

Improved Practical Matrix Sketching with Guarantees

Amey Desai

Mina Ghashami

Jeff M. Phillips *

January 27, 2015

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, the many forms of error bounds make these approaches hard to compare in theory and in practice. This paper attempts to categorize and compare most known methods under row-wise streaming updates with provable guarantees, and then to tweak some of these methods to gain practical improvements while retaining guarantees.

For instance, we observe that a simple heuristic iSVD, with no guarantees, tends to outperform all known approaches in terms of size/error trade-off. We modify the best performing method with guarantees FREQUENTDIRECTIONS under the size/error trade-off to match the performance of iSVD and retain its guarantees. We also demonstrate some adversarial datasets where iSVD performs quite poorly. In comparing techniques in the time/error trade-off, techniques based on hashing or sampling tend to perform better. In this setting we modify the most studied sampling regime to retain error guarantee but obtain dramatic improvements in the time/error trade-off.

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 [7, 30, 38, 50, 55] 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 is 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 [13], financial tickers [18, 66], on-line auctions [11], network traffic [42, 60], and telecom call records [24].

In recent years, extensive work has taken place to improve theoretical bounds in the size of B . Random projections [7, 59] and hashing [19, 64] approximate A in B as a random linear combination of rows and/or columns of A . Column sampling methods [14, 29, 30, 32, 35, 51, 58] 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 [19, 41, 65] for extensive discussion of various models and error bounds.

*Thanks to support by NSF CCF-1115677, CCF-1350888, IIS-1251019, and ACI-1443046.

Recently Liberty [50] introduced a new technique FREQUENTDIRECTIONS (abbreviated FD) which is deterministic, achieves the best bounds on the covariance $\|A^T A - B^T B\|_2$ error, the direct error $\|A - B\|_F^2$ [41] (using B as a projection), and moreover, it seems to greatly outperform the projection, hashing, and column sampling techniques in practice. Hence, this problem has seen immense progress in the last decade with a wide variety of algorithmic improvements in a variety of models [7, 19, 20, 25, 29, 32, 32, 35, 41, 50, 56, 59]; which we review thoroughly in Section 2 and assess comprehensively empirically in Section 5.

In addition, there is a family of heuristic techniques [16, 43, 44, 48, 57] (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 5) on many real and synthetic data sets that iSVD noticeably outperforms FD, yet there are adversarial examples where it fails dramatically. Thus 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 $[a_1; a_2; \dots, a_n]$ where each a_i is a row of length d . Alternatively a matrix V can be written as a set of columns $[v_1, v_2, \dots, v_d]$. We assume $d \ll n$. We will consider streaming algorithms where each element of the stream is a row a_i of A . Some of the algorithms also work in a more general setting (e.g., allowing deletions or distributed streams).

The squared Frobenius norm of a matrix A is defined $\|A\|_F^2 = \sum_{i=1}^n \|a_i\|^2$ 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 $\|A^T A - B^T B\|_2 = \max_{x: \|x\|=1} |\|Ax\|^2 - \|Bx\|^2|$.

Given a matrix A and a low-rank matrix X let $\pi_X(A) = AX^\dagger X$ 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^\dagger indicates taking the Moore-Penrose pseudoinverse of X . The singular value decomposition of A , written $\text{svd}(A)$, 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 $[v_1, v_2, \dots, v_d]$ are the right singular vectors, describing directions of most covariance in $A^T A$. S is $n \times d$ and is all 0s except for the diagonal entries $\{\sigma_1, \sigma_2, \dots, \sigma_r\}$, 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 .

Error measures. We consider two classes of error measures between input matrix A and its $\ell \times d$ sketch B . The *covariance error* ensures for any unit vector $x \in \mathbb{R}^d$ such that $\|x\| = 1$, that $\|Ax\|^2 - \|Bx\|^2$ is small. This can be mapped to the covariance of $A^T A$ since $\max_{\|x\|=1} \|\|Ax\|^2 - \|Bx\|^2\| = \|A^T A - B^T B\|_2$. To normalize *covariance error* we set

$$\text{cov-err}(A, B) = \|A^T A - B^T B\|_2 / \|A\|_F^2$$

which is always greater than 0 and will typically be less than 0.15.¹

The *projection error* describes the error in the subspace captured by B without focusing on its scale. Here we compare the best rank k subspace described by B (as B_k) compared against the same for A (as A_k). We measure the error by comparing the tail (what remains after projecting A onto this subspace) as $\|A - \pi_{B_k}(A)\|_F^2$. Specifically we normalize *projection error* so that it is comparable across datasets as

$$\text{proj-err}(A, B) = \|A - \pi_{B_k}(A)\|_F^2 / \|A - A_k\|_F^2.$$

¹The normalization term $\|A\|_F^2$ is invariant to the desired rank k and the unit vector x . Some methods have bounds on $\|Ax\|^2 - \|Bx\|^2$ that are relative to $\|A - A_k\|^2$ or $\|Ax\|^2$; but as these introduce an extra parameter they are harder to measure empirically.

Note the denominator is equivalent to $\|A - \pi_{A_k}(A)\|^2$ so this ensures the projection error is at least 1; typically it will be less than 1.5. We set $k = 10$ as a default in all experiments.²

1.2 Frequency Approximation, Intuition, and Results

There is a strong link between matrix sketching and the frequent items (similar to heavy-hitters) problem commonly studied in the streaming literature. That is, given a stream $S = \langle s_1, s_2, \dots, s_n \rangle$ of items $s_i \in [u] = \{1, 2, \dots, u\}$, represent $f_j = |\{s_i \in S \mid s_i = j\}|$; the frequency of each item $j \in [u]$. The goal is to construct an estimate \hat{f}_j (for *all* $j \in [u]$) so that $|f_j - \hat{f}_j| \leq \varepsilon n$. There are many distinct solutions to this problem, and many map to families of matrix sketching paradigms.

The Misra-Gries [53] (MG) frequent items sketch inspired the FD matrix sketching approach, and the extensions discussed herein. The MG sketch uses $O(1/\varepsilon)$ space to keep $\ell - 1 = 1/\varepsilon$ 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. There are other variants of this algorithm [26, 46, 52] that have slightly different properties [8, 22] that we will describe in Section 3.1 to inspire variants of FD.

A folklore approach towards the frequent items problem is to sample $\ell = O((1/\varepsilon^2) \log 1/\delta)$ items from the stream. Then \hat{f}_j is the count of the sample scaled by n/ℓ . This achieves our guarantees with probability at least $1 - \delta$ [49, 61], and will correspond with sampling algorithms for sketching which sample rows of the matrix.

Another frequent items sketch is called the *count-sketch* [17]. It maintains and averages the results of $O(\log 1/\delta)$ of the following sketch. Each $j \in [u]$ is randomly hashed $h : [u] \rightarrow [\ell]$ to a cell of a table of size $\ell = O(1/\varepsilon^2)$; it is added or subtracted from the row based on another random hash $s : [u] \rightarrow \{-1, +1\}$. Estimating \hat{f}_j is the count at cell $h(j)$ times $s(j)$. This again achieves the desired bounds with probability at least $1 - \delta$. This work inspires the *hashing* approaches to matrix sketching. If each element j had its effect spread out over all $O(\log 1/\delta)$ instances of the base sketch, it would then map to the *random projection* approaches to sketching, although this comparison is a bit more tenuous.

Finally, there is another popular frequent items sketch called the count-min sketch [23]. It is similar to the count-sketch but without the sign hash, and as far as we know has not been directly adapted to a matrix sketching algorithm.

1.3 Contributions

We survey and categorize the main approaches to matrix sketching in a row-wise update stream. We consider three categories: sampling, projection/hashing, and iterative and show how all approaches fit simply into one of these three categories. We also provide an extensive set of experiments to compare these algorithms along size, error (projection and covariance), and runtime on real and synthetic data.

To make this study easily and readily *reproducible*, we implement all experiments on a new extension of Emulab [5] called Adaptable Profile-Driven Testbed or APT [3]. 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.

We also consider new variants of these approaches which maintain error guarantees but significantly improving performance. We introduce several new variants of FD, one of which α -FD matches or ex-

²There are variations in bounds on this sort of error. Some measure spectral (e.g. $\|\cdot\|_2$) norm. Others provide a weaker error bound on $\|A - [\pi_B(A)]_k\|_F^2$, where the “best rank k approximation,” denoted by $[\cdot]_k$, is taken after the projection. This is less useful since then for a very large rank B (might be rank 500) it is not clear which subspace best approximates A until this projection is performed. Additionally, some approaches create a set of (usually three) matrices (e.g. *CUR*) with product equal to $\pi_{B_k}(A)$, instead of just B . This is a stronger result, but it does not hold for many approaches, so we omit consideration.

ceeds the performance of a popular heuristic iSVD. Before this new variant, iSVD is a top performer in space/error trade-off, but has no guarantees, and as we demonstrate on some adversarial data sets, can fail spectacularly. We also show how to efficiently implement and analyze *without-replacement* row sampling for matrix sketching, and how this can empirically improve upon more traditional (and easier to analyze) with-replacement row sampling.

2 Matrix Sketching Algorithms

In this section we review the main algorithms for sketching matrices. We divide them into 3 main categories: (1) sampling algorithms, these select a subset of rows from A to use as the sketch B ; (2) projection algorithms, these project the n rows of A onto ℓ rows of B , sometimes using hashing; (3) incremental algorithms, these maintain B as a low-rank version of A updated as more rows are added.

There exist other forms of matrix approximation, for instance, using sparsification techniques [7, 12, 36], or allowing more general element-wise updates at the expense of larger sketch sizes [19, 20, 59]. We are interested in preserving the right singular vectors and other statistical properties on the rows of A .

2.1 Sampling Matrix Sketching

	ℓ	cov-err	ℓ	proj-err	runtime
Norm Sampling	d/ε^2	ε (†) [32]	k/ε^2	$1 + \varepsilon \frac{\ A\ _F^2}{\ A - A_k\ _F^2}$ [32]	$\text{nnz}(A) \cdot \ell$
Leverage Sampling	d/ε^2	ε (†)	$(k \log k)/\varepsilon^2$	$1 + \varepsilon$ [51]	$\text{svd}(A) + \text{nnz}(A) \cdot \ell$
Deterministic Leverage	ℓ	-	$(k/\eta\varepsilon)^{1/\eta}(\star)$	$1 + \varepsilon$ [56]	$\text{svd}(A) + \text{nnz}(A) \cdot \ell \log \ell$

Table 1: Theoretical Bounds for Sampling Algorithms. The proj-err bounds are based on a slightly weaker $\|A - \pi_B(A)\|_F^2$ numerator instead of $\|A - \pi_{B_k}(A)\|_F^2$ one where we first enforce B_k is rank k . (†) See Appendix C; the Leverage Sampling bound assumes a constant lower bound on leverage scores. (★) Maximum of this and $\{k, (k/\varepsilon)^{1/(1+\eta)}\}$ where leverage scores follow power-law with decay exponent $1 + \eta$.

