TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

# Distributed Sliding-Window Matrix Sketching

*Author:*
Eirini ASTERI

*Supervisor:*
Prof. Minos GAROFALAKIS

*A thesis submitted in fulfillment of the requirements*
*for the degree of Diploma in*
Electrical and Computer Engineering

August 5, 2016

# *Abstract*

Streaming sketching algorithms are data-processing algorithms for the summarization of an input data stream under memory and computational constraints. Their input is a long or potentially unbounded sequence of items that can be parsed a single (or a limited number of times), and the objective is to construct a concise summary of the data –a sketch– which can be later used to approximate a quantity of interest. In this work, we focus on streaming matrix sketching methods: the input is a sequence of vectors which can be regarded as the rows of a large matrix. We briefly survey matrix sketching methods for generating various kinds of sketches. We will mostly focus on the problem of approximating the principal subspace of a large matrix under the streaming model and we will describe the state-of-the-art 'Frequent Directions' method of Liberty, and extensions to the distributed setting. We will further review very recent extensions of this work to monitoring the principal subspace of a stream over a sliding-time window. Here, the objective is to maintain a sketch that approximates the desired quantity for the most recent segment of the input. We will conclude with a novel result on the distributed construction of sketches for the sliding window model and future directions.

# Contents

# Chapter 1

# Introduction

Data Summarization is a key data mining concept which involves techniques for finding a compact description of a large dataset. Large datasets are often represented by large matrices. Such datasets are found in large image analysis, where each row of a matrix may correspond to one image and contain either raw pixels or derived feaure values, in information retrieval, where textual data in the bag-of-words model is also represented by a matrix whose rows correspond to documents and in many others large scale machine learning applications where matrices contain feature vectors. While matrix approximation and decomposition has been studied in numerical linear algebra for many decades, these methods often require more space and time than is feasible for very large scale settings, and also often worry about more precision than is required.

The last decade has witnessed an explosion of work in *matrix sketching*, where a large matrix $\mathbf{A}$ is approximated with a concisely representable matrix $\mathbf{B}$, referred to as *sketch*, which preserves most of the properties of $\mathbf{A}$. This small space constraint is critical when the full data set cannot fit in memory or disk. Much of the expensive computation can then be performed on the smaller matrix, thereby accelerating the solution for the original problem. Although there is a significant load of work in matrix sketching in recent years, we can categorize this work in three basic approaches, *random projection*, *column subset selection* or *row sampling* and *hashing*. We will briefly refer to these methods in the next section. Their comparison is not straightforward since their objectives vary.

The task of approximating a matrix $\mathbf{A}$ becomes much more challenging when it comes to the *streaming model* case. The streaming model, i.e. the computational model in which only one pass of data is required, is very attractive since the sketch can be constructed immediately upon the arrival of data. This work highlights the recent advances in algorithms for matrix sketching on either *centralized* or *distributed data*. Initially, we will focus on Edo Liberty's Frequent Directions streaming method presented in [3], which deals with the problem of approximating the principal subspace of a large matrix under the streaming model. Using this as a building block we will later consider the case of distributed model in which the data is distributed over a number of sites and we aim to approximate a summary of these at the coordinator. In this context we will revise the work of Mina Ghasami et al. [5] that deals with the distributed case employing Frequent Directions sketches.

In the following, we will present a recent work [6] concerning the problem of maintaining sketches over a stream *sliding window* using limited space. The authors present a framework that converts streaming *mergeable* *sketches*, such as Frequent Directions [3], to sliding window sketches. We will conclude with a novel result on the distributed construction of sketches for the sliding window model.

# Chapter 2

# Matrix Sketching

## 2.1 Matrix Sketching and the Stream Data Model

Many data mining algorithms assume that we are mining a database. That is, that we can have access to all our data whenever we want it. The problem of mining data becomes more challenging in the case where data arrives in a stream or streams, i.e., an unbounded sequence of data items, and if it not processed immediately or stored, then it is lost forever. Moreover, we shall assume that the data arrives so rapidly it is not feasible to store it all in a database, and then interact with them at the time of query. A very common approach to answer queries on streams is to maintain a summary of the stream. This summary may concern all data seen so far, or as we will see in Chapter 4, a *sliding window* of the stream. A sliding window can be the most recent $n$ elements of a stream, or it can be all the elements that arrived within the last $t$ time units, e.g., one day.

Through out this work we will consider the *row update streaming model*, where the stream is an ordered sequence of time-stamped rows, i.e., the whole stream can be represented by a dynamic large matrix with a constantly growing number of rows. In this work, we will revise several methods and models that aim to built a small summary of this large streaming matrix, i.e., deal with the matrix sketching problem under the streaming model assumption.

## 2.2 Matrix Sketching Approaches

**Random Projection** Random Projection method is widely used for low-rank approximation [[14]],[[15]],[[16]]. Random Projection method projects data points from a high dimensional space to a much lower dimensional space by using a random matrix $\mathbf{S} \in \mathbb{R}^{l \times d}$, such that $\mathbf{B}_{\ell \times n} = \mathbf{S}_{\ell \times n} \mathbf{A}_{n \times d}$. The idea of random mapping comes from Johnson-Lindenstrauss (JL) lemma [13]. Intuitively, the JL lemma says that if you choose a random subspace of the original space and project the points onto it, the pairwise distances between points will probably be preserved. Random Projection method can be easily computed in a streaming fashion requiring at most $\mathcal{O}(\ell d)$ space and $\mathcal{O}(\ell d)$ operations per row update.

**Row Sampling or Column Subset Selection** Sampling methods select a subset of important rows (or columns) of the original matrix, often randomly with respect to a well-defined probability distribution. Row sampling methods vary in the way they define notion of importance. A simple streaming solution to the problem is obtained by sampling rows with probability proportional to their $l_2$ norm requiring $\mathcal{O}(\ell d)$ space and $\mathcal{O}(\ell)$ update running time.

## 2.3 Frequent Directions

In [3] Edo Liberty discovers the strong relationship between matrix sketching and frequent items. Based on the well known *frequent items* algorithm of Misra and Gries [1], he proposes a simple and deterministic streaming method for approximating a large matrix $\mathbf{A}$ with a concisely representable matrix $\mathbf{B}$ such that

$$\mathbf{B}^T\mathbf{B} \approx \mathbf{A}^T\mathbf{A}.$$

For intuition purposes we will shortly describe the frequent items problem, as well as the simple and deterministic algorithm proposed by Misra and Gries [1].

### 2.3.1 Frequent Items

Consider a set of $d$ distinct items and a stream of $n$ item appearances $a_1, \ldots a_n$. Clearly, each distinct item may appear more than once in the stream. The problem of finding those items with frequency higher than a fraction $\epsilon$ of the total count $n$ is known as Frequent Items Problem. A naive way to solve this problem is by maintaining $d$ counters, one for each distinct item, but this would lead to $\Theta(n)$ space. The Frequent Items Problem dates back to 1980s and has received a very simple and elegant solution by Misra and Gries.

Misra and Gries (MG) algorithm instead of searching for the exact item frequencies $f_j$, uses $\mathcal{O}(\ell)$ space with $\ell \ll n$ and produces approximate frequencies $\hat{f}_j$, such that $|f_j - \hat{f}_j| \leq n/\ell$. Thus, instead of keeping $n$ counters, MG algorithm keeps $\ell$ counters. The algorithm relies on a simple process of repeatedly "deleting" $\ell$ distinct items of a stream of $n$ items, until less than $\ell$ distinct items remain.

So, let us consider that we have a bag of $n$ items drawn from a bounded set of $d$ distinct items. The operation of deleting $\ell$ distinct elements can be performed at most $n/\ell$ times, since at the $n/\ell$th round there will remain $n \mathbf{mod} l < l$ items. Lets assume that the process terminates after the $t$th round, i.e., after the $t$ th round $l$ distinc items remain. At the end of this process, each item has been deleted at most $t$ times, since in each round we delete distinct items. Therefore, $|f_j - \hat{f}_j| \leq t \leq n/\ell, \forall j = 1, \ldots, d$. Clearly, each item that does not appear in the final trimmed stream cannot have original frequency greater than $n/\ell$.

More precisely, the algorithm works as follows. The algorithm is initialized with $\ell$ zero valued counters. Upon the first arrival of an item $j$, one of the counters is allocated to the item $j$ and is set to one. Upon each new arrival of an item $i$, three cases can occur. If there is already a counter allocated to item $i$ then its count is increased by one. Else, if there is some zero valued counter, then it is allocated to the new item $i$ and is set to one. Otherwise, if no empty counter exists, all counters are decremented by one and the new item is discarded.

