Non-adaptive Adaptive Sampling on Turnstile Streams

Sepideh Mahabadi
Toyota Technological Institute at Chicago (TTIC)
Chicago, USA
mahabadi@ttic.edu

David P. Woodruff
Carnegie Mellon University
Pittsburgh, USA
dwoodruf@cs.cmu.edu

Ilya Razenshteyn
Microsoft Research
Redmond, USA
ilyaraz@microsoft.com

Samson Zhou
Carnegie Mellon University
Pittsburgh, USA
samsonzhou@gmail.com

ABSTRACT

Adaptive sampling is a useful algorithmic tool for data summarization problems in the classical centralized setting, where the entire dataset is available to the single processor performing the computation. Adaptive sampling repeatedly selects rows of an underlying matrix \( A \in \mathbb{R}^{n \times d} \), where \( n \gg d \), with probabilities proportional to their distances to the subspace of the previously selected rows. Intuitively, adaptive sampling seems to be limited to trivial multi-pass algorithms in the streaming model of computation due to its inherently sequential nature of assigning sampling probabilities to each row only after the previous iteration is completed. Surprisingly, we show this is not the case by giving the first one-pass algorithms for adaptive sampling on turnstile streams and using space \( \text{poly}(d, k, \log n) \), where \( k \) is the number of adaptive sampling rounds to be performed.

Our adaptive sampling procedure has a number of applications to various data summarization problems that either improve state-of-the-art or have only been previously studied in the more relaxed row-arrival model. We give the first relative-error algorithm for column subset selection on turnstile streams and using space \( \text{poly}(d, k, \log n) \), where \( k \) is the number of adaptive sampling rounds to be performed.

Our adaptive sampling procedure has a number of applications to various data summarization problems that either improve state-of-the-art or have only been previously studied in the more relaxed row-arrival model. We give the first relative-error algorithm for column subset selection on turnstile streams and using space \( \text{poly}(d, k, \log n) \), where \( k \) is the number of adaptive sampling rounds to be performed.

For volume maximization in the row-arrival model, which we match with competitive upper bounds.

CCS CONCEPTS

- Theory of computation → Streaming, sublinear and near linear time algorithms; Randomness, geometry and discrete structures; Theory and algorithms for application domains.

KEYWORDS

streaming algorithms, computational geometry, determinantal point processes, volume maximization

1 INTRODUCTION

Data summarization is a fundamental task in data mining, machine learning, statistics, and applied mathematics. The goal is to find a set \( S \) of \( k \) rows of a matrix \( A \in \mathbb{R}^{n \times d} \) that optimizes some predetermined function that quantifies how well \( S \) represents \( A \). For example, row subset selection seeks \( S \) to well-approximate \( A \) with respect to the spectral or Frobenius norm, subspace approximation asks to minimize the sum of the distances of the rows of \( A \) from \( S \), while volume maximization wants to maximize the volume of the parallelepiped spanned by the rows of \( S \). Due to their applications in data science, data summarization problems are particularly attractive to study for big data models.

The streaming model of computation is an increasingly popular model for describing large datasets whose overwhelming size places restrictions on space available to algorithms. For turnstile streams that implicitly define \( A \), the matrix initially starts as the all zeros matrix and receives a large number of updates to its coordinates. Once the updates are processed, they cannot be accessed again and hence any information not stored is lost forever. The goal is then to perform some data summarization task after the stream is completed without storing \( A \) in its entirety.

Adaptive sampling is a useful algorithmic paradigm that yields many data summarization algorithms in the centralized setting \([15–17]\). The idea is that \( S \) begins as the empty set and some row \( A_j \)
of $A$ is sampled with probability $\|A_i\|^p_{p,2}$ where $p \in \{1, 2\}$ and $\|A\|_{p,q} = \left(\sum_{i=1}^{\|A\|_p} \left(\sum_{j=1}^{d} |A_{ij}|^q \right)^{\frac{p}{q}} \right)^{\frac{1}{p}}$. As $S$ is populated, the algorithm adaptively samples rows of $A$, so that at each iteration, row $A_i$ is sampled with probability proportional to the $p^{th}$ power of the distance of the row from $S$. That is, $A_i$ is sampled with probability $\frac{\|A_i - M_k M_k^* A\|^p_{p,2}}{\|A - M_k M_k^* A\|^p_{p,2}}$, where $M$ is the matrix formed by the rows in $S$. The procedure is repeated $k$ times until we obtain $k$ rows of $A$, which then forms our summary of the matrix $A$. Unfortunately, adaptive sampling seems like an inherently sequential procedure and thus the extent of its capabilities has not been explored in the streaming model.

1.1 Our Contributions

In this paper, we show that although adaptive sampling seems like an iterative procedure, we do not need multiple passes over the stream to perform adaptive sampling. This is particularly surprising since any row $A_i$ of $A$ can be made irrelevant, i.e., zero probability of being sampled, in future rounds if some row along the same direction of $A_i$ is sampled in the present round. Yet somehow we must still output rows of $A$ while storing a sublinear number of rows more or less non-adaptively. The challenge seems compounded by the turnstile model, since updates can be made to arbitrary elements more or less non-adaptively. The challenge seems compounded by the turnstile model, since updates can be made to arbitrary elements.

Let $A \in \mathbb{R}^{d \times d}$ that arrives in a turnstile stream and post-processing query access to a matrix $P \in \mathbb{R}^{d \times d}$, we first give $L_{p,q}$ samplers for $AP$.

**Theorem 1.1.** Let $\epsilon > 0$, $q = 2$, and $p \in \{1, 2\}$. There exists a one-pass streaming algorithm that takes rows of a matrix $A \in \mathbb{R}^{n \times d}$ as a turnstile stream and post-processing query access to matrix $P \in \mathbb{R}^{d \times d}$ after the stream, and with high probability, samples an index $i \in [n]$ with probability $(1 \pm O(\epsilon))$ $\frac{\|AP\|^p_{p,2}}{\|A\|^p_{p,2}} + \frac{1}{\text{poly}(n)}$. The algorithm uses $\text{poly}(d, \frac{1}{\epsilon}, \log n)$ bits of space.

We remark that our techniques can be extended to $p \in (0, 2]$ but we only require $p \in \{1, 2\}$ for the purposes of our applications. Now, suppose we want to perform adaptive sampling of a row of $A \in \mathbb{R}^{n \times d}$ with probability proportional to its distance or squared distance from some subspace $H \in \mathbb{R}^{k \times d}$, where $i$ is any integer. Then by taking $P = I - H\mathbb{H}$ and either $L_{1,2}$ or $L_{2,2}$ sampling, we select rows of $A$ with probability roughly proportional to the distance or squared distance from $H$. We can thus simulate $k$ rounds of adaptive sampling in a stream, despite its seemingly inherent sequential nature.

**Theorem 1.2.** Let $A \in \mathbb{R}^{n \times d}$ be a matrix and $q \in [n]$ be the probability of selecting a set $S \subset [n]$ of $k$ rows of $A$ according to $k$ rounds of adaptive sampling with respect to either the distances to the selected subspace in each iteration or the squared distances to the selected subspace in each iteration. There exists an algorithm that takes inputs $A$ through a turnstile stream and $\epsilon > 0$, and outputs a set $S \subset [n]$ of $k$ indices such that if $p_S$ is the probability of the algorithm outputting $S$, then $\sum_S |p_S - q_S| \leq \epsilon$. The algorithm uses $\text{poly}\left(d, k, \frac{1}{\epsilon}, \log n\right)$ bits of space.

In other words, our output distribution is close in total variation distance to the desired adaptive sampling distribution.

Our algorithm is the first to perform adaptive sampling on a stream; existing implementations require extended access to the matrix, such as in the centralized or distributed models, for subsequent rounds of sampling. Moreover, if the set $S$ of $k$ indices output by our algorithm is $s_1, \ldots, s_k$, then our algorithm also returns a set of rows $r_1, \ldots, r_k$ so that if $R_0 = \emptyset$ and $R_t = r_1 \circ \ldots \circ r_t$ for $i \in [k]$, then $r_i = u_{s_i} + v_i$, where $u_{s_i} = A_{s_i}(I - R_i R_i^\dagger)$ is the projection of the sampled row to the space orthogonal to the previously selected rows, and $v_i$ is some small noisy vector formed by linear combinations of other rows in $A$ such that $\|v_i\|_2 \leq \epsilon \|u_s\|_2$.

Thus we do not return the true rows of $A$ corresponding to the indices in $S$, but we output a small noisy perturbation to each of the rows, which we call noisy rows and suffices for a number of applications previously unexplored in turnstile streams. Crucially, the noisy perturbation in each of our output rows can be bounded in norm not only relative to the norm of the true row, but also relative to the residual. In fact, our previous example of a long stream of small updates followed a single arbitrarily large update shows that it is impossible to return the true rows of $A$ in sublinear space. Since the arbitrarily large update can apply to any entry of the matrix, the only way an algorithm can return the entire row containing the entry is if the entire matrix is stored.

**Column subset selection.** In the row/column subset selection problem, the inputs are the matrix $A \in \mathbb{R}^{n \times d}$ and an integer $k > 0$, and the goal is to select $k$ rows/columns of $A$ to form a matrix $M$ to minimize $\|A - AM^\dagger M\|_F$ or $\|A - MM^\dagger A\|_F^2$. For the sake of presentation, we focus on the row subset selection problem for the remainder of this section. Since the matrix $M$ has rank at most $k$, then $\|A - AM^\dagger M\|_F \geq \|A - A_k^\dagger\|_F$ where $A_k^\dagger$ is the best rank $k$ approximation to $A$. Hence, we would ideally like to obtain some guarantee for $\|A - AM^\dagger M\|_F$ relative to $\|A - A_k\|_F$. Such relative error algorithms were given in the centralized setting [8, 15, 19] and for row-arrival streams [14], but no such results were previously known for turnstile streams. Our adaptive sampling framework thus provides the first algorithm on turnstile streams with relative error guarantees.

**Theorem 1.3.** Given a matrix $A \in \mathbb{R}^{n \times d}$ that arrives in a turnstile data stream, there exists a one-pass algorithm that outputs a set $M$ of $k$ (noisy) rows of $A$ such that

$$\Pr\left[\|A - AM^\dagger M\|_F^2 \leq 16(k + 1)! \|A - A_k^\dagger\|_F^2 \right] \geq \frac{2}{3}.$$

The algorithm uses $\text{poly}(d, k, \log n)$ bits of space.

**Subspace approximation.** In the subspace approximation problem, the inputs are the matrix $A \in \mathbb{R}^{n \times d}$ and an integer $k > 0$.
and the goal is to output a \(k\)-dimensional linear subspace \(H\) that minimizes \(\sum_{i=1}^{n} d(A_i, H)^p\), where \(p \in \{1, 2\}\) and \(d(A_i, H) = \|A_i(I - HH^T)\|_2\) is the distance from \(A_i\) to the subspace \(H\). A number of algorithms for the subspace approximation were given for the centralized setting \([13, 16, 18, 33]\) and more recently, \([28]\) gave the first algorithm for subspace approximation on turnstile streams. The algorithm of \([28]\) is based on sketching techniques and although it offers a superior \((1 + \epsilon)\)-approximation, their subspace has a larger number of rows and the rows may not originate from \(A\), whereas we select \(k\) noisy rows of the matrix \(A\) to form the subspace.

\[\text{Theorem 1.4. Given } p \in \{1, 2\} \text{ and a matrix } A \in \mathbb{R}^{n \times d} \text{ that arises in a turnstile data stream, there exists a one-pass algorithm that outputs a set } Z \text{ of } k \text{ (noisy) rows of } A \text{ such that}\]

\[\Pr \left[ \left( \sum_{i=1}^{n} d(A_i, Z)^p \right)^{\frac{1}{p}} \leq 4(k+1) \left( \sum_{i=1}^{n} d(A_i, A_i^*)^p \right)^{\frac{1}{p}} \right] \geq \frac{2}{3},\]

where \(A_i^*\) is the best rank \(k\) solution to the subspace approximation problem. The algorithm uses \(\log(d, k, \log n)\) bits of space.

