Single Pass Entrywise-Transformed Low Rank Approximation

Yifei Jiang [1] **Yi Li** [2] **Yiming Sun** [2] **Jiaxin Wang** [3] **David P. Woodruff** [4]

Abstract

In applications such as natural language processing or computer vision, one is given a large $n \times d$ matrix $A = (a_{i,j})$ and would like to compute a matrix decomposition, e.g., a low rank approximation, of a function $f(A) = (f(a_{i,j}))$ applied entrywise to A. A very important special case is the likelihood function $f(A) = \log(|a_{ij}| + 1)$. A natural way to do this would be to simply apply f to each entry of A, and then compute the matrix decomposition, but this requires storing all of A as well as multiple passes over its entries. Recent work of Liang et al. shows how to find a rank-k factorization to $f(A)$ for an $n \times n$ matrix A using only $n \cdot \mathrm{poly}(\epsilon^{-1}k \log n)$ words of memory, with overall error $10\|f(A) - [f(A)]_k\|_F^2 + \mathrm{poly}(\epsilon/k)\|f(A)\|_{1,2}^2$, where $[f(A)]_k$ is the best rank-k approximation to $f(A)$ and $\|f(A)\|_{1,2}^2$ is the square of the sum of Euclidean lengths of rows of $f(A)$. Their algorithm uses three passes over the entries of A. The authors pose the open question of obtaining an algorithm with $n \cdot \mathrm{poly}(\epsilon^{-1}k \log n)$ words of memory using only a single pass over the entries of A. In this paper we resolve this open question, obtaining the first single-pass algorithm for this problem and for the same class of functions f studied by Liang et al. Moreover, our error is $\|f(A) - [f(A)]_k\|_F^2 + \mathrm{poly}(\epsilon/k)\|f(A)\|_F^2$, where $\|f(A)\|_F^2$ is the sum of squares of Euclidean lengths of rows of $f(A)$. Thus our error is significantly smaller, as it removes the factor of 10 and also $\|f(A)\|_F^2 \leq \|f(A)\|_{1,2}^2$. We also give an algorithm for regression, pointing out an error in previous work, and empirically validate our results.

[1]Tianjin University, China [2]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore [3]Wuhan University of Technology, China [4]Department of Computer Science, Carnegie Mellon University, USA. Correspondence to: Yi Li <yili@ntu.edu.sg>, David P. Woodruff <dwoodruf@andrew.cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

There are numerous applications with matrices that are too large to fit in main memory, such as matrices arising in machine learning, image clustering, natural language processing, network analysis, and recommendation systems. This makes it difficult to process such matrices, and one common way of doing so is to stream through the entries of the matrix one at a time and maintain a short summary or *sketch* that allows for further processing. In some of these applications, such as network analysis or distributed recommendation systems, matters are further complicated because one would like to take the difference or sum of two or more matrices over time.

A common goal in such applications is to compute a low rank approximation to a large matrix $A \in \mathbb{R}^{n \times d}$. If the rank of the low rank approximation is k, then one can approximate A as $U \cdot V$, where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times d}$. This results in a parameter reduction, as U and V only have $(n+d)k$ parameters in total, as compared to the nd parameters required of A. Since $k \ll \min(n, d)$, this parameter reduction is significant. Not only does it result in much smaller storage, when multiplying A by a vector x, it also now only takes $O((n+d)k)$ time instead of $O(nd)$ time, since one can first compute $V \cdot x$ and then $U \cdot (V \cdot x)$.

A challenge in the above applications is that often wants to compute a low rank approximation not to A, but to an *entrywise* transformation to A by a function f. Namely, if $A = (a_{i,j})$, then we define $f(A) = f(a_{i,j})$ where we apply the function f to each entry of A. Common functions f include $f(x) = \log_2^c(|x| + 1)$ or $f(x) = |x|^\alpha$ for $0 \leq \alpha \leq 2$. Indeed, for word embeddings in natural language processing (NLP), an essential subroutine is to perform a low rank approximation of a matrix after applying the log-likelihood function to each entry, which corresponds to $f(x) = \log_2(|x| + 1)$. Note that in NLP the input matrices are often word co-occurrence count matrices, which can be created e.g., from the entire Wikipedia database. Thus, such matrices are huge, with millions of rows and columns, and hard to store in memory. This necessitates models such as the streaming model for processing such data.

We can indeed capture the above scenarios formally with the *streaming model* of computation. In this model, there is

a large underlying matrix A, and we see a long sequence of updates to its coordinates in the form (i, j, δ) with $\delta \in \{\pm 1\}$, and representing the update $A_{i,j} \leftarrow A_{i,j} + \delta$. Each pass over the data stream is very expensive, and thus one would like to minimize the number of such passes. Also, one would like to use as little memory as possible to compute a low rank approximation of the transformed matrix $f(A)$ in this model. In this paper we will consider approximately optimal low rank approximations, meaning factorizations $U \cdot V$ for which $\|U \cdot V - f(A)\|_F^2 \leq \|[f(A)]_k - f(A)\|_F^2 + \mathrm{poly}(\epsilon/k)\|f(A)\|_F^2$, where $[f(A)]_k$ is the optimal rank-k matrix approximating $f(A)$ in Frobenius norm. Recall the Frobenius norm $\|B\|_F$ of a matrix B is defined to be $(\sum_{i,j} B_{i.j}^2)^{1/2}$, which is the entrywise Euclidean norm of B.

Although there is a body of work in the streaming model computing low rank approximations of matrices (Boutsidis et al., 2016; Clarkson & Woodruff, 2009; Drineas et al., 2012; Upadhyay, 2014; Woodruff, 2014), such methods no longer apply in our setting due to the non-linearity of the function f. Indeed, a number of existing methods are based on dimensionality reduction, or *sketching*, whereby one stores $S \cdot A$ for a random matrix S with a small number of rows. If there were no entrywise transformation applied to A, then given an update $A_{i,j} \leftarrow A_{i,j} + \delta$, one could simply update the sketch $(S \cdot A)_j \leftarrow (S \cdot A)_j + S_i \cdot \delta$, where $(S \cdot A)_j$ denotes the j-th column of $S \cdot A$ and S_i denotes the i-th column of S. However, given an entrywise transformation f, which may be non-linear, and given that we may see multiple updates to an entry of A, e.g., in the difference of two datasets, it is not clear how to maintain $S \cdot f(A)$ in a stream.