Recall that Frequent Items problem's goal is to calculate some approximate frequencies $\hat{f}_j$ of $f_j$, such that

$$\forall j \in [d] \quad |f_j - \hat{f}_j| \leq \epsilon n \tag{2.1}$$

where $\epsilon \in [0, 1]$, $n$ is the total count of items and $[d] = \{1, \ldots, d\}$.

Liberty's work relies on the similarity of matrix sketching with the frequent items problem. His matrix sketching method proposed in [3] is a modified version of MG algorithm, where items are substituted with singular vector, the weights of items with singular values and the "deletion" step with a "shrinking" step utilizing singular value decomposition (svd).

The matrix approximation problem studied by Liberty receives $n$ rows of a large matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and maintains a smaller matrix $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ with $\ell \ll n$, such that

$$\forall \mathbf{x} \in \mathbb{R}^d, \|\mathbf{x}\| = 1 \quad \big|\|\mathbf{A}\mathbf{x}\|^2 - \|\mathbf{B}\mathbf{x}\|^2\big| \leq \underbrace{2\|\mathbf{A}\|_{\mathrm{F}}^2/\ell}_{\epsilon\|\mathbf{A}\|_{\mathrm{F}}^2}. \tag{2.2}$$

To illustrate the similarity between the matrix sketching and the frequency approximation problem let us consider the restricted case where the rows of $\mathbf{A}_{n\times d}$ are indicated vectors, i.e., each row $\mathbf{a}_i \in \{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$, where $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$. If we consider matrix $\mathbf{A}$ as a stream of rows, then each row - item is a $d$-dimensional vector drawn from a bounded domain of $d$ vectors $\mathbf{e}_1, \ldots, \mathbf{e}_d$. Let $f_j$ denote the frequency, i.e. number of appearances, of item $\mathbf{e}_j$ in the matrix $\mathbf{A}$. Then, $f_j$ can be easily calculated by $f_j = \|\mathbf{A}e_j^\top\|^2$. A good approximation matrix $\mathbf{B}$ can be obtained by finding a matrix that guarantees that $\hat{f}_j = \|\mathbf{B}\mathbf{e}_j^\top\|^2$ is a good approximation of $f_j$. Recall that $\mathbf{A}$ has $n$ rows and each row is a unit vector and, therefore $\|\mathbf{A}\|_{\mathrm{F}}^2 = 1$. From the above, $|f_j - \hat{f}_j| \leq \epsilon n$ is equivalent to $\big|\|\mathbf{A}\mathbf{x}\|^2 - \|\mathbf{B}\mathbf{x}\|^2\big| \leq \epsilon\|\mathbf{A}\|_{\mathrm{F}}^2$.

### 2.3.2 Frequent Directions

#### Intuition

The intuition of Frequent Directions(FD) method is surprisingly similar to the idea behind Frequent Items(FI) method. FI method periodically decreases $\ell$ different counters by the same amount. Similarly, FD method "shrinks" $\ell$ orthogonal vectors by the same amount. Thereby, just as FI method, that uses $\ell \geq 1/\epsilon$ counters, reveals each item with frequency higher than $\epsilon n$, FD method uncovers each unit vector, i.e. direction, $\mathbf{x}$ for which $\|\mathbf{A}\mathbf{x}\|^2 \geq \epsilon\|\mathbf{A}\|_2^2$.

#### Algorithm

The algorithm begins by initializing a zero valued matrix $\mathbf{B}_{\ell\times d}$. Input matrix $\mathbf{A}$ is received in a streaming fashion, i.e. one row after the other. Upon each arrival, the new row is inserted to a zero valued row of matrix $\mathbf{B}$. If no such a row exists, then the algorithm proceeds by running a "shrinking" process. During the "shrinking" process, we initially calculate the singular value decomposition of matrix $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{\Sigma} = diag([\sigma_1, \sigma_2, \ldots, \sigma_l])$

**Algorithm 1** Frequent Directions [3]

```
input : $\ell \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{n \times d}$
output $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
1: $\mathbf{B} \leftarrow \mathbf{0}_{\ell \times d}$    { Initialize all-zero sketch. }
2: for $i = 1, \dots, n$ do
3:     Copy $\mathbf{A}_i$ ($i$th row of $\mathbf{A}$) into a zero row of $\mathbf{B}$.
4:     if $\mathbf{B}$ nas no zero valued row then
5:         $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \mathsf{SVD}(\mathbf{B})$    {Compute SVD; diag($\mathbf{\Sigma}$) contains the singular values in descending order.}
6:         $\delta \leftarrow \Sigma^2_{\ell/2,\ell/2}$
7:         $\widehat{\mathbf{\Sigma}} \leftarrow \sqrt{\max(\mathbf{\Sigma}^2 - \delta \mathbf{I}_\ell)}$
8:         $\mathbf{B} \leftarrow \widehat{\mathbf{\Sigma}} \mathbf{V}^\top$    {Upbdate $\mathbf{B}$. Half the rows are zero valued.}
9:     end if
10: end for
```

with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_l \geq 0$. Considering the singular vectors $\mathbf{v}_i$ as items and the corresponding singular values $\sigma_i$ as their weights we proceed by decreasing each weight, i.e. singular value, by roughly the same amount, just as FI method decreases all counters by the same amount. After this step at least half of the rows of sketch are all zero. The process is repeated for each new arrival.

### Mergeability

One of the most useful properties of the FD sketch is *mergeability* (see Def. 1).

**Def. 1** (Mergeable Matrix Sketch). *Let $\kappa$ be a matrix sketching algorithm that given a matrix $\mathbf{A}$ and a parameter $\epsilon$ produces a matrix $\mathbf{B} \leftarrow \kappa(\mathbf{A}, \epsilon)$ such that size($\mathbf{B}$) $\leq \ell$ and err($\mathbf{A}, \mathbf{B}$) $\leq \epsilon$; here, size($\mathbf{B}$) represents an arbitrarily defined measure of the size of $\mathbf{B}$, and err($\mathbf{A}, \mathbf{B}$) represents an arbitrary function that measures the error of using $\mathbf{B}$ as a surrogate of $\mathbf{A}$. The sketching algorithm $\kappa$ is* mergeable *iff there exists a* merging *algorithm $\mathcal{A}_{\mathsf{MERGE}}(\cdot,\cdot)$ such that: for any $n_1 \times d$ matrix $\mathbf{A}_1$ and $n_2 \times d$ matrix $\mathbf{A}_2$ with corresponding sketches $\mathbf{B}_1 \leftarrow \kappa(\mathbf{A}_1, \epsilon)$ and $\mathbf{B}_2 \leftarrow \kappa(\mathbf{A}_2, \epsilon)$, the merging algorithm produces a matrix $\mathbf{B} \leftarrow \mathcal{A}_{\mathsf{MERGE}}(\mathbf{B}_1, \mathbf{B}_2)$ such that size($\mathbf{B}$) $\leq \ell$ and err($\mathbf{A}, \mathbf{B}$) $\leq \epsilon$, where $\mathbf{A}$ is the $(n_1 + n_2) \times d$ matrix formed by vertically stacking the rows of $\mathbf{A}_1$ and $\mathbf{A}_2$.*

**Lemma 2.3.1** ([3]). *The sketch produced by Frequent Directions (Alg. 1) is mergeable. Spefically, let $\mathbf{A} = [\mathbf{A}_1^\top, \mathbf{A}_2^\top]^\top$ be any $n \times d$ with rows arbitrarily partitioned into two submatrices $\mathbf{A}_1$ and $\mathbf{A}_2$ and let $\mathbf{B}_1$ and $\mathbf{B}_2$ be the FD sketches (computed by Alg. 1) on $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively, with parameter $\ell$. By the guarantees of FD, $\|\mathbf{A}_1^\top \mathbf{A}_1 - \mathbf{B}_1^\top \mathbf{B}_1\| \leq \frac{2}{\ell}\|\mathbf{A}_1\|_\mathrm{F}^2$ and $\|\mathbf{A}_2^\top \mathbf{A}_2 - \mathbf{B}_2^\top \mathbf{B}_2\| \leq \frac{2}{\ell}\|\mathbf{A}_2\|_\mathrm{F}^2$. Let $\mathbf{C}$ be the FD sketch of $\mathbf{B} = [\mathbf{B}_1^\top, \mathbf{B}_2^\top]^\top$. Then,*

$$\|\mathbf{A}^\top \mathbf{A} - \mathbf{C}^\top \mathbf{C}\| \leq \frac{2}{\ell}\|\mathbf{A}\|_\mathrm{F}^2.$$

*Proof.* The proof is given in Section 2.2 of [3]. □

Given mergeability, parallelization is trivial. A matrix can be partitioned to disjoint segments across many machines. In this setting, each machine can independently compute a sketch of the local matrix; these sketches can afterwards be combined in an arbitrary order to produce a sketch of the total matrix.