Our adaptive sampling procedure also gives a bicriteria algorithm for a better approximation but allows the dimension of the subspace to be larger.

\[\text{Theorem 1.5. Given } p \in \{1, 2\}, \epsilon > 0, \text{ and a matrix } A \in \mathbb{R}^{n \times d} \text{ that arrives in a turnstile data stream, there exists a one-pass algorithm that outputs a set } Z \text{ of poly } \left( k, \frac{1}{\epsilon}, \log \frac{1}{\epsilon} \right) \text{ (noisy) rows of } A \text{ such that}\]

\[\Pr \left[ \left( \sum_{i=1}^{n} d(A_i, Z)^p \right)^{\frac{1}{p}} \leq (1 + \epsilon) \left( \sum_{i=1}^{n} d(A_i, A_i^*)^p \right)^{\frac{1}{p}} \right] \geq \frac{2}{3},\]

where \(A_i^*\) is the best rank \(k\) solution to the subspace approximation problem. The algorithm uses \(\log(d, k, \frac{1}{\epsilon}, \log n)\) bits of space.

Projective clustering. Projective clustering is an important problem for bioinformatics, computer vision, data mining, and unsupervised learning \([31]\). The projective clustering problem takes as inputs a matrix \(A \in \mathbb{R}^{n \times d}\) and integers \(k > 0\) for the target dimension of each subspace and \(s > 0\) for the number of subspaces, and the goal is to output \(s\) \(k\)-dimensional linear subspaces \(H_1, \ldots, H_s\) that minimizes \(\sum_{i=1}^{n} d(A_i, H)^p\), where \(p \in \{1, 2\}\), \(H = H_1 \cup \ldots \cup H_s\), and \(d(A_i, H)\) is the distance from \(A_i\) to union \(H\) of \(s\) subspaces \(H_1, \ldots, H_s\). A number of streaming algorithms for projective clustering \([6, 11, 18, 20]\) are based on the notion of coresets, which are small numbers of weighted representative points. These results require a stream of (possibly high dimensional) points, which is equivalent to the row-arrival model and thus do not extend to turnstile streams. \([27]\) gives a turnstile algorithm based on random projections, but the algorithm requires space linear in the number of points. Thus our adaptive sampling procedure gives the first turnstile algorithm for projective clustering that uses space sublinear in the number of points.

\[\text{Theorem 1.6. Given } p \in \{1, 2\}, \epsilon > 0 \text{ and a matrix } A \in \mathbb{R}^{n \times d} \text{ that arrives in a turnstile data stream, there exists a one-pass algorithm that outputs a set } S \text{ of poly } \left( k, s, \frac{1}{\epsilon} \right) \text{ rows, which includes a union } T \text{ of } s\text{-dimensional subspaces such that}\]

\[\Pr \left[ \left( \sum_{i=1}^{n} d(A_i, T)^p \right)^{\frac{1}{p}} \leq (1 + \epsilon) \left( \sum_{i=1}^{n} d(A_i, H)^p \right)^{\frac{1}{p}} \right] \geq \frac{2}{3},\]

where \(H\) is the union of \(s\) \(k\)-dimensional subspaces that is the optimal solution to the projective clustering problem. The algorithm uses \(\log(d, k, \frac{1}{\epsilon}, \log n)\) bits of space.

Volume maximization. The volume maximization problem takes as inputs a matrix \(A \in \mathbb{R}^{n \times d}\) and a parameter \(k\) for the number of selected rows, and the goal is to output \(k\) rows \(r_1, \ldots, r_k\) of \(A\) that maximize the volume of the parallelepiped spanned by the rows. \([21, 22]\) give core-set constructions for volume maximization that approximate the optimal solution within a factor of \(O(k)^{k/2}\) and \(O(k)^k\) respectively, and can be implemented in the row-arrival model. Their algorithms are based on spectral spanners and local search based on directional heights and do not immediately extend to turnstile streams. Hence our adaptive sampling procedure gives the first turnstile algorithm for volume maximization that uses space sublinear in the input size.

\[\text{Theorem 1.7. Given a matrix } A \in \mathbb{R}^{n \times d} \text{ that arrives in a turnstile data stream and an approximation factor } \alpha > 1, \text{ there exists a one-pass algorithm that outputs a set } S \text{ of } k \text{ noisy rows of } A \text{ such that}\]

\[\Pr \left[ |A^k(k!)\text{Vol}(S)| \geq |\text{Vol}(M)| \right] \geq \frac{2}{3},\]

where \(|\text{Vol}(S)|\) is the volume of the parallelepiped spanned by \(S\) and \(M\) is a set of \(k\) rows that maximizes the volume. The algorithm uses \(\tilde{O}\left(\frac{ndk^2}{\alpha^2}\right)\) bits of space.

We give a lower bound for the volume maximization problem on turnstile streams that is tight up to lower order terms. Additionally, we give a lower bound for volume maximization in the random order row-arrival model, which we will also show is tight up to lower order terms. Our lower bounds complement the thorough lower bounds for extent problems given by \([2]\).

\[\text{Theorem 1.8. There exists a constant } C > 1 \text{ so that any one-pass streaming algorithm that outputs a } C\text{-approximation to the volume maximization problem with probability at least } \frac{1}{2} \text{ in the random order row-arrival model requires } \tilde{O}(n) \text{ bits of space. Moreover for any integer } p > 0, \text{ any } p\text{-pass turnstile streaming algorithm that gives an } \alpha^p \text{-approximation to the volume maximization problem requires } \Omega\left(\frac{n}{k^{\alpha^p}}\right) \text{ bits of space.}\]

Finally, we give a corresponding upper bound for volume maximization in the row-arrival model competitive with our lower bound.

\[\text{Theorem 1.9. Let } 1 < C < \frac{(\log n)}{k}. \text{ There exists a one-pass streaming algorithm in the row-arrival model that computes a subset } S \text{ of size } k \text{ of points in } \mathbb{R}^d \text{ such that}\]

\[\Pr \left[ |O(Ck)^{k/2}\text{Vol}(S)| \geq |\text{Vol}(M)| \right] \geq \frac{2}{3},\]

where \(|\text{Vol}(S)|\) is the volume of the parallelepiped spanned by \(S\) and \(M\) is a set of \(k\) rows that maximizes the volume. The algorithm uses \(O\left(\frac{n^{O(1/C)}}{d}\right)\) bits of space.
These prior versa. Hence our algorithm must first run a statistical test to determine whether the error in the CountSketch data structure caused the rows of $B$ our task would reduce to identifying temporarily suppose that only the row $z$ of $A$, parameter that we choose. Thus if $B$ is the identity matrix, so that we output an index $i$ chosen uniformly at random, adaptive sampling then imbues some randomness into the sampling procedure. Adaptive sampling, we observe that if $p$ is a single instance of the sampler outputs an index from roughly the desired distribution with probability $\Omega\left(\frac{1}{K}\right)$ and with probability $1 - \Omega\left(\frac{1}{K}\right)$, it aborts and outputs nothing. Hence for $p \in \{1,2\}$, we obtain a constant probability of success using $\text{poly}\left(\frac{1}{\Delta}, \log n\right)$ space by setting $K = \text{poly}\left(\frac{1}{\Delta}, \log n\right)$, repeating with $O(K)$ instances, and taking the output of the first successful instance.

It remains to argue that CountSketch and norm estimation generalize to $L_{p,2}$ error for matrices, which we do through standard arguments in Section 2. In fact, the data structures maintained by the generalized matrix CountSketch and $L_{p,2}$ norm estimation procedures are linear combinations of the rows of $A$, so we can right multiply the rows that are stored in the $L_{p,2}$ sampler by $P$ to simulate sampling rows of $AP$. In other words, if we had a stream of updates to the matrix $AP$, the resulting data structure on the stream would be equivalent to maintaining the data structure on a stream of updates to the matrix $A$, and then multiplying each row of the data structure by $P$ in post-processing. Hence we can also sample rows of $AP$ with probabilities proportional to the residual $\|A_iP\|_2^p$, which will be crucial for our adaptive sampler.

### Adaptive Sampler

Recall that adaptive sampling iteratively samples rows of $A \in \mathbb{R}^{n \times d}$ with probability proportional to the $p$th power of their distances from the subspace spanned by the rows that have already been sampled in previous rounds, for $k$ rounds. Thus if $H_j$ is the matrix formed by the rows sampled by step $j$, then we would like to sample $i \in [n]$ with probability $\|A_i(\mathbb{I} - H_j^\dagger H_j)\|_2^p$ with the largest squared norm. Given our $L_{p,2}$ sampler, a natural approach is to run $k$ instances of the sampler throughout the stream. Once the stream completes, we use the first instance to sample a row of $A$, which forms $H_1$ (recall that $H_1 = \emptyset$ is "used" to sample in the first iteration). Since our $L_{p,2}$ sampler supports post-processing multiplication by a matrix $P$, we subsequently use the $j$th instance to $L_{p,2}$ sample a row of $A(\mathbb{I} - H_j^\dagger H_j)$, which we then append to $H_j$ to form $H_{j+1}$. Repeating this $k$ times, we would like to argue this simulates $k$ steps of adaptive sampling.

The first issue with this approach is that our $L_{p,2}$ cannot return the original rows of $A$, but only some noisy perturbation of the sampled row. It is easy to see that returning the noisy rows of $A$ is unavoidable for sublinear space by considering a stream whose final update to some random coordinate is arbitrarily large, while the previous updates were small. Then the row containing the coordinate of the final update should be sampled with large probability, but that row can only be completely recovered if all entries of the matrix are stored. Fortunately we show that if we sample the index $x$, then we output a row $r = A_x + v$, where the noisy component $v$ CountSketch error is determined to be too large by the statistical test, the algorithm aborts; otherwise the algorithm outputs the row with the largest norm if it exceeds $T$.

Now there can still be some error if multiple rows have norms close to or exceeding $T$, but it turns out that by choosing the appropriate parameters, the probability that there exists a row whose norm exceeds $T$ is $\Omega\left(\frac{1}{K}\right)$ and the probability that the statistical test fails or that multiple rows have norms close to or exceeding $T$ is $O\left(\frac{1}{\Delta}\right)$, which incurs a relative $(1 \pm \epsilon)$ perturbation of the sampling probabilities. Thus a single instance of the sampler outputs an index from roughly the desired distribution with probability $\Omega\left(\frac{1}{K}\right)$ and with probability $1 - \Omega\left(\frac{1}{K}\right)$, it aborts and outputs nothing. Hence for $p \in \{1,2\}$, we obtain a constant probability of success using $\text{poly}\left(\frac{1}{\Delta}, \log n\right)$ space by setting $K = \text{poly}\left(\frac{1}{\Delta}, \log n\right)$, repeating with $O(K)$ instances, and taking the output of the first successful instance.
is a linear combination of rows of $A$ that satisfies $\|v\|_2 \leq \epsilon \|Ax\|_2$. Thus, the norms of the sampled rows are somewhat preserved.

