Doubly Accelerated Methods for Faster CCA and Generalized Eigendecomposition

Zeyuan Allen-Zhu [*1] **Yuanzhi Li** [*2]

Abstract

We study k-GenEV, the problem of finding the top k generalized eigenvectors, and k-CCA, the problem of finding the top k vectors in canonical-correlation analysis. We propose algorithms `LazyEV` and `LazyCCA` to solve the two problems with running times linearly dependent on the input size and on k. Furthermore, our algorithms are *doubly-accelerated*: our running times depend only on the square root of the matrix condition number, and on the square root of the eigengap. This is the first such result for both k-GenEV or k-CCA. We also provide the first gap-free results, which provide running times that depend on $1/\sqrt{\varepsilon}$ rather than the eigengap.

1 Introduction

The Generalized Eigenvector (GenEV) problem and the Canonical Correlation Analysis (CCA) are two fundamental problems in scientific computing, machine learning, operations research, and statistics. Algorithms solving these problems are often used to extract features to compare large-scale datasets, as well as used for problems in regression (Kakade & Foster, 2007), clustering (Chaudhuri et al., 2009), classification (Karampatziakis & Mineiro, 2014), word embeddings (Dhillon et al., 2011), and many others.

GenEV. Given two symmetric matrices $A, B \in \mathbb{R}^{d\times d}$ where B is positive definite. The GenEV problem is to find *generalized eigenvectors* $v_1, \dots, v_d$ where each v_i satisfies

$$v_i \in \arg\max_{v\in\mathbb{R}^d} \left|v^\top A v\right| \text{ s.t. } \begin{cases} v^\top B v = 1 \\ v^\top B v_j = 0 \ \forall j \in [i-1] \end{cases}$$

The values $\lambda_i \stackrel{\text{def}}{=} v_i^\top A v_i$ are known as the generalized eigenvalues, and it satisfies $|\lambda_1| \geq \cdots |\lambda_d|$. Following the tradition of (Wang et al., 2016; Garber & Hazan, 2015), we

assume *without loss of generality* that $\lambda_i \in [-1, 1]$.

CCA. Given matrices $X \in \mathbb{R}^{n\times d_x}, Y \in \mathbb{R}^{n\times d_y}$ and denoting by $S_{xx} = \frac{1}{n}X^\top X$, $S_{xy} = \frac{1}{n}X^\top Y$, $S_{yy} = \frac{1}{n}Y^\top Y$, the CCA problem is to find *canonical-correlation vectors* $\{(\phi_i, \psi_i)\}_{i=1}^r$ where $r = \min\{d_x, d_y\}$ and each pair

$$(\phi_i, \psi_i) \in \arg\max_{\phi\in\mathbb{R}^{d_x}, \psi\in\mathbb{R}^{d_y}} \{\phi^\top S_{xy} \psi\}$$

$$\text{such that} \quad \begin{cases} \phi^\top S_{xx}\phi = 1 \wedge \phi^\top S_{xx}\phi_j = 0 \ \forall j \in [i-1] \\ \psi^\top S_{yy}\psi = 1 \wedge \psi^\top S_{yy}\psi_j = 0 \ \forall j \in [i-1] \end{cases}$$

The values $\sigma_i \stackrel{\text{def}}{=} \phi_i^\top S_{xy}\psi_i \geq 0$ are known as the canonical-correlation coefficients, and

$1 \geq \sigma_1 \geq \cdots \geq \sigma_r \geq 0$ is *always* satisfied.

It is a fact that solving CCA *exactly* can be reduced to solving GenEV exactly, if one defines $B = \operatorname{diag}\{S_{xx}, S_{yy}\} \in \mathbb{R}^{d\times d}$ and $A = [[0, S_{xy}]; [S_{xy}^\top, 0]] \in \mathbb{R}^{d\times d}$ for $d \stackrel{\text{def}}{=} d_x + d_y$; see Lemma 2.3. (This reduction does not always hold if the generalized eigenvectors are computed only approximately.)

Despite the fundamental importance and the frequent necessity in applications, there are few results on obtaining provably efficient algorithms for GenEV and CCA until very recently. In the breakthrough result of Ma, Lu and Foster (Ma et al., 2015), they proposed to study algorithms to find top k generalized eigenvectors (k-GenEV) or top k canonical-correlation vectors (k-CCA). They designed an alternating minimization algorithm whose running time is only linear in the input matrix sparsity and nearly-linear in k. Such algorithms are very appealing because in real-life applications, it is often only relevant to obtain top correlation vectors, as opposed to the less meaningful vectors in the directions where the datasets do not correlate. Unfortunately, the method of Ma, Lu and Foster has a running time that linearly scales with κ and $1/\mathsf{gap}$, where

- $\kappa \geq 1$ is the condition number of matrix B in GenEV, or of matrices $X^\top X, Y^\top Y$ in CCA; and
- $\mathsf{gap} \in [0, 1)$ is the eigengap $\frac{\lambda_k - \lambda_{k+1}}{\lambda_k}$ in GenEV, or $\frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$ in CCA.

These parameters are usually not constants and scale with

[*]Equal contribution . Future version of this paper shall be found at `http://arxiv.org/abs/1607.06017`.
[1]Microsoft Research [2]Princeton University. Correspondence to: Zeyuan Allen-Zhu <zeyuan@csail.mit.edu>, Yuanzhi Li <yuanzhil@cs.princeton.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

the problem size.

Challenge 1: Acceleration

For many easier scientific computing problems, we are able to design algorithms that have *accelerated* dependencies on κ and $1/\mathsf{gap}$. As two concrete examples, k-PCA can be solved with a running time linearly in $1/\sqrt{\mathsf{gap}}$ as opposed to $1/\mathsf{gap}$ (Golub & Van Loan, 2012); computing $B^{-1}w$ for a vector w can be solved in time linearly in $\sqrt{\kappa}$ as opposed to κ, where κ is the condition number of matrix B (Shewchuk, 1994; Axelsson, 1985; Nesterov, 1983).

Therefore, can we obtain ***doubly-accelerated*** methods for k-GenEV and k-CCA, meaning that the running times linearly scale with both $\sqrt{\kappa}$ and $1/\sqrt{\mathsf{gap}}$? Before this paper, for the general case $k > 1$, the method of Ge *et al.* (Ge et al., 2016) made acceleration possible for parameter κ, but not for parameter $1/\mathsf{gap}$ (see Table 1).

Challenge 2: Gap-Freeness

Since gap can be even zero in the extreme case, can we design algorithms that do not scale with $1/\mathsf{gap}$? Recall that this is possible for the easier task of k-PCA. The block Krylov method (Musco & Musco, 2015) runs in time linear in $1/\sqrt{\varepsilon}$ as opposed to $1/\sqrt{\mathsf{gap}}$, where ε is the approximation ratio. There is no gap-free result previously known for k-GenEV or k-CCA even for $k = 1$.

Challenge 3: Stochasticity

For matrix-related problems, one can usually obtain stochastic running times which requires some notations to describe.

