Relative Errors for Deterministic Low-Rank Matrix Approximations

Mina Ghashami  
School of Computing  
University of Utah  
ghashami@cs.utah.edu

Jeff M. Phillips  
School of Computing  
University of Utah  
jeffp@cs.utah.edu

Abstract

We consider processing an \( n \times d \) matrix \( A \) in a stream with row-wise updates according to a recent algorithm called Frequent Directions (Liberty, KDD 2013). This algorithm maintains an \( \ell \times d \) matrix \( Q \) deterministically, processing each row in \( O(d\ell^2) \) time; the processing time can be decreased to \( O(d\ell) \) with a slight modification in the algorithm and a constant increase in space. Then for any unit vector \( x \), the matrix \( Q \) satisfies

\[
0 \leq \|Ax\|^2 - \|(Qx\|)^2 \leq \|A\|^2/\ell.
\]

We show that if one sets \( \ell = \lceil k + k/\varepsilon \rceil \) and returns \( Q_k \), a \( k \times d \) matrix that is simply the top \( k \) rows of \( Q \), then we achieve the following properties:

\[
\|A - A_k\|^2 \leq \|A\|^2 - \|(Q_k\|)^2 \leq (1 + \varepsilon)\|A - A_k\|^2
\]

and where \( \pi_{Q_k}(A) \) is the projection of \( A \) onto the rowspace of \( Q_k \) then

\[
\|A - \pi_{Q_k}(A)\|^2 \leq (1 + \varepsilon)\|A - A_k\|^2.
\]

We also show that Frequent Directions cannot be adapted to a sparse version in an obvious way that retains \( \ell \) original rows of the matrix, as opposed to a linear combination or sketch of the rows.

1 Introduction

The data streaming paradigm [27] considers computation on a large data set \( A \) where one data item arrives at a time, is processed, and then is not read again. It enforces that only a small amount of memory is available at any given time. This small space constraint is critical when the full data set cannot fit in memory or disk. Typically, the amount of space required is traded off with the accuracy of the computation on \( A \). Usually the computation results in some summary \( S(A) \) of \( A \), and this trade-off determines how accurate one can be with the available space resources. Although computational runtime is important, in this paper we mainly focus on space constraints and the types of approximation guarantees that can be made.

In truly large datasets, one processor (and memory) is often incapable of handling all of the dataset \( A \) in a feasible amount of time. Even reading a terabyte of data on a single processor can take many hours. Thus this computation is often spread among some set of processors, and then the summary of \( A \) is combined after (or sometimes during [8]) its processing on each processor. Again often each item is read once, whether it comes from a single large source or is being generated on the fly. The key computational problem shifts from updating a summary \( S(A) \) when witnessing a single new data item (the streaming model), to taking two summaries \( S(A_1) \) and \( S(A_2) \) and constructing a single new summary \( S(A_1 \cup A_2) \). In this new paradigm the goal is to have the same space-approximation trade-offs in \( S(A_1 \cup A_2) \) as possible for a streaming algorithm. When such a process is possible, the summary is known as a mergeable summary [2]. Linear sketches are trivially mergeable, so this allows many streaming algorithms to directly translate to this newer paradigm. Again, space is a critical resource since it directly corresponds with the amount of data needed to transmit across the network, and emerging cost bottleneck in big data systems.

In this paper we focus on deterministic mergeable summaries for low-rank matrix approximation, based on recent work by Liberty [23], that is already known to be mergeable [23]. Thus our focus is a more careful analysis of the space-error trade off for the algorithm, and we describe them under the streaming setting for simplicity; all bounds directly carry over into mergeable summary results. In particular we re-analyze the Frequent Directions algorithm of Liberty to show it provides relative error bounds for matrix sketching, and conjecture it achieves the optimal space, up to log factors, for any row-update based summary. This supports the strong empirical results of Liberty [23]. His analysis only provided additive error bounds which are hard to compare to more conventional ways of measuring accuracy of matrix approximation algorithms.

1.1 Problem Statement and Related Work  
In this problem \( A \) is an \( n \times d \) matrix and the stream processes each row \( a_i \) (of length \( d \)) at a time. Typically the matrix is assumed to be tall so \( n \gg d \), and sometimes the matrix will be assumed to be sparse so the number of non-zero entries \( \text{nnz}(A) \) of \( A \) will be small, \( \text{nnz}(A) \ll nd \) (e.g. \( \text{nnz}(A) = O((n + d) \log(nd)) \)).

The best rank-\( k \) approximation to \( A \) (under Frobenius or 2 norm) is denoted as \( A_k \) and can be computed in \( O(nd^2) \) time on a tall matrix using the singular value decomposition. The \( \text{svd}(A) \) produces three matrices \( U \), \( S \), and \( V \) where \( U \)
and $V$ are orthonormal, of size $n \times n$ and $d \times d$, respectively, and $S$ is $n \times d$ but only has non-zero elements on its diagonal \{$\sigma_1, \ldots, \sigma_d$\}. Let $U_k, S_k,$ and $V_k$ be the first $k$ columns of each matrix, then $A = USV^T$ and $A_k = U_kS_kV_k^T$. Note that although $A_k$ requires $O(nd)$ space, the set of matrices \{$U_k, S_k, V_k$\} require only a total of $O((n + d)k)$ space (or $O(nk)$ if the matrix is tall). Moreover, even the set \{$U, S, V$\} really only takes $O(nd + d^2)$ space since we can drop the last $n - d$ columns of $U$, and the last $n - d$ rows of $S$ without changing the result. In the streaming version, the goal is to compute something that replicates the effect of $A_k$ using less space and only seeing each row once.

We next describe the most relevant related works under various categories, and then in Section 1.2 we make an effort to catalog the upper bounds of directly comparable algorithms.

**Construction bounds.** The strongest version, (providing construction bounds) for some parameter $\varepsilon \in (0, 1)$, is some representation of a rank $k$ matrix $\hat{A}$ such that $\|A - \hat{A}\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$ for $\varepsilon = (\frac{1}{2}, F)$. Unless $A$ is sparse, then storing $\hat{A}$ explicitly may require $O(nd)$ space, so that is why various representations of $\hat{A}$ are used in its place. This can include decompositions similar to the SVD, e.g. a CUR decomposition [11, 15, 24] where $\hat{A} = CUR$ and where $U$ is small and dense, and $C$ and $R$ are sparse and skinny, or others [7] where the middle matrix is still diagonal. The sparsity is often preserved by constructing the wrapper matrices (e.g. $C$ and $R$) from the original columns or rows of $A$. There is an obvious $\Omega(n + d)$ space lower bound for any construction result in order to preserve the column and the row space.