On the other hand, sampling noisy rows of $A$ rather than the original rows of $A$ can drastically alter the subspace spanned by the matrix formed by the rows. This in turn can significantly alter the sampling probabilities in future rounds. Consider the following example. Let $A$ be a matrix that has $[0 \ 1]$ for half of its rows and $[M \ 0]$ for some large $M > 0$ for the other half of its rows. Then, with large probability we should sample some row $u = [M \ 0]$ in the first step. However, due to noise in the sampler, we will actually obtain some noisy row $v = [M' \ m]$, where $M' \approx M$ and $m \neq 0$.

Now in the second round, if we sampled $u$ in the first round, then the only possible output of the adaptive sampler is a row $[0 \ 1]$, since all the rows $[M \ 0]$ are contained in the subspace spanned by $u$. However, since we actually sampled $v$ in the first round, then the distance from $u$ to the subspace spanned by $v$ is nonzero. Furthermore, since $M$ is large, then it actually seems likely that we might sample a row $[M \ 0]$ rather than $[0 \ 1]$. Thus we might sample some row that we should have not samples or worse, we might repeatedly sample the same row!

Similarly, the noisy perturbations may cause us to completely avoid rows that we should have sampled with nonzero probability if we had access to the original rows. In fact, this example shows that we cannot guarantee that our adaptive sampler gives a multiplicative $(1 + \epsilon)$-approximation to the true sampling probabilities of each row in any round.

Our key observation is that the noisy row $r$ output by our $L_{p,2}$ sampler not only has a noisy component $v$ small in norm compared to $Ax$, but also its component in the space orthogonal to $Ax$ must be small. That is, $r$ can also be written as $r = Ax + w$, where $\|wQ\|_2 \leq \epsilon \|Ax\|_2$ for any projection matrix $Q$. This tighter bound in any orthogonal direction allows us to bound in subsequent rounds the additive error of sampling probabilities, which are based on the vector lengths in orthogonal directions.

Namely, if we write an orthonormal basis $U$ for the actual rows of $A$ and an orthonormal basis $W$ for the noisy rows that we sample, we can show that the norm of a row projected onto $W$ has a small additive perturbation from when it is projected onto $U$. Thus we require the construction of orthonormal bases $U$ and $W$ from which we can easily extract the sampling probabilities of rows both with respect to the original rows and to the noisy rows. We achieve this by designing $U$ so that the first basis vectors of $U$ are precisely the true sampled rows of $A$, followed by the noisy perturbations for each sample. We then argue that if we design $W$ so that the first basis vectors of $W$ are precisely the noisy rows that we sample, then the coefficients of each row represented in terms of $U$ and $W$ have only a small additive difference. By summing across all rows, then we can bound the total variation distance between sampling with the noisy rows of $A$ and sampling with the actual rows of $A$.

Applications. For many applications on turnstile streams, we show it suffices in each step to obtain a noisy row that is orthogonal to the previously selected rows, sampled with probability proportional to the $p^n$ power of the distance to the subspace spanned by those rows. Thus for $p = 1$, our adaptive sampler allows us to perform residual based sampling in place of subspace embedding techniques used by previous work in various applications [27, 28, 35]. Additionally for $p = 2$, our adaptive sampler allows us to simulate volume sampling, which has a wide range of applications [12, 15–17].

For volume maximization on turnstile streams, we use a combination of our $L_{2,2}$ sampler and our generalized CountSketch data structure to simulate an approximation to the greedy algorithm of choosing the row with the largest residual at each step. If the largest residual found by CountSketch exceeds a certain threshold, we use that row; otherwise any row output by our adaptive sampler will be a good approximation to the row with the largest residual. Thus the volume of the parallelepiped spanned by these rows is a good approximation to the optimal solution.

Volume maximization. We provide lower bounds for volume maximization in turnstile streams and the row-arrival model through reductions from the Gap $\rho_\infty$ problem and the distributional set-disjointness problem, respectively. For both cases we show that embedding the same instance across multiple columns gives hardness of approximating within a factor with exponential dependency on $k$. For our algorithmic results in the row-arrival model, we first note that the composable core-set techniques of [22] automatically gives a streaming algorithm for volume maximization. In fact, [22] shows the stronger guarantee that any composable core-set for the directional height of a point set suffices to give a good approximation for volume maximization. Using this idea, we give a dimensionality reduction algorithm for volume maximization in the row-arrival model competitive with our lower bounds by embedding the input into a lower dimensional space.

Recall that from Johnson-Lindenstrauss, right multiplication by a random matrix whose entries are drawn i.i.d from a Gaussian distribution suffices to preserve the directional heights of the points in an optimal set by a constant factor, say 2, and thus the volume of the largest set of $k$ points is only distorted by a factor of $2^k$. We then prove that for every other subset of $k$ points, their volume does not increase by too much by showing that the eigenvalues of the matrix representation of the points are preserved by some factor $C$ with very high probability. Thus taking a union bound over all subsets of $k$ points, all volumes are preserved by a factor $C^k$ and we obtain dimensionality reduction of the problem by applying right multiplication of the random matrix to each of the input rows.

1.3 Preliminaries

For any positive integer $n$, we use the notation $[n]$ to represent the set $\{1, \ldots, n\}$. We use the notation $x = (1 \pm \epsilon)y$ to denote the containment $(1 - \epsilon)y \leq x \leq (1 + \epsilon)y$. We write $\text{poly}(n)$ to denote some fixed constant degree polynomial in $n$ but we write $\frac{1}{\text{polylog}(n)}$ to denote some arbitrary degree polynomial in $n$. When an event has probability $1 - \frac{\epsilon}{\text{polylog}(n)}$ of occurring, we say the event occurs with high probability. We use $\tilde{O}(\cdot)$ to omit lower order terms and similarly $\text{polylog}(n)$ to omit terms that are polynomial in $\log n$.

We require the following definition of total variation distance to bound the difference between two probability distributions, such as the “ideal” sampling distributions compared to the distributions provided by our algorithms.

Definition 1.10 (Total variation distance). Let $\mu, \nu$ be two probability distributions on a finite domain $\Omega$. Then the total variation
distance between $\mu$ and $v$ is defined as $d_1(\mu,v) = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - v(x)|$.

For our purposes, a turnstile stream will implicitly define a matrix $A \in \mathbb{R}^{n \times d}$ through a sequence of $m$ updates. We use $A_i$ to denote the $i^{th}$ row of $A$ and $A_{i,j}$ to denote the $j^{th}$ entry of $A_i$. The matrix $A$ initially starts as the all zeros matrix. Each update in the stream has the form $(i_t, j_t, A_t)$, where $t \in [m]$, $i_t \in [n]$, $j_t \in [d]$, and $A_t \in \{-M, -M + 1, \ldots, M - 1, M\}$ for some large positive integer $M$. The update then induces the change $A_{i_t,j_t} \leftarrow A_{i_t,j_t} + A_t$ in $A$. We assume throughout that $m, M = \text{poly}(n)$ and $n \gg d$. We will typically only permit one pass over the stream, but for multiple passes the order of the updates remains the same in each pass.

In the row-arrival model, the stream has length $n$ and the $i^{th}$ update in the stream is precisely row $A_i$. Again we restrict each entry $A_{i,j}$ of $A$ to be in the range $\{-M, -M + 1, \ldots, M - 1, M\}$ for some large positive integer $M = \text{poly}(n)$. We assume that $A$ can be adversarially chosen in the row-arrival model, but for the random order row-arrival model, once the entries of $A$ are chosen, an arbitrary permutation of the rows of $A$ is chosen uniformly at random, and the rows of that permutation constitute the stream. For the problems that we consider, the optimal solution is invariant to permutation of the rows of $A$, so the random order does not impact the desired solution. Observe that algorithms for turnstile streams can be used in the row-arrival model, but not necessarily vice versa.

We use $I_k$ to denote the $k \times k$ identity matrix and we drop the subscript when the dimensions are clear. We use the notation $A = A_1 \circ A_2 \circ \ldots \circ A_n$ to denote that the matrix $A$ is formed by the rows $A_1, \ldots, A_n$ and the notation $A^\top$ to denote the transpose of $A$. For a matrix $M \in \mathbb{R}^{k \times d}$ with linearly independent rows, we use $M^\top \in \mathbb{R}^{d \times k}$ to denote the Moore-Penrose pseudoinverse of $M$, so that $M^\top = (MM^\top)^{-1}$ and $M^\top M = I_k$.

**Definition 1.11 (Vector/matrix norms).** For a vector $v \in \mathbb{R}^n$, we have the Euclidean norm $|v|_2 = \sqrt{\sum_{i=1}^n v_i^2}$ and more generally, $|v|_p = \left(\sum_{i=1}^n |v_i|^p\right)^{\frac{1}{p}}$. For a matrix $A \in \mathbb{R}^{n \times d}$, we denote the Frobenius norm of $A$ by $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^d A_{i,j}^2}$. More generally, we write the $L_{p,q}$ norm of $A$ by $\|A\|_{p,q} = \left(\sum_{i=1}^n \left(\sum_{j=1}^d |A_{i,j}|^q\right)^{\frac{p}{q}}\right)^{\frac{1}{p}}$, so that $\|A\|_F = \|A\|_{2,2}$.

For $A \in \mathbb{R}^{n \times d}$, we use $A_{\text{tail}}(b)$ to denote $A$ with the $b$ rows of $A$ with the largest Euclidean norm set to zeros.

**Definition 1.12 ($L_{p,q}$ sampling).** Let $A \in \mathbb{R}^{n \times d}$, $0 \leq \epsilon < 1$, and $p,q > 0$. An $L_{p,q}$ sampler with $\epsilon$-relative error is an algorithm that outputs an index $i \in [n]$ such that for each $j \in [n]$, $$\Pr[i = j] = \frac{|A_j|^p}{\|A\|^p_{p,q}} (1 + \epsilon) + O(n^{-\epsilon}),$$ for some arbitrarily large constant $c \geq 1$. In each case, the sampler is allowed to output fail with some probability $\delta$, in which case it must output $\bot$. When the underlying matrix is just a vector, i.e., $d = 1$, we drop the $q$ term and call such an algorithm an $L_p$ sampler.

**Definition 1.13 (Adaptive sampling).** Let $A \in \mathbb{R}^{n \times d}$ be a matrix from which we wish to sample and $M \in \mathbb{R}^{m \times d}$ be a matrix corresponding to a specific subspace. For $p \in \{1, 2\}$, an adaptive sampler is an algorithm that outputs an index $i \in [n]$ and the corresponding row $A_i$ such that for each $j \in [n]$, $$\Pr[i = j] = \frac{|A_j|^p}{\|A\|^p_{p,2}}.$$ where $\mathbf{P} = \mathbb{I} - M^\top M$.

In typical applications, we will wish to perform $k$ rounds of adaptive sampling with subspaces $M_1, M_2, \ldots, M_k$, where $M_1$ is the all zeros matrix, and each $M_i$ will consist of the rows sampled from rounds $1$ to $i - 1$. We will use the term adaptive sampling to refer to both a single round of sampling and multiple rounds of sampling interchangeably when the context is clear.

Note that the adaptive sampling for input matrices $A \in \mathbb{R}^{n \times d}$ and $M \in \mathbb{R}^{m \times d}$ can be seen as $L_{p,2}$ sampling on an input matrix $AP$ with $\epsilon = 0$ and $\mathbf{P} = \mathbb{I} - M^\top M$, but returning the row $A_i$ instead of $A_iP$.

### 2 L_{2,2} sampler

In this section, we first describe a turnstile streaming algorithm that takes a matrix $A \in \mathbb{R}^{n \times d}$ that arrives as a data stream and post-processing query access to a matrix $P \in \mathbb{R}^{d \times d}$, and outputs an index $i \in [n]$ of a row of AP sampled with probability roughly $\frac{|A_i|^2}{\|A\|^2_2}$.

