A Preliminaries

A.1 CountSketch and Gaussian Transforms

Definition A.1 (Sparse embedding matrix or CountSketch transform)**.** *A CountSketch transform is defined to be $\Pi = \Phi D \in \mathbb{R}^{m\times n}$. Here, D is an $n\times n$ random diagonal matrix with each diagonal entry independently chosen to be $+1$ or -1 with equal probability, and $\Phi \in \{0,1\}^{m\times n}$ is an $m\times n$ binary matrix with $\Phi_{h(i),i} = 1$ and all remaining entries 0, where $h : [n] \to [m]$ is a random map such that for each $i \in [n]$, $h(i) = j$ with probability $1/m$ for each $j \in [m]$. For any matrix $A \in \mathbb{R}^{n\times d}$, ΠA can be computed in $O(\mathrm{nnz}(A))$ time.*

Definition A.2 (Gaussian matrix or Gaussian transform)**.** *Let $S = \frac{1}{\sqrt{m}} \cdot G \in \mathbb{R}^{m\times n}$ where each entry of $G \in \mathbb{R}^{m\times n}$ is chosen independently from the standard Gaussian distribution. For any matrix $A \in \mathbb{R}^{n\times d}$, SA can be computed in $O(m \cdot \mathrm{nnz}(A))$ time.*

We can combine CountSketch and Gaussian transforms to achieve the following:

Definition A.3 (CountSketch + Gaussian transform)**.** *Let $S' = S\Pi$, where $\Pi \in \mathbb{R}^{t\times n}$ is the CountSketch transform (defined in Definition A.1) and $S \in \mathbb{R}^{m\times t}$ is the Gaussian transform (defined in Definition A.2). For any matrix $A \in \mathbb{R}^{n\times d}$, $S'A$ can be computed in $O(\mathrm{nnz}(A) + dtm^{\omega-2})$ time, where ω is the matrix multiplication exponent.*

A.2 Pythagorean Theorem, matrix form

Here we state a Pythagorean Theorem for matrices.

Theorem A.4 (Pythagorean Theorem)**.** *For any integers $m, n > 0$ and matrices $A, B \in \mathbb{R}^{m\times n}$, if $\mathrm{tr}[A^\top B] = 0$, then*

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2$$

A.3 Adaptive Sampling

We described a t-round adaptive sampling algorithm. The algorithm is originally proposed in [DRVW06]. We will use $\pi_V(A)$ to denote the matrix obtained by projecting each row of A onto a linear subspace V. If V is spanned by a subset S of rows, we denote the projection of A onto V by $\pi_{\mathrm{span}(S)}(A)$. We use $\pi_{\mathrm{span}(S),k}(A)$ for the best rank-k approximation to A whose rows lie in $\mathrm{span}(S)$.

- Start with a linear subspace V. Let $E_0 = A - \pi_V(A)$ and $S = \emptyset$
- For $j = 1$ to t, do
  - Pick a sample S_j of s_j rows of A independently from the following distribution : row i is picked with probability $P_i^{(j-1)} \geq c\frac{\|E_{j-1}^{(i)}\|_2^2}{\|E_{j-1}\|_F^2}$
  - $S = S \cup S_j$
  - $E_j = A - \pi_{\mathrm{span}}(V \cup S)(A)$.

Theorem A.5 ([DRVW06], see also Theorem 3 in [DV06])**.** *After one round of the adaptive sampling procedure described above,*

$$\mathop{\mathbf{E}}_{S_1}[\|A - \pi_{span(V\cup S_1),k}(A)\|_F^2] \leq \|A - A_k\|_F^2 + \frac{k}{cs_1}\|E_0\|_F^2.$$

B Additional Results for Sketching f-Matrix Product

B.1 Proofs of Sketch $\log(|\cdot| + 1)$-Vector Product

Theorem B.1 ([Gan07], K-Set)**.** *There exists a data structure supports updates of the form (i, Δ) to a vector $v \in \mathbb{R}^n$, where $i \in [n]$ and $\Delta \in \{-1, 1\}$, and supports a query operation at any time. The algorithm either returns the current vector $v \in \mathbb{R}^n$ or "Fail". If the $\mathrm{supp}(v) \leq k$, then the data structure returns "Fail" with probability at most $\delta \in (0, 1)$. The algorithm uses space $O[k \log n \log(k/\delta)]$ bits.*

Proof of Theorem 4.2. Firstly, in the algorithm, for the level j, we sample the universe with probability $p_j = \min(\frac{\epsilon^{-2}\operatorname{poly}(\log n/\delta)}{2^j}, 1)$. Suppose the true support of x satisfies $|\operatorname{supp}(x)| = \Theta(2^j)$. We argue that with high probability, there exists an $j^* \geq j$ such that KSET_{j^*} succeeds. To show this, it is suffice to show that KSET_j succeeds with high probability. By Chernoff bound, with probability at least $1-\Theta(\delta)$, the number of coordinates sampled in level j is $\Theta(\epsilon^{-2}\operatorname{poly}(\log n/\delta))$. By Theorem B.1, the KSET_j instance succeeds to return the sampled sub vector with probability at least $1 - O(\delta)$. Since the coordinates sampled in KSET_{j^*} is with probability at least p_j, we can bound the variance of unbiased estimator by

$$
\begin{aligned}
&\mathop{\mathbf{E}}_{S_j}\left[\left(\sum_{i\in S_j} 2^j x_i \log(|y_i|+1)\right)^2\right] - \left(\mathop{\mathbf{E}}_{S_j}\left[\sum_{i\in S_j} 2^j x_i \log(|y_i|+1)\right]\right)^2 \\
=&\mathop{\mathbf{E}}_{S_j}\left[\sum_{i\in S_j}\sum_{k\in S_j}(2^j x_i \log(|y_i|+1))(2^j x_k \log(|y_k|+1))\right] - \left(\sum_{i=1}^n \Pr[i\in S_j] 2^j x_i \log(|y_i|+1)\right)^2 \\
=&\sum_{i=1}^n\sum_{k=1}^n \Pr[i\in S_j, k\in S_j](2^j x_i \log(|y_i|+1))(2^j x_k \log(|y_k|+1)) - \left(\sum_{i=1}^n \Pr[i\in S_j] 2^j x_i \log(|y_i|+1)\right)^2 \\
=&\sum_{i=1}^n \Pr[i\in S_j] 2^{2j} x_i^2 \log^2(|y_i|+1) - \sum_{i=1}^n \Pr[i\in S_j]^2 2^{2j} x_i^2 \log^2((|y_i|+1)) \\
\leq&\sum_{i=1}^n 2^j x_i^2 \log^2(|y_i|+1) \\
\leq&\max_{i\in[n]}\left(2^j x_i^2 \log(|y_i|+1)\right)\cdot\sum_{i=1}^n \log(|y_i|+1) \\
\leq&\max_{i\in[n]} x_i^2 \cdot \max_{i\in[n]}\left(2^j \log(|y_i|+1)\right)\cdot\sum_{i=1}^n \log(|y_i|+1) \\
\leq&\|x\|_\infty^2 \cdot (\sum_{i=1}^n \log(|y_i|+1))^2 \cdot \log m
\end{aligned}
$$

where the first step uses the fact

$$
\mathop{\mathbf{E}}_{S_j}\left[\sum_{i\in S_j} 2^j x_i \log(|y_i|+1)\right] = \sum_{i=1}^n \Pr[i\in S_j] 2^j x_i \log(|y_i|+1),
$$

the second step expands the square, the fourth step uses $\Pr[i\in S_j] = 2^{-j}$, the fifth step uses the fact that $\sum_i a_i b_i \leq (\max_i a_i)\cdot\sum_i b_i$ for $b_i \geq 0$, and the last step uses the fact that

$$
\max_{i\in[n]}\left(2^j \log(|y_i|+1)\right) \leq 2^j \cdot \log m \cdot \min_{i\in[n]} \log(|y_i|+1) \leq \log m \cdot \sum_{i=1}^n \log(|y_i|+1)
$$

Applying Bernstein's inequality, we conclude the proof. □

B.2 Sketch $\sqrt{|\cdot|}$-Vector Product

Our algorithm for sketching $\sqrt{|\cdot|}$-Vector product is based on the algorithm established in [BVWY17]. The algorithm is formally presented in Algorithm 3. We first present an algorithm that approximates the inner product for only non-negative x. In the theorem, we will show the inner product for general x can be approximated as well. The high level idea is similar to the p-stable distribution algorithm established in [Ind00]. However this algorithm is much simpler in terms of hashing function chosen and distribution design. In this algorithm, we used the distribution called p-inverse distribution ([BVWY17]) over positive integers such that $\Pr[X \leq z] = 1 - 1/z^p$, where X is the p-inverse random variable. Then we scale each coordinate of $|x|^{1/p}y$ by a random variable drawn