**Projection bounds.** Alternatively, a weaker version (providing projection bounds) just finds a rank $k$ subspace $B_k$ where the projection of $A$ onto this subspace $\pi_{B_k}(A)$ represents $\hat{A}$. This bound is weaker since this cannot actually represent $\hat{A}$ without making another pass over $A$ to do the projection. An even weaker version finds a rank $r > k$ subspace $B$, where $\hat{A}$ is represented by the best rank $k$ approximation of $\pi_B(A)$; note that $\pi_B(A)$ is then also rank $r$, not $k$. However, when $B$ or $B_k$ is composed of a set of columns (and perhaps $B_k$ is only $k$ rows) then the total size is only $O(d\ell)$ (alloting constant space for each entry); so it does not depend on $n$. This is a significant advantage in tall matrices where $n \gg d$. Sometimes this subspace approximation is sufficient for downstream analysis, since the rowspace is still (approximately) preserved. For instance, in PCA the goal is to compute the most important directions in the row space.

**Streaming algorithms.** Many of these algorithms are not streaming algorithms. To the best of our understanding, the best streaming algorithm [6] is due to Clarkson and Woodruff. All bounds assume each matrix entry requires $O(\log nd)$ bits. It is randomized and it constructs a decomposition of a rank $k$ matrix $\hat{A}$ that satisfies $\|A - \hat{A}\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$, with probability at least $1 - \delta$. This provides a relative error construction bound of size $O((k/\varepsilon)(n + d)\log(nd))$ bits. They also show an $\Omega((k/\varepsilon)(n + d))$ bits lower bound.

Although not explicitly described in their paper, one can directly use their techniques and analysis to achieve a weak form of a projection bound. One maintains a matrix $B = AS$ with $m = O((k/\varepsilon)(\log(1/\delta)))$ columns where $S$ is a $d \times m$ matrix where each entry is chosen from \{-1,+1\} at random. Then setting $\hat{A} = \pi_B(A)$, achieves a relative $(1 + \varepsilon)$-error projection bound, however $B$ is rank $O((k/\varepsilon)(\log(1/\delta)))$ and hence that is the only bound on $\hat{A}$ as well. The construction lower bound suggests that there may be an $\Omega(dk/\varepsilon)$ bits lower bound for projection, but is not proven. They also study this problem in the turnstile model where each element of the matrix can be updated at each step (including subtractions). In this model they require $O((k/\varepsilon^2)(n + d)\log(nd))$ bits, and show an $\Omega((k/\varepsilon)(n + d)\log(nd))$ bits lower bound.

Another more general “coreset” result is provided by Feldman et al. [16]. In the streaming setting it requires $O((k/\varepsilon)\log n)$ space and can be shown to provide a rank $O(k/\varepsilon)$ matrix $B$ that satisfies a relative error bound of the form $\|A - \pi_B(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$.

**Column sampling.** Another line of work [1,4,10–15,24,30] considers selecting a set of rows from $A$ directly (not maintaining rows that for instance may be linear combinations of rows of $A$). This maintains sparsity of $A$ implicitly and the resulting summary may be more easily interpretable. Note, they typically consider the transpose of our problem and select columns instead of rows, and sometimes both. An algorithm [4] can construct a set of $\ell = (2k/\varepsilon^2)(1+o(1))$ columns $R$ so that $\|A - \pi_R(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$. There is an $\Omega(k/\varepsilon)$ lower bound [10], but this enforces that only rows of the original matrix are retained and does not directly apply to our problem. And these are not streaming algorithms.

Although not typically described as streaming algorithms (perhaps because the focus was on sampling columns which already have length $n$) when a matrix is processed row wise there exists algorithms that can use reservoir sampling to become streaming. The best streaming algorithm [13] samples $O(k/\varepsilon^2)$ rows (proportional to their squared norm) to obtain a matrix $R$ so that $\|A - \pi_R(A)\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon\|A\|_F^2$, a weaker additive error bound. These techniques can also build approximate decompositions of $\hat{A}$ instead of using $\pi_R(A)$, but again these decompositions are only known to work with at least 2 passes, and are thus not streaming.

**Runtime Bounds.** There is a wealth of literature on the problem of matrix approximation; most recently two algorithms [7,28] showed how to construct a decomposition of
domized algorithm). An expresion error parameter), and δ words in the RAM model where it is assumed

terms.

to do so, some parts needed to be slightly simplified.

Frequent directions. Finally we mention a recent algorithm by Liberty [23] which runs in O(nd/ε) time, maintains a matrix with 2/ε rows in a row-wise streaming algorithm, and produces a matrix ̂A of rank at most 2/ε so that for any unit vector x of length d satisfies 0 ≤ ∥Ax∥2 − ∥̂Ax∥2 ≤ ε ∥A∥2_F. We will examine a slight variation of this algorithm and describe bounds that it achieves in more familiar terms. This algorithm will be more carefully explained in Section 2.2.

Incremental PCA. We mention one additional line of work on incremental PCA [5, 19, 20, 22, 29]. These approaches attempt to maintain the PCA of a dataset A (using the SVD and a constant amount of additional bookkeeping space) as each row of A arrives in a stream. In particular, after i − 1 rows they consider maintaining A_i^+ and on a new row a, compute svd([A_i^+; a]) = U_i^+S_i^+(V_i^+)T and, then only retain its top rank k approximation as A_i^{k+1}_+ = U_k^+S_k^+(V_k^+)T. This is remarkably similar to Liberty’s algorithm [23], but is missing the Misra-Gries [26] step (we describe Liberty’s algorithm in more detail in Section 2.2). As a result, incremental PCA can have arbitrarily bad error on adversarial data.

Consider an example where the first k rows generate a matrix A_k with kth singular value σ_k = 10. Then each row thereafter a_i for i > k is orthogonal to the first k rows of A, and has norm 5. This will cause the (k + 1)th right singular vector and value σ_k+1 of svd([A_k^+; a_i]) to exactly describe the subspace of a_i with σ_k+1 = 5. Thus this row a_i will always be removed on the processing step and A_k^{k+1}+ will be unchanged from A_k^+ k. If all rows a_i for i > k are pointing in the same direction, this can cause arbitrarily bad errors of all forms of measuring approximation error considered above.