### Space, Accuracy and Running Time

The algorithm uses $\mathcal{O}(\ell d)$ space and its error $\epsilon$ decays proportionally to $1/\ell$. Other sketching approaches like random projection, row sampling and hashing produce convergence rates proportional to $1/\ell$.

The singular value decomposition of an $\ell$ by $d$ matrix requires $\mathcal{O}(d\ell^2)$. The worst case update time of FD method comes from the SVD computation and it is therefore $\mathcal{O}(d\ell^2)$. After the SVD computation half of the sketches rows are nullified, that means that the SVD computation takes place every at least $\ell/2$ rows. When the algorithm does not compute SVD, the insertion of row in the sketch takes $\mathcal{O}(d)$ time. Therefore, the algorithm has an amortized update time of $\mathcal{O}(d\ell)$. Frequent Directions method outperforms its competitors in accuracy, but not in running time. For further analysis the reader is referred to [3].

# Chapter 3

# Matrix Sketching on Distributed Data

In the majority of data-management tasks today, data is collected over a wide area, and at a constantly accelerating pace. The physical distribution of data among multiple remote sites in conjunction with the communication constraints of the underlying network makes aggregation of data in a single site and centralized processing not a viable option. The need for processing queries over collections of remote data streams made *distributed streaming model* [10] popular in recent years. The problem of tracking and approximating streaming data matrices becomes more challenging when data are distributed across multiple remoted sites. In this context we review the recent work of Ghashami et al. [5] who consider the problem of *continuous matrix approximation* in the distributed streaming model.

## 3.1 Continuous Matrix Approximation

Ghashami et al. consider the problem where each site observes a disjoint subset of rows of large matrix $\mathbf{A}$ in a streaming fashion, i.e. row after row, and all sites must collaborate to continuously maintain a smaller matrix $\mathbf{B}$ that approximates $\mathbf{A}$ (within specified error bounds).

### 3.1.1 Notation and Preliminaries

In the distributed model studied we consider $m$ distributed sites $\{S_1, \ldots, S_m\}$ and a coordinator site $C$. Each site has a two-way communication with the coordinator. Let $t_c$ denote the current time and $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{R}^{n\times d}$ the stream of rows seen by the union of all $m$ sites so far. We assume *row-wise partition* of data, which means that at each time step any item $\mathbf{a}_j \in \mathbb{R}^d$ appears at exactly one of the sites; and therefore we can denote the set of items site $S_i$ processes as stream $\mathbf{A}_i \subset \mathbf{A}$, where $\bigcap_{i=1}^m \mathbf{A}_i = \varnothing$ and $\bigcup_{i=1}^m \mathbf{A}_i = \mathbf{A}$. Each row item is associated with a weight; we define the weight to be the squared $l_2$ norm of the row, i.e. $w_n = ||\mathbf{a}_n||^2$. We assume that the squared norm of every row is bounded by a value $\beta$.

The objective is to maintain a much smaller matrix $\mathbf{B} \in \mathbb{R}^{l\times d}$ as an approximation to $\mathbf{A} \in \mathbb{R}^{n\times d}$ such that for any unit vector $\mathbf{x} \in \mathbb{R}^{d\times 1}$ we can ensure that,

$$\big|||\mathbf{A}\mathbf{x}||^2 - ||\mathbf{B}\mathbf{x}||^2\big| \leq \epsilon||\mathbf{A}||_F^2.$$

The above expression is equivalent to

$$||\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}||^2 \leq \epsilon||\mathbf{A}||_F^2.$$

Their goal is to succeed such an approximation and at the same time minimize the total communication between $C$ and all sites $S_1, \ldots, S_m$. To achieve this, they exploit the mergeability of Frequent Direction [3] sketches by maintaining on each site $S_i$, a sketch $\mathbf{B}_i$ as an approximation to local stream $\mathbf{A}_i$, and occasionally send it to $C$. Due to the mergeability of FD sketch, $C$ can maintain a sketch $\mathbf{B}$ and merge each received sketch $\mathbf{B}_i$ to $\mathbf{B}$ without increasing the approximation error. Based on this idea the authors design two different distributed matrix tracking protocols utilizing the FD streaming method.

### 3.1.2 Matrix Tracking Protocol 1

#### Description

Each site $S_i$ maintains a sketch matrix $\mathbf{B}_i \in \mathbb{R}^{2l \times d}$ of the rows seen so far at the site and not sent to the coordinator. The rows of $\mathbf{A}_i$ seen by site $S_i$ and not yet sent to the coordinator are denoted by $\mathbf{C}_i$; i.e. , $\mathbf{C}_i \subset \mathbf{A}_i$ and $\mathbf{B}_i = sketch(\mathbf{C}_i)$. Moreover, each site maintains two counters, $F_i = ||\mathbf{C}_i||_F^2$ and $\hat{F}$, an $\epsilon$-approximation of $||\mathbf{A}||_F^2$. The coordinator maintains also a sketch matrix $\mathbf{B}$ that approximates $\mathbf{A}$, and $\hat{F}$ approximating $||\mathbf{A}||_F^2$.

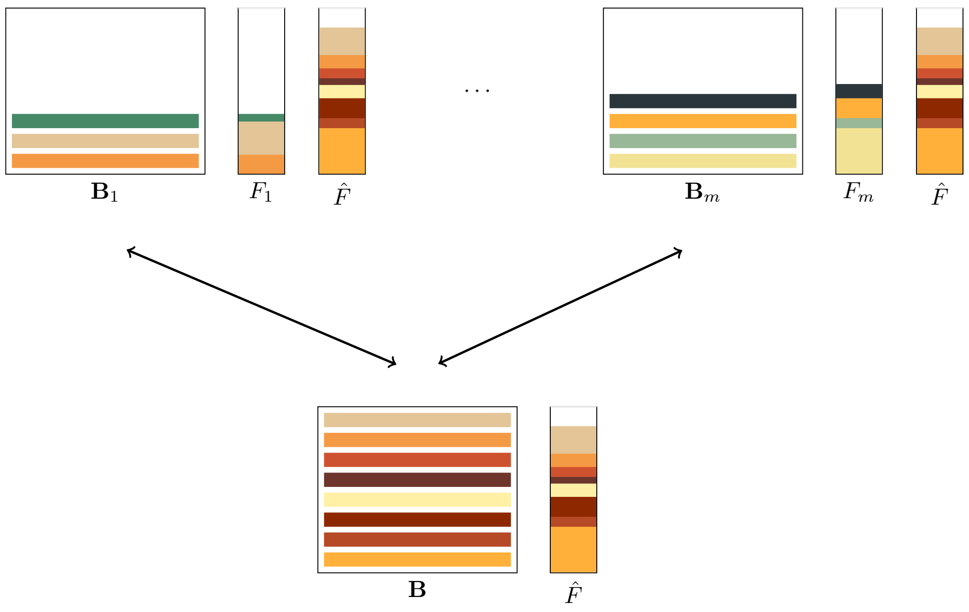

FIGURE 3.1: Distributed setting.

The algorithm runs in epoch. At the beginning, each site sends the first item received, to the coordinator for initialization of $\mathbf{B}$ and $\hat{F}$. Therefore, the received rows will be inserted in $\mathbf{B}$ and $\hat{F}$ will be initialized with the sum of weights, $l_2$ norms, associated with each received row. After initialization, coordinator broadcasts $\hat{F}$ to all sites and the first epoch begins. In each epoch, each site $S_i$ runs the FD with error parameter $\epsilon' = \epsilon/2$ on local stream $\mathbf{A}_i$, and keeps updating $\mathbf{B}_i$ and $F_i$. Once the threshold $(\epsilon/2m)\hat{F}$ is exceeded, it sends $\mathbf{B}_i$ and $F_i$ to C.

Once a new sketch $\mathbf{B}_i$ arrives at C, the coordinator runs FD on the rows of the received sketch and merge its rows into its local sketch $\mathbf{B}$. The coordinator maintains also the sum of the received weights from all sites into a temporary variable $\Delta$. Once $\Delta$ reaches a threshold $\epsilon'\hat{F}$, coordinator updates and broadcasts $\hat{F}$ to all sites.