Algorithm 3 PolySum(x, p, ϵ)

procedure Initialize(x, p) ▷ $x \geq 0$, $z \in \mathbb{Z}_{\geq 0}, 0 < p \leq 2$
  Let p-inverse distribution be defined as $\Pr[z < x] = 1 - \frac{1}{x^p}$
  Let $\mathcal{D}$ denote the pairwise independent p-inverse distribution.
  Let $k \leftarrow \Theta(\epsilon^{-2})$
  Implicitly store $n \times k$ matrix Z, where $Z_{i,j} \sim \mathcal{D}$ ▷ Only needs $\Theta(\log n)$ bits
  Initialize CS as a count-sketch instance with space $\Theta(\epsilon^{-2} \operatorname{poly} \log n)$
end procedure
procedure Update(i, Δ) ▷ $i \in [n], \Delta \in \mathbb{R}$
  for $j = 1 \to k$ **do**
    CS.Update$((i, j), |x_i|^{1/p} Z_{i,j} \cdot \Delta)$
  end for
end procedure
procedure Query
  $\widetilde{y} \leftarrow$CS.Query()
  $z \leftarrow (\frac{k}{2})$-th largest coordinates of $|\widetilde{(y)}|$
  return $z/2^{1/p}$
end procedure

from the p-inverse distribution. After this, we run a count-sketch to find the largest few coordinates in the updating scaled vector. It can be shown that the median value of these output coordinates serve as a good estimation for the p-norm of the vector $|x|^{1/p}y$. A similar idea of this kind can be found in [And17]. For the $\sqrt{\cdot}$-case, we simply chose $p = 1/2$ then $\|y\|_p^p$ is a good estimation to $\sum_i |x_i| \sqrt{|y_i|}$.

Theorem B.2. *Given a fixed vector $x \in \mathbb{R}^n$ and number $p \in (0, 2]$. There exists an one-pass streaming algorithm that makes a single pass over the stream updates to an underlying vector $y \in \mathbb{R}^n$, and outputs a number Z, such that, with probability at least $1 - \delta$,*

$$|Z - \langle x, y^p \rangle| \leq \epsilon \sum_{i=1}^n |x_i||y_i|^p.$$

The algorithm uses space $O(\epsilon^{-2} \operatorname{poly}(\log(n/\delta)))$ (excluding the space of x).

Proof. The proof of the this theorem is a straightforward application of the results in [BVWY17] by splitting x into positive and negative parts. □

B.3 More General Functions f

Furthermore, our framework can be applied to a more general set of functions. This set of function includes nearly all "nice" functions for n variables. For the ease of representation, we neglect the formal definition of the this set. It can be understood that a function in this set satisfies three properties: slow-jumping, slow-dropping and predictable. Readers that are interested, please refer to [BCWY16]. Here we give three examples for the the functions that we are able to approximate. For example, $x^2 \cdot 2^{\sqrt{\log x}}, (2 + \sin x)x^2, 1/\log(1 + x)$. Using our proposed general framework and [BCWY16], we have the following result,

Theorem B.3. *Given a vector $x \in \{-1, 0, 1\}^n$, and a function f that satisfies the above regularity condition, then there exists a one-pass streaming algorithm that makes a single pass over the stream updates to an underlying vector $y \in \mathbb{R}^n$, and outputs a number Z, such that, with probability at least $1 - \delta$,*

$$|Z - \langle x, f(|y|) \rangle| \leq \epsilon \sum_{i=1}^n f(|y_i|).$$

The algorithm uses space $O(\operatorname{poly}(\epsilon^{-1} \log(n/\delta)))$ (excluding the space of x).

Proof. The proof is a straightforward application of [BCWY16] by considering the positive part and negative part of x separately. □

Remark B.4. *We remark that the algorithm in [BCWY16] is quite complicated but has the potential to be simplified. We also note that x is not necessarily restricted on $\{-1, 0, -1\}$, but the complexity depends on ratio of the absolute values of the maximum non-zero entry and minimum non-zero entry (in absolute value) of x.*

B.4 From Vector Product Sketch to Matrix Product Sketch

With the f-vector product sketch tools established, we are now ready to present the result for sketching the matrix product, $M = f(A)B$. Notice that each entry $M_{i,j} := \langle f(A_i), B_j \rangle$ is an inner product. Thus our algorithm for the matrix sketch is simply maintaining a f-vector product sketch for each $M_{i,j}$. In our algorithm, we assume that matrix B is given, i.e., hardwired in the algorithm. Thus, if $B \in \mathbb{R}^{n\times k}$ for some $k \ll n$, we only need to keep up to $\widetilde{O}(nk)$ inner product sketches, which cost in total $\widetilde{O}(nk)$ words of space. For the ease of representation, we present our guarantee for matrix product for $f(x) := \log^c(|x|)$ for some c or for $f(x) = x^p$ for $0 \le p \le 2$, and for matrix $B \in \{-1, 0, 1\}^{n\times k}$. Our results can be generalized to a more general set of functions and matrix B using the results presented in Section B.3.

Theorem B.5. *Given a matrix $B \in \{-1, 0, 1\}^{n\times k}$, and a function $f(x) := \log^c(|x|)$ for some c or $f(x) := |x|^p$ for some $0 \le p \le 2$, then there exists a one-pass streaming algorithm that makes a single pass over the stream updates to an underlying matrix $A \in \mathbb{R}^n$ with updates of absolute value at least 1[2] and outputs a matrix $\widehat{M}$, such that, with probability at least $1 - \delta$, for all i, j,*

$$|\widehat{M}_{i,j} - M_{i,j}| \le \epsilon \sum_{k=1}^{n} f(|A_{i,k}|).$$

The algorithm uses space $O(\epsilon^{-2} nk \operatorname{poly}(\log(n/\delta)))$.

Proof. The proof of this theorem is a straightforward application of Theorem 4.2 and Theorem B.2. □

C Application in Low Rank Approximations

C.1 Leverage score and its application on samping

Classic approaches of low rank approximation first compute the leverage scores of the matrix M, and then sample rows of M based these scores.

Definition C.1 (Leverage scores, [Woo14, BSS12])**.** *Let $U \in \mathbb{R}^{n\times k}$ have orthonormal columns with $n \ge k$. We will use the notation $p_i = u_i^2/k$, where $u_i^2 = \|e_i^\top U\|_2^2$ is referred to as the i-th leverage score of U.*

Definition C.2 (Leverage score sampling, [Woo14, BSS12])**.** *Given $A \in \mathbb{R}^{n\times d}$ with rank k, let $U \in \mathbb{R}^{n\times k}$ be an orthonormal basis of the column span of A, and for each i let p_i be the squared row norm of the i-th row of U. Let $k \cdot p_i$ denote the i-th leverage score of U. Let $\beta > 0$ be a constant and $q = (q_1, \cdots, q_n)$ denote a distribution such that, for each $i \in [n]$, $q_i \ge \beta p_i$. Let s be a parameter. Construct an $n \times s$ sampling matrix B and an $s \times s$ rescaling matrix D as follows. Initially, $B = 0^{n\times s}$ and $D = 0^{s\times s}$. For the same column index j of B and of D, independently, and with replacement, pick a row index $i \in [n]$ with probability q_i, and set $B_{i,j} = 1$ and $D_{j,j} = 1/\sqrt{q_i s}$. We denote this procedure* Leverage score sampling *according to the matrix A.*

However approximating these scores is highly non-trivial, especially in the streaming setting. Fortunately, it suffices to compute the so-called *generalized leverage scores*, i.e., the leverage scores of a proxy matrix. We describe the resulting algorithm (Algorithm 2) and the intuition here and provide the complete analysis later.

Definition C.3 (generalized leverage score)**.** *Consider two accuracy parameters $\alpha \in (0, 1), \delta \in (0, 1)$, and two positive integers q and k with $q \ge k$. If there is a matrix $E \in \mathbb{R}^{n\times n}$ with rank q and that approximates the row space of $M \in \mathbb{R}^{n\times n}$ as follows,*

$$\exists X, \|XE - M\|_F \le (1 + \alpha)\|M - [M]_k\|_F + \delta,$$

[2] This gurantees that if $A_{i,j} \neq 0$, then $|A_{i,j}| \ge 1$

then the leverage scores of E are called a set of $(1+\alpha, \delta, q, k)$-generalized leverage scores of A. Suppose E has an SVD decomposition $U\Sigma V^\top$, where $U \in \mathbb{R}^{n\times q}, V \in \mathbb{R}^{n\times q}$ are orthonormal matrices, then its leverage scores are $\ell_i = \|V^i\|_2^2$ where V^i is the i-th row of V, $\forall i \in [n]$.

These scores can be computed easier. We first need to find such an matrix E. Let S be a *subspace embedding matrix* (i.e.,

$$\|SMx\|_2 \in (1 \pm \alpha)\|Mx\|_2, \quad \forall x \in \mathbb{R}^n,$$

a sufficient large matrix with random $+1, -1$ entries will have this property).