**High level idea.** First suppose we only wanted to sample a row $i$ of $A \in \mathbb{R}^{n \times d}$ with probability roughly $\frac{|A_i|^2}{\|A\|^2_2}$. By multiplying each row $A_i$ with a random scaling factor $\frac{1}{\sqrt{t_i}}$, where $t_i \in [0,1]$ is chosen independently and uniformly at random, the probability that $\frac{1}{t_i} \frac{|A_i|^2}{\|A\|^2_2} \geq \frac{|A_i|^2}{\|A\|^2_2}$ is precisely the probability that $t_i \leq \frac{|A_i|^2}{\|A\|^2_2}$, which is the desired probability of sampling row $i$.

Now suppose only one row $A_i$ satisfies $\frac{1}{t_i} \frac{|A_i|^2}{\|A\|^2_2} \geq \frac{|A_i|^2}{\|A\|^2_2}$, so that we would like to output $A_i$. If we stored all rows of $A$ as well as all scaling factors $t_i$, then we could identify and output this row, but the required space would be linear in the input size. Instead, we hash all scaled rows $\frac{1}{t_i} \frac{|A_i|^2}{\|A\|^2_2}$ to a number of buckets in a CountSketch data structure. Observe that if $\frac{1}{t_i} \frac{|A_i|^2}{\|A\|^2_2} \geq \frac{|A_i|^2}{\|A\|^2_2}$ for only one index $i$, then $\frac{1}{\sqrt{t_i}} A_i$ must also be the scaled row with the largest norm.

Moreover, it turns out that the mass of $\sum_{j=1}^n \frac{1}{t_j} \frac{|A_j|^2}{\|A\|^2_2}$ is dominated by a small number of rows. Hence with a sufficiently large number of buckets, the scaled row $i$ is the heavy hitter with the largest norm among all the heavy hitters of the scaled rows and so CountSketch will ideally identify the row $i$.

This approach can fail due to two reasons. The first potential issue is if the accuracy of CountSketch does not suffice to identify the row $A_i$ due to the noise from the tail of the mass of $\sum_{j=1}^n \frac{1}{t_j} \frac{|A_j|^2}{\|A\|^2_2}$. That is, if the noise of the tail due to the selection of the scaling factors $t_j$ prevents CountSketch from successfully identifying the heavy hitters, then this approach will fail. We can run a statistical test to identify when the noise is too large and preemptively abort...
We require generalizations of the celebrated AMS \([\Omega]\) with probability \(1 - \epsilon\), so we can run \(O\left(\frac{1}{\epsilon}\right)\) instances of the algorithm in parallel and take the first instance that does not abort.

A separate issue is resolving the assumption that only one row \(A_i\) satisfies \(\frac{1}{\epsilon} \|A_i\|_2^2 \geq \|A\|_2^2\). As it turns out, many rows can exceed this threshold, but if we instead require \(\frac{\epsilon}{\pi} \|A_i\|_2^2 \geq \frac{1}{\epsilon} \|A\|_2^2\), then the probability that some row exceeds this threshold is \(\Theta(\epsilon)\). The probability that multiple rows exceed this higher threshold is now \(O\left(\epsilon^2\right)\). Our algorithm outputs the row with the largest norm when some row exceeds the threshold, so in the case where multiple rows exceed the higher threshold we attribute the output to possible sampling probability perturbation. Hence the probability that multiple rows exceed the higher threshold only slightly perturbs the sampling probability of each row by a \((1 \pm \epsilon)\) factor.

Thus we can again repeat \(O\left(\frac{1}{\epsilon}\right)\) times until some row \(A_i\) satisfies \(\frac{1}{\epsilon} \|A_i\|_2^2 \geq \|A\|_2^2\). Similarly, if the error from CountSketch causes an inaccurate estimation of the row with the largest norm, then we might think the heaviest row does not exceed the threshold when it does in reality or vice versa. Fortunately, this only occurs when the row with the largest norm is very close to the threshold, which we again show only causes the sampling probability of each row to perturb by a \((1 \pm \epsilon)\) factor.

For technical reasons, we further increase the threshold and thus run a larger number of instances in parallel to avoid failure. We note that although CountSketch successfully identifies the row \(i\), it can only output a noisy perturbation of \(A_i\). That is, it can only output some row \(r = A_i + v\), where the noisy component \(v\) satisfies \(\|v\|_2 \leq \epsilon \|A_i\|_2\).

Finally, we note that these procedures are all performed through linear sketches and that each bucket stores aggregate rows of the matrix \(A\). Thus if we had a stream of updates to the matrix \(AP\), the resulting data structure would be equivalent to maintaining the data structure on a stream of updates to the matrix \(A\), and then multiplying each row of the data structure by \(P\) post-processing. Hence we can also sample rows of \(AP\) with probabilities proportional to \(\|A_iP\|_2^2\).

### 2.1 Streaming Algorithms with Post-Processing

We require generalizations of the celebrated AMS \([3]\) and CountSketch \([10]\) algorithms to handle Frobenius norm estimation of \(AP\) and to output the rows of \(AP\) whose norm exceed a certain fraction of the total Frobenius norm, respectively. These generalizations are streaming algorithms that perform their desired function in low space even though query access to \(P\) is only provided after the stream ends.

We give in Algorithm 1 the generalization of the AMS \([3]\) algorithm that estimates \(\|AP\|_F^2\), where \(A\) arrives in a stream and post-processing query access to \(P\) is given after the stream ends. Moreover, Algorithm 1 is a linear sketch, so it can also be used to estimate \(\|AP - M\|_F^2\) for a second arbitrary post-processing matrix \(M \in \mathbb{R}^{n \times d}\).

**Algorithm 1** Basic algorithm that estimates \(\|AP\|_F\), where \(P\) is a post-processing matrix

**Input:** Matrix \(A \in \mathbb{R}^{n \times d}\), query access to matrix \(P \in \mathbb{R}^{d \times d}\) after the stream ends, constant parameter \(\epsilon > 0\).

**Output:** \((1 + \epsilon)\)-approximation of \(\|AP\|_F\).
1. Let \(h_i \in \{-1, +1\}\) be 4-wise independent for \(i \in [n]\).
2. Let \(v \in \mathbb{R}^{1 \times d}\) be a vector of zeros.
3. **Streaming Stage:**
   4. for each update \(A_t\) to entry \(A_{i,j}\) do
      5. \(\Delta_t \cdot h_i \rightarrow v_{j}\).
6. **Processing P Stage:**
7. Output \(\|vP\|_2\).

We give in Algorithm 2 the generalization of the CountSketch \([10]\) algorithm that outputs all rows \(i\) of \(AP\) such that \(\|A_iP\|_2 \geq \epsilon \|AP\|_F\), where \(A\) arrives in a stream and post-processing query access to \(P\) is given after the stream ends. We call a row \(i\) a heavy row if \(\|A_iP\|_2 \geq \epsilon \|AP\|_F\).

**Algorithm 2** Basic algorithm that outputs heavy rows of \(\|AP\|_F\), where \(P\) is a post-processing matrix

**Input:** Matrix \(A \in \mathbb{R}^{n \times d}\), query access to matrix \(P \in \mathbb{R}^{d \times d}\) after the stream ends, constant parameter \(\epsilon > 0\).

**Output:** Small perturbations of the rows \(A_iP\) with \(\|A_iP\|_2 \geq \epsilon \|AP\|_F\).
1. \(r \leftarrow \Theta(\log n)\) with a sufficiently large constant.
2. \(b \leftarrow \Omega\left(\frac{1}{\epsilon}\right)\) with a sufficiently large constant.
3. Let \(T\) be an \(r \times b\) table of buckets, where each bucket stores an \(\mathbb{R}^{1 \times d}\) row, initialized to zeros.
4. Let \(s_{i,j} \in \{-1, +1\}\) be 4-wise independent for \(i \in [n], j \in [r]\).
5. Let \(h_i : [n] \rightarrow [b]\) be 4-wise independent for \(i \in [r]\).
6. **Streaming Stage:**
   7. for each update \(A_t\) to entry \(A_{i,j}\) do
      8. \(s_i \cdot h_i \rightarrow v_{j}\).
5. **Processing P Stage:**
   12. for \(k \in [r], \ell \in [b]\) do
      13. \(v_{k,\ell} \leftarrow v_{k,\ell}P\).
6. On query \(i \in [n]\), report median for \(k \in [r]\) \(\|v_{k, h_k(i)}\|_2\).

We first show that if \(X = AP\) and the stream updates the entries of \(X\) rather than the entries of \(A\), then we can obtain a good approximation to the heavy rows. Equivalently, the statement reads that if \(P = I\) is the identity matrix, then Algorithm 2 finds the heavy rows of \(AP\). We will ultimately show Algorithm 2 finds the heavy rows...
of AP for general P by using the same linear sketching argument as in the proof of Lemma 2.1.

For a matrix $X \in \mathbb{R}^{n \times d}$, recall that $X_{\text{tail}}(b)$ denotes $X$ with the $b$ rows of $X$ with the largest norm set to zeros. The following lemma shows that the $\frac{\epsilon}{\sqrt{d}}$ rows with the largest norm output by Algorithm 2 forms a good estimate of $X$, even with respect to the stronger Frobenius tail error.

**Lemma 2.2.** For any matrix $X \in \mathbb{R}^{n \times d}$, Algorithm 2 outputs an estimate $\tilde{X}_i$ for each row $X_i$, which together form an estimate matrix $\tilde{X}$. Then with high probability, for all $i \in [n]$, there exists a vector $v_i$ such that $\|X_i\|_2 - \|\tilde{X}_i\|_2 \leq \epsilon \left\|X_{\text{tail}}(\frac{\epsilon}{\sqrt{d}})\right\|_F$. Consequently, $\left\|X_{\text{tail}}(\frac{\epsilon}{\sqrt{d}})\right\|_F \leq \|X - \tilde{X}\|_F \leq 2 \left\|X_{\text{tail}}(\frac{\epsilon}{\sqrt{d}})\right\|_F$ with high probability, where $\tilde{X} = X - \tilde{X}_{\text{tail}}(\frac{\epsilon}{\sqrt{d}})$ denotes the top $\frac{\epsilon}{\sqrt{d}}$ rows of $X$ by norm.

Taking $b = \Theta \left(\frac{1}{\epsilon^2}\right)$ in Lemma 2.2, we have the following guarantees of CountSketch-M.

**Lemma 2.3.** Given a constant $b > 0$, there exists a one-pass streaming algorithm CountSketch-M that takes updates to entries of a matrix $A \in \mathbb{R}^{n \times d}$ as well as query access to a post-processing matrix $P \in \mathbb{R}^{d \times d}$ that arrives after the stream, and outputs all indices $i$ such that $\|A_iP\|_2 \geq \frac{1}{\sqrt{b}} \|AP\|_F$. For each index $i$, CountSketch-M also outputs a vector $r_i$ such that $r_i = A_iP + v_i$ and $\|v_i\|_2 \leq \frac{1}{\sqrt{b}} \left\|(AP)_{\text{tail}}(b)\right\|_F$.

The algorithm uses $O \left(\frac{dk \log^2 n}{\epsilon^2}\right)$ bits of space and succeeds with high probability.

### 2.2 $L_{2,2}$ Sampling Algorithm

In this section, we give an algorithm for $L_{2,2}$ sampling that will ultimately be used to simulate adaptive sampling on turnstile streams.

