000$) for various sparsity levels of $\boldsymbol{X}_i$. We also set $I = 5$ and $K = 5$. In this simulation, CSR-BCD cannot start to run; it runs

TABLE 3
The average correlations captured after 20 iterations under different ρ's. $L = 120,000$; $M = 100,000$; $K = 5$; $I = 5$; "†" means "out-of-memory"; $v_{\text{opt}} = 100$.

Algorithm	measure	ρ ($\times 10^{-5}$)				
		0.5	0.75	1	2.5	5
LasCCA	corr. cap.	**99.95**	**99.93**	**99.91**	**99.76**	**99.62**
	95time (sec.)	16.93	17.14	18.00	23.03	29.14
DisCCA-MBI	corr. cap.	99.79	99.73	99.69	99.36	99.06
	95time (sec.)	11.31	11.49	11.91	14.01	21.14
DisCCA-GP	corr. cap.	99.87	99.82	99.79	99.52	99.27
	95time (sec.)	**3.35**	**3.40**	**3.56**	**4.34**	**5.74**
CSR-BCD	corr. cap.	†	†	†	†	†
	95time (sec.)	†	†	†	†	†
MVLSA (50)	corr. cap.	60.39	55.60	45.57	46.89	36.18
	time	31.88	37.57	38.09	51.19	58.12
MVLSA (250)	corr. cap.	99.66	99.00	98.54	89.04	88.74
	time	267.47	454.18	469.46	593.91	530.42

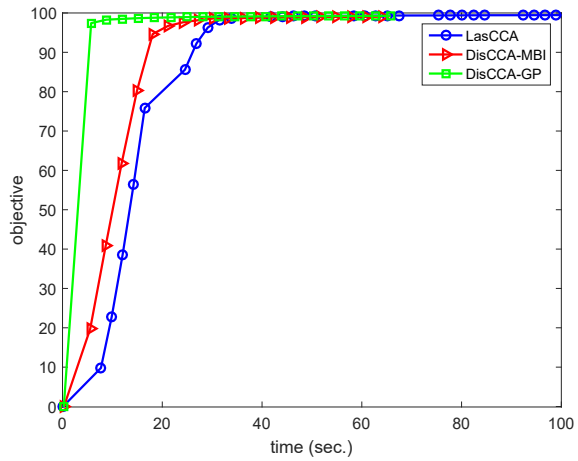


Fig. 3. Captured correlations versus iterations. $L = 120,000$; $M = 100,000$; $K = 5$; $I = 5$; $\rho = 5 \times 10^{-5}$; $v_{\text{opt}} = 100$.

out-of-memory for all the ρ's since it destroys sparsity at the very first step. The proposed algorithms, on the other hand, work well for all the different ρ's: all three algorithms give objective values that are larger than 99 (recall that $v_{\text{opt}} = 100$) for all ρ's. The performance of MVLSA (250) is also reasonable when $\rho \leq 10^{-5}$, but rapidly decreases along with ρ increasing. We also notice that although MVLSA makes the compromise to work with reduced-dimension (approximated) data via rank truncation, it still costs significantly much more time relative to the proposed algorithms.

Table 3 also shows the corresponding 95time performance of the algorithms. First, one can see that both of the multi-core algorithms, i.e., DisCCA-MBI and DisCCA-GP still outperform LasCCA in terms of runtime. Second, the runtimes of the proposed algorithms grows (roughly) linearly with ρ, which is consistent with our analysis – the major computations in the proposed algorithms have complexity order of $\mathcal{O}(\text{nnz}(\boldsymbol{X}_i)K)$ flops. Fig. 3 gives a clear picture of how the objective values of the algorithms evolve with time when $\rho = 5 \times 10^{-5}$. Here, we plot the objective values of the algorithms after *every update of any* $\boldsymbol{G}_i$. One can see that the captured sum-of-correlations of DisCCA-GP grows much faster compared to that of the other two – since DisCCA-GP updates all the $\boldsymbol{G}_i$'s simultaneously. Also, we can see that the objective of DisCCA-MBI climbs faster relative to LasCCA. The reason is that DisCCA-MBI updates the $\boldsymbol{G}_i$ that gives the maximal increase of the objective, while LasCCA updates every $\boldsymbol{G}_i$ cyclically.