1.2 Catalog of Related Bounds Tables 1, 2 and 3 summarize existing algorithms in landscape of work in low-rank matrix approximation. We grouped them into three main categories: Streaming, Fast Runtime, and Column Sampling. We also tried to write bounds in a consistent compatible format. To do so, some parts needed to be slightly simplified. The space and time bounds are given in terms of n (the number of rows), d (the number of columns), k (the specified rank to approximate), ε (the rank of input matrix A), ε (an error parameter), and δ (the probability of failure of a randomized algorithm). An expression ̂O(x) hides poly log(x) terms.

The size is sometimes measured in terms of the number of columns (#C) and/or the number of rows (#R). Otherwise, if #R or #C is not specified the space refers the number of words in the RAM model where it is assumed O(log nd) bits fit in a single word. The error is of one of several forms.

- A projection result builds a subspace G so that ̂A = π_G(A), but does not actually construct π_G(A). This is denoted by P. Ideally rank(G) = k. When that is not the case, then it is denoted P_ε where ε is replaced by the rank of G.
- A construction result builds a series of (usually 3) matrices (say C, U, and R) where A = CUR. Note again, it does not construct ̂A since it may be of larger size than all of C, U, and R together, but the three matrices can be used in place of ̂A. This is denoted C.
- ε-relative error is of the form ∥A − ̂A∥_F ≤ (1 + ε)∥A − A_k∥_F where A_k is the best rank-k approximation to A. This is denoted εR.
- ε-additive error is of the form ∥A − ̂A∥_F^2 ≤ ∥A − A_k∥_F^2 + ε ∥A∥_F^2. This is denoted εA.

This can sometimes also be expressed as a spectral norm of the form ∥A − ̂A∥_F ≤ ∥A − A_k∥_F^2 + ε ∥A∥_F^2, (note the error term ε ∥A∥_F^2 still has a Frobenius norm). This is denoted εL_2.

- In a few cases the error does not follow these patterns and we specially denote it.
- Algorithms are randomized unless it is specified. In all tables we state bounds for a constant probability of failure. If we want to decrease the probability of failure to some parameter δ, we can generally increase the size and runtime by O(log(1/δ)).

1.3 Our Results Our main result is a deterministic relative error bound for low-rank matrix approximation. A major highlight is that all proofs are, we believe, quite easy to follow.

Low-rank matrix approximation. We slightly adapt the streaming algorithm of Liberty [23], called Frequent Directions to maintain ̂A = [k + k/ε] rows, which outputs an ̂A of rank k approximation to Q (which turns out to be its top k rows). We then consider Q_k a k × d matrix, the best rank k approximation to Q (which turns out to be its top k rows). We show that

∥A − π_{Q_k}(A)∥_F^2 ≤ (1 + ε)∥A − A_k∥_F^2

and that

∥A − A_k∥_F^2 ≤ ||A||_F^2 − ||Q_k||_F^2 ≤ (1 + ε)∥A − A_k∥_F^2.

This algorithm runs in O(ndk^2/ε^2) time. If we allow ξ = c[k + k/ε] for any constant c > 1, then it can be made to run in O(ndk/ε) time with the same guarantees on Q_k.

This is the smallest space streaming algorithm known for these bounds. Also, it is deterministic, whereas previous algorithms were randomized.
### Streaming algorithms

<table>
<thead>
<tr>
<th>Paper</th>
<th>Space</th>
<th>Time</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKM06 [13] LinearTimeSVD</td>
<td>$#R = O(1/\varepsilon^2)$ $O((d + 1/\varepsilon^2)/\varepsilon^4)$</td>
<td>$O((d + 1/\varepsilon^2)/\varepsilon^4 + \text{nnz}(A))$</td>
<td>$P, \varepsilon L_2$</td>
</tr>
<tr>
<td>Sar06 [31] turnstile</td>
<td>$#R = O(k/\varepsilon^2) O((k/\varepsilon^2)^2(d + k/\varepsilon^2))$</td>
<td>$O((k/\varepsilon^2)^2(d + k/\varepsilon^2) + \text{nnz}(A))$</td>
<td>$P, \varepsilon A$</td>
</tr>
<tr>
<td>CW09 [6]</td>
<td>$#R = O(k/\varepsilon)$</td>
<td>$O(nk^2/\varepsilon^2)$</td>
<td>$P, \varepsilon R$</td>
</tr>
<tr>
<td>CW09 [6]</td>
<td>$#R = (n + d)(k/\varepsilon)$</td>
<td>$O(nk^2/\varepsilon^2 + nk/\varepsilon)$</td>
<td>$P, \varepsilon R$</td>
</tr>
<tr>
<td>FSS13 [16]</td>
<td>deterministic</td>
<td>$O((dk/\varepsilon) \log n)$</td>
<td>$n((dk/\varepsilon) \log n)^{(1)}$</td>
</tr>
</tbody>
</table>
| Lib13 [23] | deterministic, $\text{rank}(Q) \leq 2/\varepsilon$ | $\#R = 2/\varepsilon$ $O(d/\varepsilon)$ | $O(nk/\varepsilon)$ | Any unit vector $x$
| & | & | & $0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \varepsilon \|A\|^2_F$ |
| Lib13 [23] | deterministic, $\rho = \|A\|^2_F/\|A\|_2^2$ | $\#R = O(\rho/\varepsilon)$ $O(d/\varepsilon)$ | $O(nk/\varepsilon)$ | $P_{O(\rho/\varepsilon), \varepsilon L_2}$ |
| This paper | deterministic | $\#R = (k/\varepsilon + k)$ $O(dk/\varepsilon)$ | $O(nk^2/\varepsilon^2)$ | $P, \varepsilon R$ |
| & | | $\#R = c(k/\varepsilon + k)$, $c > 1$ $O(dk/\varepsilon)$ | $O(nk^2/\varepsilon^2)$ | $P, \varepsilon R$ |

Table 1: Low-rank matrix approximation algorithms in streaming model.

