Improved Practical Matrix Sketching with Guarantees

Mina Ghashami, Amey Desai, and Jeff M. Phillips [⋆]

University of Utah

Abstract. Matrices have become essential data representations for many large-scale problems in data analytics, and hence matrix sketching is a critical task. Although much research has focused on improving the error/size tradeoff under various sketching paradigms, we find a simple heuristic iSVD, with no guarantees, tends to outperform all known approaches. In this paper we adapt the best performing guaranteed algorithm, FrequentDirections, in a way that preserves the guarantees, and nearly matches iSVD in practice. We also demonstrate an adversarial dataset for which iSVD performs quite poorly, but our new technique has almost no error. Finally, we provide easy replication of our studies on APT, a new testbed which makes available not only code and datasets, but also a computing platform with fixed environmental settings.

1 Introduction

Matrix sketching has become a central challenge [3, 16, 20, 28, 33] in large-scale data analysis as many large data sets including customer recommendations, image databases, social graphs, document feature vectors can be modeled as a matrix, and sketching is either a necessary first step in data reduction or has direct relationships to core techniques including PCA, LDA, and clustering.

There are several variants of this problem, but in general the goal it to process an $n \times d$ matrix A to somehow represent a matrix B so $\|A - B\|_F$ or (examining the covariance) $\|A^T A - B^T B\|_2$ is small.

In both cases, the best rank-k approximation A_k can be computed using the singular value decomposition (svd); however this takes $O(nd \min(n, d))$ time and $O(nd)$ memory. This is prohibitive for modern applications which usually desire a small space streaming approach, or even an approach that works in parallel. For instance diverse applications receive data in a potentially unbounded and time-varying stream and want to maintain some sketch B. Examples of these applications include data feeds from sensor networks [6], financial tickers [10,41], on-line auctions [5], network traffic [23, 38], and telecom call records [13].

In recent years, extensive work has taken place to improve theoretical bounds in the size of B. Random projections [3, 37] and hashing [11, 40] approximate A in B as a random linear combination of rows and/or columns of A. Column sampling methods [7, 15–17, 19, 30, 36] choose a set of columns (and/or rows) from A to represent B; the best bounds require multiple passes over the data. We refer readers to recent work [11, 22] for extensive discussion of various models and error bounds.

[⋆] Thanks to supported by NSF CCF-1115677, IIS-1251019, and CCF-1350888.

Very recently Liberty [28] introduced a new technique FREQUENTDIRECTIONS (abbreviated FD) which is deterministic, achieves the best bounds on the covariance $\|A^TA - B^TB\|_2$ error, the direct error $\|A - B\|_F^2$ [22] (using B as a projection), and moreover, it seems to greatly outperform the projection, hashing, and column sampling techniques in practice.

However, there is a family of heuristic techniques [8, 24, 25, 27, 35] (which we refer to as iSVD, described relative to FD in Section 2), which are used in many practical settings, but are not known to have any error guarantees. In fact, we observe (see Section 3) on many real and synthetic data sets that iSVD noticeably outperforms FD, yet there are adversarial examples where it fails dramatically.

Thus in this paper we ask (and answer in the affirmative), *can one achieve a matrix sketching algorithm that matches the usual-case performance of iSVD, and the adversarial-case performance of FD, and error guarantees of FD?*

1.1 Notation and Problem Formalization

We denote an $n \times d$ matrix A 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 $[v_1, v_2, \ldots, v_d]$. We assume $d \ll n$. We will consider streaming algorithms where each element of the stream is a row a_i of A.

The squared Frobenius norm of a matrix A is defined $\|A\|_F^2 = \sum_{i=1} \|a_i\|^2$ where $\|a_i\|$ is Euclidean norm of row a_i, and it intuitively represents the total size of A. The spectral norm $\|A\|_2 = \max_{x:\|x\|=1} \|Ax\|$, and represents the maximum influence along any unit direction x. It follows that $\|A^TA - B^TB\|_2 = \max_{x:\|x\|=1} |\|Ax\|^2 - \|Bx\|^2|$.

Given a matrix A and a low-rank matrix X let $\pi_X(A) = AX^T(XX^T)^+X$ be a *projection* operation of A onto the rowspace spanned by X; that is if X is rank r, then it projects to the r-dimensional subspace of points (e.g. rows) in X. Here X^+ indicates taking the Moore-Penrose pseudoinverse of X.

The singular value decomposition of A, written $\mathsf{svd}(A)$, produces three matrices $[U, S, V]$ so that $A = USV^T$. Matrix U is $n \times n$ and orthogonal. Matrix V is $d \times d$ and orthogonal; its columns $[v_1, v_2, \ldots, v_d]$ are the right singular vectors, describing directions of most covariance in A^TA. S is $n \times d$ and is all 0s except for the diagonal entries $\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$, the *singular values*, where $r \leq d$ is the rank. Note that $\sigma_j \geq \sigma_{j+1}$, $\|A\|_2 = \sigma_1$, and $\sigma_j = \|Av_j\|$ describes the norm along direction v_j.

Frequent Directions bounds. We describe FD in detail in Section 2, here we state the error bounds precisely. From personal communication [21], in a forth-coming extension of the analysis by Liberty [28] and Ghashami and Phillips [22], it is shown that there exists a value Δ that FD, run with parameter ℓ, satisfies three facts (for $\alpha = 1$):

- Fact 1: For any unit vector x we have $\|Ax\|^2 - \|Bx\|^2 \geq 0$.
- Fact 2: For any unit vector x we have $\|Ax\|^2 - \|Bx\|^2 \leq \Delta$.
- Fact 3: $\|A\|_F^2 - \|B\|_F^2 \geq \alpha\Delta\ell$.

