Column Subset Selection via Sparse Approximation of SVD

A.Çivril[a,*], M.Magdon-Ismail[b]

[a] *Meliksah University, Computer Engineering Department, Talas, Kayseri 38280 Turkey*
[b] *Rensselaer Polytechnic Institute, Computer Science Department, 110 8th Street Troy, NY 12180-3590 USA*

Abstract

Given a real matrix $A \in \mathbb{R}^{m\times n}$ of rank r, and an integer $k < r$, the sum of the outer products of top k singular vectors scaled by the corresponding singular values provide the best rank-k approximation A_k to A. When the columns of A have specific meaning, it might be desirable to find good approximations to A_k which use a small number of columns of A. This paper provides a simple greedy algorithm for this problem in Frobenius norm, with guarantees on the performance and the number of columns chosen. The algorithm selects c columns from A with $c = \tilde{O}\left(\frac{k\log k}{\epsilon^2}\eta^2(A)\right)$ such that

$$\|A - \Pi_C A\|_F \leq (1+\epsilon)\,\|A - A_k\|_F,$$

where C is the matrix composed of the c columns, Π_C is the matrix projecting the columns of A onto the space spanned by C and $\eta(A)$ is a measure related to the *coherence* in the normalized columns of A. The algorithm is quite intuitive and is obtained by combining a greedy solution to the generalization of the well known *sparse approximation problem* and an *existence* result on the possibility of sparse approximation. We provide empirical results on various specially constructed matrices comparing our algorithm with the previous deterministic approaches based on QR factorizations and a recently proposed randomized algorithm. The results indicate that in practice, the performance of the algorithm can be significantly better than the bounds suggest.

Keywords: , subset selection, SVD, sparse approximation

1. Introduction

The usual approach to find a "good" subspace that approximates the column span of a matrix A is to take the best rank k approximation $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, which minimizes the residual error with respect to any unitarily invariant norm. In some application areas such as statistical data analysis, this approach might be undesirable since the singular vector representation is not suitable to make inferences about the actual underlying data; because they are generally combinations of all the columns of A. An example of this is the micro-array data where the combinations of the column vectors have no sensible interpretation [25]. Hence, it is of practical importance to find an approximation to A_k which is composed of a small number of columns of A. The problem also bears a theoretical importance in the sense that one might want to know how well the column vectors of a matrix can represent its spectrum. This paper considers the problem of finding a small number of columns of a matrix A such that the expression $\|A - \Pi_C A\|_F$ is close to $\|A - A_k\|_F$, for a given number $k < r = rank(A)$ where $\Pi_C = CC^+$ is the matrix projecting the columns of A onto the space spanned by the columns of C.

We give a deterministic greedy algorithm for this problem which is based on the *sparse approximation* of the SVD of A. We first generalize the problem of sparse approximation [11, 27] to one of approximating a

*Corresponding author
Email addresses: acivril@meliksah.edu.tr (A.Çivril), magdon@cs.rpi.edu (M.Magdon-Ismail)

subspace. This is conceptually the same problem with the one so-called *simultaneous sparse approximation* in signal processing and approximation theory in Hilbert spaces (e.g. [26, 34]). We then propose and analyze a greedy algorithm for this problem and derive our main result in the special case where the subspace to be approximated is the space spanned by the first k left singular vectors of A. In words, the algorithm first computes the top k left singular vectors of A, and then selects columns of A in a greedy fashion so as to "fit" the space spanned by the singular vectors, appropriately scaled according to the singular values. The performance characteristics of the algorithm depend on how well the greedy algorithm approximates the optimal choice of such columns from A, and on how good the optimal columns themselves are. We combine an existence result on the quality of the optimal columns with the analysis of the greedy algorithm to arrive at the following result:

Theorem 1.1. *Given a matrix $A \in \mathbb{R}^{m\times n}$, an integer $k < r = rank(A)$ and $\epsilon < \frac{\|A_k\|_F}{\|A-A_k\|_F}$, there exists a polynomial-time algorithm which selects a column sub-matrix $C \in \mathbb{R}^{m\times c}$ of A with $c = O\left(\frac{k\log k}{\epsilon^2}\eta^2(A)\ln\left(\frac{\|A_k\|_F}{\epsilon\|A-A_k\|_F}\right)\right)$ columns such that*

$$\|A - \Pi_C A\|_F \leq (1+\epsilon)\|A - A_k\|_F,$$

where $\eta(A)$ is a measure related to the coherence in the normalized columns of A.

The requirement on ϵ is to make sure that the expression with the natural logarithm is meaningful. The term $\eta(A)$ arises from the analysis of the generalized sparse approximation problem. In our analysis, the possibility of eliminating this parameter or replacing it with a low order polynomial in k and ϵ would yield a much more desirable result. We would like to note that such input-dependent parameters naturally arise in the analysis of sparse approximation problems [34, 35]. Yet, considering the special nature of the subspace we wish to approximate, we think an improvement is possible.

1.1. Related Work

The theoretical computer science community has investigated the low-rank matrix approximation problem which asks for a k-dimensional subspace that approximates A_k in the spectral and Frobenius norm. The solutions developed thus far have mostly focused on randomized algorithms, and the set of columns chosen by these algorithms have more than k columns which is proven to contain an arbitrarily close approximation to A_k. This approximation has the nice property of having the same dimensionality with that of A_k, but cannot directly be interpreted in terms of the columns of A. The numerical linear algebra community on the other hand, implicitly provides deterministic solutions for approximating A_k in the context of rank revealing QR factorizations, which primarily aim to determine the numerical rank of A. The algorithms developed in this framework usually focus on spectral norm and they select exactly k columns providing approximations as a function of k and n. The algorithm we provide has hybrid features in the sense that it is deterministic, and the error ratio drops with increasing number of selected columns.

The seminal paper by Frieze, Kannan and Vempala [19] gives a randomized algorithm that selects a subset of columns $C \in \mathbb{R}^{m\times c}$ of A such that $\|A - \Pi_C A\|_F \leq \|A - A_k\|_F + \epsilon\|A\|_F$, where Π_C is a projection matrix obtained by the truncated SVD of C and c is a polynomial in k, $1/\epsilon$ and $1/\delta$), where δ is the failure probability of the algorithm. Subsequent work [15, 16] introduced several improvements on the dependence of c on $k, 1/\epsilon$ and $1/\delta$ also extending the analysis to the spectral norm, while Rudelson and Vershynin [29, 30], provided results of the form $\|A - \Pi_C A\|_2 \leq \|A - A_k\|_2 + \epsilon\sqrt{\|A\|_2\|A\|_F}$. Recently, the effort has been towards eliminating the additive term in the inequality thereby yielding a relative approximation in the form $\|A - \Pi_C A\|_F \leq (1+\epsilon)\|A - A_k\|_F$. Along these lines, Deshpande and Vempala [14] and Drineas et al. [17] provided algorithms with different sampling schemes attaining the $(1+\epsilon)$ approximation where the number of columns is a function of k and ϵ. Other recent approaches for the problem we consider includes random projections [31], and sampling which exploits geometric properties of high dimensional spaces [32]. [13] also considers the subspace approximation problem in general l_p norms. Achlioptas and McSherry approaches the problem by zero-ing out and quantizing the individual elements of the matrix randomly

[1]. All of these algorithms exploit the power of randomization and they introduce a trade-off between the number of columns chosen, the error parameter and the failure probability of the algorithm.

Very recently, a deterministic algorithm for matrix reconstruction was proposed by Guruswami and Sinop [22] based on carefully implementing a scheme akin to volume sampling. Their algorithm uses optimal number of columns, but the running time is $O(m^{\omega}nr \log m)$ where ω is the exponent in matrix multiplication. Compared to [22], the algorithm we present in this paper is less sophisticated and more intuitive.

The linear algebra community has developed deterministic algorithms in the framework of *rank revealing QR (RRQR) factorizations* [7] which yield some approximation guarantees in spectral norm. Given a matrix $A \in \mathbb{R}^{n\times n}$, consider the QR factorization of the form

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{1}$$

where $R_{11} \in \mathbb{R}^{k\times k}$ and $\Pi \in \mathbb{R}^{n\times n}$ is a permutation matrix. By the interlacing property of singular values (see [20]), $\sigma_k(R_{11}) \leq \sigma_k(A)$ and $\sigma_1(R_{22}) \geq \sigma_{k+1}(A)$. If the numerical rank of A is k, i.e. $\sigma_k(A) \gg \sigma_{k+1}(A)$, then one would like to find a permutation Π for which $\sigma_k(R_{11})$ is sufficiently large and $\sigma_1(R_{22})$ is sufficiently small. A QR factorization is said to be a rank revealing QR (RRQR) factorization if $\sigma_k(R_{11}) \geq \sigma_k(A)/p(k,n)$ and $\sigma_1(R_{22}) \leq \sigma_{k+1}(A)p(k,n)$, where $p(k,n)$ is a low degree polynomial in k and n.