**Algorithm 2** P1: Deterministic Matrix Tracking (at $S_i$)

```
1: for $(\mathbf{a}_n, w_n)$ in round $j$ do
2:     Update $\mathbf{B}_i \leftarrow FD_{\epsilon'}(\mathbf{B}_i, \mathbf{a}_n)$; and $F_i + = ||\mathbf{a}_n||^2$.
3:     if $(F_i \geq \tau = (\epsilon/2m)\hat{F})$ then
4:         Send $(\mathbf{B}_i, F_i)$ to coordinator; make $\mathbf{B}_i, F_i$ empty.
5:     end if
6: end for
```

**Algorithm 3** P1: Deterministic Matrix Tracking (at $C$)

```
1: On input $(\mathbf{B}_i, F_i)$:
2: Update sketch $\mathbf{B} \leftarrow Merge_{\epsilon'}(\mathbf{B}, \mathbf{B}_i)$ and $\Delta + = F_i$.
3: if $(\Delta/\hat{F} > \epsilon/2)$ then
4:     Update $\hat{F} \leftarrow \hat{F} + \Delta$, reset and broadcast $\hat{F}$ to all sites.
5: end if
```

**Space and Communication**

An epoch ends when coordinator finds out $\Delta > (\epsilon/2)\hat{F}$. Since each $F_i \geq \tau = (\epsilon/2m)\hat{F}$, *in each epoch at most m messages* are sent to $C$. Before ending an epoch, $C$ broadcasts the new value of $\hat{F}$ to all sites ($+m$ one word messages). Overall, in each epoch at most $md/\epsilon'$ words are communicated. Thereby, the total communication is given by

$$O\left(\underbrace{m}_{\text{messages/epoch}} \quad \underbrace{\frac{1}{\epsilon}d}_{\text{sketch size / message}} \quad \underbrace{\frac{1}{\epsilon}log(\beta N)}_{\text{number of epochs}}\right) = O\left(md\frac{1}{\epsilon^2}log(\beta N)\right).$$

For further space, communication, and error analysis the reader is referred to [5].

### 3.1.3 Matrix Tracking Protocol 2

The first protocol proposed a protocol with a communication bound proportional to $1/\epsilon^2$ due to the fact that the site transmits the whole sketch of size $O(d/\epsilon)$ in each message. To reduce communications costs, one can install "local filters" to all sites to allow them to only "push" significants updates to the coordinator; of course, these distributed local filters would have to be safe, in the sense that, they should guarantee the overall error bound for the global query. By the same token, the authors propose a second protocol that reduces communication cost by transmitting only "large directions" of a sketch, instead of the full sketch. To succeed this one can compute the singular vector decomposition of the sketch matrix $[\mathbf{U}, \mathbf{\Sigma V}] = svd(\mathbf{B})$ and transmit only those singular vectors $\mathbf{v} = \mathbf{V}_{:,\mathbf{j}}$ with singular value $\sigma_j$ greater than a threshold.

### Description

All sites and coordinator maintain the same data structures as before. At the beginning $\hat{F}$ is set to zero and is broadcasted to all sites. When site $S_j$ receives a new row, it calls algorithm 4 which basically transmits $||\mathbf{Bx}||^2$ in direction $\mathbf{x}$ when when it is greater than a threshold provided by the coordinator.At each new arrival the site also update the value of $F_i$ and transmits it to the coordinator if it is higher that a threshold $\tau$. Thereby, each site transmits two kind of messages, scalar messages $F_i$ or vector messages $\sigma_j \mathbf{v}_j$.

Once the coordinator receives a scalar message it updates $\hat{F}$ and, after $m$ such messages it broadcasts $\hat{F}$ to all sites. When a vector message arrives at the coordinator, it is merges to the local sketch $\mathbf{B}$.

**Algorithm 4** P2: Deterministic Matrix Tracking (at $S_j$)

```
1:  $F_j + = ||\mathbf{a}_i||^2$
2:  if $(F_j \geq \tau = (\epsilon/m)\hat{F})$ then
3:      Send $F_j$ to coordinator; set $F_j = 0$.
4:      Set $\mathbf{B}_j \leftarrow [\mathbf{B}_j; \mathbf{a}_i]$
5:      $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = svd(\mathbf{B_j})$
6:  end if
7:  for $(\mathbf{v}_l, \sigma_l)$ such that $\sigma_l^2 \geq \frac{\epsilon}{m}\hat{F}$ do
8:      Send $\sigma_l \mathbf{v}_l$ to coordinator; set $\sigma_l = 0$.
9:  end for
10: $\mathbf{B_j} = \mathbf{U \Sigma V}^T$
```

**Algorithm 5** P2: Deterministic Matrix Tracking (at $C$)

```
1: On scalar message $F_j$ from site $S_j$
2: $\hat{F} + = F_j$ and $\#msg + = 1$.
3: if $(\#msg \geq m)$ then
4:     Set $\#msg = 0$ and broadcast $\hat{F}$ to all sites.
5: end if
6: On a vector message $\sigma \mathbf{v}$: append $\mathbf{B} \leftarrow [\mathbf{B}; \sigma \mathbf{v}]$
```

# Chapter 4

# Sliding Window Matrix Sketching

## 4.1 The Logarithmic Method

In this section, we revise the *Logarithmic Method* (LM) of [6], a framework for converting a (mergeable; see Def. 2) streaming matrix sketch method into a sliding window matrix sketching method. The framework is inspired by the idea of Exponential Histograms [7] for monitoring simple statistics – such as number of true bits in a streaming sequence– over a sliding window. Using a streaming matrix sketching method as a black box, LM maintains a variable collection –referred to as a *histogram*– of streaming sketches corresponding to different segments of the input sequence. The LM histogram is continuously updated with the insertion of new sketches for the most recent samples and the retirement of sketches associated with older samples. At any point in time, the structure can produce a matrix sketch that answers a query for the most recent samples within an allowed window of time.

Here, we consider the LM framework in the context of monitoring the principal subspace of an input stream of vectors over a sliding time window. Consider an input stream $\mathcal{S}$ which is a sequence of time-stamped $d$-dimensional vectors $\mathbf{a}_t$ represented as the rows of an $n \times d$ matrix $\mathbf{A}$, and let $W$ be the maximum width of the monitored time window. The objective is to maintain a relatively small amount of information such that such that at any time $t$ and given a query window of width $w$, we are able to generate an $\ell \times d$ matrix $\mathbf{B}$ such that

$$\|\mathbf{A}_{(t-w,t)}^\top \mathbf{A}_{(t-w,t)} - \mathbf{B}^\top \mathbf{B}\| \leq \epsilon \|\mathbf{A}_{(t-w,t)}\|_{\mathrm{F}}^2, \tag{4.1}$$

Here, $\mathbf{A}_{(t-w,t)}$ denotes the submatrix of $\mathbf{A}$ corresponding to the vectors (stream samples) that arrived within the time window $(t-w, t]$. The error function is referred to as the *covariance error* between $\mathbf{A}_{(t-w,t)}$ and $\mathbf{B}$ and is denoted by

$$\text{cov-err}(\mathbf{A}_{(t-w,t)}, \mathbf{B}) \triangleq \|\mathbf{A}_{(t-w,t)}^\top \mathbf{A}_{(t-w,t)} - \mathbf{B}^\top \mathbf{B}\|.$$

Dividing by the squared Frobenious norm of the input in the corresponding window, $\|\mathbf{A}_{(t-w,t)}\|_{\mathrm{F}}^2$, yields the *relative* covariance error. The parameters $W, \ell$ and $\epsilon$ are design-parameters that affect the accuracy of the output sketch, as well as the computational and memory requirements of the LM structure and can be dependent on each other: for example, we will later use Frequent Directions [3] (FD) as the black-box streaming matrix sketching method, and the parameter $\ell$ which determines the number of rows of the matrix sketch will be a function of the accuracy parameter $\epsilon$.

In the sequel, we describe the LM framework in the context of approximating the recent input vectors with a sketch that approximately minimizes the relative covariance error. As explained, the LM framework utilizes a streaming matrix sketching method (oblivious to the sliding window requirement). In our context, we will focus on the Frequent Directions [3] (FD) streaming matrix sketching method for approximating the principal subspace of the input stream. However, we will keep the subsequent description more general, requiring that the black-box streaming matrix sketching method satisfies certain assumptions. As we will show, the FD method satisfies those assumptions.