Consider a simple task of computing $B^{-1}w$ for some vector w, where accelerated methods solve it in time linearly in $\sqrt{\kappa}$ for κ being the condition number of B. If $B = \frac{1}{n}X^\top X$ is given in the form of a covariance matrix where $X \in \mathbb{R}^{n\times d}$, then (accelerated) stochastic methods compute $B^{-1}w$ in a time linearly in $(1 + \sqrt{\kappa'/n})$ instead of $\sqrt{\kappa}$, where $\kappa' = \frac{\max_{i\in[n]}\{\|X_i\|^2\}}{\lambda_{\min}(B)} \in \big[\kappa, n\kappa\big]$ and X_i is the i-th row of X. (See Lemma 2.6.) Since $1 + \sqrt{\kappa'/n} \le O(\sqrt{\kappa})$, stochastic methods are *no slower than* non-stochastic ones.

So, can we obtain a similar but doubly-accelerated stochastic method for k-CCA?[1] Note that, if the doubly-accelerated requirement is dropped, this task is easier and indeed possible, see Ge *et al.* (Ge et al., 2016). However, since their stochastic method is not doubly-accelerated, in certain parameter regimes, it runs even slower than non-stochastic ones (even for $k = 1$, see Table 2).

Remark. In general, if designed properly, for worst case running time:

- Accelerated results are usually better because they are no slower than non-accelerated ones in the worst-case.
- Gap-free results are better because they imply gap-dependent ones.[2]
- Stochastic results are usually better because they are no slower than non-stochastic ones in the worst-case.

1.1 Our Main Results

We provide algorithms `LazyEV` and `LazyCCA` that are *doubly-accelerated*, *gap-free*, and *stochastic*.[3]

For the general k-GenEV problem, our `LazyEV` can be implemented to run in time[4]

$$\widetilde{O}\Big(\frac{k\mathsf{nnz}(B)\sqrt{\kappa}}{\sqrt{\mathsf{gap}}} + \frac{k\mathsf{nnz}(A) + k^2 d}{\sqrt{\mathsf{gap}}}\Big) \quad \text{or}$$
$$\widetilde{O}\Big(\frac{k\mathsf{nnz}(B)\sqrt{\kappa}}{\sqrt{\varepsilon}} + \frac{k\mathsf{nnz}(A) + k^2 d}{\sqrt{\varepsilon}}\Big)$$

in the gap-dependent and gap-free cases respectively. Since our running time only linearly depends on $\sqrt{\kappa}$ and $\sqrt{\mathsf{gap}}$ (resp. $\sqrt{\varepsilon}$), our algorithm `LazyEV` is *doubly-accelerated*.

For the general k-CCA problem, our `LazyCCA` can be implemented to run in time

$$\widetilde{O}\Big(\frac{k\mathsf{nnz}(X,Y)\cdot\big(1 + \sqrt{\kappa'/n}\big) + k^2 d}{\sqrt{\mathsf{gap}}}\Big) \quad \text{or}$$
$$\widetilde{O}\Big(\frac{k\mathsf{nnz}(X,Y)\cdot\big(1 + \sqrt{\kappa'/n}\big) + k^2 d}{\sqrt{\varepsilon}}\Big)$$

in the gap-dependent and gap-free cases respectively. Here, $\mathsf{nnz}(X,Y) = \mathsf{nnz}(X) + \mathsf{nnz}(Y)$ and $\kappa' = \frac{2\max_i\{\|X_i\|^2, \|Y_i\|^2\}}{\lambda_{\min}(\mathrm{diag}\{S_{xx}, S_{yy}\})}$ where X_i or Y_i is the i-th row vector of X or Y. Therefore, our algorithm `LazyCCA` is *doubly-accelerated* and *stochastic*.

We fully compare our results with prior work in Table 2 (for $k = 1$) and Table 1 (for $k \ge 1$), and summarize our main contributions:

- For $k > 1$, we outperform all relevant prior works (see Table 1). Moreover, no known method was doubly-accelerated even in the non-stochastic setting.
- For $k \ge 1$, we obtain the first gap-free running time.
- Even for $k = 1$, we outperform most of the state-of-the-arts (see Table 2).

Note that for CCA with $k > 1$, previous result CCALin only outputs the subspace spanned by the top k correlation vectors but does not identify which vector gives the highest correlation and so on. Our `LazyCCA` provides per-vector

[1] Note that a similar problem can be also asked for k-GenEV when A and B are both given in their covariance matrix forms. We refrain from doing it in this paper for notational simplicity.

[2] If a method depends on $1/\varepsilon$ then one can choose $\varepsilon = \mathsf{gap}$ and this translates to a gap-dependent running time.

[3] Recalling Footnote 1, for notational simplicity, we only state our k-GenEV result in non-stochastic running time.

[4] Throughout the paper, we use the $\widetilde{O}$ notation to hide polylogarithmic factors with respect to $\kappa, 1/\mathsf{gap}, 1/\varepsilon, d, n$. We use $\mathsf{nnz}(M)$ to denote the time needed to multiply M to a vector.

Problem	Paper	Running time (× for outperformed)	gap-free?	negative EV?
k-GenEV	GenELin (Ge et al., 2016)	$\widetilde{O}\big(\frac{k\mathsf{nnz}(B)\sqrt{\kappa_B}}{\mathsf{gap}} + \frac{k\mathsf{nnz}(A)+k^2 d}{\mathsf{gap}}\big)$ ×	no	no
	`LazyEV` **Theorem 4.3**	$\widetilde{O}\big(\frac{k\mathsf{nnz}(B)\sqrt{\kappa_B}}{\sqrt{\mathsf{gap}}} + \frac{k\mathsf{nnz}(A)+k^2 d}{\sqrt{\mathsf{gap}}}\big)$	no	yes
	`LazyEV` **Theorem 4.4**	$\widetilde{O}\big(\frac{k\mathsf{nnz}(B)\sqrt{\kappa_B}}{\sqrt{\varepsilon}} + \frac{k\mathsf{nnz}(A)+k^2 d}{\sqrt{\varepsilon}}\big)$	yes	yes

Problem	Paper	Running time (× for outperformed)	gap-free?	stochastic?
k-CCA	AppGrad (Ma et al., 2015)	$\widetilde{O}\big(\frac{k\mathsf{nnz}(X,Y)\cdot\kappa+k^2 d}{\mathsf{gap}}\big)$ (local conv.) ×	no	no
	CCALin (Ge et al., 2016)	$\widetilde{O}\big(\frac{k\mathsf{nnz}(X,Y)\cdot\sqrt{\kappa}+k^2 d}{\mathsf{gap}}\big)$ ×	no	no
	CCALin (Ge et al., 2016)	$\widetilde{O}\big(\frac{k\mathsf{nnz}(X,Y)\cdot\big(1+\sqrt{\kappa'/n}\big)+k^2 d}{\mathsf{gap}}\big)$ ×	no	yes
	`LazyCCA` **(arXiv version)**	$\widetilde{O}\big(\frac{k\mathsf{nnz}(X,Y)\cdot\big(1+\sqrt{\kappa'/n}\big)+k^2 d}{\sqrt{\mathsf{gap}}}\big)$	no	yes
	`LazyCCA` **(arXiv version)**	$\widetilde{O}\big(\frac{k\mathsf{nnz}(X,Y)\cdot\big(1+\sqrt{\kappa'/n}\big)+k^2 d}{\sqrt{\varepsilon}}\big)$	yes	yes
	`LazyCCA` **(arXiv version)**	$\widetilde{O}\big(k\mathsf{nnz}(X,Y)\cdot\big(1+\frac{\sqrt{\kappa'}}{\sqrt{\mathsf{gap}\cdot\sigma_k}\cdot(\mathsf{nnz}(X,Y)/kd)^{1/4}}\big)\big)$	no	doubly
	`LazyCCA` **(arXiv version)**	$\widetilde{O}\big(k\mathsf{nnz}(X,Y)\cdot\big(1+\frac{\sqrt{\kappa'}}{\sqrt{\varepsilon\cdot\sigma_k}\cdot(\mathsf{nnz}(X,Y)/kd)^{1/4}}\big)\big)$	yes	doubly

