COMMUNICATION-EFFICIENT DISTRIBUTED MAX-VAR GENERALIZED CCA VIA ERROR FEEDBACK-ASSISTED QUANTIZATION

Sagar Shrestha and Xiao Fu

School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331, USA
`(shressag, xiao.fu)@oregonstate.edu`

ABSTRACT

Generalized canonical correlation analysis (GCCA) aims to learn common low-dimensional representations from multiple "views" of the data (e.g., audio and video of the same event). In the era of big data, GCCA computation encounters many new challenges. In particular, distributed optimization for GCCA—which is well-motivated in applications like internet of things and parallel computing—may incur prohibitively high communication costs. To address this challenge, this work proposes a communication-efficient distributed GCCA algorithm under the popular MAX-VAR GCCA paradigm. A quantization strategy for information exchange among the computing agents is employed in the proposed algorithm. It is observed that our design, leveraging the idea of error feedback-based quantization, can reduce communication cost by at least 90% while maintaining essentially the same GCCA performance as the unquantized version. Furthermore, the proposed method is guaranteed to converge to a neighborhood of the optimal solution in a geometric rate—even under aggressive quantization. The effectiveness of our method is demonstrated using both synthetic and real data experiments.

***Index Terms*—** Distributed GCCA, communication-efficient learning, convergence analysis.

1. INTRODUCTION

The aim of *canonical correlation analysis* (CCA) [1] is to learn a common representation from multiple distinct views of the same entity (e.g., the image and audio of a car are considered two views of the entity car). Classic CCA focuses on two-view cases. However, it is intuitively appealing to leverage more than two views when available, because the learnt representation can be free of irrelevant details not characteristic of the entity. This has strong theoretical and empirical support [2]. Hence, the so-called *generalized canonical correlation analysis* (GCCA) paradigms have also attracted much attention for decades [3–7]. (G)CCA have found numerous applications in signal processing and machine learning, e.g., blind source separation [8, 9], latent semantic analysis [10], direction-of-arrival estimation [11], clustering [12], just to name a few.

When many views of data are acquired by and stored in different agents (or, *nodes*), computing GCCA in a distributed manner is well-motivated. Consider a motivating scenario in which multiple social media platforms have different views of the users' data, e.g., images, tweets, messages, videos, etc. It is tempting to learn representations of the multi-platform users using data from these platforms simultaneously, since such platform-invariant common representation may be more effective in enhancing performance of downstream tasks, e.g., product recommendation. However, exchanging user data to perform centralized GCCA may not be allowed, and thus distributed optimization becomes a natural choice. Such considerations have led to the development of a series of distributed GCCA algorithms, e.g., [4–8, 13]. These algorithms are effective in terms of using multiple views in parallel for accelerated optimization. However, since each iteration of the algorithms needs to exchange a large amount of information among the computing agents, such algorithms may not be feasible for overhead-restricted scenarios, e.g., wireless sensor networks and internet-of-things. In fact, even for parallel computing with multiple cores within the same computing facility, large overhead may slow down the overall optimization process significantly. Therefore, reducing communication overhead of distributed GCCA is well-motivated, but effective solutions have been elusive.

To address these challenges, we propose a communication-efficient distributed algorithm for GCCA. We consider a case where each node holds a view of the data and seek a common representation using the MAX-VAR formulation of GCCA. We leverage the idea of exchange information compression/quantization from recent developments in distributed gradient descent methods, e.g., [14–16],. Unlike the existing compression frameworks, we do not compress the gradients, as our framework is not a distributed gradient descent algorithm. Instead, we propose to compress necessary information for performing an alternating optimization algorithm, using the so-called error feedback mechanism that was originally used for gradient compression [15, 17]. This way, we are able to consistently achieve 90% reduction in both downlink and uplink communications—and the performance of GCCA tasks are essentially unchanged relative to the uncompressed version. Our analysis shows that, despite aggressive quantization and stochastic approximations, the proposed method converges to a neighborhood of the optimal solution at a geometric rate. The analysis is nontrivial, since existing analytical tools of gradient compression do not cover our case. We provide both synthetic and real data experiments that showcase the communication efficiency of the proposed method.

2. PROBLEM STATEMENT

In this work, we propose a communication-efficient distributed GCCA algorithm. Our focus lies in a popular formulation of GCCA, known as the *maximal variation* (MAX-VAR) GCCA. However, the idea has the potential to cover other GCCA formulations, e.g., the

This work is supported in part by the National Science Foundation (NSF) under Project NSF ECCS-1808159 and NSF CNS-2003082, and by a gift from Intel through the MLWiNS Program.