A natural question is: *can we compute a low-rank approximation to $f(A)$ in the streaming model with a small number of passes, ideally one, and a small amount of memory, ideally $n \cdot poly(k/\epsilon)$ memory?*

Motivated by the applications above, this question was asked by Liang et al. (2020); see also earlier work which studies entrywise low rank approximation in the distributed model (Woodruff & Zhong, 2016). The work of Liang et al. (2020) studies the function $f(x) = \log_2(|x| + 1)$ and gives a three-pass algorithm for $n \times n$ matrices A achieving $n \cdot \mathrm{poly}(\epsilon^{-1} k \log n)$ memory and outputting factors U and V with the following error guarantee:

$$\|U \cdot V - f(A)\|_F^2 \leq 10\,\|[f(A)]_k - f(A)\|_F^2 + \mathrm{poly}(\epsilon/k)\,\|f(A)\|_{1,2}^2\,,$$

where for a matrix B, $\|B\|_{1,2}$ is the sum of the Euclidean lengths of its columns. We note that this error guarantee is considerably weaker than what we would like, as there is a multiplicative factor 10 and an additive error that depends on $\|f(A)\|_{1,2}$. Using the relationship between the 1-norm and the 2-norm, we have that $\|f(A)\|_{1,2}$ could be as large as $\sqrt{n}\|f(A)\|_F$, and so their additive error can be a $\sqrt{n}$ factor larger than what is desired. Also, although the memory is of the desired order, the fact that the algorithm requires 3 passes can significantly slow it down. Moreover, when data is crawled from the internet, e.g. in applications of network traffic, it may be impractical to store the entire data set (Durme & Lall, 2009). Therefore, in these settings it is impossible to make more than one pass over the data. Liang et al. (2020) say "Whether there exists a one-pass algorithm is still an open problem, and is left for future work."

1.1. Our Contributions

In this paper, we resolve this main open question of (Liang et al., 2020), obtaining a *one*-pass algorithm achieving $(n+d) \cdot \mathrm{poly}(\epsilon^{-1} k \log n)$ memory for outputting a low rank approximation for the function $f(x) = \log_2(|x| + 1)$, and achieving the stronger error guarantee:

$$\|U \cdot V - f(A)\|_F^2 \leq \|[f(A)]_k - f(A)\|_F^2 + \mathrm{poly}(\epsilon/k)\,\|f(A)\|_F^2\,.$$

We note that the $\mathrm{poly}(\epsilon/k)$ factor in both the algorithm of (Liang et al., 2020) and our algorithm can be made arbitrarily small by increasing the memory by a $\mathrm{poly}(k/\epsilon)$ factor, and thus it suffices to consider error of the form $\|[f(A)]_k - f(A)\|_F^2 + \epsilon\|f(A)\|_F^2$. We also note that our algorithm can be trivially adapted to rectangular matrices, so for ease of notation, we focus on the case $n = d$.

At a conceptual level, the algorithm of (Liang et al., 2020) uses one pass to obtain so-called approximate leverage scores of $f(A)$, then a second pass to sample columns of $f(A)$ according to these, and finally a third pass to do so-called adaptive sampling. In contrast, we observe that one can just do squared column norm sampling of $f(A)$ to obtain the above error guarantee, which is a common method for low rank approximation to A. However, in one pass it is not possible to sample actual columns of A or of $f(A)$ according to these probabilities, so we build a data structure to sample *noisy* columns by approximations to their squared norms in a single pass. This is related to block ℓ_2-sampling in a stream, see, e.g., (Mahabadi et al., 2020). However, the situation here is complicated by the fact that we must sample according to the sum of squares of f values of entries in a column of A, rather than the squared length of the column of A itself. The transformation function f's nonlinearity makes many of the techniques considered in (Mahabadi et al., 2020) inapplicable. To this end we build new hashing and sub-sampling data structures, generalizing data structures for length or squared length sampling from (Andoni et al., 2009; 2011; Levin et al., 2018), and we give a novel analysis for sampling noisy columns of A

proportional to the sum of squares of f values to their entries.

Additionally, we empirically validate our algorithm on a real-world data set, demonstrating significant advantages over the algorithm of Liang et al. (2020) in practice.

Finally, we apply our new sampling techniques to the regression problem, showing that our techniques are more broadly applicable. Although Liang et al. (2020) claim a result for regression, we point out an error in their analysis[1], showing that their algorithm obtains larger error than claimed, and the error of our regression algorithm is considerably smaller.

All omitted proofs can be found in the full version of this paper, which is given in the supplementary material.

2. Preliminaries

Notation. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a vector $x \in \mathbb{R}^n$, we denote by $|x|$ the vector whose i-th entry is $|x_i|$. For a matrix $A \in \mathbb{R}^{N\times N}$, let $A_{i,*}$ be the i-th row of A and $A_{*,j}$ be the j-th column of A. We also sometimes abbreviate $A_{i,*}$ or $A_{*,i}$ as A_i. We also define the norms $\|A\|_{p,q} = (\sum_i \|A_{i,*}\|_q^p)^{1/p}$, of which the Frobenius norm $\|A\|_F = \|A\|_{2,2}$ is a special case. Note that $\|A\|_{p,q} \equiv \|A^\top\|_{q,p}$. We use $\sigma_i(A)$ to denote the i-th singular value of A and $[A]_k$ to denote the best rank-k approximation to A. For a function f, let $f(A)$ denote the entrywise-transformed matrix $(f(A))_{ij} = f(A_{ij})$.

Low-rank Approximation. Given a matrix $M \in \mathbb{R}^{n\times n}$, an integer $k \le n$ and an accuracy parameter $\epsilon > 0$, our goal is to output an $n \times k$ matrix L with orthonormal columns for which $\|M - LL^\top M\|_F^2 \le \|M - [M]_k\|_F^2 + \epsilon\|M\|_F^2$, where $[M]_k = \arg\min_{\mathrm{rank}(M')\le k} \|M - M'\|_F^2$. Thus, $LL^\top$ provides a rank-k matrix to project the columns of M onto. The rows of $L^\top M$ can be thought of as the "latent features" in applications, and the rank-k matrix $LL^\top M$ can be factored as $L \cdot (L^\top M)$, where $L^\top M$ is a $k \times n$ matrix and L is an $n \times k$ matrix.

Best Low-rank Approximation. Consider the singular value decomposition $G = \sum_{t=1}^r \sigma_t u_t v_t^\top$, where $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0$ are the nonzero singular values of G, and $\{u_t\}$ and $\{v_t\}$ are orthonormal sets of column vectors such that $G^\top u_t = \sigma_t v_t$ and $Gv_t = \sigma_t u_t$ for each $t \le r$. The vectors $\{u_t\}$ and $\{v_t\}$ are called the left and the right singular values of G, respectively. By the Eckart-Young theorem, for any rotationally invariant norm $\|\cdot\|$, the matrix D_k that minimizes $\|G - D_k\|$ among all matrices D of rank at most k is given by $D_k = \sum_{t=1}^k G v_t v_t^\top$. This implies that $\|G - D_k\|_F^2 = \sum_{t=k+1}^r \sigma_t^2$.

