Frequent Directions for Matrix Sketching with Provable Bounds: A Generalized Approach

Qianli Liao
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA, 02139
lql@mit.edu

ABSTRACT

We[1] consider the task of matrix sketching, which is obtaining a significantly smaller representation of matrix A while retaining most of its information (or in other words, approximates A well). In particular, we investigate a recent approach called Frequent Directions (FD) initially proposed by Liberty [5] in 2013, which has drawn wide attention due to its elegancy, nice theoretical guarantees and outstanding performance in practice. Two follow-up papers [3] and [2] in 2014 further refined the theoretical bounds as well as improved the practical performance. In this report, we summarize the three papers and propose a Generalized Frequent Directions (GFD) algorithm for matrix sketching, which captures all the previous FD algorithms as special cases without losing any of the theoretical bounds. Furthermore, we implemented our GFD algorithm and show experimental results on synthetic and real-world datasets from [2]. Specifically, we explored a class of GFD algorithms called Monotonic GFDs (mGFDs) and experimentally show that they have less approximation errors and are at least one order of magnitude faster than previous best algorithms. We will make the code and data publicly available to promote peer research.

Keywords

Frequent Items, Frequent Directions,
SVD, PCA, Theoretical Bounds,
Online Algorithms, Approximation, Complexity

1. INTRODUCTION

Humans are collecting and analysing increasingly large amount of data. Very often they are stored in matrices, which are simple but powerful representations for capturing, storing and analysing data. However, many of the matrix algorithms do not scale linearly with respect to the input size. Also, some of our hardware limitations, like memory size, increase sublinearly to the speed of data collection. Thus, developing algorithms to efficiently process large matrices with limited time and space is a crucial task.

[1]I am the only author. But I tend to use "we" in the paper instead of "I" because that is how I usually write papers. I may also release it as a technical report sometime. "we" sounds more professional.

Matrix sketching is an important operation for processing large data, which plays a critical role in compression, image processing, computer vision, machine learning and most of the data scientific fields. The primary goal is to significantly **reduce the size of a matrix while retaining most of its information**. There are many ways of sketching a matrix (e.g., reducing rows vs. reducing columns). In order to be very specific, **we define the "matrix sketching" in this report as follows**: Let us say there is a large matrix A of size n by d where n is the number of samples and d is the number of variables that each sample has, and $rank(A) = r$. We want to find a matrix B of size l by d, where $l << n$ is given. As suggested by [2], [3] and [5], there are two types of criteria to evaluate the quality of the sketch B:

1. **Additive Error Sketches (AES)** (suggested by [5]): We want B to minimize $||A^T A - B^T B||_2$, where $||.||_2$ denotes the spectral norm. In section 3.3, we show that we can instead minimize $sup(||Ax||^2 - ||Bx||^2), \forall x$, where x is a unit vector, since $||A^T A - B^T B||_2 = max_{x:||x||=1}(||Ax||^2 - ||Bx||^2)$.
2. **Relative Error Sketches (RES)** (suggested by [3],[2]): We want B to minimize $\frac{||A-\pi_{B_k}(A)||_F^2}{||A-A_k||_F^2}$, where $A_k = U_k S_k V_k^T$ is the optimal[2] rank k approximation of A, where U_k, S_k, V_k are the first k columns of $[U, S, V] = svd(A)$. Here $||.||_F^2$ is Frobenius norm (i.e., Euclidean norm). $\pi_{B_k}(A) = W^T B_k$, where W is the solution of the linear least square problem[3] $min_W \; ||A - W^T B_k||_F^2$, and B_k are the first $k \leq l$ rows of B.

Here are some intuitions of the two types of sketches. The

[2]Solving the rank k approximation using this procedure is optimal. Proofs are in `http://en.wikipedia.org/wiki/Low-rank_approximation`. The idea is that SVD provides optimal approximation error. We want to minimize the ratio between our approximation error to the optimal approximation error. The reason we do not use SVD directly is that it is expensive on a large matrix A. The whole point is to avoid performing SVD on large matrices.

[3]Explained in the Appendix. See appendix for the solution of W.

first criterion AES is straightforward to understand. It is clear that an AES is a good sketch since it minimizes the difference between the covariance matrices of A and B. It is important to note that the right singular vectors V are the eigenvectors of the covariance matrix. That is, let us say $[U_A S_A V_A] = svd(A)$ and $[U_B S_B V_B] = svd(B)$, $B^T B \approx A^T A \iff V_B \approx V_A$. V_A is very important for dimensional reduction (i.e., $A_{reduced} = A * V_k$, where V_k are the first columns of V_A) and other purposes. There, however, is a weakness of the **additive** property that it is sensitive to the absolute value of $A^T A$. Thus, people are more in favor of *relative* bounds on the errors.

The second criterion RES is not completely straightforward to understand[4]. Recall that we choose W that minimizes $||A - W^T B_k||_F^2$ (i.e., least square problem). One way of understanding it is that minimizing $||A - W^T B_k||_F^2$ is minimizing the error when representing each row of A as a weighted combination of rows in B_k. Since $W^T B_k$ is of rank k, it is minimizing the error of a rank k approximation of A. It is natural to compare its error with $||A - A_k||_F^2$ since the A_k is the optimal rank k approximation of A.

Another intuition is that, by solving the linear least square problem, we get $\pi_{B_k} = W^T = AB_k^T(B_k B_k^T)^+ B_k$. $P = B_k^T(B_k B_k^T)^+ B_k$ is a **Projection Matrix** (See appendix for the definition of Projection Matrix), since $P^2 = P$. A projection matrix P has a property that $I - P$ represents the subspace that is orthogonal to P. Minimizing $||A - AP||_F^2 = ||A(I - P)||_F^2$ is minimizing the information projected to the orthogonal space, which means maximizing the information projected to P.

In this report, we are going to derive bounds for both types of sketches in Section 3.3 and Section 3.4, respectively.

To get a sketch B of matrix A, we process each row of A once and discard it. So it supports either online (where one observe one sample/row at a time) or offline settings (where one can simply stream a random permutation of rows from A).

1.1 Summary of the contributions of this project

This is a reading, research and implementation project.

1. Reading: read and summarized [5], [3] and [2].
2. Research: (1) Propose an original algorithm called Generalized Frequent Directions (GFD) and proved related bounds. (2) Improved the time complexity of the algorithms with relative error bound from $O(n)$ iterations to $O(n/l)$ iterations (while keeping the same complexity for each iteration). This is an order of $O(1/l)$ speedup.
3. Implementation: (1) Implemented GFD (2) extensively explored the parameter space of Monotonic GFD (mGFD) (3) compared mGFD with different previous algorithms on real-world and synthetic data. (4) Will make code and data publicly available

[4]It is not explained in any of the papers. In the literature, $\pi_{B_k}(A) = AB_k^T(B_k B_k^T)^+ B_k$ is given directly with little explanation.

1.2 Frequent Directions for Matrix Sketching

Frequent Directions (FD) is a new approach proposed in 2013 inspired by Frequent Items (FI, see below for descriptions). We do not know how the original authors came up with this idea, but one natural train of thought could be:

- As mentioned before, the optimal approach to perform dimension/rank reduction is to do SVD on the entire dataset (i.e., SVD gives the optimal rank k approximation).
- But it is too expensive to do SVD on the entire matrix A, let us do it in a streaming fashion with a limited sized buffer.
- What if the buffer is full? Let us discard some samples(i.e., rows).
- What are the best candidates to discard? One cannot discard samples directly. Let us represent the samples as linear combination of some basis and discard some basis (i.e., directions) instead.
- What basis to discard? Probably the least useful basis (i.e., least frequent directions)
- How to get the least frequent directions? Let's use SVD and pick the ones that have small singular values.
- Conclusion: decrease some smallest singular values.
- After discarding least useful directions, we have some space for new samples. Then we keep iterating the above steps.

I believe the original author first came up with the algorithms, then defined the Additive Error Sketches (AES) and proved the bounds. So the discoveries are most likely be "algorithm-driven" — first one has an elegant algorithm and then he/she seeks for some formalizations and bounds. This is my answer to the question how Frequent Directions (FD) relates to the matrix sketching in my peer-review. The most obvious relationship is that it is doing some streaming/iterative SVD.

1.3 Background: Frequent Items

Frequent Directions (FD) is extended from the classic problem called Frequent Items (FI), which was invented multiple times [7], [1], [4], [6]. In a FI problem, there is a stream $s_1, s_2, s_3, ..., s_n$ of items drawn repeatably from a dictionary of m elements $M = \{1, 2, 3, ..., m\}$. The goal is to estimate the frequency f_i of any item $i \in M$. Each item is only seen once. This problem is trivial if the memory size is unlimited (we can keep a counter for each type of item). To make it meaningful, we restrict the memory size to $l < m$. In this case, it is not straightforward what counters to keep in memory.

The solution[5] given by [7] is simple and elegant. When there are less than l counters, it adds a new counter to the memory

[5]A tutorial of Frequent Items can be found in `http://www.cs.berkeley.edu/~satishr/cs270/sp11/rough-notes/Streaming-two.pdf`

buffer when encountering a new type of item. If the number of counters reaches l, it decrements all the counters. If a counter reaches 0, it is discarded. It can be proved that the estimated frequency $\widetilde{f}_i$ of item i satisfy the property: $f_i - \frac{n}{l} \leq \widetilde{f}_i \leq f_i$. The proof can be found in the above web link, [7] and [5].

1.4 Frequent Directions

Notations: A the original large matrix. B the sketch of the large matrix, starting with all empty rows. B is kept as a "buffer". a_i: the i-th row of A. We look at one row of A at a time.

Frequent Directions (FD) share the following several ideas with FI:

- If B still has empty rows, it is natural to just insert a_i into B.
- If B is full, we do a SVD on B: [U S V]=svd(B). After the SVD, $B = SV^T$. Singular values serve as a continuous version of "counters" of important directions (the larger the singular value, the more variability/information alone the direction). We "decrement the counters" by decreasing the singular values with some heuristics. When some singular values becomes 0, some corresponding rows become empty again. So we can accommodate new rows from A.
- Theoretical bounds can be shown on the time, space and approximation quality (as we will discuss in the rest of the report).

2. FREQUENT DIRECTIONS ALGORITHM AND ITS GENERALIZATION

2.1 Previous algorithms

For the sake of completeness, we list the all previous algorithms for Frequent Directions (FD) below and give each one a brief description.

First, as suggested by [2], all the previous FD algorithms share the same structure as follows (Algorithm 1. The difference between them are the ways of reducing the ranks of the singular values.