Then $E = SM$ satisfies the requirement in Definition C.3, and thus we can simply use our sketching method to approximate SM and then compute its leverage scores. In Algorithm 2, we will use the concatenation of the positive and negative parts of S, since it also satisfies the requirement and empirically has better accuracy than S. The quality of the generalized scores (i.e., α and δ) will depend on the parameter s in the algorithm that are specified in our final Theorem 5.1.

The scores then can be used for sampling. Let P be a set of columns of M sampled based on these scores (defined in Line 11 of Algorithm 2). It is known that, when the scores are $(O(1), 0, q, k)$-generalized leverage scores, then the span of a P with $\Omega(q \log q)$ columns will contain a rank-q matrix which provides a $O(1)$-approximation to M [DMIMW12, BSS12, BW14, CEM+15, SWZ19b]. It is tempting to set $q = k$ to match our final goal of rank-k approximation, but all existing fast methods require $q > k$. To improve the rank-q to rank-k, we use *adaptive sampling*.

Adaptive sampling samples some extra columns from M according to their squared distances to the span of P. For a column M_{*i}, we thus need to use our sketching method to estimate $\|M_{*i}\|_2^2 - \|\Gamma_{*i}\|_2^2$, where Γ_{*i} is its projection on to the span of P. This introduces some additive errors but they can be handled by thresholding. Let $\widetilde{Y}$ be the sampled columns. Adaptive sampling ensures that there is a good rank-k approximation in the span of $Y := \widetilde{Y} \cup P$ as long as we have sampled sufficiently many columns. To obtain our final rank-k approximation, it suffices to project M to the span of Y and compute the top k singular vectors. The projection can be done by sketching and the errors are, again, small.

C.2 Proof of Theorem 5.1

Recall that there are three steps in computing the top singular vectors (see Algorithm 2):

- Compute the generalized leverage scores and sample a set P according to the scores,
- Adaptive sampling to get a set Y,
- Project to the span of Y and compute the approximation solution there.

Below we present the complete proofs for each step.

For simplicity, we use the following notion.

Definition C.4. *We say that the span of P has a $(1+\epsilon, \Delta)$-approximation subspace for M if there exists C such that*

$$\|PC - M\|_F \leq (1+\epsilon)\|M - [M]_k\|_F + \Delta.$$

C.3 Sampling by Generalized Leverage Scores

First, recall the definition of generalized leverage scores and related property from [BLS+16].

Lemma C.5 (Lemma 2 in [BLS+16])**.** *Suppose $0 < k \leq q \leq m \leq n$, $\alpha > 0$, $\Delta > 0$, and $A \in \mathbb{R}^{n\times n}$. Let $B \in \mathbb{R}^{n\times b}$ be $b = O(\alpha^{-2} q \log q)$ columns sampled from A according to a set of $(1+\alpha, \Delta, q, k)$-generalized leverage scores of A. Then with probability ≥ 0.99, the col-span of $B \in \mathbb{R}^{n\times b}$ has a rank-q $(1+2\alpha, 2\Delta)$-approximation subspace for A. That is, there exists $C \in \mathbb{R}^{b\times n}$ such that*

$$\|BC - A\|_F \leq (1+2\alpha)\|A - [A]_k\|_F + 2\Delta.$$

We need the following result about subspace embedding.

Lemma C.6 (Lemma 3 in [BLS+16]). *We say S is a $(1+\epsilon, \Delta)$-good subspace embedding if it satisfies the following.*
(Subspace Embedding). For any orthonormal $V \in \mathbb{R}^{n \times k}$ (i.e. $V^\top V = I$),

$$(1-c)\|Vx\| \leq \|SVx\| \leq (1+c)\|Vx\|$$

where $c \in (0,1)$ is a sufficiently small constant.
(Approximate Matrix Product). For any fixed $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times k}$

$$\|A^\top S^\top SB - A^\top B\|_F^2 \leq \frac{\epsilon}{k}\|A\|_F^2 \cdot \|B\|_F^2 + \Delta.$$

We are going to show that in Algorithm 2, the span of P has a good approximation subspace. Intuitively, $E = R \cdot M \in \mathbb{R}^{2s \times n}$ approximates the row space of $M \in \mathbb{R}^{n \times n}$ and $\widetilde{E} \in \mathbb{R}^{2s \times n}$ approximates $E \in \mathbb{R}^{2s \times n}$, so by the definition, the leverage scores $\{\ell_i\}$ of $\widetilde{E} \in \mathbb{R}^{2s \times n}$ are the generalized leverage scores of M. Then the conclusion follows from Lemma C.5. Formally, we have the following lemma.

Lemma C.7 (Sampling leverage scores). *Let $s = O(k \log k)$. Let $d_1 = O(k \log^2 k)$. Recall that $P \in \mathbb{R}^{n \times d_1}$ is the matrix sampled with the leverage score of $(S \cdot M) \in \mathbb{R}^{s \times n}$, as constructed in Line 11 as in Algorithm 2.*

There exists matrix $S \in \mathbb{R}^{s \times n}$, such that there exists C satisfying

$$\|PC - M\|_F^2 \leq 5\|M - [M]_k\|_F^2 + \Delta_1.$$

where $\Delta_1 = O(\epsilon^2/s)\|M\|_{1,2}^2$.

Proof. First, s is large enough so that $S \in \mathbb{R}^{s \times n}$ is a 0.1-subspace embedding matrix for subspace of dimension k; see [BLS+16, Woo14]. Then it is known that there exists $Z \in \mathbb{R}^{n \times s}$ satisfying

$$\|ZSM - M\|_F \leq (1+0.1)\|M - [M]_k\|_F.$$

Clearly, there exists $X = [Z, Z] \in \mathbb{R}^{n \times 2s}$ such that

$$\|XRM - M\|_F \leq (1+0.1)\|M - [M]_k\|_F. \tag{2}$$

Let $E = RM \in \mathbb{R}^{2s \times n}$. Then

$$\begin{aligned}\|M - X\widetilde{E}\|_F^2 &\leq 2\|M - XE\|_F^2 + 2\|XE - X\widetilde{E}\|_F^2 \\ &\leq 3\|M - [M]_k\|_F^2 + 2\|XE - X\widetilde{E}\|_F^2.\end{aligned}$$

where the first step follows from the inequality $\|A+B\|_F^2 \leq (\|A\|_F + \|B\|_F)^2 \leq 2\|A\|_F^2 + 2\|B\|_F^2$, the second step follows from Eq. (2) and the definition $E = RM \in \mathbb{R}^{2s \times n}$.

Consider the second term.

$$\begin{aligned}\|XE - X\widetilde{E}\|_F &\leq \|X\|_2\|E - \widetilde{E}\|_F \\ &\leq 2\|Z\|_2\|E - \widetilde{E}\|_F.\end{aligned}$$

where the last step we use $X = [Z, Z] \in \mathbb{R}^{n \times 2s}$.

Hence we have

$$\begin{aligned}\|E - \widetilde{E}\|_F &\leq O(\epsilon) \max_{i \in [2s], j \in [n]} |R_{i,j}| \|M\|_{1,2} \\ &\leq O(\epsilon/\sqrt{s})\|M\|_{1,2}.\end{aligned}$$

where the first step follows from our guarantee on our sketching method in Theorem 4.2, the second step follows from the construction of R, i.e. the range of each entry of the CountSketch matrix.

By Lemma C.6, we can rewrite Z as follows:

$$\begin{aligned} Z &= [M]_k (S[M]_k)^\dagger \\ &= [M]_k [M]_k^\dagger S^\dagger, \end{aligned}$$

We can upper bound $\|Z\|_2$ by $O(1)$,

$$\begin{aligned} \|Z\|_2 &\leq \|[M]_k [M]_k^\dagger\|_2 \cdot \|S^\dagger\|_2 \\ &= \|S^\dagger\|_2 \\ &= O(1). \end{aligned}$$

Putting it all together, we have

$$\|M - X\widetilde{E}\|_F^2 \leq 3\|M - [M]_k\|_F^2 + O(\epsilon^2/s)\|M\|_{1,2}^2.$$

This satisfies the definition of generalized leverage scores. Then the statement follows from Lemma C.5. □

C.4 Adaptive Sampling

Lemma C.8 (Adaptive)**.** *Let $d_1 = O(k \log^2 k)$ and $d_2 = O(k/\epsilon)$. With probability ≥ 0.99, there exists $C \in \mathbb{R}^{(d_1+d_2)\times n}$ such that YC is rank-k and*

$$\|YC - M\|_F^2 \leq 5\|M - [M]_k\|_F^2 + \Delta_1 + \Delta_2$$

where Δ_1 is defined as Lemma C.7, and $\Delta_2 = O(\epsilon\sqrt{d_1} + \epsilon^2 d_1)\|M\|_F^2$.

Proof. If p_i's are larger than a constant times the true square distances s_i's, then the statement follows from Theorem A.5. So consider the difference between $\widetilde{s}_i$ and s_i.