6.2 Real Experiment: Reuters RCV2 Corpus

• **Data Description.** In this subsection, we test the proposed algorithm on the Reuters Corpora Volume 2 (RCV2) Multilingual document data. Specifically, we employ the version provided by Amini *et al.* in [42]. The dataset contains five sets of documents. The five subsets are collections of documents originally written in English, German, French, Spanish, and Italian, respectively, and then translated to the other four languages. Therefore, each set of documents with translations has five views – the original documents and their translations. We denote the five datasets as "EN", "FR", "GR", "IT", and "SP", respectively, according to the original language of the documents. Every view $\boldsymbol{X}_i$ is a "document $\times$ vocabulary" matrix; i.e., every document is defined in the features spaces that are vocabularies of different languages. The sizes of views $\boldsymbol{X}_i$ for $i = 1, \ldots, I$ are determined as follows: If the vocabulary size of a language is larger than the number of L, the number of documents in the original language, then we set the corresponding $M_i = 0.95 \times L$ by dropping the columns of $\boldsymbol{X}_i$ with small norms; otherwise, we keep the default vocabulary size. We summarize of the sizes of the views in Table 4. The features of the views are the standard term-frequency-inverse-document-frequency (tf-idf) representation of documents that is widely used in text analytics.

• **Evaluation.** Since the views are translated version of different documents, it is believed that the views are perfectly correlated at some latent space. Thus, we evaluate the performance of the algorithms as in the synthetic data experiments by observing the sum-of-correlations captured by $\boldsymbol{X}_i\boldsymbol{Q}_i$. We also observe the 95time performance.

• **Results.** Table 5 shows the results that are obtained on RCV2 using the algorithms. Here, we set $K = 5$ and thus the optimal objective value is 100 if the views share a common latent representation as in the synthetic experiment case. One can see that for the five different experiments, LasCCA gives the best captured sum of correlations which are larger than 99 – this verifies our guess that the views of the documents share an (almost) common latent space. DisCCA-GP and DisCCA-MBI also both give good objective values, and the former performs slightly better. The baseline algorithm MVLSA does not perform as well as the proposed algorithms. In terms of the runtime performance, one can see that DisCCA-GP is the fastest as before. Similar results can also be observed in Table 6, where $K = 10$ is used (and thus the optimal objective value is 200).

TABLE 4
Summary of the RCV2 dataset.

Original Laguage	L (docs)	vocab. size of translated doc.				
		EN	SP	IT	GR	FR
EN	12,342	10,890	11,360	10,392	11,724	11,724
SP	18,758	17,820	9,588	12,089	17,656	17,820
IT	29,953	14,028	9,300	11,136	28,455	16,906
GR	26,648	15,062	9,862	11,920	18,071	24,284
FR	24,039	11,460	8,709	15,005	14,592	15,382

6.3 Real Experiment: Multilingual Word Embedding

• **Data Description.** We also test our algorithms on a large-scale parallel corpus of multilingual co-occurrence data. In

TABLE 5
The captured correlations and 95time's of the algorithms on the RCV2 dataset. $K = 5$; $I = 5$.

Algorithm	measure	original language				
		EN	SP	GR	FR	IT
LasCCA	corr. cap.	**99.68**	**99.63**	**99.41**	**99.60**	**99.40**
	95time (sec.)	6.92	15.77	23.70	21.48	12.61
DisCCA-MBI	corr. cap.	99.13	98.35	97.67	98.65	96.12
	95time (sec.)	6.64	13.94	19.66	18.39	14.09
DisCCA-GP	corr. cap.	99.46	99.27	**99.41**	99.42	99.19
	95time (sec.)	**2.62**	**5.55**	**8.41**	**7.61**	7.34
MVLSA (50)	corr. cap.	78.65	77.37	77.02	77.02	76.47
	time (sec.)	3.64	6.52	10.22	9.04	**6.76**
MVLSA (250)	corr. cap.	79.88	78.48	78.60	78.89	78.52
	time (sec.)	42.36	69.96	107.83	93.92	73.35

TABLE 6
The captured correlations and 95time's of the algorithms on the RCV2 dataset. $K = 10$; $I = 5$.