Algorithm 1 Frequent Directions algorithm for [2], [5] and [3]

```
Input: $A$,$l$,$\alpha$
Output: $B$
Code:
Initialize $B$ to be a empty matrix of size $l$ by $d$
for $i = 1$ to $n$ do
    Insert $a_i$ to the first nonzero row of B
    if $B$ has no nonzero row left then
        [U, S, V] = svd(B)
        C = $SV^T$ // only for proof purpose
        S' = ReduceRank(S)
        B = $S'V^T$
    end if
end for
```

First, the original FD paper [5] uses the following **ReduceRank** algorithm (Algorithm 2).

Algorithm 2 Function: ReduceRank of [5]

```
Input: $S, \alpha$
Output: $S'$
Code:
$\delta_i = \sigma_{l/2}^2$
S' = $\{\sqrt{\sigma_1^2 - \delta_i}, \sqrt{\sigma_2^2 - \delta_i}, ..., \sqrt{\sigma_l^2 - \delta_i}\}$
// for each negative $\sigma_j^2 - \delta_i$, set $\sqrt{\sigma_j^2 - \delta_i}$ zero.
SetToDiagMatrix(S') //set S' to a diagonal matrix form
return S'
```

Advantages: it zeros out half of the singular values every iteration, so it is very fast ($O(n/l)$ iterations in total).

Disadvantages: (1) it is not immediately clear how to derive relative error bounds using this $delta_i = \sigma_{l/2}$. At least nobody has given such a proof in the literature. In fact, we show it is not difficult, as we are going to prove the relative error bounds for its generalization. (2) the approximation quality is generally lower (see experiments) as one zeros out more singular values. However, this problem is again addressed by our Generalized Frequent Directions, where we can zero out many singular values while bounding the errors.

Then, the first follow-up paper [3] uses the following **ReduceRank** algorithm (Algorithm 3).

Algorithm 3 Function: ReduceRank of [3]

```
Input: $S, \alpha$
Output: $S'$
Code:
$\delta_i = \sigma_l^2$
S' = $\{\sqrt{\sigma_1^2 - \delta_i}, \sqrt{\sigma_2^2 - \delta_i}, ..., \sqrt{\sigma_l^2 - \delta_i}\}$
// note that $\sigma_j^2 - \delta_i$ cannot be negative, but can be 0 if $\sigma_j = \sigma_l$
SetToDiagMatrix(S') //set S' to a diagonal matrix form
return S'
```

Advantages: it provides relative error bounds (but only for $\delta_i = \sigma_l$)

Disadvantages: (1) it zeros out only one singular values every iteration, so it is very slow ($O(n)$ iterations[6] in total. It is probably too slow for any practical data. (2) the performance is not ideal because it decreases all the singular values by the same amount every iteration. It turned out to be not an ideal heuristic in practice, which does not have any theoretical advantage either.

[6]The number of iterations is the primary variability in the time complexity because the dominantly time consuming step is SVD, which is done once per iteration.

Finally, the second follow-up paper [2] uses the following **ReduceRank** algorithm (Algorithm 4).

Algorithm 4 Function: ReduceRank of [2]

```
**Input:** $S, \alpha$
**Output:** $S'$
**Code:**
$\delta_i = \sigma_l^2$
S' = $\{\sigma_1, \sigma_2, ..., \sigma_{l(1-\alpha)}, \sqrt{\sigma^2_{l(1-\alpha)+1} - \delta_i}, ..., \sqrt{\sigma_l^2 - \delta_i}\}$
// for each negative $\sigma_j^2 - \delta_i$, set $\sqrt{\sigma_j^2 - \delta_i}$ zero.
SetToDiagMatrix(S') //set S' to a diagonal matrix form
**return** S'
```

Advantages: (1) it provides relative error bounds (but only for $\delta_i = \sigma_l$) (2) it largely improved the approximation quality by using a heuristic of reducing only the last $\alpha * l$ singular values.

Disadvantages: it zeros out only one singular values every iteration, so it is very slow ($O(n)$ iterations in total). It is probably too slow for any practical data. (2) Although the this new heuristic is good in practice, we wonder:**can we do better?** Are there other similar heuristics that work as well and also allow relative error bounds?

2.2 Our Algorithm

In our generalized algorithm, we aim at addressing all the disadvantages of the previous algorithms.

To summarize the previous algorithms:

- The previous algorithms share very similar structures. It is natural to think about unifying them. Ideally, the unified algorithm should have the additive and relative error bounds. The proofs of such bounds are the main contributions of this report.
- The previous algorithm with relative bounds have a time complexity of $O(n)$ iterations. We should improved it to $O(n/l)$ iterations while maintaining all the error bounds.

Our GFD algorithm addresses these problems by providing a large class of algorithms with provable additive and relative error bounds. These algorithms are conveniently parameterized by a few variables. So one could write the code once and try any of them easily.

The main function of GFD is shown as follows (Algorithm 5):

Algorithm 5 Generalized Frequent Directions (GFD)

```
**Input:** $A$,$l$,$\alpha$
**Output:** $B$
**Code:**
Initialize $B$ to be a empty matrix of size $l$ by $d$
**for** $i = 1$ **to** $n$ **do**
  Insert $a_i$ to the first nonzero row of B
  **if** *B has no nonzero row left* **then**
    [U, S, V] = svd(B)
    C = $SV^T$ // only for proof purpose
    ***S' = GeneralizedReduceRank(S)***
    B = $S'V^T$
    ***Permute rows of B s.t. the empty rows of B are brought to the end***
  **end if**
**end for**
```

Our main contribution is the following "Generalized Reduce Rank" function (Algorithm 6):

Algorithm 6 Function: GeneralizedReduceRank

```
**Input:** $S, \vec{w}$
**Output:** $S'$
**Code:**
$\delta_i = H(\vec{w}, S)$
S' = $\{\sqrt{\sigma_1^2 - w_1\delta_i}, \sqrt{\sigma_2^2 - w_2\delta_i}, ..., \sqrt{\sigma_l^2 - w_l\delta_i}\}$
// for each negative $\sigma_j^2 - w_j\delta_i$, set $\sqrt{\sigma_j^2 - w_j\delta_i}$ zero.
SetToDiagMatrix(S') //set S' to a diagonal matrix form
**return** S'
```

$\vec{w} = [w_1, w_2, ..., w_l]$ is a weight vector (for convenience, symbols w and $\vec{w}$ are used interchangeablely. Without a subscript, they both denoting the entire weight vector). We define $0 \leq w_j \leq 1$ (useful for proving lemma 1).

H(,) is a function to determine δ_i s.t.

$$\delta_i \geq 0$$
$$\forall j\ ,\ \lambda\sigma_{pl}^2 \geq w_j\delta_i \geq 0$$

where $\lambda \geq 1$ is a constant and p is a constant $\in (0, 1]$, which means the $100 * p\%$ percentile of singular values (e.g., $\sigma_{0.9}$ should be larger than 90% of the singular values). This is important because we want to upper bound the amount of singular value we decrease by a multiple of percentile of the singular values, which is useful to prove lemma 1 and help to get relative error bounds.

It is optional to add a lower bound s.t. $\forall j\ w_j\delta_i \geq \sigma_{p_{low}l}$, where $p_{low} \leq p$ means another percentile. It reduces the number of iterations but does not affect the approximation bounds.

The λ is a scaling factor that adds more flexibility to the algorithm without affecting the theoretical bounds. When combined with the weights w, we are able to represent many more strategies of decreasing singular values. Currently, we have not found any interesting use of λ. So we can set it to 1 in most of the cases. Again, we added λ since it improves the expressive power of the model without affecting the provability of bounds.

The weights w are useful in many ways: (1) it allows us to explore a large number of algorithms and perhaps interpolations between known algorithms (2) when combined with some other objectives, the weights may be tuned in some interesting ways.

Note that the permutation of rows of B in the main algorithm is NOT necessary if w_j (see GeneralizedReduceRank function) is non-decreasing as j increases, since in which case a larger singular value will not be reduced to 0 before all smaller singular values are reduced to 0. Also, non-decreasing w_j seems to make more sense in practice (unless other special objectives are used).

2.3 Previous Algorithms as Special Cases

As said before, the algorithms in [5],[3] and [2] are special cases of our GFD algorithm. We will show how to choose H(,) and w_j to get the algorithms in above papers.

In the following text, we first describe in English that what the algorithm does and then give the **GFD parameters** that makes our GFD algorithm exactly the same as the described algorithm:

1. In [5], everytime when B is full, we decrement the all squared singular values by $\sigma^2_{l/2}$ (and then take a square root).
   GFD parameters: H(,) returns $\sigma^2_{l/2}$. $\forall j\ w_j = 1$; $\lambda = 1$; Choose p such that $\sigma_{pl} = \sigma_{l/2}$
2. In [3], everytime when B is full, we decrement all squared singular values by σ^2_l (and then take a square root).
   GFD parameters: H(,) returns σ^2_l. $\forall j\ w_j = 1$. $\lambda = 1$; Choose p such that $\sigma_{pl} = \sigma_l$
3. In [2], everytime when B is full, we keep the first $l * (1-\alpha)$ singular values unchanged, and we decrement the rest squared singular values by σ^2_l (and then take a square root), where $\alpha \in (0, 1)$ is a constant.
   GFD parameters: H(,) returns σ^2_l. $\forall j \le l * (1-\alpha)$, $w_j = 0$ and $\forall j > l * (1-\alpha)$, $w_j = 1$. $\lambda = 1$; Choose p such that $\sigma_{pl} = \sigma_l$.

3. ANALYSES

In this section, we provide theoretical analyses, approximation, runtime and space bounds for our Generalized Frequent Directions Algorithm. We adapted many procedures from [5],[3] and [2]. Since our model is a generalized version of [5],[3] and [2], we are NOT re-proving the properties. Instead, we are showing our original work, although the proofs bear many similarities with [5],[3] and [2]. Our bounds are slightly different due to our new parameters introduced. Also, we think our proofs are more organized than [5],[3] and [2] because we groupped analogous properties into same lemmas.

3.1 Notations

- A: the original matrix of size n by d, where n is the number of samples and d is the number of feature dimensions of a sample.
- B: the matrix sketch, the output matrix. It is of size l by d. $O(dl)$ is basically the space requirement of this algorithm.
- σ_j: the j-th singular value. Note that singular values are sorted in descending order. So $\sigma_j < \sigma_{j+1}, \forall j$.
- S: the singular value matrix, which is a diagonal matrix
- C_i: the C matrix (see the pseudocode above) of the i-th iteration.
- B_i: the intermediate B matrix (see the pseudocode above) of the i-th iteration. It is after the reduction of singular values and before the permutation (but actually permutation does not affect any of our analyses because $||B_i x||^2$ and $||B_i||^2_F$ are invariant under the permutations of rows of B_i).