Much research on finding RRQR factorizations has yielded improved results for $p(k,n)$ [7, 8, 9, 12, 21, 23, 28]. Tight bounds for $p(k,n)$ can be used to give deterministic low rank matrix reconstruction with respect to the spectral norm, via the following simple fact.

Theorem 1.2. *Let Π_k be the matrix of first k columns of Π in (1). Then,*

$$\|A - (A\Pi_k)(A\Pi_k)^{+}A\|_2 \leq p(k,n)\|A - A_k\|_2.$$

It is important to note that the algorithm we provide can be regarded as an analogue to the algorithm of Chan and Hansen [8] (Low-RRQR) which greedily selects the closest column to the first singular vector of the residual space at each step, starting from the original matrix A. This algorithm approximates singular vectors one by one providing a result in *spectral* norm, whereas we specifically aim at the *Frobenius* norm, and compute the whole k-dimensional best subspace at the beginning and find a sparse approximation to it.

Very recently, Boutsidis et al. [4] introduced an algorithm for the problem of selecting exactly k columns from a matrix A to approximate A_k, combining the random sampling schemes and the deterministic column pivoting strategies exploited by QR algorithms. Their algorithm provides a performance guarantee of $p(k,n) = O\left(k\sqrt{\log k}\right)$ for the Frobenius norm, with high probability.

This work is inspired by the sparse approximation problem which is an extremely active research area today (see [5] for a comprehensive survey). More specifically, we are trying to solve a *simultaneous* sparse approximation problem [10, 26, 35] as there are k *signals* to be approximated simultaneously from the *dictionary* implied by the matrix A. We essentially reformulate and attack this problem; but by trying to optimize a measure of quality directly related to matrix approximation. The algorithm we analyze is slightly different from the generalizations of Orthogonal Matching Pursuit for which several variants have been proposed [10, 11, 26, 34]. In contrast, it is a generalization of the algorithm by Natarajan [27], which was pronounced in pure linear algebraic terms.

1.2. Notation and Preliminaries

From now on $A \in \mathbb{R}^{m\times n}$ is the matrix for which we wish to find a low-rank approximation. $A_{(i)}$ denotes the i^{th} row of A for $1 \leq i \leq m$, and $A^{(j)}$, the j^{th} column of A for $1 \leq j \leq n$. A_{ij} is the element at i^{th} row and the j^{th} column. For a set of indices Λ, $\Lambda(A)$ denotes the set of columns of A with indices in Λ. Typically, we use C to denote a subset of columns of A, written $C \subset A$, i.e. C is a column sub-matrix of A. $span(C)$ denotes the subspace spanned by the column vectors in C. The Singular Value Decomposition of $A \in \mathbb{R}^{m\times n}$ of rank r is denoted by $A = U\Sigma V^T$ where $U \in \mathbb{R}^{m\times m}$ is the matrix of left singular vectors, $\Sigma \in \mathbb{R}^{m\times n}$ is the diagonal matrix containing the singular values of A in descending order, i.e. $\Sigma = diag(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ where $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r > 0$ are the singular values of A. $V \in \mathbb{R}^{n\times n}$ is the matrix of right singular vectors.

The best rank k approximation to A is $A_k = U_k \Sigma_k V_k^T$ where U_k and V_k are the first k columns of the corresponding matrices in the full SVD of A, and Σ_k is the $k \times k$ diagonal matrix of the first k singular values. The pseudo-inverse of A is denoted by $A^+ = V\Sigma^+ U^T$, where $\Sigma^+ = diag\left(\frac{1}{\sigma_1}, \ldots \frac{1}{\sigma_r}, 0, \ldots, 0\right)$. The Frobenius norm of A is $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}$, and the spectral norm of A is $\|A\|_2 = \sigma_1(A)$. We also define the maximum column norm of a matrix A, $\|A\|_{col} = \max_{i=1}^{n}\{\|A^{(i)}\|_2\}$. $S^{\perp}$ is the space orthogonal to the space spanned by the vectors in S.

2. Generalized Sparse Approximation

We first consider the problem of approximating an arbitrary subspace, which is an intuitive extension of the sparse approximation problem [5, 11, 27]. It asks for a set of smallest number of vectors from a dictionary, which defines a subspace "close enough" to a given vector. We propose the following generalization: given matrices $A \in \mathbb{R}^{m\times n}$, a set of vectors $B \in \mathbb{R}^{m\times k}$, and $\delta > 0$, find a matrix $X \in \mathbb{R}^{n\times k}$ satisfying

$$\|AX - B\|_F \leq \delta, \tag{2}$$

such that $\sum_{i=1}^{n} \nu_i(X)$ is minimum over all possible choices of X, where $\nu_i(X) = 1$ if the row $X_{(i)}$ contains non-zero entries, $\nu_i(X) = 0$ if $X_{(i)} = \overrightarrow{0}$. Intuitively, the problem asks for a minimum number of column vectors of A whose span is close to the span of B.

2.1. The Algorithm

A greedy strategy for solving this problem is to choose the column v from A at each iteration, for which $\|B^T v\|_2$ is maximum, and project the column vectors of B and the other column vectors of A onto the space orthogonal to the chosen column. The algorithm proceeds greedily on these residual matrices until the norm of the residual B drops below the required threshold δ. Naturally, if the error δ cannot be attained, the algorithm will fail after selecting a maximal independent set of columns.

Algorithm 1 A greedy algorithm for Generalized Sparse Approximation

1: **procedure** GREEDY(A, B, δ)
2: normalize each column of A to get A_0.
3: $l \leftarrow 0$, $\Lambda \leftarrow \emptyset$, $B_0 \leftarrow B$.
4: **while** $\|B_l\|_F > \delta$ **do**
5: choose $i \in \{1, \ldots, n\} - \Lambda$ such that $\|B_l^T A_l^{(i)}\|_2$ is maximum
6: $B_{l+1}^{(j)} \leftarrow B_l^{(j)} - \left({B_l^{(j)}}^T A_l^{(i)}\right) A_l^{(i)}$ for $j = 1, \ldots, k$
7: $\Lambda \leftarrow \Lambda \cup \{i\}$.
8: $A_{l+1}^{(j)} \leftarrow A_l^{(j)} - \left({A_l^{(j)}}^T A_l^{(i)}\right) A_l^{(i)}$ for $j \in \{1, \ldots, n\} - \Lambda$
9: normalize $A_{l+1}^{(j)}$ for $j \in \{1, \ldots, n\} - \Lambda$.
10: $l \leftarrow l + 1$.
11: **end while**
12: return $C = \Lambda(A)$, the selected columns.
13: **end procedure**

2.2. Implementation Details and Running Time Analysis

Line 2,8 and 9 of Greedy takes $O(mn)$ time. Line 5 takes $O(mk)$ time since $B \in \mathbb{R}^{m\times k}$. The computationally intensive part of the algorithm in the while loop is the 6th step, which takes $O(mnk)$ time with a naive implementation, since there are n matrix-vector multiplications of cost $O(mk)$. This makes a total

of $O(mnkc)$ running time complexity. We make note of a simple observation which is akin to the pivoted QR algorithms and is called a norm update: Instead of performing matrix-vector multiplications at each iteration, we remember the dot products of the chosen column with the columns of B and the other columns in A. We also introduce a matrix $D \in \mathbb{R}^{k \times n}$, where $(D_l)_{ij}$ denotes the the dot product of the i^{th} column of B and the j^{th} column of A in the l^{th} iteration. At the end of each iteration, we update D_l to get D_{l+1} where the update of each entry requires constant time. Hence, the 6th step takes $O(nk)$ time complexity for each iteration. Overall, the running time of the algorithm is $O((2mn + mk + nk)c) = O(mnc)$.

The norm update is as follows: Suppose a column vector v is chosen at iteration l. Noting that $v^T v = 1$, D_{l+1} satisfies

$$\begin{aligned}
(D_{l+1})_{ij} = {B_{l+1}^{(i)}}^T A_{l+1}^{(j)} &= \frac{\left(B_l^{(i)} - \left({B_l^{(i)}}^T v\right) v\right)^T \left(A_l^{(j)} - \left({A_l^{(j)}}^T v\right) v\right)}{\left\| A_l^{(j)} - \left({A_l^{(j)}}^T v\right) v \right\|_2} \\
&= \frac{{B_l^{(i)}}^T A_l^{(j)} + \left({B_l^{(i)}}^T v\right)\left({A_l^{(j)}}^T v\right)\left(v^T v - 2\right)}{\left\| A_l^{(j)} - \left({A_l^{(j)}}^T v\right) v \right\|_2} \\
&= \frac{(D_l)_{ij} - \left({B_l^{(i)}}^T v\right)\left({A_l^{(j)}}^T v\right)}{\left\| A_l^{(j)} - \left({A_l^{(j)}}^T v\right) v \right\|_2}.
\end{aligned}$$

The update per entry can be performed in constant time given the other values in the last expression, which are already computed. For all the operations on norms, we followed the usual Gram-Schmidt method. In practice, the exact procedure to perform such updates may require fast and numerically stable techniques such as Householder transformations, which we do not discuss in this paper. Refinements of this type can be deferred to another work specifically targeting numerical analysis and stability issues.