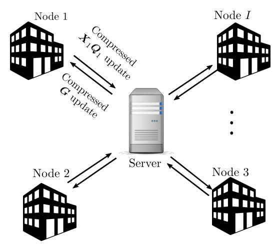

Fig. 1. Illustration of the considered scenario.

sum-of-correlations (SUMCORR) GCCA [5,6].

Consider I views, $\boldsymbol{X}_i \in \mathbb{R}^{J\times N_i}$, $i \in [I], [I] = \{1,\dots,I\}$, corresponding to different representations of J entities. Let $\boldsymbol{X}_i(j,:)$ denote the jth data point of the ith view. The goal of GCCA is to find $K << N_i, i \in [I]$ maximally correlated components among the views via linear transformations. That is, GCCA seeks $\boldsymbol{Q}_i \in \mathbb{R}^{N_i\times K}$'s such that $\boldsymbol{Z}_i(j,:) = \boldsymbol{X}_i(j,:)\boldsymbol{Q}_i$'s that are highly correlated across views. Such $\boldsymbol{Z}_i(j,:)$ can then serve as the low-dimensional and more "informative" representation of data entity j, since the information contained in $\boldsymbol{Z}_i(j,:)$ is view-invariant.

To this end, we consider the MAX-VAR formulation of GCCA [18]. Under MAX-VAR, the optimization criterion seeks to find a common latent representation for all the views, i.e.

$$\underset{\boldsymbol{G},\{\boldsymbol{Q}_i\}_{i=1}^{I}}{\text{minimize}} \ \sum_{i=1}^{I} \underbrace{\frac{1}{2}\left\|\boldsymbol{X}_i\boldsymbol{Q}_i - \boldsymbol{G}\right\|_{\text{F}}^2}_{f_i(\boldsymbol{Q}_i,\boldsymbol{G})} \quad \text{subject to } \ \boldsymbol{G}^\top\boldsymbol{G} = \boldsymbol{I} \tag{1}$$

The optimal solution of (1) can be found by taking the K largest eigenvectors of $\sum_{i=1}^{I}\boldsymbol{X}_i\boldsymbol{X}_i^\dagger$ [4]. However, simply taking eigendecomposition in a naive way may encounter scalability issues. Therefore, several methods have been proposed to tackle this challenge [4,10]. In particular, the method in [4] naturally features a distributed implementation. The algorithm updates $\boldsymbol{Q}_i$ and $\boldsymbol{G}$ in an alternating manner. The algorithm stores view i at the computing agent/node i and only uploads $\boldsymbol{X}_i\boldsymbol{Q}_i$ to a central node in each iteration. The central node broadcasts the updated $\boldsymbol{G}$ to every node. This way, the data $\boldsymbol{X}_i$ does not need to be exchanged among the nodes. The exchanged information has a size of $O(JK)$, which is also much smaller than that of the views, i.e., $O(JN_i)$, when $K \ll N_i$.

The distributed method in [4] is interesting and useful, but still faces some major challenges. The most prominent among them is the communication overhead. To be more precise, when $J \approx 1,000,0000$ and $K \approx 200$, even exchanging JK double precision real numbers takes a large amount of communication overhead ($\geq$ 1.4GB) in each iteration—let alone the fact that the algorithm may run for tens or even hundreds of iterations.

3. PROPOSED APPROACH

In the considered scenario, there are I computing agents (nodes) holding the I views. There is a central server for gathering information. Communication from the server to individual nodes is called *downlink* communication and from a node to the server is called *uplink* communication—see Fig. 1.

3.1. AltMaxVar's Framework in [4]

We follow the method introduced in [4] known as alternating optimization-based MAX-VAR GCCA (`AltMaxVar`) because of its appealing convergence properties and distributed nature. It follows the alternating optimization idea, which is summarized as follows:

$$\boldsymbol{Q}_i^{(r+1)} \leftarrow \arg\min_{\boldsymbol{Q}_i} \frac{1}{2}\left\|\boldsymbol{X}_i\boldsymbol{Q}_i - \boldsymbol{G}^{(r)}\right\|_{\text{F}}^2, \ \forall i \in [I], \tag{2a}$$

$$\boldsymbol{G}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{G}^\top\boldsymbol{G}=\boldsymbol{I}} \sum_{i=1}^{I}\left\|\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)} - \boldsymbol{G}\right\|_{\text{F}}^2. \tag{2b}$$

Exactly solving (2a) may be computationally costly. Approximate solution may involve multiple (stochastic) gradient descent steps, which will be presented later.

