10 Distributed Matrix Sketching

In order to define a distributed matrix sketching problem thoroughly, one has to specify the *distributed model*, *data model* and the *partition model* of data. The distributed model is often considered as a set of m distributed sites $\{S_1, S_2, \cdots, S_m\}$ and a central coordinator site C where each site has a two way communication channel with C. Note this model is only off by a factor 2 in communication with a model where all pairs of sites can communicate, by routing through the coordinator.

However data model and partition model can be defined in several different ways; in this lecture we consider two following settings: 1) a distributed streaming model for data, with row-wise partitioning of it among sites, 2) a setting of locally stored data in each site, and arbitrary partitioning of data among sites.

Below, we introduce each setting in detail.

10.1 Distributed Streaming, Row-wise Partition

First we define the underlying data model and partition model of problem:

The data model would be the ***"distributed streaming model"***, in which each site observes a disjoint stream of data and together attempt to monitor a function at C. Let $A = (a_1, \ldots, a_n, \ldots) \subset \mathbb{R}^d$ be an unbounded stream of items that is seen by the union of all sites. Eventhough we do not put a bound on the number of items, we assume at the current time t_{now}, n denotes the number of items the system has seen so far; that is at the current time the dataset is $A = (a_1, \ldots, a_n)$ and it forms a $n \times d$ *"distributed streaming matrix"*. The goal is to continuously track a small approximation of matrix A at coordinator. The key resources to optimize in this model are communication between sites and coordinator, and the space usage at coordinator and each site.

The partition model of data is ***"Row-wise partition"***, which means at each time step any item $a_j \in \mathbb{R}^d$ appears at exactly one of the sites and each site S_i must process its incoming elements in streaming fashion; therefore we can denote the set of items site S_i processes as stream $A_i \subset A$, where $\cap_{i=1}^m A_i = \emptyset$ and $\cup_{i=1}^m A_i = A$.

Now we formally state our goal: for any time instance t_{now} (i.e., for any n), C needs to maintain a smaller matrix $B \in \mathbb{R}^{\ell \times d}$ as an approximation to the distributed streaming matrix $A \in \mathbb{R}^{n \times d}$ such that $\ell \ll n$ and for any unit vector x: $|\|Ax\|^2 - \|Bx\|^2| \leq \varepsilon \|A\|_F^2$. The objective is to minimize the total communication between C and all sites $S_1, \ldots, S_m$.

We clearly can run a streaming matrix sketching algorithm (e.g. Frequent Directions [3]) on each site S_i, maintain a sketch B_i as an approximation to local stream A_i, and occasionally send it to C. Exploiting the mergeability property of FD , C can run FD on union of all received sketches and obtain an approximation to input stream A. This forms the main idea of our first algorithm[1] in this model.

10.1.1 Distributed Matrix Tracking Protocol 1

Our first distributed matrix sketching protocol is summarized in algorithms 10.1.1 and 10.1.2. This protocol spends $O((md/\varepsilon^2) \log \|A\|_F)$ words in communication, and at any point in time, the coordinator has an approximation to the stream A and it's Frobenius norm, as

$$0 \leq \|Ax\|^2 - \|\hat{A}x\|^2 \leq \varepsilon \|A\|_F^2 \quad \text{and} \quad (1-\varepsilon)\|A\|_F^2 < \hat{F} \leq \|A\|_F^2$$

Description. We allow each site S_i to maintain two local data structures: a small matrix B_i as an approximation to local stream A_i, and a scalar variable F_i as an approximation to $\|A_i\|_F^2$. We keep similar data structures at C too: a small matrix $\hat{A}$ as an approximation to A, and $\hat{F}$ as approximation to $\|A\|_F^2$. The algorithm runs in epoch. At the begining, all sites send the first item they receive to C, this step is for initializing $\hat{A}$ and $\hat{F}$; as the result $\hat{A}$ will filled

with those items as row vectors, and $\hat{F}$ will be set to sum of squared norms of items. Then C broadcasts $\hat{F}$ to all sites and first epoch starts. In each epoch, any site S_i runs FD (with error parameter $\varepsilon' = \varepsilon/2$) on local stream A_i, and keeps updating B_i and F_i. Once the threshold $\tau = (\varepsilon/2m)\hat{F}$ is triggered, it sends the contents of memory to C. This