Table 1: Performance comparison on k-GenEV and k-CCA.
In GenEV, $\mathsf{gap} = \frac{\lambda_k - \lambda_{k+1}}{\lambda_k} \in [0,1]$ and $\kappa_B = \frac{\lambda_{\max}(B)}{\lambda_{\min}(B)} > 1$.
In CCA, $\mathsf{gap} = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k} \in [0,1]$, $\kappa = \frac{\lambda_{\max}(\mathrm{diag}\{S_{xx}, S_{yy}\})}{\lambda_{\min}(\mathrm{diag}\{S_{xx}, S_{yy}\})} > 1$, $\kappa' = \frac{2\max_i\{\|X_i\|^2, \|Y_i\|^2\}}{\lambda_{\min}(\mathrm{diag}\{S_{xx}, S_{yy}\})} \in [\kappa, 2n\kappa]$, and $\sigma_k \in [0,1]$.

Remark 1. Stochastic methods depend on a modified condition number κ'. The reason $\kappa' \in [\kappa, 2n\kappa]$ is in Fact 2.5.

Remark 2. All non-stochastic CCA methods in this table have been outperformed because $1 + \sqrt{\kappa'/n} \le O(\kappa)$.

Remark 3. Doubly-stochastic methods are not necessarily interesting. We discuss them in Section 1.2.

guarantees on all the top k correlation vectors.

1.2 Our Side Results on Doubly-Stochastic Methods

Recall that when considering acceleration, there are two parameters κ and $1/\mathsf{gap}$. One can also design stochastic methods with respect to both parameters κ and $1/\mathsf{gap}$, meaning that

$$\text{with a running time proportional to} \quad 1 + \frac{\sqrt{\kappa'/n^c}}{\sqrt{\mathsf{gap}}}$$

instead of $\frac{1+\sqrt{\kappa'/n}}{\sqrt{\mathsf{gap}}}$ (stochastic) or $\frac{\sqrt{\kappa}}{\sqrt{\mathsf{gap}}}$ (non-stochastic). The constant c is usually $1/2$. We call such methods *doubly-stochastic*.

Unfortunately, doubly-stochastic methods are usually slower than stochastic ones. Take 1-CCA as an example. The best stochastic running time (obtained exclusively by us) for 1-CCA is $\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{1+\sqrt{\kappa'/n}}{\sqrt{\mathsf{gap}}}\big)$. In contrast, if one uses a doubly-stochastic method —either (Wang et al., 2016) or our `LazyCCA`— the running time becomes $\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(1+\frac{\sqrt{\kappa'}/n^{1/4}}{\sqrt{\mathsf{gap}\cdot\sigma_1}}\big)$. Therefore, for 1-CCA,

$$\boxed{\text{doubly-stochastic methods are faster than stochastic ones} \quad \textbf{\textit{only when}} \quad \frac{\kappa'}{\sigma_1} \le o(n^{1/2})\ .}$$

The above condition is usually *not* satisfied. For instance,

- κ' is usually around n for most interesting data-sets, cf. the experiments of (Shalev-Shwartz & Zhang, 2014);
- κ' is between $n^{1/2}$ and $100n$ in all the CCA experiments of (Wang et al., 2016); and
- by Fact 2.5 it satisfies $\kappa' \ge d$ so κ' cannot be smaller than $o(n^{1/2})$ unless $d \ll n^{1/2}$.[5] Even worse, parameter $\sigma_1 \in [0,1]$ is usually much smaller than 1. Note that σ_1 is scaling invariant: even if one scales X and Y up by the same factor, σ_1 remains unchanged.

Nevertheless, to compare our `LazyCCA` with *all* relevant prior works, we obtain doubly-stochastic running times for k-CCA as well. Our running time matches that of (Wang et al., 2016) when $k = 1$, and no doubly-stochastic running time for $k > 1$ was known before our work.

1.3 Other Related Works

For the easier task of PCA and SVD, the first gap-free result was obtained by Musco and Musco (Musco & Musco, 2015), the first stochastic result was obtained by Shamir (Shamir, 2015), and the first accelerated stochastic result was obtained by Garber *et al.* (Garber & Hazan, 2015; Garber et al., 2016). The shift-and-invert preconditioning technique of Garber *et al.* is also used in this paper.

For another related problem PCR (principle compo-

[5]Note that item (3) $\kappa' \ge d$ may not hold in the more general setting of CCA, see Remark A.1.

Problem	Paper	Running time (× for outperformed)	gap-free?	negative EV?
1-GenEV	GenELin (Ge et al., 2016)	$\widetilde{O}\big(\frac{\mathsf{nnz}(B)\sqrt{\kappa_B}}{\mathsf{gap}} + \frac{\mathsf{nnz}(A)}{\mathsf{gap}}\big)$ ×	no	no
	`LazyEV` **Theorem 4.3**	$\widetilde{O}\big(\frac{\mathsf{nnz}(B)\sqrt{\kappa_B}}{\sqrt{\mathsf{gap}}} + \frac{\mathsf{nnz}(A)}{\sqrt{\mathsf{gap}}}\big)$	no	yes
	`LazyEV` **Theorem 4.4**	$\widetilde{O}\big(\frac{\mathsf{nnz}(B)\sqrt{\kappa_B}}{\sqrt{\varepsilon}} + \frac{\mathsf{nnz}(A)}{\sqrt{\varepsilon}}\big)$	yes	yes
Problem	**Paper**	**Running time (× for outperformed)**	**gap-free?**	**stochastic?**
1-CCA	AppGrad (Ma et al., 2015)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{\kappa}{\mathsf{gap}}\big)$ ×	no	no
	CCALin (Ge et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{\sqrt{\kappa}}{\mathsf{gap}}\big)$ ×	no	no
	ALS (Wang et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{\sqrt{\kappa}}{\mathsf{gap}^2}\big)$ ×	no	no
	SI (Wang et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{\sqrt{\kappa}}{\sqrt{\mathsf{gap}\cdot\sigma_1}}\big)$ ×	no	no
	CCALin (Ge et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{1+\sqrt{\kappa'/n}}{\mathsf{gap}}\big)$ ×	no	yes
	ALS (Wang et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{1+\sqrt{\kappa'/n}}{\mathsf{gap}^2}\big)$ ×	no	yes
	`LazyCCA` **(arXiv version)**	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{1+\sqrt{\kappa'/n}}{\sqrt{\mathsf{gap}}}\big)$	no	yes
	`LazyCCA` **(arXiv version)**	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(\frac{1+\sqrt{\kappa'/n}}{\sqrt{\varepsilon}}\big)$	yes	yes
	SI (Wang et al., 2016)	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\Big(1+\frac{\sqrt{\kappa'}/n^{1/4}}{\sqrt{\mathsf{gap}\cdot\sigma_1}}\Big)$ ***(see Remark 3)***	no	doubly
	`LazyCCA` **(arXiv version)**	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(1+\frac{\sqrt{\kappa'}/n^{1/4}}{\sqrt{\mathsf{gap}\cdot\sigma_1}}\big)$	no	doubly
	`LazyCCA` **(arXiv version)**	$\mathsf{nnz}(X,Y)\cdot\widetilde{O}\big(1+\frac{\sqrt{\kappa'}/n^{1/4}}{\sqrt{\varepsilon\cdot\sigma_1}}\big)$	yes	doubly

