Unbiased Space Saving Sketches for Inner Product Estimation in Disaggregated Data

Algorithmic Machine Learning and Data Science, Fall 2023

Suemy Inagaki
suemy.inagaki@nyu.edu

Felipe de Oliveira
felipe.oliveira@nyu.edu

Abstract

In the realm of big data, estimating inner product in extensive streaming data is crucial. Existing methods tackle this for pre-aggregated data but face challenges due to its costliness. Our study comprehensively explores related research, elaborates on our proposed sketches and demonstrates experimental results on synthetic data. In particular, we investigate the adaptation of Unbiased Space Saving algorithms to estimate inner product between disaggregated streaming datasets, aiming to address Subset Sum Estimation, the Frequent Item Problem, and Inner Product Estimation for disaggregated data.

1 Introduction

The estimation of inner product in large streaming data holds significance within big data scenarios [7]. This task finds application in various areas, such as analyzing correlations between extensive streams or estimating the join size of tables.

Other pertinent challenges in this domain include the frequent items problem and subset sum estimation. The former aims to identify the most frequent items within a stream, while the latter involves estimating subset sums in a vast dataset without storing all data. Disaggregated subset-sum is a variant that requires data to be disaggregated. It becomes essential in contexts such as clickstream data, where calculating user clicks on an ad may demand costly aggregation steps.

Our focus lies in addressing the inner product problem for disaggregated data. In this sense, we propose a method that resolves Inner Product Estimation and Subset Sum Estimation for disaggregated data and the Frequent Items Problem. Although algorithms like Fast-AGMS exist for estimating the Inner Product in disaggregated data [7], they cannot be applied to problems like subset sum estimation or the frequent items problem.

In this context, Daniel Ting introduced Unbiased Space Saving algorithms, addressing both disaggregated subset sum and the frequent item problem [6]. Ting's experiments demonstrate the sketch's superiority over methods operating on pre-aggregated data, like priority sampling, when estimating subset sum.

Building upon Ting's work [6], our project seeks to answer an important question: Can we adapt Unbiased Space Saving to estimate the inner product between two disaggregated streaming datasets? The primary contribution lies in developing an algorithm capable of addressing Subset Sum Estimation, the Frequent Items Problem, and Inner Product Estimation for disaggregated data.

This work is structured as follows: the next section covers related works. In Section 4, we describe our contributions and analysis of the research question. In Section 5 we explain and prove our proposed sketches. In Section 6 we show our experimental results for synthetic data. Finally, Section 7 presents the conclusion.

2 Problem Definition

Let a be a data stream with T_a items. In this work, time means the number of items streamed, so it takes T_a to stream all the items of a and time t to stream the first t items of the streaming. When not mentioned, we will assume that the data is disaggregated, i.e., that a is a vector with possible repeated items. We will denote by $\mathbf{n}_a(t)$ the frequency vector of the streaming a until the time t and by $n_a(i,t)$ the frequency of the item i until time t. Our goal is to estimate the inner product between the frequency vector of data streams a and b without calculating $\mathbf{n}_a(T_a)$ and $\mathbf{n}_b(T_b)$ explicitly as it can be too expensive to calculate. So our objective is to compute an estimator W that estimate:

$$\langle \mathbf{n}_a(T_a), \mathbf{n}_b(T_b) \rangle = \sum_{i \text{ item}} n_a(i,T_a) n_b(i,T_b)$$

A list of notations can be found on Table 1 .

Notation	Definition
T_a	Total number of items on streaming a
t	Number of rows encountered or time
$\hat{N}_a(i,t)$	Estimate for the count of item i at time t on the streaming a
$\hat{N}_{\min,a}(t)$	Count in the smallest bin at time t on the streaming a
$n_a(i,t)$	True count for item i at time t on the streaming a
$\mathbf{N}_a(t), \mathbf{n}_a(t)$	Vector of estimated and true counts of the streaming a at time t
$\mathbf{N}_a.keys()$	Vector of items with non-zero estimate

Table 1: Notations for disaggregated data algorithms

3 Related Works

In this section, we'll delve into several algorithms. The first two, Weighted Threshold Sampling and Weighted Priority Sampling, are tailored for Inner Product Estimation for aggregated data. Then, the Unbiased Space Saving algorithm is used for solving subset sum estimation and frequent item problems for disagregated data. Finally, the last two, AGMS and Fast AGMS, are designed for Inner Product Estimation for disaggregated data. Later on, we'll use these algorithms to compare against the results obtained using the method we propose.

3.1 Weighted Threshold Sampling

Threshold sampling weighted [3] is an inner product sketch that requires aggregated data, i.e., assume access to the frequency vector instead of the streaming data. So in this case, each entry of a vector means the frequency of an item. We will denote the vector of frequencies of the streaming a by f_a and the frequency of the item i as $f_a(i)$ as here the time is always the final of the streaming.