Algorithm 10.1.1 P1: Deterministic Matrix Tracking (at S_i)[1]

for $a_n \in A_i$ in round j **do**
  Update $B_i \leftarrow \mathsf{FD}_{\varepsilon'}(B_i, a_n)$; and $F_i \mathrel{+}= \|a_n\|^2$.
  if $(F_i \geq \tau = (\varepsilon/2m)\hat{F})$ **then**
    Send (B_i, F_i) to coordinator; make B_i, F_i empty.

At coordinator side, C runs FD on rows of received sketches $\{B_i\}$ and merge them into local sketch $\hat{A}$. Moreover, it maintains the sum of scalar values it receives in temporary variable Δ, once $\Delta > \varepsilon'\hat{F}$ it updates $\hat{F}$, broadcasts the new value to all sites and goes to next epoch.

Algorithm 10.1.2 P1: Deterministic Matrix Tracking (at C)[1]

On input (B_i, F_i):
Update sketch $\hat{A} \leftarrow \mathsf{Merge}_{\varepsilon'}(\hat{A}, B_i)$ and $\Delta \mathrel{+}= F_i$.
if $(\Delta/\hat{F} > \varepsilon')$ **then**
  Update $\hat{F} \leftarrow \hat{F} + \Delta$, reset Δ and broadcast $\hat{F}$ to all sites.

Space and Communication Analysis

Since each site and coordinator maintains a local sketch of size $1/\varepsilon' \times d$ rows, the space would be $O(d/\varepsilon') = O(d/\varepsilon)$ words. For the communication analysis, note that an epoch ends when coordinator finds out $\Delta > \varepsilon'\hat{F}$, and since each value sent to C is of value equal to or larger than $(\varepsilon'/m)\hat{F}$, we can say in each epoch atmost m messages, each containing $1/\varepsilon'$ d-dimensional row vectors, are sent to C. Before ending an epoch, C broadcasts new value of $\hat{F}$ to all sites, and that adds another m messages, each one word, to the communication. Overall, in each epoch atmost md/ε' words (or m/ε' vectors) are communicated.

In order to bound number of epoches, let's denote value of $\hat{F}$ in i-th epoch as $\hat{F}_i$, then we can say $\hat{F}_i = \hat{F}_{i-1} + \Delta \geq \hat{F}_{i-1} + \varepsilon'\hat{F}_{i-1}$, solving this recursive formula we obtain $\hat{F}_i \geq (1+\varepsilon')^i \hat{F}_0$. Moreover, note $\hat{F}_i \leq \|A\|_F^2$ because there might be some items seen by sites but not sent to C yet. Therefore in last epoch $\|A\|_F^2 \geq (1+\varepsilon')^i \hat{F}_0$ and this bounds number of epoches as

$$i \leq \log_{1+\varepsilon'}(\frac{\|A\|_F^2}{\hat{F}_0}) = O(\frac{1}{\varepsilon'}\log\|A\|_F^2)$$

So in overal, this algorithm requires $O((md/\varepsilon^2)\log(\beta N))$ total words of communication.