#### 2.2.1 Algorithm Description

Given subroutines that estimate $\|AP\|_F$ and the heavy rows of $AP$, we implement our $L_{2,2}$ sampler in Algorithm 3. Our algorithm first takes each row $A_i$ of matrix $A$ and forms a row $B_i = \frac{A_i}{\sqrt{t_i}}$, where $t_i$ is a scaling factor drawn uniformly at random from $[0,1]$. Note that we have the following observation:

**Observation 2.4.** For $\gamma > 0$, $\Pr \left[\|B_iP\|_2 \geq \gamma \|AP\|_F\right] = \frac{\|AP\|_F^2}{\gamma^2 \|AP\|_F^2}$.

Intuitively, Observation 2.4 claims that by setting $T \simeq \|AP\|_F$, we can identify a row of $BP$ whose norm exceeds $T$ to effectively $L_{2,2}$ sample a row of $AP$. For technical reasons, we set $T = \sqrt{\frac{C \log n}{\epsilon^2}} \|AP\|_F$. Our algorithm then uses CountSketch-M to find heavy rows of $BP$ and AMS-M to give an estimate $\hat{F}$ of $\|AP\|_F$ to determine whether there exists a row of $BP$ whose norm exceeds $T$.

Our algorithm also uses CountSketch-M and a separate instance of AMS-M to compute $\tilde{S}$, which estimates the error in the tail of $BP$ and also indicates how accurate CountSketch-M is. If $\tilde{S}$ is large, then our estimations for each row of $BP$ from CountSketch-M may be inaccurate, so our algorithm must abort. Otherwise, if $\tilde{S}$ is sufficiently small, then our estimations for each row of $BP$ is somewhat accurate. Thus if the row of $BP$ with the largest norm exceeds $\sqrt{\frac{C \log n}{\epsilon^2}} \hat{F}$, which is our estimation for $T$, then we output that particular row rescaled by $\sqrt{T_i}$ to recover the (noisy) original row of AP.

**Algorithm 3** Single $L_{2,2}$ Sampler

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$ that arrives as a stream, matrix $P \in \mathbb{R}^{d \times d}$ that arrives after the stream, approximation parameter $\epsilon > 0$.

**Output:** Noisy row $r$ of AP sampled roughly proportional to the squared row norms of AP.

1. **Pre-processing Stage:**
   - $b \leftarrow \Omega \left(\frac{1}{\epsilon^2}\right)$, $r \leftarrow \Theta(\log n)$ with sufficiently large constants.
   - For $i \in [n]$, generate independent scaling factors $t_i \in [0,1]$ uniformly at random.
   - Let $B$ be the matrix consisting of rows $B_i = \frac{A_i}{\sqrt{t_i}}$.
   - Let AMS-M$_1$ and AMS-M$_2$ track the Frobenius norms of $AP$ and $BP$, respectively.
   - Let CountSketch-M be an $r \times b$ table, where each entry is a vector in $\mathbb{R}^d$, to find the heavy hitters of $B$.

2. **Streaming Stage:**
   - for each row $A_i$ do  
     - Presented in row-arrival model but also works for turnstile streams.
     - Update CountSketch-M with $B_i = \frac{A_i}{\sqrt{t_i}}$.
     - Update linear sketch AMS-M$_1$ with $A_i$.
     - Update linear sketch AMS-M$_2$ with $B_i = \frac{A_i}{\sqrt{t_i}}$.

3. **Processing P Stage:**
   - After the stream, obtain matrix $P$.
   - Multiply each vector $v$ in each entry of the CountSketch-M table by $P$: $v \leftarrow vP$.
   - Multiply each vector $v$ in AMS-M$_1$ by $P$: $v \leftarrow vP$.
   - Multiply each vector $v$ in AMS-M$_2$ by $P$: $v \leftarrow vP$.

4. **Extraction Stage:**
   - Use AMS-M$_1$ to compute $\hat{F}$ with $\|AP\|_F \leq \hat{F} \leq 2 \|AP\|_F$.
   - Extract the $\frac{\epsilon}{\sqrt{d}}$ (noisy) rows of $BP$ that are estimated by CountSketch-M to have the largest norms.
   - Let $M \in \mathbb{R}^{d \times d}$ be the matrix with $\frac{1}{\sqrt{d}}$-nonzero rows consisting of these top (noisy) rows.
   - Use AMS-M$_2$ to compute $\tilde{S}$ with $\|BP - M\|_F \leq \tilde{S} \leq 2 \|BP - M\|_F$.
   - Let $r_i$ be the (noisy) row of $AP$ in CountSketch-M with the largest norm.
   - Let $C > 0$ be some large constant so that the probability of failure is $O \left(\frac{1}{n^C}\right)$.

5. if $\tilde{S} > \sqrt{\frac{C \log n}{\epsilon^2}} \hat{F}$ or $\|r_i\|_2 < \sqrt{\frac{C \log n}{\epsilon^2}} \hat{F}$ then
   - return $\text{FAIL}$.
   - else
   - return $r = \sqrt{T_i} r_i$.

#### 2.2.2 Analysis

Conditioning on only a single row $B_iP$ satisfying $\|B_iP\|_2 \geq T = \sqrt{\frac{C \log n}{\epsilon^2}} \|AP\|_F$, we could immediately identify this
row if we had access to all rows of BP, as well as \( \|AP\|_F \), but this requires too much space. Instead, we use COUNTSKETCH-M to find the heavy rows of BP and compare their norms to an estimate of \( \epsilon \). However, if the error caused by COUNTSKETCH-M is high due to the randomness of the data structure, then the estimations of the row norms may be inaccurate and so our algorithm should abort. Our algorithm uses an estimator \( \hat{\epsilon} \) to compute the tail of BP, which bounds the error caused by COUNTSKETCH-M. We first show that the event of our algorithm aborts because the tail estimator \( \hat{\epsilon} \) is too large has small probability and is independent of the index \( i \) and the value of \( t_i \).

**Lemma 2.5.** For each \( j \in [n] \) and value of \( t_j \),

\[
\Pr \left[ \hat{\epsilon} > \sqrt{\frac{C \log n}{\epsilon}} \right] \leq \frac{1}{\poly(n)}.
\]

The probability of sampling each row \( i \in [n] \) will still be slightly distorted due to the noise from COUNTSKETCH-M, since we do not have exact values for the norm of each row. Similarly, if multiple rows exceed the threshold, we will output the row with the largest norm, which also alters the sampling probability of each row. We now show that these events only slightly perturb the probability of sampling each index \( i \) and moreover, the output row is a small noisy perturbation of the original row.

**Lemma 2.6.** Conditioned on a fixed value of \( \hat{\epsilon} \), the probability that Algorithm 3 outputs (noisy) row \( i \) is \( (1 \pm O(\epsilon)) \frac{\|AP\|_F^2}{\|P\|_F^2} + \frac{1}{\poly(n)} \).

Since each row is sampled with roughly the desired probability, we now analyze the space complexity of the resulting \( L_{2,2} \) sampler.

**Theorem 2.7.** Given \( \epsilon > 0 \), there exists a one-pass streaming algorithm that takes rows of a matrix \( A \in \mathbb{R}^{n \times d} \) as a turnstile stream and a matrix \( P \in \mathbb{R}^{d \times d} \) after the stream, and outputs noisy row \( r \) of \( AP \) with probability \( (1 \pm O(\epsilon)) \frac{\|AP\|_F^2}{\|P\|_F^2} + \frac{1}{\poly(n)} \). The algorithm uses \( O \left( \frac{n}{\epsilon^2} \log^3 n \log \frac{1}{\delta} \right) \) space to succeed with probability \( 1 - \delta \).

### 3 Noisy Adaptive Squared Distance Sampling

Given a matrix \( A \in \mathbb{R}^{n \times d} \) that arrives in a data stream, either turnstile or row-arrival, we want to simulate \( k \) rounds of adaptive sampling. That is, in the first round we want to sample some row \( r_1 \) of \( A \), such that each row \( A_j \) is selected with probability proportional to its squared row norm \( \|A_j\|_2^2 \). Once rows \( r_1, \ldots, r_{j-1} \) are selected, then the \( j \)th round of adaptive sampling samples each row \( A_j \) with probability proportional to the squared row norm of the orthogonal component to \( R_{j-1} \), \( \|A_j (I - R_{j-1}^T R_{j-1})\|_2^2 \), where for each \( j \leq k \), \( R_{j} = r_1 \circ \cdots \circ r_j \).

Observe that if only a single round of adaptive sampling were required, the problem would reduce to \( L_{2,2} \) sampling, which we can perform in a stream through Algorithm 3. In fact, the post-processing stage of Algorithm 3 would not be necessary since the post-processing matrix would be \( P = I \), which is the identity matrix. Moreover, the sketch of \( A \) of Algorithm 3 is oblivious to the choice of the post-processing matrix \( P \), so we would like to repeat this \( k \) times by creating \( k \) separate instances of the \( L_{2,2} \) sampler of Algorithm 3 and for the \( j \)th instance, multiply by the post-processing matrix \( P_j = I - R_{j-1}^T R_{j-1} \). Unfortunately, if \( f(j) \) is the index of the row of \( AP \) that is selected in the \( j \)th round, the row \( r_j \) that the \( L_{2,2} \) sampler outputs is not \( f(j)P_j \) but rather a noisy perturbation of it, which means in future rounds we are not sampling with respect to a subspace containing \( AP \), but rather a subspace containing \( r_j \). This is particularly a problem if \( r_j \) is parallel to another row \( A_i \) that is not contained in the subspace of \( AP \), then in future rounds the probability of sampling \( A_i \) is zero, when it should in fact be nonzero. Although the above example shows that the noisy perturbation does not preserve relative sampling probabilities for each row, we show that the perturbations give a good additive approximation to the sampling probabilities. That is, we bound the total variation distance between sampling with respect to the true rows of \( A \) and sampling with respect to the noisy rows of \( A \). We give our algorithm in full in Algorithm 4.

#### Algorithm 4 Noisy Adaptive Sampler

**Input:** Matrix \( A \in \mathbb{R}^{n \times d} \) that arrives as a stream \( A_1, \ldots, A_{\ell} \in \mathbb{R}^{d \times d} \) parameter \( k \) for number of rows to be sampled, constant parameter \( \epsilon > 0 \).

**Output:** \( k \) Noisy and projected rows of \( A \).

1. Create instances \( A_{11}, \ldots, A_{1k} \) of the \( L_{2,2} \) sampler of Algorithm 3 where the number of buckets \( b = \Theta \left( \frac{\log^2 n}{\epsilon^2} \right) \) is sufficiently large.
2. Let \( M \) be empty \( 0 \times d \) matrix.
3. **Streaming Stage:**
   1. for each row \( A_i \) do
   2. Update sketch \( A_{i1}, \ldots, A_{ik} \).
4. **Post-processing Stage:**
   7. for \( j = 1 \) to \( j = k \) do
   8. Post-processing matrix \( P \leftarrow \frac{1}{M} \).
   9. Update \( A_j \) with post-processing matrix \( P \).
   10. Let \( r_j \) be the noisy row output by \( A_j \).
   11. Append \( r_j \) to \( M \): \( M \leftarrow M \circ r_j \).
12. return \( M \).

For the purpose of the analysis, we first show that if the \( L_{2,2} \) sampler outputs row \( r_1 \) that is a noisy perturbation of row \( A_{f(1)}P \), then not only can we bound \( \left\| A_{f(1)}P - r_1 \right\|_2 \) as in Lemma 2.2, but also we can bound the norm of the component of \( r_1 \) orthogonal to \( A_{f(1)}P \). This is significant because future rounds of sampling will focus on the norms of the orthogonal components for the sampling probabilities.