3.2 Three Lemmas for Proving the Bounds of Approximation Errors

In this section, we will present three lemmas that help us prove the additive and relative bounds of approximation errors. Most contents of this section are pure mathematical derivations. Readers are encouraged temporally skip this section and look at the Section 3.3 and Section 3.4 to learn about what those bounds are — we will refer back to the lemmas from there.

Lemma 1 Motivation: First we will introduce Lemma 1 that capture the differences of the matrix sketch B before and after reducing the singular values. C is used to denote the state before decreasing, while B is used to denote the state of B after decreasing. See Algorithm 5 for details of this notation.

Lemma 1:

$$\forall x\ s.t. ||x||^2 = 1$$

$$0 \le ||C_i x||^2 - ||B_i x||^2 = \delta_i \sum_{j=1}^{j=l} w_j \langle v_j, x\rangle^2 \le \delta_i \quad (1)$$

$$\frac{\delta_i}{\lambda} \sum_{j=1}^{(1-p)l+1} w_j \le ||C_i||^2_F - ||B_i||^2_F \le \delta_i \sum_{j=1}^{j=l} w_j \quad (2)$$

Proof: First, **note that $||B_i x||^2$ and $||B_i||^2_F$ are invariant under the permutations of rows of B_i**. So we can prove the lemma for the B_i before the permutation. The same inequalities hold after the row permutation.

Let us prove the right side of the inequality:

$$\begin{aligned}
&||C_i x||^2 - ||B_i x||^2 \\
&= ||S_i V_i x||^2 - ||S_i' V_i x||^2 \\
&= \sum_{j=1}^{j=l} (\sigma_j \langle v_j, x\rangle)^2 - \sum_{j=1}^{j=l} (\sigma_j' \langle v_j, x\rangle)^2 \\
&= \sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2) \langle v_j, x\rangle^2 \\
&= \sum_{j=1}^{j=l} w_j \delta_i \langle v_j, x\rangle^2 \\
&= \delta_i \sum_{j=1}^{j=l} w_j \langle v_j, x\rangle^2 && \textit{by } 0 \le w_j \le 1 \\
&\le \delta_i \sum_{j=1}^{j=l} \langle v_j, x\rangle^2 && \textit{by } \sum_{j=1}^{j=l} \langle v_j, x\rangle^2 = 1, \forall\, \textit{unit } x \\
&= \delta_i
\end{aligned}$$

the left side of the inequality is easy:

$$\begin{aligned}
&||C_i x||^2 - ||B_i x||^2 \\
&= \delta_i \sum_{j=1}^{j=l} w_j \langle v_j, x\rangle^2 && \textit{by } 0 \le w_i \le 1 \\
&\ge 0 && \textit{by nonnegativity of terms}
\end{aligned}$$

Let's then prove second inequality:

$$\begin{aligned}
&||C_i||_F^2 - ||B_i||_F^2 \\
&= ||S_i V_i||_F^2 - ||S_i' V_i||_F^2 \\
&= \sum_{j=1}^{j=l} \sigma_j^2 - \sum_{j=1}^{j=l} \sigma_j'^2 \\
&= \sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2)
\end{aligned}$$

Recall that we want to prove:

$$\sum_{j=1}^{(1-p)l+1} w_j \frac{\delta_i}{\lambda} \le \sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2) \le \sum_{j=1}^{j=l} w_j \delta_i$$

Recall the property of H(,), $\forall j$, $\lambda \sigma_{pl}^2 \ge w_j \delta_i \ge 0$, where $\lambda \ge 1$ is a constant and p is a constant $\in (0, 1]$.

Let us analyse the lower bound of $\sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2)$:

- Let us think about the easy case when $\lambda = 1$, $\sigma_{pl}^2 \ge w_j \delta_i$. After subtracting $w_j \delta_i$ from each squared singular value, it is clear that at least $(1-p)l + 1$ singular values (top $(1-p)l + 1$ ones) are non-negative. So $w_j \delta_i$ was substracted entirely from them. That is $\forall\ i = 1, ..., (1-p)l + 1, \sigma_j^2 - \sigma_j'^2 = w_j \delta_i$. For the rest of the singular values, there is no guarantees, that is, $\forall\ i = (1-p)l + 2, ..., l, \sigma_j^2 - \sigma_j'^2 \le w_j \delta_i$. So $\sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2) \ge \sum_{j=1}^{(1-p)l+1} w_j \delta_i$.

- In the case when $\lambda \ge 1$, $\lambda \sigma_{pl}^2 \ge w_j \delta_i$. So $\sigma_{pl}^2 \ge \frac{w_j \delta_i}{\lambda}$. Thus, at least $(1-p)l + 1$ squared singular values (top $(1-p)l + 1$ ones) are non-negative after substracting $W = \frac{w_j \delta_i}{\lambda}$ from them. However, the true value substracted from them is greater than W, so $\forall\ i = 1, ..., (1-p)l + 1, \sigma_j^2 - \sigma_j'^2 \ge \frac{w_j \delta_i}{\lambda}$. Therefore, $\sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2) \ge \frac{\sum_{j=1}^{(1-p)l+1} w_j \delta_i}{\lambda}$.

Integrality and comments about p and l: For the sake of mathematical rigorousness, we assume can only choose the percentile p such that pl is an integer. Suppose we have l singular values $\sigma_1, \sigma_2, ..., \sigma_l$ sorted in descending order (WLOG assuming strictly descending) from left to right. If we decrease σ_{pl}^2 (the pl-th singular value from **right to left**) from each squared singular values, the pl-th result (from right to left) is 0. The pl-th result from **right to left** is the $(1-p)l + 1$-th result from **left to right**. So the $(1-p)l$-th result is the first positive value and the $(1-p)l + 2$ result is the first negative value (both from left to right).

Thus, we have:

$$\frac{\sum_{j=1}^{(1-p)l+1} w_j \delta_i}{\lambda} \le \sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2)$$

The upper bound of $\sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2)$ is obvious. Because $\forall\ j$, $\sigma_j'^2 = max(\sigma_j^2 - w_j \delta_i, 0)$, each squared singular value cannot decrease more than $w_j \delta_i$. Summing over all j, we get $||C_i||_F^2 - ||B_i||_F^2 \le \sum_{j=1}^{j=l} w_j \delta_i$.

Therefore,

$$\sum_{j=1}^{(1-p)l+1} w_j \frac{\delta_i}{\lambda} \le \sum_{j=1}^{j=l} (\sigma_j^2 - \sigma_j'^2) \le \sum_{j=1}^{j=l} w_j \delta_i$$

□

Lemma 2 Motivation: In the Lemma 2, we are going to relate the samples added every iteration to the difference of B before and after reducing rank.

Lemma 2: let D_i be all the samples inserted to the blank areas of B_{i-1}. We know first t samples in C_i are B_{i-1} and the rest are D_i.

$$\forall x \ s.t. ||x||^2 = 1$$

$$||D_i x||^2 = ||C_i x||^2 - ||B_{i-1}x||^2 \tag{3}$$

$$||D_i||_F^2 = ||C_i||_F^2 - ||B_{i-1}||_F^2 \tag{4}$$

Proof: Recall that we initialize B to be a empty matrix, so $||B_0||_F^2 = 0$ and $||B_0 x||^2 = 0$. It is easy to verify that $||D_1 x||^2 = ||C_1 x||^2$ and $||D_1||_F^2 = ||C_1||_F^2$. So these equations hold for $i = 1$.

Let the j-th sample of C_i be c_j. We know first t samples in C_i are B_{i-1} and the rest are D_i.

For the first equality:

$$\begin{aligned}
&||C_i x||^2 - ||B_{i-1}x||^2 \\
&= \sum_{j=1}^{j=t}(c_j x)^2 + \sum_{i=t+1}^{i=l}(c_j x)^2 - \sum_{j=1}^{j=t}(c_j x)^2 \\
&= \sum_{i=t+1}^{i=l}(c_j x)^2 \\
&= ||D_i x||^2
\end{aligned}$$

The first transition $||B_{i-1}x||^2 = \sum_{j=1}^{j=t}(c_j x)^2$ is correct because the first t samples in C_i are B_{i-1}.

For the second equality:

$$\begin{aligned}
&||C_i||_F^2 - ||B_{i-1}||_F^2 \\
&= \sum_{j=1}^{j=t}||c_j||^2 + \sum_{i=t+1}^{i=l}||c_j||^2 - \sum_{j=1}^{j=t}||c_j||^2 \\
&= \sum_{i=t+1}^{i=l}||c_j||^2 \\
&= ||D_i||_F^2
\end{aligned}$$

The first transition $||B_{i-1}||_F^2 = \sum_{j=1}^{j=t}||c_j||^2$ is correct because the first t samples in C_i are B_{i-1}.

□

Lemma 3 Motivation: In Lemma 1 and 2, we studied relationships between the sample encountered per iteration and B before and after reducing rank. In this lemma, we will combine Lemma 1 and 2 to derive the relationship between all the samples (i.e., A) and the final B. This is easily done by adding up the inequalities/equalities of all iterations and telescope all the intermediate terms.

Lemma 3: Let $\Delta = \sum_i \delta_i$.

$$\forall x \ s.t. ||x||^2 = 1$$

$$0 \le ||Ax||^2 - ||Bx||^2 \le \Delta \tag{5}$$

$$||A||_F^2 - ||B||_F^2 \ge \frac{\Delta}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j \tag{6}$$

Proof: For the inequality (first one), we use Lemma 2 Equation (3). Say there are T iterations in total (i.e., B_q is B).

$$\begin{aligned}
&||Ax||^2 \\
&= \sum_{i=1}^{i=T}||D_i x||^2 \\
&= \sum_{i=1}^{i=T}(||C_i x||^2 - ||B_{i-1}x||^2) && \textit{by Lemma } 2 \\
&\le \sum_{i=1}^{i=T}(||B_i x||^2 + \delta_i - ||B_{i-1}x||^2) && \textit{by Lemma } 1 \\
&= ||B_T x||^2 + \sum_i \delta_i - ||B_0 x||^2 \\
&= ||Bx||^2 + \sum_i \delta_i \\
&= ||Bx||^2 + \Delta
\end{aligned}$$

$$\begin{aligned}
&||Ax||^2 \\
&= \sum_{i=1}^{i=T}(||C_i x||^2 - ||B_{i-1}x||^2) && \textit{by Lemma } 2 \\
&\ge \sum_{i=1}^{i=T}(||B_i x||^2 - ||B_{i-1}x||^2) && \textit{by Lemma } 1 \\
&= ||B_T x||^2 - ||B_0 x||^2 \\
&= ||Bx||^2
\end{aligned}$$

Thus, $0 \le ||Ax||^2 - ||Bx||^2 \le \Delta$

For the second inequality, we use Lemma 2 Equation (4)

$$\begin{aligned}
&||A||_F^2 \\
&= \sum_{i=1}^{i=T}||D_i||_F^2 \\
&= \sum_{i=1}^{i=T}(||C_i||_F^2 - ||B_{i-1}||_F^2) && \textit{by Lemma } 2 \\
&\ge \sum_{i=1}^{i=T}(||B_i||_F^2 + \frac{\delta_i}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j - ||B_{i-1}||_F^2) && \textit{by Lemma } 1 \\
&= ||B_T||_F^2 + \frac{\sum_i \delta_i}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j - ||B_0||_F^2 \\
&= ||B||_F^2 + \frac{\Delta}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j
\end{aligned}$$

Thus, $||A||_F^2 - ||B||_F^2 \ge \frac{\Delta}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j$

□

3.3 Additive Error Bound for Generalized Frequent Directions

Theorem 1: Additive Error Bound for Generalized Frequent Directions.