For (2b), one can find the optimal solution using the so-called Procrustes Projection [19] as follows:

$$\boldsymbol{G}^{(r+1)} \leftarrow \boldsymbol{U}_{\boldsymbol{Y}^{(r+1)}}\boldsymbol{V}_{\boldsymbol{Y}^{(r+1)}}^\top, \tag{3}$$

where $\boldsymbol{Y}^{(r+1)} = \sum_{i=1}^{I}\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)}$, $\text{svd}(\boldsymbol{Y}^{(r+1)}, '\text{econ}') = \boldsymbol{U}_{\boldsymbol{Y}^{(r+1)}}\boldsymbol{\Sigma}_{\boldsymbol{Y}^{(r+1)}}\boldsymbol{V}_{\boldsymbol{Y}^{(r+1)}}^\top$, and $\text{svd}(\boldsymbol{Y}^{(r+1)}, '\text{econ}')$ is the thin SVD operator.

Notice that the $\boldsymbol{Q}_i$-update is local to the node and the $\boldsymbol{G}$ update only requires the transformed views. Therefore, for distributed implementation, the node computes (2a) and sends $\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)}$ to the server, and the server computes (2b) and broadcasts $\boldsymbol{G}^{(r+1)}$ to the nodes. Each exchanged data is of size $\mathcal{O}(JK)$ which is costly for large scale data, as discussed earlier.

3.2. Proposed: Quantized Communication

The proposed method is illustrated in Algorithm 1. In order to reduce the communication cost of the method in (2), we quantize both the uplink and downlink messages. For example, a node may quantize $\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)}$ before sending it to the server. However, a better approach is to quantize and transmit the change in $\boldsymbol{X}_i\boldsymbol{Q}_i$, i.e., $\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)} - \boldsymbol{X}_i\boldsymbol{Q}_i^{(r)}$. It is because we expect the change in $\boldsymbol{X}_i\boldsymbol{Q}_i$ to converge to zero as the iteration progresses, which results in the quantization error coverging to zero. Compressing such change can still be problematic; it is quite noisy and can hinder convergence guarantees in some cases [17]. A solution is to employ *error feedback* (EF) mechanism which has shown to be essential to convergence in some compressed gradient methods or improve the convergence rate in others [16,17]. Note that we do not use EF to compress gradients, but the information needed to perform alternating optimization between the nodes and the server.

To see how compression and EF is employed in the proposed framework, let the server maintain an estimate of $\boldsymbol{M}_i^{(r+1)} = \boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)}, \forall i$, denoted by $\widehat{\boldsymbol{M}}_i^{(r+1)}$, and the nodes maintain an estimate of $\boldsymbol{G}^{(r+1)}$, denoted by $\widehat{\boldsymbol{G}}^{(r+1)}$. First, both the server and nodes start from the same initial point as follows. Node i randomly initializes $\boldsymbol{Q}_i^{(0)}$ and transmits $\boldsymbol{X}_i\boldsymbol{Q}_i^{(0)}$ to the server using full precision, i.e., without compression. Then the server computes $\boldsymbol{G}^{(0)}$ using (3), and broadcasts $\boldsymbol{G}^{(0)}$ using full precision. This allows us to set the estimates $\widehat{\boldsymbol{M}}_i^{(0)} \leftarrow \boldsymbol{X}_i\boldsymbol{Q}_i^{(0)}$, $\forall i$, $\widehat{\boldsymbol{G}}^{(0)} \leftarrow \boldsymbol{G}^{(0)}$. Note that other initialization methods, such as in [4], can also be employed for the proposed method. In the following, we see how the computation and communication are carried out by the nodes and the server.

Node operations. At iteration r, node i computes $\boldsymbol{Q}_i^{(r+1)}$ using $\widehat{\boldsymbol{G}}^{(r)}$ as follows:

$$\boldsymbol{Q}_i^{(r+1)} \leftarrow \arg\min_{\boldsymbol{Q}_i} f_i\left(\boldsymbol{Q}_i, \widehat{\boldsymbol{G}}^{(r)}\right). \tag{4}$$

Note that the above can be solved by any off-the-shelf least squares solvers, e.g., the conjugate gradient. It also needs not to be solved exactly, but approximated to reduce computational cost (see Sec. 3.4).

Let $\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)}$ denote the information that needs to be transmitted from node i to the server, which is defined as follows:

$$\begin{aligned}\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)} &= \boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)} - \widehat{\boldsymbol{M}}_i^{(r)} \\ &= \underbrace{\boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)} - \boldsymbol{X}_i\boldsymbol{Q}_i^{(r)}}_{\text{exact change}} + \underbrace{\boldsymbol{X}_i\boldsymbol{Q}_i^{(r)} - \widehat{\boldsymbol{M}}_i^{(r)}}_{\text{previous estimation error}}.\end{aligned} \tag{5}$$

This is called error feedback because the estimation error from previous iteration, induced by compression, is fedback along with the current change in $\boldsymbol{X}_i\boldsymbol{Q}_i$. If the node transmits $\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)}$, the server can exactly recover $\boldsymbol{M}_i^{(r+1)} = \boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)} + \widehat{\boldsymbol{M}}_i^{(r)}$. However, to reduce communication overhead, the node quantizes $\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)}$ using a compressor $\boldsymbol{\mathcal{C}}(\cdot) : \mathbb{R}^{J\times K} \to \mathbb{Q}^{J\times K}$, where $\mathbb{Q}^{J\times K} \subset \mathbb{R}^{J\times K}$ is a quantized domain that is a subset of the real domain.