**Lemma 3.1.** Given a matrix \( A \in \mathbb{R}^{n \times d} \) and a matrix \( P \in \mathbb{R}^{d \times d} \), as defined in Line 8 and round \( i \leq k \), of Algorithm 4, suppose index \( j \in [n] \) is sampled (in round \( i \)). Then with high probability, the sampled (noisy) row \( r_i \) satisfies \( r_i = A_jP + v \), with

\[
\left\| v \right\|_2 \leq \frac{\epsilon \sqrt{C} \|AP\|_F}{\sqrt{C \log n} \|AP\|_F} \left\| A_jP \right\|_2,
\]

for any projection matrix \( Q \in \mathbb{R}^{d \times d} \). Hence, \( v \) is orthogonal to each noisy row \( r_{y} \), where \( y \in [i-1] \).
Recall that our sampler only returns noisy rows \( r_1 \), rather than the true rows of \( AP_1 \), where \( P_1 \) is any post-processing matrix. This is problematic for multiple rounds of sampling, since \( r_1 \) is then used to form the next post-processing matrix \( P_{k+1} \), rather than the true row. We next show that the total variation distance has not been drastically altered by sampling with respect to the noisy rows rather than the true rows.

The main idea is that because the noise in each direction is proportional to the total mass in the subspace by Lemma 3.1, we can bound the total perturbation in the squared norms of each row of \( AP_1 \). We first argue that if we obtain a noisy row in the first sampling iteration but then we obtain the true rows in the subsequent iterations, then the total variation distance between the resulting probability distribution of sampling each row is close to the ideal probability distribution of sampling each row if we had obtained the true rows over all iterations. It then follows from triangle inequality that the actual sampling distribution induced by obtaining noisy rows in each round is close to the ideal sampling distribution if we had obtained the true rows.

To bound the perturbation in the sampling probability of each row, we require a change of basis matrix from a representation of vectors in terms of the true rows of \( A \) to a representation of vectors in terms of the noisy rows of \( A \). This change of basis matrix crucially must be close to the identity matrix, in order to preserve the perturbation in the squared norms.

**Lemma 3.2.** Let \( f(1) \) be the index of a noisy row \( r_1 \) sampled in the first iteration of Algorithm 4. Let \( P_1 \) be a process that projects away from \( A_{f(1)} \) and iteratively selects \( k-1 \) additional rows of \( A \) through adaptive sampling (with \( p = 2 \)). Let \( P_2 \) be a process that projects away from \( r_1 \) and iteratively select \( k-1 \) additional rows of \( A \) through adaptive sampling (with \( p = 2 \)). Then for \( \epsilon < \frac{1}{d} \), the total variation distance of \( P_1 \) and \( P_2 \) is \( O(k\epsilon) \).

Since the total variation distance induced by a single noisy row is small, we obtain that the total variation distance between offline adaptive sampling and our adaptive sampler is small by rescaling the error parameter. Thus we now provide the full guarantees for our adaptive sampler.

**Theorem 3.3.** Given a matrix \( A \in \mathbb{R}^{n \times d} \) that arrives in a turnstile stream, there exist a one-pass algorithm \textsc{AdaptiveStream} that outputs a set of \( k \) indices such that the probability distribution for each set of \( k \) indices has total variation distance \( \epsilon \) of the probability distribution induced by adaptive sampling with respect to squared distances to the selected subspace in each iteration. The algorithm uses \( O\left(\frac{d^2k^5}{\epsilon^3} \log^6 n\right) \) bits of space.

Note that the proof of Lemma 3.2 also showed that \( \|AY\|_F^2 \) is within a \( (1+O(\epsilon)) \) factor of \( \|AZ\|_F^2 \). In Theorem 3.3, we have now sampled \( k \) noisy rows rather than a single noisy row followed by \( k-1 \) true rows, but we also rescale the error parameter down to \( O\left(\frac{\epsilon}{\sqrt{k}}\right) \).

**Corollary 3.4.** Suppose Algorithm 4 samples noisy rows \( r_1, \ldots, r_k \) rather than the actual rows \( A_{f(1)}, \ldots, A_{f(k)} \). Let \( T_k = A_{f(1)} \circ \cdots \circ A_{f(k)} \), \( Z_k = I - T_k^*T_k \), \( R_k = r_1 \circ \cdots \circ r_k \) and \( Y_k = I - R_k^*R_k \). Then \( \frac{1}{1-\epsilon} \|AY_k\|_F^2 \leq \|AZ_k\|_F^2 \leq (1+\epsilon) \|AY_k\|_F^2 \) with probability at least \( 1-\epsilon \).

At first glance, it might seem strange that Corollary 3.4 obtains increased accuracy with higher probability, but recall that Algorithm 4 has a space dependency on \( \frac{1}{\epsilon^2} \).

### 4 VOLUME MAXIMIZATION IN THE ROW-ARRIVAL MODEL

In this section, we consider the volume maximization problem on row-arrival streams. As before, we are given the rows of the matrix \( A \in \mathbb{R}^{n \times d} \) and a parameter \( k \), and the goal is to output \( k \) rows of the matrix whose volume is maximized. Throughout the section, we will use the equivalent view that the stream consists of \( n \) points from \( \mathbb{R}^d \).

#### 4.1 Volume Maximization via Composable Core-sets.

We first observe that we can get an approximation algorithm for volume maximization in the row-arrival model by using algorithms of [21, 22] for composable core-sets for volume maximization. We recall the definition of composable core-sets, such as given in [23].

**Definition 4.1 (\( \alpha \)-composable core-set).** Let \( V \subset \mathbb{R}^d \) be an input set. Then a function \( c : V \rightarrow W \), where \( W \subset V \), is an \( \alpha \)-composable core-set for an optimization problem with a maximization objective with respect to a function \( f : 2^W \rightarrow \mathbb{R} \) if for any collection of set \( V_1, \ldots, V_m \subset \mathbb{R}^d \),

\[
    f(c(V_1) \cup \cdots \cup c(V_m)) \geq \frac{1}{\alpha} f(V_1 \cup \cdots \cup V_m).
\]

[22] gives composable core-sets for volume maximization.

**Theorem 4.2.** [22] There exists a polynomial time algorithm for computing an \( O(k^{1/2}) \)-composable core-set of size \( O(k) \) for the volume maximization problem.

We can partition the stream into consecutive blocks and apply a core-set for each block to get a streaming algorithm for volume maximization.

**Corollary 4.3.** There exists a one pass streaming algorithm in the row-arrival model that computes a \( O(k^{1/(2\epsilon)}) \)-approximation to the volume maximization problem, using \( O\left(\frac{k^2}{\epsilon^2} n^d k \right) \) space.

#### 4.2 Exponential Dependence on \( d \)

In this section, we give a streaming algorithm whose space complexity depends exponentially on the dimension \( d \). Our main tool is the \( \epsilon \)-kernels of \cite{1} improved by [9] for directional width of a point set. We first define the concept of the directional width.

**Definition 4.4 (Directional width \cite{1}).** Given a point set \( P \subset \mathbb{R}^d \) and a unit direction vector \( x \in \mathbb{R} \), the directional width of \( P \) with respect to \( x \) is defined to be \( w(x, P) = \max_{p \in P} \langle x, p \rangle - \min_{p \in P} \langle x, p \rangle \).

The following lemma shows the existence of core-sets with size exponential in the directional width of a point set but independent of the number of points.
Lemma 4.5. [9] For any $0 < \epsilon < 1$, there exists a one pass streaming algorithm that computes an $\epsilon$-core-set $Q \subseteq P$, such that for any unit direction vector $x \in \mathbb{R}^d$, we have $\omega(x, Q) \geq (1 - \epsilon)\omega(x, P)$. Moreover, the algorithm uses space $O\left(\left(\frac{1}{\epsilon} \log \frac{1}{\epsilon}\right)^{d-1}\right)$ and the core-set has size $|Q| \leq O\left(\frac{1}{\epsilon^{d-1}}\right)$.

Now let us define the directional height (which was implicitly defined in [21]), and its relation to directional width.

Definition 4.6 (Directional height). Given a point set $P \subseteq \mathbb{R}^d$ and a unit direction vector $x \in \mathbb{R}$, the directional height of $P$ with respect to $x$ is defined to be $h(x, P) = \max_{p \in P} \langle x, p \rangle$, where $\mathcal{H}^x$ is the orthogonal complement of $\mathcal{H}$.

Lemma 4.7. For a point set $P \subseteq \mathbb{R}^d$, an $\epsilon$-core-set $Q$ for directional height with a $(2\epsilon)$-core-set for directional height.

Finally we define $k$-directional height and observe that a core-set for directional height leads to a core-set for $k$-directional height.

Definition 4.8 (k-directional height [21]). Given a point set $P \subseteq \mathbb{R}^d$ and a $(k - 1)$-dimensional subspace $\mathcal{H}$, $x \in \mathbb{R}^d$, the $k$-directional height of $P$ with respect to $\mathcal{H}$ is defined to be $h_k(\mathcal{H}, P) = \max_{x \in \mathcal{H}^x} h(x, P)$, where $\mathcal{H}^x$ is the orthogonal complement of $\mathcal{H}$.

Observation 4.9. For a point set $P \subseteq \mathbb{R}^d$, an $\epsilon$-core-set $Q$ for directional height is an $\epsilon$-core-set for $k$-directional height.

In fact, a core-set for directional height is stronger than a core-set for $k$-directional height since it preserves the height in all directions $x \in \mathcal{H}^x$; thus their maximum is preserved too.

Lemma 4.10 ([21]). For a point set $P \subseteq \mathbb{R}^d$, let $Q$ be its $\epsilon$-core-set for $k$-directional height. Then the solution of the $k$-volume maximization on $Q$ is within a factor of $1/(1 - \epsilon)^k$ of the solution of $k$-volume maximization over $P$.

Lemma 4.11. There exists a one pass streaming algorithm that outputs a $2^k$-approximation to volume maximization, using $O\left(8^d\right)$ space.

4.3 Dimensionality Reduction

In this section, we show how to reduce the dimension of each point to $d = O(k)$. Using the result of the previous section, this will give a trade-off algorithm, improving over Corollary 4.3 in terms of the dependence on the parameter $\frac{1}{\epsilon}$. We prove the following lemma.

Lemma 4.12. Let $C$ be a trade-off parameter such that $1 < C < (\log n)/k$. There exists a randomized streaming algorithm that uses $O\left(n^{O(1/C)}d\right)$ space to computes a subset of size $d$ whose volume maximization solution is a $O(Ck)^{k/2}$ approximation to the optimal solution.

Note that this result improves the algorithm of Corollary 4.3 for $\frac{k}{\log n} < \epsilon$: setting $C = 1/\epsilon$, this provides an algorithm with memory usage of $O(n^\epsilon)$, with approximation factor of $O\left(k^{\epsilon/k}d^2\right)$, improving the dependence of the approximation factor on $\frac{1}{\epsilon}$ by an exponential factor.

We now continue with the proof of Lemma 4.12. Consider a random matrix $G \in \mathbb{R}^{d \times r}$, for $r = \frac{\log n}{m}$, where each of its entries is an independent and identically distributed (i.i.d.) random variable drawn from the Gaussian distribution $N(0, 1/r)$. Consider the matrix $AG$ and observe that its rows exist in an $r$ dimensional space. Therefore, we can use the streaming algorithm of Lemma 4.11 to find a subset of $k$ rows of $AG$ that serves as a good estimator for the maximum volume. This approach requires $O\left(2^{2k}\right) = O\left(1^{O(\log n/C)}\right)$ memory space.