[1] We have confirmed this with the authors.

For a matrix A, we denote by $\|A\|_2$ its operator norm, which is equal to its largest singular value.

Useful Inequalities. The first one is the matrix Bernstein inequality.

Lemma 1 (Matrix Bernstein). *Let $X_1, \ldots, X_s$ be s independent copies of a symmetric random matrix $X \in \mathbb{R}^{d\times d}$ with $\mathbb{E}(X) = \mathbf{0}$, $\|X\|_2 \le \rho$ and $\left\|\mathbb{E}(X^2)\right\|_2 \le \sigma^2$. Let $W = \frac{1}{s}\sum_{i=1}^s X_i$. For any $\epsilon > 0$, it holds that*

$$\Pr\{\|W\|_2 > \epsilon\} \le 2d \cdot e^{-s\epsilon^2/(\sigma^2 + \rho\epsilon/3)}.$$

Below we list a few useful inequalities regarding the function $f(x) = \log(1 + |x|)$.

Proposition 2. *For $x > 0$, it holds that $\ln(1+x) > x/(1+2x)$.*

Proposition 3. *It holds for all $x, y \in \mathbb{R}$ and all $a \ge 0$ that $f(x + y) \le f(x) + f(y)$ and $f(ax) \le af(x)$. As a consequence, for $x, y \in \mathbb{R}^n$ it holds that $\|f(x + y)\|_2^2 \le (\|f(x)\|_2 + \|f(y)\|_2)^2$.*

Proposition 4. *It holds for all $x, y \ge 0$ that $f(\sqrt{x^2 + y^2})^2 \le f(x)^2 + f(y)^2$.*

Lemma 5. *Let $a_1, \ldots, a_m$ be real numbers and $\epsilon_1, \ldots, \epsilon_m$ be 4-wise independent random variables on the set $\{-1, 1\}$ (i.e., Rademacher random variables). It holds that*

$$\mathbb{E}\, f\left(\sum_i \epsilon_i a_i\right)^2 \le C \sum_i f(a_i)^2.$$

where $C > 0$ is an absolute constant.

Lemma 6. *For arbitrary vectors $y, z \in \mathbb{R}^n$ such that $\|f(y)\|_2^2 \ge \xi^{-2}\|f(z)\|_2^2$ for some $\xi \in (0, 1)$, it holds that $(1 - 3\xi^{2/3})\|f(y)\|_2^2 \le \|f(y + z)\|_2^2 \le (1 + 3\xi)\|f(y)\|_2^2$.*

3. Algorithm

Our algorithm uses two important subroutines: a subsampling data structure called an H-Sketch, and a sketch for approximating the inner product of a transformed vector and a raw vector called LogSum. The former is inspired from a subsampling algorithm of (Levin et al., 2018) and is meant to sample a noisy approximation to a column from a distribution which is close to the desired distribution. In fact, one can show that it is impossible to sample the actual columns in a single pass, hence, we have to resort to noisy approximations and show they suffice. The latter LogSum sketch is the same as in (Liang et al., 2020), which approximates the inner product $\langle f(x), y\rangle$ for vectors x, y. Executing these sketches in parallel is highly non-trivial since the subsampling algorithm of (Levin et al., 2018) samples columns of A according to their ℓ_2 norms, but here we must sample them according to the squares of their ℓ_2 norms after applying f to each entry.

Algorithm 1 Basic heavy hitter substructure

Input: $A \in \mathbb{R}^{n\times n}$, ν, ϕ
Output: a data structure H

$w \leftarrow O(1/(\phi^2\nu^3))$
Prepare a pairwise independent hash function $h : [n] \to [w]$
Prepare 4-wise independent random signs $\{\epsilon_i\}_{i=1}^n$
Prepare a hash table H with w buckets, where each bucket stores a vector in $\mathbb{R}^n$.
for each $v \in [w]$ **do**
    $H_v \leftarrow \sum_{i\in h^{-1}(v)} \epsilon_i A_i$
end for
return H

Algorithm 2 Complete heavy hitter structure $\mathcal{D}$

Input: $A \in \mathbb{R}^{n\times n}$, ν, ϕ, δ
Output: a data structure H

$R \leftarrow O(\log(n/\delta))$
for each $r \in [R]$ **do**
    Initialize a basic substructure $H^{(r)}$ (Algorithm 1) with parameters ν and ϕ
end for

function QUERY(i)
    for each $r \in [R]$ **do**
        $v_r \leftarrow H^{(r)}_{h_r(i)}$ ▷ h_r is the hash function in $H^{(r)}$
    end for
    $r^* \leftarrow$ index r of the median of $\{\|f(v_r)\|_2\}_{r\in[R]}$
    return v_{r^*}
end function

Roughly speaking, the above combination gives us a small set of $\mathrm{poly}(k/\epsilon)$ noisy columns of $f(A)$, sampled approximately from the squared ℓ_2 norm of each column of $f(A)$, after which we can appeal to squared column-norm sampling results for low rank approximation in (Frieze et al., 2004), which argue that if you then compute the top-k left singular vectors of $f(A)$, forming the columns of the $n \times k$ matrix L, then $LL^\top f(A)$ is a good rank-k approximation to $f(A)$. The final output of the low-rank approximation will be two factors, L and $L^\top f(A)$. The algorithm in (Liang et al., 2020) first computes L by an involved algorithm in three passes, and then computes $L^\top f(A)$ in another pass using LOGSUM sketches. Our algorithm follows the same outline but we shall demonstrate how to compute L in only one pass, which is our sole focus for low-rank approximation in this paper. Note that our ultimate goal, which we only achieve approximately, is to sample columns of $f(A)$ proportional to their squared ℓ_2 norms. This is a fundamentally different sampling scheme from that of (Liang et al., 2020), which performs leverage score sampling followed by adaptive sampling, which are not amenable to a one-pass implementation.

3.1. H-Sketch

We first present a basic heavy hitter structure in Algorithm 1, and a complete heavy hitter structure in Algorithm 2 by repeating the basic structure R times. The complete heavy hitter structure supports a query function. Below we analyze the guarantee of this heavy hitter data structure.