Let $\Gamma = Q_p^\top M \in \mathbb{R}^{d_1 \times n}$ where Q_p is obtained from QR-decomposition as in Line 13 of Algorithm 2.

$$|\widetilde{s}_i - s_i| = \left| \|\Gamma_i\|_2^2 - \|\widetilde{\Gamma}_i\|_2^2 + \|\widetilde{M_i}\|_2^2 - \|M_i\|_2^2 \right|.$$

By our guarantee in Theorem 4.2,

$$|\Gamma_{ji} - \widetilde{\Gamma}_{ji}| \leq \epsilon \|(Q_p)_j\|_F \|M_i\|_F = \epsilon \|M_i\|_2 \tag{3}$$

where the last inequality follows since $(Q_p)_j$'s are basis vectors and have length 1.

So

$$\begin{aligned} \left|\Gamma_{ji}^2 - \widetilde{\Gamma}_{ji}^2\right| &= \left|(\Gamma_{ji} - \widetilde{\Gamma}_{ji}) \cdot (\Gamma_{ji} + \widetilde{\Gamma}_{ji})\right| \\ &= \left|\Gamma_{ji} - \widetilde{\Gamma}_{ji}\right| \cdot \left|\Gamma_{ji} + \widetilde{\Gamma}_{ji}\right| \\ &\leq \left|\Gamma_{ji} - \widetilde{\Gamma}_{ji}\right| \cdot \left(\left|\Gamma_{ji} - \widetilde{\Gamma}_{ji}\right| + 2|\Gamma_{ji}|\right) \\ &= 2\left|\Gamma_{ji} - \widetilde{\Gamma}_{ji}\right| \cdot |\Gamma_{ji}| + \left|\Gamma_{ji} - \widetilde{\Gamma}_{ji}\right|^2 \\ &\leq 2\epsilon \|M_i\|_2 |\Gamma_{ji}| + \epsilon^2 \|M_i\|_2^2 \end{aligned} \tag{4}$$

where the third step follows from triangle inequality, and the last step follows Eq. (3).

And

$$\begin{aligned} \left| \|\Gamma_i\|_2^2 - \|\widetilde{\Gamma}_i\|_2^2 \right| &\leq \sum_{j \in [d_1]} |\Gamma_{ji}^2 - \widetilde{\Gamma}_{ji}^2| \\ &\leq 2\epsilon \|M_i\|_2 \sum_{j \in [d_1]} |\Gamma_{ji}| + O(\epsilon^2 d_1)\|M_i\|_2^2 \\ &\leq O(\epsilon\sqrt{d_1})\|M_i\|_2 \|\Gamma_i\|_2 + O(\epsilon^2 d_1)\|M_i\|_2^2 \\ &\leq O(\epsilon\sqrt{d_1})\|M_i\|_2^2 + O(\epsilon^2 d_1)\|M_i\|_2^2 \\ &= O(\epsilon\sqrt{d_1} + \epsilon^2 d_1)\|M_i\|_2^2. \end{aligned}$$

where the first step follows from triangle inequality, the second step follows from (4), the third step follows form Cauchy-Swartz inequality, the fourth step follows $\Gamma_i = Q_p^\top M_i \in \mathbb{R}^{d_1}$ and $Q_p \in \mathbb{R}^{d_1 \times n}$ is an orthonormal matrix.

Therefore,

$$|\widetilde{s}_i - s_i| \leq O(\epsilon\sqrt{d_1} + \epsilon^2 d_1)\|M_i\|_2^2 := \delta_i.$$

Suppose that the algorithm sets $\eta := O(\epsilon\sqrt{d_1} + \epsilon^2 d_1)$ such that the threshold $\delta_i \leq \eta\widetilde{z}_i \leq 2\delta_i$.

If $\widetilde{s}_i \leq \delta_i$, then $s_i \leq 2\delta_i$, and thus $p_i \geq s_i/2$. If $\widetilde{s}_i > \delta_i$, then $s_i \leq \widetilde{s}_i + \delta_i \leq 2\widetilde{s}_i \leq 2p_i$, and thus $p_i \geq s_i/2$. Now, if $\sum_i p_i$ is not too large compared to $\sum_i s_i$, then we are done by applying Theorem A.5.

Let $C \subseteq [n]$ denote the set of indices i such that $\widetilde{s}_i \geq \delta_i$. If $\sum_{i\in C} \widetilde{s}_i \geq 2\sum_{i\in[n]} \delta_i$, then

$$\sum_{i\in C} s_i \geq \sum_{i\in C} \widetilde{s}_i - \sum_{i\in[n]} \delta_i \geq \frac{1}{2}\sum_{i\in C} \widetilde{s}_i \tag{5}$$

and thus

$$\begin{aligned}
\sum_{i\in[n]} p_i \leq & \sum_{i\in C} \widetilde{s}_i + 2\sum_{i\in[n]} \delta_i \\
& \leq 2\sum_{i\in C} \widetilde{s}_i \\
& \leq 4\sum_{i\in C} s_i \\
& \leq 4\sum_{i\in[n]} s_i.
\end{aligned}$$

where the first step follows from the definition of p_i, i.e. $p_i = \max\{\widetilde{s}_i, \eta\widetilde{z}_i\}$ and the assumption $\eta\widetilde{z}_i \leq 2\delta_i$, the second step follows from the assumption $\sum_{i\in C} \widetilde{s}_i \geq 2\sum_{i\in[n]} \delta_i$, the third step uses (5), the fourth step is because $s_i \geq 0$ and $C \subset [n]$.

So we are done in this case.

In the other case when $\sum_{i\in C} \widetilde{s}_i < 2\sum_{i\in[n]} \delta_i$, we have

$$\begin{aligned}
\sum_{i\in[n]} s_i \leq & \sum_{i\in[n]} \widetilde{s}_i + \delta_i \\
& \leq \sum_{i\in C} 2s_i + \sum_{i\notin C} 2\delta_i \\
& \leq 4\sum_{i\in[n]} \delta_i \\
& = O(\epsilon\sqrt{d_1} + \epsilon^2 d_1)\|M\|_F^2 := \Delta_2.
\end{aligned}$$

where the first step follows from $\delta_i = |s_i - \widetilde{s}_i|$ and triangle inequality, the second step uses the construction of the set C and the third step uses the assumption $\sum_{i\in C} \widetilde{s}_i < 2\sum_{i\in[n]} \delta_i$.

This means that Γ is close to M, and thus $[\Gamma]_k$ (the best rank-k approximation to Γ) will be the desired approximation in the span of P (and thus the span of Y since $P \subseteq Y$). This completes the proof. □

C.5 Computing Approximation Solutions

Lemma C.9. *Let $d_1 = O(k\log^2 k)$ and $d_2 = O(k/\epsilon)$. There is an algorithm that outputs a matrix L , such that*

$$\|LL^\top M - M\|_F^2 \leq 10\|M - [M]_k\|_F^2 + 2\Delta_1 + 2\Delta_2 + \Delta_3$$

where Δ_1 is defined as Lemma C.7, Δ_2 is defined as Lemma C.8 and $\Delta_3 = O(\epsilon^2(d_1 + d_2))\|M\|_F^2$.

Proof. Since $Q \in \mathbb{R}^{n \times (d_1+d_2)}$ is orthonormal, $Q^\top Q = I_{d_1+d_2}$. We need the following auxiliary result: for any $A \in \mathbb{R}^{(d_1+d_2)\times n}$,

$$\|QAM - M\|_F^2 = \|QAM - QQ^\top M\|_F^2 + \|M - QQ^\top M\|_F^2. \tag{6}$$

This is because

$$\begin{aligned}
&\left(QAM - QQ^\top M\right)^\top \left(M - QQ^\top M\right) \\
=& M^\top A^\top Q^\top M - M^\top QQ^\top M - M^\top A^\top Q^\top QQ^\top M + MQQ^\top QQ^\top M \\
=& M^\top A^\top Q^\top M - M^\top QQ^\top M - M^\top A^\top Q^\top M + MQQ^\top M \\
=& 0
\end{aligned}$$

where the second step uses $Q^\top Q = I_{d_1+d_2}$.

Then (6) simply follows from Theorem A.4.

We also need the following result: for any $A \in \mathbb{R}^{(d_1+d_2)\times n}$,

$$\|QA\|_F^2 = \|A\|_F^2 \tag{7}$$

This is because

$$\|QA\|_F^2 = \mathrm{tr}[A^\top Q^\top QA] = \mathrm{tr}[A^\top A] = \|A\|_F^2$$

where the second step we uses the fact that $Q^\top Q = I_{d_1+d_2}$.