### Algorithms with Fast Runtime

<table>
<thead>
<tr>
<th>Paper</th>
<th>Space</th>
<th>Time</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM01 [1]</td>
<td>$O(nd)$</td>
<td>$O(nd)$</td>
<td>$|A - A_k|_2 \leq |A - A_k|<em>2 + 10(\max</em>{i,j}</td>
</tr>
<tr>
<td>CW13 [7]</td>
<td>$O((k/\varepsilon)^2 \log^4(k/\varepsilon)/(nk/\varepsilon) \log^2(k/\varepsilon) (nk/\varepsilon) \log(k/\varepsilon))$</td>
<td>$|A - A_k|_2 + O(nk^2/\varepsilon^4 + k^3/\varepsilon^6)$</td>
<td>$C, \varepsilon R$</td>
</tr>
<tr>
<td>NN13 [28]</td>
<td>$O(nk^{1.37}e^{-3.37} + k^{2.37}e^{-4.37})$</td>
<td>$O(nk^{1.37}e^{-3.37} + k^{2.37}e^{-4.37})$</td>
<td>$C, \varepsilon R$</td>
</tr>
</tbody>
</table>

Table 2: Low-rank matrix approximation algorithms with fast runtime.
### Column Sampling algorithms

<table>
<thead>
<tr>
<th>Paper</th>
<th>Space</th>
<th>Time</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>FKV04 [17]</td>
<td>$O(k^2/\varepsilon^6 \max(k^4, \varepsilon^{-2}))$</td>
<td>$O(k^n/\varepsilon^6 \max(k^4, \varepsilon^{-2}))$</td>
<td>P, εA</td>
</tr>
<tr>
<td>DVO6 [10]</td>
<td>#C = $O(k/\varepsilon + k^4 \log k)$</td>
<td>$O(nnz(A)(k/\varepsilon + k^4 \log k) + (n + d)(k^2/\varepsilon^2 + k^3 \log(k/\varepsilon) + k^4 \log^2 k))$</td>
<td>P, εR</td>
</tr>
<tr>
<td>DKM06 [13]</td>
<td>“LinearTimeSVD”</td>
<td>#C = $O(k/\varepsilon^2)$</td>
<td>$(n + 1)/\varepsilon^4 + nnz(A)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#C = $O(k/\varepsilon^2)$</td>
<td>$((k/\varepsilon^2)^2(n + k/\varepsilon^4) + nnz(A))$</td>
</tr>
<tr>
<td>DKM06 [13]</td>
<td>“ConstantTimeSVD”</td>
<td>#C+R = $O(k^2/\varepsilon^3)$</td>
<td>$O((1/\varepsilon^2 + nk/\varepsilon^4 + nnz(A))$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#C+R = $O(k^2/\varepsilon^3)$</td>
<td>$O((k^2/\varepsilon^2 + nk^3/\varepsilon^4))$</td>
</tr>
<tr>
<td>DMM08 [15]</td>
<td>“CUR”</td>
<td>#C = $O(k^2/\varepsilon^2)$</td>
<td>$(n/\varepsilon^2)$</td>
</tr>
<tr>
<td>MD09 [24]</td>
<td>“ColumnSelect”</td>
<td>#C = $O(k^2/\varepsilon^2)$</td>
<td>$(n/\varepsilon^2)$</td>
</tr>
<tr>
<td>BDM11 [4]</td>
<td>#C = $2k/\varepsilon(1 + o(1))$</td>
<td>$O((ndk + dk^3)\varepsilon^{-2/3})$</td>
<td>$P_{2k/\varepsilon(1 + o(1))}, \varepsilon R$</td>
</tr>
</tbody>
</table>

Table 3: Column Sampling based low-rank matrix approximation algorithms.

We note that it is sometimes desirable for the bounds to be written without squared norms, for instance as $\|A - \pi_{Q_k}(A)\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$. For $\varepsilon > 0$, if we take the square root of both sides of the bound above $\|A - \pi_{Q_k}(A)\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$, then we still get a $\sqrt{1 + \varepsilon} \leq (1 + \varepsilon)$ approximation.

No sparse Frequent Directions. We also consider trying to adapt the Frequent Directions algorithm to column sampling (or rather row sampling), in a way that the rows it maintains are the rows from the original matrix $A$ (possibly re-weighted). This would implicitly preserve the sparsity of $A$ in $Q$. We show that this is, unfortunately, not possible, at least not in the most obvious adaptation.

1.4 Matrix Notation Here we quickly review some notation. An $n \times d$ matrix $A$ can be written as a set of $n$ rows as $[a_1; a_2; \ldots; a_n]$ where each $a_i$ is a row of length $d$. Alternatively a matrix $V$ can be written as a set of columns $[v_1, v_2, \ldots, v_d]$.

The Frobenius norm of a matrix $A$ is defined $\|A\|_F = \sqrt{\sum_{i=1}^n ||a_i||^2}$ where $||a_i||$ is Euclidean norm of $a_i$. Let $A_k$ be the best rank $k$ approximation of the matrix $A$, specifically $A_k = \arg\max_{C: \text{rank}(C) \leq k} \|A - C\|_F$.

Given a row $r$ and a matrix $X$ let $\pi_X(r)$ be a projection operation of $r$ onto the subspace spanned by $X$. In particular, we will project onto the row space of $X$, and this can be written as $\pi_X(r) = rX^T((XX^T)^+X$ where $Y^+$ indicates the Moore-Penrose pseudoinverse of $Y$. But whether it projects to the row space or the column space will not matter since we will always use the operator inside of a Frobenius norm.

This operator can be defined to project matrices $R$ as well, denoted as $\pi_X(R)$, where this can be thought of as projecting each row of the matrix $R$ individually.

2 Review of Related Algorithms

We begin by reviewing two streaming algorithms that our results can be seen as an extension. The first is an algorithm for heavy-hitters from Misra-Gries [26] and its improved analysis by Berinde et al. [3]. We re-prove the relevant part of these results. Next we describe the algorithm of Liberty [23] for low-rank matrix approximation that our analysis is based on. We again re-prove his result, with a few additional intermediate results we will need for our extended analysis. One familiar with the work of Misra-Gries [26], Berinde et al. [3], and Liberty [23] can skip this section, although we will refer to some lemmas re-proven below.

2.1 Relative Error Heavy-Hitters Let $A = \{a_1, \ldots, a_n\}$ be a set of $n$ elements where each $a_i$ is in $[u]$. Let $f_j = |\{a_i \in A \mid a_i = j\}|$ for $j \in [u]$. Assume without loss of generality that $f_j \geq f_{j+1}$ for all $j$, and define $F_k = \sum_{j=1}^k f_j$. This is just for notation, and not known ahead of time by algorithms.

The Misra-Gries algorithm [26] finds counts $f_j$ so that for all $j \in [u]$ we have $0 \leq f_j - f_j \leq n/\ell$. It only uses $\ell$ counters and $\ell$ associated labels and works in a streaming manner as follows, starting with all counters empty (i.e. a count of 0). It processes each $a_i$ in (arbitrary) order.

