Distributed Principal Component Analysis with Limited Communication

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study efficient distributed algorithms for the fundamental problem of principal component analysis and leading eigenvector computation on the sphere, when the data are randomly distributed among a set of computational nodes. We propose a new quantized variant of Riemannian gradient descent to solve this problem, and prove that the algorithm converges with high probability under a set of necessary spherical-convexity properties. We give bounds on the number of bits transmitted by the algorithm under common initialization schemes, and investigate the dependency on the problem dimension in each case.

1 Introduction

The need to process ever increasing quantities of data has motivated significant research interest into efficient *distributed* algorithms for standard tasks in optimization and data analysis; see, e.g., [15, 26, 10]. In this context, one of the fundamental metrics of interest is *communication cost*, measured as the number of bits sent and received by the processors throughout computation. Communication-efficient variants are known for several classical tasks and algorithms, from basic ones such as mean estimation [27, 19, 8] to variants of gradient descent [18, 11] or even higher-order optimization [2, 14].

By contrast, much less is known concerning the communication cost for classical problems in data analysis, such as Principal Component Analysis (PCA) [12, 5], or spectral clustering [22]. In this paper, we will focus on the closely-related problem of computing the leading eigenvector of the data covariance matrix, i.e., determining the first principal component, in the setting where d-dimensional data comes from an unknown distribution, and is split among n machines.

Given the fundamental nature of PCA, there is a rich body of work on solving *sequential* variants, both in the Euclidean domain [23, 24, 25] and, more recently, based on Riemannian optimization. This is because maximizing the Rayleigh quotient can be re-interpreted in terms of a hyper-sphere, leading to an unconstrained Riemannian optimization problem [1, 31, 30]. These references have exclusively focused on the *sequential* setting, where the data can be stored and processed by a single node. However, in many settings the large volume of data requires processing by multiple nodes, which has inspired significant work on reducing the distribution overheads of PCA, e.g. [4, 10], specifically on obtaining efficient algorithms in terms of *round complexity*.

More precisely, an impressive line of work considered the *deterministic* setting [16, 4, 6, 9], where the input is partitioned arbitrarily between machines, for which both round-efficient and communication-efficient solutions are now known. The general approach is to first perform a careful low-rank approximation of the local data and then communicate iteratively to reconstruct a faithful estimate of the covariance matrix. Due to the worst-case setting, the number of communication rounds scales polynomially in the eigengap δ and in the precision ε, making them very costly in practice.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Thus, on the practical side, it is common to assume a setting with stochastic rather than worst-case data partitioning. Specifically, Garber et al. [10] consider this setting, and note that standard solutions based on distributed Power Method or Lanczos would require a number of communication rounds which increases with the sample size. This makes these algorithms impractical at large scale. They circumvent this issue by replacing explicit Power Method iterations with a set of convex optimization problems, which can be solved in a round-efficient manner. Huang and Pan [13] proposed a round-efficient solution, by leveraging the connection to Riemannian optimization. Their proposal relies on a procedure which allows the update of node-local variables via a surrogate gradient, whose deviation from the global gradient should be bounded. Thus, communication could only be used to reach consensus on the local variables among the nodes. Unfortunately, upon close inspection, we have noted significant technical gaps in their analysis, which we discuss in detail in the next section.

This line of work in the stochastic setting focused exclusively on *latency cost*, i.e., the number of communication rounds for solving this problem, without specifically considering *bandwidth cost*, i.e., the number of bits which need to be transmitted per round. For highly-dimensional data and/or highly-distributed settings, it is known that bandwidth cost can be a significant bottleneck, and an important amount of work deals with it for other classic problems, e.g. [27, 11, 14]. By contrast, much less is known concerning the communication complexity of distributed PCA.

Contribution. In this paper, we address this gap, providing a new algorithm for distributed leading eigenvector computation, which specifically minimizes the total number of bits sent and received by the nodes. Our approach leverages the connection between PCA and Riemannian optimization; specifically, we propose the first distributed variant of Riemannian gradient descent, which supports *quantized communication*. We prove convergence of this algorithm utilizing a set of necessary convexity-type properties, and bound its communication complexity under different initializations.