2.3. Performance Analysis

In this section, we provide a theorem establishing the performance characteristics of the algorithm in the general setting, i.e. where B is an arbitrary set of columns. We will essentially bound the number of columns chosen by the algorithm attaining the desired error δ. The analysis yields an approximation factor which includes a term related to the smallest singular value of a certain sub-matrix. We begin with the following definition, which provides a general upper bound for the spectral norm of the pseudo-inverse of any sub-matrix of a matrix. We would like to note that similar definitions have appeared in [34] while analyzing algorithms for the sparse approximation problem.

Definition 2.1. *[**Coherence**] Given a matrix $A \in \mathbb{R}^{m \times n}$ of rank r, let A_0 be the matrix A with normalized columns. Then, $\eta(A)$ is the maximum of the inverses of the least singular value of $m \times r$ full-rank sub-matrices of A_0. Namely,*

$$\eta(A) = \max_{\substack{C_0 \subseteq A_0 \\ C_0 \in \mathbb{R}^{m \times r} \\ rank(C_0) = r}} \frac{1}{\sigma_r(C_0)}. \tag{3}$$

Theorem 2.2. *Greedy chooses a sub-matrix C of no more than $\left\lceil 18 Opt(\delta/2)\eta^2(A) \ln\left(\frac{\|B\|_F}{\delta}\right)\right\rceil$ columns, satisfying*

$$\|CC^+B - B\|_F \leq \delta$$

where C^+ is the pseudo-inverse of C and $Opt(\delta/2)$ is the number of non-zero rows in an optimal solution for the generalized sparse approximation problem with error $\delta/2$.

We will prove the theorem in a sequence of lemmas. Most of these lemmas are generalizations of the ones in [27] and follow a similar reasoning. We extensively use the Cauchy-Schwartz inequality to carry the results in one dimensional case to the general case for which the Frobenius norm is utilized. We first provide the following notation which will also be used in the next section: Let t be the total number of iterations of Greedy. At the beginning of the l^{th} iteration of the algorithm, for $0 \leq l < t$, let Γ_l be an optimal solution to the generalized sparse approximation problem with error parameter $\delta/2$, i.e. Γ_l minimizes $\sum_{i=1}^{n} \nu_i(X)$ over $X \in \mathbb{R}^{n \times k}$ such that $\|A_l X - B_l\|_F \leq \delta/2$, where $\nu_i(X) = 1$ if the row $X_{(i)}$ contains non-zero entries, $\nu_i(X) = 0$ if $X_{(i)} = \overrightarrow{0}$. Let $\nu(\Gamma_l) = \sum_{i=1}^{n} \nu_i(\Gamma_l)$ and $Q_l = A_l \Gamma_l$. Define

$$\lambda = \max_{0 \leq l < t} \frac{\nu(\Gamma_l)\|\Gamma_l\|_F^2}{\|B_l\|_F^2}. \tag{4}$$

Assuming that the Greedy has not terminated, the following lemma states that the next step makes significant progress.

Lemma 2.3. *For the l^{th} iteration of Greedy, $\|B_l^T A_l\|_{col} \geq \frac{\|B_l\|_F^2}{2\sqrt{\nu(\Gamma_l)}\|\Gamma_l\|_F}$.*

Proof. We can write $B_l^{(j)} = \left(\sum_{i=1}^{n} A_l^{(i)} (\Gamma_l)_{ij}\right) + E^{(j)}$ for $j = 1, \ldots, k$, where $E \in \mathbb{R}^{m \times k}$ is a matrix such that $\|E\|_F \leq \delta/2$, and $\|E^{(j)}\|_2 = \delta_j/2$ for $i = 1, \ldots, k$ for which $\sum_{i=1}^{k} \delta_j^2 \leq \delta^2$. Then,

$$\|B_l\|_F^2 = \sum_{j=1}^{k} {B_l^{(j)}}^T B_l^{(j)} = \sum_{j=1}^{k} \sum_{i=1}^{n} (\Gamma_l)_{ij} {B_l^{(j)}}^T A_l^{(i)} + \sum_{j=1}^{k} {B_l^{(j)}}^T E^{(j)}. \tag{5}$$

We will first bound the double summation in the above expression.

$$\begin{aligned}
\sum_{j=1}^{k} \sum_{i=1}^{n} (\Gamma_l)_{ij} {B_l^{(j)}}^T A_l^{(i)} &\leq \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{k} (\Gamma_l)_{ij}^2 \right)^{1/2} \left(\sum_{j=1}^{k} \left({B_l^{(j)}}^T A_l^{(i)} \right)^2 \right)^{1/2} \right) \\
&\leq \max_{1 \leq i \leq n} \left\{ \left(\sum_{j=1}^{k} \left({B_l^{(j)}}^T A_l^{(i)} \right)^2 \right)^{1/2} \right\} \sum_{i=1}^{n} \left(\sum_{j=1}^{k} (\Gamma_l)_{ij}^2 \right)^{1/2} \\
&\leq \|B_l^T A_l\|_{col} \sqrt{\nu(\Gamma_l)} \|\Gamma_l\|_F.
\end{aligned}$$

The first line is due to Cauchy-Schwartz inequality. The last inequality bounds the double summation in the second line as follows. Define n dimensional vectors a and b such that $a_i = \left(\sum_{j=1}^{k} (\Gamma_l)_{ij}^2\right)^{1/2}$ and $b_i = 1$ if there exists a non-zero entry in the i^{th} row of Γ_l, $b_i = 0$ if all the elements in the i^{th} row of Γ_l are zero, for $i = 1, \ldots, n$. Then, applying Cauchy-Schwartz inequality to a and b, we obtain $\sum_{i=1}^{n} \left(\sum_{j=1}^{k} (\Gamma_l)_{ij}^2\right)^{1/2} = \sum_{i=1}^{n} a_i b_i \leq \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$. Since $\sum_{i=1}^{n} a_i^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} (\Gamma_l)_{ij}^2 = {\|\Gamma_l\|_F}^2$, and $\sum_{i=1}^{n} b_i^2 = \nu(\Gamma_l)$, we have that $\sum_{i=1}^{n} \left(\sum_{j=1}^{k} (\Gamma_l)_{ij}^2\right)^{\frac{1}{2}} \leq \sqrt{\nu(\Gamma_l)} \|\Gamma_l\|_F$.

We will now bound the second term in (5).

$$\begin{aligned}
\sum_{j=1}^{k} {B_l^{(j)}}^T E^{(j)} &\leq \sum_{j=1}^{k} \|{B_l^{(j)}}^T\|_2 \|E^{(j)}\|_2 \quad (Cauchy - Schwartz) \\
&= \frac{1}{2} \sum_{j=1}^{k} \delta_j \|{B_l^{(j)}}^T\|_2 \\
&\leq \frac{1}{2} \left(\sum_{j=1}^{k} \delta_j^2 \right)^{1/2} \left(\sum_{j=1}^{k} \|{B_l^{(j)}}^T\|_2^2 \right)^{1/2} \qquad (Cauchy - Scwartz) \\
&\leq \frac{1}{2} \delta \|B_l\|_F \\
&\leq \frac{1}{2} \|B_l\|_F^2,
\end{aligned}$$

where the last inequality is due to the fact that $\|B_l\|_F > \delta$, i.e. the algorithm is still running. Combining these bounds in (5), we have $\|B_l\|_F^2 \leq \|B_l^T A_l\|_{col} \sqrt{\nu(\Gamma_l)} \|\Gamma_l\|_F + 1/2 \|B_l\|_F^2$, which gives $\|B_l\|_F^2 \leq 2\|B_l^T A_l\|_{col} \sqrt{\nu(\Gamma_l)} \|\Gamma_l\|_F$. The lemma then immediately follows.

Thus, there exists a column in the residual A_l which will reduce the residual B_l significantly, because B_l has a large projection onto this column. Therefore, since every step of Greedy makes significant progress, there cannot be too many steps, which is the content of the next lemma.