- If $a_i$ matches a label, increment the associated counter.
- If not, and there is an empty counter, change the label
of the counter to \( a_i \) and set its counter to 1.

- Otherwise, if there are no empty counters, then decrement all counters by 1.

To return \( \hat{f}_j \), if there is a label with \( j \), then return the associated counter; otherwise return 0.

Let \( r \) be the total number of times that all counters are decremented. We can see that \( r < n/\ell \) since each time one counter is decremented then all \( \ell \) counters (plus the new element) are decremented and must have been non-empty before hand. Thus this can occur at most \( n/\ell \) times otherwise we would have decremented more counts than elements. This also implies that \( f_j - \hat{f}_j \leq r < n/\ell \) since we only do not count an element if it is removed by one of \( r \) decrements.

This simple, clever algorithm, and its variants, have been rediscovered several times [9, 18, 21, 25].

Define \( F_k = \sum_{j=1}^n f_j \) and let \( R_k = \sum_{j=k+1}^n f_j = n - F_k \). The value \( R_k \) represents the total counts that cannot be described (even optimally) if we only use \( k \) counters. A bound on \( F_k - \hat{F}_k \) in terms of \( R_k \) is more interesting than one in terms of \( n \), since this algorithm is only useful when there are only really \( k \) items that matter and the rest can be ignored. We next reprove a result of Berinde et al. [3] (in their Appendix A).

**Lemma 2.1.** (Berinde et al. [3]) The number of decrements is at most \( r \leq R_k/\ell \).

**Proof.** On each of \( r \) decrements at least \( \ell - k \) counters not in the top \( k \) are decremented. These decrements must come from \( R_k \), so each can be charged to at least one count in \( R_k \); the inequality follows. \( \square \)

**Theorem 2.1.** When using \( \ell = [k + k/\varepsilon] \) in the Misra-Gries algorithm \( F_k - \hat{F}_k \leq \varepsilon R_k \) and \( f_j - \hat{f}_j \leq \frac{\varepsilon}{k} R_k \).

If we use \( \ell = [k + 1/\varepsilon] \), then \( f_j - \hat{f}_j \leq \frac{\varepsilon}{k} R_k \).

**Proof.** Using Lemma 2.1 we have \( r \leq R_k/\ell \). Since for all \( j \) we have \( f_j - \hat{f}_j \leq r \), then \( F_k - \hat{F}_k \leq \frac{k}{\ell} R_k \). Finally, setting \( \ell = k + k/\varepsilon \) results in \( F_k - \hat{F}_k \leq \varepsilon R_k \) and \( r \leq \frac{R_k}{\ell - k} = \frac{\varepsilon}{k} R_k \).

Setting \( \ell = k + 1/\varepsilon \) results in \( f_j - \hat{f}_j \leq r \leq \frac{R_k}{\ell - k} = \varepsilon R_k \) for any \( j \). \( \square \)

This result can be viewed as a warm up for the rank \( k \) matrix approximation to come, as those techniques will follow a very similar strategy.

### 2.2 Additive Error Frequent Directions

Recently Liberty [23] discovered how to apply this technique towards sketching matrices. Next we review his approach, and for perspective and completeness re-prove his main results.

**Algorithm.** The input to the problem is an \( n \times d \) matrix \( A \) that has \( n \) rows and \( d \) columns. It is sometimes convenient to think of each row \( a_i \) as a point in \( \mathbb{R}^d \). We now process \( A \) one row at a time in a streaming fashion always maintaining an \( \ell \times d \) matrix such that for any unit vector \( x \in \mathbb{R}^d \)

\[
(2.1) \quad \|Ax\|^2 - \|Qx\|^2 \leq \|A\|_F^2/\ell,
\]

This invariant (2.1) guarantees that in any “direction” \( x \) (since \( x \) is a unit vector in \( \mathbb{R}^d \)), that \( A \) and \( Q \) are close, where close is defined by the Frobenius norm of \( \|A\|_F^2 \) over \( \ell \).

Liberty’s algorithm is described in Algorithm 2.1. At the start of each round, the last row of \( Q \) will be all zeros. To process each row \( a_i \), we replace the last row (the \( \ell \)th row) of \( Q \) with \( a_i \) to create a matrix \( Q_i \). We take the SVD of \( Q_i \) as \([U, S, V] = \text{svd}(Q_i)\). Let \( \delta = s_{\ell}^2 \), the last (and smallest) diagonal value of \( S \), and in general let \( s_j \) be the \( j \)th diagonal value so \( S = \text{diag}(s_1, s_2, \ldots, s_{\ell}) \). Now set \( s'_j = \sqrt{s_j^2 - \delta} \) for \( j \in [\ell] \), and notice that all values are non-negative and \( s'_0 = 0 \). Let \( S' = \text{diag}(s'_1, s'_2, \ldots, s'_\ell) \). Finally set \( Q = S'V^T \).

**Algorithm 2.1 Frequent Directions (Liberty [23])**

Initialize \( Q^0 \) as an all zeros \( \ell \times d \) matrix.

for each row \( a_i \in A \) do

Set \( Q_+ \leftarrow Q^{i-1} \) with last row replaced by \( a_i \)

\[ [Z, S, Y] = \text{svd}(Q_+) \]

\[ C' = SY^T \quad \text{[only for notation]} \]

Set \( \delta_i = s_{\ell}^2 \) \[ \text{[the \( \ell \)th entry of \( S \), squared]} \]

Set \( S' = \text{diag}(\sqrt{s_1^2 - \delta_i}, \sqrt{s_2^2 - \delta_i}, \ldots, \sqrt{s_{\ell-1}^2 - \delta_i}, 0) \)

Set \( Q^i = S'Y^T \)

return \( Q = Q^n \)

It is useful to interpret each row of \( Y^T \) as a “direction,” where the first row is along the direction with the most variance, all rows are orthogonal, and all rows are sorted in order of variance given that they are orthogonal to previous rows. Then multiplying by \( S' \) scales the \( j \)th row \( y_j \) of \( Y^T \) by \( s'_j \). Since \( s'_0 = 0 \), then the last row of \( Q^i \) must be zero.