Table 2: Performance comparison on 1-GenEV and 1-CCA.
In GenEV, $\mathsf{gap} = \frac{\lambda_1-\lambda_2}{\lambda_1} \in [0,1]$ and $\kappa_B = \frac{\lambda_{\max}(B)}{\lambda_{\min}(B)} > 1$.
In CCA, $\mathsf{gap} = \frac{\sigma_1-\sigma_2}{\sigma_1} \in [0,1]$, $\kappa = \frac{\lambda_{\max}(\mathrm{diag}\{S_{xx},S_{yy}\})}{\lambda_{\min}(\mathrm{diag}\{S_{xx},S_{yy}\})} > 1$, $\kappa' = \frac{2\max_i\{\|X_i\|^2,\|Y_i\|^2\}}{\lambda_{\min}(\mathrm{diag}\{S_{xx},S_{yy}\})} \in [\kappa, 2n\kappa]$, and $\sigma_1 \in [0,1]$.

Remark 1. Stochastic methods depend on modified condition number κ'; the reason $\kappa' \in [\kappa, 2n\kappa]$ is in Def. 2.4.

Remark 2. All non-stochastic CCA methods in this table have been outperformed because $1 + \sqrt{\kappa'/n} \le O(\kappa)$.

Remark 3. Doubly-stochastic methods are not necessarily interesting. We discuss them in Section 1.2.

Remark 4. Some CCA methods have a running time dependency on $\sigma_1 \in [0,1]$, and this is intrinsic and *cannot* be removed. In particular, if we scale the data matrix X and Y, the value σ_1 stays the same.

Remark 5. The only (non-doubly-stochastic) doubly-accelerated method before our work is SI (Wang et al., 2016) (for 1-CCA only). Our `LazyEV` is faster than theirs by a factor $\Omega(\sqrt{n\kappa/\kappa'} \times \sqrt{1/\sigma_1})$. Here, $n\kappa/\kappa' \ge 1/2$ and $1/\sigma_1 \ge 1$ are two scaling-invariant quantities usually much greater than 1.

nent regression), we recently obtained an accelerated method (Allen-Zhu & Li, 2017) as opposed the previously non-accelerated one (Frostig et al., 2016); however, the acceleration techniques there are not relevant to this paper.

For GenEV and CCA, many scalable algorithms have been designed recently (Ma et al., 2015; Wang & Livescu, 2015; Michaeli et al., 2015; Witten et al., 2009; Lu & Foster, 2014). However, as summarized by the authors of CCALin, these cited methods are more or less heuristics and do not have provable guarantees. Furthermore, for $k > 1$, the AppGrad method (Ma et al., 2015) only provides local convergence guarantees and thus requires a warm-start whose computational complexity is not discussed in their paper.

Finally, our algorithms on GenEV and CCA are based on finding vectors one-by-one, which is advantageous in practice because one does not need k to be known and can stop the algorithm whenever the eigenvalues (or correlation values) are too small. Known approaches for $k > 1$ cases (such as GenELin, CCALin, AppGrad) find all k vectors at once, therefore requiring k to be known beforehand. As a separate note, these known approaches do not need the user to know the desired accuracy *a priori* but our `LazyEV` and `LazyCCA` algorithms do.

2 Preliminaries

We denote by $\|x\|$ or $\|x\|_2$ the Euclidean norm of vector x. We denote by $\|A\|_2$, $\|A\|_F$, and $\|A\|_{S_q}$ respectively the spectral, Frobenius, and Schatten q-norm of matrix A (for $q \geq 1$). We write $A \succeq B$ if A, B are symmetric and $A - B$ is positive semi-definite (PSD), and write $A \succ B$ if A, B are symmetric but $A - B$ is positive definite (PD). We denote by $\lambda_{\max}(M)$ and $\lambda_{\min}(M)$ the largest and smallest eigenvalue of a symmetric matrix M, and by κ_M the condition number $\lambda_{\max}(M)/\lambda_{\min}(M)$ of a PSD matrix M.

Throughout this paper, we use $\mathsf{nnz}(M)$ to denote the time to multiply matrix M to any arbitrary vector. For two matrices X, Y, we denote by $\mathsf{nnz}(X,Y) = \mathsf{nnz}(X) + \mathsf{nnz}(Y)$, and by X_i or Y_i the i-th row vector of X or Y. We also use $\mathsf{poly}(x_1, x_2, \dots, x_t)$ to represent a quantity that is asymptotically at most polynomial in terms of variables $x_1, \dots, x_t$. Given a column orthonormal matrix $U \in \mathbb{R}^{n\times k}$, we denote by $U^\perp \in \mathbb{R}^{n\times(n-k)}$ the column orthonormal matrix consisting of an arbitrary basis in the space orthogonal to the span of U's columns.

Given a PSD matrix B and a vector v, $v^\top B v$ is the B-seminorm of v. Two vectors v, w are B-orthogonal if $v^\top B w = 0$. We denote by B^{-1} the Moore-Penrose pseudoinverse of B if B is not invertible, and by $B^{1/2}$ the matrix square root of B (satisfying $B^{1/2} \succeq 0$). All occurrences of B^{-1}, $B^{1/2}$ and $B^{-1/2}$ are for analysis purpose only. Our final algorithms only require multiplications of B to vectors.

Definition 2.1 (GenEV). *Given symmetric matrices $A, B \in \mathbb{R}^{d\times d}$ where B is positive definite. The **generalized eigenvectors** of A with respect to B are $v_1, \dots, v_d$, where each v_i is*

$$v_i \in \operatorname*{arg\,max}_{v\in\mathbb{R}^d} \left\{ \left|v^\top A v\right| \ \text{s.t.}\ \begin{array}{l} v^\top B v = 1 \\ v^\top B v_j = 0 \ \forall j \in [i-1] \end{array} \right\}$$

*The **generalized eigenvalues** $\lambda_1, \dots, \lambda_d$ satisfy $\lambda_i = v_i^\top A v_i$ which can be negative.*

Following (Wang et al., 2016; Garber & Hazan, 2015), we assume *without loss of generality* that $\lambda_i \in [-1, 1]$.