At the technical level, our result is based on a new *sequential* variant of Riemannian gradient descent on the sphere. This may be of general interest and the algorithm can be generalized to Riemannian manifolds of bounded sectional curvatures using more sophisticated geometric bounds. We strengthen the gradient dominance property proved in [30], by proving that our objective is weakly-quasi-convex in a ball of some minimizer and combining that with a known quadratic-growth condition. We leverage these properties by adapting results of [7] in the Riemannian setting and choosing carefully the learning-rate in order to guarantee convergence for the single-node version.

This algorithm and its analysis are specifically-designed to be amenable to distribution: specifically, we port it to the distributed setting, and add communication-compression in the tangent spaces of the sphere, using a variant of the lattice quantization technique of [8], which was developed in the context of standard gradient descent. We provide a non-trivial analysis for the communication complexity of the resulting algorithm under different initializations. We show that, under random initialization, a solution can be reached with total communication that is quadratic in the dimension d, and linear in $1/\delta$ with δ the eigengap; when the initialization is done to the leading eigenvector of one of the local covariance matrices, this dependency can be reduced to linear in d, which is optimal [3].

2 Setting and Related Work

Setting. We assume to be given m total samples coming from some distribution $\mathcal{D}$, organized as a global $m \times d$ data matrix M, which is partitioned row-wise among n processors, with node i being assigned the matrix M_i, consisting of m_i consecutive rows, such that $\sum_{i=1}^{n} m_i = m$. As is common, let $A := M^T M = \sum_{i=1}^{d} M_i^T M_i$ be the global covariance matrix and $A_i := M_i^T M_i$ the local covariance matrix owned by node i. We denote by $\lambda_1, \lambda_2, ..., \lambda_d$ the eigenvalues of A in descending order and by $v_1, v_2, ..., v_d$ the corresponding eigenvectors. Then, it is well-known that, if the gap $\delta = \lambda_1 - \lambda_2$ between the two leading eigenvalues of A is positive, we can approximate the leading eigenvector by solving the following empirical risk minimization problem up to accuracy ε:

$$x^\star = \text{argmin}_{x \in \mathbb{R}^d} \left(-\frac{x^T A x}{\|x\|^2} \right) = \text{argmin}_{x \in \mathbb{R}^d, \|x\|_2 = 1} \left(-x^T A x \right) = \text{argmin}_{x \in \mathbb{S}^{d-1}} \left(-x^T A x \right), \quad (1)$$

where $\mathbb{S}^{d-1}$ is the $(d-1)$-dimensional sphere.

We define $f : \mathbb{S}^{d-1} \to \mathbb{R}$, with $f(x) = -x^T A x$ and $f_i : \mathbb{S}^{d-1} \to \mathbb{R}$, with $f_i(x) = -x^T A_i x$. Since the inner product is bilinear, we can write the global cost as the sum of the local costs: $f(x) = \sum_{i=1}^{n} f_i(x)$.

In our analysis we use important notions from the geometry of the sphere, that we recall in Appendix A.

Related Work. As discussed, there has been a tremendous amount of research on efficient variants of PCA and related problems [5, 23, 24, 25, 1, 31, 30]. Due to space constraints, we focus on related work on communication-efficient algorithms. In particular, we discuss the relationship to previous *round-efficient* algorithms; to our knowledge, this is the first work to specifically focus on the bit complexity of this problem. More precisely, previous work implicitly assumes that algorithms are able to transmit real numbers at unit cost.

The straightforward approach to solve the minimization problem (1) in a distributed setting, where the data rows are partitioned, would be to use a distributed version of the Power Method, Riemannian gradient descent (RGD), or the Lanczos algorithm. In order to achieve an ε-approximation of the minimizer $x^\star$, the latter two algorithms require $\tilde{O}(\lambda_1 \log(1/\varepsilon)/\delta)$ rounds, where the $\tilde{O}$ notation hides poly-logarithmic factors in d. Distributed Lanczos and accelerated RGD would improve this by an $O(\sqrt{\lambda_1/\delta})$ factor. However, Garber et al. [10] point out that, when δ is small, e.g. $\delta = \Theta(1/\sqrt{Kn})$, then unfortunately the number of communication rounds would increase with the sample size, which renders these algorithms non-scalable.