Sampling algorithms assign a probability p_i for each row a_i and then selecting ℓ rows from A into B using this probability. In B , each row has its squared norm rescaled to w_i as a function of p_i and $\|a_i\|$. One can achieve additive error bound using importance sampling with $p_i = \|a_i\|^2 / \|A\|_F^2$ and $w_i = \|a_i\|^2 / (\ell p_i) = \|A\|_F^2 / \ell$, as analyzed by Drineas *et al.* [31] and [38]. These algorithms typically advocate sampling ℓ items independently (with replacement) using ℓ distinct reservoir samplers, taking $O(\ell)$ time per element. Another version [35] samples each row independently, and only retains ℓ rows in expectation. We discuss two improvements to this process in Section 3.2.

Much of the related literature describes selecting columns instead of rows (called the *column subset selection problem*) [15]. This is just a transpose of the data and has no real difference from what is described here. There are also techniques [35] that select both columns and rows, but are orthogonal to our goals.

This family of techniques has the advantage that the resulting sketch is *interpretable* in that each row of B corresponds to data point in A , not just a linear combination of them.

Leverage Sampling. An insightful adaptation changes the probability p_i using *leverage scores* [34] or *simplex volume* [27, 28]. These techniques take into account more of the structure of the problem than simply the rows norm, and can achieve stronger relative error bounds. But they also require an extra parameter k as part of the algorithm, and for the most part require much more work to generate these modified p_i scores. We use Leverage Sampling [35] as a representative; it samples rows according to leverage scores (described below). Simplex volume calculations [27, 28] were too involved to be practical. There are also recent techniques to improve on the theoretical runtime for leverage sampling [33] by approximating the desired

values p_i , but as the exact approaches do not demonstrate consistent tangible error improvements, we do not pursue this complicated theoretical runtime improvement.

To calculate leverage scores, we first calculate the svd of A (the task we hoped to avoid). Let U_k be the matrix of the top k left singular vectors, and let $U_k(i)$ represent the i th row of that matrix. Then the *leverage score* for row i is $s_i = \|U_k(i)\|^2$, the fraction of squared norm of a_i along subspace U_k . Then set p_i proportional to s_i (e.g. $p_i = s_i/k$. Note that $\sum_i s_i = k$).

Deterministic Leverage Scores. Another option is to deterministically select rows with the highest s_i values instead of at random. It can be implemented with a simple priority queue of size ℓ . This has been applied to using the leverage scores by Papailiopoulos *et al.* [56], which again first requires calculating the svd of A . We refer to this algorithm as Deterministic Leverage Sampling.

2.2 Projection Matrix Sketching

	ℓ	cov-err	ℓ	proj-err	runtime
Random Projection	d/ε^2 [59]	$\varepsilon/\rho(A)$	d/ε^2 [59]	$1 + \varepsilon$	$\text{nnz}(A) \cdot \ell$
Fast JLT	d/ε^2 [59]	$\varepsilon/\rho(A)$	d/ε^2 [59]	$1 + \varepsilon$	$nd \log d + (d/\varepsilon^2) \log n$ [9]
Hashing	d^2/ε^2 [20, 54]	$\varepsilon/\rho(A)$	d^2/ε^2 [20, 54]	$1 + \varepsilon$	$\text{nnz}(A) + n\text{poly}(d/\varepsilon)$
OSNAP	$d^{1+o(s/\varepsilon)}/\varepsilon^2$ [54]	$\varepsilon/\rho(A)$	$d^{1+o(s/\varepsilon)}/\varepsilon^2$ [54]	$1 + \varepsilon$	$\text{nnz}(A) \cdot s + n\text{poly}(d/\varepsilon)$

Table 2: Theoretical Bounds for Projection Algorithms (via an ℓ_2 subspace embedding; see Appendix C.2). Where ℓ is the number of rows maintained, and $\rho(A) = \frac{\|A\|_F^2}{\|A\|_2^2} \geq 1$ is the *numeric rank* of A .

These methods linearly project the n rows of A to ℓ rows of B . A survey by Woodruff [65] (especially Section 2.1) gives an excellent account of this area. In the simplest version, each row $a_i \in A$ would map to a row $b_j \in B$ with element $s_{j,i}$ (j th row and i th column) of a projection matrix S , and each $s_{j,i}$ is a Gaussian random variable with 0 mean and $\sqrt{n/\ell}$ standard deviation. That is, $B = SA$, where S is $\ell \times n$. This follows from the celebrated Johnson-Lindenstrauss lemma [45] as first shown by Sarlos [59] and strengthened by Clarkson and Woodruff [19]. Gaussian random variables $s_{j,i}$ can be replaced with (appropriately scaled) $\{-1, 0, +1\}$ or $\{-1, +1\}$ random variables [6]. We call the version with scaled $\{-1, +1\}$ random variables as Random Projection.

Fast JLT. Using a sparse projection matrix X would improve the runtime, but these lose guarantees if the input is also sparse (if the non-zero elements do not align). This is circumvented by rotating the space with a Hadamard matrix [9], which can be applied more efficiently using FFT tricks, despite being dense. More precisely, we use three matrices: P is $\ell \times n$ and has entries with iid 0 with probability $1 - q$ and a Gaussian random variable with variance ℓ/q with probability $q = \min\{1, \Theta((\log^2 n)/d)\}$. H is $n \times n$ and a random Hadamard (this requires n to be padded to a power of 2). D is diagonal with random $\{-1, +1\}$ in each diagonal element. And then the projection matrix is $S = PHD$, although algorithmically the matrices are applied implicitly. We refer to this algorithm as Fast JLT. Ultimately, the runtime is brought from $O(nd\ell)$ to $O(nd \log d + (d/\varepsilon^2) \log n)$. The second term in the runtime can be improved with more complicated constructions [10, 25] which we do not pursue here; we point the reader here [62] for a discussion of some of these extensions.

Sparse Random Projections. Clarkson and Woodruff [20] analyzed a very sparse projection matrix S , conceived of earlier [25, 64]; it has exactly 1 non-zero element per column. To generate S , for each column choose a random value between 1 and ℓ to be the non-zero, and then choose a -1 or $+1$ for that location.

Thus each rows can be processed in time proportional to its number of non-zeros; it is randomly added or subtracted from 1 row of B , as a count sketch [17] on rows instead of counts. We refer this as Hashing.

A slight modification by Nelson and Nguyen [54], called OSNAP, stacks s instances of the projection matrix S on top of each other. If HASHING used ℓ' rows, then OSNAP uses $\ell = s \cdot \ell'$ rows (we use $s = 4$).

2.3 Iterative Matrix Sketching

	ℓ	cov-err	ℓ	proj-err	runtime
Frequent Directions	$k + 1/\varepsilon$	$\varepsilon \frac{\ A - A_k\ _F^2}{\ A\ _F^2}$ [40]	k/ε	$1 + \varepsilon$ [41]	ndl
Iterative SVD	ℓ	-	ℓ	-	ndl^2

Table 3: Theoretical Bounds for Iterative Algorithms.

The main structure of these algorithms is presented in Algorithm 2.1, they maintain a sketch $B_{[i]}$ of $A_{[i]}$, the first i rows of A . The sketch of $B_{[i-1]}$ always uses at most $\ell - 1$ rows. On seeing the i th row of A , it is appended to $[B_{[i-1]}; a_i] \rightarrow B_{[i]}$, and if needed the sketch is reduced to use at most $\ell - 1$ rows again using some REDUCERANK procedure. Notationally we use σ_j as the j th singular value in S , and σ'_j as the j th 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**
 $[U, S, V] \leftarrow \text{svd}(B_{[i]})$
 $C_{[i]} = SV^T$ # Only needed for proof notation
 $S' \leftarrow \text{REDUCERANK}(S)$
 $B_{[i]} \leftarrow S'V^T$
return $B = B_{[n]}$

Iterative SVD. The simplest variant of this procedure is a heuristic rediscovered several times [16, 43, 44, 48, 57], with a few minor modifications, and we refer to as *iterative SVD* or iSVD. Here $\text{ReduceRank}(S, V)$ simply keeps $\sigma'_j = \sigma_j$ for $j < \ell$ and sets $\sigma'_\ell = 0$. This has no worst case guarantees (despite several claims).

Frequent Directions. Recently Liberty [50] proposed an algorithm called Frequent Directions (or FD), further analyzed by Ghashami and Phillips [41], and then together jointly with Woodruff [40]. The REDUCERANK step sets each $\sigma'_j = \sqrt{\sigma_j^2 - \delta_i}$ where $\delta_i = \sigma_\ell^2$.

Liberty also presented a faster variant FastFD, that instead sets $\delta_i = \sigma_{\ell/2}^2$ (the $(\ell/2)$ th squared singular value of $B_{[i]}$) and updates new singular values to $\sigma'_j = \max\{0, \sqrt{\sigma_j^2 - \delta_i}\}$, hence ensuring at most half of the rows are all zeros after each such step. This reduces the runtime from $O(nd\ell^2)$ to $O(nd\ell)$ at expense of a sketch sometimes only using half of its rows.

3 New Matrix Sketching Algorithms

Here we describe our new variants on FD that perform better in practice and are backed with error guarantees. In addition, we explain a couple of new matrix sketching techniques that makes subtle but tangible improvements to the other state-of-the-art algorithms mentioned above.

3.1 New Variants On FREQUENTDIRECTIONS

	ℓ	cov-err	ℓ	proj-err	runtime
Fast α -FD	$(k + 1/\varepsilon)/\alpha$	$\varepsilon \frac{\ A - A_k\ _F^2}{\ A\ _F^2}$	$k/(\varepsilon\alpha)$	$1 + \varepsilon$	ndl/α
SpaceSaving Directions	$k + 1/\varepsilon$	$\varepsilon \frac{\ A - A_k\ _F^2}{\ A\ _F^2}$	k/ε	$1 + \varepsilon$	ndl
Compensative FD	$k + 1/\varepsilon$	$\varepsilon \frac{\ A - A_k\ _F^2}{\ A\ _F^2}$	k/ε	$1 + \varepsilon$	ndl

Table 4: Theoretical Bounds for New Iterative Algorithms.