Error Analysis

Theorem 10.1.1. *Distributed Matrix Tracking Protocol 1 described in 10.1.1 and 10.1.2 runs under distributed streaming model where union of all streams is $A \subset \mathbb{R}^d$, maintains a sketch of size $1/\varepsilon$ rows at each site and coordinator where $\varepsilon \in (0,1)$ is the error parameter, and approximates Frobenius and spectral norm of A as*

$$(1-\varepsilon')\|A\|_F^2 < \hat{F} \leq \|A\|_F^2 \qquad \textit{and} \qquad 0 \leq \|Ax\|^2 - \|\hat{A}x\|^2 \leq \varepsilon'\|A\|_F^2, \ \ \forall x \in S^{d-1}$$

Proof. We first prove the Frobenius error bound. At any point in time $\|A\|_F^2 = \hat{F} + \sum_{i=1}^m F_i$, since for all $1 \leq i \leq m$, $0 \leq F_i \leq \tau = \varepsilon'/m\hat{F}$, the right hand side of inequality holds. For left hand side, we can say

$$\hat{F} = \|A\|_F^2 - \sum_{i=1}^m F_i \ \ \geq \ \ \|A\|_F^2 - \sum_{i=1}^m \frac{\varepsilon'}{m}\hat{F} \ \ = \ \ \|A\|_F^2 - \varepsilon'\hat{F} \ \ \geq \ \ (1-\varepsilon')\|A\|_F^2$$

And that completes the proof for Frobenius error bound.

For spectral error bound, note at any point in time, the stream A is the union of streams that all sites are receiving, in other words $A = \cup_{i=1}^{m} A_i$. Therefore $\|Ax\|^2 = \sum_{i=1}^{m} \|A_i x\|^2$. Since each site S_i runs FD on A_i and maintains sketch B_i, the following bound holds

$$0 \leq \|A_i x\|^2 - \|B_i x\|^2 \leq \varepsilon' \|A_i\|_F^2 \tag{10.1}$$

Moreover, on coordinator side, the same bound holds between set of received sketches $\{B_i\}_{i=1}$ and matrix $\hat{A}$,

$$0 \leq \sum_{i=1} \|B_i x\|^2 - \|\hat{A}x\|^2 \leq \varepsilon' \sum_{i=1} \|B_i\|_F^2 \tag{10.2}$$

Using these two inequalities we can say

$$\begin{aligned}
\|Ax\|^2 &= \sum_{i=1}^{m} \|A_i x\|^2 \\
&\leq \sum_{i=1}^{m} (\|B_i x\|^2 + \varepsilon' \|A_i\|_F^2) && \text{due to 10.1} \\
&\leq \sum_{i=1}^{m} (\|B_i x\|^2) + \varepsilon' \|A\|_F^2 \\
&\leq \|\hat{A}x\|^2 + \varepsilon' \sum_{i=1} (\|B_i\|_F^2) + \varepsilon' \|A\|_F^2 && \text{due to 10.2} \\
&\leq \|\hat{A}x\|^2 + 2\varepsilon' \|A\|_F^2 \\
&= \|\hat{A}x\|^2 + \varepsilon \|A\|_F^2 && \text{using } \varepsilon = \varepsilon'/2
\end{aligned}$$

And that proves right hand side of the bound, for left hand side:

$$\|Ax\|^2 = \sum_{i=1}^{m} \|A_i x\|^2 \quad \geq \quad \sum_{i=1}^{m} \|B_i x\|^2 \quad \geq \quad \|\hat{A}x\|^2$$

□

Improving Communication Bound. Protocol 1 has a communication bound proportional to $1/\varepsilon^2$, and that's due to transmitting the full sketch $B_i \in \mathbb{R}^{2/\varepsilon \times d}$ in each message. An easy to lower dependence to ε, would be to send "large directions" of B_i, instead of full sketch. For that, one should take $U, S, V = \mathsf{svd}(B_i)$ and transmit vectors $V_{:,j}$ whose $S_{j,j} \geq \tau$. Protocol 2[1], described in next section, implements this idea and obtains a lower communication.