We want to minimize $||A^TA - B^TB||_2 = max_{x:||x||=1}(||Ax||^2 - ||Bx||^2)$.

We will give a bound on $||Ax||^2 - ||Bx||^2$. We will show:

$$\begin{aligned} &\forall x\ s.t. ||x||^2 = 1 \\ 0 \leq &||Ax||^2 - ||Bx||^2 \leq \epsilon ||A||_F^2 \end{aligned} \tag{7}$$

Proof:

$$\begin{aligned} \Delta &\leq \frac{\lambda}{\sum_{j=1}^{(1-p)l+1} w_j} ||A||_F^2 \quad Lemma\,3\,Equation(6) \\ 0 \leq ||Ax||^2 - ||Bx||^2 &\leq \frac{\lambda}{\sum_{j=1}^{(1-p)l+1} w_j} ||A||_F^2 \quad Lemma\,3\,Equation(5) \end{aligned}$$

Then we choose p, w and l such that $\epsilon = \frac{\lambda}{\sum_{j=1}^{(1-p)l+1} w_j}$.

Interpretation: This bound tells us that alone all directions x, A and its approximation B are close enough.

Comments: As a rule of thumb, the higher the p, the higher the upper bound of decrease, the less iterations the algorithm needs and the faster the algorithm is.

Note that it is not completely straightforward/meaningful to discuss the range of ϵ since GFD is NOT a single algorithm, it is instead a collection of algorithms. They are very diverse, some of which are good and some of which are bad. In order to discuss ϵ, one needs to fix some parameters such as w and λ. Our contribution is that for any of the algorithms in GFD, we can examine its additive error bound and relative error bound by simply plugging in the parameters (instead of deriving it again).

$\square$

3.4 Relative Error Bounds for Generalized Frequent Directions

Mimicing the procedures of [2], we provide relative error bounds for our algorithm.

We will think from a low-rank approximation perspective. Let us use A_k to denote the rand k approximation of matrix A using standard SVD. Let us say [U S V] = svd(A). Then $A_k = U_k S_k V_k^T$, where U_k, S_k, V_k are the first k columns of U, S, V.

Let B_k be the rank k approximation of B, that is $B_k = S'_k V_k^T$, where the S'_k, V_k here are the first *columns* rows of S', V in the each iteration of our algorithm.

Let's assume the ideal rank k approximation is provided by SVD on the original data. We can evaluate our Frequent-Direction(FD) based rank k approximation by comparing to the approximation obtained by SVD:

- **Goal** find a rank k approximation of A using our algorithm (instead of directly using SVD, since SVD on A is expensive).
- **Approximation algorithm:** Run our Generalized FD algorithm to get B. Let the first k rows of B be B_k. As suggested by [2] and discussed in the introduction section, the approximated rank k approximation of A is $\widetilde{A_k} = AB_k^T(B_kB_k^T)^+B_k$, where $(.)^+$ means taking the Moore-Penrose Pseudoinverse. It means projecting A onto the rowspace spanned by B_k.

  There is a property about $\widetilde{A_k}$: if we run SVD on B to get right singular vectors V and pick the first k columns from it, calling it V_k, then $||\widetilde{A_k}||_F^2 = ||AV_k||_F^2$. Actually, this property holds for any (even random) A and B. We will use this property in the proof of Theorem 2. It is easy to verify this property.
- **The quality of approximation:** We can measure the quality of our FD-approximation $\widetilde{A_k}$ by comparing its error to that of the SVD-approximation A_k. That is, we will show below that

$$||A - \widetilde{A_k}||_F^2 \leq (1+\epsilon)||A - A_k||_F^2$$

Theorem 2: the bound for relative error of matrix sketching: We will show the following:

$$||A - \widetilde{A_k}||_F^2 \leq (1+\epsilon)||A - A_k||_F^2$$

In order to prove this theorem, we first prove a lemma:

Lemma 4:

$$\begin{aligned} &\forall x\ s.t.\ ||x||^2 = 1 \\ &||Ax||^2 - ||Bx||^2 \leq \Delta \leq \frac{||A - A_k||_F^2}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda} - k} \end{aligned} \tag{8}$$

(9)

Proof: Let v_i be the i-th singular vector of A.

$$\begin{aligned} &\frac{\Delta}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j \\ &\leq ||A||_F^2 - ||B||_F^2 && by\ Lemma\ 3 \\ &= \sum_{j=1}^{k} ||Av_j||^2 + \sum_{j=k+1}^{d} ||Av_j||^2 - ||B||_F^2 && property\ 1 \\ &= \sum_{j=1}^{k} ||Av_j||^2 + ||A - A_k||_F^2 - ||B||_F^2 && property\ 2 \\ &\leq \sum_{j=1}^{k} ||Av_j||^2 + ||A - A_k||_F^2 - \sum_{j=1}^{k} ||Bv_j||^2 && property\ 3 \\ &= \sum_{j=1}^{k}(||Av_j||^2 - ||Bv_j||^2) + ||A - A_k||_F^2 \\ &\leq k\Delta + ||A - A_k||_F^2 && Lemma\ 1 \end{aligned}$$

Thus

$$\frac{\Delta}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j \le k\Delta + ||A-A_k||_F^2$$

$$\Delta(-k+\frac{1}{\lambda}\sum_{j=1}^{(1-p)l+1} w_j) \le ||A-A_k||_F^2$$

$$\Delta \le \frac{||A-A_k||_F^2}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k}$$

$$||Ax||^2 - ||Bx||^2 \le \frac{||A-A_k||_F^2}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k} \qquad \textit{lemma } 1$$

Property 1:

$$\begin{aligned}||A||_F^2 &= \sum_{j=1}^{d} ||Av_j||^2\\ &= \sum_{j=1}^{k}||Av_j||^2 + \sum_{j=k+1}^{d}||Av_j||^2\end{aligned}$$

Property 2:

$$\sum_{j=k+1}^{d}||Av_j||^2 = ||A-A_k||_F^2$$

Proof: Let u_j be the j-th left singular vector,

$$\begin{aligned}\sum_{j=k+1}^{d}||Av_j||^2 &= \sum_{j=k+1}^{d}||u_j\sigma_j||^2 && SVD\ definition\\ &= \sum_{j=k+1}^{d}\sigma_j^2 && ||u_j||=1\\ &= \sum_{j=1}^{j=d}\sigma_j^2 - \sum_{j=1}^{j=k}\sigma_j^2\\ &= ||A||_F^2 - ||A_k||_F^2\\ &= ||A-A_k||_F^2 && Pythagorean\ Theorem[2]\end{aligned}$$

Property 3: Let h_i be the i-th column of the right singular vectors of B

$$\sum_{j=1}^{k}||Bv_j||^2 \le \sum_{j=1}^{k}||Bh_j||^2 \le ||B||_F^2$$

□

Proof of Theorem 2: Similar to [5]. Let v_i be the i-th column of V_k (the first k columns of the right singular vectors of B). **Note: v_i was the right singular vector of A in lemma 4. Do not be confused.**

Let q_i be the i-th column of the right singular vectors of A (which is never actually computed but we just use it for the proof).

$$\begin{aligned}&||A-\widetilde{A_k}||_F^2\\ &= ||A||_F^2 - ||\widetilde{A_k}||_F^2 && Pythagorean\ Theorem\\ &= ||A||_F^2 - ||AV_k||_F^2\\ &= ||A||_F^2 - \sum_{i=1}^{k}||Av_i||^2\\ &\le ||A||_F^2 - \sum_{i=1}^{k}||Bv_i||^2 && By\ Lemma\ 1\\ &\le ||A||_F^2 - \sum_{i=1}^{k}||Bq_i||^2 && \sum_{i=1}^{k}||Bv_i||^2 \ge \sum_{i=1}^{k}||Bq_i||^2\\ &\le ||A||_F^2 - \sum_{i=1}^{k}(||Aq_i||^2-\Delta)\\ &= ||A||_F^2 - ||Ak||_F^2 + k\Delta\\ &= ||A-Ak||_F^2 + k\Delta\\ &\le ||A-Ak||_F^2 + k\frac{||A-A_k||_F^2}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k} && Lemma\ 4\\ &= (1+\frac{k}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k})||A-A_k||_F^2\end{aligned}$$

Thus

$$||A-\widetilde{A_k}||_F^2 \le (1+\frac{k}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k})||A-A_k||_F^2$$

Note that I use $\sum_j w_j$ and $\sum_{j=1}^{j=l} w_j$ interchangeablely.

If we choose w_j and p such that $\frac{k}{\frac{\sum_{j=1}^{(1-p)l+1} w_j}{\lambda}-k} = \epsilon$, we get:

Again, note that it is not completely straightforward/meaningful to discuss the range of ϵ since GFD is NOT a single algorithm, it is instead a collection of algorithms. They are very diverse, some of which are good and some of which are bad. In order to discuss ϵ, one needs to fix some parameters such as w and λ. Our contribution is that for any of the algorithms in GFD, we can examine its additive error bound and relative error bound by simply plugging in the parameters (instead of deriving it again).

$$||A-\widetilde{A_k}||_F^2 \le (1+\epsilon)||A-A_k||_F^2$$

□

3.5 Time Complexity

The time complexity of this algorithm is basically determined by the number of iterations, which depends on the parameters (H(,) and w).

The number of iterations depends on how many singular values are decreased to 0 in each iteration. Each 0 singular value will clear one row of B. In paper [3] and [2], the algorithm guarantees that at least one singular value is set to 0 every iteration. So there will be $O(n)$ iterations, implying $O(n)$ small SVD operations, which is very slow in practice. In the original paper [5], the algorithm guarantees that half of the singular values are set to 0, which is considerably faster. It only requires $O(n/l)$ iterations. Since the above algorithms are special cases of GFD, we can achieve either of properties above.