This method is based on Threshold sampling, a method that was long studied in statistics under the name "Poisson Sampling" but was popularized in computer science by [4]. The idea behind this method is to sample entries from f_a and f_b and use them to estimate the inner product $\langle f_a, f_b \rangle$. To do that, the algorithm samples items with probability proportional to their size, i.e., $\mathbb{P}[f_a(i) \text{ is sampled }] \propto f_a(i)$. To achieve reliable inner product estimates, the method strategically enhances the probability of items being jointly sampled on both vectors, ensuring a coordinated approach to data selection. The algorithm is presented on Algorithm 1

Algorithm 1 Weighted Threshold Sampling

```
   Assume access to a length $n$ vector of frequencies $f_a$, to an uniformly random hash function
   $h : \{1, \ldots, n\} \to [0, 1]$ and to the target sketch size $m$.
1: for $i$ such that $f_a(i) \neq 0$ do
2:     define $\tau_i = m\frac{f_a(i)^2}{\|f_a\|_2^2}$.
3:     if $h(i) \leq \tau_i$ then
4:         Sample the item $i$
5:     end if
6: end for
7: Return The items sampled and $\tau_a = m/\|f_a\|_2^2$
```

After sampling some items from vectors a and b, this algorithm estimate the inner product by Algorithm 2

Algorithm 2 Inner Product estimator for Weighted Threshold Sampling and Weighted Priority Sampling

```
   Input: Sketches $S(a)$ and $S(b)$ constructed by Algorithms 1 or 3 with the same hash function.
1: Compute the set $C$ of items on both sketches.
2: Return $w = \sum_{i \in C} \frac{f_a(i) f_b(i)}{min(1, f_a(i)^2 \tau_a, f_b(i)^2 \tau_b)}$
```

This algorithm has a theoretical guarantee stated on Fact 1

Fact 1. *For vectors $f_a, f_b \in \mathbb{R}^n$ and target sketch size m, let $S(a)$ and $S(b)$ be sketches returned by Algorithm 1. Let W be the inner product estimate returned by Algorithm 2. We have $\mathbb{E}[W] = \langle f_a, f_b \rangle$ and*

$$Var[W] \leq \frac{2}{m} max \left(\|f_{a_I}\|_2^2 \|f_b\|_2^2, \|f_a\|_2^2 \|f_{b_I}\|_2^2 \right)$$

Moreover, $\mathbb{E}[|S(a)|] \leq m$ and $\mathbb{E}[|S(b)|] \leq m$. Where $I = \{i : f_a(i) \neq 0 \text{ and } f_b(i) \neq 0\}$

The main drawback of this algorithm is that it can't ensure the exact size of the sketch. To address this issue, the same authors proposed Weighted Priority Sampling.

3.2 Weighted Priority Sampling

To tackle the problem of varying the size of the Sketch, instead of comparing $h(i)/f_a(i)^2$ with $m/\|f_a\|_2^2$ as Weighted Threshold Sampling, this method sample the m items with smallest $h(i)/f_a(i)^2$

value. The algorithm is presented in Algorithm 3

Algorithm 3 Weighted Priority Sampling

```
Assume access to a length $n$ vector of frequencies $f_a$, to an uniformly random hash function $h : \{1, \dots, n\} \to [0, 1]$ and to the target sketch size $m$.
1: for $i$ such that $f_a(i) \neq 0$ do
2:     define $\tau_a = (m+1) - th$ smallest value of $h(i)/f_a(i)^2$ or $\infty$ if $a$ has less than $m+1$ non-zero entries.
3:     if $h(i)/f_a(i)^2 \leq \tau_a$ then
4:         Sample the item $i$
5:     end if
6: end for
7: Return The items sampled and $\tau_a$
```

This algorithm also has provable guarantees as stated on Fact 2

Fact 2. *For vectors $f_a, f_b \in \mathbb{R}^n$ and target sketch size m, let $S(a)$ and $S(b)$ be sketches returned by Algorithm 3. Let W be the inner product estimate returned by Algorithm 2. We have $\mathbb{E}[W] = \langle f_a, f_b \rangle$ and*

$$Var[W] \leq \frac{2}{m-1} max \left(\|f_{a_I}\|_2^2 \, \|f_b\|_2^2 \, , \|f_a\|_2^2 \, \|f_{b_I}\|_2^2 \right)$$

Moreover, $S(a)$ and $S(b)$ have exactly m values when f_a and f_b have at least m non-zero entries

While effective, these methods necessitate pre-aggregated data, hindering their feasibility for scenarios involving disaggregated streams, where upfront frequency calculations prove computationally expensive.

3.3 Unbiased Space Saving

Unbiased Space Saving, which we will hereafter refer to as USS, is an algorithm derived from the *Space Saving* algorithm [5]. It was introduced in [6] for the disaggregated subset sum and frequent items problem. The algorithm has good practical results and some theoretical guarantees, which we will state because many of them will be extended to our proposed algorithm.