Algorithm	measure	original language				
		EN	SP	GR	FR	IT
LasCCA	corr. cap.	**198.88**	**198.53**	**198.42**	**198.44**	**198.45**
	95time	14.24	32.37	48.77	39.39	25.31
DisCCA-MBI	corr. cap.	197.65	191.86	187.64	193.98	194.82
	95time	10.05	25.51	–	33.28	21.25
DisCCA-GP	corr. cap.	198.60	197.77	197.85	197.73	197.79
	95time	3.78	12.76	18.87	16.17	10.49
MVLSA (50)	corr. cap.	171.74	167.45	164.99	165.65	164.14
	time	**3.45**	**6.96**	**10.55**	**9.24**	**6.93**
MVLSA (250)	cor. cap.	176.45	172.43	172.64	173.13	172.22
	time	41.97	70.08	108.61	96.25	72.98

particular, we use the data that were originally introduced in [43] and can be downloaded from http://wordvectors.org/eacl14-data.tar. We have $I = 4$ views, which are constructed from English, Spanish, German and French, respectively. In a nutshell, the raw data contain a matrix per language that records the Point-wise Mutual Information (PMI) of the word co-occurrence for that language. We view the rows of the matrix as our data points and the columns as features. We use English words to form our first view, $\boldsymbol{X}_1$, which contains $L = 180,834$ words and each word is defined by $M_1 = 130,000$ features. Note that we set $M_1 = \ldots = M_4 = 130,000$, and the features correspond to the columns in each view that have the M_i highest energies. There are 1.21% non-zero entries out of the 2.3508×10^{10} entries of $\boldsymbol{X}_1$. Using dictionaries, we pick out the corresponding words in French, German and Spanish to form $\boldsymbol{X}_2$, $\boldsymbol{X}_3$ and $\boldsymbol{X}_4$, respectively. For those English terms which do not have a translation in $\boldsymbol{X}_i$ for $i > 1$, we simply let the corresponding row of $\boldsymbol{X}_i$ be zero. Consequently, we have X_i for $i = 1, \ldots, 4$ all being $180,834 \times 130,000$ sparse matrices.

• **Evaluation.** We evaluate the quality of the output embeddings of the English words by different algorithms. Specifically, we employ the evaluation tool provided at wordvectors.org [44], which automatically evaluates the output low-dimensional representations of the English words on several word embedding tasks by comparing the algorithm-learned embeddings with the judgment of humans. The outputs are scores between zero and one, and a score equal to one means a perfect alignment between the learned result and human judgment. In all the real experiments, we evaluate the results on the first 11 tasks of wordvectors.org.

TABLE 7
English word embedding evaluation. $K = 50$.

TASK	DisCCA-MBI	DisCCA-GP	LasCCA	MVLSA	SVD
EN-WS-353-REL	0.5511	0.5568	**0.5777**	0.5056	0.5096
EN-WS-353-SIM	**0.7021**	0.6996	0.6848	0.6731	0.6381
EN-WS-353-ALL	0.6132	0.6159	**0.6227**	0.5796	0.5679
EN-MTurk-287	0.6468	0.6479	0.6504	**0.6564**	0.6484
EN-YP-130	**0.4409**	0.4371	0.4158	0.4314	0.3673
EN-RW-STANFORD	0.426	0.4269	**0.431**	0.4	0.3978
EN-MEN-TR-3k	0.6762	**0.6766**	0.6739	0.6357	0.6626
EN-SIMLEX-999	0.3818	0.3815	0.382	**0.4198**	0.3091
EN-MTurk-771	**0.6028**	0.6012	0.5996	0.5797	0.5396
EN-MC-30	0.6394	0.6385	**0.6496**	0.5713	0.5655
EN-RG-65	0.53	0.5287	**0.5472**	0.5015	0.5127
Average	0.5680	0.5646	**0.5688**	0.5453	0.5206
Median	**0.6028**	0.6012	0.5996	0.5713	0.5396