In general, GFD algorithm can do better than $O(n)$ iterations, if we set a lower bound of decrement of squared singular values to be a percentile of the squared singular values. In this case, we zero out $O(l)$ singular values per iteration and thus process $O(l)$ samples per iteration (i.e., per SVD). The number of SVD performed is $O(n/l)$ throughout the entire algorithm (just like [5]). This does not affect any bounds we derived, because we did not assume any lower bound requirement of the decrement in any of the proofs.

3.6 Space Complexity

The space complexity of this algorithm is the size of matrix B, which is $O(dl)$.

4. MONOTONIC GENERALIZED FREQUENT DIRECTIONS (MGFD)

There are many parameter choices in GFD, each of them corresponds to an sketching algorithm. We cannot exhaustively explore all of them. In this report, we just explored one family of algorithms called **Monotonic Generalized Frequent Directions (mGFD)**. A mGFD models w as a sigmoid function $\frac{1}{1+e^{\omega(\tau-x)}}$ to reduce the number of parameters. We choose $\delta_i = H(,) = \sigma_i$ and σ_{pl} to be two of the percentiles of the sigular values. Thus, we have the following parameters in mGFD:

- $\tau' \in (0,1)$: $\tau' = \frac{\tau}{l}$ where τ is defined above as the "offset" of the sigmoid function. It is more convenient to have a ratio than an absolute value since the sketch length l may change very often.
- $\omega' \in (0,+\infty)$: $\omega' = \frac{\omega}{l}$, where ω is defined above as the "smoothness" of the sigmoid function. The lower the ω', the smoother the sigmoid function is. Again, it is more convenient to have a ratio than an absolute value. Because for a fixed ω', the function keeps the same shape across different buffer sizes l.
- $\delta_i = \sigma_i$: the amount of decrease before multiplying weights. It is the return value of $H(,)$.
- σ_{pl}: the upper bound of decrease (percentile) before multiplying the scaling factor
- $\sigma_{p_{low}l}$: the lower bound of decrease percentile (optional. It provides $O(n/l)$ bound for the number of iterations if used. It does not affect other theoretical bounds.)
- λ: the scaling factor of the upper bound.

Note that here we only explore the **monotonic increasing version of mGFD**.

We show the matlab code to get sigmoid arrays below and examples of sigmoid arrays with different lengths parameters (Appendix Figure 16 and 17). Note that we did not try monotonic decreasing sigmoid functions in our experiments.

```
function output = sigmoid_array ...
(length,steepness_ratio,shift_ratio)
array = 1:length - 1;
steepness = steepness_ratio/length;
shift = length*shift_ratio;
output = 1./( ...
    1+exp(steepness*(shift-array)) );
```

5. EXPERIMENTS

We implemented our GFD algorithm and explored the parameter space of **mGFD** on Spam, Birds and Adversarial datasets from [2] to evaluate its performance in practice.

5.1 Experimental settings

Experiments were run on a 2010 machine with an 8-core processor and 36GB RAM. The operating system is Ubuntu 12.04. The programming language is Matlab, which is very standard for linear algebra experiments. The experiments about approximation errors are not dependant on running time. The running time experiments were conducted multiple times and the average is chosen such that it is not very influenced by any programming language overhead. In fact, the code is simple and there is little overhead. And Matlab is optimized for linear algebra operations.

5.2 Datasets

We used the spams, birds and the Adversarial datasets from [2].

- Spam: Each row represents a spam message. Each column is the features of that spam message. "This dataset has has dramatic and abrupt feature drift over the stream, but not as much as Adversarial"[7].
- Birds: Each row represents an image of a bird. Each column is the features of that image.
- Adversarial[8]: in order to model dramatic change over the stream, the authors constructed two orthogonal spaces π_1 and π_2 and project random noise points onto them. All points projected to π_1 comes before the points projected to π_2 so that the points in the second half of the stream are in a very different distribution from those in the first half.

[7]Quotes from [2]
[8]The name adversarial comes from the fact that it makes iSVD very bad in [2]. It is not really "adversarial" here because our models work well on it.

5.3 Evaluation criteria

The evaluation criteria are the same as [2]. We mainly used two error measures, corresponding to the additive bound and relative bound, respectively.

- Covariance error: $||A^TA - B^TB||_2/||A||_F^2$
- Projection error: $||A - \widetilde{A_k}||_F^2/||A - A_k||_F^2$

The k is the **projection rank**. Recall that $\widetilde{A_k} = AB_k^T(B_kB_k^T)^+B_k$, where B_k is the first $k \le l$ rows of B. Instead of setting $k = 10$ as in [2], we set $k = 20$ in most of the experiments and also tried varying k with a fixed sketch size.

5.4 Tested Algorithms

1. FD: the original algorithm in [5]:
2. FD-last: the algorithm in [3]. Same as FD but reducing the squared singular values by the amount of the square of the last singular value instead of that of the 50-percentile singular value.
3. 0.2-FD: the algorithm in [2]. The best known algorithm before this report.
4. mGFD (w monotonic non-decreasing ones)

5.5 mGFD parameters

We explored the parameters space of mGFD as follows:

- Offset/shift τ': 0.1, 0.3, 0.5, 0.7, 0.9, 0.95
- Steepness ω': 1, 5, 10, 20, 50, 100, 300, 600, the higher the less smooth the curve
- $\delta_i = \sigma_i$ the amount (percentile) of decrease before multiplying weights:, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95 (e.g., 0.7 means decreasing by the 70th percentile singular value, which is exactly greater than 70% of the singular values.)
- σ_{pl}: the upper bound (percentile) of decrease before multiplying λ: 0.05, 0.1, 0.3, 0.5, 0.7, 0.9 (e.g., 0.1 means bounding the decrease by the 10th percentile singular value, which is exactly no more than 10% of the singular values.)
- $\sigma_{{p_{low}}l} = 0.01$. Lower bound of the decrease. It reduces the number of iterations but does not affect the approximation bounds.
- Scaling factor $\lambda = 1$. This means we do not scale the decrease of the singular values

This is not a very fine-grained parameter space. However we can get a rough idea of the performance of mGFD models.

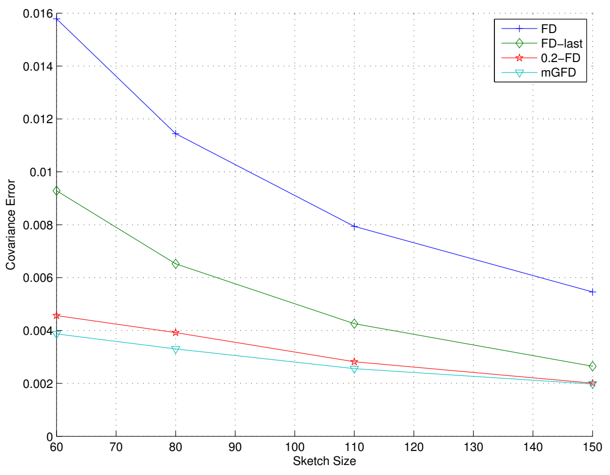

Figure 2: Results on the Spam Dataset: Covariance error vs. sketch size.

5.6 Results on Spam Dataset

5.6.1 *Scatter plot of mGFD models*

In Figure 1, we plot each mGFD in our parameter space as a point in a 2-D space, where the y-axis represents its projection error and the x-axis represents its covariance error. Each model is plotted as a number, which represents its average running time. We also plot the 0.2-FD model (the previous best model) as a solid green dot in the plot. For this plot we choose $k = 20$ and $sketch\ size = 150$. The running time of 0.2-FD 276.7716 seconds (on exactly the same machine and same settings).

There are several observations:

1. At least in our parameter space of mGFD, there are better performing ones than 0.2-FD.
2. 0.2-FD is about one order of magnitude slower than most of our mGFD models.

5.6.2 *Performance vs. sketch size*

To see the performance vs. sketch size. We arbitrarily pick a relatively good mGFD configuration (may not be the best) with parameter being $\tau' = 0.90$, $\omega' = 50$, $\sigma_i = 0.7$, $\sigma_{pl} = 0.1$. Figure 2 shows the covariance error and Figure 3 shows the projection error, compared with previous algorithms.

5.6.3 *Runtime Comparison*

With the same mGFD configuration above, we could compare the runtime of different algorithms. Notably, mGFD is more than one order of magnitude faster than 0.2-FD while having the same (even better approximation quality). This is shown in Figure 4.

5.6.4 *Projection error vs. Projection k*

With the same mGFD configuration above, we could also show in Figure 5 the projection error vs. the rank k that we are reducing the matrix to. The sketch size is 150.

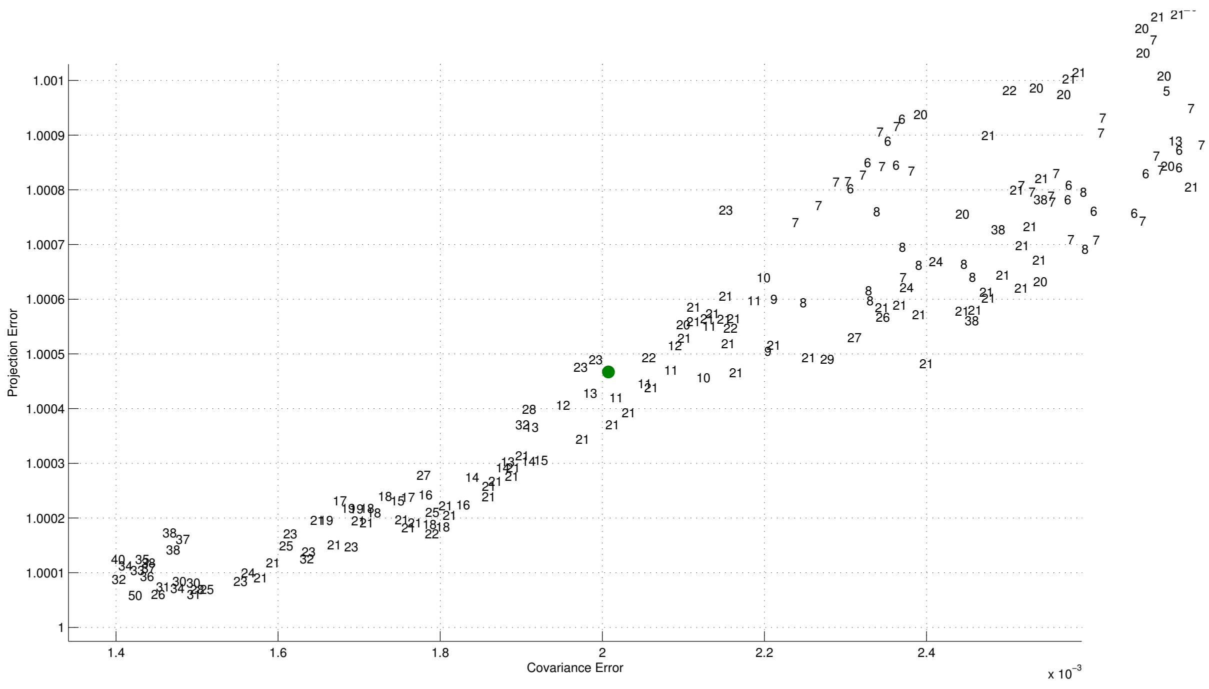

Figure 1: Results on the Spam Dataset [2]: Plot each mGFD model as a number. The y-axis represents its projection error and the x-axis represents its covariance error. The number represents its average running time in seconds. 0.2-FD (the previous best model [2]) as a solid green dot in the plot. The running time of 0.2-FD 276.7716 seconds, one order of magnitude slower than most mGFDs.