USS pseudo code is presented on Algorithm 4

Algorithm 4 Unbiased Space Saving

```
1: Initialize $S$ to be an empty list of (item, count) pairs.
2: for each new item $x$ do
3:     if $x$ is in $S$ then
4:         Increment the count of $x$ in $S$.
5:     else
6:         Find the pair $(x_{\min}, \hat{N}_{min,a}(t))$ in $S$ with the smallest count.
7:         Update $(x_{\min}, \hat{N}_{min,a}(t))$ to $(x, \hat{N}_{min,a}(t) + 1)$ with probability $1/(\hat{N}_{min,a}(t) + 1)$.
8:     end if
9: end for
```

Fact 3. *For any item x, Algorithm 4 gives an unbiased estimate of the count of x, i.e, $\mathbb{E}[\hat{N}_a(i,t)] = n_a(i,t)$*

The proof can be found on Theorem 1 of [6]

One drawback of USS algorithm is that, conditional on appearing in the sketch, individual item counts become biased upwards. This happens because items absent from the sample are falsely assumed to have zero count, artificially boosting the estimates of those actually present.

An important property of USS is that it theoretically guarantees that eventually, all frequent items will be on the sketch on i.i.d streams. It is formally stated on Fact 4

Fact 4. *Assume that items are drawn from a possibly infinite, discrete distribution with probabilities $p_1 \geq p_2 \geq \dots$ and that the sketch size is m. If $p_1 > \frac{1}{m}$, then as the number of items $t \to \infty$, the first item is on the sketch eventually.*

The proof can be found on Theorem 3 of [6]

We can use this Fact to prove a corollary and use it to define the set of frequent items.

Corollary 4.1. *If $\frac{p_i}{\sum_{j\geq i} p_j} > \frac{1}{m-i+1}$ for all $i < k$ and for some $k < m$, then for all $i < k$, the item i will be on the sketch eventually*

The proof can be found on Corollary 4 of [6]

Other result that strengths this algorithm is the Corollary 4.2. It informally states that the estimate for each frequent item, denoted as $\hat{N}_a(i,t)$, is as expected for an iid stream, i.e, $p_i \cdot t$.

Corollary 4.2. *Given the condition of corollary 4.1, the estimate $\hat{p}_i(t) = \hat{N}_a(i,t)/t$ is strongly consistent for all $i < k$ as $t \to \infty$*

The proof can be found on Corollary 5 of [6]. So for well-behaved streams, not only will these frequent items eventually show up in the sketch, but also their estimate counts will also be accurate. The original paper also proved that the counts for infrequent items in the tail are essentially the same and converge in distribution to a PPS sample. This is formally stated on Fact 5 and Fact 6. Building upon the established inclusion of all frequent items, the subsequent facts pertain exclusively to the leftover bins and their contained infrequent elements.

Fact 5. *If $p_1 < 1/m$ then $0 \leq t/m - \hat{N}_{min,a}(t) \leq m(logt)^2 + m$ and $0 \leq \hat{N}_{max,a}(t) - t/m \leq (logt)^2 + 1$ eventually.*

Fact 6. *If $p_1 < 1/m$, then the items in the sketch converge in distribution to a PPS sample*

3.4 AGMS

The AGMS sketch [1] is the first sketch-based algorithm for inner-product estimation. It can be used both on aggregated and disaggregated data. Let's call U the universe of possible items and h_j a family of 4-wise independent random hash function that maps U in $\{+1, -1\}$. Different hash functions are independent. The i-th entry of the size n $AGMS$ sketch of the N length streaming a is defined as the random variable $x_a[j] = \sum_{i=0}^{N-1} h_j(a_i)$.

As a new data item arrives, we need to update all the entries of the sketch. This operation is proportional to the size of the sketch. Given two streamings a and b, can be shown that $W[j] = x_a[j]x_b[j]$ is an unbiased estimator of the inner product of $\mathbf{n}_a(t)$ and $\mathbf{n}_b(t)$.

To decrease the variance, we compute the mean of all the n estimate $W[j]$, i.e, $Y = \frac{1}{n}\sum_{j=1}^{n} W[j]$.To make the estimator even more stable, the original author proposes to return Z, the median of m independent Y.

The following result is a theoretical guarantee that this algorithm works:

Fact 7. *Taking* $n = O(1/\epsilon^2)$ *and* $m = O(log(\frac{1}{\delta}))$*, then with probability at least* $1 - \delta$*,* $Z \in (\langle \boldsymbol{n}_a(t), \boldsymbol{n}_b(t)\rangle \pm \epsilon \left\| \boldsymbol{n}_a(t) \right\|_2 \left\| \boldsymbol{n}_b(t) \right\|_2)$*. The processing time required to maintain each sketch is* $O(1/\epsilon^2 log\frac{1}{\delta})$ *per update*

The main problem of this estimator is the cost. To decrease the error we have to increase the sketch size n. As the space and update-lime grow linearly with n, this cost makes the use of this algorithm in practice unfeasible.