Their analysis shows that *any* algorithm that follows these facts (for any $\alpha \in (0,1]$, and any $k \leq \alpha\ell$ including $k = 0$ where A_0 is the all zeros matrix) satisfies $\Delta \leq \|A - A_k\|_F^2/(\alpha\ell - k)$; hence for any unit vector x we have $0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \Delta \leq \|A - A_k\|_F^2/(\alpha\ell - k)$. Furthermore *any* such algorithm also satisfies $\|A - \pi_{B_k}(A)\| \leq \alpha\ell\Delta \leq \|A - A_k\|_F^2 \alpha\ell/(\alpha\ell - k)$, where $\pi_{B_k}(\cdot)$ projections onto B_k, the top k singular vectors of B. Since these results are actually proven only with $\alpha = 1$, we reprove these slightly extended versions in Appendix A.

FD maintains an $\ell \times d$ matrix B (i.e. using $O(\ell d)$ space), with $\alpha = 1$. Thus setting $\ell = k + 1/\varepsilon$ achieves $\|A^T A - B^T B\|_2 \leq \varepsilon\|A - A_k\|_F^2$, and setting $\ell = k + k/\varepsilon$ achieves $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$.

1.2 Frequency Approximation, Intuition, and Results

FD is inspired by an algorithm by Misra and Gries [32] for the streaming frequent items problem. That is, given a stream $S = \langle s_1, s_2, \ldots, s_n \rangle$ of items $s_i \in [u] = \{1, 2, \ldots, u\}$, represent $f_j = |\{s_i \in S \mid s_i = j\}|$; the frequency of each item $j \in [u]$. The MG sketch uses $O(1/\varepsilon)$ space to construct an estimate $\hat{f}_j$ (for *all* $j \in [u]$) so that $0 \leq f_j - \hat{f}_j \leq \varepsilon n$. In brief it keeps $\ell - 1 = 1/\varepsilon$ counters, each labeled by some $j \in [u]$: it increments a counter if the new item matches the associated label or for an empty counter, and it decrements all counters if there is no empty counter and none match the stream element. $\hat{f}_j$ is the associated counter value for j, or 0 if there is no associated counter.

Intuitively (as we will see in Section 2), FD works similarly treating the singular vectors of B as labels and the squared singular values as counters.

This MG algorithm and variants have been rediscovered several times [14, 26, 31] and can be shown to process elements in $O(1)$ time. In particular, the SpaceSaving (SS) algorithm [31] has similar guarantees ($0 \leq \hat{f}_j - f_j \leq \varepsilon n$ for all $j \in [u]$) and also uses ℓ counters/labels. But when no counter is available, it does the unintuitive step of replacing the label of the least counter with the current stream element, and incrementing it by one. $\hat{f}_j$ is the associated counter, or otherwise the value of the minimum counter. A masterful empirical study by Cormode and Hadjieleftheriou [12] demonstrates that the SpaceSaving approach can outperform the standard MG step, and Agarwal *et al.* [4] shows that one can isomorphically convert between them by adding the number of decrements to each estimate from the MG sketch.

Main results. To improve on FD empirically, it is natural to ask if variants mimicking SpaceSaving can be applied. We present two approaches SPACESAVING DIRECTIONS (abbreviated SSD, which directly mimics the SpaceSaving algorithm) and COMPENSATIVE FREQUENTDIRECTIONS (abbreviated CFD, which mimics the conversion from MG to SS sketch). These are not isomorphic in the matrix setting as is the case in the items setting, but we are able to show strong error guarantees for each, asymptotically equivalent to FD. However, while these sometimes improve empirically on FD, they do not match iSVD.

Rather, we achieve our ultimate goal with another approach PARAMETRIZED FREQUENTDIRECTIONS; it has parameter α and is abbreviated α-FD. It smoothly translates between FD and iSVD (FD = 1-FD and iSVD = 0-FD), and for $\alpha = 0.2$

we empirically demonstrate that it nearly matches the performance of iSVD on several synthetic and real data sets. It also has the same asymptotic guarantees as FD. Furthermore, we construct an adversarial data set where iSVD performs dramatically worse than FD and all proposed algorithms, including α-FD. We observe similar effects where α-FD and FD outperform iSVD on a large real-world data set that shares some properties with the adversarial synthetic one.

Finally, to ensure our results are easily and readily *reproducible*, we implement all experiments on a new extension of Emulab [39] called APT [34]. It allows one to check out a virtual machine with the same specs as we run our experiments, load our precise environments and code and data sets, and directly reproduce all experiments.

2 Algorithms

The main structure of the algorithm we will study is presented in Algorithm 2.1, where $S' \leftarrow \text{ReduceRank}(S)$ is a subroutine that differs for each variant we consider. It sets at least one non-zero in S to 0 in S'; this leads to a reduced rank for $B_{[i]}$, in particular with one row as all 0s. Notationally we use σ_j as the jth singular value in S, and σ'_j as the jth singular value in S'.

Algorithm 2.1 (Generic) FD Algorithm

```
Input: $\ell, \alpha \in (0,1], A \in \mathbb{R}^{n \times d}$
$B_{[0]} \leftarrow$ all zeros matrix $\in \mathbb{R}^{\ell \times d}$
for $i \in [n]$ do
    Insert $a_i$ into a zero valued rows of $B_{[i-1]}$; result is $B_{[i]}$
    if ($B_{[i]}$ has no zero valued rows) then
        $[U, S, V] \leftarrow \mathsf{svd}(B_{[i]})$
        $C_{[i]} = SV^T$                                   # Only needed for proof notation
        $S' \leftarrow \text{ReduceRank}(S)$
        $B_{[i]} \leftarrow S'V^T$
return $B = B_{[n]}$
```

For FD, ReduceRank sets each $\sigma'_j = \sqrt{\sigma_j^2 - \delta_i}$ where $\delta_i = \sigma_\ell^2$.

For iSVD, ReduceRank keeps $\sigma'_j = \sigma_j$ for $j < \ell$ and sets $\sigma'_\ell = 0$.

The runtime of FD can be improved [28] by doubling the space, and batching the svd call. A similar approach is possible for variants we consider.