Since all our proposed algorithms on FREQUENTDIRECTIONS share the same structure, to avoid repeating the proof steps, we abstract out three properties that these algorithms follow and prove that *any* algorithm with these properties satisfy the desired error bounds. This slightly generalizes (allowing for $\alpha \neq 1$) a recent framework [40]. We prove these generalizations in Appendix A.

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 properties below (for some parameter $\alpha \in (0, 1]$ and some value $\Delta > 0$):

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

Lemma 3.1. *Any B satisfying the above three properties satisfies*

$$0 \leq \|A^T A - B^T B\|_2 \leq \frac{1}{\alpha\ell - k} \|A - A_k\|_F^2,$$

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

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

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 $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon) \|A - A_k\|_F^2$. FD maintains an $\ell \times d$ matrix B (i.e. using $O(\ell d)$ space), and it is shown [41] that there exists a value Δ that FD satisfies three above-mentioned properties with $\alpha = 1$.

Parameterized FD. Parameterized FD uses the following subroutine (Algorithm 3.1) 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 Properties hold. We use $\Delta = \sum_{i=1}^n \delta_i$.

Lemma 3.2. *For any unit vector x and any $\alpha \geq 0$: $0 \leq \|C_{[i]}x\|^2 - \|B_{[i]}x\|^2 \leq \delta_i$.*

Algorithm 3.1 REDUCERANK-PFD(S, α)

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

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

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

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 [41]) that

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

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

Lemma 3.3. For any $\alpha \in (0, 1]$, $\|A\|_F^2 - \|B\|_F^2 = \alpha\Delta\ell$, proving Property 3.

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

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

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 Properties, provides the following results.

Theorem 3.1. 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 $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon) \|A - A_k\|_F^2$.

Fast Parameterized FD. Fast Parameterized FD(or Fast α -FD) improves the runtime performance of parameterized FD in the same way Fast FD improves the performance of FD. More specifically, in REDUCERANK we set δ_i as the $(\ell - \ell\alpha/2)$ th squared singular value, i.e. $\delta_i = \sigma_t^2$ for $t = \ell - \ell\alpha/2$. Then we update the sketch by only changing the last $\alpha\ell$ singular values: we set $\sigma_j'^2 = \max(\sigma_j^2 - \delta_i, 0)$. This sets at least $\alpha\ell/2$ singular values to 0 once every $\alpha\ell/2$ steps. Thus the algorithm takes total time $O(nd + n/(\alpha\ell/2) \cdot d\ell^2) = O(nd\ell/\alpha)$.

It is easy to see that Fast α -FD inherits the same worst case bounds as α -FD on cov-err and proj-err, if we use twice as many rows. That is, setting $\ell = 2(k + 1/\varepsilon)/\alpha$ yields $\|A^T A - B^T B\|_2 \leq \varepsilon \|A - A_k\|_F^2$ and setting $\ell = 2(k + k/\varepsilon)/\alpha$ yields $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon) \|A - A_k\|_F^2$. In experiments we consider Fast 0.2-FD.

SpaceSaving Directions. Motivated by an empirical study [22] showing that the SpaceSaving algorithm [52] tends to outperform its analog Misra-Gries [53] in practice, we design an algorithm called SPACESAVING DIRECTIONS (abbreviated SSD) to try to extend these ideas to matrix sketching. It uses Algorithm 3.2 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 3.2 REDUCERANK-SS(S)

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

We can also show similar error bounds for SSD. It shows that a simple transformation of the output sketch $B \leftarrow \text{SSD}(A)$ satisfies the three Properties, although B itself does not. We defer these proofs to Appendix B, just stating the main bounds here.

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

Compensative Frequent Directions. Inspired by the isomorphic transformation [8] between the Misra-Gries [53] and the SpaceSaving sketch [52], which performed better in practice on the frequent items problem [22], we consider another variant of FD for matrix sketching. In the frequent items problem, this would return an identical result to SSD, but in the matrix setting it does not.

We call this approach COMPENSATIVE FREQUENT DIRECTIONS (abbreviated CFD). In original FD, the computed sketch B underestimates the Frobenius norm of stream [41], in CFD we try to compensate for this. Specifically, 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 $\hat{\sigma}_j = \sqrt{\sigma_j'^2 + \Delta}$, 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 $\|A - \pi_B^k(A)\|_F^2 \leq \|A - A_k\|_F^2 \cdot \ell/(\ell - k)$. 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 $\ell = k/\varepsilon$ yields $\|A - \pi_{B_k}(A)\|_F^2 \leq (1 + \varepsilon) \|A - A_k\|_F^2$.

3.2 New Without Replacement Sampling Algorithms

	ℓ	cov-err	ℓ	proj-err	runtime
Priority	d/ε^2	ε	ℓ	-	$\text{nnz}(A) \log \ell$
VarOpt	d/ε^2	ε	ℓ	-	$\text{nnz}(A) \log \ell$

Table 5: Theoretical Bounds for New Sampling Algorithms.

As mentioned above, most sampling algorithms use *sampling with replacement* (SwR) of rows. This is likely because, in contrast to *sampling without replacement* (SwoR), it is easy to analyze and for weighted samples conceptually easy to compute. SwoR for unweighted data can easily be done with variants of reservoir sampling [63]; however, variants for weighted data have been much less resolved until recently [21, 37].

Priority Sampling. A simple technique [37] for SwoR on weighted elements first assigns each element i a random number $u_i \in \text{Unif}(0, 1)$. This implies a priority $\rho_i = w_i/u_i$, based on its weight w_i (which for matrix rows $w_i = \|a_i\|_2^2$). We then simply retain the ℓ rows with largest priorities, using a priority queue of size ℓ . Thus each step takes $O(\log \ell)$ time, but on randomly ordered data would take only $O(1)$ time in expectation since elements with $\rho_i \leq \tau$, where τ is the ℓ th largest priority seen so far, are discarded.

Retained rows are given a squared norm $\hat{w}_i = \max(w_i, \tau)$. Rows with $w_i \geq \tau$ are always retained with original norm. Small weighted rows are kept proportional to their squared norms. The technique, Priority Sampling, is simple to implement, but requires a second pass on retained rows to assign final weights.

VarOpt Sampling. VarOpt (or Variance Optimal) sampling [21] is a modification of priority sampling that takes more care in selecting the threshold τ . In priority sampling, τ is generated so $\mathbf{E}[\sum_{a_i \in B} \hat{w}_i] = \|A\|_F^2$, but if τ is set more carefully, then we can achieve $\sum_{a_i \in B} \hat{w}_i = \|A\|_F^2$ deterministically. VarOpt selects each row with some probability $p_i = \min(1, w_i/\tau)$, with $\hat{w}_i = \max(w_i, \tau)$, and so exactly ℓ rows are selected.

The above implies that for a set L of ℓ rows maintained, there is a fixed threshold τ that creates the equality. We maintain this value τ as well as the t weights smaller than τ inductively in L . If we have seen at least $\ell + 1$ items in the stream, there must be at least one weight less than τ . On seeing a new item, we use the stored priorities $\rho_i = w_i/u_i$ for each item in L to either (a) discard the new item, or (b) keep it and drop another item from the reservoir. As the priorities increase, the threshold τ must always increase. It takes amortized constant time to discarding a new item or $O(\log \ell)$ time to keep the new item, and does not require a final pass on L . We refer to it as VarOpt.

A similar algorithm using priority sampling was considered in a distributed streaming setting [39], which provided a high probability bound on cov-err. A constant probability of failure bound for $\ell = O(d/\varepsilon^2)$ and $\text{cov-err} \leq \varepsilon$, follows with minor modification from Section C.1. It is an open question to bound the projection error for these algorithms, but we conjecture the bounds will match those of Norm Sampling.

4 Experimental Setup

We used an OpenSUSE 12.3 machine with 32 cores of Intel(R) Core(TM) i7-4770S CPU(3.10 GHz) and 32GB of RAM. Randomized algorithms were run five times; we report the median error value.

DataSet	# datapoints	# attributes	rank	numeric rank	nnz%	excess kurtosis
Birds	11789	312	312	12.50	100	1.72
Random Noisy	10000	500	500	14.93	100	0.95
CIFAR-10	60000	3072	3072	1.19	99.75	1.34
Connectus	394792	512	512	4.83	0.0055	17.60
Spam	9324	499	499	3.25	0.07	3.79
Adversarial	10000	500	500	1.69	100	5.80

Table 6: Dataset Statistics.

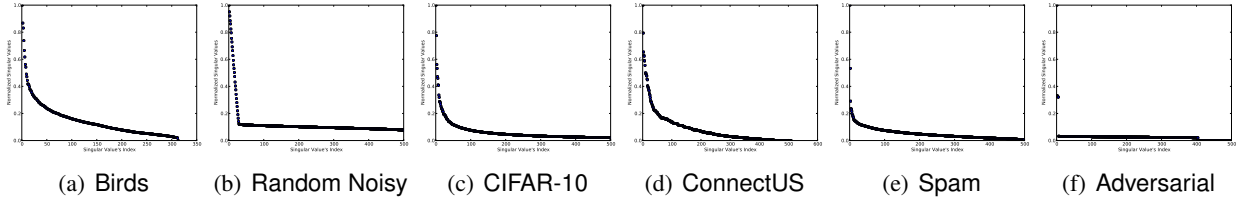


Figure 1: Singular values distribution for datasets in Table 6. The x -axis is singular value index, and the y -axis shows normalized singular values such that the highest singular value is one, i.e., each value divided by largest singular value of dataset

Datasets. We compare performance of the algorithms on both synthetic and real datasets. In addition, we generate adversarial data to show that iSVD performs poorly under specific circumstances, this explains why there is no theoretical guarantee for them. Each data set is an $n \times d$ matrix A , and the n rows are processed one-by-one in a stream.

Table 6 lists all datasets with information about their n , d , $\text{rank}(A)$, numeric rank $\|A\|_F^2 / \|A\|_2^2$, percentage of non-zeros (as nnz%, measuring sparsity), and excess kurtosis. We follow Fisher’s distribution with baseline kurtosis (from normal distribution) is 0; positive excess kurtosis reflects fatter tails and negative excess kurtosis represents thinner tails.