Let $X \in \mathbb{R}^{n\times n}$ denote the $YC \in \mathbb{R}^{n\times n}$ in Lemma C.8. Recall that Q_y is obtained from QR-decomposition of $Y \in \mathbb{R}^{n\times(d_1+d_2)}$, so we can write $Y = Q_y R_y$. For simplicity, let Q denote Q_y and R denote R_y. Then we have

$$\begin{aligned}
\|Q[Q^\top M]_k - M\|_F^2 =& \|Q[Q^\top M]_k - QQ^\top M\|_F^2 + \|QQ^\top M - M\|_F^2 \\
=& \|[Q^\top M]_k - Q^\top M\|_F^2 + \|QQ^\top M - M\|_F^2 \\
\le& \|RC - Q^\top M\|_F^2 + \|QQ^\top M - M\|_F^2 \\
=& \|QRC - QQ^\top M\|_F^2 + \|QQ^\top M - M\|_F^2 \\
=& \|QRC - M\|_F^2 \\
=& \|X - M\|_F^2
\end{aligned} \tag{8}$$

where the first step uses (6) by setting $A = [Q^\top M]_k$, the second step uses (7), the third step uses the fact that $\mathrm{rank}(RC) \le \mathrm{rank}(QRC) = \mathrm{rank}(Y) \le k$ and $[Q^\top M]_k \in \mathbb{R}^{(d_1+d_2)\times n}$ is the best rank-k approximation for $Q^\top M \in \mathbb{R}^{(d_1+d_2)\times n}$, the fourth step again uses Eq. (7), the fifth step uses the Eq. (6) by setting $A = RC$, and the last step uses that $QRC = YC = X \in \mathbb{R}^{n\times n}$.

Therefore,

$$\begin{aligned}
\|Q[Q^\top M]_k - M\|_F^2 \le& \|M - X\|_F^2 \\
\le& 5\|M - [M]_k\|_F^2 + \Delta_1 + \Delta_2.
\end{aligned} \tag{9}$$

where the first step follows from (8), the second step follows from Lemma C.8.

Let $W \in \mathbb{R}^{(d_1+d_2)\times k}$ denote the top k singular vectors of $Q^\top M \in \mathbb{R}^{(d_1+d_2)\times n}$. Since $\widetilde{W} \in \mathbb{R}^{(d_1+d_2)\times k}$ are the top

Table 1: Table of parameters

Notation	Choice	Location	Comment
s	$O(k \log k)$	Lemma C.7	size of oblivious sketching matrix
d_1	$O(k \log^2 k)$	Lemma C.7	size of column sampling matrix
d_2	$O(k/\epsilon)$	Lemma C.8	size of adaptive column sampling
Δ_1	$O(\epsilon^2/s)\|M\|_{1,2}^2$	Lemma C.7	error from oblivious sketching matrix
Δ_2	$O(\sqrt{\epsilon^2 d_1} + \epsilon^2 d_1)\|M\|_F^2$	Lemma C.8	error from column sampling matrix
Δ_3	$O(\epsilon^2(d_1+d_2))\|M\|_F^2$	Lemma C.9	error from adaptive column sampling

k singular vectors of $\widetilde{\Pi} \in \mathbb{R}^{(d_1+d_2)\times n}$, we have

$$
\begin{aligned}
& \|(\widetilde{W}\widetilde{W}^\top - I)Q^\top M\|_F^2 \qquad (10)\\
\le\ & 2\|(\widetilde{W}\widetilde{W}^\top - I)\widetilde{\Pi}\|_F^2 + 2\|(\widetilde{W}\widetilde{W}^\top - I)(Q^\top M - \widetilde{\Pi})\|_F^2 \\
\le\ & 2\|(WW^\top - I)\widetilde{\Pi}\|_F^2 + 2\|(\widetilde{W}\widetilde{W}^\top - I)(Q^\top M - \widetilde{\Pi})\|_F^2 \\
\le\ & 2\|(WW^\top - I)Q^\top M\|_F^2 + 2\|(WW^\top - I)(Q^\top M - \widetilde{\Pi})\|_F^2 + 2\|(\widetilde{W}\widetilde{W}^\top - I)(Q^\top M - \widetilde{\Pi})\|_F^2 \\
\le\ & 2\|[Q^\top M]_k - Q^\top M\|_F^2 + 2\|WW^\top - I\|_2^2\|(Q^\top M - \widetilde{\Pi})\|_F^2 + 2\|\widetilde{W}\widetilde{W}^\top - I\|_2^2\|(Q^\top M - \widetilde{\Pi})\|_F^2 \\
\le\ & 2\|[Q^\top M]_k - Q^\top M\|_F^2 + 4\|(Q^\top M - \widetilde{\Pi})\|_F^2 \\
\le\ & 2\|[Q^\top M]_k - Q^\top M\|_F^2 + O(\epsilon^2)\|Q\|_F^2\|M\|_F^2 \\
\le\ & 2\|[Q^\top M]_k - Q^\top M\|_F^2 + O(\epsilon^2)\cdot(d_1+d_2)\|M\|_F^2 \\
=\ & 2\|[Q^\top M]_k - Q^\top M\|_F^2 + O(\epsilon^2(d_1+d_2))\|M\|_F^2 \qquad (11)
\end{aligned}
$$

where the first step uses the fact that $\|A+B\|_F^2 \le 2\|A\|_F^2 + 2\|B\|_F^2$, the second step uses the fact that $(\widetilde{W}\widetilde{W}^\top - I)Q^\top M = [Q^\top M]_k - Q^\top M$ and $[Q^\top M]_k$ is the best rank k approximation of $Q^\top M$, the third step uses the fact that $\|AB\|_F \le \|A\|_2 \cdot \|B\|_F$ for any matrices A, B, and $WW^\top Q^\top M = [Q^\top M]_k$, since W are the top k singular vectors of $Q^\top M$, the fourth step uses $\|AA^\top - I\|_2 \le 1$ for all orthonormal matrix $A \in \mathbb{R}^{(d_1+d_2)\times k}$ since $(AA^\top - I)^2 = I - AA^\top$, the fifth step uses convergence grantee in Theorem 4.2, the sixth step follows from Q is an orthonormal matrix with $d_1 + d_2$ columns.

We now bound the error using the above two claims.

Noting $L = Q\widetilde{W} \in \mathbb{R}^{(d_1+d_2)\times k}$, hence by (7) we have

$$\|LL^\top M - QQ^\top M\|_F^2 = \|\widetilde{W}\widetilde{W}^\top Q^\top M - Q^\top M\|_F^2 \qquad (12)$$

Therefore

$$
\begin{aligned}
\|LL^\top M - M\|_F^2 &= \|LL^\top M - QQ^\top M\|_F^2 + \|M - QQ^\top M\|_F^2 \\
&\le 2\|Q[Q^\top M]_k - QQ^\top M\|_F^2 + O(\epsilon^2(d_1+d_2))\|M\|_F^2 + \|M - QQ^\top M\|_F^2 \\
&\le 2\|Q[Q^\top M]_k - M\|_F^2 + O(\epsilon^2(d_1+d_2))\|M\|_F^2 \\
&\le 10\|M - [M]_k\|_F^2 + 2\Delta_1 + 2\Delta_2 + O(\epsilon^2(d_1+d_2))\|M\|_F^2 \\
&= 10\|M - [M]_k\|_F^2 + 2\Delta_1 + 2\Delta_2 + \Delta_3,
\end{aligned}
$$

where the first step uses the fact that $L = Q\widetilde{W}$ and (6) with $A = \widetilde{W}L^\top$, the second step uses (12) and (10), the third step uses (6) with $A = [Q^\top M]_k$, the fourth step uses Lemma C.8, and the last step is the definition of Δ_3. □

C.6 Main result

Theorem C.10. *There exists an algorithm (procedure* LowRankApprox *in Algorithm 2) that with parameter settings as in Table 1, runs in query time* $nk\,\mathrm{poly}(\log n, 1/\epsilon)$ *and space* $\widetilde{O}(nk/\epsilon^2)$*, outputs a matrix* $L \in \mathbb{R}^{n\times k}$ *so that*

$$\|LL^\top M - M\|_F^2 \le 10\|M - [M]_k\|_F^2 + O(\epsilon d_1)\|M\|_F^2 + O(\epsilon^2/s)\|M\|_{1,2}^2.$$

holds with probability at least $9/10$.

Proof of guarantee.

$$\begin{aligned}
&\|LL^\top M - M\|_F^2 \\
\leq\ & 10\|M-[M]_k\|_F^2 + 2\Delta_1 + 2\Delta_2 + \Delta_3 \\
=\ & 10\|M-[M]_k\|_F^2 + O(\epsilon^2/s)\|M\|_{1,2}^2 + O(\sqrt{\epsilon^2 d_1} + \epsilon^2 d_1)\|M\|_F^2 + O(\epsilon^2(d_1+d_2))\|M\|_F^2 \\
=\ & 10\|M-[M]_k\|_F^2 + O(\epsilon^2/s)\|M\|_{1,2}^2 + O(\sqrt{\epsilon^2 d_1} + \epsilon^2 d_1)\|M\|_F^2 + O(\epsilon d_1)\|M\|_F^2 \\
=\ & 10\|M-[M]_k\|_F^2 + O(\epsilon^2/s)\|M\|_{1,2}^2 + O(\epsilon d_1)\|M\|_F^2
\end{aligned}$$

where the first step uses Lemma C.9, the third step uses the definition of $d_1 = O(k\log^2 k)$ and $d_2 = O(k/\epsilon)$ so $d_1 + d_2 = O(d_1/\epsilon)$. □

Proof of time and space. The largest matrix we ever need to store during the process has size $n \times (d_1 + d_2) = O(\epsilon^{-1} nk \log^2 k)$. The space needed by LogSum is bounded by $\widetilde{O}(\epsilon^{-2}nk)$ by Theorem 4.2. So the overall space used is at most $\widetilde{O}(\epsilon^{-2}nk)$.