Let $M = \|f(A)\|_F^2$. We define $I_\epsilon = \{i \in [n] : \|f(A_i)\|_2^2 \geq \epsilon M\}$, the set of the indices of the ϵ-heavy columns. Let α be a small constant to be determined later.

Lemma 7. *With probability at least* 0.9*, all columns in* $I_{\alpha\phi}$ *are isolated from each other under* h*.*

Lemma 8. *For each* $u \in [n]$*, it holds with probability at least* $2/3$ *that*

$$\left\| f\left(\sum_{i\notin(I_{\alpha\phi}\cup\{u\})} \mathbf{1}_{\{h(i)=h(u)\}}\epsilon_i A_i \right)\right\|_2^2 \leq 3C\frac{M}{w},$$

where $C > 0$ *is an absolute constant.*

Lemma 9. *Suppose that* $\nu \in (0, 0.05]$ *and* $\alpha = 0.3/C > \beta$*, where* C *is the absolute constant in Lemma 8. With probability at least* $1-\delta$*, for all* $i \in [n]$*, the output* v_{r^*} *of Algorithm 2 satisfies that*

(a) $(1-\nu)\|f(A_i)\|_2^2 \leq \|f(v_{r^*})\|_2^2 \leq (1+\nu)\|f(A_i)\|_2^2$ *for all* $i \in I_\phi$*;*

(b) $\|f(v_{r^*})\|_2^2 \leq 0.92\phi M$ *for all* $i \notin I_{\alpha\phi}$*;*

(c) $\|f(v_{r^*})\|_2^2 \leq (1+\nu^{3/2})^2\phi M$*.*

Proof. Fix $u \in [n]$. With probability at least $0.9 - 1/3 > 0.5$, the events in Lemmata 7 and 8 happen. Condition on those events.

From the proof of Lemma 8, we know that $i \in I_\phi$ do not collide with other elements in $I_{\alpha\phi}$. Hence, it follows from Lemma 6 (where $\xi^2 \leq 3C/(\phi w) \leq (\nu/3)^3$) that

$$(1-\nu)\|f(A_i)\|_2^2 \leq \left\|f(H_{h(i)})\right\|_2^2 \leq (1+\nu)\|f(A_i)\|_2^2,$$

provided that $w \geq 3^4C/(\phi\nu^3)$.

When $i \notin I_{\alpha\phi}$, we have

$$\begin{aligned}\left\|f(H_{h(i)})\right\|_2^2 &\leq 3C\left(\alpha\phi + \frac{1}{w}\right)M \leq (0.9+\nu^{3/2})\phi M \\ &\leq 0.92\phi M.\end{aligned}$$

Algorithm 3 Sampling using H-Sketch

Require: (i) An estimate $\widehat{M}$ such that $M \le \widehat{M} \le KM$; (ii) a complete heavy hitter structure $\mathcal{D}_0$ of parameters $(O(1), O(\epsilon^3/(KL^3)), 1/(\hat{L}+1))$; (iii) $\hat{L}$ complete heavy hitter structures (see Algorithm 1), denoted by $\mathcal{D}_1, \dots, \mathcal{D}_{\hat{L}}$, where $\mathcal{D}_j$ ($j \in [\hat{L}]$) has parameters $(O(1), O(\epsilon^3/L^3), 1/(\hat{L}+1))$ and is applied to the columns of A downsampled at rate 2^{-j};

$L \leftarrow \log(Kn/\epsilon)$, $\hat{L} \leftarrow \log n$
$\zeta \leftarrow$ a random variable uniformly distributed in $[1/2, 1]$
for $j = 0, \dots, \hat{L}$ **do**
  $\Lambda_j \leftarrow$ top $\Theta(L^3/\epsilon^3)$ heavy hitters from $\mathcal{D}_j$
end for
$j_0 \leftarrow \log(4K\epsilon^{-3}L^3)$
$\zeta \leftarrow$ uniform variable in $[1/2, 1]$
for $j = 1, \dots, j_0$ **do**
  Let $\lambda_1^{(j)}, \dots, \lambda_s^{(j)}$ be the elements in Λ_0 contained in $[(1+\epsilon)\zeta\frac{\widehat{M}}{2^j}, (2-\epsilon)\zeta\frac{\widehat{M}}{2^j}]$
  $\widetilde{M}_j \leftarrow |\lambda_1^{(j)}| + \cdots + |\lambda_s^{(j)}|$
end for
for $j = j_0 + 1, \dots, L$ **do**
  Find the largest ℓ for which Λ_ℓ contains s_j elements $\lambda_1^{(j)}, \dots, \lambda_{s_j}^{(j)}$ in $[(1+\epsilon)\zeta\frac{\widehat{M}}{2^j}, (2-\epsilon)\zeta\frac{\widehat{M}}{2^j}]$ for $(1-\sqrt{20}\epsilon)L^2/\epsilon^2 \le s_j \le 2(1+\sqrt{20}\epsilon)L^2/\epsilon^2$
  if such an ℓ exists **then**
    $\widetilde{M}_j \leftarrow (|\lambda_1^{(j)}| + \cdots + |\lambda_{s_j}^{(j)}|)2^\ell$
    $W_j \leftarrow \Lambda_\ell$
  else
    $\widetilde{M}_j \leftarrow 0$
  end if
end for
$j^* \leftarrow$ sample from $[L]$ according to pdf $\Pr(j^* = j) = \widetilde{M}_j / \sum_j \widetilde{M}_j$
$i^* \leftarrow$ sample from W_{j^*} according to pdf $\Pr(i^* = i) = |\lambda_i^{(j)}|/\widetilde{M}_j$
$v_{j^*,i^*} \leftarrow$ vector returned by QUERY(i^*) on $\mathcal{D}_{j^*}$
return v_{j^*,i^*}

When $i \in I_{\alpha\phi} \setminus I_\phi$, we have that H_i contains only i and columns from $[n] \setminus I_{\alpha\phi}$. Hence by Proposition 3,

$$\begin{aligned}\left\|f(H_{h(i)})\right\|_2^2 &\le \left(\|f(A_i)\|_2 + \sqrt{\frac{3C}{w}M}\right)^2 \\ &\le \left(\sqrt{\phi M} + \sqrt{\nu^3 \phi M}\right)^2 \\ &\le (1+\nu^{3/2})^2 \phi M,\end{aligned}$$

provided that $w \ge 3C/(\phi\nu^3)$.

Finally, repeating $O(\log(n/\delta))$ times and taking the median and a union bound over all n columns gives the claimed result. □

Next we analyze the sampling algorithm, presented in Algorithm 3, which simulates sampling a column from A according to the column norms. The following theorem is our guarantee.