**Analysis.** Let \( \Delta = \sum_{i=1}^n \delta_i \).

**Lemma 2.2.** For any unit vector \( x \in \mathbb{R}^d \) we have

\[
\|C^i x\|^2 - \|Q^i x\|^2 \leq \delta_i.
\]
Proof. Let \( Y_j \) be the \( j \)th column of \( Y \), then

\[
\|C^i x\|^2 = \sum_{j=1}^\ell s_j^2(y_j, x)^2
\]

\[
= \sum_{j=1}^\ell ((s_j')^2 + \delta_i)(y_j, x)^2
\]

\[
= \sum_{j=1}^\ell (s_j')^2(y_j, x)^2 + \delta_i \sum_{j=1}^\ell (y_j, x)^2
\]

\[
\leq \|Q^i x\|^2 + \delta_i
\]

Subtracting \( \|Q^i x\|^2 \) from both sides completes the proof. \( \square \)

**Lemma 2.3.** For any unit vector \( x \in \mathbb{R}^d \) we have

\[
0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \Delta.
\]

Proof. Notice that \( \|C^i x\|^2 = \|Q^{i-1} x\|^2 + \|a_i x\|^2 \) for all \( 2 \leq i \leq n \) and that \( \|Q^1 x\|^2 = \|a_1 x\|^2 \). By substituting this into inequality from Lemma 2.2, we get

\[
\|Q^{i-1} x\|^2 + \|a_i x\|^2 \leq \|Q^i x\|^2 + \delta_i
\]

Subtracting \( \|Q^{i-1} x\|^2 \) from both sides and summing over \( i \) reveals

\[
\|Ax\|^2 = \sum_{i=1}^n \|a_i x\|^2
\]

\[
\leq \sum_{i=1}^n (\|Q^i x\|^2 - \|Q^{i-1} x\|^2 + \delta_i)
\]

\[
= \|Q^n x\|^2 - \|Q^0 x\|^2 + \sum_{i=1}^n \delta_i
\]

\[
= \|Q^n x\|^2 + \Delta.
\]

Subtracting \( \|Q^n x\|^2 = \|Qx\|^2 \) from both sides proves the second inequality of the lemma.

To see the first inequality observe \( \|Q^{i-1} x\|^2 + \|a_i x\|^2 = \|C^i x\|^2 \geq \|Q^i x\|^2 \) for all \( 1 \leq i \leq n \). Then we can expand

\[
\|Ax\|^2 = \sum_{i=1}^n \|a_i x\|^2
\]

\[
= \sum_{i=1}^n (\|Q^i x\|^2 - \|Q^{i-1} x\|^2)
\]

\[
\geq \sum_{i=1}^n (\|Q^i x\|^2 - \|Q^{i-1} x\|^2)
\]

\[
= \|Qx\|^2.
\]

\( \square \)

**Lemma 2.4.** (Liberty [23]) Algorithm 2.1 maintains for any unit vector \( x \) that

\[
0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \|A\|^2 / \ell
\]

and

\[
T = \Delta \ell = \|A\|^2 - \|Q\|^2.
\]

Proof. In the \( i \)th round of the algorithm \( \|C^i\|^2 = \|Q^i\|^2 + \ell \delta_i \) and \( \|C^i\|^2 = \|Q^{i-1}\|^2 + \|a_i\|^2 \). By solving for \( \|a_i\|^2 \) and summing over \( i \) we get

\[
\sum_{i=1}^n \|a_i\|^2 = \sum_{i=1}^n \|Q^i\|^2 - \|Q^{i-1}\|^2 + \ell \delta_i
\]

\[
= \|Q^n\|^2 + \ell \Delta.
\]

This proves the second part of the lemma. Using that \( \|Q^n\| \geq 0 \) we obtain \( \Delta \leq \|A\|^2 / \ell \). Substituting this into Lemma 2.3 yields \( 0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \|A\|^2 / \ell. \) \( \square \)

3 New Relative Error Bounds for Frequent Directions

We now generalize the relative error type bounds for Misra-Gries (in Section 2.1) to the Frequent Directions algorithm (in Section 2.2).

Before we proceed with the analysis of the algorithm, we specify some parameters and slightly modify Algorithm 2.1. We always set \( \ell = \lceil k + k / \epsilon \rceil \). Also, instead of returning \( Q \) in Algorithm 2.1, as described by Liberty, we return \( Q_k \). Here \( Q_k \) is the best rank \( k \) approximation of \( Q \) and can be written \( Q_k = S_k Y_k^T \) where \( S_k \) and \( Y_k \) are the first \( k \) rows of \( S \) and \( y \), respectively. Note that \( y = [y_1, \ldots, y_d] \) are the right singular vectors of \( Q \).

This way \( Q_k \) is also rank \( k \) (and size \( k \times d \)), and will have nice approximation properties to \( A_k \). Recall that \( A_k = U_k \Sigma_k V_k^T \) where \( [U, \Sigma, V] = \text{svd}(A) \) and \( U_k, \Sigma_k, V_k \) are the first \( k \) columns of these matrices, representing the first \( k \) principal directions. Let \( V = [v_1, \ldots, v_d] \) be the right singular vectors of \( A \).

**Lemma 3.1.** \( \Delta \leq \|A - A_k\|^2 / (\ell - k) \).

Proof. Recall that \( T = \Delta \ell = \|A\|^2 - \|Q\|^2 \) is the total squared norm subtracted from all of any set of orthogonal directions throughout the algorithm. Now if \( r = \text{rank}(A) \) we
have:

\[ T = \|A\|_F^2 - \|Q\|_F^2 \]
\[ = \sum_{i=1}^{k} \|Av_i\|^2 + \sum_{i=k+1}^{r} \|Av_i\|^2 - \|Q\|_F^2 \]
\[ = \sum_{i=1}^{k} \|Av_i\|^2 + \|A - A_k\|_F^2 - \|Q\|_F^2 \]
\[ \leq \sum_{i=1}^{k} \|Av_i\|^2 + \|A - A_k\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ = \|A - A_k\|_F^2 + \sum_{i=1}^{k} (\|Av_i\|^2 - \|Qv_i\|^2) \]
\[ \leq \|A - A_k\|_F^2 + k\Delta \]

Third transition holds because \(\sum_{i=1}^{k} \|Qv_i\|^2 < \|Q\|_F^2\) and fifth transition is due to Lemma 2.3 that \(\|Av_i\|^2 - \|Qv_i\|^2 \leq \Delta\). Now we solve for \(T = \Delta\ell \leq \|A - A_k\|_F^2 + k\Delta\) to get \(\Delta \leq \|A - A_k\|_F^2/\ell - k\). \(\square\)

Now we can show that projecting \(A\) onto \(Q_k\) provides a relative error approximation.