2.1 Parameterized FD

Parameterized FD uses the following subroutine (Algorithm 2.2) to reduce the rank of the sketch; it zeros out row ℓ. This method has an extra parameter $\alpha \in [0,1]$ that describes the fraction of singular values which will get affected in the ReduceRank subroutine. Note iSVD has $\alpha = 0$ and FD has $\alpha = 1$. The intuition is that the smaller singular values are more likely associated with noise terms and the larger ones with signals, so we should avoid altering the signal terms in the ReduceRank step.

Here we show error bounds asymptotically matching FD for α-FD (for constant $\alpha > 0$), by showing the three Facts hold. We use $\Delta = \sum_{i=1}^{n} \delta_i$.

Algorithm 2.2 ReduceRank-PFD(S, α)

```
$\delta_i \leftarrow \sigma_\ell^2$
return diag$(\sigma_1, \ldots, \sigma_{\ell(1-\alpha)}, \sqrt{\sigma_{\ell(1-\alpha)+1}^2 - \delta_i}, \ldots, \sqrt{\sigma_\ell^2 - \delta_i})$
```

Lemma 1. *For any unit vector x and any $\alpha \geq 0$: $0 \leq \|C_{[i]}x\|^2 - \|B_{[i]}x\|^2 \leq \delta_i$.*

Proof. The right hand side is shown by just expanding $\|C_{[i]}x\|^2 - \|B_{[i]}x\|^2$.

$$\|C_{[i]}x\|^2 - \|B_{[i]}x\|^2 = \sum_{j=1}^{\ell} \sigma_j^2 \langle v_j, x\rangle^2 - \sum_{j=1}^{\ell} {\sigma'}_j^2 \langle v_j, x\rangle^2 = \sum_{j=1}^{\ell} (\sigma_j^2 - {\sigma'}_j^2)\langle v_j, x\rangle^2$$
$$= \delta_i \sum_{j=(1-\alpha)\ell+1}^{\ell} \langle v_j, x\rangle^2 \leq \delta_i \|x\|^2 = \delta_i$$

To see the left side of the inequality $\delta_i \sum_{j=(1-\alpha)\ell+1}^{\ell} \langle v_j, x\rangle^2 \geq 0$. □

Then summing over all steps of the algorithm (using $\|a_i x\|^2 = \|C_{[i]}x\|^2 - \|B_{[i-1]}x\|^2$) it follows (see Lemma 2.3 in [22]) that

$$0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \sum_{i=1}^{n} \delta_i = \Delta,$$

proving Fact 1 and Fact 2 about α-FD for any $\alpha \in [0, 1]$.

Lemma 2. *For any $\alpha \in (0, 1]$, $\|A\|_F^2 - \|B\|_F^2 = \alpha\Delta\ell$, proving Fact 3.*

Proof. We expand that $\|C_{[i]}\|_F^2 = \sum_{j=1}^{\ell} \sigma_j^2$ to get

$$\|C_{[i]}\|_F^2 = \sum_{j=1}^{(1-\alpha)\ell} \sigma_j^2 + \sum_{j=(1-\alpha)\ell+1}^{\ell} \sigma_j^2$$
$$= \sum_{j=1}^{(1-\alpha)\ell} {\sigma'}_j^2 + \sum_{j=(1-\alpha)\ell+1}^{\ell} ({\sigma'}_j^2 + \delta_i) = \|B_{[i]}\|_F^2 + \alpha\ell\delta_i.$$

By using $\|a_i\|^2 = \|C_{[i]}\|_F^2 - \|B_{[i-1]}\|_F^2 = (\|B_{[i]}\|_F^2 + \alpha\ell\delta_i) - \|B_{[i-1]}\|_F^2$, and summing over i we get

$$\|A\|_F^2 = \sum_{i=1}^{n} \|a_i\|^2 = \sum_{i=1}^{n} \|B_{[i]}\|_F^2 - \|B_{[i-1]}\|_F^2 + \alpha\ell\delta_i = \|B\|_F^2 + \alpha\ell\Delta.$$

Subtracting $\|B\|_F^2$ from both sides, completes the proof. □

The combination of the three Facts, provides the following results.

Theorem 1. *Given an input matrix $A \in \mathbb{R}^{n \times d}$, α-FD with parameter ℓ returns a sketch $B \in \mathbb{R}^{\ell \times d}$ that satisfies for all $k > \alpha\ell$*

$$0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \|A - A_k\|_F^2/(\alpha\ell - k)$$

and projection of A onto B_k, the top k rows of B satisfies

$$\|A - \pi_{B_k}(A)\|_F^2 \leq \frac{\alpha\ell}{\alpha\ell - k}\|A - A_k\|_F^2.$$

Setting $\ell = (k+1/\varepsilon)/\alpha$ yields $0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \varepsilon\|A - A_k\|_F^2$ and setting $\ell = (k + k/\varepsilon)/\alpha$ yields $\|A - \pi_{B_k}(A)\|_F^2 \leq (1+\varepsilon)\|A - A_k\|_F^2$.

2.2 SpaceSaving Directions

SpaceSaving Directions (abbreviated SSD) uses Algorithm 2.3 for ReduceRank. Like the SS algorithm for frequent items, it assigns the counts for the second smallest counter (in this case squared singular value $\sigma_{\ell-1}^2$) to the direction of the smallest. Unlike the SS algorithm, we do not use $\sigma_{\ell-1}^2$ as the squared norm along each direction orthogonal to B, as that gives a consistent over-estimate.

Algorithm 2.3 ReduceRank-SS(S)

```
$\delta_i \leftarrow \sigma_{\ell-1}^2$
return $\mathsf{diag}(\sigma_1, \ldots, \sigma_{\ell-2}, 0, \sqrt{\sigma_\ell^2 + \delta_i})$.
```

Then to understand the error bounds for ReduceRank-SS, we will consider an arbitrary unit vector x. We can decompose $x = \sum_{j=1}^d \beta_j v_j$ where $\beta_j^2 = \langle x, v_j \rangle^2 > 0$ and $\sum_{j=1}^d \beta_j^2 = 1$. For notational convenience, without loss of generality, we assume that $\beta_j = 0$ for $j > \ell$. Thus $v_{\ell-1}$ represents the entire component of x in the null space of B (or $B_{[i]}$ after processing row i).