The server receives $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)})$ from node i and updates the estimate $\widehat{\boldsymbol{M}}_i^{(r+1)}$ as $\widehat{\boldsymbol{M}}_i^{(r+1)} \leftarrow \widehat{\boldsymbol{M}}_i^{(r)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)})$. Using the above operation, node i also maintains a copy of $\widehat{\boldsymbol{M}}_i^{(r+1)}$ in order to compute the estimation error in (5).

Server Operations. Server operation proceeds in a similar manner. First, the server computes $\boldsymbol{G}^{(r+1)}$, using $\widehat{\boldsymbol{M}}_i^{(r+1)}$, as follows:

$$\boldsymbol{G}^{(r+1)} \leftarrow \arg\min_{\boldsymbol{G}^\top\boldsymbol{G}=\boldsymbol{I}} \sum_{i=1}^{I} \left\| \widehat{\boldsymbol{M}}_i^{(r+1)} - \boldsymbol{G} \right\|_{\mathrm{F}}^2 .$$

The $\boldsymbol{G}$-update proceeds as $\boldsymbol{G}^{(r+1)} \leftarrow \boldsymbol{U}_{\widehat{\boldsymbol{Y}}^{(r+1)}}\boldsymbol{V}_{\widehat{\boldsymbol{Y}}^{(r+1)}}^\top$, where $\widehat{\boldsymbol{Y}}^{(r+1)} = \sum_{i=1}^{I} \widehat{\boldsymbol{M}}^{(r+1)}$. We define $\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r+1)}$, the information that needs to be exchanged with the nodes, as $\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)} = \boldsymbol{G}^{(r+1)} - \widehat{\boldsymbol{G}}^{(r)}$. The server compresses $\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)}$ and broadcasts $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)})$ to the nodes.

The nodes receive $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)})$ and update the estimate $\widehat{\boldsymbol{G}}$ as $\widehat{\boldsymbol{G}}^{(r+1)} \leftarrow \widehat{\boldsymbol{G}}^{(r)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)})$. Using the above operation, the server also maintains a copy of $\widehat{\boldsymbol{G}}^{(r+1)}$ in order to compute $\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r+1)}$.

3.3. Choice of Compressor

There are more than one option for the compressor $\boldsymbol{\mathcal{C}}(\cdot)$, e.g., quantization based [14, 16], or sparsification based [15, 20]. In this work, we employ a stochastic quantization based compressor introduced in [16] due to its flexibility of accommodating multi-precision quantization, small computation cost, and good performance in practice.

For any data, $\boldsymbol{\Delta} \in \mathbb{R}^{J\times K}$, $\boldsymbol{\Delta} \neq \boldsymbol{0}$, $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta})$ is computed as follows. We first divide the range from 0 to 1 into S intervals, where S corresponds to the number of levels of quantization. For each element $\boldsymbol{\Delta}(j,k)$, we can find an interval $[p/S, (p+1)/S], p \in \{0,\ldots,S-1\}$, such that the normalized value $|\boldsymbol{\Delta}(j,k)|/\|\boldsymbol{\Delta}\|_{\max} \in [p/S, (p+1)/S]$. We define a Bernoulli random variable, $h(\boldsymbol{\Delta}(j,k), S)$, which takes the value p/S with probability $1 - ({|\boldsymbol{\Delta}(j,k)|S}/{\|\boldsymbol{\Delta}\|_{\max}} - p)$, and $(p+1)/S$ otherwise. The result is obtained by unnormalizing $h(\boldsymbol{\Delta}(j,k), S)$ using the sign and magnitude information as $[\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta})]_{j,k} = \|\boldsymbol{\Delta}\|_{\max}\mathrm{sgn}(\boldsymbol{\Delta}(j,k)) \cdot h(\boldsymbol{\Delta}(j,k), s)$, where $\mathrm{sgn}(\cdot)$ is the sign operator.