Lemma 2.4. *$t \leq \left\lceil 8\lambda \ln\left(\frac{\|B\|_F}{\delta}\right)\right\rceil$, where t is the number of Greedy iterations.*

PROOF. Let i be the index of the chosen column at step l and let j be a column index of B. Then, by the execution of the algorithm, $B_{l+1}^{(j)} = B_l^{(j)} - \left({B_l^{(j)}}^T A_l^{(i)}\right) A_l^{(i)}$. Since $B_{l+1}^{(j)}$ is orthogonal to $A_l^{(i)}$ and $\|A_l^{(i)}\|_2 = 1$, we can write $\|B_{l+1}^{(j)}\|_2^2 = \|B_l^{(j)}\|_2^2 - |{B_l^{(j)}}^T A_l^{(i)}|^2$. Summing over all column indices of B_{l+1}, we obtain

$$\begin{aligned}
\|B_{l+1}\|_F^2 = \sum_{j=1}^{k} \|B_{l+1}^{(j)}\|_2^2 &= \sum_{j=1}^{k} \|B_l^{(j)}\|_2^2 - \sum_{j=1}^{k} |{B_l^{(j)}}^T A_l^{(i)}|^2 \\
&= \|B_l\|_F^2 - \|B_l^T A_l^{(i)}\|_2^2 \\
&= \|B_l\|_F^2 - \|B_l^T A_l\|_{col}^2 \\
&\leq \|B_l\|_F^2 - \frac{\|B_l\|_F^4}{4\nu(\Gamma_l)\|\Gamma_l\|_F^2} \quad (Lemma\, 2.3) \\
&= \|B_l\|_F^2 \left(1 - \frac{1}{4\lambda}\right) \quad (Equation\, (4)),
\end{aligned}$$

where the third line follows since the algorithm chooses i to maximize $\|B_l^T A_l^{(i)}\|_2$. Hence, $\|B_l\|_F^2 \leq (1 - 1/4\lambda)^t \|B_0\|_F^2$. Since the algorithm stops when $\|B_t\|_F^2 \leq \delta^2$, it suffices for t to satisfy $(1 - 1/4\lambda)^t \|B_0\|_F^2 \leq \delta^2$. Rearranging, and taking logarithms we obtain $t \ln(1 - 1/4\lambda) \leq \ln(\delta^2/\|B_0\|_F^2)$. Since $\ln(1 - 1/4\lambda) \leq -1/4\lambda$ from Taylor series expansion, we get that $t \geq 4\lambda \ln(\|B\|_F^2/\delta^2) = 8\lambda \ln(\|B\|_F/\delta)$ iterations are enough for Greedy to terminate.

What remains is to bound λ. First, we will bound $\|\Gamma_l\|_F$ in terms of $\|B_l\|_F$ both of which appear in the expression for λ. Let $\pi_l = \{i|(\Gamma_l)_{(i)} \neq \overrightarrow{0}\}$ be the indices of rows of Γ_l which are not all zero. Recall that these indices denote which columns are chosen by the optimal solution for A_l. Let $\tau_l = \{i_1, i_2, \ldots, i_l\}$ be the indices of the first l columns picked by the algorithm. Given an index set γ, let the set of column vectors $\{A^{(i)}|i \in \gamma\}$ be denoted by $\gamma(A)$.

Lemma 2.5. *$\pi_l(A_0) \cup \tau_l(A_0)$ is a linearly independent set for all $l \geq 0$.*

PROOF. Note that for $l = 0$, we only have $\pi_0(A_0)$ and by the definition of the optimality of U_0, this set should be linearly independent. For $l \geq 1$, we will argue by contradiction. Assume that the given set, $\pi_l(A_0) \cup \tau_l(A_0)$ is not a linearly independent set. Hence, some linear combination of some vectors from the set sum to 0. Since, by the execution of the algorithm, $\tau_l(A_0)$ is a linearly independent set, at least one of these vectors should be from $\pi_l(A_0)$, and this vector u can be written as a linear combination of some other vectors in $\pi_l(A_0) \cup \tau_l(A_0)$. To this end, recall that π_l denotes the indices of columns of A_l chosen by the optimal solution Γ_l, and $\pi_l(A_0)$ is the set of columns of A_0 with these indices. Consider a column vector v in $\pi_l(A_0)$. According to the algorithm, at the end of the l^{th} iteration, the residual vector v_l (which is in $\pi_l(A_l)$) is precisely the projection of v onto the space orthogonal to the vectors chosen by the algorithm, namely $\tau_l(A_0)$. Since this is the case for all possible v's, we have that $\pi_l(A_l)$ is the projection of $\pi_l(A_0)$ onto the space orthogonal to $\tau_l(A_0)$. Hence, according to our last assumption, u_l which is the projection of u onto the space orthogonal to $\tau_l(A_0)$ can be expressed as a linear combination of some other vectors in $\pi_l(A_l)$ since no vector from $\tau_l(A_0)$ can contribute in the expansion of u_l. This contradicts the optimality of Γ_l, i.e. that the number of columns it "selects" from A_l is the smallest among all possible choices.

Lemma 2.6. *For $0 \leq l < t$, $\|\Gamma_l\|_F \leq \frac{3}{2}\eta(A)\|B_l\|_F$.*

PROOF. Consider the column indices $\{i_1, i_2, \ldots, i_l\}$ of the first l vectors chosen by the algorithm. Specifically, let $\tau_l(A_l) = \{A_l^{(i_1)}, A_l^{(i_2)}, \ldots, A_l^{(i_l)}\}$ be the columns in A_l chosen by the algorithm in the order selected. Note that these vectors are orthogonal due to the algorithm. At the end of the l^{th} iteration of the algorithm, for $i \in \pi_l$, we can write

$$A_l^{(i)} = \frac{A_{l-1}^{(i)} - v_l^{(i)}}{\sqrt{1 - \|v_l^{(i)}\|_2^2}}, \tag{6}$$

where $v_l^{(i)}$ is in the span of $A_l^{(i_l)}$ and $\sqrt{1 - \|v_l^{(i)}\|_2^2}$ is the normalization factor implied by the algorithm. Similarly, we can express $A_{l-1}^{(i)}$ in terms of $A_{l-2}^{(i)}$, i.e.

$$A_{l-1}^{(i)} = \frac{A_{l-2}^{(i)} - v_{l-1}^{(i)}}{\sqrt{1 - \|v_{l-1}^{(i)}\|_2^2}},$$

where $v_{l-1}^{(i)}$ is in the span of $A_l^{(i_{l-1})}$. Note that, since the vectors in $\tau_l(A_l)$ are orthogonal, we have $\|v_l^{(i)} + v_{l-1}^{(i)}\|_2^2 = \|v_l^{(i)}\|_2^2 + \|v_{l-1}^{(i)}\|_2^2$. Using this, we can recursively express $A_l^{(i)}$ in (6) as

$$A_l^{(i)} = \frac{{A_0}^{(i)} - v^{(i)}}{\sqrt{1 - \|v^{(i)}\|_2^2}}, \tag{7}$$

for some $v^{(i)} \in span(\tau_l(A_0))$. (Note that $span(\tau_l(A_l) = span(\tau_l(A_0)))$. Thus, noting that $Q_l^{(j)} = \sum_{i \in \pi_l} A_l^{(i)} (\Gamma_l)_{ij}$, and $v^{(i)}$ can be expressed as a linear combination of the column vectors of $\tau_l(A_0)$, we have

$$Q_l^{(j)} = \sum_{i \in \pi_l} (\Gamma_l)_{ij} \frac{{A_0}^{(i)} - v^{(i)}}{\sqrt{1 - \|v^{(i)}\|_2^2}} = \sum_{i \in \pi_l} \frac{(\Gamma_l)_{ij}}{\sqrt{1 - \|v^{(i)}\|_2^2}} {A_0}^{(i)} + \sum_{i \in \tau_l} \delta_i {A_0}^{(i)}, \tag{8}$$

where δ_i's are appropriate coefficients in the expansion of $v^{(i)}$. Now, let S_l be the matrix of the columns from $\pi_l(A_0) \cup \tau_l(A_0)$. Note that, S_l is a column sub-matrix of A_0 which has full rank by Lemma 2.5. Since S_l is a linearly independent set, Q_l has a unique expansion in the basis S_l given by $W_l = S_l^+ Q_l$. Specifically, for $i \in \pi_l$, $(W_l)_{ij} = (\Gamma_l)_{ij}/\sqrt{1 - \|v^{(i)}\|_2^2}$, and for $i \in \tau_l$, $(W_l)_{ij} = \delta_i$. Since $\sqrt{1 - \|v^{(i)}\|_2^2} < 1$, $|(\Gamma_l)_{ij}| \leq |(W_l)_{ij}|$ for $i \in \pi_l$. For $i \in \tau_l$, we have $(\Gamma_l)_{ij} = 0$ and hence trivially $|(\Gamma_l)_{ij}| \leq |(W_l)_{ij}|$. Applying this inequality to the j^{th} column of Γ_l, we obtain $\|\Gamma_l^{(j)}\|_2 \leq \|W_l{}^{(j)}\|_2 \leq \|S_l^+\|_2 \|Q_l^{(j)}\|_2$. The last inequality is due to sub-multiplicativity of the spectral norm. Since $\|A_l\Gamma_l - B_l\|_F = \|Q_l - B_l\|_F \leq \delta/2$, we have $Q_l^{(j)} = B_l^{(j)} + E^{(j)}$, where E is a matrix with $\|E\|_F \leq \delta/2$. We then obtain