To analyze this algorithm, at iteration $i \geq \ell$, we consider a $d \times d$ matrix $\bar{B}_{[i]}$ that has the following properties: $\|B_{[i]}v_j\|^2 = \|\bar{B}_{[i]}v_j\|^2$ for $j < \ell - 1$ and $j = \ell$, and $\|\bar{B}_{[i]}v_j\|^2 = \delta_i$ for $j = \ell - 1$ and $j > \ell$. This matrix provides the constant but bounded overcount similar to the SS sketch. Also let $A_{[i]} = [a_1; a_2; \ldots; a_i]$.

Lemma 3. *For any unit vector x we have $0 \leq \|\bar{B}_{[i]}x\|^2 - \|A_{[i]}x\|^2 \leq 2\delta_i$*

Proof. We prove the first inequality by induction on i. It holds for $i = \ell - 1$, since $B_{[\ell-1]} = A_{[\ell-1]}$, and $\|\bar{B}_{[i]}x\|^2 \geq \|B_{[i]}x\|^2$. We now consider the inductive step at i. Before the reduce-rank call, the property holds, since adding row a_i to both $A_{[i]}$ (from $A_{[i-1]}$) and $C_{[i]}$ (from $B_{[i-1]}$) increases both squared norms equally (by $\langle a_i, x \rangle^2$) and the left rotation by U^T also does not change norms on the right. On the reduce-rank, norms only change in directions v_ℓ and $v_{\ell-1}$. Direction v_ℓ increases by δ_i, and in $\bar{B}_{[i]}$ the directions $v_{\ell-1}$ also does not change, since it is set back to δ_i, which it was before the reduce-rank.

We prove the second inequality also by induction, where it also trivially holds for the base case $i = \ell - 1$. Now we consider the inductive step, given it holds for $i-1$. First obverse that $\delta_i \geq \delta_{i-1}$ since δ_i is at least the $(\ell-1)$st squared singular value of $B_{[i-1]}$, which is at least δ_{i-1}. Thus, the property holds up to the reduce rank step, since again, adding row a_i and left-rotating does not affect the difference in norms. After the reduce rank, we again only need to consider the two directions changed $v_{\ell-1}$ and v_ℓ. By definition

$$\|A_{[i]}v_{\ell-1}\|^2 + 2\delta_i \geq \delta_i = \|\bar{B}_{[i]}v_{\ell-1}\|^2,$$

so direction $v_{\ell-1}$ is satisfied. Then

$$\|\bar{B}_{[i]}v_\ell\|^2 = \|B_{[i]}v_\ell\|^2 = \delta_i + \|C_{[i]}v_\ell\|^2 \leq 2\delta_i$$

and $0 \leq \|A_{[i]}v_\ell\|^2 \leq \|\bar{B}_{[i]}v_\ell\|^2$. Hence $\|\bar{B}_{[i]}v_\ell\|^2 - \|A_{[i]}v_\ell\|^2 \leq 2\delta_i - 0$, satisfying the property for direction v_ℓ, and completing the proof. □

Now we would like to prove the three Facts needed for relative error bounds for $B = B_{[n]}$. But this does not hold since $\|B\|_F^2 = \|A\|_F^2$ (an otherwise nice property), and $\|\bar{B}\|_F^2 \gg \|A\|_F^2$. Instead, we first consider yet another matrix $\hat{B}$ defined as follows with respect to B. B and $\hat{B}$ have the same right singular values V. Let $\delta = \delta_n$, and for each singular value σ_j of B, adjust the corresponding singular values of $\hat{B}$ to be $\hat{\sigma}_j = \max\{0, \sqrt{\sigma_j^2 - 2\delta}\}$.

Lemma 4. *For any unit vector x we have $0 \leq \|Ax\|^2 - \|\hat{B}x\|^2 \leq 2\delta$ and $\|A\|_F^2 - \|\hat{B}\|_F^2 \geq \delta(\ell-1)$.*

Proof. Directions v_j for $j > \ell-1$, the squared singular values are shrunk by at least δ. The squared singular value is already 0 for direction $v_{\ell-1}$. And the singular value for direction v_ℓ is shrunk by δ to be exactly 0. Since before shrinking $\|B\|_F^2 = \|A\|_F^2$, the second expression in the lemma holds.

The first expression follows by Lemma 3 since $\bar{B}$ only increases the squared singular values in directions v_j for $j = \ell-1$ and $j > \ell$ by δ, which are 0 in $\hat{B}$. And other directions v_j are the same for $\bar{B}$ and B and are at most 2δ larger than in A. □

Thus $\hat{B}$ satisfies the three Facts. We can now state the following property about B directly, setting $\alpha = (1/2)$, adjusting ℓ to $\ell-1$, then adding back the at most $2\delta = \Delta \leq \|A - A_k\|_F^2/(\alpha\ell - \alpha - k)$ to each directional norm.

Theorem 2. *After obtaining a matrix B from SSD on a matrix A with parameter ℓ, the following properties hold:*

- $\|A\|_F^2 = \|B\|_F^2$.
- *for any unit vector x and for $k < \ell/2 - 1/2$, we have $|\|Ax\|^2 - \|Bx\|^2| \leq \|A - A_k\|_F^2/(\ell/2 - 1/2 - k)$.*
- *for $k < \ell/2 - 1$ we have $\|A - \pi_B^k(A)\|_F^2 \leq \|A - A_k\|_F^2(\ell-1)/(\ell-1-2k)$.*

Setting $\ell = 2k + 2/\varepsilon + 1$ yields $0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \varepsilon\|A - A_k\|_F^2$ and setting $\ell = 2k + 1 + 2k/\varepsilon$ yields $\|A - \pi_{B_k}(A)\|_F^2 \leq (1+\varepsilon)\|A - A_k\|_F^2$.