Lemma 4.13. Let $G \in \mathbb{R}^{d \times m}$, for $r = \Omega\left(\frac{n}{\log n}\right)$, have each of its entries is drawn i.i.d from the Gaussian distribution $N(0, 1/r)$. With high probability, the maximum volume of the optimal $k$-subset of the rows of $AG$ is within $2^k$ of the maximum volume of the optimal $k$-subset of the rows of $A$.

We now show that for every other subset $S$ of $k$ points from the rows of $A$, their volume does not increase by much with very high probability, so that we can union bound over all such subsets. The following lemma may seem counterintuitive at first, since the parameter $C$ appears in the approximation factor but not the probability. However, recall that the algorithm pays for the parameter $C$ in the space of the algorithm.

Lemma 4.14. Let $S$ be a subset of size $k$ from the rows of $A$. Then after applying $G$, its volume does not increase by more than a factor of $(\sqrt{2Ck} + 2)^k = O\left(Ck^{k/2}\right)$ with probability at least $1 - n^{-k}$.

Thus we can union bound over all subsets of size $k$ of the $n$ rows of $A$, to argue that with high probability, none of them will have a volume increase by more than a $(\sqrt{2Ck} + 2)^k$ factor. This completes the proof of Lemma 4.12.

5 VOLUME MAXIMIZATION LOWER BOUNDS

In this section, we complement our adaptive sampling based volume maximization algorithms with lower bounds on turnstile streams that are tight up to lower order terms. Our lower bounds hold even for multiple passes through the turnstile stream. Additionally, we give a lower bound for volume maximization in the random order row-arrival model that is competitive with the algorithms in Section 4.

5.1 Turnstile Streams

We first consider lower bounds for turnstile streams. In the Gap $\ell_\infty$ problem, Alice and Bob are given vectors $x$ and $y$ respectively with $x, y \in [0, m]^n$ for some $m > 0$ and the promise that either $|x_i - y_i| \leq 1$ for all $i \in [n]$ or there exists some $i \in [n]$ such that $|x_i - y_i| = m$. The goal is for Alice and Bob to perform some communication protocol to decide whether there exists an index $i \in [n]$ such that $|x_i - y_i| = m$, possibly over multiple rounds of communication. To succeed with probability $\frac{1}{2}$, Alice and Bob must use at least $\Omega\left(\frac{n}{m}\right)$ bits of total communication, even if they can communicate over multiple rounds.

Theorem 5.1. [7] Any protocol that solves the Gap $\ell_\infty$ problem with probability at least $\frac{8}{9}$ requires $\Omega\left(\frac{n}{m}\right)$ total bits of communication.

We first reduce an instance of the Gap $\ell_\infty$ problem to giving an $\alpha$-approximation to the volume maximization problem when $k = d = 1$. 

1261
We now present streaming lower bounds for the row-arrival model. This forms the YES case. The goal is for Alice and Bob to perform least $C$ the random order row-arrival model. For constant $\alpha$ and the rows of that permutation constitute the stream in $A$ be adversarial.

Recall that for problems that are invariant to the permutation of $A$ requires $\Omega\left(\frac{n}{k^{\alpha}}\right)$ bits of space.

### 5.2 Row-Arrival Model

We now present streaming lower bounds for the row-arrival model. We consider a version of the distributional set-disjointness communication problem $\text{DIS}_{n,d}$ in which Alice is given the set of vectors $U = \{u_1, \ldots, u_n\}$ and Bob is given the set of vectors $V = \{v_1, \ldots, v_n\}$. With probability $\frac{1}{2}$, $U$ and $V$ are chosen uniformly at random among all instances with the following properties:

- Any vector in $U \cup V$ is in $\{0,1\}^d$ and moreover its weight is exactly $\frac{d}{2}$.
- $U \cap V$ is non-empty.

This forms the NO case. Otherwise with probability $\frac{3}{4}$, $U$ and $V$ are chosen uniformly at random among all instances with the following properties:

- Any vector in $U \cup V$ is in $\{0,1\}^d$ and moreover its weight is exactly $\frac{d}{2}$.
- $U \cap V = \emptyset$.

This forms the YES case. The goal is for Alice and Bob to perform some communication protocol to decide whether the instance is a YES or a NO instance, i.e., whether $U \cap V = \emptyset$, possibly over multiple rounds of communication.

The following result, originally due to Razborov [32] and generalized by others [26, 36], lower bounds the communication complexity of any randomized protocol that solves $\text{DIS}_{n,d}$ with probability at least $\frac{1}{2}$, even given multiple rounds of communication.

**Theorem 5.4.** [26, 32, 36] Any protocol for $\text{DIS}_{n,d}$ that fails with probability at most $\frac{1}{2}$ requires $\Omega(n)$ bits of total communication.

We first reduce an instance of the distributional set-disjointness problem to giving a $C^k$ approximation to the volume maximization problem in the row-arrival model when the order of the stream can be adversarial.

**Theorem 5.5.** For constant $p$ and $C = \frac{16}{\log_2(p)}$, any $p$-pass streaming algorithm that outputs a $C^k$ approximation to the $(2k)$-volume maximization problem in the row-arrival model with probability at least $\frac{1}{2}$ requires $\Omega(n)$ bits of space.

Recall that for problems that are invariant to the permutation of the rows of the input matrix $A$, once the entries of $A$ are chosen, an arbitrary permutation of the rows of $A$ is chosen uniformly at random, and the rows of that permutation constitute the stream in the random order row-arrival model.

**Corollary 5.6.** For $C = \frac{16}{\log_2(p)}$, any one-pass streaming algorithm that outputs a $C^k$ approximation to the $2k$-volume maximization problem in the random order row-arrival model with probability at least $\frac{1}{2}$ requires $\Omega(n)$ bits of space.

### A NOISY DISTANCE SAMPLING

#### A.1 $L_{1,2}$ Sampler

Recall that for a matrix $A \in \mathbb{R}^{n \times d}$, we define the $L_{p,q}$ norm of $A$ by

$$
\|A\|_{p,q} = \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{d} |A_{i,j}|^q\right)^{\frac{p}{q}}\right)^{\frac{1}{p}}.
$$

In this section, we describe an algorithm for sampling rows of a matrix $AP$ with probability proportional to $\|AP\|_2$, which we call $L_{1,2}$ sampling. By comparison, in Section 2 we sampled rows of $AP$ with probability proportional to $\|AP\|_2^2$, which can be seen as $L_{2,2}$ sampling.

Before describing our general $L_{1,2}$ sampler, we need a subroutine similar to AMS-M for estimating $\|AP\|_2$, when the data stream updates entries of $A$ and query access to $P$ is only given in post-processing. We first describe a turnstile streaming algorithm of [4] that can be used to compute a constant factor approximation to $\|AP\|_2$ and then we show that it can be modified to approximate $\|AP\|_{1,2}$ due its nature of being a linear sketch. For each $j$, define the level sets $S_j$ by $\{i \in [n] : \|A_{i,j}\|_2 < \|A_{i,j}\|_2 \geq \frac{\|A_{i,j}\|_2}{2^{j}}\}$. The algorithm of [4] approximates the number of rows in each level set $S_j$ by first implicitly subsampling rows at different rates. The rows that are sampled at each rate then form a level and the rows in a particular level are then aggregated across a number of buckets. The norms of the aggregates across each bucket are then computed and by rescaling the number of aggregates that are in each level set, we obtain an accurate estimate of the sizes of the level sets. The sizes of the level sets are then used to output a good approximation to $\|AP\|_{1,2}$.

Crucially, the aggregates of the rows in the algorithm of [4] is a linear combination of the rows. Hence by taking the aggregates and multiplying by $P$ after the stream ends, we obtain aggregates of the rows of $AP$, which can then be used to estimate the sizes of the level sets of $\|AP\|_{1,2}$. The algorithm of [4] uses $d \cdot \text{polylog}(n)$ space by storing aggregates of entire rows for each bucket across multiple levels. Thus, we have the following:

**Lemma A.1.** [4] There exist a fixed constant $\xi > 1$ and a one-pass turnstile streaming algorithm $\text{ESTIMATOR-M}$ that takes updates to entries of a matrix $A \in \mathbb{R}^{n \times d}$, as well as query access to post-processing matrices $P \in \mathbb{R}^{d \times d}$ and $M \in \mathbb{R}^{n \times d}$ that arrive after the stream, and outputs a quantity $\hat{F}$ such that $\|AP - M\|_{1,2} \leq \hat{F} \leq \xi \|AP - M\|_{1,2}$. The algorithm uses $d \cdot \text{polylog}(n)$ bits of space and succeeds with high probability.

Using the $L_{1,2}$ estimator, we can develop a $L_{1,2}$ sampler similar to our $\ell_2$ sampler.

We first show the probability that the tail is too large, i.e., $\hat{S} > \frac{c \cdot \text{polylog}(n)}{\epsilon \cdot \hat{F}}$, is independent of the index $i$ and the value of $\ell_i$. The proof is almost verbatim to Lemma 2.5, but the thresholds now depend on $\|AP\|_{1,2}$ rather than $\|AP\|_F$.

**Lemma A.2.** For each $j \in [n]$ and value of $\ell_j$,

$$
\Pr\left[\hat{S} > \frac{c \cdot \text{polylog}(n)}{\epsilon \cdot \hat{F}}\right] = O(\epsilon) + \frac{1}{\text{poly}(n)}.
$$
Algorithm 5 Single $L_{1,2}$ Sampler

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$ that arrives as a stream $A_1, \ldots, A_n \in \mathbb{R}^d$, matrix $P \in \mathbb{R}^{d \times d}$ that arrives after the stream, constant parameter $\epsilon > 0$.

**Output:** Noisy row $r$ of $AP$ sampled roughly proportional to the row norms of $AP$.

1. **Pre-processing Stage:**
   1. $b \leftarrow \Omega \left( \frac{1}{\epsilon^2} \right)$, $r \leftarrow \Theta(\log n)$ with sufficiently large constants
   2. For $i \in [n]$, generate independent scaling factors $t_i \in [0, 1]$ uniformly at random.
   3. Let $B$ be the matrix consisting of rows $B_i = \frac{1}{t_i} A_i$.
   4. Let $\text{Estimator-M}$ and $\text{AMS-M}$ track the $L_{1,2}$ norm of $A$ and Frobenius norm of $B$, respectively.
   5. Let $\text{CountSketch-M}$ be an $r \times b$ table, where each entry is a vector $\mathbb{R}^{d}$.

2. **Streaming Stage:**
   1. For each row $A_i$ do
      1. Update $\text{CountSketch-M}$ with $B_i = \frac{1}{t_i} A_i$.
      2. Update linear sketch $\text{Estimator-M}$ with $A_i$.
      3. Update linear sketch $\text{AMS-M}$ with $B_i = \frac{1}{t_i} A_i$.

3. **Processing $P$ Stage:**
   1. After the stream, obtain matrix $P$.
   2. Multiply each vector $v$ in each entry of the $\text{CountSketch-M}$ table by $P$: $v \leftarrow vP$.
   3. Multiply each vector $v$ in $\text{AMS-M}$ by $P$: $v \leftarrow vP$.
   4. Multiply each vector $v$ in $\text{Estimator-M}$ by $P$: $v \leftarrow vP$.