3.5 Fast AGMS

To overcome this update cost of AGMS, Fast AGMS was proposed by [2]. The idea is to use a hash function that maps each item to a specific entry instead of updating all of them. More formally, Let's call U the universe of possible items and h a 4-wise independent random hash function that maps U in $\{+1, -1\}$. Besides, let' call φ a 2-universal hash function that maps U in $\{1 \dots n\}$. When a new data item i from a streaming a arrives, only the counter $\varphi(i)$ is updated with the value of $h(i)$, i.e, $x_a[\varphi(i)] \mathrel{+}= h(i)$.

If we have two streamings a and b and x_a, x_b are computed as described above with the same hash functions h and φ, then the inner product between the frequency vectors of the streamings, $\langle \mathbf{n}_a(t), \mathbf{n}_b(t)\rangle$, is estimated by $W = \sum_{i=1}^{n} x_a[i]x_b[i]$. If we compute m independent W and return the median Z, Fast AGMS has the same guarantees that AGMS, but with cost update of $O(log\frac{1}{\delta})$ instead of $O(\frac{1}{\epsilon^2} log\frac{1}{\delta})$.

For a fair comparison, we returned the W estimator in our experiments.

4 Contributions

Our contributions are:

1. Experimental analysis of Unbiased Space Saving for calculating the inner product between two streams.

2. Adaptation of Unbiased Space Saving to Coordinated Space Saving and analysis of the tradeoff between coordination and Unbiased behavior.

3. Comparative analysis among the various methods presented in this study.

5 Proposed Sketches

Given the analysis above, a natural question is whether Unbiased Space Saving can be used to calculate the inner product between two streams. This question is natural given that it has guarantees of preserving the most frequent items, which in a concentrated distribution are the ones that contribute the most to the inner product. To analyze that we proposed 2 distinct approaches. We call them by Independent Unbiased Space Saving (I-USS) and Coordinated Unbiased Space Saving (C-USS). Both approaches compute the USS sketch and use it to estimate the inner product, the main difference is how each one compute the probability of changing the label of the smallest bin.

5.1 I-USS

As shown in Algorithm 5 we basically compute the USS sketch. For each item that arrives on the stream, we pick a random number in $[0, 1]$. We assume that this operation in independent in each trial.

Algorithm 5 Independent Unbiased Space Saving

```
1: Initialize $S$ to be an empty list of (item, count) pairs.
2: for each new item $x$ do
3:     if $x$ is in $S$ then
4:         Increment the count of $x$ in $S$.
5:     else
6:         Find the pair $(x_{\min}, \hat{N}_{min,a}(t))$ in $S$ with the smallest count.
7:         Pick a random number $r$ uniformly distributed in $[0, 1]$
8:         Update $(x_{\min}, \hat{N}_{min,a}(t))$ to $(x, \hat{N}_{min,a}(t) + 1)$ if $r \leq 1/(\hat{N}_{min,a}(t) + 1)$.
9:     end if
10: end for
```

As $\mathbb{P}[r \leq 1/(\hat{N}_{min,a}(t) + 1)] = 1/(\hat{N}_{min,a}(t) + 1)$ and each trial is independent, all the guarantees of USS still hold. Once our sketches $\mathbf{N}_a(T_a)$ and $\mathbf{N}_b(T_b)$ are computed, to estimate the inner product we just compute the inner product between the sketches restricted to keys that appear on both. That is stated on Algorithm 6.

Algorithm 6 Inner Product Estimator

```
    Input: Sketches $\mathbf{N}_a(T_a)$ and $\mathbf{N}_b(T_b)$ constructed by Algorithm 5 or Algorithm 7
    Output Estimate $w$ of $\langle \mathbf{n}_a(T_a), \mathbf{n}_b(T_b) \rangle$
1: Compute $I = \mathbf{N}_a.keys() \cap \mathbf{N}_b.keys()$
2: return $\sum_{i \in I} \hat{N}_a(i, t)\hat{N}_b(i, t)$
```

We can show that this algorithm provides an unbiased estimator of the inner product between the vector of frequencies of the streamings.

Theorem 1. $\mathbb{E}[\langle \boldsymbol{N}_a(T_a), \boldsymbol{N}_b(T_b) \rangle] = \langle \boldsymbol{n}_a(T_a), \boldsymbol{n}_b(T_b) \rangle$

Proof. We have that $\langle \mathbf{N}_a(T_a), \mathbf{N}_b(T_b) \rangle = \sum_{i\ item} \hat{N}_a(i, T_a)\hat{N}_b(i, T_b)$. So, by linearity of expectation it follows that:

$$\mathbb{E}[\mathbf{N}_a(T_a), \mathbf{N}_b(T_b)\rangle] = \sum_{i \text{ item}} \mathbb{E}[\hat{N}_a(i, T_a)\hat{N}_b(i, T_b)]$$