Standard distributed convex approaches, e.g. [15, 26], do not directly extend to our setting due to non-convexity and the unit-norm constraint. Garber et al. [10] proposed a variant of the Power Method called *Distributed Shift-and-Invert (DSI)* which converges in roughly $O\left(\sqrt{\frac{b}{\delta\sqrt{n}}}\log^2(1/\varepsilon)\log(1/\delta)\right)$ rounds, where b is a bound on the squared ℓ_2-norm of the data. Huang and Pan [13] aimed to improve the dependency of the algorithm on ε and δ, and proposed an algorithm called *Communication-Efficient Distributed Riemannian Eigensolver (CEDRE)*. This algorithm is shown to have round complexity $O(\frac{b}{\delta\sqrt{n}}\log(1/\varepsilon))$, which does not scale with the sample size for $\delta = \Omega(1/\sqrt{Kn})$, and has only logarithmic dependency on the accuracy ε.

Technical issues in [13]. (see also Appendix E for a more thorough discussion) Huang and Pan [13] proposed an interesting approach, which could provide the most round-efficient distributed algorithm to date. Despite we find the main idea of this paper really creative, we have unfortunately identified a significant gap in their analysis, which we now describe.

Specifically, one of their main results, Theorem 3, uses the local gradient dominance shown in [30]; yet, the application of this result is invalid, as it is done without knowing in advance that the iterates of the algorithms continue to remain in the ball of initialization. This is compounded by another error on the constant of gradient dominance (Lemma 2), which we believe is caused by a typo in the last part of the proof of Theorem 4 in [30], which unfortunately is propagated into their proof. More precisely, the objective is indeed gradient dominated, but with a constant which vanishes when we approach the equator, in contrast with the $2/\delta$ which is claimed *globally* (we reprove this result independently in Proposition 4). Thus, starting from a point lying in some ball of the minimizer probably yields to a new point where the objective is more poorly gradient dominated.

This is a non-trivial technical problem which, in the case of gradient descent, can be addressed by a choice of the learning rate depending on the initialization. Given these issues, we perform a new and formally-correct analysis for gradient descent based on new convexity properties which guarantee convergence directly in terms of iterates, and not just function values. We would like however to note that our focus in this paper is on bit and not round complexity.

3 Step 1: Computing the Leading Eigenvector at One Node

3.1 Convexity-like Properties and Smoothness

We first analyze the case that all the matrices A_i are simultaneously known to all machines. This practically means that we can run gradient descent in one of the machines (let it be the master), since we have all the data stored there. All the proofs for this section are in Appendix B.

Our problem reads

$$\min_{x \in \mathbb{S}^{d-1}} -x^T A x$$

where $A = M^T M$ is the global covariance matrix. If $\delta = \lambda_1 - \lambda_2 > 0$, this problem has exactly two global minima: v_1 and $-v_1$. Let $x \in \mathbb{S}^{d-1}$ be an arbitrary point. Then x can be written in the form $x = \sum_{i=1}^{d} \alpha_i v_i$. Fixing the minimizer v_1, we have that a ball in $\mathbb{S}^{d-1}$ around v_1 is of the form

$$B_a = \{x \in \mathbb{S}^{d-1} \mid \alpha_1 \geq a\} \tag{2}$$

for some a. Without loss of generality, we may assume $a > 0$ (otherwise, consider the ball around $-v_1$ to establish convergence to $-v_1$). In this case, the region $\mathcal{A}$ as defined in Def. 11 is identical to B_a and $x^* := v_1$.

We begin by investigating the convexity properties of the function $-x^T Ax$. In particular, we prove that this function is weakly-quasi-convex (see Def. 13) with constant $2a$ in the ball B_a. See also Appendix A for a definition of the Riemannian gradient $\text{grad} f$ and the logarithm $\log$.

Proposition 1. *The function $f(x) = -x^T Ax$ satisfies*

$$2a(f(x) - f^*) \leq \langle \text{grad} f(x), -\log_x(x^*) \rangle$$

for any $x \in B_a$ with $a > 0$.

We continue by providing a quadratic growth condition for our cost function, which can also be found in [29, Lemma 2] in a slightly different form. Here dist is the intrinsic distance in the sphere, that we also define in Appendix A.