Theorem 10. *Let $\epsilon > 0$ be a constant small enough. Algorithm 3 outputs v_{j^*,i^*} which satisfies that, with probability at least 0.9, there exists $u \in [n]$ such that*

$$\begin{aligned}(1 - O(\epsilon))\left\|f(A_u)\right\|_2^2 \le \left\|f(v_{j^*,i^*})\right\|_2^2 \\ \le (1 + O(\epsilon))\left\|f(A_u)\right\|_2^2.\end{aligned}$$

Furthermore, there exists an absolute constant $c \in (0, 1/2]$ such that

$$\Pr\{u = i\} \ge c\frac{\|f(A_i)\|_2^2}{\|f(A)\|_F^2}$$

for all i belonging to some set $I \subseteq [n]$ such that $\sum_{i\in I}\|f(A_i)\|_2^2 \ge (1-6\epsilon)M$, provided that ϵ further satisfies that $\epsilon \le c/C$ for some absolute constant $C > 0$.

Proof. The analysis of the algorithm is largely classical, for which we define the following notions:

(1) $T_j = \zeta M/2^j$;

(2) $S_j = \left\{i \in [n] : \|f(A_i)\|_2^2 \in (T_j, 2T_j]\right\}$ is the j-th *level set* of A;

(3) a level $j \in [L]$ is *important* if $|S_j| \ge \epsilon 2^j/L$;

(4) $\mathcal{J} \subseteq [L]$ is the set of all important levels.

It follows from the argument in (Li et al., 2021), or an argument similar to (Andoni et al., 2009) that the columns we miss contribute to only an $O(\epsilon)$-fraction of the norm, and for each level $j \in \{1, \dots, j_0\} \cup \mathcal{J}$, each of the recovered columns λ_i ($i \in [s_j]$) corresponds to some $u = u(i) \in S_j$ and satisfies that $(1-O(\epsilon))\|f(A_{u_i})\|_2^2 \le \lambda_i \le (1+O(\epsilon))\|f(A_{u_i})\|_2^2$.

Next we prove the second part. For a fixed $i \in [n]$, define events

$$\mathcal{E}_i = \{i \text{ falls in a level } j \in [j_0] \cup \mathcal{J}\}$$

and a set of "good" columns

$$I = \{i : \Pr\{\mathcal{E}_i\} \ge \beta\}$$

for some constant $\beta \le 1/2$. Since all non-important levels always contribute to at most a 2ϵ-fraction of M, it follows that the bad columns contribute to at most a $2\epsilon/(1-\beta)$-fraction of M, that is,

$$\sum_{i\notin I}\|f(A_i)\|_2^2 \le \frac{2\epsilon}{1-\beta}\cdot M.$$

Next we define the event that

$$\mathcal{F}_i = \{\mathcal{E}_i \text{ and } \|f(A_i)\|_2^2 \in [(1+\epsilon)T_j, 2(1-\epsilon)T_j]\},$$

then it holds for all $i \in I$ that $\Pr\{\mathcal{F}_i\} \geq \Pr\{\mathcal{E}_i\} - O(\epsilon) \geq 0.9\beta$ for ϵ sufficiently small.

Let $\mathcal{G}_j$ denote the event that the magnitude level j is chosen, and $j(i)$ is the index of the magnitude level containing column i. Then for those i's with $j = j(i) \in [j_0] \cup \mathcal{J}$,

$$\Pr\{\mathcal{G}_j|\mathcal{F}_i\} = \frac{(1 \pm O(\epsilon))\widetilde{M}_j}{(1 \pm O(\epsilon))\sum_j \widetilde{M}_j} = \frac{(1 \pm O(\epsilon))M_j}{(1 \pm O(\epsilon))M}$$
$$= (1 \pm O(\epsilon))\frac{M_j}{M}$$

and

$$\Pr\{u = i \mid \mathcal{G}_j \cap \mathcal{F}_i\} = \frac{\lambda_t^{(j)}}{\widetilde{M}_j} = \frac{1 \pm O(\epsilon)\|f(A_i)\|_2^2}{(1 \pm O(\epsilon))M_j}$$
$$= (1 \pm O(\epsilon))\frac{\|f(A_i)\|_2^2}{M_j}$$

Hence

$$\Pr\{u = i\} = \Pr\{u = i \mid \mathcal{G}_j \cap \mathcal{F}_i\}\Pr\{\mathcal{G}_j|\mathcal{F}_i\}\Pr\{\mathcal{F}_i\}$$
$$\geq 0.9\beta(1 - O(\epsilon))\frac{\|f(A_i)\|_2^2}{M} \geq 0.8\beta\frac{\|f(A_i)\|_2^2}{M},$$

provided that ϵ is sufficiently small. □

Now we show how to obtain an overestimate $\widehat{M}$ for M. We assume that all entries of A are integer multiples of $\eta = 1/\operatorname{poly}(n)$ and are bounded by $\operatorname{poly}(n)$, which is a common and necessary assumption for streaming algorithms, otherwise storing a single number would take too much space. Let $\tilde{f}(x) = \log^2(1 + |\eta x|)$, then $\|f(A)\|_F^2 = \sum_{i,j} \tilde{f}(\eta^{-1}A)$, where $\eta^{-1}A$ has integer entries. Hence, we can run the algorithm implied by Theorem 2 of (Braverman et al., 2016) on $\eta^{-1}A$ in parallel in order to obtain a constant-factor estimate to $\|f(A)\|_F^2$. To justify this application of the theorem, we verify in Appendix B.3 that the function $\tilde{f}(|x|)$ is slow-jumping, slow-dropping and predictable on nonnegative integers as defined by (Braverman et al., 2016).

Finally, we calculate the sketch length. The overall sketch length is dominated by that of Algorithm 3. In Algorithm 3, there are $\hat{L} = O(\epsilon^{-1}\log n)$ heavy hitter structures $\mathcal{D}_1, \ldots, \mathcal{D}_{\hat{L}}$, each of which has a sketch length of $O(1/(\phi^2\nu^3)\log(nL)) = \operatorname{poly}(L, 1/\epsilon, \log n) = \operatorname{poly}(\log n, 1/\epsilon)$. There is an additional heavy hitter structure $\mathcal{D}_0$ of sketch length $O(\operatorname{poly}(K, L, 1/\epsilon, \log n)) = \operatorname{poly}(\log n, 1/\epsilon)$. Hence the overall sketch length is $\operatorname{poly}(\log n, 1/\epsilon)$. Each cell of the sketch stores an n-dimensional vector. We summarize this in the following theorem.