But we know that $\hat{N}_a(i, T_a)$ and $\hat{N}_b(i, T_b)$ are independent random variables by construction, so it follows that

$$\begin{aligned}
\mathbb{E}[\mathbf{N}_a(T_a), \mathbf{N}_b(T_b)\rangle] &= \sum_{i \text{ item}} \mathbb{E}[\hat{N}_a(i, T_a)\hat{N}_b(i, T_b)] \\
&= \sum_{i \text{ item}} \mathbb{E}[\hat{N}_a(i, T_a)]\,\mathbb{E}[\hat{N}_b(i, T_b)] \\
&= \sum_{i \text{ item}} n_a(i, T_a)n_b(i, T_b) \\
&= \langle \mathbf{n}_a(T_a), \mathbf{n}_b(T_b)\rangle
\end{aligned}$$

Where the second equality holds by independence and the third one follows by the Fact 3. $\square$

However, even being correct in expectation, this method doesn't ensure coordination, which is essential for the inner product. We can separate the inner product into 3 types of products:

1. frequent item $\times$ frequent item
2. frequent item $\times$ infrequent item
3. infrequent item $\times$ infrequent item

Drawing upon the discussion in section 3.3, we can reasonably anticipate that all terms of type 1 will eventually emerge within our estimate. These terms hold the most representative value for the inner product, as they yield the highest product value. However, terms of type 2 can also be significant, and we aim to retain them as well. A clear example of this can be observed in Figure 1, where the data has been aggregated solely for the purpose of simplifying the visual representation.

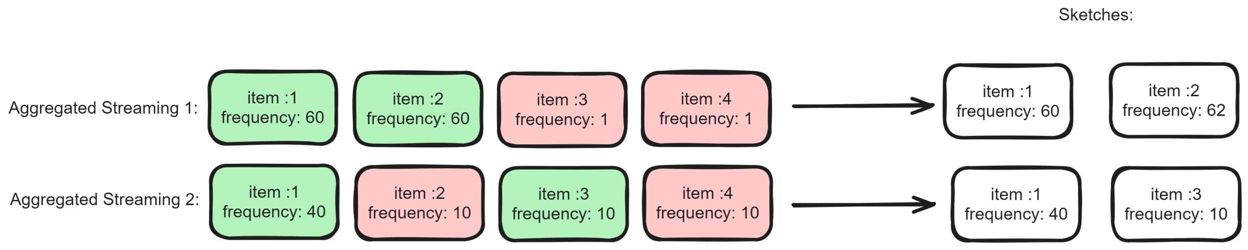


Figure 1: An example of data where I-USS potentially fails to capture representative products of the inner product. In this particular case, it overlooked item 2 in the second sketch, as there were no differentiating factors making item 2 more probable than items 3 or 4. This omission results in a significant error in the final inner product calculation.

To tackle this limitation, we propose the Coordinated Unbiased Space Saving (C-USS)

5.2 C-USS

Let's call U the universe of possible items on the stream The difference between I-USS and C-USS is that now we will use a uniform hash function $h : U \to [0, 1]$ to pick a number between 0 and 1 and use this value to decide if we will change the label of the smallest bin. It helps with the coordination since we will use the same hash function on all streamings, forcing us to have the same result for the same keys. This procedure can be seen on Algorithm 7

Algorithm 7 Coordinated Unbiased Space Saving

```
   Assume access to an uniform random hash function $h : U \to [0, 1]$
1: Initialize $S$ to be an empty list of (item, count) pairs.
2: for each new item $x$ do
3:     if $x$ is in $S$ then
4:         Increment the count of $x$ in $S$.
5:     else
6:         Find the pair $(x_{\min}, \hat{N}_{min,a}(t))$ in $S$ with the smallest count.
7:         Update $(x_{\min}, \hat{N}_{min,a}(t))$ to $(x, \hat{N}_{min,a}(t) + 1)$ if $h(x) \leq 1/(\hat{N}_{min,a}(t) + 1)$.
8:     end if
9: end for
```

As $\mathbb{P}[h(x) \leq 1/(\hat{N}_{min,a}(t)+1)] = 1/(\hat{N}_{min,a}(t)+1)$ for each x, all the guarantees off USS still hold, however we lost the independence between $\hat{N}_a(i,t)$ and $\hat{N}_b(i,t)$. To see that, imagine that x is the next item on both streamings a and b and x is the smallest bin on both. Let's call A and B the events where x is choose to be stored on the first and second sketch respectively. We have that:

$$\begin{aligned} \mathbb{P}[A \cap B] &= \mathbb{P}[h(x) \leq 1/\hat{N}_{min,a}(t) \cap h(x) \leq 1/\hat{N}_{min,b}(t)] \\ &= \mathbb{P}[h(x) \leq min(\hat{N}_{min,a}(t), \hat{N}_{min,b}(t))] \\ &\neq \mathbb{P}[A]\,\mathbb{P}[B] \end{aligned}$$

Due to the same reason, we can't ensure that it will be unbiased for estimating the inner product, but we still maintain the name "Unbiased" on the algorithm as it still has the same properties as USS, as the unbiased behavior for the subset sum estimation problem.