10.1.2 Distributed Matrix Tracking Protocol 2

This protocol is described in algorithms 10.1.3 and 10.1.4. The P2 spends $O(\frac{m}{\varepsilon} \log \|A\|_F)$ communication, and gaurrantees that at all times coordinator approximates $\|Ax\|$ as

$$0 \leq \|Ax\|^2 - \|\hat{A}x\|^2 \leq \varepsilon \|A\|_F^2$$

Description. All sites and coordinator maintain the same data structures as before. Initially $\hat{F}$ is set to zero and is broadcasted to all sites. When site S_j receives a new row, it calls Algorithm 10.1.3, which basically sends $\|B_j x\|^2$ in direction x when it is greater than some threshold provided by the coordinator, if one exists.

On the coordinator side, it either receives a vector form message σv, or a scalar message F_j. For a scalar F_j, it adds it to $\hat{F}$. After at most m such scalar messages, it broadcasts $\hat{F}$ to all sites. For vector message $r = \sigma v$, the coordinator updates $\hat{A}$ by appending r to $\hat{A} \leftarrow [\hat{A}; r]$. The coordinator's protocol is summarized in Algorithm 10.1.4.

Algorithm 10.1.3 P2: Deterministic Matrix Tracking (at S_j)[1]

$F_j \mathrel{+}= \|a_i\|^2$
if ($F_j \geq \frac{\varepsilon}{m}\hat{F}$) **then**
  Send F_j to coordinator; set $F_j = 0$.
Set $B_j \leftarrow [B_j; a_i]$
$[U, \Sigma, V] = \mathsf{svd}(B_j)$
for ((v_ℓ, σ_ℓ) such that $\sigma_\ell^2 \geq \frac{\varepsilon}{m}\hat{F}$) **do**
  Send $\sigma_\ell v_\ell$ to coordinator; set $\sigma_\ell = 0$.
$B_j = U\Sigma V^T$

Algorithm 10.1.4 P2: Deterministic Matrix Tracking (at C)[1]

On a scalar message F_j from site S_j:
Set $\hat{F} \mathrel{+}= F_j$ and $\#\mathsf{msg} \mathrel{+}= 1$.
if ($\#\mathsf{msg} \geq m$) **then**
  Set $\#\mathsf{msg} = 0$ and broadcast $\hat{F}$ to all sites.
On a vector message $r = \sigma v$:
append $\hat{A} \leftarrow [\hat{A}; r]$

10.2 Communication Model, Arbitrary Partition

In this section, we consider a similar distributed setting, in which the data is (already) stored at each sites, and the goal is to do a *one-time* computation of a target function with the constraint of incurring a small communication among sites. The partition model we consider is ***Arbitrary partition***, in which each site holds a finite $n \times d$ matrix A_j such that $A = \sum_{j=1}^{m} A_j$.

In [2], authors proposed the *"Adaptive Compress"* protocol that uses $O(mdk/\varepsilon)$ words of communication and on termination, leaves a matrix $\hat{A}_i \in \mathbb{R}^{n \times d}$ in any site S_i, such that $\hat{A} = \sum_{i=1}^{m} \hat{A}_i$ obtains the relative error bound

$$\|A - \hat{A}\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$$

Note that $\hat{A}$ is not computed explicitly. The key idea behind this alogrithm is that subspace embedding matrices and scaled random sign matrices can preserve columnspace and rowspace of any matrix, respectively, if they are constructed carefully. Employing this idea, coordinator can choose two initial seeds and broadcast them to all sites to multiply their local data into these random matrices. Then each site sends the result, which is a small matrix now, back to coordinator. Coordinator sums them up and computes spectral space of it.

Bibliography

[1] Mina Ghashami, Feifei Li, and Jeff M. Phillips. Continuous matrix approximation on distributed data. In *Proceedings of the 40th International Conference on Very Large Data Bases*, 2014.

[2] Ravindran Kannan, Santosh S Vempala, and David P Woodruff. Principal component analysis and higher correlations for distributed data. *arXiv preprint arXiv:1304.3162*, 2013.

[3] Edo Liberty. Simple and deterministic matrix sketching. In *SIGKDD*, 2013.