• **Results.** Table 7 shows the scores given by DisCCA-MBI, DisCCA-GP, LasCCA and MVLSA when $K = 50$. We run the proposed algorithms for 10 iterations and observe the results. For MVLSA, we follow the procedure in [7] to truncate the rank of the views to 640 as pre-processing. We use the results given by MLSA to initialize our algorithms. We also present the result of applying SVD to $\boldsymbol{X}_1$, i.e., English, as another baseline. One can see that the proposed algorithms outperform MLSA on 9 out of 11 tasks. DisCCA and LasCCA perform similarly over all the tasks. Another observation is that using multiple languages and GCCA do help give better word embeddings in this case. Compared to SVD that only uses one language, DisCCA, LasCCA and MVLSA all give higher scores. Table 8 shows similar results when $K = 100$. One can see that the proposed algorithms give the best scores over 7 tasks, while MVLSA and SVD gives the best scores for one and three tasks, respectively.

In Table 9, we increase K to 300. The first observation is that all the algorithms work better under such K, which is reasonable since more dimensions are used. One can see that DisCCA-MBI gives the best evaluation scores on 8 tasks. SVD also gives good scores and performs the best on 2 tasks. LasCCA and DisCCA-GP work reasonably well but slightly worse relative to DisCCA-MBI and SVD. This might be because on such a large-scale problem, DisCCA-MBI's greedy update strategy helps. MVLSA is not very promising in such case. Our understanding is that both truncating the rank of the views and enforcing a common $\boldsymbol{G}$ of the views (cf. Eq. (23)) are the reasons for observing such performance degradation. The results in this subsection are encouraging, since they clearly show that using the considered GCCA formulation and the proposed algorithms, the performance of large-scale multiview data analytics has been improved.

7 Conclusion

In this paper, we investigated the problem of computing canonical components of large-scale sparse multiview data. A judicious equivalent reformulation of the SUMCOR GCCA problem was proposed. Under this reformulation, three algorithms based on different update strategies were proposed: LasCCA sequentially computes and updates the canonical components of the views; DisCCA-MBI computes them in a greedy fashion; and DisCCA-GP updates the canonical components in parallel. The algorithms are highly scalable when dealing with SUMCOR GCCA, and offer

TABLE 8
English word embedding evaluation. $K = 100$.

TASK	DisCCA-MBI	DisCCA-GP	LasCCA	MVLSA	SVD
EN-WS-353-REL	**0.5915**	0.5722	0.5762	0.5462	0.5672
EN-WS-353-SIM	**0.7275**	0.7102	0.7026	0.6936	0.6777
EN-WS-353-ALL	**0.6465**	0.6343	0.6353	0.6051	0.6188
EN-MTurk-287	0.6311	0.6413	0.6365	**0.6763**	0.6068
EN-YP-130	**0.5201**	0.5041	0.4726	0.448	0.4363
EN-RW-STANFORD	0.434	0.4446	**0.4464**	0.4361	0.4408
EN-MEN-TR-3k	0.7183	0.7153	0.7126	0.678	**0.7252**
EN-SIMLEX-999	0.3822	0.4016	**0.4057**	0.419	0.3438
EN-MTurk-771	0.6245	**0.6254**	0.6187	0.5919	0.5893
EN-MC-30	0.6901	0.7035	0.697	0.6198	**0.7435**
EN-RG-65	0.6244	0.6475	0.6505	0.592	**0.6872**
Average	0.5991	**0.6**	0.5958	0.5733	0.5851
Median	0.6245	0.6343	**0.6353**	0.592	0.6068

TABLE 9
English word embedding evaluation. $K = 300$.

TASK	DisCCA-MBI	DisCCA-GP	LasCCA	MVLSA	SVD
EN-WS-353-REL	**0.6548**	0.5858	0.5887	0.5374	0.6415
EN-WS-353-SIM	**0.7494**	0.7199	0.7118	0.6901	0.7417
EN-WS-353-ALL	**0.6888**	0.6488	0.6509	0.611	0.6804
EN-MTurk-287	**0.6131**	0.5441	0.568	0.5414	0.548
EN-YP-130	0.546	**0.5797**	0.5402	0.4781	0.4987
EN-RW-STANFORD	**0.4741**	0.4642	0.4547	0.4484	0.4606
EN-MEN-TR-3k	0.7615	0.747	0.7369	0.7079	**0.7668**
EN-SIMLEX-999	0.4259	**0.4651**	0.456	0.4354	0.4123
EN-MTurk-771	**0.6796**	0.6534	0.63	0.6101	0.6481
EN-MC-30	0.8134	0.7889	0.7791	0.7422	**0.8472**
EN-RG-65	0.7605	0.7354	0.7255	**0.7635**	0.7214
Average	**0.6516**	0.6302	0.622	0.5969	0.6333
Median	**0.6796**	0.6488	0.63	0.6101	0.6481