Theorem 11. *Suppose that $A \in (\eta\mathbb{Z})^{n\times n}$ with $|A_{ij}| \leq \operatorname{poly}(n)$ is given in a turnstile stream, where $\eta = 1/\operatorname{poly}(n)$. There exists a randomized sketching algorithm which maintains a sketch of $n\operatorname{poly}(\epsilon^{-1}\log n)$ space and outputs a vector $v_{j^*,i^*} \in \mathbb{R}^n$ which satisfies the same guarantee as given in Theorem 10.*

3.2. Low-Rank Approximation

Suppose that we have an approximate sampling of the rows of $f(A)$ so that we obtain a sample $f(A_i) + E_i$ with probability p_i satisfying

$$p_i \geq c\frac{\|f(A_i)\|_2^2}{\|f(A)\|_F^2} \tag{1}$$

for some absolute constant $c \leq 1$. The p_i's are known to us (if $c = 1$, then we do not need to know the p_i).

The following is our main theorem in this section, which is analogous to Theorem 2 of (Frieze et al., 2004).

Theorem 12. *Let V denote the subspace spanned by s samples drawn independently according to the distribution (1), where each sample has the form $f(A_i) + E_i$ for some $i \in [n]$. Suppose that $\|E_i\|_2 \leq \gamma\|f(A_i)\|_2$ for some $\gamma > 0$. Then with probability at least $9/10$, there exists an orthonormal set of vectors $y_1, y_2, \ldots, y_k$ in V such that*

$$\left\|f(A) - f(A)\sum_{j=1}^k y_j y_j^\top\right\|_F^2$$
$$\leq \min_{D:\operatorname{rank}(D)\leq k} \|f(A) - D\|_F^2 + \frac{10k}{sc}(1+\gamma)^2\|f(A)\|_F^2.$$

The theorem shows that the subspace spanned by a sample of columns chosen according to (1) contains an approximation to $f(A)$ that is nearly the best possible. Note that if the top k right singular vectors of S belong to this subspace, then $f(A)\sum_{t=1}^k v_t v_t^\top$ would provide the required approximation to $f(A)$ and we would be done.

Now, the difference between Theorem 10 and the assumption (1) is that we do not have control over p_i for an $O(\epsilon)$-fraction of the rows (in squared row norm contribution) in Theorem 10. Let A' be the submatrix of A after removing those rows, then $\|f(A)\|_F \leq (1 + O(\epsilon))\|f(A')\|_F$. We can apply Theorem 12 to A' and take more samples such that we obtain s rows from A' (which holds with $1 - \exp(-\Omega(s))$ probability by a Chernoff bound). We therefore have the following corollary.

Corollary 13. *Let y_i's be as in Algorithm 4 and c and ϵ be as in Theorem 10. It holds with probability at least 0.7 that*

$$\left\|f(A) - f(A)\sum_j y_j y_j^\top\right\|_F^2$$

Algorithm 4 Rank-k Approximation

Input: $A \in \mathbb{R}^{n\times n}$, rank parameter k, number of samples s

1: Initialize s parallel instances of (modified) Algorithm 3
2: Let $(h_1, \hat{p}_1), \ldots, (h_q, \hat{p}_q)$ be the returned vectors and the sampling probability from the s instances of (modified) Algorithm 3
3: $F \leftarrow$ concatenated matrix $\left(\frac{h_1}{\sqrt{s\hat{p}_1}} \quad \cdots \quad \frac{h_s}{s\hat{p}_s}\right)$
4: Compute the top k left singular vectors of F, forming $L \in \mathbb{R}^{n\times k}$
5: **return** L

$$\leq \min_{D:\mathrm{rank}(D)\leq k} \|f(A) - D\|_F^2 + \left(\frac{30k}{sc} + \epsilon\right)\|f(A)\|_F^2 .$$

Note that Algorithm 3 can be easily modified to return the sampling probability of the sampled column, which is just $\lambda_i^{(j)} / \sum_j \widetilde{M}_j$. However, for each sample, we may lose control of it with a small constant probability. To overcome this, inspecting the proof of H-Sketch, we see that for fixed stream downsampling and fixed ζ in Algorithm 3, repeating each heavy hitter structure $O(\log(L/\delta))$ times and taking the median of each $\widetilde{M}_j$ will lower the failure probability of estimating the contribution of each important level to $\delta/(L+1)$, allowing for a union bound over all levels. Hence, with probability at least $1-\delta$, we can guarantee that we obtain a $(1 \pm O(\epsilon))$-approximation to $\|(f(A))_u\|_2^2$ and thus a $(1 \pm O(\epsilon))$-approximation to $\|f(A)\|_F^2$. Hence the returned $\hat{p}_u$ is a $(1 \pm O(\epsilon))$-approximation to the true row-sampling probability $p_u = \|(f(A))_u\|_2^2 / \|f(A)\|_F^2$. Different runs of the sampling algorithm may produce different values of $\hat{p}_u$ for the same u but they are all $(1 \pm O(\epsilon))$-approximations to p_u. We can guarantee this for all our s samples by setting $\delta = O(1/s)$, which allows for a union bound over all s samples.

Therefore, at the cost of an extra $O(\log s)$ factor in space, we can assume that $\hat{p}_u = (1 \pm O(\epsilon))p_u$ for all s samples. The overall algorithm is presented in Algorithm 4.

The following main theorem follows from Corollary 13 and the argument in (Frieze et al., 2004).

Theorem 14. *Let $s = O(k/\epsilon)$ be the number of samples and $y_1, \ldots, y_k$ be the output of Algorithm 4. It holds with probability at least 0.7 that*

$$\left\|f(A) - LL^\top f(A)\right\|_F^2 \leq \min_{D:\mathrm{rank}(D)\leq k} \|f(A) - D\|_F^2 + \epsilon\|f(A)\|_F^2 .$$

4. Experiments

To demonstrate the benefits of our algorithm empirically, we conducted experiments on low-rank approximation with a real NLP data set and used the function $f(x) = \log(|x| + 1)$.

The data we use is based on the Wikipedia data used by Liang et al. (2020). The data matrix $A' \in \mathbb{R}^{n\times n}$ ($n = 10^4$) contains information about the correlation among the n words. Its entries are $A'_{i,j} = p_j \log(N_{i,j}N/(N_iN_j)+1)$, where $N_{i,j}$ is the number of times words i and j co-occur in a window size of 10, N_i is the number of times word i appears and N_i's are in a decreasing order, $N = \sum_i N_i$ and $p_j = \max\{1, (N_j/N_{10})^2\}$ is a weighting factor which adds larger weights to more frequent words. Since A', and thus $f(A')$, have almost the same column norms, we instead consider $A = A' - \mathbf{1}\mathbf{1}^\top A'$, where $\mathbf{1}$ is the vector of all 1 entries.