**Lemma 3.2.** \(\|A - \pi_{Q_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2\).

**Proof.** Using the vectors \(v_i\) as right singular vectors of \(A\), and letting \(r = \text{rank}(A)\), then we have

\[ \|A - \pi_{Q_k}(A)\|_F^2 = \|A\|_F^2 - \|\pi_{Q_k}(A)\|_F^2 \]
\[ = \|A\|_F^2 - \sum_{i=1}^{k} \|Ay_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qy_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} (\|Av_i\|^2 - \|Qv_i\|^2) \]
\[ = \|A\|_F^2 - \|A - A_k\|_F^2 + k\Delta \]
\[ \leq \|A - A_k\|_F^2 + \frac{k}{\ell - k}\|A - A\|_F^2 \]
\[ = \frac{\ell}{\ell - k}\|A - A_k\|_F^2 \]

Note that first line is true due to Pythagorean theorem. Second transition holds by Lemma 2.3 and since \(\sum_{i=1}^{k} \|Qy_i\|^2 \geq \sum_{i=1}^{k} \|Qv_i\|^2\) third transition holds too. Fourth transition comes from Lemma 2.3 and sixth transition is driven by Lemma 3.1.

Finally, setting \(\ell = [k + k/\varepsilon]\) results in \(\|A - \pi_{Q_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2\). \(\square\)

We would also like to relate the Frobenius norm of \(Q_k\) directly to that of \(A_k\), instead of projecting \(A\) onto it (which cannot be done in a streaming setting, at least not in \(\omega(n)\) space). However, \(\|A - Q_k\|_F\) does not make sense since \(Q_k\) has a different number of rows than \(A\). However, we can decompose \(\|A - A_k\|_F^2 = \|A\|_F^2 - \|A_k\|_F^2\) since \(A_k\) is a projection of \(A\) onto a (the best rank \(k\)) subspace, and we can use the Pythagorean Theorem. Now we can compare \(\|A\|_F^2 - \|A_k\|_F^2\) to \(\|A\|_F^2 - \|Q_k\|_F^2\).

**Lemma 3.3.** \(\|A\|_F^2 - \|A_k\|_F^2 \leq \|A\|_F^2 - \|Q_k\|_F^2 \leq (1 + \varepsilon)(\|A\|_F^2 - \|A_k\|_F^2)\).

**Proof.** The first inequality can be seen since

\[ \|A_k\|_F^2 = \sum_{i=1}^{k} \|Ay_i\|^2 \geq \sum_{i=1}^{k} \|Qy_i\|^2 \]
\[ \geq \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ = \|Q\|_F^2 \]

And the second inequality follows by

\[ \|A\|_F^2 - \|Q_k\|_F^2 = \|A\|_F^2 - \sum_{i=1}^{k} \|Qy_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ \leq \|A\|_F^2 - \sum_{i=1}^{k} \|Qv_i\|^2 \]
\[ \leq \|A\|_F^2 - \|A - A_k\|_F^2 + k\Delta \]
\[ \leq \|A - A_k\|_F^2 + \frac{k}{\ell - k}\|A - A\|_F^2 \]
\[ = \frac{\ell}{\ell - k}\|A - A_k\|_F^2 \]

Finally, setting \(\ell = k + k/\varepsilon\) results in \(\|A\|_F^2 - \|Q_k\|_F^2 \leq (1 + \varepsilon)(\|A\|_F^2 - \|A_k\|_F^2)\). \(\square\)

One may ask why not compare \(\|A_k\|_F\) to \(\|Q_k\|_F\) directly, instead of subtracting from \(\|A\|_F^2\). First note that the above bound does guarantee that \(\|A_k\|_F \geq \|Q_k\|_F\). Second, in situations where a rank \(k\) approximation is interesting, then most of the mass from \(A\) should be in its top \(k\) components. Then \(\|A_k\|_F > \|A - A_k\|_F\) so the above bound is actually tighter. To demonstrate this we can state the following conditional statement comparing \(\|A_k\|_F\) and \(\|Q_k\|_F\).

**Lemma 3.4.** If \(\|A - A_k\|_F \leq \|A_k\|_F\), then

\[(1 - \varepsilon)\|A_k\|_F^2 \leq \|Q_k\|_F^2 \leq \|A_k\|_F^2.\]
Proof. The second inequality follows from Lemma 3.3, by subtracting \( \|A\|_F^2 \). The first inequality uses Lemma 2.3 as follows.
\[
\|A_k\|_F^2 = \sum_{i=1}^k \|A v_i\|^2 \leq \sum_{i=1}^k (\|Q v_i\|^2 + \Delta) \\
\leq \|Q_k\|_F^2 + k\Delta \\
\leq \|Q\|_F^2 + \frac{k}{\ell - k} \|A - A_k\|_F^2 \\
\leq \|Q\|_F^2 + \frac{c}{\ell} \|A\|_F^2.
\]
\[\Box\]

Finally, we summarize all of our bounds about Algorithm 2.1.

**Theorem 3.1.** Given an input \( n \times d \) matrix \( A \), by setting \( \ell = \left[ k + k/\varepsilon \right] \) Algorithm 2.1 runs in time \( O(ndk^2/\varepsilon^2) \) and produces an \( \ell \times d \) matrix \( Q \) that for any unit vector \( x \in \mathbb{R}^d \) satisfies
\[
0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \|A\|_F^2/\ell
\]
and the projection of \( Q \) along its top \( k \) right singular values is a \( k \times d \) matrix \( Q_k \) which satisfies
\[
\|A - \pi_{Q_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2.
\]
and
\[
\|A\|_F^2 - \|A_k\|_F^2 \leq \|A\|_F^2 - \|Q_k\|_F^2 \leq (1 + \varepsilon)(\|A\|_F^2 - \|A_k\|_F^2).
\]

Liberty [23] also observes that by increasing \( \varepsilon \) by a constant \( c > 1 \) and then processing every \( \ell (c - 1) \) elements in a batch setting (each round results in a \( c\ell \) row matrix \( Q \)) then the runtime can be reduced to \( O(\frac{2}{\varepsilon^2} ndk) = O(ndk/\varepsilon) \) at the expense of more space. The same trick can be applied here to use \( \ell = c[k + k/\varepsilon] \) rows in total \( O(ndk/\varepsilon) \) time.