2.3 Compensative Frequent Directions

In original FD, the computed sketch B underestimates Frobenius norm of stream [22]. In Compensative FrequentDirections (abbreviated CFD), we keep track of the total mass $\Delta = \sum_{i=1}^{n} \delta_i$ subtracted from squared singular values (this requires only an extra counter). Then we slightly modify the FD algorithm. In the final step where $B = S'V^T$, we modify S' to $\hat{S}$ by setting each singular value $\hat{\sigma}_j = \sqrt{\sigma'^2_j + \Delta}$, then we instead return $B = \hat{S}V^T$.

It now follows that for any $k \leq \ell$, including $k = 0$, that $\|A\|_F^2 = \|B\|_F^2$, that for any unit vector x we have $|\|Ax\|_F^2 - \|Bx\|_F^2| \leq \Delta \leq \|A - A_k\|_F^2/(\ell - k)$ for any $k < \ell$, and since V is unchanged that $\|A - \pi_B^k(A)\|_F^2 \leq \|A - A_k\|_F^2 \ell/(\ell - k)$. Also as in FD, setting $\ell = k + 1/\varepsilon$ yields $0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \varepsilon\|A - A_k\|_F^2$ and setting $\ell = kk/\varepsilon$ yields $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$.

3 Experiments

Herein we describe an extensive set of experiments on a wide variety of large input data sets. We focus on comparing the amount of space used by each type of sketch (measured in rows) against several error measures. We show improvements over FD (and in one instance iSVD) by our proposed algorithm 0.2-FD. All experiments are easily reproducible through a configuration we have prepared on the APT [34] system.

Each data set is an $n \times d$ matrix A, and the n rows are processed one-by-one in a stream. We also compare against some additional baseline streaming techniques, exemplifying the three alternatives to techniques based on FD. We perform a separate set of experiments to compare accuracy of our algorithms against each other and against exemplar algorithms in hashing, random projection and column sampling line of works.

Competing algorithms. For randomized algorithms, we average over 5 trials.

Random Projection: In this method [29, 37], sketch B is constructed by multiplying a projection matrix R into the input matrix A. R is a $\ell \times n$ matrix where each entry $R_{i,j} \in \{-1/\sqrt{\ell}, 1/\sqrt{\ell}\}$ uniformly. In fact matrix R randomly projects columns of A from dimension n to dimension ℓ. This method needs $O(\ell d)$ space and update time per row is $O(\ell d)$; each column of R can be generated as needed for each row, and then discarded.

Hashing: In this method [40], there are two hash functions $h : [n] \to [\ell]$ and $s : [n] \to \{-1, +1\}$ which map each row of A to a row of sketch B and to either $+1$ or -1, respectively. More precisely, B is initialized to be a $\ell \times d$ zero matrix, then when we process row a_i, we change B as $B_{h(i)} = B_{h(i)} + s(i)a_i$. Again the part of the hash function for each row of A can be generated as needed, and then discarded; space is still $O(\ell d)$, but the update time is now only $O(d + \log \ell)$.

Sampling: Column Sampling [16, 17, 36] (which translates to row sampling in our setting), samples ℓ rows a_i of matrix A with replacement proportional to $\|a_i\|^2$ and rescales each chosen rows to have norm $\|A\|_F/\sqrt{\ell}$. This method requires $O(d\ell)$ space and the amortized update time per row is $O(d + \ell)$ when

DataSet	# Datapoints	# Attributes	Rank	Numeric Rank
Random Noisy	10000	500	500	m=50 21.62, m=30 15.39, m=20 11.82, m=10 8.79
Adversarial	10000	500	500	1.69
Birds [2]	11788	312	312	12.50
Spam [1]	9324	499	499	3.25
connectUS	394792	512	512	4.83

Table 1. Datasets; numeric rank is defined $\|A\|_F^2/\|A\|_2^2$.

implemented as ℓ independent reservoir sampler. We do not consider other column sampling techniques with better leverage score-based error guarantees since they are quite complex [18] to operate in a stream.

FD and **iSVD** are described in detail in Section 2.

Datasets. We compare performance of our algorithms on both synthetic and real datasets; see a summary in Table 1. We also generate adversarial data to show that iSVD performs poorly under specific circumstances, this explains why there is no theoretical guarantee for them.

For Random Noisy, we generate the input $n \times d$ matrix A synthetically, mimicking the approach by Liberty [28]. We compose $A = SDU + F/\zeta$, where SDU is the m-dimensional signal (for $m < d$) and G/ζ is the (full) d-dimensional noise with ζ controlling the signal to noise ratio. Each entry $F_{i,j}$ of F is generated i.i.d. from a normal distribution $N(0,1)$, and we set $\zeta = 10$. For the signal, $S \in \mathbb{R}^{n \times m}$ again with each $S_{i,j} \sim N(0,1)$ i.i.d; D is diagonal with entries $D_{i,i} = 1-(i-1)/d$ linearly decreasing; and $U \in \mathbb{R}^{m \times d}$ is just a random rotation. We use $n = 10000$, $d = 500$, and consider $m \in \{10, 20, 30, 50\}$ (the default is $m = 50$).

In order to create Adversarial data, we constructed two orthogonal subspaces $S_1 = \mathbb{R}^{m_1}$ and $S_2 = \mathbb{R}^{m_2}$ ($m_1 = 400$ and $m_2 = 4$). Then we picked two separate sets of random vectors Y and Z and projected them on S_1 and S_2, respectively. Normalizing the projected vectors and concatenating them gives us the input matrix A. All vectors in $\pi_{S_1}(Y)$ appear in the stream before $\pi_{S_2}(Z)$; this represents a very sudden and orthogonal shift. As the theorems predict, FD and our proposed algorithms adjust to this change and properly compensate for it. However, since $m_1 \geq \ell$, then iSVD cannot adjust and always discards all new rows in S_2 since they always represent the smallest singular value of $B_{[i]}$.