different options for practitioners based on trade-offs between computational resource, time, and communication overhead. DisCCA-MBI and DisCCA-GP are also the first distributed algorithms for large-scale GCCA. Convergence properties were studied. Simulations and real experiments show that all of the proposed algorithms scale well to large-size problems and give promising results when applied to real datasets.

REFERENCES

[1] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936.

[2] S. Bickel and T. Scheffer, "Multi-view clustering." in *Proc. ICDM 2004*, vol. 4, 2004, pp. 19–26.

[3] Y. Cui, X. Z. Fern, and J. G. Dy, "Non-redundant multi-view clustering via orthogonalization," in *Proc. ICDM 2007*, 2007, pp. 133–142.

[4] S. M. Kakade and D. P. Foster, "Multi-view regression via canonical correlation analysis," in *Learning Theory*. Springer, 2007, pp. 82–96.

[5] I. Rustandi, "Predictive fMRI analysis for multiple subjects and multiple studies (thesis)," Ph.D. dissertation, Carnegie Mellon University, 2010.

[6] P. Dhillon, D. P. Foster, and L. H. Ungar, "Multi-view learning of word embeddings via CCA," in *NIPS*, 2011, pp. 199–207.

[7] P. Rastogi, B. Van Durme, and R. Arora, "Multiview LSA: Representation learning via generalized cca," in *NAACL*, 2015.

[8] A. O'Sullivan, N. M. Adams, and I. Rezek, "Canonical correlation analysis for detecting changes in network structure," in *Proc. ICDM 2012*, 2012, pp. 250–257.

[9] Z. Ding and Y. Fu, "Low-rank common subspace for multi-view learning," in *Proc. ICDM 2014*, 2014, pp. 110–119.

[10] R. Arora and K. Livescu, "Multi-view learning with supervision for transformed bottleneck features," in *Proc. ICASSP 2014*, 2014, pp. 2499–2503.

[11] Q. Zhang, L. Zhang, B. Du, W. Zheng, W. Bian, and D. Tao, "MMFE: Multitask multiview feature embedding," in *ICDM 2015*. IEEE, 2015, pp. 1105–1110.

[12] J. D. Carroll, "Generalization of canonical correlation analysis to three or more sets of variables," in *Proceedings of the 76th annual convention of the American Psychological Association*, vol. 3, 1968, pp. 227–228.

[13] A. Tenenhaus and M. Tenenhaus, "Regularized generalized canonical correlation analysis," *Psychometrika*, vol. 76, no. 2, pp. 257–284, 2011.

[14] A. Tenenhaus, C. Philippe, V. Guillemot, K.-A. Le Cao, J. Grill, and V. Frouin, "Variable selection for generalized canonical correlation analysis," *Biostatistics*, p. kxu001, 2014.

[15] J. Rupnik, P. Skraba, J. Shawe-Taylor, and S. Guettes, "A comparison of relaxations of multiset cannonical correlation analysis and applications," *arXiv preprint arXiv:1302.0974*, 2013.

[16] I. Rustandi, M. A. Just, and T. Mitchell, "Integrating multiple-study multiple-subject fmri datasets using canonical correlation analysis," in *Proc. MICCAI 2009 Workshop*, 2009.

[17] M. T. Chu and J. L. Watterson, "On a multivariate eigenvalue problem, part i: Algebraic theory and a power method," *SIAM Journal on scientific computing*, vol. 14, no. 5, pp. 1089–1106, 1993.

[18] J. R. Kettenring, "Canonical analysis of several sets of variables," *Biometrika*, vol. 58, no. 3, pp. 433–451, 1971.

[19] P. Horst, "Relations among m sets of measures," *Psychometrika*, vol. 26, no. 2, pp. 129–149, 1961.