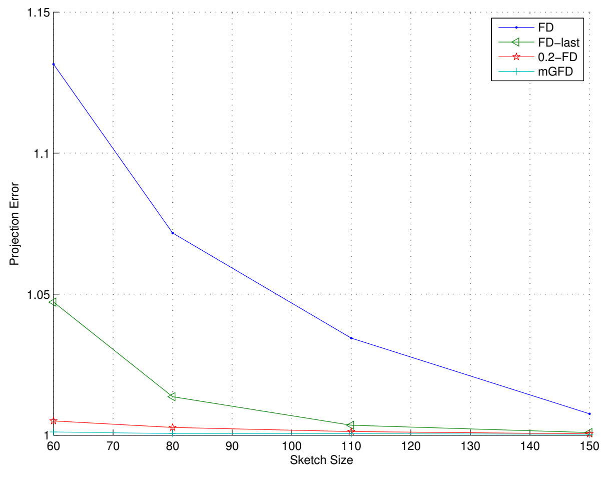

Figure 3: Results on the Spam Dataset: Projection error vs. sketch size. The projection rank $k = 20$.

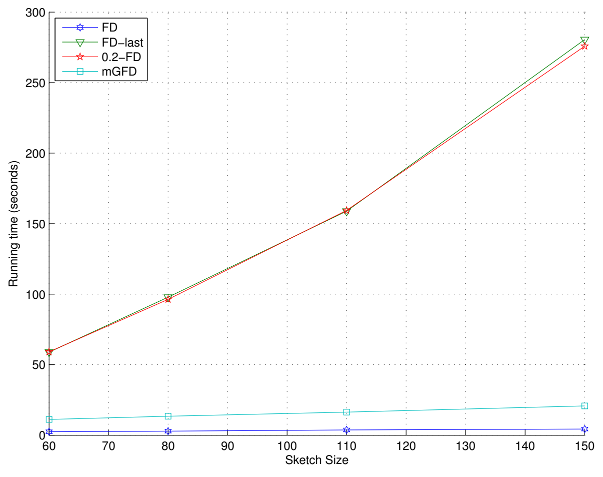

Figure 4: Results on the Spam Dataset: runtime comparison. The projection rank $k = 20$.

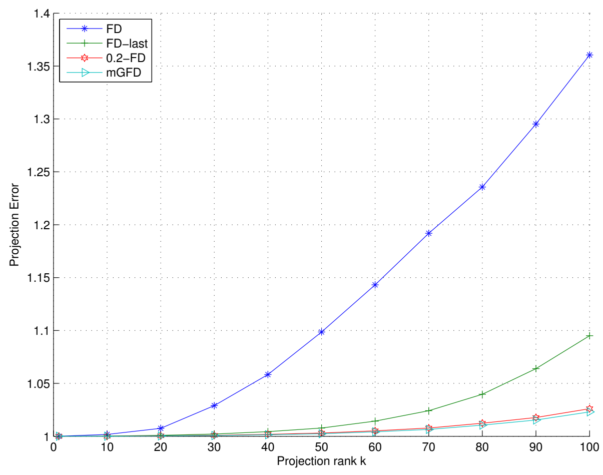

Figure 5: Results on the Spam Dataset: projection error vs. projection rank k. The sketch size is 150.

5.7 Results on Birds Dataset

5.7.1 *Scatter plot of mGFD models*

In Figure 6, we plot each mGFD in our parameter space as a point in a 2-D space, where the y-axis represents its projection error and the x-axis represents its covariance error. Each model is plotted as a number, which represents its average running time. We also plot the 0.2-FD model (the previous best model) as a solid green dot in the plot. For this plot we choose $k = 20$ and $sketch\ size = 150$. The running time of 0.2-FD 268.0580 seconds (on exactly the same machine and same settings).

There are several observations:

1. At least in our parameter space of mGFD, there are better performing ones than 0.2-FD.
2. 0.2-FD is about one order of magnitude slower than most of our mGFD models.

5.7.2 *Performance vs. sketch size*

To see the performance vs. sketch size. We arbitrarily pick a relatively good mGFD configuration (may not be the best) with parameter being $\tau' = 0.90$, $\omega' = 50$, $\sigma_i = 0.7$, $\sigma_{pl} = 0.1$. Figure 7 shows the covariance error and Figure 8 shows the projection error.

5.7.3 *Runtime Comparison*

With the same mGFD configuration above, we could compare the runtime of different algorithms. Notably, mGFD is more than one order of magnitude faster than 0.2-FD while having the same (even better approximation quality). This is shown in Figure 9.

5.7.4 *Projection error vs. Projection k*

With the same mGFD configuration above, we could also show in Figure 10 the projection error vs. the rank k that we are reducing the matrix to. The sketch size is 150.

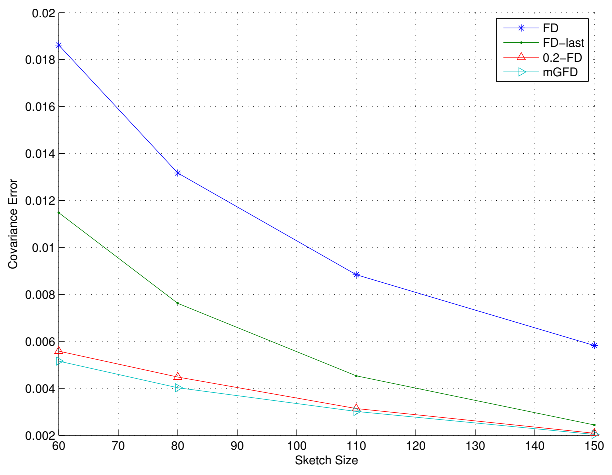

Figure 7: Results on the Birds Dataset: Covariance error vs. sketch size

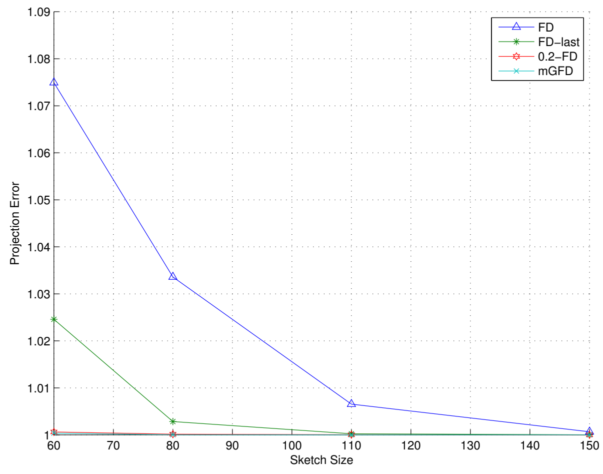

Figure 8: Results on the Birds Dataset: Projection error vs. sketch size. The projection rank $k = 20$.

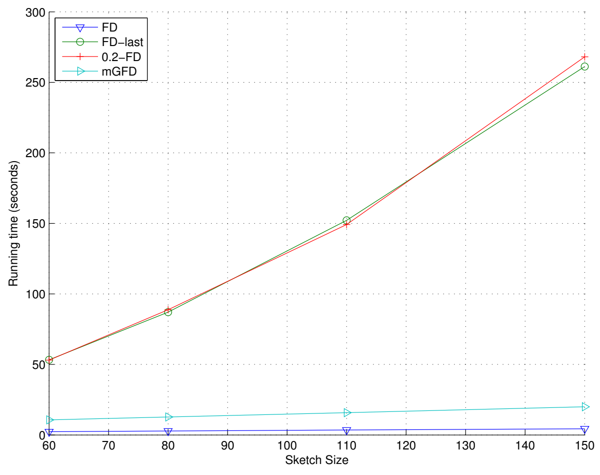

Figure 9: Results on the Birds Dataset: runtime comparison. The projection rank $k = 20$.

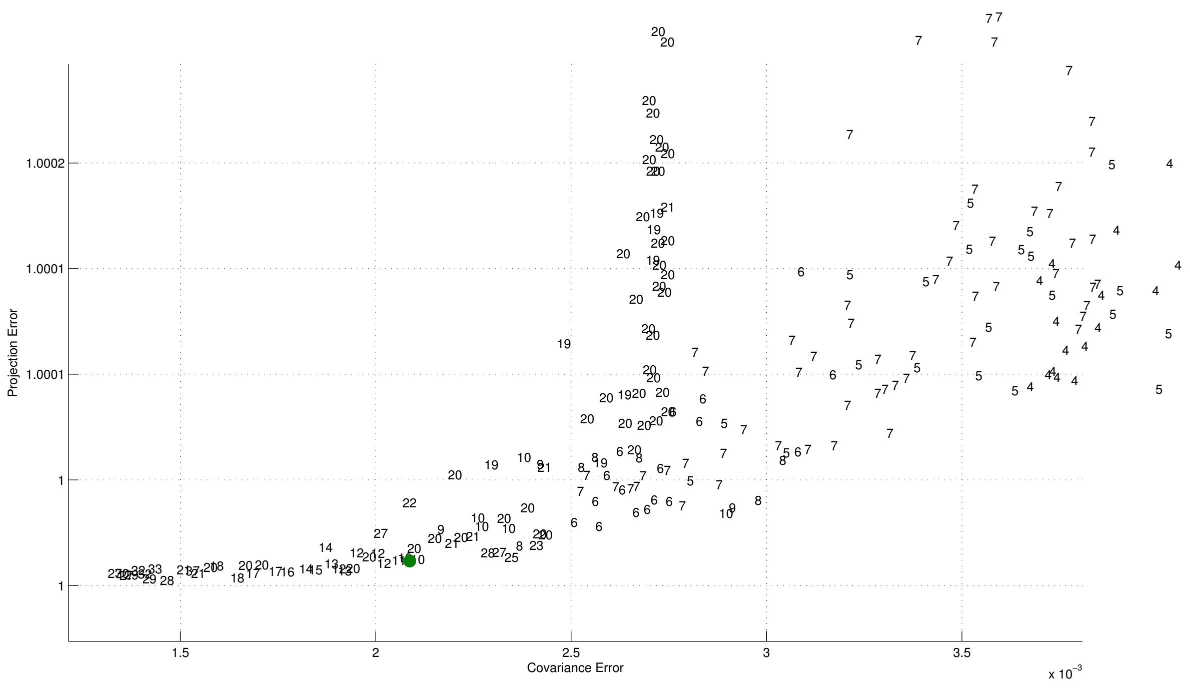

Figure 6: Results on the Birds Dataset [2]: Plot each mGFD model as a number. The y-axis represents its projection error and the x-axis represents its covariance error. The number represents its average running time in seconds. 0.2-FD (the previous best model [2]) as a solid green dot in the plot. The running time of 0.2-FD 268.0580 seconds, one order of magnitude slower than most mGFDs.