3.4. Acceleration via SGD at the Nodes

As mentioned in [4, 5, 7], when $\boldsymbol{X}_i$'s are large and sparse, one can use first-order methods such as conjugate gradient or gradient descent to solve the $\boldsymbol{Q}_i$-update problem. However, using the full data (e.g., computing the full gradient) can still be computationally costly when the nodes have limited compute power (such as edge devices), or when the views are large and dense. In this case, the nodes can run *stochastic gradient descent* (SGD) instead. Specifically, at each iteration r, node i runs T steps of SGD as follows:

$$\boldsymbol{Q}_i^{(r,t+1)} \leftarrow \boldsymbol{Q}_i^{(r,t)} - \alpha\widehat{\nabla}_{\boldsymbol{Q}_i} f(\boldsymbol{Q_i}^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)}),$$

where $\widehat{\nabla}_{\boldsymbol{Q}_i} f_i(\boldsymbol{Q_i}^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)})$ is the minibatch-estimated gradient for $f_i(\boldsymbol{Q}_i^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)})$ with respect to a uniformly sampled minibatch at random, α is the step size, and $\boldsymbol{Q}_i^{(r+1)} \leftarrow \boldsymbol{Q}_i^{(r,T)}$ and $\boldsymbol{Q}_i^{(r,0)} \leftarrow \boldsymbol{Q}_i^{(r)}$. The algorithm is termed as *c̲ommunication-qu̲antize̲d MAX-VAR* (`CuteMaxVar`) and summarized in Algorithm 1.

Algorithm 1: `CuteMaxVar`

```
// At the nodes
Initialize $\boldsymbol{Q}_i^{(0)}$, and set $\widehat{\boldsymbol{M}}_i^{(0)} \leftarrow \boldsymbol{X}_i\boldsymbol{Q}_i^{(0)}, \forall i$;
Transmit $\boldsymbol{X}_i\boldsymbol{Q}_i^{(0)}$ to the server using full precision, $\forall i$;
// At the server
$\widehat{\boldsymbol{M}}_i^{(0)} \leftarrow \boldsymbol{X}_i\boldsymbol{Q}_i^{(0)}, \forall i$;
Compute $\boldsymbol{G}^{(0)}$ using (3), and set $\widehat{\boldsymbol{G}}^{(0)} \leftarrow \boldsymbol{G}^{(0)}$;
Broadcast $\boldsymbol{G}^{(0)}$ to the nodes using full precision;
$r \leftarrow 0$ ;
while some stopping criteria is not met do
    // At the Nodes
    if r = 0 then
        $\widehat{\boldsymbol{G}}^{(0)} \leftarrow \boldsymbol{G}^{(0)}$;
    else
        $\widehat{\boldsymbol{G}}^{(r)} \leftarrow \widehat{\boldsymbol{G}}^{(r-1)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r-1)})$ ;
    end
    for $i \leftarrow 1 : I$ do
        for $t \leftarrow 0 : T-1$ do
            Sample a minibatch and compute $\widehat{\nabla}_{\boldsymbol{Q}_i} f(\boldsymbol{Q_i}^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)})$;
            $\boldsymbol{Q}_i^{(r,t+1)} \leftarrow \boldsymbol{Q}_i^{(r,t)} - \alpha\widehat{\nabla}_{\boldsymbol{Q}_i} f(\boldsymbol{Q_i}^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)})$;
        end
        $\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)} \leftarrow \boldsymbol{X}_i\boldsymbol{Q}_i^{(r+1)} - \widehat{\boldsymbol{M}}_i^{(r)}$ ;
        Transmit $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)})$ to the server ;
        $\widehat{\boldsymbol{M}}_i^{(r+1)} \leftarrow \widehat{\boldsymbol{M}}_i^{(r)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)})$ ;   // Node's copy
    end
    // At the Server
    Receive $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)}), \forall i$ from the nodes ;
    $\widehat{\boldsymbol{M}}_i^{(r+1)} \leftarrow \widehat{\boldsymbol{M}}_i^{(r)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{Q}_i}^{(r)})$ ;
    $\boldsymbol{G}^{(r+1)} \leftarrow \boldsymbol{U}_{\boldsymbol{Y}^{(r+1)}}\boldsymbol{V}_{\boldsymbol{Y}^{(r+1)}}^\top$ where $\widehat{\boldsymbol{Y}}^{(r+1)} = \sum_{i=1}^{I} \widehat{\boldsymbol{M}}^{(r+1)}$;
    $\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)} \leftarrow \boldsymbol{G}^{(r+1)} - \widehat{\boldsymbol{G}}^{(r)}$ ;
    Broadcast $\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)})$ to the nodes ;
    $\widehat{\boldsymbol{G}}^{(r+1)} \leftarrow \widehat{\boldsymbol{G}}^{(r)} + \boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}_{\boldsymbol{G}}^{(r)})$ ;   // Server's copy
    $r \leftarrow r + 1$ ;
end
Output: $\boldsymbol{G}^{(r)}, \{\boldsymbol{Q}_i^{(r)}\}_{i=1}^{I}$.
```