For Random Noisy, we generate the input $n \times d$ matrix A synthetically, mimicking the approach by Liberty [50]. We compose $A = SDU + F/\zeta$, where SDU is the m -dimensional signal (for $m < d$) and F/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 we generate 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 $m \in \{10, 20, 30, 50\}$ with $m = 30$ as default.

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 4 real-world datasets. ConnectUS is taken from the University of Florida Sparse Matrix collection [4]. ConnectUS represents 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. Birds [1] 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 [2] 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. CIFAR-10 is a standard computer vision benchmark dataset for deep learning [47].

The singular values distribution of the datasets is given in Figure 1. The x -axis is the singular value index, and the y -axis shows the normalized singular values, i.e. singular values divided by σ_1 , where σ_1 is the largest singular value of dataset. Birds, ConnectUS and Spam have consistent drop-offs in singular values. Random Noisy has initial sharp and consistent drops in singular values, and then a more gradual decrease. The drop-offs in CIFAR-10 and Adversarial are more dramatic.

We will focus most of our experiments on three data sets Birds (dense, tall, large numeric rank), Spam (sparse, not tall, negative kurtosis, high numeric rank), and Random Noisy (dense, tall, synthetic). However, for some distinctions between algorithms require considering much larger datasets; for these we use CIFAR-10 (dense, not as tall, small numeric rank) and ConnectUS (sparse, tall, medium numeric rank). Finally, Adversarial and, perhaps surprisingly ConnectUS are used to show that using iSVD (which has no guarantees) does not always perform well.

5 Experimental Evaluation

We divide our experimental evaluation into four sections: The first three sections contain comparisons within algorithms of each group (sampling, projection, and iterative), while the fourth compares accuracy and run time of exemplar algorithm in each group against each other.

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 $\text{err} = \|A^T A - B^T B\|_2 / \|A\|_F^2$ (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 consider $\text{proj-err} = \|A - \pi_{B_k}(A)\|_F^2 / \|A - A_k\|_F^2$, always using $k = 10$ (Projection Error); for FD we should have $\text{proj-err} \leq \ell/(\ell - 10)$, and ≥ 1 in general. We also measure run-time as sketch size varies.

Within each class, the algorithms are not dramatically different across sketch sizes. But across classes, they vary in other ways, and so in the global comparison, we will also show plots comparing runtime to cov-err or proj-err, which will help demonstrate and compare these trade-offs.

5.1 Sampling Algorithms

Figure 2 shows the covariance error, projection error, and runtime for the sampling algorithms as a function of sketch size, run on the Birds, Spam, and Random Noisy(30) datasets with sketch sizes from $\ell = 20$ to 100. We use parameter $k = 10$ for Leverage Sampling, the same k used to evaluate proj-err.

First note that Deterministic Leverage performs quite differently than all other algorithms. The error rates can be drastically different: smaller on Random Noisy proj-err and Birds proj-err, while higher on Spam proj-err and all cov-err plots. The proven guarantees are only for matrices with Zipfian leverage score sequences and proj-err, and so when this does not hold it can perform worse. But when the conditions are right it outperforms the randomized algorithms since it deterministically chooses the best rows.

Otherwise, there is very small difference between the error performance of all randomized algorithms, within random variation. The small difference is perhaps surprising since Leverage Sampling has a stronger error guarantee, achieving a relative proj-err bound instead of an additive error of Norm Sampling, Priority Sampling and VarOpt Sampling which only use the row norms. Moreover Leverage Sampling and Deterministic Leverage Sampling are significantly slower than the other approaches since they require first computing the SVD and leverage scores. We note that if $\|A - A_k\|_F^2 > c\|A\|_F^2$ for a large enough constant c , then for that choice of k , the tail is effectively fat, and thus not much is gained by the relative error bounds. Moreover, Leverage Sampling bounds are only stronger than Norm Sampling in a variant of proj-err where $[\pi_B(A)]_k$

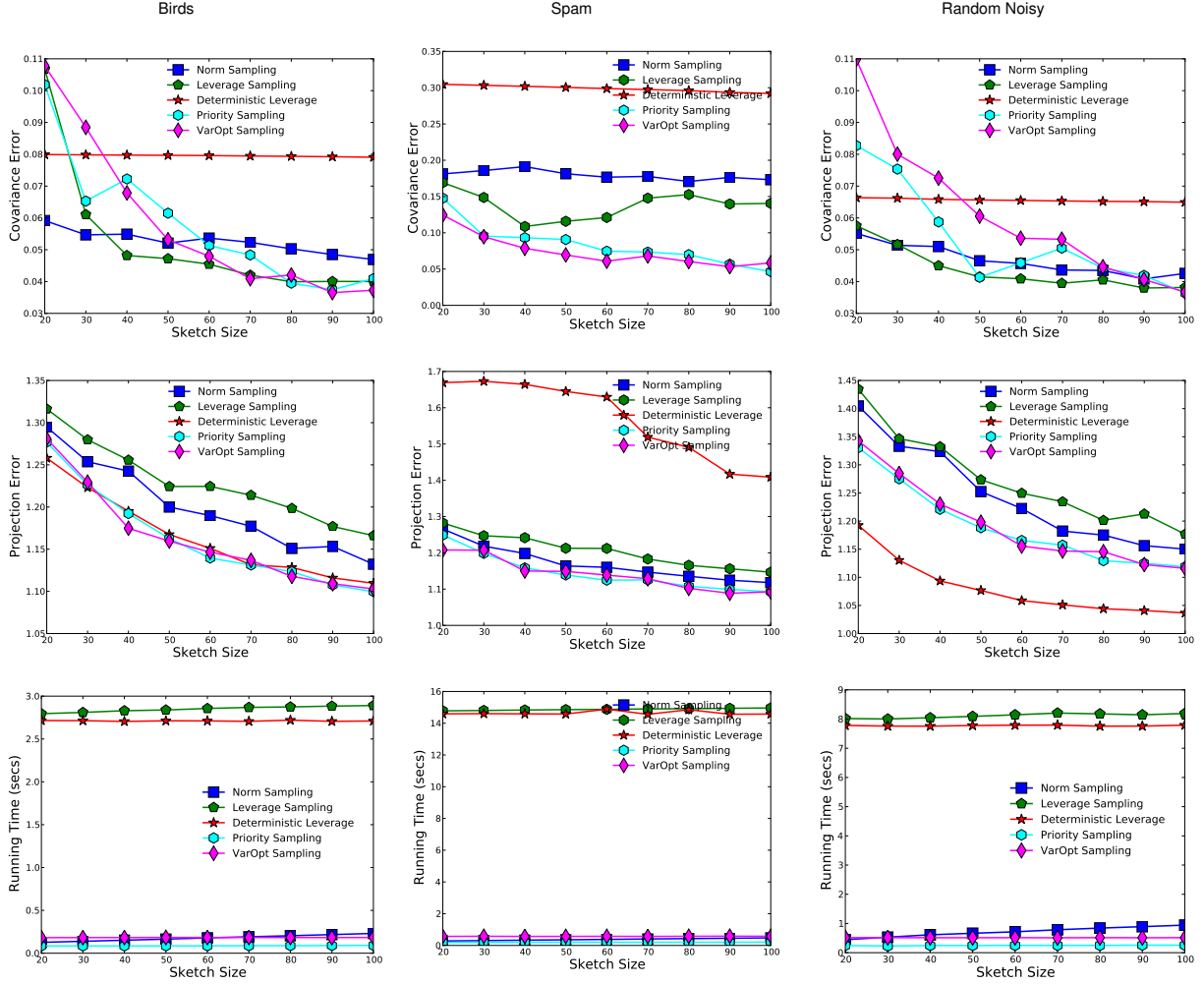


Figure 2: Sampling algorithms on Birds(left), Spam(middle), and Random Noisy(30)(right).

(with best rank k applied *after* projection) instead of $\pi_{B_k}(A)$, and cov-err bounds are only known (see Appendix C.1) under some restrictions for Leverage Sampling, while unrestricted for the other randomized sampling algorithms.

5.2 Projection Algorithms

Figure 3 plots the covariance and projection error, as well as the runtime for various sketch sizes of 20 to 100 for the projection algorithms.

Otherwise, there were two clear classes of algorithms. For the same sketch size, Hashing and OSNAP perform a bit worse on projection error (most clearly on Noisy Random), and roughly the same in covariance error, compared to Random Projections and Fast JLT. Note that Fast JLT seems consistently better than others in cov-err, but we have chosen the best q parameter (sampling rate) by trial and error, so this may give an unfair advantage. Moreover, Hashing and OSNAP also have significantly faster runtime, especially as the sketch size grows. While Random Projections and Fast JLT appear to grow in time roughly linearly with sketch size, Hashing and OSNAP are basically constant. Section 5.4 on larger datasets and sketch sizes, shows that if the size of the sketch is not as important as runtime, Hashing and OSNAP have the advantage.