We consider three real-world datasets. Birds [2] has each row represent an image of a bird, and each column a feature. PCA is a common first approach in analyzing this data, so we center the matrix. Spam [1] has each row represent a spam message, and each column some feature; it has dramatic and abrupt feature drift over the stream, but not as much as Adversarial. ConnectUS is from University of Florida Sparse Matrix collection [9], representing a recommendation system. Each column is a user, and each row is a webpage, tagged 1 if favorable, 0 otherwise. It contains 171 users that share no webpages preferences with any other users.

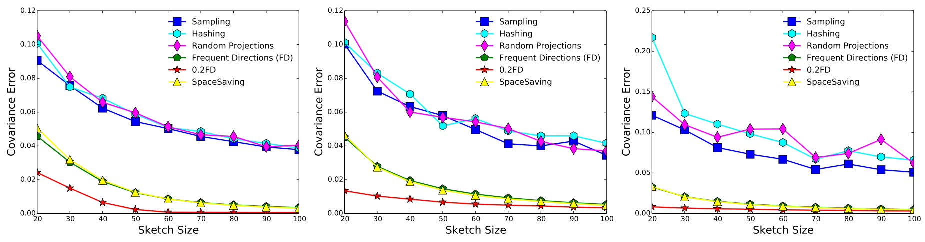

Fig. 1. Covariance Error: Random Noisy(50) (left), Birds (middle), and Spam (right).

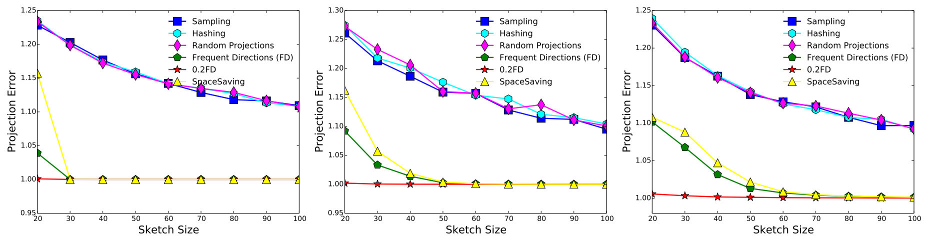

Fig. 2. Projection Error: Random Noisy(50) (left), Birds (middle), and Spam (right).

Approximation error vs. sketch size. We measure error for all algorithms as we change the parameter ℓ (Sketch Size) determining the number of rows in matrix B. We measure covariance error as err $= \|A^T A - B^T B\|_2 / \|A\|_F^2$ (Covariance Error); this indicates for instance for FD, that err should be at most $1/\ell$, but could be dramatically less if $\|A - A_k\|_F^2$ is much less than $\|A\|_F^2$ for some not so large k. We also consider proj-err $= \|A - \pi_{B_k}(A)\|_F^2 / \|A - A_k\|_F^2$, always using $k = 10$ (Projection Error); for FD we should have proj-err $\leq \ell/(\ell - 10)$, and ≥ 1 in general.

We see in Figure 1 for Covariance Error and Figure 2 for Projection Error that all baseline algorithms Sampling, Hashing, and Random Projections perform *much* worse than FD and the variants we consider. Each of Sampling, Hashing, and Random Projections perform about the same. Moreover, FD typically is out-performed by iSVD and our best proposed method 0.2-FD. Thus, we now focus only on FD, iSVD, and the new proposed methods which operate in a smaller error regime. For simplicity now we only examine Covariance Error, Projection Error acts similarly.

Next we consider Parametrized FD; we denote each variant as α-FD in Figure 3. We explore the effect of the parameter α, and run variants with $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$, comparing against FD ($\alpha = 1$) and iSVD ($\alpha = 0$). Note that the guaranteed error gets worse for smaller α, so performance being equal, it is preferable to have larger α. Yet, we observe empirically that FD is consistently the worst algorithm, and iSVD is fairly consistently the best, and as α decreases, the observed error improves. The difference can be quite dramatic; for instance in the Spam dataset, for $\ell = 20$, FD has err $= 0.032$ while iSVD and 0.2-FD have err $= 0.008$. Yet, as ℓ approaches 100, all algorithms seems to be approaching the same small error. We also explore the effect on α-FD in Figure 4 on Random Noisy data by varying $m \in \{10, 20, 30\}$, and $m = 50$ in Figure 3. We observe that all algorithms get smaller error for smaller m (there are fewer "directions"

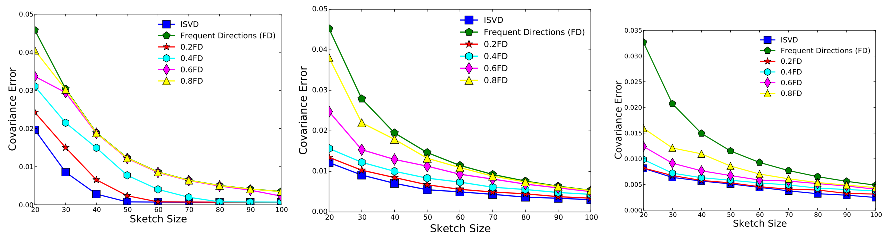

Fig. 3. Parametrized FD on Random Noisy(50) (left), Birds (middle), and Spam (right).

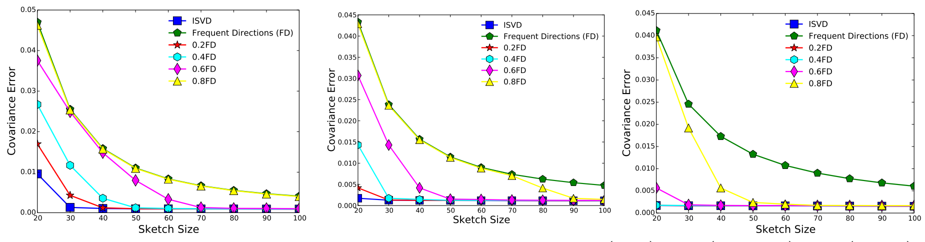

Fig. 4. Parametrized FD on Random Noisy for $m = 30$ (left), 20 (middle), 10 (right).