Definition 2.2 (CCA). *Given $X \in \mathbb{R}^{n\times d_x}, Y \in \mathbb{R}^{n\times d_y}$, letting $S_{xx} = \frac{1}{n}X^\top X$, $S_{xy} = \frac{1}{n}X^\top Y$, $S_{yy} = \frac{1}{n}Y^\top Y$, the **canonical-correlation vectors** are $\{(\phi_i, \psi_i)\}_{i=1}^r$ where $r = \min\{d_x, d_y\}$ and for all $i \in [r]$:*

$$(\phi_i, \psi_i) \in \operatorname*{arg\,max}_{\phi\in\mathbb{R}^{d_x}, \psi\in\mathbb{R}^{d_y}} \left\{ \phi^\top S_{xy} \psi \quad \text{such that} \right.$$

$$\left. \left\{ \begin{array}{l} \phi^\top S_{xx}\phi = 1 \ \wedge\ \phi^\top S_{xx}\phi_j = 0 \ \forall j \in [i-1] \\ \psi^\top S_{yy}\psi = 1 \ \wedge\ \psi^\top S_{yy}\psi_j = 0 \ \forall j \in [i-1] \end{array} \right\} \right\}$$

*The corresponding **canonical-correlation coefficients** $\sigma_1, \dots, \sigma_r$ satisfy $\sigma_i = \phi_i^\top S_{xy}\psi_i \in [0, 1]$.*

We emphasize that σ_i always lies in $[0, 1]$ and is scaling-invariant. When dealing with a CCA problem, we also denote by $d = d_x + d_y$.

Lemma 2.3 (CCA to GenEV). *Given a CCA problem with matrices $X \in \mathbb{R}^{n\times d_x}, Y \in \mathbb{R}^{n\times d_y}$, let the canonical-correlation vectors and coefficients be $\{(\phi_i, \psi_i, \sigma_i)\}_{i=1}^r$ where $r = \min\{d_x, d_y\}$. Define $A = \begin{pmatrix} 0 & S_{xy} \\ S_{xy}^\top & 0 \end{pmatrix}$ and $B = \begin{pmatrix} S_{xx} & 0 \\ 0 & S_{yy} \end{pmatrix}$. Then, the GenEV problem of A with respect to B has $2r$ eigenvalues $\{\pm\sigma_i\}_{i=1}^r$ and corresponding generalized eigenvectors $\left\{ \begin{pmatrix} \phi_i \\ \psi_i \end{pmatrix}, \begin{pmatrix} -\phi_i \\ \psi_i \end{pmatrix} \right\}_{i=1}^n$. The remaining $d_x + d_y - 2r$ eigenvalues are zeros.*

Definition 2.4. *In CCA, let A and B be as defined in Lemma 2.3. We define condition numbers*

$$\kappa \stackrel{\text{def}}{=} \kappa_B = \tfrac{\lambda_{\max}(B)}{\lambda_{\min}(B)} \text{ and } \kappa' \stackrel{\text{def}}{=} \tfrac{2\max_i\{\|X_i\|^2, \|Y_i\|^2\}}{\lambda_{\min}(B)} \ .$$

Fact 2.5. *$\kappa' \in [\kappa, 2n\kappa]$ and $\kappa' \geq d$. (See full version.)*

Lemma 2.6. *Given matrices $X \in \mathbb{R}^{n\times d_x}, Y \in \mathbb{R}^{n\times d_y}$, let A and B be as defined in Lemma 2.3. For every $w \in \mathbb{R}^d$, the $\mathtt{Katyusha}$ method (Allen-Zhu, 2017) finds a vector $w' \in \mathbb{R}^d$ satisfying $\|w' - B^{-1}Aw\| \leq \varepsilon$ in time*

$$O\Big(\mathsf{nnz}(X,Y) \cdot \big(1 + \sqrt{\kappa'/n}\big) \cdot \log\frac{\kappa\|w\|^2}{\varepsilon}\Big) \ .$$

3 Leading Eigenvector via Two-Sided Shift-and-Invert

We introduce $\mathtt{AppxPCA}^\pm$, the multiplicative approximation algorithm for computing the *two-sided* leading eigenvector of a symmetric matrix. $\mathtt{AppxPCA}^\pm$ uses the shift-and-invert framework (Garber & Hazan, 2015; Garber et al., 2016), and shall become our building block for the $\mathtt{LazyEV}$ and $\mathtt{LazyCCA}$ algorithms in the subsequent sections.

Our pseudo-code Algorithm 1 is a modification of Algorithm 5 in (Garber & Hazan, 2015), and reduces the eigenvector problem to oracle calls to an arbitrary matrix inversion oracle $\mathcal{A}$. The main differences between $\mathtt{AppxPCA}^\pm$ and (Garber & Hazan, 2015) are two-fold.

First, given a symmetric matrix M, $\mathtt{AppxPCA}^\pm$ simultaneously considers an upper-bounding shift together with a lower-bounding shift, and try to perform power methods with respect to $(\lambda I - M)^{-1}$ and $(\lambda I + M)^{-1}$. This allows us to determine approximately how close λ is to the largest *and* the smallest eigenvalues of M, and decrease λ accordingly. In the end, $\mathtt{AppxPCA}^\pm$ outputs an approximate eigenvector of M that corresponds to a negative eigenvalue if needed. Second, we provide a multiplicative-error guarantee rather than additive as appeared in (Garber & Hazan, 2015). Without such guarantee, our final running time will depend on $\frac{1}{\mathsf{gap}\cdot\lambda_{\max}(M)}$ rather than $\frac{1}{\mathsf{gap}}$.[6]