**Def. 2** (Mergeable Matrix Sketch)**.** *Let $\kappa$ be a matrix sketching algorithm that given a matrix $\mathbf{A}$ and a parameter $\epsilon$ produces a matrix $\mathbf{B} \leftarrow \kappa(\mathbf{A}, \epsilon)$ such that size$(\mathbf{B}) \leq \ell$ and err$(\mathbf{A}, \mathbf{B}) \leq \epsilon$; here, size$(\mathbf{B})$ represents an arbitrarily defined measure of the size of $\mathbf{B}$, and err$(\mathbf{A}, \mathbf{B})$ represents an arbitrary function that measures the error of using $\mathbf{B}$ as a surrogate of $\mathbf{A}$. The sketching algorithm $\kappa$ is* mergeable *iff there exists a* merging *algorithm $\mathcal{A}_{\mathsf{MERGE}}(\cdot, \cdot)$ such that: for any $n_1 \times d$ matrix $\mathbf{A}_1$ and $n_2 \times d$ matrix $\mathbf{A}_2$ with corresponding sketches $\mathbf{B}_1 \leftarrow \kappa(\mathbf{A}_1, \epsilon)$ and $\mathbf{B}_2 \leftarrow \kappa(\mathbf{A}_2, \epsilon)$, the merging algorithm produces a matrix $\mathbf{B} \leftarrow \mathcal{A}_{\mathsf{MERGE}}(\mathbf{B}_1, \mathbf{B}_2)$ such that size$(\mathbf{B}) \leq \ell$ and err$(\mathbf{A}, \mathbf{B}) \leq \epsilon$, where $\mathbf{A}$ is the $(n_1 + n_2) \times d$ matrix formed by vertically stacking the rows of $\mathbf{A}_1$ and $\mathbf{A}_2$.*

**Assumption 1.** *We assume that the black-box streaming matrix sketching method used by the LM framework is mergeable.*

### 4.1.1 The LM Histogram

The LM method generates and maintains a dynamic structure, referred to as the *histogram*. In a high-level, the histogram is a collection of *blocks*. Each block $\mathcal{B}$ is associated with a non-overlapping segment of the input stream specified by a starting and an ending timestamp. It can be thought of as a tuple with four attributes: 1. a starting time, $t_{\text{start}}$, 2. an ending time, $t_{\text{end}}$, 3. an $\ell \times d$ matrix sketch, and 4. a block size. Let $\mathbf{A}_{(t_{\text{start}}, t_{\text{end}})}$ denote the matrix whose rows are the stream samples that arrived within the specified time window. The matrix sketch $\mathbf{B}$ of block $\mathcal{B}$ is associated with that part of the input and generated according to a streaming algorithm in a way that satisfies the same guarantee as 4.1 in the specified window, that is,

$$\|\mathbf{A}^{\top}_{(t_{\text{start}}, t_{\text{end}})} \mathbf{A}_{(t_{\text{start}}, t_{\text{end}})} - \mathbf{B}^{\top}\mathbf{B}\| \leq c \cdot \epsilon \cdot \|\mathbf{A}_{(t_{\text{start}}, t_{\text{end}})}\|_{\text{F}}^2, \tag{4.2}$$

for some constant $c$ and $\epsilon > 0$. We will subsequently formalize this statement (see Lemma 4.1.2). Finally, the size attribute of block $\mathcal{B}$ is equal to the squared Frobenious norm of the corresponding input, *i.e.*, $\|\mathbf{A}_{(t_{\text{start}}, t_{\text{end}})}\|_{\text{F}}^2$.

Blocks are organized into *levels*, labeled $1$ to $L$. Each level is a list of up to $b$ blocks, where $b$ is a design parameter and will be later set to $\Theta(1/\epsilon)$. Therefore, the LM histogram should be alternatively thought of as a list of lists of blocks. Levels with smaller labels contain more recent blocks, that is, all blocks in level $i$ correspond to more recent input samples than those of level $i + 1$. Within a level, blocks are sorted based on their timestamps with the most recent block occupying the right-most position.

Finally, to assist the description, we divide the life-cycle of the block into three stages. When a new block is initialized, it is referred to as the *active* block. At any point in time, the LM structure maintains a single active block. The active block is *not* part of the histogram. When a new active block is generated its starting time is initialized at the current time-stamp (which is equal to the ending time of the most recent block in the LM histogram), its size is initialized to zero and its sketch to the all-zero $\ell \times d$ matrix. The active block is associated with the most recent samples of the input stream and more specifically all samples that have arrived after $t_s$. The blocks contained in the LM histogram are *inactive*. Finally, as explained, the histogram is used to monitor a sliding time window of maximum width $W$. At time $t$, if the ending time of a block is less than $t - W$, *i.e.*, if the time window covered by a block no longer overlaps with the monitored window, then that block has *expired*.

### 4.1.2 Updating the LM histogram

The steps to update the LM histogram when a new time-stamped vector arrives are outlined in Alg. 6.

**Update the active block** When a new stream sample $\mathbf{a}_t \in \mathbb{R}^{1\times d}$ arrives at time $t$, it is inserted in the active block $\mathcal{B}^\star$. More precisely, the sketch of the active block is updated according to the black-box streaming algorithm. The size of the active block is also incremented by an amount equal to the $\ell_2$-norm of the input sample.

If the size of the active block exceeds $\ell$, then the active block is rendered *inactive*: its ending time is set to $t$ and the block is appended Level 1 of the LM histogram. All blocks in the LM histogram are considered inactive.

Finally, a new active block is initialized with starting time $t$, zero size, and an all-zero matrix sketch.

**Merge blocks if necessary** As blocks are gradually inserted into level 1 of the histogram, the number of blocks in the first level will eventually exceed its maximum length $b$. At that point, the two oldest (right-most) blocks of the level are *merged* into a single block which is subsequently appended to the second level. The new block covers the entire time window covered by the two former blocks and its size is the sum of the two individual sizes. Finally, its sketch is generating by merging the sketches of those two blocks.

Observe that based on the above description, all blocks in level 1 of the histogram have a size between $\ell$ and $2\ell$, while based on the merging process, blocks at level $i$ have size between $2^{i-1}\ell$ and $2^i\ell$.

**Remove expired blocks** Finally, as new blocks are inserted in the LM histogram, its size could grow indefinitely. However, as time elapses, some blocks become irrelevant for the monitored window. If $W$ is the maximum duration of the monitored time window, then when the ending time of a block is less than $t - W$ that block becomes irrelevant and can be safely removed from the histogram. Therefore, each time the histogram is updated, we can assume that we perform a clean-up step during which we remove all *expired* blocks.

**Algorithm 6** LM-Update (Logarithmic Method Update) [6]

```
input : Current LM histogram $\mathcal{L}$                      { List of lists of blocks. }
        A tuple $(\mathbf{a}_t, t)$ with a vector $\mathbf{a}_t \in \mathbb{R}^{1\times d}$ and time-stamp $t$.
output  Updated LM histogram $\mathcal{L}$.
 1: Remove all blocks $\mathcal{B}$ in $\mathcal{L}$ for which $\mathcal{B}.t_e < t - W$          { Clean up. }
 2: $\mathcal{B}^\star.t_{\text{end}} \leftarrow t$                         { Update end time of active block. }
 3: $\mathcal{B}^\star.\text{size} \leftarrow \mathcal{B}^\star.\text{size} + \|\mathbf{a}_t\|_2^2$                { Update size of active block. }
 4: $\mathcal{B}^\star$.sketch ← SketchUpdate($\mathcal{B}^\star$.sketch, $\mathbf{a}_t$)      { Update sketch of active
    block (using the black-box algorithm). }
 5: if $\mathcal{B}^\star.\text{size} \geq \ell$ then
 6:     $\mathcal{L}[1]$.appendLeft($\mathcal{B}^\star$)          { Append active block to left of level 1. }
 7:     $\mathcal{B}^\star$ ← NewBlock()                          { Initialize new active block. }
 8:     $\mathcal{B}^\star.t_{\text{start}} \leftarrow t$
 9:     $\mathcal{B}^\star.t_{\text{end}} \leftarrow t$
10:     $\mathcal{B}^\star.\text{size} \leftarrow 0$
11:     $\mathcal{B}^\star$.sketch ← SketchInit()
12: end if
13: $i \leftarrow 1$
14: while $\mathcal{L}[i]$.length $> b$ do
15:     $\mathcal{B}_1 \leftarrow \mathcal{L}[i]$.popRight() { Get and remove the oldest (right-most) block
        of level $i$, }
16:     $\mathcal{B}_2 \leftarrow \mathcal{L}[i]$.popRight()            { as well as the second oldest block. }
17:     $\mathcal{B}$ ← NewBlock()
18:     $\mathcal{B}.t_{\text{start}} \leftarrow \mathcal{B}_1.t_{\text{start}}$
19:     $\mathcal{B}.t_{\text{end}} \leftarrow \mathcal{B}_2.t_{\text{end}}$
20:     $\mathcal{B}$.sketch ← SketchMerge($\mathcal{B}_1$.sketch, $\mathcal{B}_2$.sketch)
21:     $\mathcal{B}.\text{size} \leftarrow \mathcal{B}_1.\text{size} + \mathcal{B}_2.\text{size}$
22:     $\mathcal{L}[i+1]$.appendLeft($\mathcal{B}$)   { Append new block to (the left of) the next
        level. }
23: end while
```

### 4.1.3 Configuration and Guarantees

In the sequel, we describe the guarantees of the LM framework. Those guarantees are contigent upon the choice of the configuration parameters $(b, \ell, \epsilon, W)$. Here, we re-state the main result of [6] using their parameter configuration. Before that, we state one more assumption on the streaming matrix sketching method which is used as a black box.