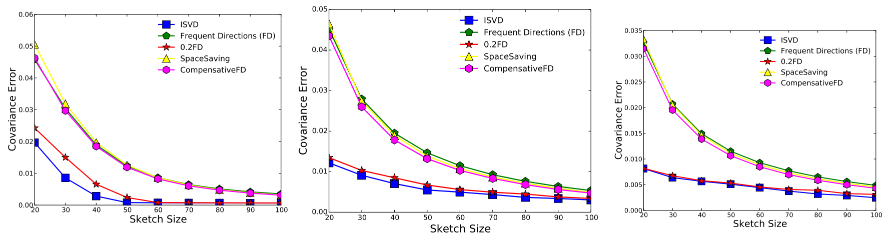

Fig. 5. SpaceSaving algos on Random Noisy(50) (left), Birds (middle), and Spam (right).

to approximate), but that each α-FD variant reaches 0.005 err before $\ell = 100$, sooner for smaller α; eventually "snapping" to a smaller 0.002 err level.

In Figure 5, we compare iSVD, FD, and 0.2-FD with the other variants based on the SS streaming algorithm: CFD and SSD. We see that these typically perform slightly better than FD, but not nearly as good as 0.2-FD and iSVD. Perhaps it is surprising that although SpaceSavings variants empirically improve upon MG variants for frequent items, 0.2-FD (based on MG) can largely outperform the all SS variants on matrix sketching.

Finally, we show that iSVD is not always better in practice. Using the Adversarial construction in Figure 6, we see that iSVD can perform much worse than the other techniques. Although at $\ell = 20$, iSVD and FD roughly perform the same (with about err $= 0.09$), iSVD does not improve much as ℓ increases, obtaining only err $= 0.08$ for $\ell = 100$. On the other hand, FD (as well as CFD and SSD) decrease markedly and consistently to err $= 0.02$ for $\ell = 100$. Moreover, all version of α-FD obtain roughly err$=0.005$ already for $\ell = 20$. The large-norm directions are the first 4 singular vectors (from the second part of the stream) and once these directions are recognized as having the largest singular vectors, they are no longer decremented in any Parametrized FD algorithm.

To conclude we demonstrate the scalability of these approaches on a much larger real data set ConnectUS. Figure 7 shows variants of Parameterized FD,

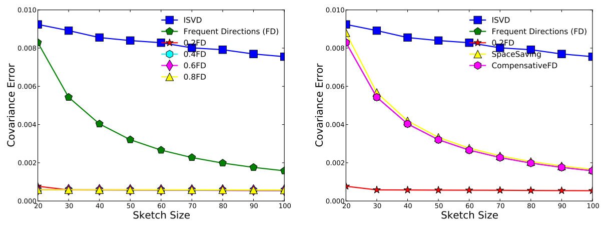

Fig. 6. Demonstrating dangers of iSVD on Adversarial data.

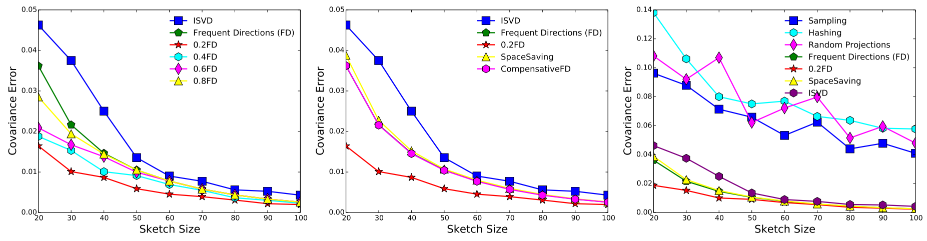

Fig. 7. Parameterized FD (left), SpaceSaving-based (middle), and all leading (right) algorithms on ConnectUS dataset.

those compared against SpaceSaving variants, and all leading algorithms on this dataset. As the derived bounds on covariance error based on sketch size do not all depend on n, the number of rows in A, it is not surprising that the performance of most algorithms is unchanged. There are just a couple differences to point out. First, no algorithm converges as close to 0 error as with the other smaller data sets; this is likely because with the much larger size, there is some variation that can not be captured even $\ell = 100$ rows of a sketch. Second, iSVD performs noticeably worse than the other FD-based algorithms (although still significantly better than the leading randomized algorithms). This likely has to do with the sparsity of ConnectUS combined with a data drift. After building up a sketch on the first part of the matrix, sparse rows are observed orthogonal to existing directions. The orthogonality, the same difficult property as in Adversarial, likely occurs here because the new rows have a small number of non-zero entrees, and all rows in the sketch have zeros in these locations; these correspond to the webpages marked by one of the unconnected users.

Reproducibility. All experiments are conducted on a Linux Ubuntu 12.04 machine with 16 cores of Intel(R) Xeon(R) CPU(2.10GHz) and 48GB of RAM. We provide public access to all of our results using a testbed facility *APT* [34]. APT is a platform where researchers can perform experiments and keep them public for verification and validation of the results. We provide our code, datasets, and experimental results in our APT profile with detailed description on how to reproduce, available at: `http://aptlab.net/p/MatrixApx/FrequentDirection`.