[6]This is why the SI method of (Wang et al., 2016) also uses

Algorithm 1 $\texttt{AppxPCA}^{\pm}(\mathcal{A}, M, \delta_\times, \varepsilon, p)$

```
Input: $\mathcal{A}$, an approximate matrix inversion method; $M \in \mathbb{R}^{d\times d}$, a symmetric matrix satisfying $-I \preceq M \preceq I$; $\delta_\times \in (0, 0.5]$, a multiplicative error; $\varepsilon \in (0,1)$, a numerical accuracy parameter; and $p \in (0,1)$, the confidence parameter.
1: $\widehat{w}_0 \leftarrow \texttt{RanInit}(d)$; $s \leftarrow 0$; $\lambda^{(0)} \leftarrow 1+\delta_\times$;    ◇ $\widehat{w}_0$ is a random unit vector, see Def. 3.2
2: $m_1 \leftarrow \lceil 4\log\big(\frac{288d\theta}{p^2}\big)\rceil$, $m_2 \leftarrow \lceil \log\big(\frac{36d\theta}{p^2\varepsilon}\big)\rceil$;    ◇ $\theta$ is the parameter of RanInit, see Def. 3.2
3: $\widetilde{\varepsilon}_1 \leftarrow \frac{1}{64m_1}\big(\frac{\delta_\times}{48}\big)^{m_1}$ and $\widetilde{\varepsilon}_2 \leftarrow \frac{\varepsilon}{8m_2}\big(\frac{\delta_\times}{48}\big)^{m_2}$
4: repeat    ◇ $m_1 = T^{\mathrm{PM}}(8, 1/32, p)$ and $m_2 = T^{\mathrm{PM}}(2, \varepsilon/4, p)$, see Lemma B.1
5:     $s \leftarrow s+1$;
6:     for $t = 1$ to $m_1$ do
7:         Apply $\mathcal{A}$ to find $\widehat{w}_t$ satisfying $\big\|\widehat{w}_t - (\lambda^{(s-1)}I - M)^{-1}\widehat{w}_{t-1}\big\| \le \widetilde{\varepsilon}_1$;
8:     $w_a \leftarrow \widehat{w}_{m_1}/\|\widehat{w}_{m_1}\|$;    ◇ $w_a$ is roughly $(\lambda^{(s-1)}I - M)^{-m_1}\widehat{w}_0$ then normalized
9:     Apply $\mathcal{A}$ to find $v_a$ satisfying $\big\|v_a - (\lambda^{(s-1)}I - M)^{-1}w_a\big\| \le \widetilde{\varepsilon}_1$;
10:    for $t = 1$ to $m_1$ do
11:        Apply $\mathcal{A}$ to find $\widehat{w}_t$ satisfying $\big\|\widehat{w}_t - (\lambda^{(s-1)}I + M)^{-1}\widehat{w}_{t-1}\big\| \le \widetilde{\varepsilon}_1$;
12:    $w_b \leftarrow \widehat{w}_{m_1}/\|\widehat{w}_{m_1}\|$;    ◇ $w_b$ is roughly $(\lambda^{(s-1)}I + M)^{-m_1}\widehat{w}_0$ then normalized
13:    Apply $\mathcal{A}$ to find $v_b$ satisfying $\big\|v_b - (\lambda^{(s-1)}I + M)^{-1}w_b\big\| \le \widetilde{\varepsilon}_1$;
14:    $\Delta^{(s)} \leftarrow \frac{1}{2}\cdot\frac{1}{\max\{w_a^\top v_a, w_b^\top v_b\} - \widetilde{\varepsilon}_1}$ and $\lambda^{(s)} \leftarrow \lambda^{(s-1)} - \frac{\Delta^{(s)}}{2}$;
15: until $\Delta^{(s)} \le \frac{\delta_\times \lambda^{(s)}}{12}$
16: $f \leftarrow s$;
17: if the last $w_a^\top v_a \ge w_b^\top v_b$ then
18:     for $t = 1$ to $m_2$ do
19:         Apply $\mathcal{A}$ to find $\widehat{w}_t$ satisfying $\big\|\widehat{w}_t - (\lambda^{(f)}I - M)^{-1}\widehat{w}_{t-1}\big\| \le \widetilde{\varepsilon}_2$;
20:     return $(+, w)$ where $w \stackrel{\mathrm{def}}{=} \widehat{w}_{m_2}/\|\widehat{w}_{m_2}\|$.
21: else
22:     for $t = 1$ to $m_2$ do
23:         Apply $\mathcal{A}$ to find $\widehat{w}_t$ satisfying $\big\|\widehat{w}_t - (\lambda^{(f)}I + M)^{-1}\widehat{w}_{t-1}\big\| \le \widetilde{\varepsilon}_2$;
24:     return $(-, w)$ where $w \stackrel{\mathrm{def}}{=} \widehat{w}_{m_2}/\|\widehat{w}_{m_2}\|$.
25: end if
```

We prove in full version the following theorem:

Theorem 3.1 ($\texttt{AppxPCA}^{\pm}$, informal). *Let $M \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigenvalues $1 \ge \lambda_1 \ge \cdots \ge \lambda_d \ge -1$ and eigenvectors $u_1, \dots, u_d$. Let $\lambda^* = \max\{\lambda_1, -\lambda_d\}$. With probability at least $1-p$, $\texttt{AppxPCA}^{\pm}$ produces a pair (sgn, w) satisfying*

- *if $sgn = +$, then w is an approx. positive eigenvector:*
$$w^\top M w \ge \Big(1 - \frac{\delta_\times}{2}\Big)\lambda^* \bigwedge \sum_{\substack{i\in[d] \\ \lambda_i \le (1-\delta_\times/2)\lambda^*}} (w^\top u_i)^2 \le \varepsilon$$
- *if $sgn = -$, then w is an approx. negative eigenvector:*
$$w^\top M w \le -\Big(1 - \frac{\delta_\times}{2}\Big)\lambda^* \bigwedge \sum_{\substack{i\in[d] \\ \lambda_i \ge -(1-\delta_\times/2)\lambda^*}} (w^\top u_i)^2 \le \varepsilon$$

The number of oracle calls to $\mathcal{A}$ is $\widetilde{O}(\log(1/\delta_\times))$, and each time we call $\mathcal{A}$ it satisfies

- $\frac{\lambda_{\max}(\lambda^{(s)}I - M)}{\lambda_{\min}(\lambda^{(s)}I - M)}, \frac{\lambda_{\max}(\lambda^{(s)}I + M)}{\lambda_{\min}(\lambda^{(s)}I + M)} \in [1, \frac{96}{\delta_\times}]$ *and*
- $\frac{1}{\lambda_{\min}(\lambda^{(s)}I - M)}, \frac{1}{\lambda_{\min}(\lambda^{(s)}I + M)} \le \frac{48}{\delta_\times \lambda^*}$.

We remark here that, unlike the original shift-and-invert method which chooses a random (Gaussian) unit vector in Line 1 of $\texttt{AppxPCA}^{\pm}$, we have allowed this initial vector to be generated from an arbitrary θ-conditioned random vector generator (for later use), defined as follows:

Definition 3.2. *An algorithm $\texttt{RanInit}(d)$ is a θ-conditioned random vector generator if $w = \texttt{RanInit}(d)$ is a d-dimensional unit vector and, for every $p \in (0,1)$, every unit vector $u \in \mathbb{R}^d$, with probability at least $1-p$, it satisfies $(u^\top w)^2 \le \frac{p^2\theta}{9d}$.*

This modification is needed in order to obtain our efficient implementations of GenEV and CCA. One can construct a θ-conditioned random vector generator as follows:

Proposition 3.3. *Given a PSD matrix $B \in \mathbb{R}^{d\times d}$, if we set $\texttt{RanInit}(d) \stackrel{\mathrm{def}}{=} \frac{B^{1/2}v}{(v^\top B v)^{0.5}}$ where v is a random Gaussian vector, then $\texttt{RanInit}(d)$ is a θ-conditioned random vector*

shift-and-invert but depends on $\frac{1}{\mathsf{gap}\cdot\sigma_1}$ in Table 2.

generator for $\theta = \kappa_B$.

4 LazyEV: Generalized Eigendecomposition

In this section, we construct an algorithm `LazyEV` that, given symmetric matrix $M \in \mathbb{R}^{d\times d}$, computes approximately the k leading eigenvectors of M that have the largest *absolute* eigenvalues. Then, for the original k-GenEV problem, we set $M = B^{-1/2}AB^{-1/2}$ and run `LazyEV`. This is our plan to find the top k leading generalized eigenvectors of A with respect to B.

Our algorithm `LazyEV` is formally stated in Algorithm 2. The algorithm applies k times $\texttt{AppxPCA}^{\pm}$, each time computing an approximate leading eigenvector of M with a multiplicative error $\delta_\times/2$, and projects the matrix M into the orthogonal space with respect to the obtained leading eigenvector. We state our main approximation theorem below.

Theorem 4.1 (informal)**.** *Let $M \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigenvalues $\lambda_1, \dots, \lambda_d \in [-1, 1]$ and corresponding eigenvectors $u_1, \dots, u_d$, and $|\lambda_1| \ge \cdots \ge |\lambda_d|$.*

If $\varepsilon_{\mathsf{pca}}$ is sufficiently small,[7] *`LazyEV` outputs a (column) orthonormal matrix $V_k = (v_1, \dots, v_k) \in \mathbb{R}^{d\times k}$ which, with probability at least $1 - p$, satisfies:*

(a) $\|V_k^\top U\|_2 \le \varepsilon$ where $U = (u_j, \dots, u_d)$ and j is the smallest index satisfying $|\lambda_j| \le (1 - \delta_\times)\lambda_k$.