6 Experiments

In this section, we will present the experimental results of inner product estimation problem for the proposed algorithms *I-USS* and *USS-Coordinated* compared to other algorithms such as *Threshold Sampling*, *Priority Sampling*, and *Fast-AGMS*. The selected datasets are all synthetic and will be introduced in Section 6.1. We performed 100 iterations for each test and calculated the average relative error. We conducted tests by varying the sketch size among 100, 200, and 300, as well as some parameters associated with each dataset. We demonstrate that, for more concentrated distributions, *USS-Coordinated* performs even better than *Threshold Sampling* and *Priority Sampling*, which require pre-aggregation of data.

6.1 Datasets

All the data we used are synthetic. For each test, we utilized two streams, each containing 40 thousand items. And since we're addressing the Inner Product Estimation problem between two streams, we consistently generate data from the same type of distribution for both streams. The only exception is for the shifted case, where we estimate the inner product between one streaming and a shifted version of the same streaming.

1. **Zipfian**: We generated synthetic data following a Zipf distribution with parameters 1.1, 2, 3, and 4. Each dataset contains 40,000 items in random orders.
2. **Weibull**: We utilized the Weibull distribution that is discretized to integer values, where we varied the scale parameter between 0.1 and 1 and generated distributions more heavy tailed.
3. **Truncated Weibull** This is a similar distribution to previous one, except for the fact that the truncated weibull has limits for the numbers generated. We set this limit to [1, 1000] for all truncated weibull distributions we used.
4. **Geometric**: We generated synthetic data from a geometric distribution with parameter 0.03, which is a long-tailed distribution.

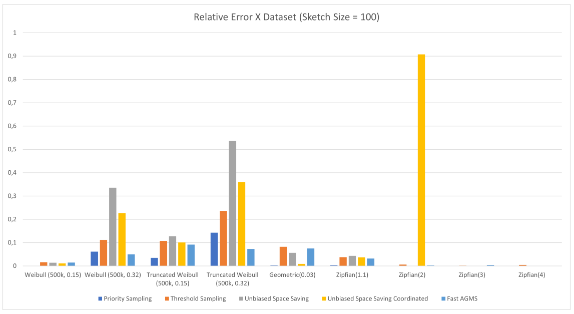
Furthermore, we set a Weibull distribution and sorted the streaming elements, simulating data reception in "clusters." Additionally, we generated shifted data based on the original distribution. For testing purposes, we varied the shift parameter between 1 and 19. These scenarios reflect real situations in database systems and are, therefore, crucial to test [7].

Finally, to compare the Priority Sampling and Threshold Sampling methods (for aggregated data) with I-USS, C-USS, and Fast-AGMS, we utilized the same data. However, before passing the data to the aggregated data methods, we aggregated the data.

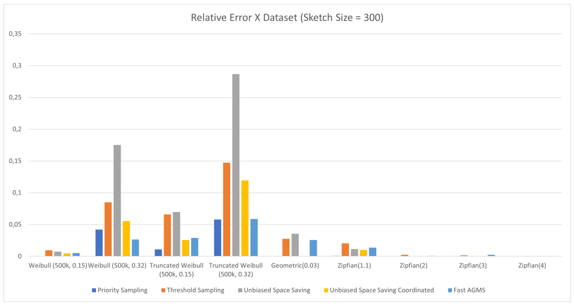
6.2 Results

6.2.1 General Testings

As we can observe in Figure 2, all algorithms return a higher average relative error when they receive less concentrated streams. This becomes evident when comparing the datasets *Weibull(500k, 0.15)* and *Weibull(500k, 0.32)*. Additionally, a significant difference in the average relative error can be seen when we increase the sketch size from 100 (a) to 300 (a). The algorithm most impacted by this change was USS-Coordinated, which was the worst method for the Zipfian dataset with a sketch size of 100 but became one of the best when we increased the sketch size.



(a) General testing for multiple data streaming. Sketch size = 100



(b) General testing for multiple data streaming. Sketch size = 300

Figure 2: Average Relative Error (ARE) for multiple data distributions

	P.S	T.S	USS	USS-C	F-AGMS
Weibull (500k, 0.15)	0,0008	0,0121	0,0077	0,0064	0,0105
Weibull (500k, 0.32)	0,0481	0,0932	0,1817	0,0641	0,0408
Truncated Weibull (500k, 0.15)	0,0178	0,0801	0,0836	0,0419	0,0280
Truncated Weibull (500k, 0.32)	0,0860	0,1767	0,3435	0,2043	0,0636
Geometric(0.03)	0,0000	0,0423	0,0357	0,0011	0,0359
Zipfian(1.1)	0,0014	0,0257	0,0271	0,0176	0,0164
Zipfian(2)	0,0000	0,0044	0,0000	0,5438	0,0014
Zipfian(3)	0,0000	0,0018	0,0000	0,0000	0,0051
Zipfian(4)	0,0000	0,0019	0,0000	0,0000	0,0000

Table 2: Heatmap of General Testing Results (sketch size = 200)

Lenegds: *P.S: Weighted Priority Sampling, T.S: Weighted Threshold Sampling, USS: Independent Unbiased Space Saving, USS-C: Coordinated Unbiased Space Saving, F-AGMS: Fast AGMS*

Moreover, in the heatmap table shown in Table 2, it's evident that Priority Sampling is the method with the best results for inner product estimation. However, it's worth noting that this method requires prior data aggregation. Concerning USS, USS-C, and F-AGMS, they generally exhibit similar performances. This stands as an advantage for the USS-C algorithm, as it not only estimates the inner product between two streams but also can be employed for subset sum estimation and frequent item estimation problems with the same guarantees as proven by Ting [6].