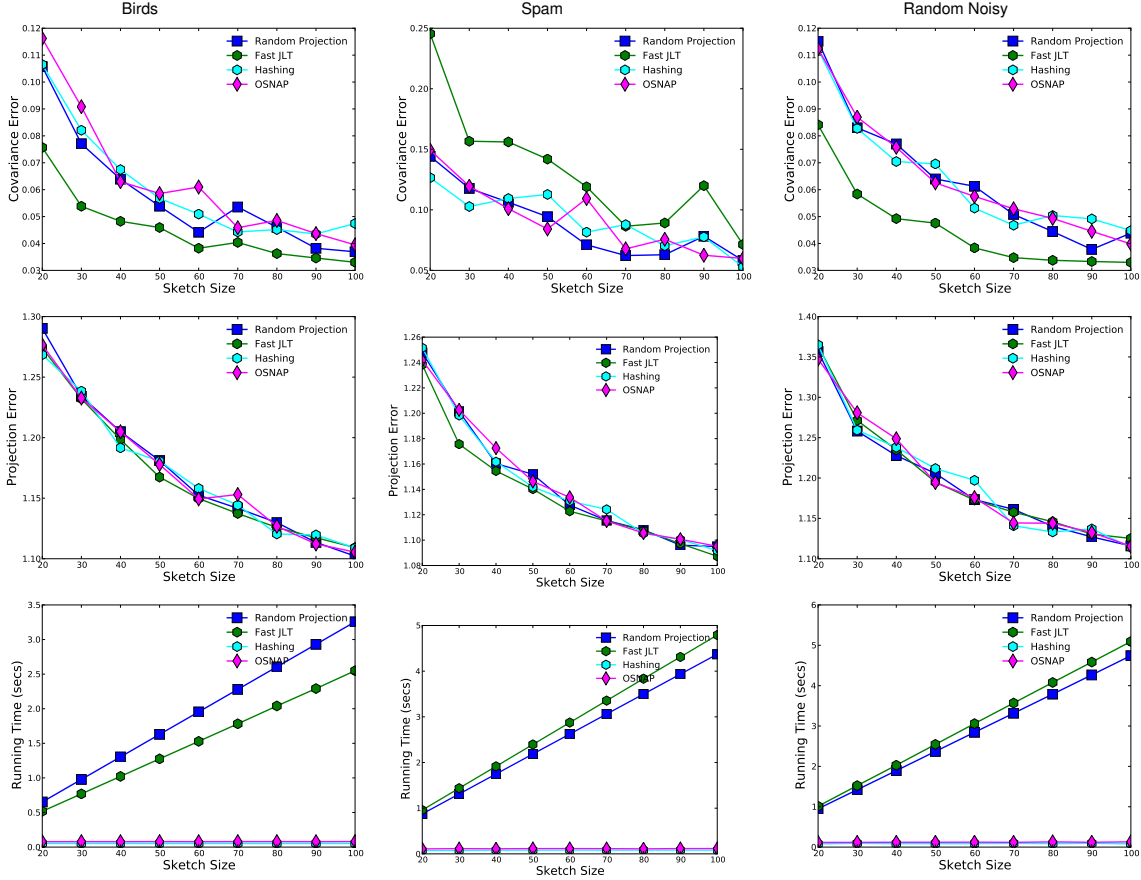


Figure 3: Projection algorithms on Birds(left), Spam(middle), and Random Noisy(30)(right).

5.3 Iterative Algorithms

Here we consider variants of FD. We first explore the α parameter in Parametrized FD, writing each version as α -FD. Then we compare against all of the other variants using explores from Parametrized FD.

In Figure 4 and 5, 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 on datasets Birds, Spam, and Random Noisy 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 $\text{err} = 0.032$ while iSVD and 0.2-FD have $\text{err} = 0.008$. Yet, as ℓ approaches 100, all algorithms seems to be approaching the same small error. In Figure 5, we explore the effect of α -FD on Random Noisy data by varying $m \in \{10, 20, 50\}$, and $m = 30$ in Figure 4. We observe that all algorithms get smaller error for smaller m (there are fewer “directions” 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.

Next in Figure 6, we compare iSVD, FD, and 0.2-FD with two groups of variants: one based on SS streaming algorithm (CFD and SSD) and another based on Fast FD. We see that CFD and SSD typically perform slightly better than FD in cov-err and same or worse in proj-err, 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 all the SS variants on matrix

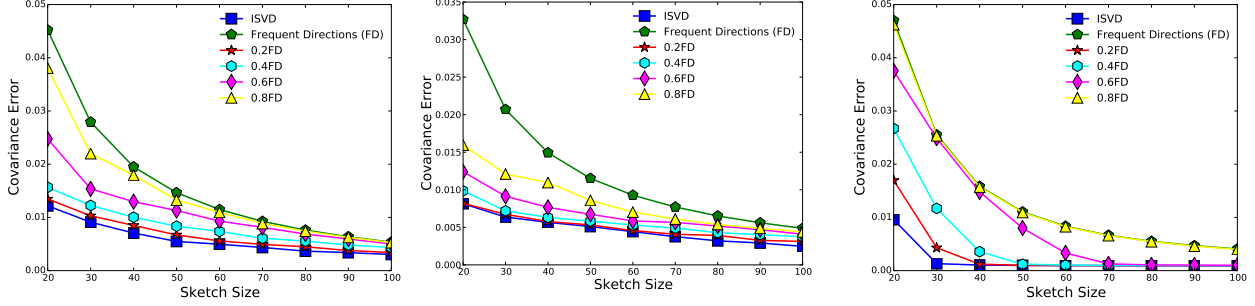


Figure 4: Parametrized FD on Birds (left), and Spam (middle), Random Noisy(30) (right).

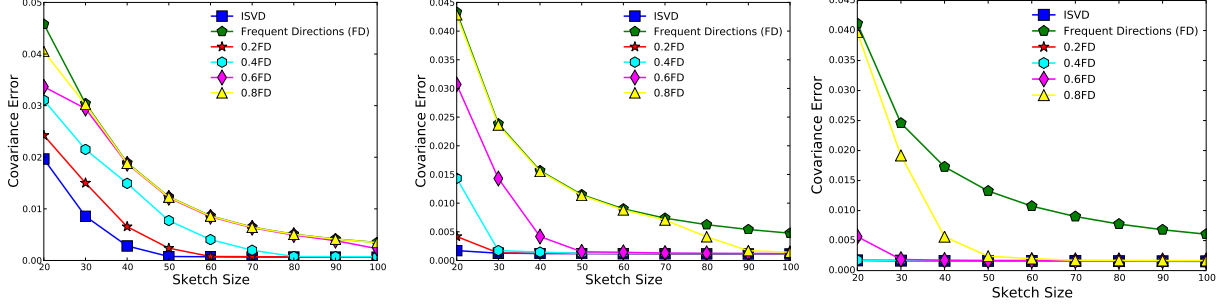


Figure 5: Parametrized FD on Random Noisy for $m = 50$ (left), 20 (middle), 10 (right).

sketching.

All variants achieve a very small error, but 0.2-FD, iSVD, and Fast 0.2-FD consistently matches or outperforms others in both cov-err and proj-err while Fast FD incurs more error compare to other algorithms. We also observe that Fast FD and Fast 0.2-FD are significantly (sometimes 10 times) faster than FD, iSVD, and 0.2-FD. Fast FD takes less time, sometimes half as much compared to Fast 0.2-FD, however, given its much smaller error Fast 0.2-FD seems to have the best all-around performance.

Data adversarial to iSVD. Next, using the Adversarial construction we show that iSVD is not always better in practice. In Figure 7, we see that iSVD can perform much worse than other techniques. Although at $\ell = 20$, iSVD and FD roughly perform the same (with about $\text{err} = 0.09$), iSVD does not improve much as ℓ increases, obtaining only $\text{err} = 0.08$ for $\ell = 100$. On the other hand, FD (as well as CFD and SSD) decrease markedly and consistently to $\text{err} = 0.02$ for $\ell = 100$. Moreover, all version of α -FD obtain roughly $\text{err}=0.005$ 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 wrap up this section, we demonstrate the scalability of these approaches on a much larger real data set **ConnectUS**. Figure 8 shows variants of Parameterized FD, and other iterative variants on this dataset. As the derived bounds on covariance error based on sketch size do not 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 with $\ell = 100$ 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

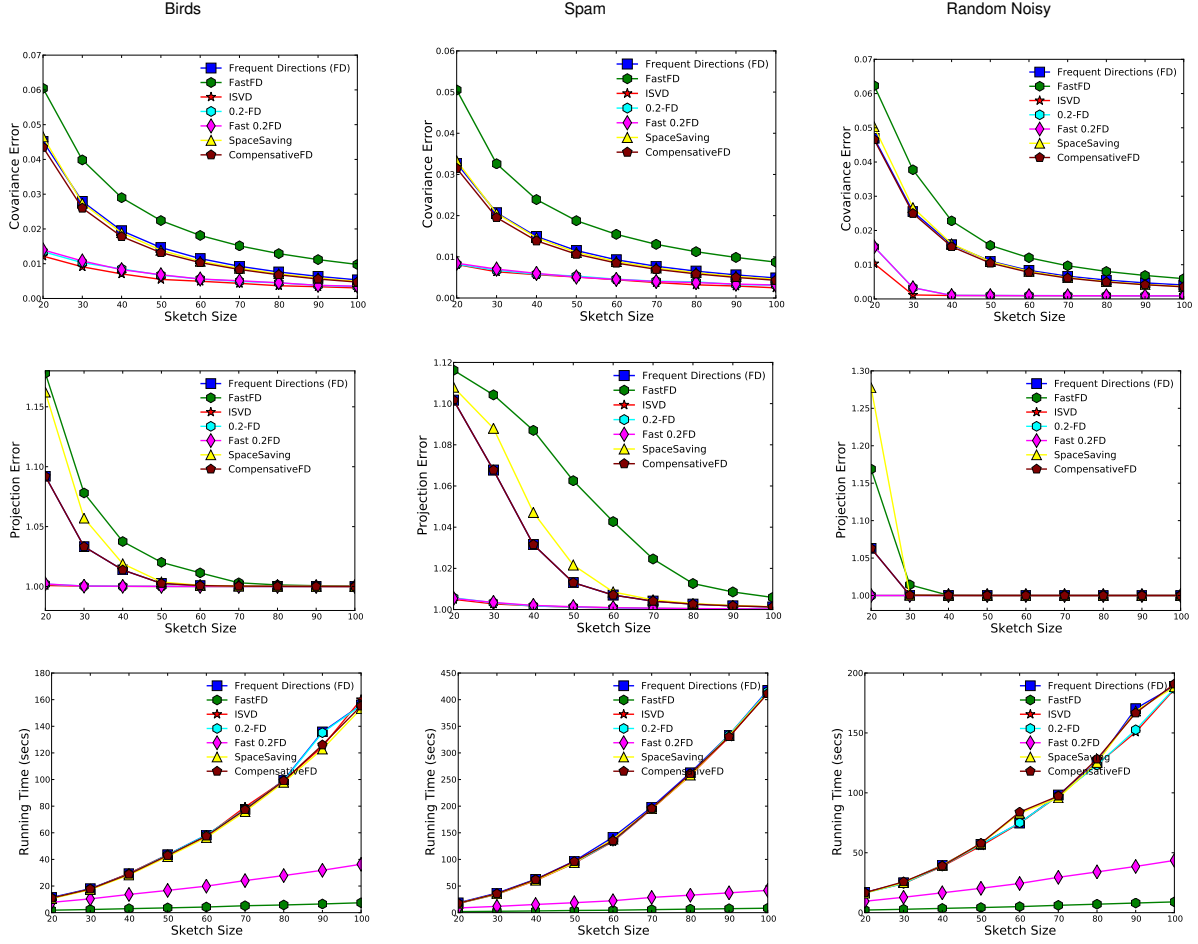


Figure 6: Iterative algorithms on Birds(left), Spam(middle), and Random Noisy(30)(right).