**Assumption 2.** *The mergeable streaming matrix sketching method used as a black box by the LM framework must take a parameter accuracy parameters $\epsilon \geq 0$ and $\ell$ such that given an input $\mathbf{A} \in \mathbb{R}^{n\times d}$ can generate a sketch $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ such that*

$$\|\mathbf{A}^\top\mathbf{A} - \mathbf{B}^\top\mathbf{B}\| \leq \epsilon\|\mathbf{A}\|_{\mathrm{F}}^2.$$

**Theorem 1** (Theorem 6.1 [6])**.** *We assume access to a mergeable streaming matrix sketching method used as a black box satisfies assumptions 1 and 2, configured with accuracy parameter $\epsilon/8$ and $\ell$. Then for $b = 8/\epsilon$ the LM histogram can generate a matrix sketch that works on time-based sliding window and achieves maximum relative covariance error $\epsilon$ for a given window $W$.*

The next lemma states that in fact a similar guarantee applies to the sketch of each block of the histogram with respect to the corresponding segment of the input stream. The Lemma is useful in the proof of 1, but will also be useful for us in the sequel, when we describe the distributed construction of the LM histogram.

**Lemma 4.1.2.** *Consider a block $\mathcal{B}$ in the histogram produced by the LM (with $b = 8/\epsilon$ and black-box streaming matrix sketching method configured to achieve covariance error upper bounded by $\epsilon/8$), with starting time $t_{start}$, ending time $t_{end}$ and sketch matrix $\mathbf{B}$. Also, let $\mathbf{A}_{(t_{start},t_{end})}$ be the matrix formed by the input samples that arrived in this time window, i.e., the rows of $\mathbf{A}_{(t_{start},t_{end})}$ are the row vectors $\mathbf{a}_t$ for which $t \in (t_{start}, t_{end})$. Then,*

$$\|\mathbf{A}_{(t_{start},t_{end})}^\top\mathbf{A}_{(t_{start},t_{end})} - \mathbf{B}^\top\mathbf{B}\| \leq \frac{\epsilon}{8}\|\mathbf{A}_{(t_{start},t_{end})}\|_{\mathrm{F}}^2, \tag{4.3}$$

*Proof.* If $\mathcal{B}$ is the active block, *i.e.*, the block that maintains the sketch of the most recent incoming rows of $\mathbf{A}$, then $t_{\text{end}}$ is effectively the timestamp of the most recent sample. The black-box streaming matrix sketching algorithm produces the sketch of this block (with accuracy parameter $\epsilon/8$) on all samples that arrived after $t_{\text{start}}$ and hence the lemma holds trivially by the black-box guarantees of the streaming matrix sketching algorithm for each time $t$. This is also true up to the point when the block becomes inactive, and hence the lemma holds trivially for all blocks of level 1.

Assume the property holds for all blocks in level $i$. We will show that it holds for all blocks in level $i+1$. A block $\mathcal{B}$ in level $i+1$ is generating by *merging* two blocks of level $i$. Recall that merging two sketches is also a black-box operation provided by the streaming matrix sketching method of choice. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the level-$i$ blocks that are merged to generate block $\mathcal{B}$. Also, let $(t_{s1}, t_{e1}, \mathbf{B}_1, s_1)$ and $(t_{s2}, t_{e2}, \mathbf{B}_2, s_2)$ be the attributes of the two blocks. The merged blocks are consecutive, which means that $t_{e1} = t_{s2}$. The input samples associated by the new block are given by the concatenation of the samples of the two former blocks. The desired guarantee holds as a result of the mergeability of the black-box streaming matrix sketching algorithm. $\square$

## 4.2 Merging Multiple LM Histograms

In this section we consider a distributed construction of the LM histogram. In particular, we consider a setting with $m$ computational nodes, labeled $1, \dots, m$, with the $i$th node monitoring an individual input stream $S_i$ and maintaining an LM Histogram LMH$^{(i)}$ with accuracy paramter $\epsilon$. The $m$ streams are non-overlapping. Let $S^{\oplus}$ denote the stream formed by the union of the $m$ individual streams. The objective is to generating a single LM Histogram LMH$^{\oplus}$ for $S^{\oplus}$ combining the partial histograms LMH$^{(i)}$, $i = 1, \dots, m$.

Generating the LM histogram for the combined stream is a non-trivial task; the partial histograms LMH$^{(i)}$, $i = 1, \dots, m$ have potentially different sizes (different number of levels and blocks) and their blocks cover different time windows depending on the input stream at each node. It is hence a challenging task to generate LMH$^{\oplus}$ using only information of the partial histograms LMH$^{(i)}$, $i = 1, \dots, m$.

In the sequel, we describe our approach for generating LMH$^{\oplus}$ from the $m$ histograms LMH$^{(i)}$, $i = 1, \dots, m$. We assume that the coordinating node receives the $m$ individual histograms from the corresponding nodes, but no access in the individual input streams. Our approach relies on using each LM histogram LMH$^{(i)}$ as a log based on which we generate a pseudo-stream $\overline{S}_i$. We show that the pseudo-stream[1] $\overline{S}_i$ can in turn be used to answer a query about the corresponding original stream $S_i$, similar to the histogram LMH$^{(i)}$. The advantage is that those pseudo-streams can now be combined into a single pseudo-stream $\overline{S}^{\oplus}$ which can be used to approximately answer queries for the original combined stream $S^{\oplus}$. Finally, the combined histogram LMH$^{\oplus}$ can be constructed by applying the LM method on the combined pseudo-stream, introducing a small additive error. Our analysis relies on mild assumptions on the streaming matrix sketching method used as a black-box. We show that those assumptions are in fact satisfied by the FD method.

### 4.2.1 The LM Merging Algorithm

#### The pseudo-stream as a sketch

Consider a single node and let $S$ denote its input stream. The stream can be represented by a matrix $\mathbf{A}$: the rows of $\mathbf{A}$ correspond to the input stream samples ordered according to their timestamps. Let LMH denote the corresponding LM histogram build on $S$. Given LMH, we can generate a pseudo-stream $\overline{S}$ as follows. We sequentially process every block of the histogram in reverse chronological order, *i.e.*, starting with the oldest block. Block $\mathcal{B}$ contains $\ell \times d$ sketch. We generate $\ell$ stream samples: one sample for each row of $\mathbf{B}$. The samples can be arbitrarily timestamped with any time within the range of the block. For simplicity assume that all samples are timestamped with the starting time $\mathcal{B}.t_{\text{start}}$. We repeat this procedure for all blocks in the histogram, as well as the active block.

Similarly to the original stream, the pseudo-stream $\overline{S}$ itself can be represented by a matrix $\overline{\mathbf{A}}$. The number of rows $\overline{\mathbf{A}}$ does not exceed $Lb\ell$.

[1]We use the term *pseudo-* to emphasize that this is an artificially generated stream.

The following lemma establishes that $\overline{\mathbf{A}}$ can be used to approximately compute the principal subspace of $\mathbf{A}$ with a bounded relative covariance error, similarly to the LM histogram itself.

**Lemma 4.2.3.** *Consider an arbitrary query time $t_q$ within the maximum monitored window, and let $\mathbf{A}_{\geq t_q}$ denote the part of the input corresponding to the input samples that arrived after $t_q$. Similarly, define $\overline{\mathbf{A}}_{\geq t_q}$ as the submatrix of $\overline{\mathbf{A}}$ corresponding to samples whose timestamp is larger than $t_q$. Then,*

$$\|\mathbf{A}_{\geq t_q}^\top \mathbf{A}_{\geq t_q} - \overline{\mathbf{A}}_{\geq t_q}^\top \overline{\mathbf{A}}_{\geq t_q}\| \leq \frac{5\epsilon}{8} \|\mathbf{A}_{\geq t_q}\|_\mathrm{F}^2.$$

*Proof.* Recall that each row of $\overline{\mathbf{A}}$ coincides with a row of a sketch of some block in the LM histogram. In particular, for a given query time $t_q$, the submatrix $\overline{\mathbf{A}}_{\geq t_q}$ is exactly the vertical concatenation of the sketches of all blocks whose starting time is equal or greater than $t_q$.