3.5. Convergence Analysis

Let $\boldsymbol{P} = \sum_{i=1}^{I} \boldsymbol{X}_i(\boldsymbol{X}_i^\top\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^\top$ and $\boldsymbol{U}_1 = \boldsymbol{U}_{\boldsymbol{P}}(:, 1:K)$, $\boldsymbol{U}_2 = \boldsymbol{U}_{\boldsymbol{P}}(:, K+1:J)$, where $\boldsymbol{U}_{\boldsymbol{P}}\boldsymbol{\Sigma}_{\boldsymbol{P}}\boldsymbol{V}_{\boldsymbol{P}}^\top = \mathrm{svd}(\boldsymbol{P})$. Note that the optimal solution of Problem (1) is $\boldsymbol{G}^\star = \boldsymbol{U}_1$. Therefore we can use the subspace distance metric, $\mathrm{dist}(\mathcal{R}(\boldsymbol{G}^{(r)}), \mathcal{R}(\boldsymbol{U}_1)) = \|\boldsymbol{U}_2^\top\boldsymbol{G}^{(r)}\|_2$ to measure the progress towards the optimal. We establish the following theorem regarding the convergence of `CuteMaxVar`.

Theorem 1 *Assume that there exist a $\delta \in (0,1]$ such that $\mathbb{E}\|\boldsymbol{\mathcal{C}}(\boldsymbol{\Delta}) - \boldsymbol{\Delta}\|_{\mathrm{F}}^2 \leq (1-\delta)\|\boldsymbol{\Delta}\|_{\mathrm{F}}^2$ and $\mathbb{E}\|\widehat{\nabla}_{\boldsymbol{Q}_i} f(\boldsymbol{Q_i}^{(r,t)}, \widehat{\boldsymbol{G}}^{(r)})\|_{\mathrm{F}}^2 \leq \sigma^2$,*

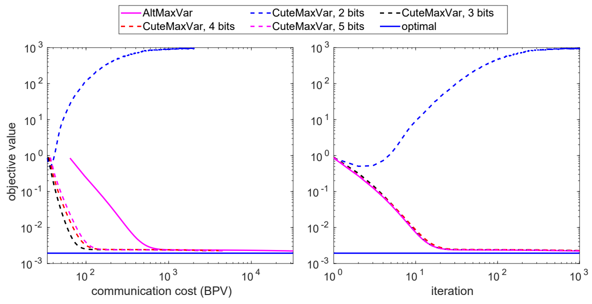

Fig. 2. Communication cost [left] and convergence rate [right] under various compression levels.

where the expectation is over the randomness of compressor and sample seed of SGD, respectively. Let the eigenvalues of $\boldsymbol{P}$ be $\lambda_1, \ldots, \lambda_J$ in descending order. Assume $\lambda_K > \lambda_{K+1}$, the range space of $\boldsymbol{G}^{(0)}$, is not orthogonal to any components in $\mathcal{R}(\boldsymbol{U}_1)$. Further, assume that one solves the $\boldsymbol{Q}_i$-subproblem in each iteration to an accuracy such that $\mathbb{E}\|\boldsymbol{Q}_i^{(r+1)} - \widetilde{\boldsymbol{Q}}_i^{(r+1)}\|_2 \leq \kappa$, where $\widetilde{\boldsymbol{Q}}_i^{(r+1)} = (\boldsymbol{X}_i^\top \boldsymbol{X}_i)^{-1}\boldsymbol{X}_i\boldsymbol{G}^{(r)}$. Then, with probability at least $1-\omega$,

$$\operatorname{dist}\left(\mathcal{R}(\boldsymbol{G}^{(r)}), \mathcal{R}(\boldsymbol{U}_1)\right) \leq \tan(\gamma)\left(\frac{\lambda_{K+1}}{\lambda_K}\right)^r + C,$$

where $\operatorname{dist}\left(\mathcal{R}(\boldsymbol{G}^{(r)}), \mathcal{R}(\boldsymbol{U}_1)\right) = \|\boldsymbol{U}_2^\top \boldsymbol{G}^{(r)}\|_2$ measures the subspace distance, $C = \mathcal{O}({}^{\lambda_K}/_{\lambda_K - \lambda_{K+1}})$ is a constant, and $\omega = \sum_{i=1}^{I} r\|\boldsymbol{X}_i\|_2\kappa + {}^{2Ir\sqrt{(1-\delta)(T\alpha^2\sigma^2+2K)}}/_{\delta}$.

Theorem 1 asserts that, if the $\boldsymbol{Q}_i$-subproblem is solved to a good accuracy and the compressor is sufficiently accurate, `CuteMaxVar` converges to a neighborhood of the optimal with high probability in a geometric rate. Due to page limitations, the proof is put in the online technical report [21] (see [22] for more details). Note that our analysis is different from existing gradient compression methods, since `CuteMaxVar` is not a gradient descent algorithm. Instead, we recast our updates as an inexact and noisy version of the *orthogonal iterations* [23] and draw our conclusions from there.