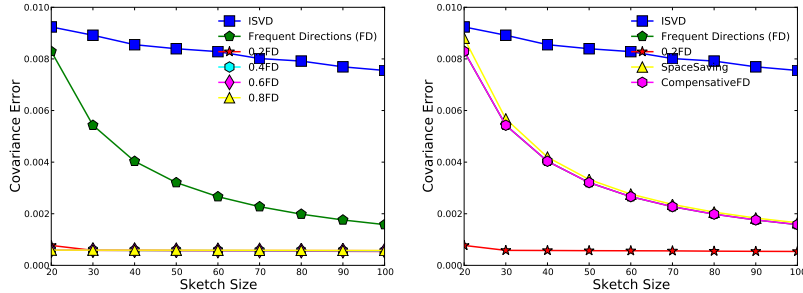


Figure 7: Demonstrating dangers of iSVD on Adversarial data.

unconnected users.

5.4 Global Comparison

Figure 9 shows the covariance error, projection error, as well as the runtime for various sketch sizes of $\ell = 20$ to 100 for the the leading algorithms from each category.

We can observe that the iterative algorithms achieve much smaller errors, both covariance and projection, than all other algorithms, sometimes matched by Deterministic Leverage. However, they are also significantly slower (sometimes a factor of 20 or more) than the other algorithms. The exception is Fast FD and

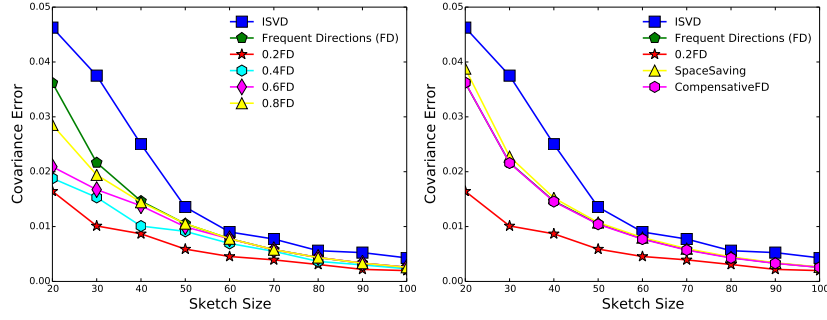


Figure 8: Parameterized FD (left), other iterative (right) algorithms on ConnectUS dataset.

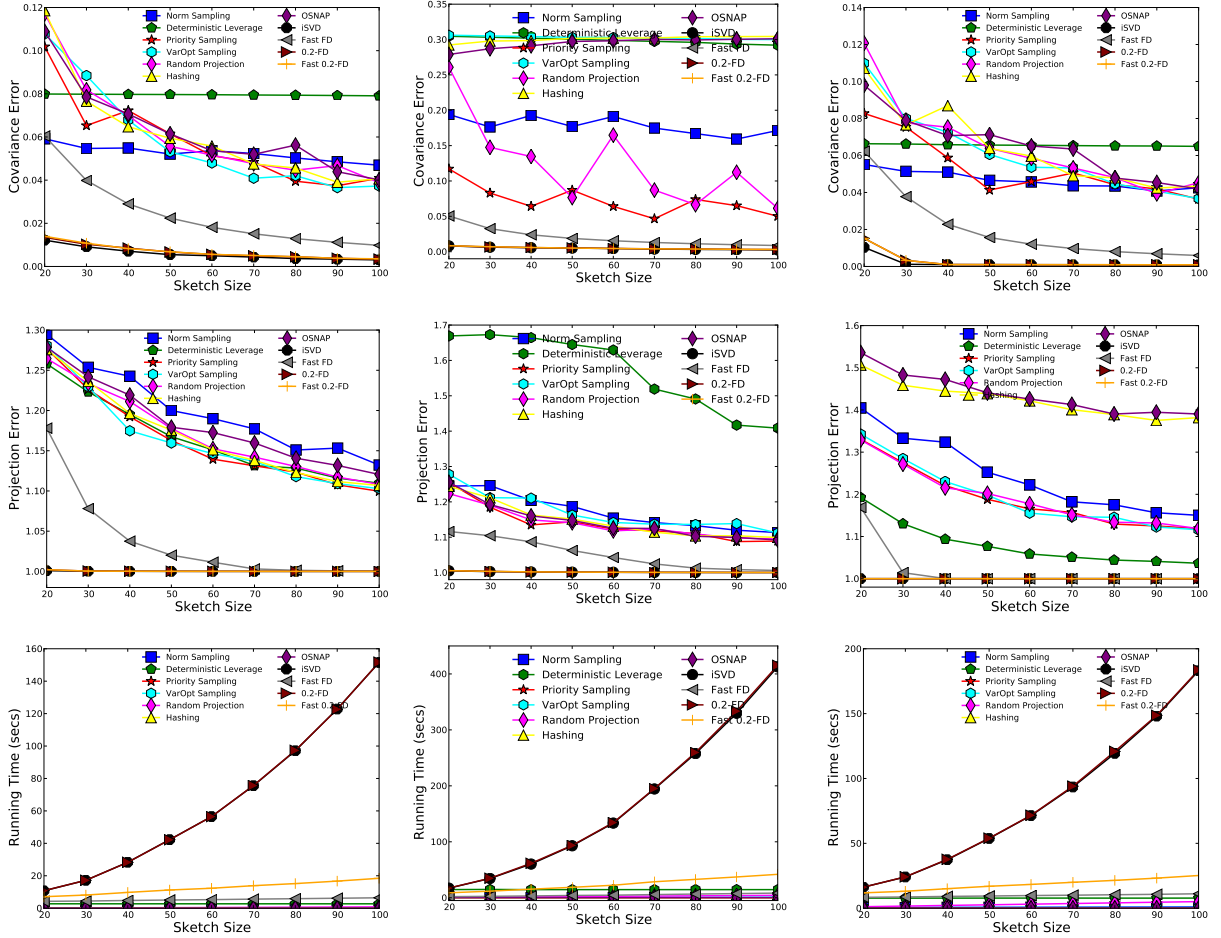


Figure 9: Leading algorithms on Birds(left), Spam(middle), and Random Noisy(30)(right).

Fast 0.2-FD, which are slower than the other algorithms, but not significantly so.

We also observe that for the most part, there is a negligible difference in the performance between the sampling algorithms and the projection algorithms, except for the Random Noisy dataset where Hashing and OSNAP result in worse projection error.

However, if we allow a much large sketch size for faster runtime and small error, then these plots do not effectively demonstrate which algorithm performs best. Thus in Figure 10 we run the leading algorithms on Birds as well as larger datasets, ConnectUS which is sparse and CIFAR-10 which is dense. We plot the error

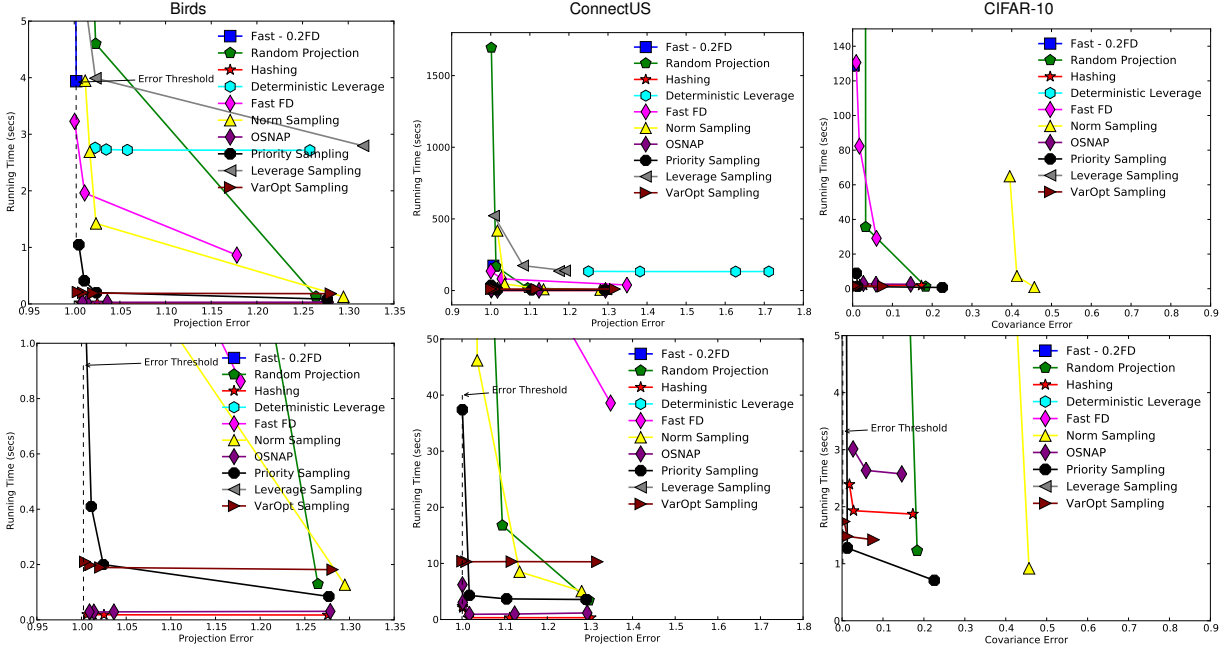


Figure 10: Projection error versus time on Birds and ConnectUS as well as Covariance error versus time on CIFAR-10. The second line shows close-ups

versus the runtime for various sketch sizes ranging up to $\ell = 10,000$. The top row of the plots shows most data points to give a holistic view, and the second row zooms in on the relevant portion.

For some plots, we draw an *Error Threshold* vertical line corresponding to the error achieved by Fast 0.2-FD using $\ell = 20$. Since this error is typically very low, but in comparison to the sampling or projection algorithms Fast 0.2-FD is slow, this threshold is a useful target error rate for the other leading algorithms.

We observe that Fast FD can sometimes match this error with slightly less time (see on Birds), but requires a larger sketch size of $\ell = 100$. Additionally VarOpt, Priority Sampling, Hashing, and OSNAP can often meet this threshold. Their runtimes can be roughly 100 to 200 times faster, but require sketch sizes on the order of $\ell = 10,000$ to match the error of Fast 0.2-FD with $\ell = 20$.