References

1. Concept drift in machine learning and knowledge discovery group. http://mlkd.csd.auth.gr/concept_drift.html.

2. vision.caltech. http://www.vision.caltech.edu/visipedia/CUB-200-2011.html.
3. Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In *STOC*, 2001.
4. Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *PODS*, 2012.
5. Arvind Arasu, Shivnath Babu, and Jennifer Widom. An abstract semantics and concrete language for continuous queries over streams and relations. 2002.
6. Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. *Lecture Notes in Computer Science*, pages 3–14, 2001.
7. Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction. In *FOCS*, 2011.
8. Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV*, 2002.
9. Steven Buss. Connectus data set – Florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices/Buss/connectus.html.
10. Jianjun Chen, David J DeWitt, Feng Tian, and Yuan Wang. Niagaracq: A scalable continuous query system for internet databases. *ACM SIGMOD Record*, 29(2):379–390, 2000.
11. Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *STOC*, 2009.
12. Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. In *VLDB*, 2008.
13. Corinna Cortes, Kathleen Fisher, Daryl Pregibon, and Anne Rogers. Hancock: a language for extracting signatures from data streams. In *KDD*, 2000.
14. Erik D Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *ESA*, 2002.
15. Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *APPROX*, 2006.
16. Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *SODA*, 2003.
17. Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36:158–183, 2006.
18. Petros Drineas, Malik Magdon-Ismail, Michael Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3441–3472, 2012.
19. Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.
20. Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *FOCS*, 1998.
21. Mina Ghashami, Edo Liberty, and Jeff M. Phillips. Frequent directions : Simple and deterministic matrix sketchings. personal communication, 2014.
22. Mina Ghashami and Jeff M. Phillips. Relative errors for deterministic low-rank matrix approximation. In *SODA*, 2014.
23. Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Quicksand: Quick summary and analysis of network data. Technical report, DIMACS 2001-43, 2001.
24. Gene H Golub and Charles F Van Loan. *Matrix computations.* JHUP, 2012.
25. Peter Hall, David Marshall, and Ralph Martin. Incremental eigenanalysis for classification. In *British Machine Vision Conference*, 1998.

26. Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM ToDS*, 28:51–55, 2003.
27. A. Levey and Michael Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE ToIP*, 9:1371–1374, 2000.
28. Edo Liberty. Simple and deterministic matrix sketching. In *KDD*, 2013.
29. Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *PNAS*, 104(51):20167–20172, 2007.
30. Michael W. Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *PNAS*, 106:697–702, 2009.
31. Ahmed Metwally, Divyakant Agrawal, and Amr El. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM ToDS*, 31:1095–1133, 2006.
32. J. Misra and D. Gries. Finding repeated elements. *Sc. Comp. Prog.*, 2:143–152, 1982.
33. Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS*, 1998.
34. Rob Ricci. Apt (adaptable profile-driven testbed). http://www.flux.utah.edu/project/apt, 2014.
35. David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 77:125–141, 2008.
36. Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM*, 54(4):21, 2007.
37. Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, 2006.
38. Mark Sullivan and Andrew Heybey. A system for managing large databases of network traffic. In *Proceedings of USENIX*, 1998.
39. Emulab testbed. http://www.flux.utah.edu/project/emulab.
40. Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.
41. Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, 2002.

A Appendix

Consider any algorithm that takes an input matrix $A \in \mathbb{R}^{n\times d}$ and outputs a matrix $B \in \mathbb{R}^{\ell\times d}$ which follows three facts below (for some parameter $\alpha \in (0,1]$ and some value Δ):

– Fact 1: For any unit vector x we have $\|Ax\|^2 - \|Bx\|^2 \geq 0$.
– Fact 2: For any unit vector x we have $\|Ax\|^2 - \|Bx\|^2 \leq \Delta$.
– Fact 3: $\|A\|_F^2 - \|B\|_F^2 \geq \alpha\Delta\ell$.

Lemma 5. *In any such algorithm, for any unit vector x:*

$$0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \|A - A_k\|_F^2/(\alpha\ell - k)$$

Proof. In the following, y_i correspond to the singular vectors of A ordered with respect to a decreasing corresponding singular value order.

$$\begin{aligned}
\alpha\Delta\ell &\leq \|A\|_F^2 - \|B\|_F^2 && \text{via Fact 3}\\
&= \sum_{i=1}^{k}\|Ay_i\|^2 + \sum_{i=k+1}^{d}\|Ay_i\|^2 - \|B\|_F^2 && \|A\|_F^2 = \sum_{i=1}^{d}\|Ay_i\|^2\\
&= \sum_{i=1}^{k}\|Ay_i\|^2 + \|A - A_k\|_F^2 - \|B\|_F^2 &&\\
&\leq \|A - A_k\|_F^2 + \sum_{i=1}^{k}\left(\|Ay_i\|^2 - \|By_i\|^2\right) && \sum_{i=1}^{k}\|By_i\|^2 < \|B\|_F^2\\
&\leq \|A - A_k\|_F^2 + k\Delta. && \text{via Fact 2}
\end{aligned}$$

Solving $\alpha\Delta\ell \leq \|A - A_k\|_F^2 + k\Delta$ for Δ to obtain $\Delta \leq \|A - A_k\|_F^2/(\alpha\ell - k)$, which combined with Fact 1 and Fact 2 proves the lemma. □

Lemma 6. *Any such algorithm described above, satisfies the following error bound*

$$\|A - \pi_{B_k}(A)\| \leq \alpha\ell/(\alpha\ell - k)\|A - A_k\|_F^2$$

Where $\pi_{B_k}(\cdot)$ represents the projection operator onto B_k, the top k singular vectors of B.

Proof. Here, y_i correspond to the singular vectors of A as above and v_i to the singular vectors of B in a similar fashion.

$$\begin{aligned}
\|A - \pi_{B_k}(A)\|_F^2 &= \|A\|_F^2 - \|\pi_{B_k}(A)\|_F^2 = \|A\|_F^2 - \sum_{i=1}^{k} \|Av_i\|^2 && \text{Pythagorean theorem} \\
&\le \|A\|_F^2 - \sum_{i=1}^{k} \|Bv_i\|^2 && \text{via Fact 1} \\
&\le \|A\|_F^2 - \sum_{i=1}^{k} \|By_i\|^2 && \text{since } \sum_{i=1}^{j} \|Bv_i\|^2 \ge \sum_{i=1}^{j} \|By_i\|^2 \\
&\le \|A\|_F^2 - \sum_{i=1}^{k} (\|Ay_i\|^2 - \mathit{\Delta}) && \text{via Fact 2} \\
&= \|A\|_F^2 - \|A_k\|_F^2 + k\mathit{\Delta} \\
&\le \|A - A_k\|_F^2 + \frac{k}{\alpha\ell - k}\|A - A_k\|_F^2 && \text{by } \mathit{\Delta} \le \|A - A_k\|_F^2/(\alpha\ell - k) \\
&= \frac{\alpha\ell}{\alpha\ell - k}\|A - A_k\|_F^2.
\end{aligned}$$

This completes the proof of lemma. □