6.2.2 Varying Weibull Parameter

The Weibull distribution takes two parameters, the shape parameter, and the scale parameter. In this experiment, we examined the behavior of each method while varying the scale parameter between 0 and 1. What we observed in Figure 3 is that the USS-Coordinated algorithm performs better than the Threshold Sampling and Priority Sampling algorithms for less concentrated distributions, despite these methods having the 'advantage' of working with pre-aggregated data. However, the USS-Coordinated method has a higher average relative error than the Fast-AGMS for all distributions.

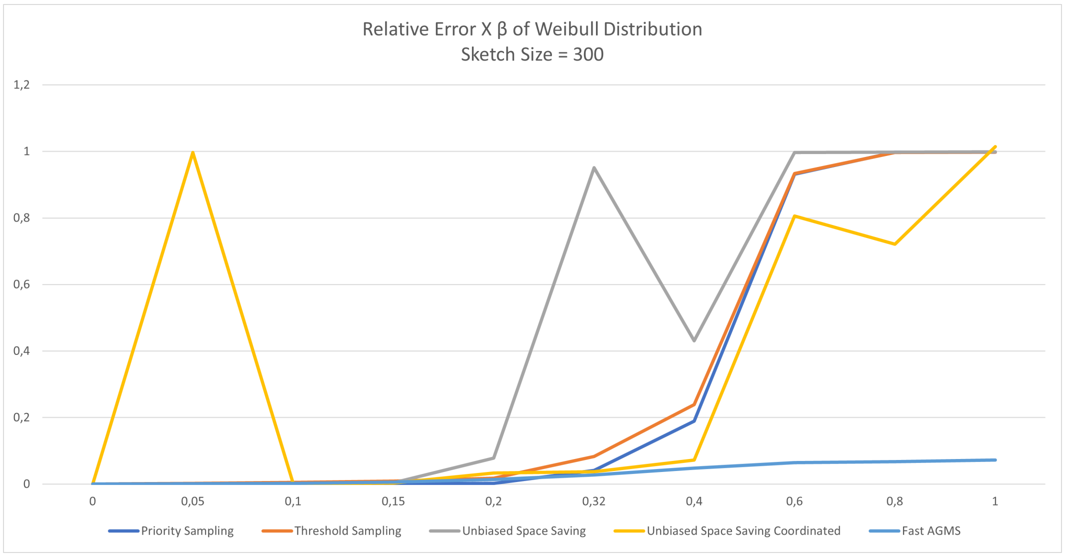


Figure 3: Average Relative Error (ARE) for Weibull distribution with shape parameter = 500k and scale parameter varying between 0 and 1

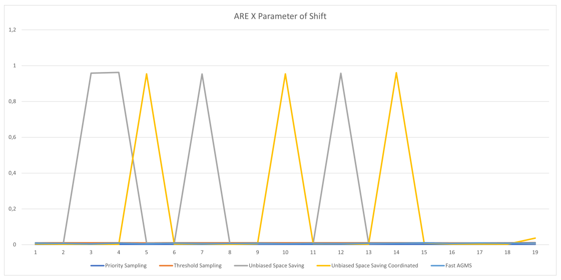
6.2.3 Shifted Streamings

To perform this test, we generated the Weibull(500k, 0.15) stream and generated the second stream based on the first one. We initially calculated the frequencies of each item in the first stream and sorted it using the frequency of each item as a key. Then, we applied a shift of k. For instance, if $k = 1$, the second most frequent item is replaced by the most frequent item, the third most frequent item is replaced by the second most frequent item, and so forth. Consequently, the first item (the item with the highest frequency) is replaced by the last item, which is the least frequent item.

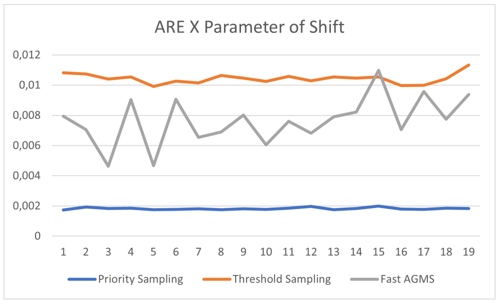
To illustrate, consider a list of items with frequencies: [a: 2334, b: 1234, c: 123, d: 12]. Now, let's assume we want to perform a shift with $k = 1$. In this case, the elements would be replaced as follows: $\{b : a, c : b, d : c, a : d\}$. So, if the original stream was: [a, b, a, a, b, c, d, a, b, d, c, ...], the new stream would be: [d, a, d, d, a, b, c, d, a, c, b, ...]