Since we only call LogSum 4 times in the whole process, the query time hence follows from Theorem 4.2. □

Notice that $\|M\|_{1,2} \geq \|M\|_F \geq \|M-[M]_k\|_F$, so we can rescale ϵ to get Theorem 5.1.

D Examples Demonstrating the Differences Between A and $\log(A)$

D.1 $\mathrm{rank}(A) \gg \mathrm{rank}(\log A)$

In this section, we provide a matrix $A \in \mathbb{R}^{n\times n}$ with rank-n, however, the $\mathrm{rank}(\log A) = 1$.

Recall the definition of Vandermonde matrix.

Definition D.1. *An $m \times n$ Vandermonde matrix usually is defined as follows*

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \cdots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \cdots & \alpha_m^{n-1} \end{bmatrix}$$

or $V_{i,j} = \alpha_i^{j-1}, \forall i \in [m], j \in [n]$

Theorem D.2. *Let A denote a $n \times n$ Vandermonde matrix with $\alpha_i \neq \alpha_j, \forall i \neq j$. Then* $\mathrm{rank}(A) = n$ *and* $\mathrm{rank}(\log(A)) = 1$.

Proof. By definition of A, we have,

$$\begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \cdots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-1} \end{bmatrix}$$

Note that, we can compute the determinant of matrix A,

$$\det(A) = \prod_{1\leq i<j\leq n} (\alpha_j - \alpha_i).$$

Since $\alpha_j \neq \alpha_i, \forall j \neq i$, thus $\det(A) \neq 0$ which implies $\mathrm{rank}(A) = n$.

By definition of $\log(A)$, we have,

$$\log(A) = \begin{bmatrix} 0 & \log(\alpha_1) & 2\log(\alpha_1) & \cdots & (n-1)\log(\alpha_1) \\ 0 & \log(\alpha_2) & 2\log(\alpha_2) & \cdots & (n-1)\log(\alpha_2) \\ 0 & \log(\alpha_3) & 2\log(\alpha_3) & \cdots & (n-1)\log(\alpha_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \log(\alpha_n) & 2\log(\alpha_n) & \cdots & (n-1)\log(\alpha_n) \end{bmatrix}$$
$$= \begin{bmatrix} \log(\alpha_1) \\ \log(\alpha_2) \\ \log(\alpha_3) \\ \vdots \\ \log(\alpha_n) \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 2 & \cdots & (n-1) \end{bmatrix}.$$

Therefore $\mathrm{rank}(\log(A)) = 1$. □

D.2 $\mathrm{rank}(A) \ll \mathrm{rank}(\log A)$

In this section, we provide a matrix $A \in \mathbb{R}^{n \times n}$ with rank-$n/2$, however, the $\mathrm{rank}(\log A) = n$.

Theorem D.3. *There is a matrix $A \in \mathbb{R}^{n \times n}$ such that $\mathrm{rank}(A) = n/2$ and $\mathrm{rank}(\log(A)) = n$.*

Proof. Let B denote a 2×2 matrix as follows

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

It is not hard to see that $\mathrm{rank}(B) = 2$ and $\mathrm{rank}(\log(B)) = 1$. We define matrix A by copying B by $n/2$ times on A's diagonal blocks,

$$A = \begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ 0 & B & 0 & \cdots & 0 \\ 0 & 0 & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & B \end{bmatrix}.$$

Then we have $\mathrm{rank}(A) = n/2$ and $\mathrm{rank}(\log(A)) = n$. □

Due to the following fact, copying rank-1 matrix several times won't give a better theorem D.3.

Fact D.4. *For any* rank-1 *matrix A, the* $\mathrm{rank}(\log(A)) \leq 2$.

Proof. Without loss of generality, let's assume A can be written as

$$A = \alpha^\top \beta = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \cdot \begin{bmatrix} \beta_1 & \beta_2 & \cdots \beta_n \end{bmatrix}$$

Let B denote $\log(A)$, then it is easy to that $B_{i,j} = \log(\alpha_i) + \log(\beta_i)$. Therefore matrix B can be decomposed into the following case

$$B = \log(\alpha) \cdot \mathbf{1} + \mathbf{1}^\top \log(\beta)^\top$$

Thus, $\mathrm{rank}(B) \leq 2$. □

Algorithm 4 Linear regression by f-matrix product sketch

procedure LINEARREGRESSION($M, b, n, d,$) ▷ Theorem E.1
  Implicitly form $A = \log M$ ▷ $A \in \mathbb{R}^{n \times d}$
  $s \leftarrow \mathrm{poly}(d/\epsilon)$
  Choosing a sketching matrix $S \in \mathbb{R}^{s \times n}$
  $\widetilde{SA} \leftarrow$ SKETCHLOG(S, M)
  $\widetilde{x} \leftarrow \min_{x \in \mathbb{R}^d} \|\widetilde{SA}x - Sb\|_2$
  return $\widetilde{x}$
end procedure

E Application of f-Matrix Product Sketch in Linear Regression

In this section, we consider the application to linear regression. Linear regression is a fundamental problem in machine learning, and there is a long line of work using sketching/hashing idea to speed up the running time [CW13, MM13, PSW17, LHW17, ALS+18, DSSW18, SWZ19b, CWW19].

Recall that for a matrix $M \in \mathbb{R}^{n \times d}$, we use $\log(M)$ to denote the $n \times d$ matrix where the entry at i-th row and j-th column of matrix $\log M$ is $\log(M_{i,j})$.

Theorem E.1 (Linear regression)**.** *Given matrix $M \in \mathbb{R}^{n \times d}$ and vector $b \in \mathbb{R}^d$ where $n \gg d$. Let $A = \log M \in \mathbb{R}^{n \times d}$. There is an one-pass algorithm (Algorithm 4) that uses $\mathrm{poly}(d, \log n, 1/\epsilon)$ space, receives the update of M in the stream, and outputs vector $\widetilde{x} \in \mathbb{R}^d$ such that*

$$\|A\widetilde{x} - b\|_2 \leq (1+\epsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 + \tau,$$

holds with probability at least $9/10$ and where $\tau = \|b\|_2 / \mathrm{poly}(d/\epsilon)$.

Proof. Without loss of generality, we assume that $\|A\|_2 = 1$ in the proof.

Let $x^* \in \mathbb{R}^d$ denote the optimal solution of this problem,

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

Let OPT denote $\|Ax^* - b\|_2$. Let $x' \in \mathbb{R}^d$ denote the optimal solution of this problem,

$$\min_{x \in \mathbb{R}^d} \|SAx - Sb\|_2.$$

By property of sketching matrix, we have

$$\|Ax' - b\|_2 \leq (1+\epsilon)\|Ax - b\|_2.$$

Note that $x' = (SA)^\dagger Sb$. Let $\widetilde{x} \in \mathbb{R}^d$ denote the optimal solution of

$$\min_{x \in \mathbb{R}^d} \|\widetilde{SA}x - Sb\|_2.$$

It means $\widetilde{x} = (\widetilde{SA})^\dagger Sb$. We have

$$\begin{aligned}
& \|A\widetilde{x} - b\|_2 \\
\leq & \|Ax' - b\|_2 + \|A\widetilde{x} - Ax'\|_2 \\
\leq & (1+\epsilon)\,\mathrm{OPT} + \|A\widetilde{x} - Ax'\|_2 \\
= & (1+\epsilon)\,\mathrm{OPT} + \underbrace{\|A(\widetilde{SA})^\dagger Sb - A(SA)^\dagger Sb\|_2}_{C_1}
\end{aligned} \tag{13}$$

where the first step follows by triangle inequality, the second step follows by $\|Ax' - b\|_2 \leq (1+\epsilon)\,\mathrm{OPT}$, the third step follows by definition of x' and x^*.