Among these fast algorithms requiring large sketch sizes we observe that VarOpt scales better than Priority Sampling, and that these two perform best on CIFAR-10, the large dense dataset. They also noticeably outperform Norm Sampling both in runtime and error for the same sketch size. On the sparse dataset ConnectUS, algorithms Hashing and OSNAP seem to dominate Priority Sampling and VarOpt, and of those two Hashing performs slightly better.

To put this space in perspective, on CIFAR-10 ($n = 60,000$ rows, 1.4GB memory footprint), to approximately reach the *error threshold* Hashing needs $\ell = 10,000$ and 234MB in 2.4 seconds, VarOpt Sampling requires $\ell = 5,000$ and 117MB in 1.2 seconds, Fast FD requires $\ell = 100$ and 2.3MB in 130 seconds, and Fast 0.2-FD requires $\ell = 20$ and 0.48MB in 128 seconds. All of these will easily fit in memory of most modern machines. The smaller sketch by Fast 0.2-FD will allow expensive downstream applications (such as deep learning) to run much faster. Alternatively, the output from VarOpt Sampling (which maintains interpretability of original rows) could be fed into Fast 0.2-FD to get a compressed sketch in less time.

Reproducibility. We provide public access to our results using a testbed facility *APT* [3]. *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://aptilab.net/p/MatrixApx/MatrixApproxComparision>.

References

- [1] <http://www.vision.caltech.edu/visipedia/CUB-200-2011.html>.
- [2] http://mlkd.csd.auth.gr/concept_drift.html.
- [3] <https://www.flux.utah.edu/project/apt>.
- [4] <http://www.cise.ufl.edu/research/sparse/matrices>.
- [5] <http://www.flux.utah.edu/project/emulab>.
- [6] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66:671–687, 2003.
- [7] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 2001.
- [8] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st Symposium on Principles of Database Systems*, 2012.
- [9] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of 38th ACM symposium on Theory of computing*, 2006.
- [10] Nir Ailon and Edo Liberty. An almost optimal unrestricted fast Johnson-Lindenstrauss transform. In *Proceedings of 22nd ACM-SIAM Symposium on Discrete Algorithms*, 2011.
- [11] Arvind Arasu, Shivnath Babu, and Jennifer Widom. An abstract semantics and concrete language for continuous queries over streams and relations. 2002.
- [12] Sanjeev Arora, Elad Hazan, and Satyen Kale. A fast random sampling algorithm for sparsifying matrices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2006.
- [13] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. *Lecture Notes in Computer Science*, pages 3–14, 2001.
- [14] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction. In *Proceedings of 52nd Annual Symposium on Foundations of Computer Science*, 2011.
- [15] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [16] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the 7th European Conference on Computer Vision*, 2002.
- [17] Moses Charikan, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of International Colloquium on Automata, Languages, and Programming*, 2002.
- [18] Jianjun Chen, David J DeWitt, Feng Tian, and Yuan Wang. Niagaraq: A scalable continuous query system for internet databases. *ACM SIGMOD Record*, 29:379–390, 2000.

- [19] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 2009.
- [20] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, 2013.
- [21] Edith Cohen, Nick Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. Stream sampling for variance-optimal estimation of subset sums. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [22] Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. In *Proceedings of the 34th International Conference on Very Large Data Bases*, 2008.
- [23] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55:58–75, 2005.
- [24] Corinna Cortes, Kathleen Fisher, Daryl Pregibon, and Anne Rogers. Hancock: a language for extracting signatures from data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.
- [25] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson-Lindenstrauss transform. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, 2010.
- [26] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of European Symposium on Algorithms*, 2002.
- [27] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *Proceedings of 51st IEEE Symposium on Foundations of Computer Science*, 2010.
- [28] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, 2006.
- [29] Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 2006.
- [30] Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.
- [31] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36:132–157, 2006.
- [32] 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.
- [33] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012.
- [34] Petros Drineas and Michael W. Mahoney. Effective resistances, statistical leverage, and applications to linear equation solving. In *arXiv:1005.3097*, 2010.
- [35] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.

- [36] Petros Drineas and Anastasios Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 111:385–389, 2011.
- [37] Nick Duffield, Carsten Lund, and Mikkel Thorup. Priority sampling for estimation of arbitrary subset sums. *Journal of the ACM*, 54:32, 2007.
- [38] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51:1025–1041, 2004.
- [39] Mina Ghashami, Feifei Li, and Jeff M. Phillips. Continuous matrix approximation on distributed data. In *Proceedings of the 40th International Conference on Very Large Data Bases*, 2014.
- [40] Mina Ghashami, Edo Liberty, Jeff M Phillips, and David P Woodruff. Frequent directions: Simple and deterministic matrix sketching. *arXiv preprint arXiv:1501.01711*, 2015.
- [41] Mina Ghashami and Jeff M Phillips. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of 25th ACM-SIAM Symposium on Discrete Algorithms*, 2014.
- [42] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Quicksand: Quick summary and analysis of network data. Technical report, DIMACS Technical Report, 2001.
- [43] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHUP, 2012.
- [44] Peter Hall, David Marshall, and Ralph Martin. Incremental eigenanalysis for classification. In *Proceedings of the British Machine Vision Conference*, 1998.
- [45] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26:189–206, 1984.
- [46] Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28:51–55, 2003.
- [47] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Technical Report*, 2009.
- [48] A. Levey and Michael Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9:1371–1374, 2000.
- [49] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the samples complexity of learning. *Journal of Computer and System Sciences*, 62:516–527, 2001.
- [50] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.
- [51] Michael W. Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106:697–702, 2009.
- [52] Ahmed Metwally, Divyakant Agrawal, and Amr El. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems*, 31:1095–1133, 2006.
- [53] Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2:143–152, 1982.

- [54] Jelani Nelson and Huy L. Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of 54th IEEE Symposium on Foundations of Computer Science*, 2013.
- [55] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 1998.
- [56] Dimitris Papapaliopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014.
- [57] David A Ross, Jongwoo Lim, Rwei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77:125–141, 2008.
- [58] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM*, 54:21, 2007.
- [59] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.
- [60] Mark Sullivan and Andrew Heybey. A system for managing large databases of network traffic. In *Proceedings of USENIX Annual Technical Conference*, 1998.
- [61] Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.
- [62] Suresh Venkatasubramanian and Qiushi Wang. The Johnson-Lindenstrauss transform: An empirical study. In *Proceedings of ALENEX Workshop on Algorithms Engineering and Experimentation*, 2011.
- [63] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11:37–57, 1985.
- [64] Killian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of 26th International Conference on Machine Learning*, 2009.
- [65] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10:1–157, 2014.
- [66] Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002.

A Generalized Error Bounds for FD

.

Lemma A.1. *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.

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

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 Property 1 and Property 2 proves the lemma. \square

Lemma A.2. 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.

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

This completes the proof of lemma. \square

B Error Bounds for SSD

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 $A_{[i]} = [a_1; a_2; \dots; a_i]$.

Lemma B.1. *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 observe 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. \square

Now we would like to prove the three Properties 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 B.2. *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 B.1 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 . \square

Thus \hat{B} satisfies the three Properties. 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 B.1. *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 $\|A - \pi_B^k(A)\|_F^2 \leq \|A - A_k\|_F^2 (\ell - 1) / (\ell - 1 - 2k)$.

C Adapting Known Error Bounds

Not many algorithms state bounds in terms of covariance error or the projection bounds in the particular setting we consider. Here we show using properties of these algorithms they achieve covariance error bound too.

C.1 Column Sampling Algorithms

Almost all column sampling algorithms have a projection bound in terms of $\|A - \pi_B A\|_F^2 \leq f(\varepsilon) \|A - A_k\|_F^2$ where $A \in \mathbb{R}^n \times d$ is the input matrix and $B \in \mathbb{R}^{\ell \times d}$ is the output sketch. Here we derive the other type of bound, cov-err, for the two main algorithms in this regime; i.e. Norm Sampling and Leverage Sampling. In our proof, we use a variant of Chernoff-Hoeffding inequality: Consider a set of r independent random variables $\{X_1, \dots, X_r\}$ where $0 \leq X_i \leq \Delta$. Let $M = \sum_{i=1}^r X_i$, then for any $\alpha \in (0, 1/2)$

$$\Pr[|M - \mathbf{E}[M]| > \alpha] \leq 2 \exp\left(\frac{-2\alpha^2}{r\Delta^2}\right).$$

Lemma C.1. Let $B \in \mathbb{R}^{\ell \times d}$ with $\ell = O(d/\varepsilon^2)$ be the output of Norm Sampling. Then with probability 99/100, for all unit vectors $x \in \mathbb{R}^d$

$$\text{cov-err}(A, B) = \frac{|\|Bx\|^2 - \|Ax\|^2|}{\|A\|_F^2} \leq \varepsilon.$$

Proof. Consider any unit vector $x \in \mathbb{R}^d$. Define ℓ independent random variables $X_i = \langle b_i, x \rangle^2$ for $i = 1, \dots, \ell$. Recall that Norm Sampling selects row a_j to be row b_i in the sketch matrix B with probability $\Pr(b_i \leftarrow a_j) = \|a_j\|^2 / \|A\|_F^2$, and rescales the sampled row as $\|b_i\|^2 = \|a_j\|^2 / (\ell \Pr(b_i \leftarrow a_j)) = \|A\|_F^2 / \ell$. Knowing these, we bound each X_i as $0 \leq X_i \leq \|b_i\|^2 = \|A\|_F^2 / \ell$ therefore $\Delta_i = \|A\|_F^2 / \ell$ for all X_i s. Setting $M = \sum_{i=1}^{\ell} X_i = \|Bx\|^2$, we observe

$$\mathbf{E}[M] = \sum_{i=1}^{\ell} \mathbf{E}[X_i] = \sum_{i=1}^{\ell} \sum_{j=1}^n \Pr(b_i \leftarrow a_j) \left\langle \frac{a_j}{\sqrt{\ell \Pr(b_i \leftarrow a_j)}}, x \right\rangle^2 = \sum_{i=1}^{\ell} \sum_{j=1}^n \frac{1}{\ell} \langle a_j, x \rangle^2 = \|Ax\|^2.$$

Finally using the Chernoff-Hoeffding bound and setting $\alpha = \varepsilon \|A\|_F^2$ yields

$$\Pr[|\|Bx\|^2 - \|Ax\|^2| > \varepsilon \|A\|_F^2] \leq 2 \exp\left(\frac{-2(\varepsilon \|A\|_F^2)^2}{\ell (\|A\|_F^2 / \ell)^2}\right) = 2 \exp(-2\varepsilon^2 \ell) \leq \delta.$$

Letting the probability of failure for that x be $\delta = 1/100$, and solving for ℓ in the last inequality, we obtain $\ell \geq \frac{1}{2\varepsilon^2} \ln(2/\delta) = \frac{1}{2\varepsilon^2} \ln(200)$.