We perform this test for $1 \leq k \leq 19$ and use the original streaming and the shifted streaming to estimate the inner product.

As observed in the graphs in Figure 4, our proposed algorithms did not perform well with the shifted streams. Despite occasionally displaying very low relative errors, there was inconsistency in the results. Conversely, Priority Sampling, Threshold Sampling, and Fast AGMS yielded highly favorable results, as depicted in the figure (b) below. Fast AGMS, in particular, emerged as the most robust algorithm in handling shifted data.



(a) All Algorithms Comparison

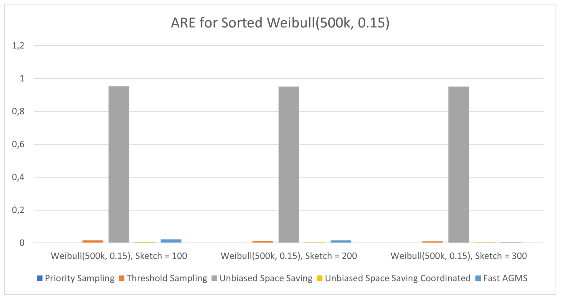


(b) Priority Sampling, Threshold Sampling and Fast AGMS

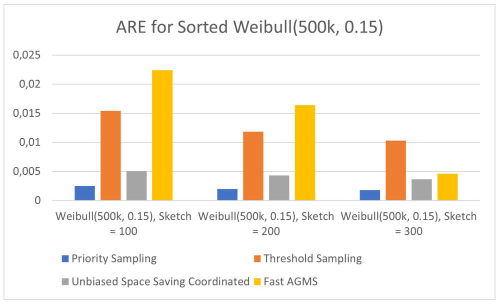
Figure 4: Average Relative Error (ARE) for Weibull distribution with shape parameter = 500000 and scale parameter = 0.15 when shift parameter varies from 1 to 19.

6.2.4 Sorted Streamings

In this experiment, we generated two streams of synthetic data following the *Weibull(500k, 0.15)* distribution and sorted them, causing the data to arrive in clustered groups. The left graph in Figure 5 illustrates the outcomes of all algorithms. However, the average relative error of I-USS was significantly higher than the others. Consequently, in the right graph, we excluded I-USS to compare only the remaining 4 algorithms. As a result, we observed that upon increasing the sketch size from 100 to 300, the USS-Coordinated method exhibited a higher average relative error than only the Priority Sampling.



(a) All Algorithms Comparison



(b) Ignoring I-USS

Figure 5: Average Relative Error (ARE) for Weibull distribution with shape parameter = 500000 and scale parameter = 0.15

7 Conclusion

In this work, we introduced two adaptations to the algorithm proposed by Ting [6]: the Independent Unbiased Space Saving (I-USS) and the Coordinated Unbiased Space Saving (C-USS). Both adaptations can handle Subset Sum Estimation and the Frequent Items Problem in an unbiased manner. While the former can perform unbiased inner product estimation for disaggregated data, the latter presents a tradeoff between coordination (improving inner product estimation) and unbiasedness. We aimed for greater coordination to enhance inner product estimation results; however, we couldn't prove the algorithm's maintenance of unbiasedness in this scenario.

Our experimental results demonstrate that C-USS outperformed I-USS, emphasizing the importance of coordination in inner product estimation. Furthermore, C-USS results were comparable to Fast-AGMS in some datasets.

Considering that C-USS addresses three problems with disaggregated data, while Fast-AGMS handles only one and Inner Product Estimation based on Priority Sampling requires aggregated data, we emphasize C-USS' significance.

Our code is available in: https://github.com/suemyinagaki/NYU-AMLDS-final-project

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 20–29, New York, NY, USA, 1996. Association for Computing Machinery.

[2] Graham Cormode and Minos Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Proceedings of the 31st international conference on Very large data bases*, pages 13–24, 2005.

[3] Majid Daliri, Juliana Freire, Christopher Musco, Aécio Santos, and Haoxiang Zhang. Sampling methods for inner product sketching, 2023.

[4] Nick Duffield, Carsten Lund, and Mikkel Thorup. Learn more, sample less: control of volume and variance in network measurement. *IEEE Transactions on Information Theory*, 51(5):1756–1775, 2005.

[5] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory*, ICDT'05, page 398–412, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] Daniel Ting. Data sketches for disaggregated subset sum and frequent item estimation. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1129–1140, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Feiyu Wang, Qizhi Chen, Yuanpeng Li, Tong Yang, Yaofeng Tu, Lian Yu, and Bin Cui. Joinsketch: A sketch algorithm for accurate and unbiased inner-product estimation. *Proc. ACM Manag. Data*, 1(1), may 2023.