$$\begin{aligned}
\|\Gamma_l\|_F^2 &= \sum_{j=1}^{k} \|\Gamma_l^{(j)}\|_2^2 \\
&\leq \|S_l^+\|_2^2 \sum_{j=1}^{k} \|Q_l^{(j)}\|_2^2 \\
&\leq \|S_l^+\|_2^2 \sum_{j=1}^{k} \left(\|B_l^{(j)} + E^{(j)}\|_2^2\right) \\
&\leq \|S_l^+\|_2^2 \sum_{j=1}^{k} \left(\|B_l^{(j)}\|_2 + \|E^{(j)}\|_2\right)^2,
\end{aligned}$$

where the last step is due to the triangle inequality. We continue by expanding the last expression and note that $\|E\|_F \leq \delta/2$, which implies $\sum_{j=1}^{k} \|E^{(j)}\|_2^2 \leq \delta^2/4$:

$$\begin{aligned}
\|\Gamma_l\|_F^2 &\leq \|S_l^+\|_2^2 \sum_{j=1}^{k} \left(\|B_l^{(j)}\|_2 + \|E^{(j)}\|_2\right)^2 \\
&= \|S_l^+\|_2^2 \left(\sum_{j=1}^{k} \|B_l^{(j)}\|_2^2 + \sum_{j=1}^{k} \|E^{(j)}\|_2^2 + 2\sum_{j=1}^{k} \|B_l^{(j)}\|_2 \|E^{(j)}\|_2\right) \\
&\leq \|S_l^+\|_2^2 \left(\|B_l\|_F^2 + \frac{\delta^2}{4} + 2\sum_{j=1}^{k} \|B_l^{(j)}\|_2 \|E^{(j)}\|_2\right) \\
&\leq \|S_l^+\|_2^2 \left(\frac{5}{4}\|B_l\|_F^2 + 2\sum_{j=1}^{k} \|B_l^{(j)}\|_2 \|E^{(j)}\|_2\right) \quad (\|B_l\|_F > \delta).
\end{aligned}$$

Applying the Cauchy-Schwartz inequality to the second term in the parentheses, we obtain

$$
\begin{aligned}
\|\Gamma_l\|_F^2 &\leq \|S_l^+\|_2^2 \left(\frac{5}{4}\|B_l\|_F^2 + 2 \left(\sum_{j=1}^{k} \|B_l^{(j)}\|_2^2 \right)^{1/2} \left(\sum_{j=1}^{k} \|E^{(j)}\|_2^2 \right)^{1/2} \right) \\
&= \|S_l^+\|_2^2 \left(\frac{5}{4}\|B_l\|_F^2 + 2\|B_l\|_F \|E\|_F \right) \\
&\leq \|S_l^+\|_2^2 \left(\frac{5}{4}\|B_l\|_F^2 + \delta\|B_l\|_F \right) \quad (\|E\|_F \leq \delta/2) \\
&\leq \|S_l^+\|_2^2 \left(\frac{5}{4}\|B_l\|_F^2 + \|B_l\|_F^2 \right) \quad (\|B_l\|_F > \delta) \\
&= \frac{9}{4}\|S_l^+\|_2^2 \|B_l\|_F^2.
\end{aligned}
$$

Hence, we have $\|\Gamma_l\|_F \leq \frac{3}{2}\|S_l^+\|_2 \|B_l\|_F$. Now, note that the rank of S_l is less than or equal to r, the rank of A_0. S_l can be obtained by deleting columns of a full-rank sub-matrix C_0 of A_0, which has exactly r columns. $\|S_l^+\|_2$, which is the inverse of the least singular value of S_l is smaller than that of such a matrix C_0 (see [20], Thm 8.1.7). Then, by the definition of $\eta(A)$, we clearly have $\|S_l^+\|_2 \leq \|{C_0}^+\|_2 \leq \eta(A)$ and the lemma follows.

We can now prove the main theorem. Recall that $Opt(\delta/2)$ is the number of non-zero rows in an optimal solution for the generalized sparse approximation problem with error $\delta/2$.

Proof of Theorem 2.2:. We first note that the number of non-zero rows in an optimal solution is non-increasing as the algorithm proceeds, that is $\nu(\Gamma_l) \geq \nu(\Gamma_{l+1})$ for $l > 0$. Let Γ be an optimal solution to $\|AX - B\|_F \leq \delta/2$. Then, since $A_0 = AD$ for some diagonal scaling matrix D, we have that the matrix $\Gamma_0 = D^{-1}\Gamma$ is an optimal solution to $\|A_0 X - B\|_F \leq \delta/2$. Clearly, the number of non-zero rows in Γ_0 is the same as that of Γ. Thus, $Opt(\delta/2) = \nu(\Gamma_0)$ and we get

$$
\lambda = \max_{0 \leq l < t} \frac{\nu(\Gamma_0)\|\Gamma_l\|_F^2}{\|B_l\|_F^2} \leq 18 Opt(\delta/2)\eta^2(A),
$$

where the inequality is due to Lemma 2.6. Combining this with Lemma 2.4, we have that the number of iterations of the algorithm is bounded by

$$
t \leq \left\lceil 18 Opt(\delta/2)\eta^2(A) \ln \left(\frac{\|B\|_F}{\delta} \right) \right\rceil.
$$

We finally note that due to the algorithm, each column of AX is a linear combination of the selected columns C. Thus, there exists a $c \times k$ coefficient matrix Ω such that $AX = C\Omega$. Since

$$
\min_{Y \in \mathbb{R}^{c \times k}} \|CY - B\|_F = \|CC^+B - B\|_F,
$$

we have that the selected columns satisfy $\|CC^+B - B\|_F \leq \|C\Omega - B\|_F = \|AX - B\|_F \leq \delta$. This completes the proof. ■

. In the next section, we show that if B is chosen to span the subspace defined by the first k singular vectors of the matrix A, $Opt(\delta/2)$ has some desired properties. We also show how Theorem 2.2 can be used to bound $\|A - CC^+A\|_F$, yielding the main result.

Algorithm 2 The SVD Approximation Algorithm

1: **procedure** ApproximateSVD(A, k)
2: compute U_k and Σ_k of A
3: return Greedy(A, $U_k\Sigma_k$, $\epsilon\|A - A_k\|$)
4: **end procedure**

2.4. Greedy Approximation of SVD

The algorithm for approximating the truncated SVD of A is based on the greedy approach that we have introduced for the generalized sparse approximation problem.

The algorithm first computes U_k, the top k left singular vectors of A and Σ_k the first k singular values of A, which can be performed by standard Lanczos-type methods. The columns of A are then selected in a greedy fashion so as to "fit" them to the subspace spanned by the columns of $U_k\Sigma_k$. Intuitively, we select columns of A which are close to the columns of $U_k\Sigma_k$ and the analysis shows that the sub-matrix C of A we obtain is provably close to the "best" rank-k approximation to A. The error parameter δ which is given as an input to Greedy is $\epsilon\|A - A_k\|_F$. The choice of this parameter determines the additive error in the result and is crucial for the analysis. Recall from previous section that the number of columns chosen by the greedy algorithm for generalized sparse approximation problem depends on $Opt(\delta/2)$ which is the number of columns in an optimal solution at error $\delta/2$. A major part of this section is devoted to show the existence of an optimal solution satisfying certain criteria, given the parameters for the algorithm in this section. The following lemma will establish this.

Lemma 2.7. *Given a matrix $A \in \mathbb{R}^{m\times n}$, there exists a matrix $\Gamma \in \mathbb{R}^{n\times k}$ which satisfies*

1. $\nu(\Gamma) = O(k \log k/\epsilon^2)$ *(the number of non-zero rows)*
2. $\|U_k\Sigma_k - A\Gamma\|_F \leq \epsilon\|A - A_k\|_F$.

Proof. We will make use of a result which is originally provided in [17]. The authors give a randomized algorithm which constructs, with non-zero probability, a set of columns with a particular approximation property which immediately translates to an existence result by the probabilistic method. For a set of columns $C \subset A$, denote the sampling matrix which selects the columns by $S \in \mathbb{R}^{n\times c}$ so that $C = AS$. Consider also a diagonal matrix $D \in \mathbb{R}^{c\times c}$ which scales the selected columns C. Let V_k be the matrix of the first k right singular vectors of A. Let V_{r-k} be the matrix containing the last $r-k$ right singular vectors of A, and let Σ_k and Σ_{r-k} be the diagonal matrices containing the first k and the last $r-k$ singular values of A. The full proof of this result appears in [18] (Sections 6.3.1 and 6.3.5) by the same authors, in a slightly different context.