(b) For every $i \in [k]$, $(1-\delta_\times)|\lambda_i| \le |v_i^\top M v_i| \le \frac{1}{1-\delta_\times}|\lambda_i|$.

Above, property (a) ensures the k columns of V_k have negligible correlation with the eigenvectors of M whose absolute eigenvalues are $\le (1 - \delta_\times)\lambda_k$; property (b) ensures the Rayleigh quotients $v_i^\top M v_i$ are all correct up to a $1 \pm \delta_\times$ error. We in fact have shown two more useful properties in the full version that may be of independent interest.

The next theorem states that, if $M = B^{-1/2}AB^{-1/2}$, our `LazyEV` can be implemented without the necessity to compute $B^{1/2}$ or $B^{-1/2}$.

Theorem 4.2 (running time)**.** *Let $A, B \in \mathbb{R}^{d\times d}$ be two symmetric matrices satisfying $B \succ 0$ and $-B \preceq A \preceq B$. Suppose $M = B^{-1/2}AB^{-1/2}$ and $\texttt{RanInit}(d)$ is defined in Proposition 3.3 with respect to B. Then, the computation of $\overline{V} \leftarrow B^{-1/2}\texttt{LazyEV}(\mathcal{A}, M, k, \delta_\times, \varepsilon_{\mathsf{pca}}, p)$ can be implemented to run in time*

- $\widetilde{O}\Big(\frac{k\mathsf{nnz}(B)+k^2d+k\Upsilon}{\sqrt{\delta_\times}}\Big)$ *where Υ is the time to multiply $B^{-1}A$ to a vector, or*
- $\widetilde{O}\Big(\frac{k\sqrt{\kappa_B}\mathsf{nnz}(B)+k\mathsf{nnz}(A)+k^2d}{\sqrt{\delta_\times}}\Big)$ *if we use Conjugate gradient to multiply $B^{-1}A$ to a vector.*

[7]Meaning $\varepsilon_{\mathsf{pca}} \le O\big(\mathsf{poly}(\varepsilon, \delta_\times, \frac{|\lambda_1|}{|\lambda_{k+1}|}, \frac{1}{d})\big)$. The complete specifications of $\varepsilon_{\mathsf{pca}}$ is included in the full version. Since our final running time only depends on $\log(1/\varepsilon_{\mathsf{pca}})$, we have not attempted to improve the constants in this polynomial dependency.

Choosing parameter $\delta_\times$ as either gap or ε, our two main theorems above immediately imply the following results for the k-GenEV problem: (proved in full version)

Theorem 4.3 (gap-dependent GenEV, informal)**.** *Let $A, B \in \mathbb{R}^{d\times d}$ be two symmetric matrices satisfying $B \succ 0$ and $-B \preceq A \preceq B$. Suppose the generalized eigenvalue and eigenvector pairs of A with respect to B are $\{(\lambda_i, u_i)\}_{i=1}^d$, and it satisfies $1 \ge |\lambda_1| \ge \cdots \ge |\lambda_d|$. Then, `LazyEV` outputs $\overline{V}_k \in \mathbb{R}^{d\times k}$ satisfying*

$$\overline{V}_k^\top B\overline{V}_k = I \quad \text{and} \quad \|\overline{V}_k^\top B\overline{W}\|_2 \le \varepsilon$$
$$\text{in time} \quad \widetilde{O}\Big(\frac{k\sqrt{\kappa_B}\mathsf{nnz}(B) + k\mathsf{nnz}(A) + k^2d}{\sqrt{\mathsf{gap}}}\Big)$$

Here, $\overline{W} = (u_{k+1}, \dots, u_d)$ and $\mathsf{gap} = \frac{|\lambda_k|-|\lambda_{k+1}|}{|\lambda_k|}$.

Theorem 4.4 (gap-free GenEV, informal)**.** *In the same setting as Theorem 4.3, our `LazyEV` outputs $\overline{V}_k = (\overline{v}_1, \dots, \overline{v}_k) \in \mathbb{R}^{d\times k}$ satisfying $\overline{V}_k^\top B\overline{V}_k = I$ and*

$$\forall s \in [k]\colon \big|\overline{v}_s^\top A\overline{v}_s\big| \in \Big[(1-\varepsilon)|\lambda_s|, \frac{|\lambda_s|}{1-\varepsilon}\Big]$$
$$\text{in time} \quad \widetilde{O}\Big(\frac{k\sqrt{\kappa_B}\mathsf{nnz}(B) + k\mathsf{nnz}(A) + k^2d}{\sqrt{\varepsilon}}\Big) \ .$$

5 Ideas Behind Theorems 4.1 and 4.2

In Section 5.1 we discuss how to ensure accuracy: that is, why does `LazyEV` guarantee to approximately find the top eigenvectors of M. In the full version of this paper, we also discuss how to implement `LazyEV` without compute $B^{1/2}$ explicitly, thus proving Theorem 4.2.

5.1 Ideas Behind Theorem 4.1

Our approximation guarantee in Theorem 4.1 is a natural generalization of the recent work on fast iterative methods to find the top k eigenvectors of a PSD matrix M (Allen-Zhu & Li, 2016). That method is called `LazySVD` and we summarize it as follows.

At a high level, `LazySVD` finds the top k eigenvectors one-by-one and *approximately*. Starting with $M_0 = M$, in the s-th iteration where $s \in [k]$, `LazySVD` computes approximately the leading eigenvector of matrix M_{s-1} and call it v_s. Then, `LazySVD` projects $M_s \leftarrow (I - v_sv_s^\top)M_{s-1}(I - v_sv_s^\top)$ and proceeds to the next iteration.

While the algorithmic idea of `LazySVD` is simple, the analysis requires some careful linear algebraic lemmas. Most notably, if v_s is an approximate leading eigenvector of M_{s-1}, then one needs to prove that the small eigenvectors of M_{s-1} somehow still "embed" into that of M_s after projection. This is achieved by a gap-free variant of the Wedin theorem plus a few other technical lemmas, and we recommend interested readers to see the high-level overview section of (Allen-Zhu & Li, 2016).