Now the question is how to bound the term C_1 in Eq. (13). We can upper bound C in the following way,

$$\begin{aligned} C_2 = & \|A(\widetilde{SA})^\dagger Sb - A(SA)^\dagger Sb\|_2 \\ \leq & \|A\|_2 \|(\widetilde{SA})^\dagger - (SA)^\dagger\|_2 \|Sb\|_2 \\ = & \|(\widetilde{SA})^\dagger - (SA)^\dagger\|_2 \|Sb\|_2 \\ \lesssim & \underbrace{\|(\widetilde{SA})^\dagger - (SA)^\dagger\|_2}_{C_2} \|b\|_2 \end{aligned}$$

the third step follows by $\|A\|_2 = 1$, and the the last step follows by $\|Sb\|_2 = O(1) \cdot \|b\|_2$. Next, we show how to bound the term C_2 in the above equation, using Lemma F.1, F.3 and F.2, we have

$$\begin{aligned} C_2 = & \|(\widetilde{SA})^\dagger - (SA)^\dagger\|_2 \\ \lesssim & \max(\|(\widetilde{SA})^\dagger\|_2^2, \|(SA)^\dagger\|_2^2) \cdot \|\widetilde{SA} - SA\|_2 \\ \lesssim & \|(SA)^\dagger\|_2^2 \cdot \|\widetilde{SA} - SA\|_2 \\ \lesssim & \|A^\dagger\|_2^2 \cdot \|\widetilde{SA} - SA\|_2 \\ \leq & \|A^\dagger\|_2^2 / \kappa^2 \operatorname{poly}(d/\epsilon) \\ = & \|A\|_2^2 / \operatorname{poly}(d/\epsilon) \\ \lesssim & 1/\operatorname{poly}(d/\epsilon) \end{aligned}$$

where the second step follows by Lemma F.1, the third step follows by Lemma F.3, the fourth step follows by property of sketching matrix S, the fifth step follows by size of S and Lemma F.2 the last step follows by $\|A\|_2 = 1$. □

F Tools

In this section, we introduce several basic perturbation results.

[Wed73] presented a perturbation bound of Moore-Penrose inverse the spectral norm,

Lemma F.1 ([Wed73], Theorem 1.1 in [MZ10]). *Given two matrices $A, B \in \mathbb{R}^{d_1 \times d_2}$ with full column rank, we have*

$$\|A^\dagger - B^\dagger\|_2 \lesssim \max(\|A^\dagger\|_2^2, \|B^\dagger\|_2^2) \cdot \|A - B\|_2.$$

Lemma F.2 (Latala's theorem [Lat05], Theorem 5.37 in [Ver10]). *Let A be a random $n \times d$ matrix whose entries $A_{i,j}, \forall (i,j) \in [n] \times [d]$ are independent centered random variables with finite fourth moment. Then*

$$\mathbf{E}[\|A\|_2] \lesssim \max_{i \in [n]} \left(\sum_{j=1}^d \mathbf{E}[A_{i,j}^2] \right)^{1/2} + \max_{j \in [d]} \left(\sum_{i=1}^n \mathbf{E}[A_{i,j}^2] \right)^{1/2} + \left(\sum_{i=1}^n \sum_{j=1}^d \mathbf{E}[A_{i,j}^4] \right)^{1/4}.$$

Lemma F.3. *Let $B = A + E$, if $|E_{i,j}| \leq \epsilon'$, where $\epsilon' = \epsilon/(d_1 d_2 \kappa(A) 10)$, then*

$$(1 - \epsilon)\|A^\dagger\| \leq \|B^\dagger\| \leq (1 + \epsilon)\|A^\dagger\|.$$

Proof. Given the definition of B, we can rewrite $BB^\top$ into four terms,

$$BB^\top = (A + E)(A + E)^\top = AA^\top + EA^\top + AE^\top + EE^\top.$$

Since we can bound

$$\begin{aligned} \|EA^\top + AE^\top + EE^\top\|_2 \leq & 3\|E\|_F \|A\|_2 \\ \leq & 3\epsilon' d_1 d_2 \sigma_1(A) \\ \leq & \epsilon \sigma_{\min}(A)/3 \end{aligned}$$

Thus,

$$AA^\top - \epsilon\sigma_{\min}(A)/3 \cdot I \preceq BB^\top \preceq AA^\top + \epsilon\sigma_{\min}(A)/3 \cdot I$$

Thus,

$$\begin{aligned}\|B^\dagger\| &= \frac{1}{\sigma_{\min}(B)} \\ &\leq \frac{1}{\sigma_{\min}(A) - \sigma_{\min}(A)\epsilon/3} \\ &\leq (1+\epsilon)\frac{1}{\sigma_{\min}(A)} \\ &= (1+\epsilon)\|A^\dagger\|_2.\end{aligned}$$

This completes the proof. □

Theorem F.4 (Generalized rank-constrained matrix approximations, Theorem 2 in [FT07])**.** *Given matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{q\times d}$, let the SVD of B be $B = U_B\Sigma_B V_B^\top$ and the SVD of C be $C = U_C\Sigma_C V_C^\top$. Then,*

$$B^\dagger(U_B U_B^\top A V_C V_C^\top)_k C^\dagger = \underset{\text{rank}-k\ X\in\mathbb{R}^{p\times q}}{\arg\min} \|A - BXC\|_F$$

where $(U_B U_B^\top A V_C V_C^\top)_k \in \mathbb{R}^{p\times q}$ is of rank at most k and denotes the best rank-k approximation to $U_B U_B^\top A V_C V_C^\top \in \mathbb{R}^{n\times d}$ in Frobenius norm.

G Complete Experimental Results

To demonstrate the advantage of our proposed method, we complement the theoretical analysis with empirical study on synthetic and real data. We consider the low rank approximation task with $f(x) = \log(x)$ and $f(x) = \sqrt{x}$, vary the amount of space used by our method, and compare the errors of the solutions obtained to the optimum. The we provide additional experiments testing some other aspects of the method such as robustness to the parameter values.

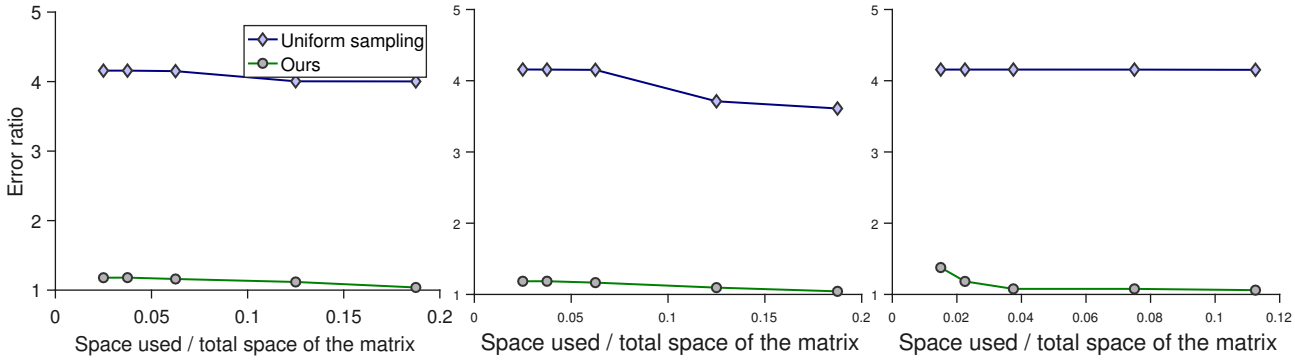
Setup Given a data stream in the form of (i_t, j_t, δ_t), we use the algorithm in Section 5 to compute the top $k = 10$ singular vectors L, and then compare the error of this solution to the error of the optimal solution (i.e., the true top k singular vectors). Let A denote the accumulated matrix, $M = f(A)$ denote the transformed one, and U denote the top k singular vectors of M. Then the evaluation criterion is

$$\text{error-ratio}(L) = \frac{\|M - LL^\top M\|_F}{\|M - UU^\top M\|_F}.$$

Clearly, the error ratio is at least 1, and a value close to 1 demonstrates that our solution is nearly optimal.

Besides demonstrating the effectiveness, we also exam the tradeoff between the solution quality and the space used. Recall that there is a parameter in the the sketching methods controlling the amount of space used (line 20 in LogSum and line 6 in PolySum). We vary its value, and set the parameters in other steps of our algorithm so that the amount of space used is dominated by that of the sketching. We then plot how the error ratios change with the amount of space used. The plotted results are the average of 5 runs; the variances are too small to plot.

Finally, we also report the results of a baseline method: uniformly at random sample a subset T of columns from A, and then compute the top k singular vectors of $f(T)$. The space occupied by the columns sampled is similar to the space required by our algorithm for fair comparison. Since our algorithm is randomized, the expected amount of space occupied is used to determine the sample size of the baseline, and is also used for the plots. In the experiments, the actual amount occupied is within about 10% of the expected value.



(a) LogData, $n = 10^4$ (b) LogData, $n = 3 \cdot 10^4$ (c) LogData, $n = 5 \cdot 10^4$

Figure 2: Error ratios on the synthetic data LogData. The x-axis is the ratio between the amount of space used by the algorithms and the total amount of space occupied by the data matrix. The y-axis is the ratio between the error of the solutions output by the algorithms and the optimal error.

Implementation and Parameter Setting. In our algorithm for low rank approximation, an FJLT matrix S is used [Ach03, AC06]. In the step of adaptive sampling, instead of setting the threshold η, for simplicity we let $q_i = \max\{\widetilde{s}_i, 0\}$ and set $p_i = q_i + \sum_i q_i/n$.