Lemma 2.8 ([18]). *There exists a randomized algorithm which selects a set of $c = O(k \log k/\epsilon^2)$ columns from A, with the corresponding sampling matrix $S \in \mathbb{R}^{n\times c}$ satisfying $C = AS$, and a diagonal scaling matrix $D \in \mathbb{R}^{c\times c}$ such that $rank(V_k^T SD) = rank(V_k)$ and*

$$\|\Sigma_{r-k}V_{r-k}^T SD(V_k^T SD)^+\|_F \leq \epsilon\|A - A_k\|_F,$$

where Σ_{r-k} is the diagonal matrix containing the smallest $r-k$ singular values of A, and V_{r-k} is the matrix containing the last $r-k$ right singular vectors of A.

We will show that the matrix $\Gamma = S(AS)^+U_k\Sigma_k \in \mathbb{R}^{n\times k}$ satisfies the claims given in Lemma 2.7, where S is the matrix mentioned in Lemma 2.8. First, note that $S \in \mathbb{R}^{n\times c}$ is a sampling matrix which has a single entry of 1 in each column. Then, the resulting matrix Γ has at most c non-zero rows, which means $\nu(\Gamma) \leq c = O(k \log k/\epsilon^2)$. This establishes the first claim.

We establish the second claim as follows: Let $C = AS$ be the column sub-matrix whose existence is guaranteed by Lemma 2.8. We have

$$\begin{aligned}\epsilon^2\|A-A_k\|_F^2 &\geq \|\Sigma_{r-k}V_{r-k}^T SD(V_k^T SD)^+\|_F^2 \\ &= \|\Sigma_k - \Sigma_k V_k^T SD(V_k^T SD)^+\|_F^2 + \|\Sigma_{r-k}V_{r-k}^T SD(V_k^T SD)^+\|_F^2,\end{aligned}$$

where the first term in the last expression is just 0 as $V_k^T SD(V_k^T SD)^+ = I_k$. Combining the last two terms into one expression, we have

$$\begin{aligned}\epsilon^2\|A-A_k\|_F^2 &\geq \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - \begin{pmatrix} \Sigma_k V_k^T \\ \Sigma_{r-k}V_{r-k}^T \end{pmatrix} SD(V_k^T SD)^+ \right\|_F^2 \\ &= \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_{r-k} \end{pmatrix} \begin{pmatrix} V_k^T \\ V_{r-k}^T \end{pmatrix} SD(V_k^T SD)^+ \right\|_F^2 \\ &= \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - (\Sigma V^T SD)(\Sigma_k V_k^T SD)^+ \Sigma_k \right\|_F^2 \\ &= \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - (\Sigma V^T SD)Y \right\|_F^2,\end{aligned}$$

where $Y = (\Sigma_k V_k^T SD)^+\Sigma_k$. Let A, B be arbitrary matrices. Then, $\min_X \|A-BX\|_F{}^2 = \|A - BB^+A\|_F{}^2$ (see [20]). We continue as follows:

$$\begin{aligned}\left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - (\Sigma V^T S)Y \right\|_F^2 &\geq \min_{X\in\mathbb{R}^{c\times k}} \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - (\Sigma V^T SD)X \right\|_F^2 \\ &= \left\| \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} - (\Sigma V^T SD)(\Sigma V^T SD)^+ \begin{pmatrix} \Sigma_k \\ 0 \end{pmatrix} \right\|_F^2 \\ &= \left\| \begin{pmatrix} I_k \\ 0 \end{pmatrix} \Sigma_k - (\Sigma V^T SD)(\Sigma V^T SD)^+ \begin{pmatrix} I_k \\ 0 \end{pmatrix} \Sigma_k \right\|_F^2 \\ &= \left\| U \begin{pmatrix} I_k \\ 0 \end{pmatrix} \Sigma_k - (U\Sigma V^T SD)(\Sigma V^T SD)^+ U^T U_k\Sigma_k \right\|_F^2 \\ &= \left\| U_k\Sigma_k - (U\Sigma V^T SD)(U\Sigma V^T SD)^+ U_k\Sigma_k \right\|_F^2 \\ &= \left\| U_k\Sigma_k - ASD(ASD)^+ U_k\Sigma_k \right\|_F^2 \\ &= \left\| U_k\Sigma_k - AS(AS)^+ U_k\Sigma_k \right\|_F^2 \\ &= \|U_k\Sigma_k - A\Gamma\|_F^2\end{aligned}$$

where we have used $U\Sigma V^T = A$ and $(ASD)^+ = D^+(AS)^+$. This establishes the second claim and the lemma.

We now, give the proof of Theorem 1.1.

Proof of Theorem 1.1:. First, note that $A\Gamma = AS(AS)^+U_k\Sigma_k = CC^+U_k\Sigma_k$. By Theorem 2.2, we have

$$U_k\Sigma_k = CC^+U_k\Sigma_k + E$$

for some matrix E satisfying $\|E\|_F \leq \epsilon\|A-A_k\|_F$. Multiplying both sides by V_k^T, we get

$$U_k\Sigma_k V_k^T = CC^+U_k\Sigma_k V_k^T + EV_k^T,$$

which is clearly

$$A_k = CC^+A_k + EV_k^T.$$

Rearranging and adding A to the both sides of the equation, we obtain $A - CC^+A_k = A - A_k + EV_k^T$. Taking norms of both sides, and noting that C^+A is the minimizer of $\|A - CX\|_F$ over X, we obtain

$$\begin{aligned}
\|A - CC^+A\|_F &\leq \|A - CC^+A_k\|_F \\
&= \|A - A_k + EV_k^T\|_F, \\
&\leq \|A - A_k\|_F + \|E\|_F\|V_k^T\|_2 \\
&\leq \|A - A_k\|_F + \epsilon\|A - A_k\|_F \\
&= (1+\epsilon)\|A - A_k\|_F.
\end{aligned}$$

Third line follows due to the triangle inequality and sub-multiplicativity of the Frobenius norm. The fourth line is due to the fact that $\|E\|_F \leq \epsilon\|A - A_k\|_F$ and $\|V_k^T\|_2 = 1$. Combining Theorem 2.2 and Lemma 2.7 gives the desired result. ■

3. Numerical Results

In this section, we present numerical experiments using the algorithm ApproximateSVD, comparing it to a few other significant algorithms providing bounds for the performance metric we have analyzed, in the case of *exactly* k columns are chosen. We report the error ratios $\|A - CC^+A\|_2/\|A - A_k\|_2$, $\|A - CC^+A\|_F/\|A - A_k\|_F$ for various matrices and different values of k along with the running times on one of the matrices. We make use of 3 different types of $n \times n$ matrices for $n = 400$ and $n = 1000$, a total of 6 different matrices. Running times are only reported for $n = 1000$. Below are the matrices that are used in our experiments:

- *Log:* a random matrix A with singular values equally spaced between 1 and $10^{-\log n}$. More specifically, $A = U\Sigma V^T$, where Σ is the diagonal matrix with entries of the logarithmic distribution, and U and V are random orthogonal matrices.
- *Scaled Random:* a random matrix A created by assigning each entry a number between -1 and 1 from uniform distribution, and then scaling the i^{th} row of that matrix by $(20\epsilon)^{i/n}$ where ϵ is the machine precision. In our case, $\epsilon = 2.22 \cdot 10^{-16}$. This matrix was utilized in [21].
- *Kahan:* a matrix

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \zeta & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \zeta^{n-1} \end{pmatrix} \cdot \begin{pmatrix} 1 & -\phi & \dots & -\phi \\ 0 & 1 & \dots & -\phi \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

  with $\zeta, \phi > 0$, and $\phi^2 + \zeta^2 = 1$. Kahan matrices are first mentioned in [24]. These matrices have low numerical rank and they provably yield bad results for the commonly used pivoted QR algorithm [6]. Along the same lines in [21], we set $\phi = 0.285$ for our experiments.