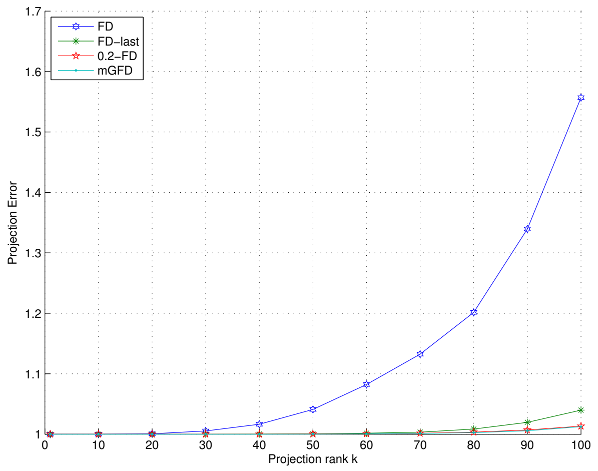

Figure 10: Results on the Birds Dataset: projection error vs. projection rank k. The sketch size is 150.

5.8 Results on Adversarial Dataset

5.8.1 *Scatter plot of mGFD models*

In Figure 11, we plot each mGFD in our parameter space as a point in a 2-D space, where the y-axis represents its projection error and the x-axis represents its covariance error. Each model is plotted as a number, which represents its average running time. We also plot the 0.2-FD model (the previous best model) as a solid green dot in the plot. For this plot we choose $k = 20$ and *sketch size* $= 150$. The running time of 0.2-FD 299.2864 seconds (on exactly the same machine and same settings).

There are several observations:

1. At least in our parameter space of mGFD, there are better performing ones than 0.2-FD.
2. 0.2-FD is about one order of magnitude slower than most of our mGFD models.

5.8.2 *Performance vs. sketch size*

To see the performance vs. sketch size. We arbitrarily pick a relatively good mGFD configuration (may not be the best) with parameter being $\tau' = 0.70$, $\omega' = 10$, $\sigma_i = 0.1$, $\sigma_{pl} = 0.1$. Figure 12 shows the covariance error and Figure 13 shows the projection error.

5.8.3 *Runtime Comparison*

With the same mGFD configuration above, we could compare the runtime of different algorithms. Notably, mGFD is

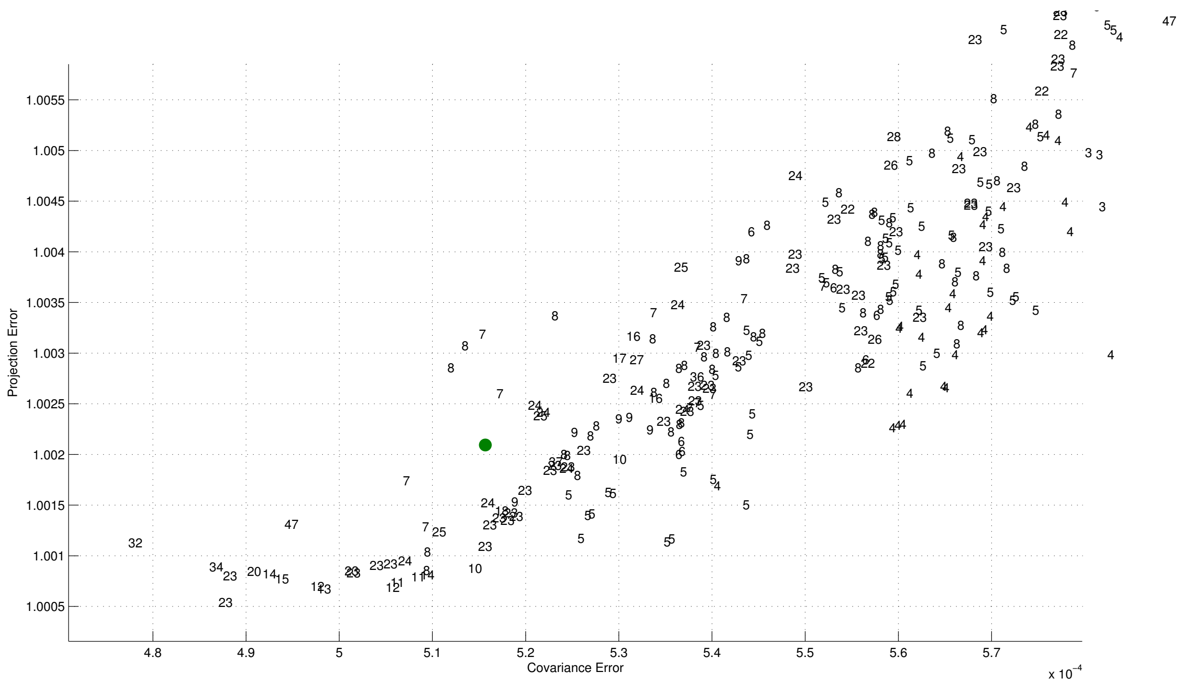

Figure 11: Results on the Adversarial Dataset [2]: Plot each mGFD model as a number. The y-axis represents its projection error and the x-axis represents its covariance error. The number represents its average running time in seconds. 0.2-FD (the previous best model [2]) as a solid green dot in the plot. The running time of 0.2-FD 299.2864 seconds, one order of magnitude slower than most mGFDs.

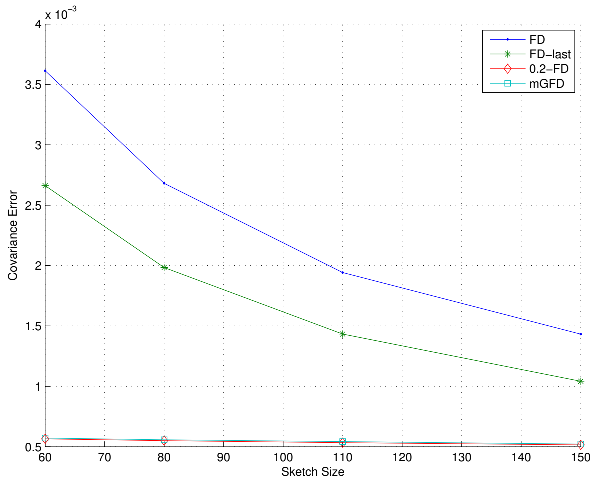

Figure 12: Results on the Adversarial Dataset: Covariance error vs. sketch size

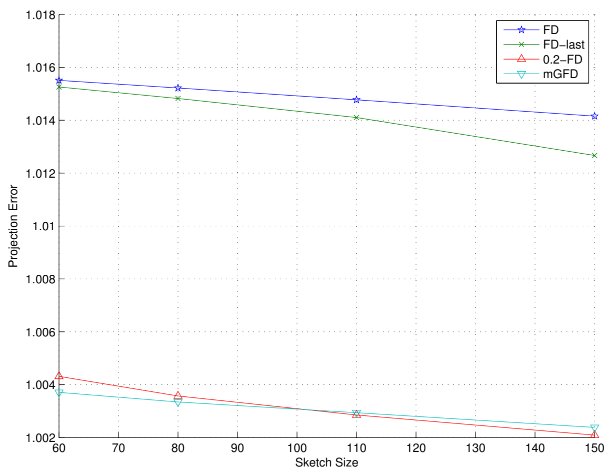

Figure 13: Results on the Adversarial Dataset: Projection error vs. sketch size. The projection rank $k = 20$.

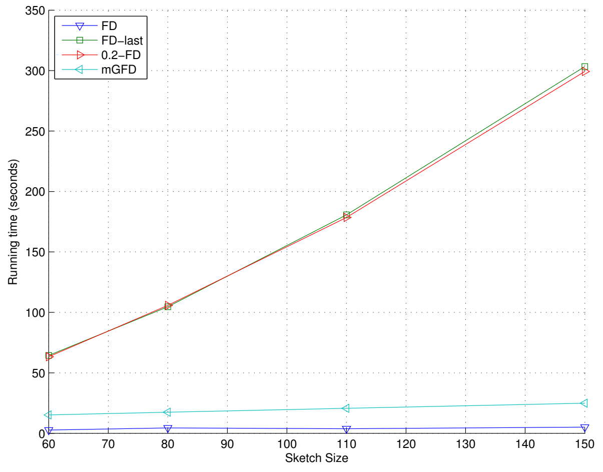

Figure 14: Results on the Adversarial Dataset: runtime comparison. The projection rank $k = 20$.

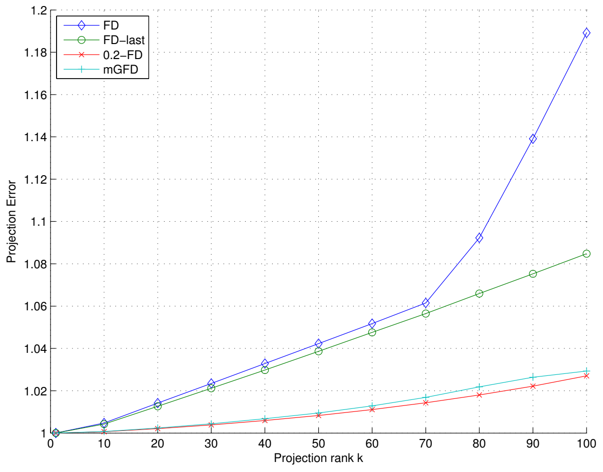

Figure 15: Results on the Adversarial Dataset: projection error vs. projection rank k. The sketch size is 150.

more than one order of magnitude faster than 0.2-FD while having the same (even better approximation quality). This is shown in Figure 14.

5.8.4 Projection error vs. Projection k

With the same mGFD configuration above, we could also show in Figure 15 the projection error vs. the rank k that we are reducing the matrix to. The sketch size is 150.