4. **Extraction Stage:**
   1. Use $\text{Estimator-M}$ to compute $\hat{F}$ with $\|AP\|_{L_2} \leq \hat{F} \leq \xi \|AP\|_{L_2}$.
   2. Extract the $\frac{\epsilon}{\sqrt{C}}$ (noisy) rows of $BP$ with the largest estimated norms by $\text{CountSketch-M}$.
   3. Let $M$ be the $\frac{\epsilon}{\sqrt{C}}$-sparse matrix consisting of these top (noisy) rows.
   4. Use $\text{AMS-M}$ to compute $\hat{S}$ with $\|BP - M\|_F \leq \hat{S} \leq 2\|BP - M\|_F$.
   5. Let $r_j$ be the (noisy) row in $\text{CountSketch-M}$ with the largest norm.
   6. Let $C > 0$ be some large constant so that the probability of failure is $O\left(\frac{1}{C\log C}\right)$.
   7. If $\hat{S} > \frac{C\log n}{\epsilon^2} \hat{F}$ or $\|r_j\|_2^2 < \frac{C\log n}{\epsilon^2} \hat{F}$ then
      1. return FAIL.
   8. else
      1. return $r = t_i r_i$.

**Lemma A.3.** Conditioned on a fixed value of $\hat{F}$, the probability that Algorithm 5 outputs (noisy) row $i$ is $(1 \pm O(\epsilon)) \frac{\|AP\|_2}{\hat{F}} + \frac{1}{\text{poly}(n)}$.

We now provide the full guarantees of the $L_{1,2}$ sampler.

**Theorem A.4.** Given $\epsilon > 0$, there exists a one-pass streaming algorithm that takes rows of a matrix $A \in \mathbb{R}^{n \times d}$ as a data stream and a matrix $P \in \mathbb{R}^{d \times d}$ after the stream, and outputs (noisy) row $i$ of $AP$ with probability $(1 \pm O(\epsilon)) \frac{\|AP\|_2}{\|AP\|_{L_2}} + \frac{1}{\text{poly}(n)}$. The algorithm uses $O\left(d \log \left(\frac{1}{\epsilon}, \log n\right)\right)$ bits of space and succeeds with high probability.

A.2 Noisy Adaptive Distance Sampling

Our algorithm for noisy adaptive distance sampling, given in Algorithm 6, is similar to Section 3, except it uses the $L_{1,2}$ sampling primitive of Theorem A.4 instead of the $L_{2,2}$ sampler.

Algorithm 6 Noisy Adaptive Sampler

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$ that arrives as a stream $A_1, \ldots, A_n \in \mathbb{R}^d$, parameter $k$ for number of sampled rows, constant parameter $\epsilon > 0$.

**Output:** $k$ Noisy and projected rows of $A$.

1. Create instances $\mathcal{A}_1, \ldots, \mathcal{A}_k$ of the $L_{1,2}$ sampler of Algorithm 5 where the number of buckets $b = \Theta\left(\frac{1}{\epsilon^4} n\right)$ is sufficiently large.
2. Let $M$ be empty $0 \times d$ matrix.
3. **Streaming Stage:**
   1. For each row $A_i$ do
      1. Append $A_i$ to $M$.
      2. Update sketch $\text{Estimator-M}$ with $A_i$.
      3. Update sketch $\text{AMS-M}$ with $A_i$.
      4. Update sketch $\text{CountSketch-M}$ with $A_i$.
   2. Post-processing Stage:
      1. for $j = 1$ to $k$ do
         1. Post-processing matrix $P \leftarrow I - M^T M$.
         2. Update $\mathcal{A}_j$ with post-processing matrix $P$.
         3. Let $r_j$ be the noisy row output by $\mathcal{A}_j$.
      4. Append $r_j$ to $M$: $M \leftarrow M \cup r_j$.
5. return $M$.

We first bound the norm of the perturbation of the sampled row at each instance.

**Lemma A.5.** Given a matrix $A \in \mathbb{R}^{n \times d}$ and a matrix $P \in \mathbb{R}^{d \times d}$, as defined in Line 8 and round $i \leq k$, of Algorithm 5, suppose index $j \in [n]$ is sampled (in round $i$). Then with high probability, the sampled (noisy) row $r_i$ satisfies $r_i = A_j P + v_e$ with

$$\|v_e\|_2 \leq \frac{3}{\epsilon^3} \frac{\|AP\|_{L_2}}{\|AP\|_{L_2}} \|A_j P\|_2,$$

for any projection matrix $Q \in \mathbb{R}_{d \times d}$. Hence, $v_e$ is orthogonal to each noisy row $r_y$, where $y \in [k - 1]$.

We now bound the total variation distance between the distribution of sampled rows and the distribution of adaptive sampling with respect to distances to selected subspace. The proof is almost verbatim to Lemma 3.2, except we now consider the probabilities with respect to the distances to the previous subspace, rather than the squared distances.

**Lemma A.6.** Let $f(1)$ be the index of a noisy row $r_1$, sampled in the first iteration of Algorithm 6. Let $P_1$ be a process that projects away from $A_{f(1)}$ and iteratively selects $k - 1$ additional rows of $A$ through adaptive sampling (with $p = 1$). Let $P_2$ be a process that projects away from $r_1$ and iteratively selects $k - 1$ additional rows of $A$ through adaptive sampling (with $p = 1$). Then for $\epsilon < \frac{1}{\sqrt{2}}$, the total variation distance of $P_1$ and $P_2$ is $O(k\epsilon)$.

**Corollary A.7.** Suppose Algorithm 6 samples noisy rows $r_1, \ldots, r_k$ rather than the actual rows $A_{f(1)}, \ldots, A_{f(k)}$. Let $T_k = A_{f(1)} \circ \ldots \circ A_{f(k)}$. We now provide the full guarantees of the $L_{1,2}$ sampler.
\( A_f(k), Z_k = I - T_k^T T_k, R_k = r_1 \cdots r_k \) and \( Y_k = I - R_k^T R_k \). Then 
\[
(1 - \epsilon) \| A Y_k \|_{1,2} \leq \| A Z_k \|_{1,2} \leq (1 + \epsilon) \| A Y_k \|_{1,2}
\]
with probability at least \( 1 - \epsilon \).

ACKNOWLEDGMENTS

D. Woodruff acknowledges support in part from the National Science Foundation under Grant No. CCF-1815840.

REFERENCES

[1] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. 2005. Geometric approximation via coresets. Combinatorial and computational geometry 52 (2005), 1–30.
[2] Pankaj K. Agarwal and R. Sharathkumar. 2015. Streaming Algorithms for Extent Problems in High Dimensions. Algorithmica 72, 1 (2015), 83–98.
[3] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. J. Comput. Syst. Sci. 58, 1 (1999), 137–147.
[4] Alexandru Andoni, Khansh Do Ba, Piotr Indyk, and David P. Woodruff. 2009. Efficient Sketches for Earth-Mover Distance, with Applications. In 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS. 324–330.
[5] Alexandru Andoni, Robert Krauthgamer, and Krzysztof Onak. 2011. Streaming Algorithms via Precision-Sampling. In IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS. 363–372.
[6] Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. 2002. Approximate clustering via core-sets. In Proceedings on 34th Annual ACM Symposium on Theory of Computing. 250–257.
[7] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2004. An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. 68, 4 (2004), 702–732.
[8] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. 2009. An improved approximation algorithm for the column subset selection problem. In Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. 968–977.
[9] Timothy M. Chan. 2006. Faster core-set constructions and data-stream algorithms in fixed dimensions. Computational Geometry 35, 1-2 (2006), 20–35.
[10] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. Theor. Comput. Sci. 312, 1 (2004), 3–15.
[11] Ke Chen. 2009. On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. SIAM J. Comput. 39, 3 (2009), 923–947.
[12] Ali Çiçek and Malik Magdon-Ismail. 2009. On selecting a maximum volume sub-matrix of a matrix and related problems. Theor. Comput. Sci. 410, 47-49 (2009), 4801–4811.
[13] Kenneth L. Clarkson and David P. Woodruff. 2015. Input Sparsity and Hardness for Robust Subspace Approximation. In IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS. 310–329.
[14] Michael B. Cohen, Cameron Musco, and Christopher Musco. 2017. Input Sparsity Time Low-rank Approximation via Ridge Leveraging Score Sampling. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017. 1758–1777.
[15] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. 2006. Matrix Approximation and Projective Clustering via Volume Sampling. Theory of Computing 2, 12 (2006), 225–247.
[16] Amit Deshpande and Kasturi R. Varadarajan. 2007. Sampling-based dimension reduction for subspace approximation. In Proceedings of the 39th Annual ACM Symposium on Theory of Computing. 641–650.
[17] Amit Deshpande and Santosh Vempala. 2006. Adaptive Sampling and Fast Low-Rank Approximation. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings. 292–303.
[18] Dan Feldman, Moritz Monemizadeh, Christian Sohler, and David P. Woodruff. 2010. Coresets and Sketches for High Dimensional Subspace Approximation Problems. In Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. 630–649.
[19] Venkatesan Guruswami and Ali Kemal Sinop. 2012. Optimal column-based low-rank matrix reconstruction. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. 1207–1214.
[20] Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing. 291–300.
[21] Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Aliereza Rezaei. 2019. Composable Core-sets for Determinant Maximization: A Simple Near-Optimal Algorithm. In International Conference on Machine Learning. 4245–4263.
[22] Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Aliereza Rezaei. 2020. Composable Core-sets for Determinant Maximization Problems via Spectral Spanners. In Proceedings of the thirty-first annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics.
[23] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. 2014. Composable core-sets for diversity and coverage maximization. In Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PDS. 100–108.
[24] Rajesh Jayaram and David P. Woodruff. 2018. Perfect Lp Sampling in a Data Stream. In 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS. 544–555.
[25] Hossein Jowhari, Mert Saglam, and Gábor Tardos. 2011. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PDS. 49–58.
[26] Bala Kalyanasundaram and Georg Schnitger. 1992. The Probabilistic Communication Complexity of Set Intersection. SIAM J. Discrete Math. 5, 4 (1992), 345–557.
[27] Michael Kerber and Sharath Raghvendra. 2015. Approximation and Streaming Algorithms for Projective Clustering via Random Projections. In Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG.
[28] Rosi Levin, Anish Prasad Sevekari, and David P. Woodruff. 2018. Robust Subspace Approximation in a Stream. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS. 10706–10716.
[29] Sepideh Mahabadi, Ilya Razenshtein, David P. Woodruff, and Sanmson Zhou. 2020. Non-Adaptive Sampling on Turnstile Streams. CoRR abs/2004.10969 (2020).
[30] Moritz Monemizadeh and David P. Woodruff. 2010. 1-Pass Relative-Error Lp-Sampling with Applications. In Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. 1143–1160.
[31] Cecilia M. Procopiuc. 2017. Projective Clustering. In Encyclopedia of Machine Learning and Data Mining. 1018–1025.
[32] Alexander A. Razborov. 1992. On the Distributionsal Complexity of Disjoinsness. Theor. Comput. Sci. 106, 2 (1992), 385–390.
[33] Nariankadu D. Shyamalkumar and Kasturi R. Varadarajan. 2012. Efficient Subspace Approximation Algorithms. In Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. 1207–1214.
[34] Christian Sohler and David P. Woodruff. 2011. Subspace embeddings for the L2-norm with applications. In Proceedings of the 54th Annual IEEE Symposium on Theory of Computing, STOC. 755–764.
[35] Christian Sohler and David P. Woodruff. 2018. Strong Coresets for k-Median and Subspace Approximation: Goodbye Dimension. In 99th IEEE Annual Symposium on Foundations of Computer Science, FOCS. 802–813.
[36] David P. Woodruff and Qin Zhang. 2012. Tight bounds for distributed functional monitoring. In Proceedings of the 44th Symposium on Theory of Computing, STOC. 941–960.