Let $\mathcal{B}_q$ be the block of the LM histogram for which $\mathcal{B}_q.t_\text{start} \leq t_q \leq \mathcal{B}_q.t_\text{end}$. The matrix $\overline{\mathbf{A}}_{\geq t_q}$ contains only rows of sketches of more recent blocks; it does not contain any of the rows of $\mathcal{B}_q$.sketch. On the contrary, $\mathbf{A}_{\geq t_q}$ contains rows whose timestamp may fall in the range of $\mathcal{B}_q$. Those rows are not approximated at all by $\overline{\mathbf{A}}_{\geq t_q}$. Hence, their contribution in the total covariance error is equal to the sum of their $\ell_2$ norms, or equivalently, the squared Frobenious norm $\|\mathbf{A}_{(t_q,\mathcal{B}_q.t_\text{end})}\|_\mathrm{F}^2$ The error due to this factor is upper bounded by the size of the bucket that contains $t_q$. But as [6] show, the size of the largest block in the histogram is upper bounded by $\epsilon\|\mathbf{A}_{\geq t_q}\|_\mathrm{F}^2/2$.

The remaining rows of $\mathbf{A}_{\geq t_q}$ are being approximated by the corresponding sketches of the blocks that are more recent than $\mathcal{B}_q$. Since the LM histogram is generated based on the configuration of Thm. 1, we know by Lemma 4.1.2 that the sketch of each block approximates the corresponding segment of the input with relative covariance error $\epsilon/8$. Finally, by the decomposability property (Lemma 4.3.5) we know that this implies those rows of $\mathbf{A}_{\geq t_q}$ are approximated by $\overline{\mathbf{A}}_{\geq t_q}$ with a relative covariance error upper bounded by $\epsilon/8$.

Combining the two sources of error, we obtain the desired result. □

**Algorithm 7** LM-Unfold

```
input  : LM Histogram $\mathcal{L}$
output : (Pseudo-) Stream $\overline{\mathcal{S}}$
 1: $L \leftarrow \mathcal{L}$.length
 2: $\overline{\mathcal{S}} \leftarrow$ Empty Stream
 3: for $l = L, \ldots, 1$ do
 4:    for each block $\mathcal{B}$ in $\mathcal{L}[l]$ (least to most recent) do
 5:       $\mathbf{B} \leftarrow \mathcal{B}$.sketch          {$\ell \times d$ sketch associated with block $\mathcal{B}$.}
 6:       for $i = 1, \ldots, \ell$ do
 7:          $\overline{\mathcal{S}}$.append$((\mathbf{B}_{i,:}, \mathcal{B}.t_\text{start}))$    {Create tuple using $i$th row of sketch
             and append to stream.}
 8:       end for
 9:    end for
10: end for
```

### Merging Pseudo Streams

Now, consider again the multiple node setting. Recall that $S^{(i)}$ is the input stream of the $i$th node, and let $\mathbf{A}^{(i)}$ be the $n_i \times d$ matrix formed by vertically stacking the $n_i$ samples in $S^{(i)}$. Further, $S^{\oplus}$ denotes the stream formed by the union of the $m$ individual streams and $\mathbf{A}^{\oplus}$ the $n \times d$ matrix formed by vertically stacking the $n$ samples in $S^{\oplus}$, where $n = n_1 + \ldots, n_m$. We similarly define $\overline{S}^{\oplus}$ and $\overline{\mathbf{A}}^{\oplus}$.

**Lemma 4.2.4.** *For any given query time $t_q$ within the monitored time window, we have*

$$\|\mathbf{A}^{\oplus}_{\geq t_q}{}^{\top}\mathbf{A}^{\oplus}_{\geq t_q} - \overline{\mathbf{A}}^{\oplus}_{\geq t_q}{}^{\top}\overline{\mathbf{A}}^{\oplus}_{\geq t_q}\| \leq \frac{5\epsilon}{8}\|\mathbf{A}^{\oplus}_{\geq t_q}\|_{\mathrm{F}}^2.$$

*Proof.* The stream $\mathbf{A}^{\oplus}_{\geq t_q}$ (and similarly $\overline{\mathbf{A}}^{\oplus}_{\geq t_q}$) can be decomposed into the $m$ streams $\mathbf{A}^{(i)}_{\geq t_q}$ (and $\overline{\mathbf{A}}^{(i)}_{\geq t_q}$, respectively). By Lemma 4.2.3, we have $\|\mathbf{A}^{(i)}_{\geq t_q}{}^{\top}\mathbf{A}^{(i)}_{\geq t_q} - \overline{\mathbf{A}}^{(i)}_{\geq t_q}{}^{\top}\overline{\mathbf{A}}^{(i)}_{\geq t_q}\| \leq \frac{5\epsilon}{8}\|\mathbf{A}^{(i)}_{\geq t_q}\|_{\mathrm{F}}^2$, for all $i = 1, \ldots, m$. Then the desired result follows by lemma 4.3.5. □

### Construct the LM Histogram of the combined pseudo-stream

We obtain the desired LM histogram LMH$^{\oplus}$ by applying the LM on the combined pseudo stream $\overline{S}^{\oplus}$ (*i.e.*, the rows of $\overline{\mathbf{A}}$).

## 4.2.2 Analysis of the Merging Algorithm

Before we proceed with the guarantees of our merging algorithm, we formally state an assumption on the streaming matrix sketching method.

**Assumption 3.** *The streaming matrix sketching method used as a black box, has the following property: given an $n \times d$ input $\mathbf{A}$ it produces a sketch $\mathbf{B}$ with $\|\mathbf{B}\|_{\mathrm{F}}^2 \leq \|\mathbf{A}\|_{\mathrm{F}}^2$. The same holds for a sketch $\mathbf{B}$ produced by the merging sketches of non-overlapping parts of $\mathbf{A}$.*

**Remark 4.2.1.** *This property is satisfied for the sketches generated by the FD method.*

**Theorem 2.** *Consider $m$ LM histograms LMH$^{(i)}$, $i = 1, \ldots, m$ generated on $m$ non-overlapping streams $S^{(i)}$, $i = 1, \ldots, m$, respectively, with accuracy parameter $\epsilon$ (parameter $b = 8/\epsilon$, and streaming sketch size $\ell$). Let LMH$^{\oplus}$ be the histogram generated on the combined pseudo-stream $S^{\oplus}$ with accuracy parameter $\epsilon'$. Then, for any query $t_q$, LMH$^{\oplus}$ returns an $\ell \times d$ sketch $\mathbf{C}$ such that*

$$\|\mathbf{A}^{\oplus}_{\geq t_q}{}^{\top}\mathbf{A}^{\oplus}_{\geq t_q} - \mathbf{C}^{\top}\mathbf{C}\| \leq (\epsilon + \epsilon')\|\mathbf{A}^{\oplus}_{\geq t_q}\|_{\mathrm{F}}^2. \tag{4.4}$$

*Proof.* Recall that the LM histogram LMH$^{\oplus}$ is constructed on the pseudo-stream $\overline{S}^{\oplus}$. For any query $t_q$ in the valid time window, the joint histogram (by construction) can produce an $\ell \times d$ sketch $\mathbf{C}$ such that

$$\|\overline{\mathbf{A}}^{\oplus}_{\geq t_q}{}^{\top}\overline{\mathbf{A}}^{\oplus}_{\geq t_q} - \mathbf{C}^{\top}\mathbf{C}\| \leq \epsilon'\|\overline{\mathbf{A}}^{\oplus}_{\geq t_q}\|_{\mathrm{F}}^2.$$

With respect to the original stream $S^{\oplus}$, we have

$$\|\mathbf{A}_{\geq t_q}^{\oplus}{}^{\top}\mathbf{A}_{\geq t_q}^{\oplus} - \mathbf{C}^{\top}\mathbf{C}\| = \|\mathbf{A}_{\geq t_q}^{\oplus}{}^{\top}\mathbf{A}_{\geq t_q}^{\oplus} - \overline{\mathbf{A}}_{\geq t_q}^{\oplus}{}^{\top}\overline{\mathbf{A}}_{\geq t_q}^{\oplus} + \overline{\mathbf{A}}_{\geq t_q}^{\oplus}{}^{\top}\overline{\mathbf{A}}_{\geq t_q}^{\oplus} - \mathbf{C}^{\top}\mathbf{C}\| \tag{4.5}$$

$$\leq \|\mathbf{A}_{\geq t_q}^{\oplus}{}^{\top}\mathbf{A}_{\geq t_q}^{\oplus} - \overline{\mathbf{A}}_{\geq t_q}^{\oplus}{}^{\top}\overline{\mathbf{A}}_{\geq t_q}^{\oplus}\| + \|\overline{\mathbf{A}}_{\geq t_q}^{\oplus}{}^{\top}\overline{\mathbf{A}}_{\geq t_q}^{\oplus} - \mathbf{C}^{\top}\mathbf{C}\| \tag{4.6}$$

$$\leq \epsilon\|\mathbf{A}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2 + \epsilon'\|\overline{\mathbf{A}}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2, \tag{4.7}$$

where the first inequality follows by the triangle inequality on the spectral norm. It remains to show that $\|\overline{\mathbf{A}}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2 \leq \|\mathbf{A}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2$. Note that the rows of $\overline{\mathbf{A}}_{\geq t_q}^{\oplus}$ correspond to rows of the sketches of the $m$ LM histograms. In particular, the matrix contains exactly all sketches of all blocks in the $m$ histograms for which the starting time was at least equal to $t_q$. The squared Frobenious norm of each block sketch is upper bounded by the squared Frobenious norm of the corresponding part of the input. The input corresponding to those blocks is exactly the rows of $\overline{\mathbf{A}}_{\geq t_q}^{\oplus}$. It hence follows that $\|\overline{\mathbf{A}}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2 \leq \|\mathbf{A}_{\geq t_q}^{\oplus}\|_{\mathrm{F}}^2$. That concludes the proof of Theorem 2. □