Proposition 2. *The function $f(x) = -x^T Ax$ satisfies*

$$f(x) - f^* \geq \frac{\delta}{4} \text{dist}^2(x, x^*)$$

for any $x \in B_a$ with $a > 0$.

According to Def. 14, this means that f satisfies quadratic growth with constant $\mu = \delta/2$.

Next, we prove that quadratic growth and weak-quasi-convexity imply a "strong-weak-convexity" property. This is similar to [7, Proposition 3.1].

Proposition 3. *A function $f \colon \mathbb{S}^{d-1} \to \mathbb{R}$ which satisfies quadratic growth with constant $\mu > 0$ and is $2a$-weakly-quasi-convex with respect to a minimizer x^* in a ball of this minimizer, satisfies for any x in the same ball the inequality*

$$f(x) - f^* \leq \frac{1}{a} \langle \text{grad} f(x), -\log_x(x^*) \rangle - \frac{\mu}{2} \text{dist}^2(x, x^*).$$

Interestingly, using Proposition 3, we can recover the gradient dominance property proved in [30, Theorem 4].

Proposition 4. *The function $f(x) = -x^T Ax$ satisfies*

$$\|\text{grad} f(x)\|^2 \geq \delta a^2 (f(x) - f^*)$$

for any $x \in B_a$ with $a > 0$.

The proof of Proposition 4 can be done independently based only on Proposition 3 and [7, Lemma 3.2]. This verifies that our results until now are optimal regarding the quality of initialization since gradient dominance proved in [30] is optimal in that sense (i.e. the inequalities in the proof are tight).

Proposition 5. *The function $f(x) = -x^T Ax$ is geodesically $2\lambda_1$-smooth on the sphere.*

Thus, the smoothness constant of f, as defined in Def. 12, equals $\gamma = 2\lambda_1$. Similarly, let γ_i denote the smoothness constant of f_i, which equals twice the largest eigenvalue of A_i. In order to estimate γ in a distributed fashion using only the local data matrices A_i, we shall use the over-approximation $\gamma \leq n \max_{i=1,\ldots,n} \gamma_i$.

Smoothness is essential in order to bound the quantization error in the distributed case. However, in Lemma 17 of Appendix B, we provide an even stronger smoothness property that is used in our single-node convergence result.

3.2 Convergence

We now consider Riemannian gradient descent with learning rate $\eta > 0$ and started from a point $x^{(0)} \in B_a$:

$$x^{(t+1)} = \exp_{x^{(t)}}(-\eta \,\mathrm{grad} f(x^{(t)})),$$

where $\exp$ is the exponential map of the sphere, defined in Appendix A.

Using Proposition 3 and a proper choice of η, we can establish a convergence rate for the instrinsic distance of the iterates to the minimizer.

Proposition 6. *Riemannian gradient descent for $f(x) = -x^T Ax$ started from a point $x^{(0)} \in B_a$ and with step-size $\eta \leq a/\gamma$ where $\gamma \geq 2\lambda_1$, produces iterates that satisfy*

$$\mathrm{dist}^2(x^{(t+1)}, x^*) \leq (1 - a\mu\eta)\,\mathrm{dist}^2(x^{(t)}, x^*) \qquad \textit{for all } t \geq 0 \textit{ and } \mu = \delta/2.$$

Note that this result implies directly that if our initialization $x^{(0)}$ lies in the ball B_a, the distance of $x^{(1)}$ to the center x^* decreases and thus all subsequent iterates continue being in the initialization ball. This is essential since it guarantees that the convexity-like properties for f continue to hold during the whole optimization process. The proof of Proposition 6 is in Appendix B.

This result is also valid in a more general setting when applied to an arbitrary $f\colon \mathbb{S}^{d-1} \to \mathbb{R}$ that is $2a$-weakly-quasi convex and γ-smooth, and that satisfies quadratic growth with constant μ and the corresponding Lemma 17. The Euclidean analogue of this result can be found in [7, Lemma 4.4].

4 Step 2: Computing the Leading Eigenvector among Several Nodes

We now present our version of distributed gradient descent for leading eigenvector computation and measure its bit complexity until reaching accuracy ϵ in terms of the intrinsic distance of an iterate $x^{(T)}$ from the minimizer x^* (x^* is the leading eigenvector closest to the initialization point).