The variation in the results were negligible with respect to the random choices in the construction of the first two classes of matrices, hence we report results of one randomly generated matrix in each class. We have implemented the following 3 algorithms in C++ along with ApproximateSVD and performed experiments on an Intel Core 2 Duo T4200 at 2.16 Ghz, 4 GB machine:

- Pivoted-QR: The algorithm of Golub and Businger [6]. [21] shows that it chooses a sub-matrix C satisfying $\|A - CC^+A\|_2 \leq 2^k\sqrt{n-k}\|A - A_k\|_2$. We report the running times of choosing exactly k columns, not of a complete decomposition.
- Low-RRQR: The algorithm introduced by Chan and Hansen in [8], which provides $\|A - CC^+A\|_2 \leq 2^{k+1}\sqrt{(k+1)n}\|A - A_k\|_2$. This algorithm also involves computation of a singular vector at each iteration, and requires a full QR decomposition as a preliminary step. We report the running times including this preliminary step for which we used pivoted-QR, followed by k iterations of the algorithm.
- Hybrid: The algorithm by Boutsidis et al. [4], which combines random sampling techniques and deterministic approaches. It guarantees $\|A - CC^+A\|_F = \Theta\left(k\log^{1/2}k\right)\|A - A_k\|_F$. We report the error ratios of the algorithm run using the specific sampling distribution tailored to the norm. This algorithm first chooses (on average) c columns randomly of the matrix A. These columns are related to the right singular vectors of A. It then makes use of a deterministic procedure to cut down the number of columns to k. The number c is theoretically of order $O(k\log k)$, but in practice the authors suggest to use a value between $2k$ and $10k$ [3]. We have chosen $c = 6k$ and used Pivoted-QR algorithm as the deterministic step. We run the algorithm 40 times to boost the success probability and get the best error ratio, as suggested in [4].

For the computation of a partial SVD (top k singular values and singular vectors), which are required for Hybrid and ApproximateSVD, we have used a C version of the SVDPACK library [2], which utilizes Lanczos-type methods.

k	$\|A - CC^+A\|_2/\|A - A_k\|_2$				$\|A - CC^+A\|_F/\|A - A_k\|_F$			
	P-QR	L-RRQR	Hybrid	AprxSVD	P-QR	L-RRQR	Hybrid	AprxSVD
1	1.035	1.035	1.035	1.035	1.035	1.035	1.035	1.035
2	1.042	1.058	1.007	**1.003**	1.030	1.029	1.039	**1.020**
3	1.069	1.093	1.019	**1.005**	1.042	1.045	1.049	**1.034**
4	1.105	1.101	1.060	**1.045**	1.055	1.062	1.068	**1.042**
5	1.137	1.116	1.117	**1.035**	1.072	1.074	1.072	**1.051**
6	1.130	1.144	1.079	**1.042**	1.089	1.092	1.089	**1.064**
7	1.153	1.160	1.114	**1.093**	1.098	1.104	1.109	**1.075**
8	1.192	1.195	1.125	**1.094**	1.111	1.120	1.114	**1.083**
9	1.233	1.213	1.189	**1.110**	1.128	1.136	1.145	**1.097**
10	1.275	1.220	1.202	**1.130**	1.145	1.147	1.151	**1.107**
20	1.500	1.404	1.409	**1.256**	1.274	1.266	1.296	**1.222**
30	1.508	1.678	1.533	**1.406**	1.372	1.395	1.404	**1.327**
40	1.813	1.678	1.668	**1.536**	1.483	1.509	1.522	**1.432**
50	1.935	1.896	1.851	**1.612**	1.596	1.621	1.582	**1.539**

Table 1: Error ratios of Low-Rank Approximation Algorithms for *Log* 400×400. In bold for each k is the best method.

We show the error ratios of the algorithms on matrices of size 400×400 in Tables 1, 2 and 3. The behavior of the algorithms on the matrices of size 1000×1000 are quite similar, and for convenience we give the results for these matrices in Figures 1(a) to 3(b). In Frobenius norm, ApproximateSVD consistently outperforms the other algorithms tested, especially when k is small. This is due to the rationale of the algorithm, that it is trying to choose column vectors whose span is as close as possible to A_k. It is intuitively reasonable to expect that the distance between any column vector to the subspace chosen by ApproximateSVD should be close to the distance between that vector to the optimal subspace, which is quantitatively expressed via the ratio of the Frobenius norm errors. The only exception is the matrix *Scaled Random* for large values of k. Note that, even the Pivoted-QR algorithm works very well for this type of matrix. ApproximateSVD also presents very good results in spectral norm except small values of k on *Kahan*. Low-RRQR gives the best results

k	$\|A - CC^+A\|_2/\|A - A_k\|_2$				$\|A - CC^+A\|_F/\|A - A_k\|_F$			
	P-QR	L-RRQR	Hybrid	AprxSVD	P-QR	L-RRQR	Hybrid	AprxSVD
1	1.015	1.015	1.015	1.015	1.080	1.080	1.080	1.080
2	1.119	1.042	1.067	**1.016**	1.067	1.048	1.073	**1.040**
3	1.118	1.060	1.115	**1.024**	1.085	1.080	1.097	**1.069**
4	1.231	1.185	1.108	**1.042**	1.119	1.121	1.129	**1.095**
5	1.101	1.164	1.120	**1.078**	1.135	1.136	1.160	**1.111**
6	1.183	1.213	1.192	**1.079**	1.154	1.158	1.218	**1.142**
7	1.225	1.173	1.213	**1.132**	1.191	1.167	1.223	**1.168**
8	1.276	1.234	1.161	**1.090**	1.219	1.192	1.215	**1.190**
9	1.339	1.257	1.305	**1.158**	1.249	1.210	1.258	**1.231**
10	1.317	1.328	**1.307**	**1.307**	1.265	**1.233**	1.282	1.241
20	1.577	1.597	1.676	**1.417**	1.450	**1.435**	1.451	1.456
30	1.673	2.137	**1.527**	1.723	**1.621**	1.705	1.695	1.708
40	2.067	2.171	1.997	**1.912**	**1.753**	1.833	1.845	1.905
50	2.222	1.939	**1.936**	2.244	1.936	**1.935**	1.929	2.085

Table 2: Error ratios of Low-Rank Approximation Algorithms for *Scaled Random* 400×400. In bold for each k is the best method.

k	$\|A - CC^+A\|_2/\|A - A_k\|_2$				$\|A - CC^+A\|_F/\|A - A_k\|_F$			
	P-QR	L-RRQR	Hybrid	AprxSVD	P-QR	L-RRQR	Hybrid	AprxSVD
1	10.343	10.343	10.343	10.343	4.383	4.383	4.383	4.383
2	8.539	1.342	1.314	**1.308**	2.759	1.064	1.103	**1.063**
3	9.401	**1.308**	1.387	1.381	2.879	1.084	1.069	**1.068**
4	9.806	**1.320**	1.388	1.381	2.989	1.083	**1.068**	**1.068**
5	10.218	**1.343**	1.394	1.381	3.102	1.083	**1.068**	**1.068**
6	10.638	**1.264**	1.383	1.381	3.216	1.103	1.069	**1.068**
7	11.063	**1.273**	1.594	1.381	3.332	1.101	1.123	**1.068**
8	11.496	**1.296**	1.619	1.381	3.450	1.099	1.121	**1.068**
9	11.988	**1.248**	1.622	1.381	3.579	1.122	1.141	**1.068**
10	12.449	**1.250**	1.642	1.381	3.705	1.119	1.113	**1.068**
20	17.340	**1.321**	1.612	1.381	5.055	1.136	1.114	**1.068**
30	18.477	1.406	1.612	**1.382**	5.373	1.183	1.117	**1.068**
40	18.215	1.589	1.612	**1.382**	5.300	1.181	1.133	**1.068**
50	15.633	1.437	1.612	**1.382**	4.580	1.141	1.114	**1.068**

Table 3: Error ratios of Low-Rank Approximation Algorithms for *Kahan* 400×400. In bold for each k is the best method.

Table 4: Running times of Low-Rank Approximation Algorithms for *Scaled Random* 1000×1000

k	P-QR	L-RRQR	Hybrid	AprxSVD
5	0.047	2.359	2.235	0.375
10	0.078	2.625	3.235	0.501
20	0.140	3.047	4.875	0.891
30	0.203	3.468	7.001	1.079
40	0.265	4.234	8.985	1.468
50	0.359	4.313	12.359	1.922
75	0.453	5.109	19.078	2.798
100	0.578	6.063	25.546	3.687

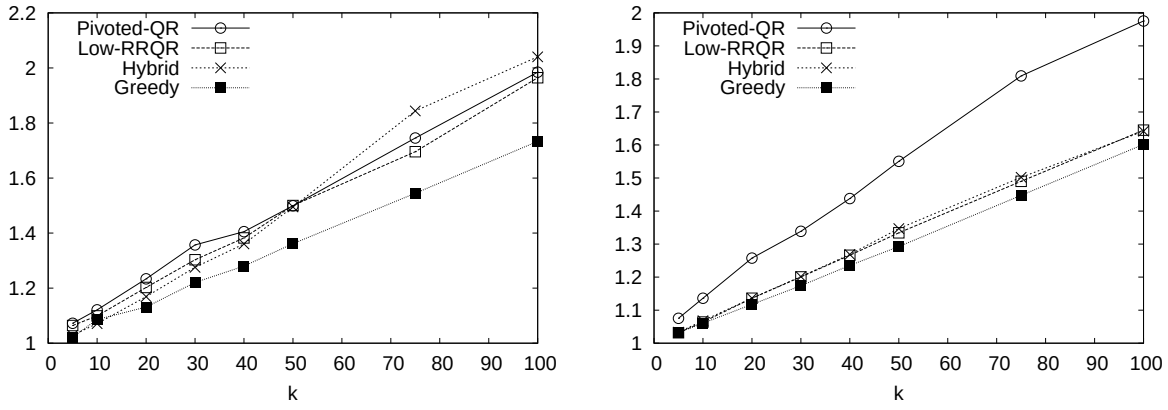


Figure 1: Error ratios of Low-Rank Approximation Algorithms for *Log* 1000×1000 in (a) Spectral Norm and (b) Frobenius Norm