We compare the accuracy and runtime with the previous three-pass algorithm of Liang et al. (2020). The task is to find $L \in \mathbb{R}^{n\times k}$ with orthornormal columns to minimize the error ratio

$$e(L) = \frac{\left\|f(A) - LL^\top f(A)\right\|_F}{\|f(A) - UU^\top f(A)\|_F},$$

where $U \in \mathbb{R}^{n\times k}$ has the top k left singular vectors of $f(A)$ as columns. The numerator $\|f(A) - LL^\top f(A)\|_F$ is the approximation error and the denominator $\|f(A) - UU^\top f(A)\|_F$ is the best approximation error, both in Frobenius norm.

4.1. Algorithm Implementation

We present our empirical results for the one-pass algorithm and a faster implementation of the two-pass algorithm in Figure 1 and Figure 2. Both algorithms run in 0.1% of the runtime of Liang et al. (2020)'s three-pass algorithm. The one-pass algorithm is less accurate than the two-pass algorithm when the space usage is small, which is not unexpected, because the second pass enables noiseless column samples. Still, as discussed below, one-pass algorithms are essential in certain internet NLP applications. Even our two-pass algorithm has a considerable advantage over the prior three-pass algorithm, by matching its accuracy in significantly less time and one fewer pass.

For the two-pass algorithm, we sample the columns of A using Algorithm 3 in the first pass and only recover the positions of the heavy columns in each magnitude level. Taking s samples will incur $O(s\,\mathrm{poly}(\epsilon^{-1}\log n))$ heavy hitters in total. In the second pass we obtain precise $f(A)$ values for our samples and thus noiseless column samples. Then we calculate the sampling probability $\hat{p}_u$ according to the error-free column norms.

To reduce the runtime, we do not run s independent copies of Algorithm 3 for s samples; instead, we take m samples from each single run of Algorithm 3 and run s/m indepen-

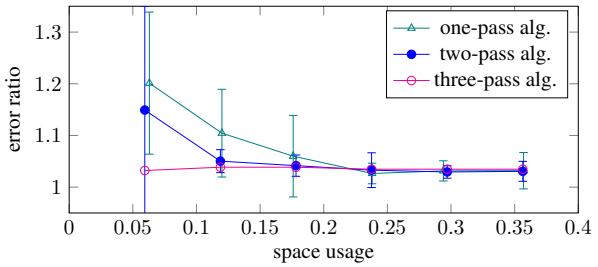

Figure 1. Error ratios of the one-pass, two-pass and three-pass algorithms. The x-axis is the ratio between the space of the sketch maintained by the tested algorithm and the space to store the input matrix. The y-axis is the error ratio $e(L)$. Solid dots denote the mean of the error ratios over 10 independent trials and the vertical bars denote the standard deviation of the one-pass and two-pass algorithms.

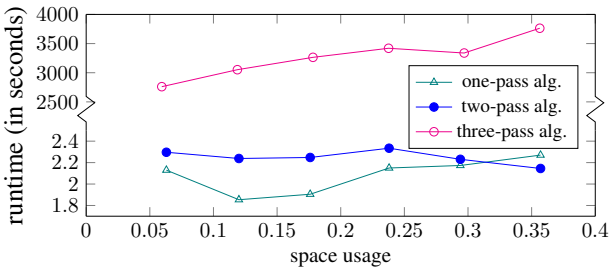

Figure 2. Runtimes of the one-pass, two-pass and three-pass algorithms. The x-axis is the ratio between the space of the sketch maintained by the tested algorithm and the space to store the input matrix. The y-axis is the runtime in seconds. Solid dots denote the mean of 10 independent trials of the one-pass and two-pass algorithms. The standard deviations are all less than 1.5 seconds and thus omitted.

dent copies of Algorithm 3. Hence the s samples we obtain are not fully independent.

All experiments are conducted under MATLAB 2019b on a laptop with a 2.20GHz CPU and 16GB RAM.

We set $k = 10$, $m = 100$ and plot the error ratios of our algorithm in Figure 1. For each value of space usage, the mean and standard deviation are reported from 10 independent runs. In the same figure, we also plot the results of the three-pass algorithm of Liang et al. (2020) at comparable levels of space usage. Since the three-pass algorithm is considerably slower, we run the three-pass algorithm only once. Additionally we plot the runtimes of all algorithms in Figure 2.

We can observe that even at the space usage of approximately 12% of the input data, the error ratio of our two-pass algorithm is stably around 1.05. The one-pass algorithm is less accurate than the one-pass algorithm when the space usage is less than 20%, which is not unexpected, because the second pass enables noiseless column samples. Overall, the error ratio of the one- and two-pass algorithms is similar to that of the three-pass algorithm for space usage level at least 0.2, while the runtime of both algorithms is at most 0.1% of that of the three-pass algorithm, which is a significant improvement.

5. Application to Linear Regression

In this section, we consider approximately solving the linear regression problem using the H-Sketch from Section 3.1.

We shall need to sample rows from the concatenated matrix $Q = \begin{pmatrix} f(A) & b \end{pmatrix}$, where $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ are given in a turnstile stream, and $f(x) = \ln(1 + |x|)$ is the transformation function. This can still be achieved using the same H-Sketch in Section 3.1, applied to the concatenated matrix $\begin{pmatrix} A & b \end{pmatrix}$, with the transformation function $f(x)$ $(x \in \mathbb{R}^{d+1})$ replaced with $g(x) = \begin{pmatrix} f(x_{1:d}) & x_{d+1} \end{pmatrix}$, where $x_{1:d}$ denotes the first d coordinate of x and x_{d+1} the last coordinate of x. Then, using identical arguments in Section 3.1 for the squared ℓ_2-norm on the first d coordinates and a standard Count-Sketch argument for the last coordinate, it is straightforward to show an analogous version of Theorem 10 as below. We omit the identical proof.

Theorem 15. *Let $\epsilon > 0$, and let H_{j_0,i_0} be the vector returned by Algorithm 3. With probability at least 0.9, it holds that there exists $u \in [n]$ such that*

$$(1 - O(\epsilon)) \left(\|f(A_u)\|_2^2 + |b_u|^2 \right) \le \|g(H_{j_0,i_0})\|_2^2 \le (1 + O(\epsilon)) \left(\|f(A_u)\|_2^2 + |b_u|^2 \right).$$

Theorem 15 states that each sample is a noisy version of the u-th row of Q. Let $p_u = \|Q_u\|_2^2 / \|Q\|_F^2$ be the true sampling probability of the u-th row. As argued at the beginning at Section 3.2, we may assume, at the cost of an $O(\log s)$ factor in space, that every sample is good, i.e., the returned sampling probability $\hat{p}_u$ satisfies that $\hat{p}_u = (1 \pm O(\epsilon))p_u$ and the noise in each sample is at most an $O(\epsilon)$-fraction in its ℓ_2 norm.