4. NUMERICAL RESULTS

Synthetic Experiments. The method in [4], `AltMaxVar` can be understood as the unquantized version of `CuteMaxVar`. Therefore, we compare the communication efficiency of `CuteMaxVar` and `AltMaxVar`. We use the optimal solution computed using eigendecomposition as another baseline. All the curves are averaged from 50 Monte Carlo trials. Each view is synthesized using $\boldsymbol{X}_i = \boldsymbol{Z}\boldsymbol{A}_i + \eta\boldsymbol{N}_i$; $\boldsymbol{Z} \in \mathbb{R}^{J\times D}$ is the latent factor with $J \geq D$, $\boldsymbol{A}_i \in \mathbb{R}^{D\times M_i}$ the "mixing matrix", $\eta^2 \in \mathbb{R}$ the noise variance, and $\boldsymbol{N}_i$ the noise. The matrices $\boldsymbol{Z}$, $\boldsymbol{A}$, and $\boldsymbol{N}_i$ are sampled from the standard i.i.d. normal distribution. To measure the communication efficiency, we define bits per variable `BPV` as $\texttt{BPV}(r) = q_{\text{full}} + rq$, where r is the iteration index, q_{full} is the full precision—32 bits in our case—and q represents the number of bits per scalar used by the compressor. Also, we define compression ratio (`CR`) as follows: $\texttt{CR} = 1 - {}^{qR_C}/_{q_{\text{full}}R_A}$, where R_C and R_A are the numbers of iterations required by `CuteMaxVar` and `AltMaxVar` to achieve a certain convergence criterion, respectively.

Fig. 2 shows the communication cost (`BPV`) and convergence speed (in iterations) under various levels of quantization. We set $(J, N, D, K, I, \eta) = (500, 25, 20, 5, 3, 0.01)$. To handle the $\boldsymbol{Q}_i$-subproblem, we use a batch size of 150 for `AltMaxVar` and `CuteMaxVar`, and run the algorithms for T =10 inner iterations. One can see that using $q = 3$ bits provides us the best communication efficiency. When $q \geq 3$, the compressed algorithm converges to the optimal solution using essentially the same number of iterations as the uncompressed version. Using $q = 2$ causes divergence because the assumption in Theorem 1 on the compressor may not be violated. When the algorithm reaches 1.5 times of the optimal objective value, the implementations using q =3, 4, and 5 bits for real-value encoding attain `CR`s of 0.9062, 0.8681, and 0.8428, respectively—i.e., 90% communication reduction can be attained relative to the unquantized version `AltMaxVar`.

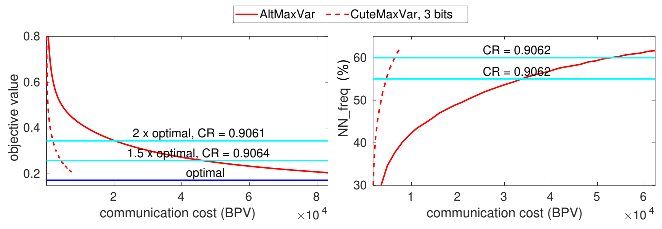

Fig. 3. Training objective [left] and `NN_freq` [right] vs. communication cost for the sentence embedding task; q =3 bits.

Real-Data Experiment. We also observe the performance of proposed method on the Europarl Corpus [24]. We are given high dimensional representations of sentences in different language, from which we learn low dimensional representations using GCCA. We select 3 languages and use each language as a view, and follow the method described in [6] to obtain a 524,288-dimensional representation for each sentence. We use 160,000 sentences in each language as the training data and 10,000 sentences as the testing data. We expect that the learnt representations for the same sentence are similar across all languages. `NN_freq` [6] captures this idea by counting the frequency of $\|\boldsymbol{X}_\ell^{(\text{test})}(a,:)\boldsymbol{Q}_\ell - \boldsymbol{X}_m^{(\text{test})}(b,:)\boldsymbol{Q}_m\|_2^2$ for different ℓ and m being the smallest when $a = b$.

Fig. 3 depicts the communication cost of `AltMaxVar` and `CuteMaxVar` for the aforementioned task when setting $K = 10$. We use $q = 3$. For the $\boldsymbol{Q}_i$-subproblem, we set $T = 20$ and use SGD with a batch size of 2,000. One can see that `CuteMaxVar` consistently achieves a compression ratio of above 0.9 to reach different objective values.