**4 No Sparse Frequent Directions**

In this section we consider extending the Frequent Directions algorithm described in the previous section to a sparse version. The specific goal is to retain a (re-weighted) set of \( \ell \) rows \( Q \) of an input matrix \( A \) so that for any unit vector \( x \in \mathbb{R}^d \) that 0 \leq \|Ax\|^2 - \|Qx\|^2 \leq \|A\|_F^2/\ell, \) and also to hopefully extend this so that \( \|A - \pi_{Q_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2 \) as above. This is an open problem left by Liberty [23]. It is also a useful goal in many scenarios when returning a set of \( k \) singular vectors of a matrix that are linear combinations of inputs are hard to interpret; in this case returning a weighted set of actual rows is much more informative.

In this section we show that this is not possible by directly extending the Frequent Directions algorithm.

In particular we consider processing one row in the above framework. The input to the problem is an \( \ell \times d \) matrix \( Q = \{w_1 \bar{r}_1; \ldots; w_j \bar{r}_j\} \) where each \( w_j \) is a scalar (initially set to \( w_j = \|r_j\| \)). The output of one round should be an \( \ell - 1 \times d \) matrix \( \tilde{Q} = \{w_1 \bar{r}_1; \ldots; w_{j-1} \bar{r}_{j-1}; w_{j+1} \bar{r}_{j+1}; \ldots; w_d \bar{r}_d\} \) where one of the rows, namely \( r_j \), is removed and the rest of the rows are re-weighted.

**Requirements.** To make this process work we need the following requirements. Let \( \delta_i = \min_j \|\perp_{Q_{-j}}(Q)\|^2 \) represent the smallest amount of squared norm resulting from removing one row from \( Q \) by the procedure above. \(^1\) Assume we remove this row, although removing any row is just as difficult but would create even more error.

(P1) The Frobenius norm \( \|\tilde{Q}\|_F^2 \) must be reduced so \( \|\tilde{Q}\|_F^2 \leq \|Q\|_F^2 - c\ell \delta_i \) for some absolute constant \( c \).

The larger the constant (ideally \( c = 1 \)) the smaller the bound on \( \ell \).

(P2) For any unit vector (direction) \( x \in \mathbb{R}^d \) the difference in norms between \( Q \) and \( \tilde{Q} \) must be bounded as \( \|Qx\|^2 \leq \|\tilde{Q}x\|^2 + \delta_i \).

In the direction \( v \) which defines the norm \( \delta_i = \|\perp_{Q_{-j}}(Q)\|^2 = \|Qv\|^2 \) we have \( 0 = \|\tilde{Q}v\|^2 \) and the inequality is tight. And for instance a vector \( u \) in the span of \( Q_{-j} \) we have that \( \|Qu\|^2 \neq \|\tilde{Q}u\|^2 \), which makes the right hand side larger.

If both (P1) and (P2) hold, then we can run this procedure for all rows of \( A \) and obtain a final matrix \( Q \). Using similar analysis as in Section 2.2, for any unit vector \( x \in \mathbb{R}^d \) we can show
\[
\|Ax\|^2 - \|Qx\|^2 \leq \sum_i \delta_i \leq \|A\|_F^2/(c\ell).
\]

**Hard construction.** Consider a \( \ell \times d \) matrix \( Q \) with \( d > \ell \). Let each row of \( Q \) be of the form \( r_j = [1, 0, 0, \ldots, 0, 1, 0, \ldots, 0] \) where there is always a 1 in the first column, and another in the \((j + 1)\)th column for row \( j \); the remaining entries are 0. Let \( x = [1, 0, 0, \ldots, 0] \) be the direction strictly along the dimension represented by the first column.

Now \( \delta_i = \min_j \|\perp_{Q_{-j}}(Q)\|^2 \) = 1, since for any \( j \)th row \( r_j \), when doing an orthogonal projection to \( Q_{-j} \) the remaining vector is always exactly in the \( j \)th column where that row has a squared norm of 1. For notational simplicity, let us assume we choose to remove row \( \ell \).

We now must re-weight rows \( r_1 \) through \( r_{\ell-1} \); let the new weights be \( \hat{w}_j^2 = w_j^2 - \alpha_j \),
\(^1\) Define \( \perp_X(Y) \) as the orthogonal projection of \( Y \) onto \( X \). It projects each row of \( Y \) onto the subspace orthogonal to the basis of \( X \). It can be interpreted as \( \perp_X(Y) = Y - \pi_X(Y) \). Also, we let \( Q_{-j} \) be the matrix \( Q \) after removing the \( j \)th row.
In order to satisfy (P1) we must have
\[ \|Q\|_p^2 - \|\hat{Q}\|_p^2 = \sum_{j=1}^\ell w_j^2 - \sum_{j=1}^{\ell-1} \hat{w}_j^2 = w_\ell^2 + \sum_{j=1}^{\ell-1} \alpha_j \geq c\delta_i. \]
Since \( w_\ell^2 = 2 \) and \( \delta_i = 1 \) we must have \( \sum_{j=1}^{\ell-1} \alpha_j \geq c\ell - 2. \)
In order to satisfy (P2) we consider the vector \( x \) as defined above. We can observe
\[ \|Qx\|^2 = \sum_{j=1}^\ell w_j^2 (\bar{r}_j, x)^2 = \sum_{j=1}^\ell w_j^2 (1/2). \]
and
\[ \|\hat{Q}x\|^2 = \sum_{j=1}^{\ell-1} \hat{w}_j^2 (\bar{r}_j, x)^2 = (1/2) \sum_{j=1}^{\ell-1} (w_j^2 - \alpha_j) = (\|Qx\|^2 - 1) - (1/2) \sum_{j=1}^{\ell-1} \alpha_j. \]
Thus we require that \( \sum_{j=1}^{\ell-1} \alpha_j \leq 2\delta_i - 2 = 0 \), since recall \( \delta_i = 1. \)
Combining these requirements yields that
\[ 0 \geq \sum_{j=1}^{\ell-1} \alpha_j \geq c\ell - 2 \]
which is only valid when \( c \leq 2/\ell. \)
Applying the same proof technique as in Section 2.2 to this process reveals, at best, a bound so that for any direction \( x \in \mathbb{R}^d \) we have
\[ \|Ax\|^2 - \|Qx\|^2 \leq \|A\|_F^2 / 2. \]

Acknowledgements: We thank Edo Liberty for encouragement and helpful comments, as well as pointing out several mistakes in an earlier version of this paper. And thank David P. Woodruff, Christos Boutsidis, Dan Feldman, and Christian Sohler for helping interpret some results.

References