Lattice Quantization. For estimating the Riemannian gradient in a distributed manner with limited communication, we use a quantization procedure developed in [8]. The original quantization scheme involves randomness, but we use a *deterministic* version of it, by picking up the closest point to the vector that we want to encode. This is similar to the quantization scheme used by [3] and has the following properties.

Proposition 7. *[8, 3] Denoting by b the number of bits that each machine uses to communicate, there exists a quantization function*

$$Q\colon \mathbb{R}^{d-1} \times \mathbb{R}^{d-1} \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^{d-1},$$

which, for each $w, y > 0$, consists of an encoding function $\mathrm{enc}_{w,y} : \mathbb{R}^{d-1} \to \{0,1\}^b$ and a decoding one $\mathrm{dec}_{w,y} : \{0,1\}^b \times \mathbb{R}^{d-1} \to \mathbb{R}^{d-1}$, such that, for all $x, x' \in \mathbb{R}^{d-1}$,

- $\mathrm{dec}_{w,y}(\mathrm{enc}_{w,y}(x), x') = Q(x, x', y, w)$, *if* $\|x - x'\| \leq y$.
- $\|Q(x, x', y, w) - x\| \leq w$, *if* $\|x - x'\| \leq y$.
- *If $y/w > 1$, the cost of the quantization procedure in number of bits satisfies* $b = \mathcal{O}((d-1)\log_2\left(\frac{y}{w}\right)) = \mathcal{O}(d\log\left(\frac{y}{w}\right))$.

In the following, the quantization takes place in the tangent space of each iterate $T_{x^{(t)}}\mathbb{S}^{d-1}$, which is linearly isomorphic to $\mathbb{R}^{d-1}$. We denote by Q_x the specification of the function Q at $T_x\mathbb{S}^{d-1}$. The vector inputs of the function Q_x are represented in the local coordinate system of the tangent space that the quantization takes place at each step. For decoding at $t > 0$, we use information obtained in the previous step, that we need to translate to the same tangent space. We do that using parallel transport Γ, which we define in Appendix A.

Algorithm

We present now our main algorithm, which is inspired by quantized gradient descent firstly designed by [21], and its similar version in [3].

---

1. Choose an arbitrary machine to be the master node, let it be i_0.
2. Choose $x^{(0)} \in \mathbb{S}^{d-1}$ (we analyze later specific ways to do that).
3. Consider the following parameters

$$\sigma := 1 - a\mu\eta,\ K := \frac{2}{\sqrt{\sigma}},\ \theta := \frac{\sqrt{\sigma}(1-\sqrt{\sigma})}{4},$$
$$\sqrt{\xi} := \theta K + \sqrt{\sigma},\ R^{(t)} = \gamma K(\sqrt{\xi})^t D$$

where D is a bound for the diameter of our working domain. In the case of the sphere, D can be chosen to be π.
(We assume that $a\mu\eta \leq \frac{1}{2}$, otherwise we run the algorithm with $\sigma = \frac{1}{2}$.)
In $T_{x^{(0)}}\mathbb{S}^{d-1}$:

4. Compute the local Riemannian gradient $\text{grad} f_i(x^{(0)})$ at $x^{(0)}$ in each node.
5. Encode $\text{grad} f_i(x^{(0)})$ in each node and decode in the master node using its local information:

$$q_i^{(0)} = Q_{x^{(0)}}\left(\text{grad} f_i(x^{(0)}), \text{grad} f_{i_0}(x^{(0)}), 2\gamma\pi, \frac{\theta R^{(0)}}{2n}\right).$$

6. Sum the decoded vectors in the master node:

$$r^{(0)} = \sum_{i=1}^{n} \text{grad} f_i(x^{(0)}).$$

7. Encode the sum in the master and decode in each machine i using its local information:

$$q^{(0)} = Q_{x^{(0)}}\left(r^{(0)}, \text{grad} f_i(x^{(0)}), \frac{\theta R^{(0)}}{2} + 2\gamma\pi, \frac{\theta R^{(0)}}{2}\right).$$

For $t \geq 0$:

8. Take a gradient step using the exponential map:

$$x^{(t+1)} = \exp_{x^{(t)}}(-\eta q^{(t)})$$

with step-size η (the step-size is discussed later).
In $T_{x^{(t+1)}}\mathbb{S}^{d-1}$:

9. Compute the local Riemannian gradient $\text{grad} f_i(x^{(t+1)})$ at $x^{(t+1)}$ in each node.
10. Encode $\text{grad} f_i(x^{(t+1)})$ in each node and decode in the master node using its (parallelly transported) local information from the previous step:

$$q_i^{(t+1)} = Q_{x^{(t+1)}}\left(\text{grad} f_i(x^{(t+1)}), \Gamma_{x^{(t)}}^{x^{(t+1)}} q_i^{(t)}, \frac{R^{(t+1)}}{n}, \frac{\theta R^{(t+1)}}{2n}\right).$$

11. Sum the decoded vectors in the master node:

$$r^{(t+1)} = \sum_{i=1}^{n} \text{grad} f_i(x^{(t+1)}).$$

12. Encode the sum in the master and decode in each machine using its local information in the previous step after parallel transport:

$$q^{(t+1)} = Q_{x^{(t+1)}}\left(r^{(t+1)}, \Gamma_{x^{(t)}}^{x^{(t+1)}} q^{(t)}, \left(1 + \frac{\theta}{2}\right) R^{(t+1)}, \frac{\theta R^{(t+1)}}{2}\right).$$

Convergence We first control the convergence of iterates simultaneously with the convergence of quantized gradients.

Note that

$$\sqrt{\xi} = \frac{1-\sqrt{\sigma}}{2} + \sqrt{\sigma} = \frac{1+\sqrt{\sigma}}{2} \leq \frac{\sqrt{2}\sqrt{1+\sigma}}{2} = \sqrt{\frac{1+\sigma}{2}}.$$

Lemma 8. *If $x^{(0)} \in B_a$ and $\eta \leq \frac{a}{\gamma}$, the previous quantized gradient descent algorithm produces iterates $x^{(t)}$ and quantized gradients $q^{(t)}$ that satisfy*

$$\text{dist}^2(x^{(t)}, x^*) \leq \xi^t D^2, \quad \|q_i^{(t)} - \text{grad} f_i(x^{(t)})\| \leq \frac{\theta R^{(t)}}{2n}, \quad \|q^{(t)} - \text{grad} f(x^{(t)})\| \leq \theta R^{(t)}.$$

The proof is a Riemannian adaptation of the similar one in [21] and [3] and is presented in Appendix C. We recall that since the sphere is positively curved, it provides a landscape easier for optimization. It is quite direct to derive a general Riemannian method for manifolds of bounded curvature using more advanced geometric bounds, however this exceeds the scope of this work which focuses on leading eigenvector computation.

We now move to our main complexity result.

Theorem 9. *Let $x^{(0)} \in B_a$ and $\eta \leq \frac{a}{\gamma}$. Then, the previous quantized gradient descent algorithm needs at most*

$$b = \mathcal{O}\left(nd\frac{1}{a\delta\eta}\log\left(\frac{n}{a\delta\eta}\right)\log\left(\frac{D}{\epsilon}\right)\right)$$

bits in total to estimate the leading eigenvector with an accuracy ϵ measured in intrinsic distance.

The proof is based on the previous Lemma 8 in order to count the number of steps that the algorithm needs to estimate the minimizer with accuracy ϵ and Proposition 7 to count the quantization cost in each round. We present the proof again in Appendix C.

5 Step 3: Dependence on Initialization

5.1 Uniformly random initialization

The cheapest choice to intialize quantized gradient descent is a point in the sphere chosen uniformly at random. According to Theorem 4 in [30], such a random point $x^{(0)}$ will lie, with probability at least $1-p$, in the ball B_a (see Def. 2) where

$$a \geq c\frac{p}{\sqrt{d}}. \tag{3}$$

Here, c is a universal constant (we estimated numerically that c is lower bounded by 1, see Appendix D, thus we can use $c = 1$). By choosing the step-size η as

$$\eta = \frac{cp}{\sqrt{d}\gamma},$$

we are guaranteed that $\eta \leq \frac{a}{\gamma}$ with probability at least $1-p$. Our analysis above therefore applies (up to probability $1-p$) and the general communication complexity result becomes

$$\mathcal{O}\left(nd\frac{1}{a\delta\eta}\log\frac{n}{a\delta\eta}\log\frac{D}{\epsilon}\right) \leq \mathcal{O}\left(nd\frac{1}{\eta\gamma\delta\eta}\log\frac{n}{\eta\gamma\delta\eta}\log\frac{D}{\epsilon}\right) = \mathcal{O}\left(nd\frac{1}{\eta^2\gamma\delta}\log\frac{n}{\eta^2\gamma\delta}\log\frac{D}{\epsilon}\right).$$