5. CONCLUSION

In this work, we proposed a provably convergent communication-efficient method for distributed MAX-VAR GCCA. By utilizing a carefully designed compression scheme, we showed that the proposed method consistently achieves a reduction of 90% in both downlink and uplink communication overhead, without any degradation in the convergence properties compared to the uncompressed version. We also offered tailored convergence analysis to support our design. In particular, we showed that the proposed algorithm approaches a global optimum with a geometric rate, even under heavy communication compression. Our proof is a nontrivial generalization of gradient compression and error feedback to the domain of eigenvalue computation, which may also be of interest in other computational linear algebra applications.

6. REFERENCES

[1] 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.

[2] M. Sørensen, C. I. Kanatsoulis, and N. D. Sidiropoulos, “Generalized canonical correlation analysis: A subspace intersection approach,” *IEEE Trans. Signal Process.*, vol. 69, pp. 2452–2467, 2021.

[3] P. Horst, *Generalized canonical correlations and their application to experimental data*. Journal of Clinical Psychology, 1961, no. 14.

[4] X. Fu, K. Huang, M. Hong, N. D. Sidiropoulos, and A. M.-C. So, “Scalable and flexible multiview MAX-VAR canonical correlation analysis,” *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4150–4165, 2017.

[5] X. Fu, K. Huang, E. E. Papalexakis, H. A. Song, P. Talukdar, N. D. Sidiropoulos, C. Faloutsos, and T. Mitchell, “Efficient and distributed generalized canonical correlation analysis for big multiview data,” *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2304–2318, 2018.

[6] C. I. Kanatsoulis, X. Fu, N. D. Sidiropoulos, and M. Hong, “Structured SUMCOR multiview canonical correlation analysis for large-scale data,” *IEEE Trans. Signal Process.*, vol. 67, no. 2, pp. 306–319, 2018.

[7] X. Fu, K. Huang, E. E. Papalexakis, H.-A. Song, P. P. Talukdar, N. D. Sidiropoulos, C. Faloutsos, and T. Mitchell, “Efficient and distributed algorithms for large-scale generalized canonical correlations analysis,” in *Proc. IEEE ICDM 2016*, 2016, pp. 871–876.

[8] A. Bertrand and M. Moonen, “Distributed canonical correlation analysis in wireless sensor networks with application to distributed blind source separation,” *IEEE Trans. Signal Process.*, vol. 63, no. 18, pp. 4800–4813, 2015.

[9] Y.-O. Li, T. Adali, W. Wang, and V. D. Calhoun, “Joint blind source separation by multiset canonical correlation analysis,” *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3918–3929, 2009.

[10] P. Rastogi, B. Van Durme, and R. Arora, “Multiview LSA: Representation learning via generalized CCA,” in *Proc. NAACL*, 2015, pp. 556–566.

[11] Q. Wu and K. M. Wong, “Un-music and un-cle: An application of generalized correlation analysis to the estimation of the direction of arrival of signals in unknown correlated noise,” *IEEE Trans. Signal Process.*, vol. 42, no. 9, pp. 2331–2343, 1994.

[12] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, “Multi-view clustering via canonical correlation analysis,” in *Proc. ICML*, 2009, pp. 129–136.

[13] C. Hovine and A. Bertrand, “Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network,” in *Proc. IEEE EUSIPCO*, 2021, pp. 2159–2163.

[14] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, “signSGD: Compressed optimisation for non-convex problems,” in *Proc. ICML.* PMLR, 2018, pp. 560–569.

[15] D. Basu, D. Data, C. Karakus, and S. Diggavi, “Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations,” in *Proc. NeurIPS*, vol. 32, 2019.

[16] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient SGD via gradient quantization and encoding,” in *Proc. NeurIPS*, vol. 30, 2017, pp. 1709–1720.

[17] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, “Error feedback fixes signSGD and other gradient compression schemes,” in *Proc. ICML.* PMLR, 2019, pp. 3252–3261.

[18] 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. Washington, DC, 1968, pp. 227–228.

[19] P. H. Schönemann, “A generalized solution of the orthogonal procrustes problem,” *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[20] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, “Sparsified SGD with memory,” in *Proc. NeurIPS*, vol. 31, 2018, pp. 4447–4458.

[21] S. Shrestha and X. Fu, “Communication-efficient distributed MAX-VAR generalized CCA via error feedback-assisted quantization,” Tech. Rep., 2021. [Online]. Available: http://web.engr.oregonstate.edu/~fuxia/shrestha2021comm.pdf

[22] S. Shrestha and X. Fu, “Communication-efficient distributed linear and deep generalized canonical correlation analysis,” *arXiv preprint arXiv:2109.12400*, 2021.

[23] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, vol. 3.

[24] P. Koehn, “Europarl: A parallel corpus for statistical machine translation,” *Machine Translation Summit, 2005*, pp. 79–86, 2005.