## 4.3 Other

**Lemma 4.3.5.** *Consider an $n \times d$ matrix $\mathbf{A}$ and an arbitrary partitioning of its $n$ rows into $k$ sets $S_1, \ldots, S_k$, (i.e., $\cup_{i=1}^k S_i = [n]$ and $S_i \cap S_j = \emptyset \forall i, j \in [n], i \neq j$). Let $\mathbf{A}_{S_j}$ denote the submatrix of $\mathbf{A}$ formed by the rows indexed by $S_j$, and $\mathbf{B}_j$ be sketch of $\mathbf{A}_{S_j}$ with $d$ columns (and an arbitrary number of rows) satisfying the property that*

$$\|\mathbf{A}_{S_j}^\top \mathbf{A}_{S_j} - \mathbf{B}_j^\top \mathbf{B}_j\| \leq \epsilon \|\mathbf{A}_{S_j}\|_{\mathrm{F}}^2, \tag{4.8}$$

*for some $\epsilon > 0$, for $j = 1, \ldots, k$. Finally, let $\mathbf{B}$ be the matrix formed by vertically stacking $\mathbf{B}_j$, $j = 1, \ldots, k$. Then,*

$$\|\mathbf{A}^\top \mathbf{A} - \mathbf{B}^\top \mathbf{B}\| \leq \epsilon \|\mathbf{A}\|_{\mathrm{F}}^2. \tag{4.9}$$

*Proof.* Observe that

$$\mathbf{A}^\top \mathbf{A} = \sum_{i=1}^n \mathbf{A}_{i,:}^\top \mathbf{A}_{i,:} = \sum_{j=1}^k \sum_{i \in S_j} \mathbf{A}_{i,:}^\top \mathbf{A}_{i,:} = \sum_{j=1}^k \mathbf{A}_{S_j}^\top \mathbf{A}_{S_j}. \tag{4.10}$$

Similarly, $\mathbf{B}^\top \mathbf{B} = \sum_{j=1}^k \mathbf{B}_j^\top \mathbf{B}_j$. Combining the above, we have

$$\begin{aligned} \|\mathbf{A}^\top \mathbf{A} - \mathbf{B}^\top \mathbf{B}\| &= \|\sum_{j=1}^k \mathbf{A}_{S_j}^\top \mathbf{A}_{S_j} - \sum_{j=1}^k \mathbf{B}_j^\top \mathbf{B}_j\| = \|\sum_{j=1}^k \mathbf{A}_{S_j}^\top \mathbf{A}_{S_j} - \mathbf{B}_j^\top \mathbf{B}_j\| \\ &\leq \sum_{j=1}^k \|\mathbf{A}_{S_j}^\top \mathbf{A}_{S_j} - \mathbf{B}_j^\top \mathbf{B}_j\| \leq \sum_{j=1}^k \epsilon \|\mathbf{A}_{S_j}\|_{\mathrm{F}}^2 = \epsilon \|\mathbf{A}\|_{\mathrm{F}}^2, \end{aligned} \tag{4.11}$$

where the first inequality follows for the triangle inequality on the spectral norm, and the second by the assumption in (4.8). This completes the proof. □

## 4.4 Frequent Directions

**Lemma 4.4.6.** *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ be the input of the FD Alg. 1, and $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ be the corresponding output. Then, $\|\mathbf{B}\|_{\mathrm{F}}^2 \leq \|\mathbf{A}\|_{\mathrm{F}}^2$.*

*Proof.* It is shown in [3] (Claim 1) that $0 \succeq \mathbf{B}^\top \mathbf{B} \succeq \mathbf{A}^\top \mathbf{A}$. In turn, $\forall \mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}^\top \mathbf{B}^\top \mathbf{B} \mathbf{x} - \mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} \leq 0$. Applying the previous inequality for the vectors of the standard basis and summing the inequalities, we have

$$0 \geq \sum_{i=1}^d \mathbf{e}_i^\top \mathbf{B}^\top \mathbf{B} \mathbf{e}_i - \mathbf{e}_i^\top \mathbf{A}^\top \mathbf{A} \mathbf{e}_i \tag{4.12}$$

$$= \mathrm{Tr}\Big(\mathbf{B}^\top \mathbf{B}\Big) - \mathrm{Tr}\Big(\mathbf{A}^\top \mathbf{A}\Big) = \|\mathbf{B}\|_{\mathrm{F}}^2 - \|\mathbf{A}\|_{\mathrm{F}}^2, \tag{4.13}$$

which completes the proof. □

# Bibliography

[1] J. Misra and D. Gries, "Finding repeated elements," *Science of Computer Programming*, vol. 2, no. 2, pp. 143–152, Nov. 1982.

[2] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in sets and bags," *ACM Transactions on Database Systems*, vol. 28, no. 1, pp. 51–55, Mar. 2003.

[3] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13)*, ACM, New York, NY, USA, pp. 581–588.

[4] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, "Frequent Directions: Simple and deterministic matrix sketching," *arXiv:1501.01711*, 2015.

[5] M. Ghashami, J. M. Phillips, and F. Li, "Continuous matrix approximation on distributed data," in *Proc. of the VLDB Endowment*, vol. 7, no. 10, pp. 809–820, Jun. 2014.

[6] Z. Wei, X. Liu, F. Li, S. Shang, X. Du, and J.-R. Wen, "Matrix sketching over sliding windows," in *Proc. of the 2016 International Conference on Management of Data (SIGMOD '16)*, ACM, New York, NY, USA, pp. 1465–1480.

[7] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1794–1813, 2002.

[8] O. Papapetrou, M. Garofalakis, and A. Deligiannakis, "Sketching distributed sliding-window data streams," *The VLDB Journal*, vol. 24, no. 3, pp. 345–368, Jun. 2015.

[9] P. K. Agarwal, G. Cormode, Z. Huang, J. Phillips, Z. Wei, and K. Yi, "Mergeable summaries," in *Proc. of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems (PODS '12)*, ACM, New York, NY, USA, pp. 23–34.

[10] M. N. Garofalakis, "Distributed data streams," in *Encyclopedia of Database Systems*, pp. 883–890, 2009.

[11] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends in Machine Learning*, vol. 3, no. 2, pp. 123–224, Feb. 2011.

[12] D. Woodruff, "Sketching as a tool for numerical linear algebra," *Foundations and Trends in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[13] W. B. Johnson and J. Lindenstrauss, ”Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, 26(189-206):1, 1984.

[14] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, “Randomized algorithms for the low-rank approximation of matrices," in *Proc. of the National Academy of Sciences*, vol. 104, no. 51, pp. 20167–20172, Dec. 2007.

[15] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, “Latent semantic indexing: a probabilistic analysis," in *Proc. of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems (PODS '98)*, ACM, New York, NY, USA, pp. 159–168.

[16] T. Sarlos, “Improved approximation algorithms for large matrices via random projections," in *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06)*, IEEE Computer Society, Washington, DC, USA, pp. 143–152.

[17] S. S. Vempala, *The random projection method*, American Mathematical Society, 2004.

[18] P. Drineas and R. Kannan, “Pass efficient algorithms for approximating large matrices," in *Proc. of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '03)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 223–232.

[19] R. I. Oliveira, “Sums of random hermitian matrices and an inequality by Rudelson," *Electron. Commun. Probab.*, vol. 15, no. 19, pp. 203–212, 2010.

[20] M. Rudelson and R. Vershynin, “Sampling from large matrices: An approach through geometric functional analysis," *Journal of the ACM (JACM)*, vol. 54, no. 4, art. 21, Jul. 2007.