For the sketching subroutine, instead of specifying the desired ϵ, we directly set the size of the data structure, so as to exam the tradeoff between space and accuracy. Then we set $s = d_1 = d_2$ and set their value so that the space used in the corresponding step is at most that used by the sketch method. In particular, we set them equal to the size upper bounds in line 20 in LogSum or line 6 in PolySum.

G.1 Synthetic Data

Data Generation. The following data sets are generated. Note that although we don't provide theoretical analysis for $f(x) = \sqrt{x}$, one could follow that for $f(x) = \log(x)$ to get similar guarantees, and we also generate synthetic data to test our method in this case.

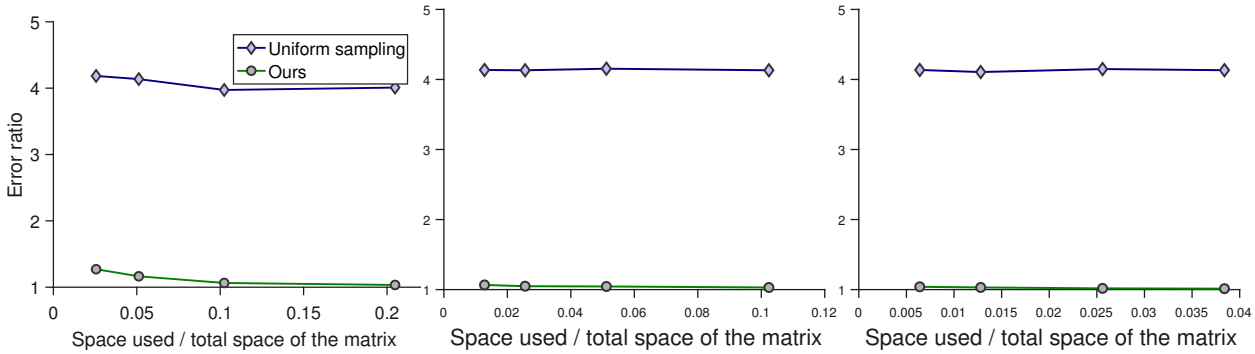
1. LogData: This is for the experiments with $f(x) = \log(x)$. First generate a matrix M of $n \times n$ where the entries are i.i.d. Gaussians. To break the symmetry of the columns, scale the length of the i-th column to $4/i$. Finally, generate matrix A with $A_{ij} = \exp(M_{ij})$. Each entry A_{ij} is divided into equally into 5 updates $(i, j, A_{ij}/5)$, and all the updates arrive in a random order. The size n can be 10000, 30000, and 50000.

2. SqrtData: This is for the experiments with $f(x) = \sqrt{x}$. The data and update stream are generated similarly as LogData, except that $A_{ij} = M_{ij}^2$. We tested on sizes $n = 10000$ and $n = 30000$.

Results. Figure 2 shows the results on the synthetic data LogData, and Figure 3 shows those on SqrtData. In general, the error ratio of our method is much better than that of the uniform sampling baseline: ours is close to 1 while that of uniform sampling is about 4. It also shows that our method can greatly reduces the amount of space needed by orders while merely comprising the solution quality, and this advantage is more significant on larger data sets. For example, when $n = 50000$, using space about 5% of the matrix size leads to only about 5% extra error over the optimum. Finally, we note that these observations are consistent on both $f(x) = \log(x)$ and $f(x) = \sqrt{x}$.

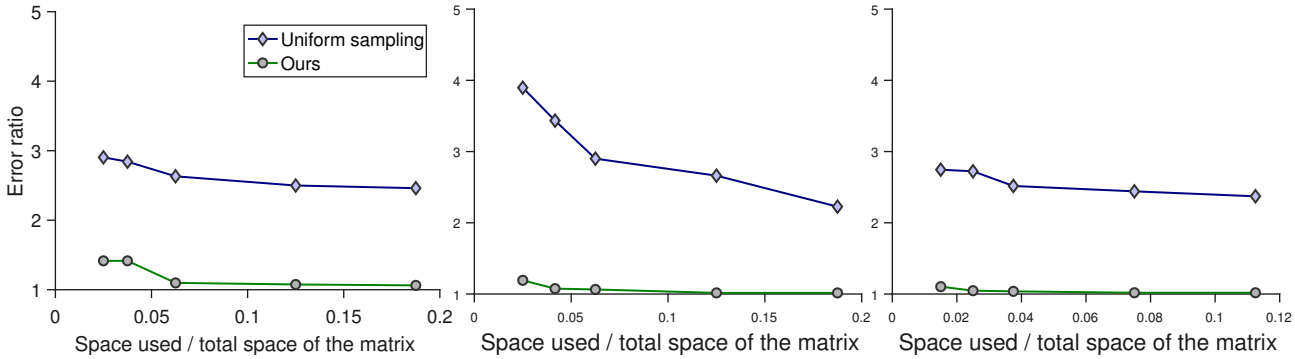
G.2 Real Data

We exam our method on the real world data from the NLP application word embedding, which is a motivating example for proposing our approach. Our method with $f(x) = \log(x + 1)$ is used. The parameters are set in a similar way as for the synthetic data.



(a) SqrtData, $n = 10^4$ (b) SqrtData, $n = 3 \cdot 10^4$ (c) SqrtData, $n = 5 \cdot 10^4$

Figure 3: Error ratios on the synthetic data SqrtData. The x-axis is the ratio between the amount of space used by the algorithms and the total amount of space occupied by the data matrix. The y-axis is the ratio between the error of the solutions output by the algorithms and the optimal error.



(a) Real data, $n = 10^4$ (b) Real data, $n = 3 \cdot 10^4$ (c) Real data, $n = 5 \cdot 10^4$

Figure 4: Error ratios on the real data (Wikipedia). The x-axis is the ratio between the amount of space used by the algorithms and the total amount of space occupied by the data matrix. The y-axis is the ratio between the error of the solutions output by the algorithms and the optimal error.

Data Collection. The data set is the entire Wikipedia corpus [Wik12] consisting of about 3 billion tokens. Details can be found in the appendix and only a brief description is provided here. The matrix to be factorized is M with $M_{ij} = p_j \log \frac{N_{ij} N}{N_i N_j}$ where N_{ij} is the number of times words i and j co-occur in a window of size 10, N_i is the number of times word i appears, N is the total number of words in the corpus, and p_j is a weighting factor depending set to $p_j = \max\{1, (N_j/N_{10})^2\}$, which puts larger weights on more frequent words since they are less noisy [PSM14, LG14]. Note that N_i's and N can be computed easily, so essentially the only dynamically update part is $\log N_{ij}$.

The data stream is generated as a window of size 10 slides along the sentences in the corpus and we collect the co-occurrence counts of the word pairs in the window. Here the count is weighted, i.e., if two words appear in a distance of t inside the window, then the count update value is $1/t$ as in [PSM14]. We consider the matrix for the most frequent n words, where $n = 10000$, 30000, and 50000.

Results. Figure 4 shows the results on the real data. The observations are similar to those on the synthetic data: the errors of our method are much better than the baseline, and are close to the optimum; the method is very space efficient without increasing the error much. These results again demonstrate its effectiveness.

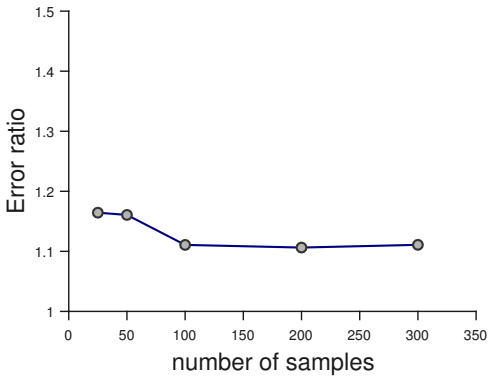


Figure 5: Error ratios when using different sample size in the algorithm. The x-axis is the ratio between the amount of space used by the algorithms and the total amount of space occupied by the data matrix. The y-axis is the ratio between the error of the solutions output by the algorithms and the optimal error.

G.3 The Effect of the Sample Size

In our algorithm we have parameters $s = d_1 = d_2$ that determine the sample sizes in different steps of the algorithm. In previous experiments, we set them equal to the size upper bounds in line 20 in LogSum or line 6 in PolySum. Here we consider varying their values. In particular, we use the Wikipedia data with $n = 10000$ and $f(x) = \log(x)$ and set the size upper bound of the sketch method to be 200. Then we set $s = d_1 = d_2 = \gamma$ and vary the value of γ.

Results. Figure 5 shows the results with various sample sizes. It is observed that smaller sample sizes lead to worse errors as expected, but overall the results are quite stable across different sizes. This demonstrates the robustness of our method to these parameters. It is also observed that after a certain value, increasing the sample size doesn't lead to better error, which should be due to the approximation error introduced by the sketch. The results suggest that in general the sample size should be set approximately equal to the size upper bound in the sketch method.