However, this only holds for a single direction x ; we need this to hold for all unit vectors x . It can be shown, that allowing $\alpha = O(\varepsilon) \cdot \|A\|_F^2$, we actually only need this to hold for a net T of size $t = 2^{O(d)}$ such directions x [65]. Then by the union bound, setting $\delta = 1/(100t)$ this will hold for all unit vectors in T , and thus (after scaling ε by a constant) we can solve for $\ell = O((1/\varepsilon^2) \ln(t)) = O(d/\varepsilon^2)$. Hence with probability at least 99/100 we have $\text{cov-err}(A, B) = \frac{|\|Bx\|^2 - \|Ax\|^2|}{\|A\|_F^2} \leq \varepsilon$ for all unit vectors x . \square

We would like to apply a similar proof for Leverage Sampling, but we do not obtain a good bound for Δ in the Chernoff-Hoeffding bound. We rescale norm of each selected row b_i as

$$\|b_i\|^2 = \frac{\|a_j\|^2}{\ell} \cdot \frac{1}{\Pr(b_i \leftarrow a_j)} = \frac{\|a_j\|^2}{\ell} \cdot \frac{S_k}{s_j^{(k)}},$$

where $s_j^{(k)}$ is the rank- k leverage score of a_j , and $S_k = \sum_{j=1}^n s_j^{(k)} = k$. Unfortunately, $s_j^{(k)}$ can be arbitrarily small compared to S_k (e.g., if a_j lies almost entirely outside the best rank- k subspace). And thus we do not have a finite bound on Δ . As such we assume that $S_k/s_j^{(k)} \leq \beta$ for an absolute constant β , and then obtain a bound based on β .

Lemma C.2. *Let $B \in \mathbb{R}^{\ell \times d}$ with $\ell = O(d\beta^2/\varepsilon^2)$ be the output of Leverage Sampling. Under the assumption that rank- k leverage score of row a_j is bounded as $s_j^{(k)} \geq \beta S_k$ for a fixed constant $\beta > 0$, then with probability 99/100, for all unit vectors $x \in \mathbb{R}^d$*

$$\text{cov-err} = |\|Bx\|^2 - \|Ax\|^2|/\|A\|_F^2 \leq \varepsilon$$

Proof. Similar to Lemma C.1, we define ℓ random variables $X_i = \langle b_i, x \rangle^2$ for $i = 1, \dots, \ell$. Leverage Sampling algorithm selects row a_j to be row b_i with probability $\Pr(b_i \leftarrow a_j) = s_j^{(k)}/S_k$ and rescales sampled rows as $\|b_i\|^2 = \|a_j\|^2/(\ell \Pr(b_i \leftarrow a_j)) = (\|a_j\|^2/\ell)(S_k/s_j^{(k)}) \leq \|a_j\|^2/(\beta\ell) \leq \|A\|_F^2/(\beta\ell)$. Therefore $\Delta_i = \|A\|_F^2/(\beta\ell)$ for all X_i s. Setting $M = \sum_{i=1}^{\ell} X_i = \|Bx\|^2$ we observe:

$$\mathbf{E}[M] = \sum_{i=1}^{\ell} \mathbf{E}[X_i] = \sum_{i=1}^{\ell} \sum_{j=1}^n \Pr(b_i \leftarrow a_j) \left\langle \frac{a_j}{\sqrt{\ell \Pr(b_i \leftarrow a_j)}}, x \right\rangle^2 = \sum_{i=1}^{\ell} \sum_{j=1}^n \frac{1}{\ell} \langle a_j, x \rangle^2 = \|Ax\|^2$$

Using Chernoff-Hoeffding bound with parameter $\alpha = \varepsilon\|A\|_F^2$ gives:

$$\Pr[|\|Bx\|^2 - \|Ax\|^2| > \varepsilon\|A\|_F^2] \leq 2 \exp\left(\frac{-2\varepsilon^2\|A\|_F^4}{\sum_{j=1}^{\ell} (\beta^2/\ell^2)\|A\|_F^4}\right) = 2 \exp\left(\frac{-2\ell\varepsilon^2}{\beta^2}\right) \leq \delta$$

Again letting the probability of failure for that x be $\delta = 1/100$, we obtain $\ell \geq \frac{\beta^2}{2\varepsilon^2} \ln(2/\delta) = \frac{\beta^2}{2\varepsilon^2} \ln(200)$.

In order to hold this for all unit vectors x , again we consider a net T of size $t = 2^{O(d)}$ unit directions x [65]. Then by the union bound, setting $\delta = 1/(100t)$ this will hold for all such vectors in T , and thus we can solve for $\ell = O((\beta^2/\varepsilon^2) \ln(t)) = O(d\beta^2/\varepsilon^2)$. Hence with probability at least 99/100 we have $\text{cov-err}(A, B) = \frac{|\|Bx\|^2 - \|Ax\|^2|}{\|A\|_F^2} \leq \varepsilon$ for all unit vectors x . \square

C.2 Random Projection Algorithms

Most of random projection algorithms state a bound (with constant probability) that they can create a matrix $B = SA$ such that $\|Bx\| = (1 \pm \varepsilon)\|Ax\|$ (e.g. $(1 - \varepsilon)\|Ax\| \leq \|Bx\| \leq (1 + \varepsilon)\|Ax\|$) for all $x \in \mathbb{R}^d$.

Here we relate this to cov-err and proj-err. We first show that whether this bound is squared only affects ε by a constant factor.

Lemma C.3. *For $\varepsilon \in (0, \frac{1}{4})$, the inequality $\|Bx\| = (1 \pm \varepsilon)\|Ax\|$ implies $\|Bx\|^2 = (1 \pm 3\varepsilon)\|Ax\|^2$.*

Proof. The upper bound follows since $(1 + \varepsilon)^2 = 1 + 2\varepsilon + \varepsilon^2 \leq 1 + 3\varepsilon$ for $\varepsilon \in (0, \frac{1}{4})$. The lower bound follows since $(1 - \varepsilon)^2 = 1 + \varepsilon^2 - 2\varepsilon \geq 1 - 2\varepsilon$ for $\varepsilon \in (0, \frac{1}{4})$. \square

Now to relate this bound to cov-err, we will use $\rho(A) = \|A\|_F^2 / \|A\|_2^2$, the *numeric rank* of A , which is always at least 1.

Lemma C.4. *Given a matrix B such that $\|Bx\| = (1 \pm \varepsilon)\|Ax\|$ for all $x \in \mathbb{R}^d$, then when $\varepsilon \in (0, \frac{1}{4})$*

$$\text{cov-err} = \|A^T A - B^T B\|_2 / \|A\|_F^2 \leq 3\varepsilon / \rho(A).$$

Proof. Using Lemma C.3, $|\|Ax\|^2 - \|Bx\|^2| \leq 3\varepsilon\|Ax\|^2$. Now restrict $\|x\| = 1$, so then

$$x^T(A^T A - B^T B)x = x^T A^T A x - x^T B^T B x = |\|Ax\|^2 - \|Bx\|^2| \leq 3\varepsilon\|Ax\|^2.$$

Since this holds for all x such that $\|x\| = 1$, then it holds for the $x = x^*$ which maximizes the left hand size so $x^{*T}(A^T A - B^T B)x^* = \|A^T A - B^T B\|_2$ so

$$\|A^T A - B^T B\|_2 = x^{*T}(A^T A - B^T B)x^* \leq 3\varepsilon\|Ax^*\|^2 \leq 3\varepsilon\|A\|_2^2.$$

Dividing both sides by $\|A\|_F^2 = \rho(A)/\|A\|_2^2$ completes the proof. \square

We next relate this to the proj-err. We note that it may be possible that the full property $\|Bx\| = (1 \pm \varepsilon)\|Ax\|$ may not be necessary to obtain a bound on proj-err, but we are not aware of an explicit statement otherwise. There are bounds (see [65]) where one reconstructs a matrix \hat{A} which obtains the bounds below in place of $\pi_{B_k}(A)$, and these have roughly $1/\varepsilon$ dependence on ε ; however, they also have a factor n in their size.

Lemma C.5. *Given a matrix B such that $\|Bx\| = (1 \pm \varepsilon)\|Ax\|$ for all $x \in \mathbb{R}^d$, then when $\varepsilon \in (0, \frac{1}{4})$*

$$\text{proj-err} = \|A - \pi_{B_k}(A)\|_F^2 / \|A - A_k\|_F^2 \leq (1 + 24\varepsilon).$$

Proof. Let $V = [v_1, v_2, \dots, v_d]$ be the right singular vectors of A , so that $\|A_k\|_F^2 = \sum_{i=1}^k \|Av_i\|^2$ and $\|A - A_k\|_F^2 = \sum_{i=k+1}^d \|Av_i\|^2$. It follows from $\|B_k\|_F^2 \geq \sum_{i=1}^k \|Bv_i\|^2$ and Lemma C.3 that

$$\|B - B_k\|_F^2 \leq \sum_{i=k+1}^d \|Bv_i\|^2 \leq \sum_{i=k+1}^d \frac{1}{1 - 3\varepsilon} \|Av_i\|^2 \leq \frac{1}{1 - 3\varepsilon} \|A - A_k\|_F^2.$$

Let $R = [r_1, r_2, \dots, r_d]$ be the right singular vectors of B . Then by matrix Pythagorean and Lemma C.3

$$\begin{aligned} \|A - \pi_{B_k}(A)\|^2 &= \sum_{i=k+1}^d \|Ar_i\|^2 \leq \sum_{i=k+1}^d (1 + 3\varepsilon) \|Br_i\|^2 = (1 + 3\varepsilon) \|B - B_k\|_F^2 \\ &\leq \frac{1 + 3\varepsilon}{1 - 3\varepsilon} \|A - A_k\|_F^2 \leq (1 + 24\varepsilon) \|A - A_k\|_F^2. \end{aligned} \quad \square$$