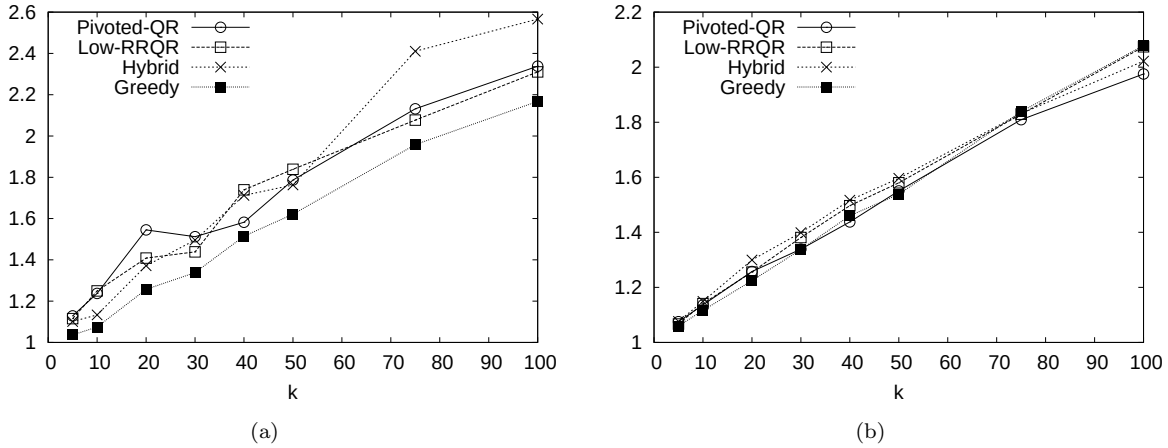
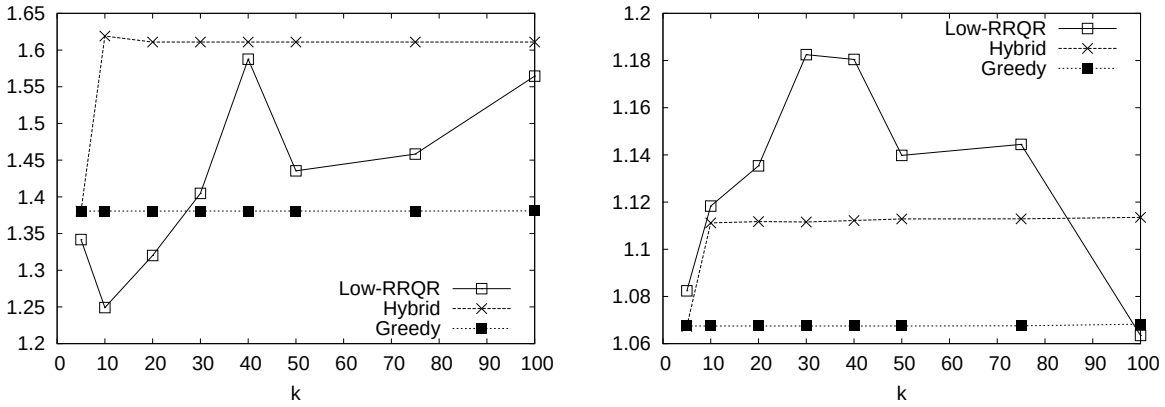


Figure 2: Error ratios of Low-Rank Approximation Algorithms for *Scaled Random* 1000×1000 in (a) Spectral Norm and (b) Frobenius Norm



(a) (b)

Figure 3: Error ratios of Low-Rank Approximation Algorithms for *Kahan* 1000×1000 in (a) Spectral Norm and (b) Frobenius Norm

for small k on this matrix. We would like to note that, Low-RRQR is an algorithm that greedily selects a column which is close to the singular vector associated with the largest singular value of the "uncovered" space at each step, whereas our algorithm computes the k dimensional space to be approximated at the beginning. Hence, Low-RRQR gives better results in spectral norm for low-rank matrices with rapidly decreasing singular values, like *Kahan*. We would also like to note that the error ratios of the algorithms which make use of the k-dimensional subspace spanned by A_k, namely Hybrid and ApproximateSVD stay constant for large values of k, as the column subspace is pretty much determined by the leading first few singular vectors of the matrix. Pivoted-QR performs poorly on *Kahan* as expected.

Table 4 gives the running times of the algorithms on the 1000×1000 *Scaled Random* matrix. Pivoted-QR is the fastest algorithm, and ApproximateSVD is faster than Low-RRQR. If the time-consuming preliminary decomposition in Low-RRQR is disregarded, these two algorithms have quite similar behavior in terms of running time. Hybrid is the slowest of all due to the large number of repetitions.

4. Discussion

This work is a modest attempt to bridge the gap between sparse approximation and low-rank matrix approximation. We have presented an algorithm that approximates the space spanned by the top k left singular vectors of a matrix A by solving a generalization of the sparse approximation problem. The bulk of the analysis is based on the generalized case of approximating an arbitrary space. Hence, the term $\eta(A)$ that appears in the analysis is in fact a general bound. We believe that a more refined analysis focusing on the specific problem of approximating U_k will yield much better theoretical guarantees. As an example, a direct existence result in the special case eliminating $\eta(A)$ is likely in Lemma 2.6. In practice, the algorithm gives superior results than the theoretical guarantees suggest. A smoothed analysis [33] might give further insight into the performance of the algorithm.

Acknowledgments: We would like to thank the anonymous referees for their helpful comments.

References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *STOC '01: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 611–618, 2001.

[2] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. SVDPACKC (version 1.0) user's guide. Technical report, University of Tennessee, 1993.

[3] C. Boutsidis, M. W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–69, 2008.

[4] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *SODA '09: Proceedings of the 19th Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009.

[5] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[6] P. A. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numerische Mathematik*, 7:269–276, 1965.

[7] T. F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88/89:67–82, 1987.

[8] T. F. Chan and P.C. Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1:33–44, 1994.

[9] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15:592–622, 1994.

[10] J. Chen and X. Huo. Theoretical results on sparse representations of multiple-measurement vectors. *IEEE Transactions on Signal Processing*, 54:4634–4643, 2006.

[11] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13:57–98, 1997.

[12] F. R. de Hoog and R. M. M. Mattheijb. Subset selection for matrices. *Linear Algebra and its Applications*, 422:349–359, 2007.

[13] A. Deshpande and K. Varadarajan. Sampling-based dimension reduction for subspace approximation. In *STOC '07: Proceedings of the 39th Annual ACM Aymposium on Theory of computing*, pages 641–650, 2007.

[14] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *RANDOM'06: 10th International Workshop on Randomization and Computation*, pages 292–303, 2006.

[15] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *SODA '99: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 291–299, 1999.

[16] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1), 2006.

[17] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *RANDOM'06: 10th International Workshop on Randomization and Computation*, pages 316–326, 2006.

[18] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[19] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6), 2004.

[20] G. H. Golub and C.H. Van Loan. *Matrix Computations*. Johns Hopkins U. Press, 1996.

[21] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17(4), 1996.

[22] V. Guruswami and A. K. Sinop. Optimal column-based low-rank matrix reconstruction. Manuscript, 2011.

[23] Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58:213–232, 1992.

[24] W. Kahan. Numerical linear algebra. *Canadian Mathematical Bulletin*, 9:757–801, 1966.

[25] F. G. Kuruvilla, P. J. Park, and S. L. Schreiber. Vector algebra in the analysis of genome-wide expression data. *Genome Biology*, 3, 2002.

[26] A. Lutoborski and V. N. Temlyakov. Vector greedy algorithms. *Journal of Complexity*, 19:458–473, 2003.

[27] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[28] C. T. Pan and P. T. P. Tang. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics*, 39:740–756, 1999.

[29] M. Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[30] M. Rudelson and R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the Association for Computing Machinery*, 54(4), 2007.

[31] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 143–152, 2006.

[32] N. D. Shyamalkumar and K. Varadarajan. Efficient subspace approximation algorithms. In *SODA '07: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 532–540, 2007.

[33] D. A. Spielman and S. H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[34] J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[35] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation part I: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.