Algorithm 2 $\texttt{LazyEV}(\mathcal{A}, M, k, \delta_\times, \varepsilon_{\mathsf{pca}}, p)$

Input: $\mathcal{A}$, an approximate matrix inversion method; $M \in \mathbb{R}^{d\times d}$, a matrix satisfying $-I \preceq M \preceq I$; $k \in [d]$, the desired rank; $\delta_\times \in (0,1)$, a multiplicative error; $\varepsilon_{\mathsf{pca}} \in (0,1)$, a numerical accuracy; and $p \in (0,1)$, a confidence parameter.

```
1: $M_0 \leftarrow M$; $V_0 = []$;
2: for $s = 1$ to $k$ do
3:     $(\sim, v'_s) \leftarrow \texttt{AppxPCA}^{\pm}(\mathcal{A}, M_{s-1}, \delta_\times/2, \varepsilon_{\mathsf{pca}}, p/k)$;    ◇ $v'_s$ is an approximate two-sided leading eigenvector of $M_{s-1}$
4:     $v_s \leftarrow \big((I - V_{s-1}V_{s-1}^\top)v'_s\big)/\big\|(I - V_{s-1}V_{s-1}^\top)v'_s\big\|$;    ◇ project $v'_s$ to $V_{s-1}^\perp$
5:     $V_s \leftarrow [V_{s-1}, v_s]$;
6:     $M_s \leftarrow (I - v_s v_s^\top)M_{s-1}(I - v_s v_s^\top)$    ◇ we also have $M_s = (I - V_sV_s^\top)M(I - V_sV_s^\top)$
7: end for
8: return $V_k$.
```

In this paper, to relax the assumption that M is PSD, and to find leading eigenvectors whose *absolute* eigenvalues are large, we have to make several non-trivial changes. On the algorithm side, LazyEV uses our two-sided shift-and-invert method in Section 3 to find the leading eigenvector of M_{s-1} with largest absolute eigenvalue. On the analysis side, we have to make sure all lemmas properly deal with negative eigenvalues. For instance:

- If we perform a projection $M' \leftarrow (I - vv^\top)M(I - vv^\top)$ where v correlates by at most ε with all eigenvectors of M whose absolute eigenvalues are smaller than a threshold μ, then, after the projection, we need to prove that these eigenvectors can be approximately "embedded" into the eigenspace spanned by all eigenvectors of M' whose *absolute* eigenvalues are smaller than $\mu + \tau$. The approximation of this embedding should depend on ε, μ and τ.

The full proof of Theorem 4.1 is in the arXiv version. It relies on a few matrix algebraic lemmas (including the aforementioned "embedding lemma").

6 Conclusion

In this paper we propose new iterative methods to solve the generalized eigenvector and the canonical correlation analysis problems. Our methods find the most significant k eigenvectors or correlation vectors, and have running times that linearly scales with k.

Most importantly, our methods are *doubly-accelerated*: the running times have square-root dependencies both with respect to the condition number of the matrix (i.e., κ) and with respect to the eigengap (i.e., gap). They are the first doubly-accelerated iterative methods at least for $k > 1$. They can also be made gap-free, and are the first gap-free iterative methods even for 1-GenEV or 1-CCA.

Although this is a theory paper, we believe that if implemented carefully, our methods can outperform not only previous iterative methods (such as GenELin, AppGrad, CCALin), but also the commercial mathematics libraries for sparse matrices of dimension more than $10,000$. We leave it a future work for such careful comparisons.

References

Allen-Zhu, Zeyuan. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. In *STOC*, 2017.

Allen-Zhu, Zeyuan and Li, Yuanzhi. LazySVD: Even Faster SVD Decomposition Yet Without Agonizing Pain. In *NIPS*, 2016.

Allen-Zhu, Zeyuan and Li, Yuanzhi. Faster Principal Component Regression and Stable Matrix Chebyshev Approximation. In *Proceedings of the 34th International Conference on Machine Learning*, ICML '17, 2017.

Allen-Zhu, Zeyuan and Orecchia, Lorenzo. Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent. In *Proceedings of the 8th Innovations in Theoretical Computer Science*, ITCS '17, 2017.

Allen-Zhu, Zeyuan and Yuan, Yang. Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives. In *ICML*, 2016.

Allen-Zhu, Zeyuan, Richtárik, Peter, Qu, Zheng, and Yuan, Yang. Even faster accelerated coordinate descent using non-uniform sampling. In *ICML*, 2016.

Arora, Sanjeev, Rao, Satish, and Vazirani, Umesh V. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2), 2009.

Aujol, J-F and Dossal, Ch. Stability of over-relaxations for the forward-backward algorithm, application to fista. *SIAM Journal on Optimization*, 25(4):2408–2433, 2015.

Axelsson, Owe. A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT Numerical Mathematics*, 25(1):165–187, 1985.

Chaudhuri, Kamalika, Kakade, Sham M, Livescu, Karen, and Sridharan, Karthik. Multi-view clustering via canonical correlation analysis. In *ICML*, pp. 129–136, 2009.

Dhillon, Paramveer, Foster, Dean P, and Ungar, Lyle H. Multi-view learning of word embeddings via cca. In *NIPS*, pp. 199–207, 2011.

Frostig, Roy, Musco, Cameron, Musco, Christopher, and Sidford, Aaron. Principal Component Projection Without Principal Component Analysis. In *ICML*, 2016.

Garber, Dan and Hazan, Elad. Fast and simple PCA via convex optimization. *ArXiv e-prints*, September 2015.

Garber, Dan, Hazan, Elad, Jin, Chi, Kakade, Sham M., Musco, Cameron, Netrapalli, Praneeth, and Sidford, Aaron. Robust shift-and-invert preconditioning: Faster and more sample efficient algorithms for eigenvector computation. In *ICML*, 2016.

Ge, Rong, Jin, Chi, Kakade, Sham M., Netrapalli, Praneeth, and Sidford, Aaron. Efficient Algorithms for Large-scale Generalized Eigenvector Computation and Canonical Correlation Analysis. In *ICML*, 2016.

Golub, Gene H. and Van Loan, Charles F. *Matrix Computations*. The JHU Press, 4th edition, 2012. ISBN 1421407949.

Kakade, Sham M and Foster, Dean P. Multi-view regression via canonical correlation analysis. In *Learning theory*, pp. 82–96. Springer, 2007.

Karampatziakis, Nikos and Mineiro, Paul. Discriminative features via generalized eigenvectors. In *ICML*, pp. 494–502, 2014.

Lu, Yichao and Foster, Dean P. Large scale canonical correlation analysis with iterative least squares. In *NIPS*, pp. 91–99, 2014.

Ma, Zhuang, Lu, Yichao, and Foster, Dean. Finding linear structure in large datasets with scalable canonical correlation analysis. In *ICML*, pp. 169–178, 2015.

Michaeli, Tomer, Wang, Weiran, and Livescu, Karen. Nonparametric canonical correlation analysis. *arXiv preprint*, abs/1511.04839, 2015.

Musco, Cameron and Musco, Christopher. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *NIPS*, pp. 1396–1404, 2015.

Nesterov, Yurii. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, volume 269, pp. 543–547, 1983.

Shalev-Shwartz, Shai. SDCA without Duality, Regularization, and Individual Convexity. In *ICML*, 2016.

Shalev-Shwartz, Shai and Zhang, Tong. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization. In *Proceedings of the 31st International Conference on Machine Learning*, ICML 2014, pp. 64–72, 2014.

Shamir, Ohad. A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate. In *ICML*, pp. 144—-153, 2015.

Shewchuk, Jonathan Richard. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Wang, Weiran and Livescu, Karen. Large-scale approximate kernel canonical correlation analysis. *arXiv preprint*, abs/1511.04773, 2015.

Wang, Weiran, Wang, Jialei, Garber, Dan, and Srebro, Nathan. Efficient Globally Convergent Stochastic Optimization for Canonical Correlation Analysis. In *NIPS*, 2016.

Witten, Daniela M, Tibshirani, Robert, and Hastie, Trevor. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, pp. kxp008, 2009.