Substituting $\eta^2 = \frac{p^2c^2}{d\gamma^2}$, the number of bits satisfies (up to probability $1-p$) the upper bound

$$\boxed{b = \mathcal{O}\left(n\frac{d^2}{p^2}\frac{\gamma}{\delta}\log\frac{nd\gamma}{p\delta}\log\frac{D}{\epsilon}\right).}$$

5.2 Initialization in the Leading Eigenvector of the Covariance Matrix in the Master Node

A reasonable strategy to get a more accurate initialization is to perform SVD to one of the local covariance matrices, for instance A_{i_0} (in the master node i_0), and compute its leading eigenvector, let it be v^{i_0}. Then we communicate this vector to all machines in order to use its quantized approximation

$x^{(0)}$ as initialization. We define:

$$\tilde{x}^{(0)} = Q\left(v^{i_0}, v^i, \|v^{i_0} - v^i\|, \frac{\langle v^{i_0}, v_1\rangle}{2(\sqrt{2}+2)}\right)$$
$$x^{(0)} = \frac{\tilde{x}^{(0)}}{\|\tilde{x}^{(0)}\|}$$

where Q is the lattice quantization scheme in $\mathbb{R}^d$ (i.e. we quantize the leading eigenvector of the master node as a vector in $\mathbb{R}^d$ and then project back to the sphere). The input and output variance in this quantization can be bounded by constants that we can practically estimate (see Appendix D).

Proposition 10. *Assume that our data are i.i.d. and sampled from a distribution $\mathcal{D}$ bounded in ℓ_2 norm by a constant b. Given that the eigengap δ and the number of machines n satisfy*

$$\delta \geq \Omega\left(\sqrt{n}\sqrt{\log\frac{d}{p}}\right), \tag{4}$$

we have that the previous quantization costs $\mathcal{O}(nd)$ many bits and $\langle x^{(0)}, x^*\rangle$ is lower bounded by a constant with probability at least $1-p$.

Thus, if bound 4 is satisfied, then the communication complexity becomes

$$\boxed{b = \mathcal{O}\left(nd\frac{\gamma}{\delta}\log\frac{n\gamma}{\delta}\log\frac{D}{\epsilon}\right)}$$

many bits in total with probability at least $1-p$ (notice that this can be further simplified using bound 4), since a in Theorem 9 is lower bounded by a constant and the communication cost of quantizing v^{i_0} does not affect the total communication cost.

Condition 4 is quite typical in this literature; see [13] and references therein (beginning of page 2), as we also briefly discussed in the introduction. Notice that $\sqrt{n}$ appears in the numerator and not the denominator, only because we deal with the sum of local covariance matrices and not the average, thus our eigengap is n times larger than the eigengap of the normalized covariance matrix.

6 Discussion

We proposed a communication efficient algorithm to compute the leading eigenvector of a covariance data matrix in a distributed fashion using only limited amount of bits to communicate. To the best of our knowledge, this is the first algorithm for this problem with bounded bit complexity. To that end, we focused on gradient descent for two reasons: first, it is an algorithm which provides robust convergence in terms of iterates and can circumvent the issues that arise due to a cost function that only has local convexity-like properties; second, it is the only first-order algorithm known to be amenable to quantization.

There are, however, limitations in our analysis that can serve as an excellent basis for future work:

First, our choice of step-size is quite conservative since the local convexity properties of the cost function improve when approaching the optimum. Thus, instead of setting the step-size to be $\eta = a/\gamma$, where a is a lower bound for $\langle x^{(0)}, x^*\rangle$, we can re-run the whole analysis with an adaptive step-size $\eta = \langle x^{(t)}, x^*\rangle/\gamma$ based on an improved value for a from $\langle x^{(t)}, x^*\rangle$. This could improve the dependency on the dimension of our complexity results.