Below we present our algorithm for linear regression in Algorithm 5, assuming that every sample is good in the sense that $\hat{p}_u = (1 \pm O(\epsilon))p_u$ and the noise in each sample is at most an $O(\epsilon)$-fraction in ℓ_2-norm. The guarantee is given in Theorem 16 and the proof is deferred to Section D.

Theorem 16. *Given matrix $A \in \mathbb{R}^{n\times d}$ and vector $b \in \mathbb{R}^n$, let $\kappa = \kappa(f(A))$ be the condition number of the transformed matrix. Let $s = O(\frac{d\kappa^2}{\epsilon^2} \log \frac{d}{\delta})$, then Algorithm 5 outputs a vector $\tilde{x} \in \mathbb{R}^d$, which, with probability at least $1 - \delta$, satisfies*

$$\left\| \tilde{M}\tilde{x} - \tilde{b} \right\|_2 \le (1 + \epsilon) \min_{x\in\mathbb{R}^d} \|f(A)x - b\|_2 + \Delta,$$

Algorithm 5 Linear Regression

Require: $A \in \mathbb{R}^{n\times d}$, number of samples s

1: Initialize s parallel instances of Algorithm 3
2: Run Algorithm 3 on the concatenated matrix $\begin{pmatrix} f(A) & b \end{pmatrix}$ to obtain s row samples $h_1, \ldots, h_s$ and the corresponding sampling probabilities $\hat{p}_1, \ldots, \hat{p}_s$
3: $T \leftarrow$ vertical concatenation of $\frac{h_1}{\sqrt{s\hat{p}_1}}, \ldots, \frac{h_s}{\sqrt{s\hat{p}_s}}$
4: $\tilde{M} \leftarrow$ first d columns of T, $\tilde{b} \leftarrow$ last column of T
5: $\tilde{x} \leftarrow \arg\min_{x\in\mathbb{R}^d} \|\tilde{M}x - \tilde{b}\|_2$

where

$$\Delta = \epsilon\left(\sqrt{d + \frac{\|b\|_2^2}{\|f(A)\|_2^2}}\kappa\,\|b\|_2 + \sqrt{\|f(A)\|_F^2 + \|b\|_2^2}\right).$$

The total space used by Algorithm 5 is $\tilde{O}(d^2\kappa^2 \log\frac{1}{\delta}) \cdot \mathrm{poly}(\log n, \frac{1}{\epsilon})$.

Finally, we note an error in (Liang et al., 2020). Let $G = f(A)$ and S be a subspace embedding sketching matrix of $s = \mathrm{poly}(d/\epsilon)$ rows for the column space of G. In their proof of the regression problem in Section E, the upper bound of C_2 is wrong as it claims that $\|(\widetilde{SG})^\dagger\|_2 \le C\|(SG)^\dagger\|_2$; a correct bound should be $\|(\widetilde{SG})^\dagger\|_2 \le 10nd\kappa\eta\|(SG)^\dagger\|_2$, where $\eta = \max_{i,j}|(\widetilde{SG})_{i,j} - (SG)_{i,j}|$ could be linear in $\sqrt{n}$ by their LOGSUM guarantee. The proof in (Liang et al., 2020) does not account for such a dependence on n and κ. Correcting the proof would yield a similar bound as ours but with an addtive error $\Delta = n^2\kappa^3\,\mathrm{poly}(\frac{\epsilon}{d})\|b\|_2$, which depends polynomially on n. Our additive error has no dependence on n but depends on $\|b\|_2^2/\|f(A)\|_2^2$ and has an additional additive term of $\epsilon\|Q\|_2$, which is an artefact of sampling the rows of Q.

Acknowledgements

Y. Li was partially supported and Y. Sun was supported by Singapore Ministry of Education (AcRF) Tier 2 grant MOE2018-T2-1-013. D. Woodruff thanks support from NSF grant No. CCF-1815840, Office of Naval Research grant N00014-18-1-256, and a Simons Investigator Award.

References

Andoni, A., Ba, K. D., Indyk, P., and Woodruff, D. P. Efficient sketches for earth-mover distance, with applications. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pp. 324–330, 2009.

Andoni, A., Krauthgamer, R., and Onak, K. Streaming algorithms via precision sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 363–372, 2011. doi: 10.1109/FOCS.2011.82.

Boutsidis, C., Woodruff, D. P., and Zhong, P. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pp. 236–249, 2016.

Braverman, V., Chestnut, S. R., Woodruff, D. P., and Yang, L. F. Streaming space complexity of nearly all functions of one variable on frequency vectors. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pp. 261–276, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341912. doi: 10.1145/2902251.2902282. URL https://doi.org/10.1145/2902251.2902282.

Clarkson, K. L. and Woodruff, D. P. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 205–214, 2009.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13: 3475–3506, 2012.

Durme, B. V. and Lall, A. Streaming pointwise mutual information. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, pp. 1892–1900, Red Hook, NY, USA, 2009. Curran Associates Inc. ISBN 9781615679119.

Frieze, A. M., Kannan, R., and Vempala, S. S. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

Levin, R., Sevekari, A. P., and Woodruff, D. P. Robust subspace approximation in a stream. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 10706–10716, 2018.

Li, Y., Woodruff, D., and Yasuda, T. Exponentially improved dimensionality reduction for ℓ_1: Subspace embeddings and independence testing. Accepted to *COLT* 2021. Full version available at arXiv:2104.12946 [cs.DS], 2021.

Liang, Y., Song, Z., Wang, M., Yang, L., and Yang, X. Sketching transformed matrices with applications to natural language processing. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, pp. 467–481, 2020.

Mahabadi, S., Razenshteyn, I. P., Woodruff, D. P., and Zhou, S. Non-adaptive adaptive sampling on turnstile streams. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pp. 1251–1264, 2020.

Upadhyay, J. Differentially private linear algebra in the streaming model. *arXiv preprint arXiv:1409.5414*, 2014.

Woodruff, D. P. Low rank approximation lower bounds in row-update streams. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 1781–1789, 2014.

Woodruff, D. P. and Zhong, P. Distributed low rank approximation of implicit functions of a matrix. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 847–858. IEEE, 2016.