6. DISCUSSION

We mentioned that by changing **GFD parameters**, we can obtain exactly the same algorithms in [5],[3] and [2], without losing theoretical guarantees. As shown by [2] and our experiments, these algorithms have significantly different runtime, space and approximation quality in practice. Furthermore, our parameter space is large and the only configurations we have explored are the Monotonic (increasing) Frequent Directions. Efficient parameter selection based on online validations of performance would be an intriguing direction for future investigation. Evolutionary algorithms or gradient based learning could be potentially useful.

7. CONCLUSIONS

In this report, we proposed a unified framework **Generalized Frequent Directions (GFD)** that captures the results from [5],[3] and [2] as special cases, without losing any of the theoretical guarantees. We proved the related time, space and approximation quality bounds. We experimentally explored a class of GFD algorithms called Monotonic Frequent Directions (mGFD) on sythetic and real-world datasets and show that they achieves the state-of-the art approximation performance while being at least one order of magnitude faster than the previous best model.

8. ACKNOWLEDGMENTS

This research was conducted from 12/2/2014 to 12/10/2014 as a two-week project for 6.854 Advanced Algorithms. I would like to extend thanks to Professor David Karger, teaching assistant Georgios Vlachos and Will Hasenplaugh for their help and instructions throughout this semester.

9. REFERENCES

[1] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *AlgorithmsâĂŤESA 2002*, pages 348–360. Springer, 2002.
[2] M. Ghashami, A. Desai, and J. M. Phillips. Improved practical matrix sketching with guarantees. In *Algorithms-ESA 2014*, pages 467–479. Springer, 2014.
[3] M. Ghashami and J. M. Phillips. Relative errors for deterministic low-rank matrix approximations. In *SODA*, pages 707–717. SIAM, 2014.
[4] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.
[5] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.
[6] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Database Theory-ICDT 2005*, pages 398–412. Springer, 2005.
[7] J. Misra and D. Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.
[8] K. B. Petersen. The matrix cookbook.

APPENDIX

A. SINGULAR VALUE DECOMPOSITION

$$[USV] = svd(A)$$

Let us say A is of size n by d and rank(A) = r. The properties we might need to use about SVD are:

- U is a n by r matrix, containing right singular vectors
- S is a diagonal matrix of size r by r
- V is a matrix of size r by r, containing right singular vectors. V are the eigenvectors of the covariance matrix $A^T A$.
- V_k is the first k columns of V.
- AV_k is a dimensional reduction using Principle components Analyses (PCA) on A.
- $|AV_k|^2 = |US|^2 = \sum_{i=1}^{k} \sigma_i$

B. SOLVING LINEAR LEAST SQUARE PROBLEM

Task: suppose we have matrix $Y \in \mathbb{R}^{n\times d}$ and $X \in \mathbb{R}^{l\times d}$, where $l < n$, find a $W \in \mathbb{R}^{n\times l}$ such that $||Y - W^T X||_F^2$ is minimized:

$$min_W \ ||Y - W^T X||_F^2$$

Solution:

$$\begin{aligned}
&min_W \ ||Y - W^T X||_F^2 \\
&= Tr\{(Y - W^T X)^T (Y - W^T X)\} \\
&= Tr\{(Y^T - X^T W)(Y - W^T X)\} \\
&= Tr(Y^T Y - X^T W Y - Y^T W^T X + X^T W W^T X) \\
&= ||Y||_F^2 - Tr(X^T W Y) - Tr(Y^T W^T X) + Tr(X^T W W^T X) \\
&= ||Y||_F^2 - 2Tr(Y^T W^T X) + Tr(X^T W W^T X)
\end{aligned}$$

where the last transition works because $Tr(A^T) = Tr(A)$.

Take a derivative of the above equation:

$$\begin{aligned}
&\frac{\partial ||Y - W^T X||_F^2}{\partial W} \\
&= \frac{\partial ||Y||_F^2 - 2Tr(Y^T W^T X) + Tr(X^T W W^T X)}{\partial W} \\
&= \frac{\partial ||Y||_F^2 - 2Tr(X Y^T W^T) + Tr(X^T W W^T X)}{\partial W} && \textit{property } 1 \\
&= -2XY^T + \frac{\partial Tr(X^T W W^T X)}{\partial W} && \textit{property } 2 \\
&= -2XY^T + 2XX^T W && \textit{property } 3
\end{aligned}$$

property 1: $Tr(ABC) = Tr(CAB)$

property 2: $\frac{\partial Tr(AW)}{\partial W} = A^T$ (See Matrix Cookbook [8])

property 3: $\frac{\partial Tr(AXBX^T C)}{\partial W} = A^T C^T X B^T + CAXB$ (See [8])

Set the derivative to 0, we get:

$$\begin{aligned}
0 &= -2XY^T + 2XX^T W \\
XX^T W &= XY^T \\
W &= (XX^T)^{-1} XY^T
\end{aligned}$$

To account for the case where XX^T is not invertible, we can use Moore Penrose pseudoinverse $(XX^T)^+$. See Wikipeidia[9]

So the solution to the Linear Least Square problem is:

$$\begin{aligned}
W &= (XX^T)^+ XY^T \\
W^T &= YX^T (XX^T)^+
\end{aligned}$$

C. PROJECTION MATRIX

Definition: A square matrix P is a **Projection Matrix** iff $P^2 = P$.

Lemma A.1 $P = B_k^T (B_k B_k^T)^+ B_k$ is a Projection Matrix.

Proof:

$$\begin{aligned}
P^2 &= (B_k^T (B_k B_k^T)^+ B_k)(B_k^T (B_k B_k^T)^+ B_k) \\
&= B_k^T (B_k B_k^T)^+ (B_k B_k^T)(B_k B_k^T)^+ B_k \\
&= B_k^T (B_k B_k^T)^+ B_k && (B_k B_k^T)(B_k B_k^T)^+ = I \\
&= P
\end{aligned}$$

D. SIGMOID ARRAYS WITH DIFFERENT PARAMETERS

Figure 16 and 17. These functions are used as weights w. The parameters of each sigmoid arrays are shown below each plot.

[9]"A common use of the Moore Penrose pseudoinverse is to compute a 'best fit' (least squares) solution to a system of linear equations that lacks a unique solution."(Quoted from Wikipeidia Moore Penrose pseudoinverse Page)

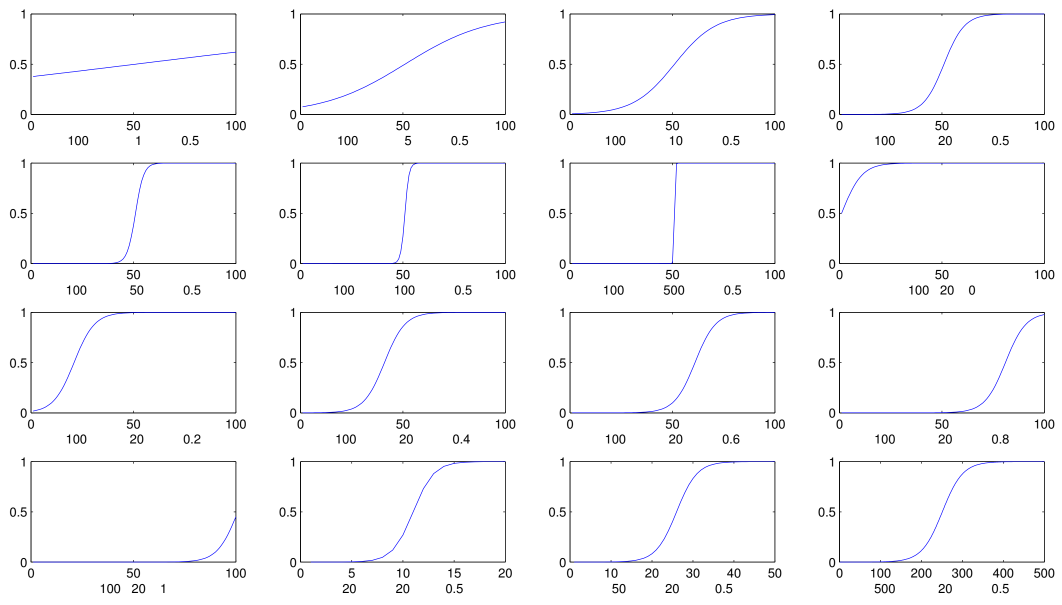

Figure 16: Examples of sigmoid functions (with parameters listed under each plot). They are monotonic increasing.

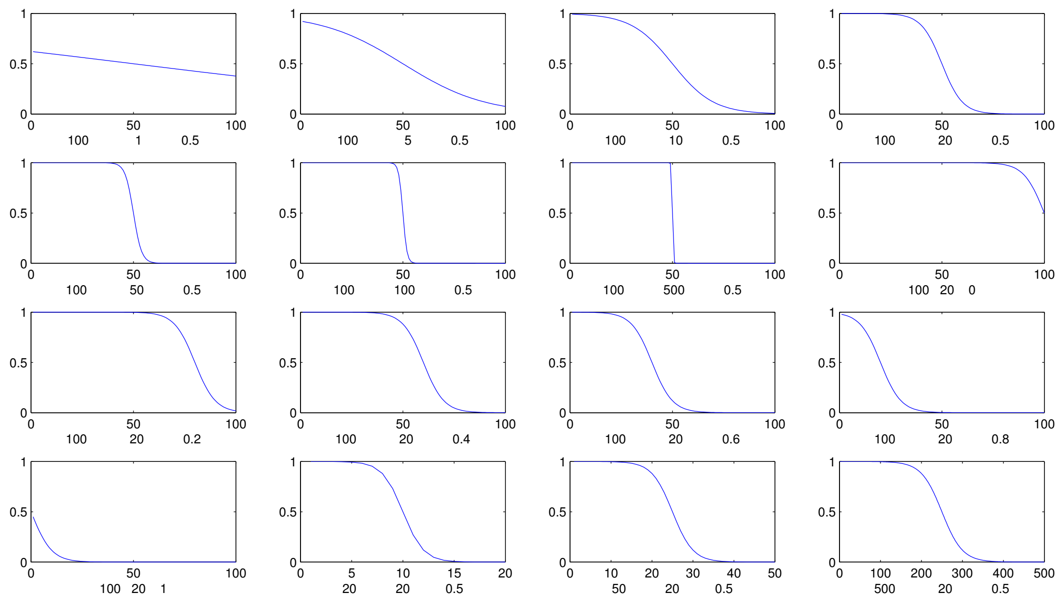

Figure 17: Examples of Flipped sigmoid functions (with parameters listed under each plot). They are monotonic decreasing. Note that we did not try monotonic decreasing sigmoid functions in our experiments.