Second, the version of Riemannian gradient descent that we have chosen is expensive, since one needs to calculate the exponential map in each step using sin and cos calculations. Practitioners usually use cheaper retraction-based versions, and it would be of interest to study how our analysis can be modified if the exponential map is substituted by a retraction (i.e., any first-order approximation).

Third, it would be interesting to see whether the ideas and techniques presented in this work can also be used for other data science problems in a distributed fashion through communication-efficient Riemannian methods. This could be, for example, PCA for the top k eigenvectors using optimization on the Stiefel or Grassmann manifold [1] and low-rank matrix completion on the fixed-rank manifold [28].

Bibliography

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] Foivos Alimisis, Peter Davies, and Dan Alistarh. Communication-efficient distributed optimization with quantized preconditioners. *arXiv preprint arXiv:2102.07214*, 2021.

[3] Dan Alistarh and Janne H Korhonen. Improved communication lower bounds for distributed optimisation. *arXiv preprint arXiv:2010.08222*, 2020.

[4] Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. *arXiv preprint arXiv:1408.5823*, 2014.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 236–249, 2016.

[7] Jingjing Bu and Mehran Mesbahi. A note on nesterov's accelerated method in nonconvex optimization: a weak estimate sequence approach, 2020.

[8] Peter Davies, Vijaykrishna Gurunathan, Niusha Moshrefi, Saleh Ashkboos, and Dan Alistarh. Distributed variance reduction with optimal communication. *arXiv e-prints*, pages arXiv–2002, 2020.

[9] Jianqing Fan, Dong Wang, Kaizheng Wang, and Ziwei Zhu. Distributed estimation of principal eigenspaces. *Annals of statistics*, 47(6):3009, 2019.

[10] Dan Garber, Ohad Shamir, and Nathan Srebro. Communication-efficient algorithms for distributed stochastic principal component analysis. In *International Conference on Machine Learning*, pages 1203–1212. PMLR, 2017.

[11] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[12] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[13] Long-Kai Huang and Sinno Pan. Communication-efficient distributed pca by riemannian optimization. In *International Conference on Machine Learning*, pages 4465–4474. PMLR, 2020.

[14] Rustem Islamov, Xun Qian, and Peter Richtárik. Distributed second order methods with fast rates and compressed communication. *arXiv preprint arXiv:2102.07158*, 2021.

[15] Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. *arXiv preprint arXiv:1409.1458*, 2014.

[16] Ravi Kannan, Santosh Vempala, and David Woodruff. Principal component analysis and higher correlations for distributed data. In *Conference on Learning Theory*, pages 1040–1057. PMLR, 2014.

[17] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition, 2020.

[18] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[19] Jakub Konečnỳ and Peter Richtárik. Randomized distributed mean estimation: Accuracy vs. communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018.

[20] S. Li. Concise Formulas for the Area and Volume of a Hyperspherical Cap. *Asian Journal of Mathematics & Statistics*, 4(1):66–70, 2011.

[21] Sindri Magnússon, Hossein Shokri-Ghadikolaei, and Na Li. On maintaining linear convergence of distributed learning and optimization under limited communication. *IEEE Transactions on Signal Processing*, 68:6101–6116, 2020.

[22] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001.

[23] Erkki Oja and Juha Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of mathematical analysis and applications*, 106(1):69–84, 1985.

[24] Ohad Shamir. A stochastic pca and svd algorithm with an exponential convergence rate. In *International Conference on Machine Learning*, pages 144–152. PMLR, 2015.

[25] Ohad Shamir. Fast stochastic algorithms for svd and pca: Convergence properties and convexity. In *International Conference on Machine Learning*, pages 248–256. PMLR, 2016.

[26] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

[27] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pages 3329–3337. PMLR, 2017.

[28] Bart Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[29] Zhiqiang Xu, Xin Cao, and Xin Gao. Convergence analysis of gradient descent for eigenvector computation. International Joint Conferences on Artificial Intelligence, 2018.

[30] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *arXiv preprint arXiv:1605.07147*, 2016.

[31] Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pages 1617–1638. PMLR, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]