[20] J. Vía, I. Santamaría, and J. Pérez, "A learning algorithm for adaptive canonical correlation analysis of several data sets," *Neural Networks*, vol. 20, no. 1, pp. 139–152, 2007.

[21] L.-H. Zhang, L.-Z. Liao, and L.-M. Sun, "Towards the global solution of the maximal correlation problem," *Journal of Global Optimization*, vol. 49, no. 1, pp. 91–107, 2011.

[22] X. Fu, K. Huang, M. Hong, N. D. Sidiropoulos, and A. M.-C. So, "Scalable and flexible max-var generalized canonical correlation analysis via alternating optimization," *IEEE Trans. Signal Process.*, to appear, 2017.

[23] Y. Lu and D. P. Foster, "Large scale canonical correlation analysis with iterative least squares," in *NIPS*, 2014, pp. 91–99.

[24] Z. Ma, Y. Lu, and D. Foster, "Finding linear structure in large datasets with scalable canonical correlation analysis," in *ICML 2015*, 2015.

[25] R. Ge, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford, "Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis," *arXiv preprint arXiv:1604.03930*, 2016.

[26] Z. Allen-Zhu and Y. Li, "Doubly accelerated methods for faster cca and generalized eigendecomposition," *arXiv preprint arXiv:1607.06017*, 2016.

[27] W. Wang, J. Wang, and N. Srebro, "Globally convergent stochastic optimization for canonical correlation analysis," *arXiv preprint arXiv:1604.01870*, 2016.

[28] X. Fu, K. Huang, E. Papalexakis, H. Song, P. Talukdar, N. D. Sidiropoulos, C. Faloutsos, and T. Mitchell, "Efficient and distributed algorithms for large-scale generalized correlation analysis," in *Proc. ICDM 2016*. IEEE, 2016.

[29] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[30] P. Schönemann, "A generalized solution of the orthogonal Procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[31] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[32] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[33] Y. Nesterov, *Introductory lectures on convex optimization*. Springer Science & Business Media, 2004, vol. 87.

[34] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[35] G. H. Golub and C. F. V. Loan., *Matrix Computations*. The Johns Hopkins University Press, 1996.

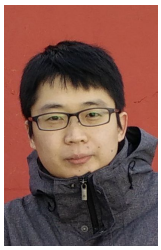
[36] E. Van der Burg, *Nonlinear canonical correlation and some related techniques*. DSWO press, 1988, vol. 11.

[37] G. H. Golub and H. Zha, *The canonical correlations of matrix pairs and their numerical computation*. Springer, 1995.

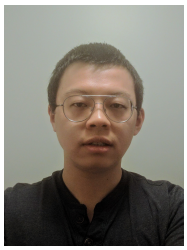
[38] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[39] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal

processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, 2016.
[40] B. Chen, S. He, Z. Li, and S. Zhang, "Maximum block improvement and polynomial optimization," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 87–107, 2012.
[41] J. Tranter, N. D. Sidiropoulos, X. Fu, and A. Swami, "Fast unit-modulus least squares with applications in beamforming," *IEEE Transactions on Signal Processing*, vol. 65, no. 11, pp. 2875–2887, 2017.
[42] M.-R. Amini, N. Usunier, and C. Goutte, "Learning from multiple partially observed views - an application to multilingual text categorization," in *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, 2009, pp. 28–36. [Online]. Available: http://books.nips.cc/papers/files/nips22/NIPS2009_0688.pdf
[43] M. Faruqui and C. Dyer, "Improving vector space word representations using multilingual correlation." Association for Computational Linguistics, 2014.
[44] ——, "Community evaluation and exchange of word vectors at wordvectors.org," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, USA: Association for Computational Linguistics, June 2014.

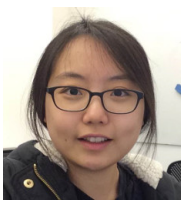


Xiao Fu (S'12-M'15) is an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, United States. He received his Ph.D. degree in Electronic Engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, 2014. He was a Postdoctoral Associate in the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, United States, from 2014-2017. His research interests include the broad area of signal processing and machine learning. He received a Best Student Paper Award at ICASSP 2014, and co-authored a Best Student Paper Award at IEEE CAMSAP 2015.



Kejun Huang (S'13) received the B.Eng. in Communication Engineering from Nanjing University of Information Science and Technology, Nanjing, China in 2010. In 2016, he received his Ph.D. degree in Electrical and Computer Engineering from University of Minnesota. His research interests include signal processing, machine learning, and data analytics. His current research focuses on identifiability, algorithms, and performance analysis for factor analysis of big matrix and tensor data.

Evangelos Papalexakis is an Assistant Professor in the Computer Science Department at the University of California, Riverside. He earned his Diploma and M.Sc. in Electronic and Computer Engineering at the Technical University of Crete, Chania, Greece, and his Ph.D. in Computer Science at Carnegie Mellon University, in 2010, 2011, and 2016 respectively. He has considerable experience in tensor decompositions, data mining, computing, and signal processing. He is currently interested in discovering how knowledge and information is expressed and stored in the brain, through analyzing brain scan data coupled with external information. He is also interested in anomaly detection on very large graphs, especially when temporal or multi-view information is present. He has spent two summers as an intern at Microsoft Research Silicon Valley, working at the Search Labs and the Interaction & Intent group.



Hyun Ah Song is a Ph.D. student in the Machine Learning Department at Carnegie Mellon University, USA. She received B.S. in Environmental Science and Engineering at Ewha Womans University, South Korea in 2011, and M.S. in Electrical Engineering at Korea Advanced Institute of Science and Technology (KAIST) in 2013.

Partha Pratim Talukdar was born in Guwahati, India in 1981. He received his B.E.(Hons) Computer Science from BITS, Pilani in 2003, and the MSE & PhD in Computer and Information Science (CIS) from the University of Pennsylvania, Philadelphia in 2005 and 2010, respectively. Since 2014, he has been with the Indian Institute of Science, Bangalore as an Assistant Professor. Previously, he was a Postdoctoral Fellow in the Machine Learning Department at Carnegie Mellon University, working with Tom Mitchell on the NELL project. Dr. Talukdar is broadly interested in Machine Learning, Natural Language Processing, and Cognitive Neuroscience, with particular interest in large-scale learning and inference. He is a recipient of IBM Faculty Award, Googles Focused Research Award, and Accenture Open Innovation Award. He is a co-author of a book on Graph-based Semi-Supervised Learning published by Morgan Claypool Publishers.

Nicholas D. Sidiropoulos (F'09) is a Professor and the Chair of the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, USA. His research spans topics in signal processing theory and algorithms, optimization, communications, and factor analysis - with a long-term interest in tensor decomposition and its applications. His current focus is primarily on signal and tensor analytics for learning from big data. He received the NSF/CAREER award in 1998, and the IEEE Signal Processing (SP) Society Best Paper Award in 2001, 2007, and 2011. He served as IEEE SP Society Distinguished Lecturer (2008-2009), and as Chair of the IEEE Signal Processing for Communications and Networking Technical Committee (2007-2008). He received the 2010 IEEE SP Society Meritorious Service Award, and the 2013 Distinguished Alumni Award from the Dept. of ECE, University of Maryland. He is a Fellow of IEEE (2009) and a Fellow of EURASIP (2014).

Christos Faloutos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), 21 "best paper" awards (including 3 "test of time" awards), and four teaching awards. Five of his advisees have won KDD or SCS dissertation awards. He is an ACM Fellow, he has served as a member of the executive committee of SIGKDD; he has published over 300 refereed articles, 17 book chapters and two monographs. He holds nine patents and he has given over 40 tutorials and over 20 invited distinguished lectures. He has a long-term interest in tensor decompositions and their practical applications in data mining, having published many well-appreciated papers in the area. His broad research interests include large-scale data mining with emphasis on graphs and time sequences; anomaly detection, tensors, and fractals.

Tom M. Mitchell founded the Machine Learning Department at Carnegie Mellon University, where he is the E. Fredkin University Professor. His research uses machine learning to develop computers that are learning to read the web, and uses brain imaging to study how the human brain understands what it reads. Mitchell is a member of the U.S. National Academy of Engineering, a Fellow of the American Association for the Advancement of Science (AAAS), and a Fellow and Past President of the Association for the Advancement of Artificial Intelligence (AAAI). He believes the field of machine learning will be the fastest growing branch of computer science during the 21st century.