1 Overview

In the previous lecture we introduced streaming and sketching and discussed algorithms for the $F_0 = \text{support}(x)$ streaming problem. In this lecture we cover more streaming/sketching and discuss several applications.

- Power of random sign matrices
- $\ell_2$-norm estimation
- Linear regression
- Dimensionality reduction

2 Turnstile Streaming Model

Recall that in the turnstile model, we have some high dimensional vector $x \in \mathbb{R}^n$, and each update is of the form $x_i \leftarrow x_i + v$. We wish to support queries $f(x)$ such as $f(x) = x_i$ or $f(x) = |\text{support}(x)|$. Obviously, we could store $x$ in $O(n)$ space or the whole stream ($O(m)$ space), but our goal is to come up with algorithms that use much less space. We proved the following last time

**Claim 1.** Any deterministic algorithm for $f(x) = |\text{support}(x)|$ requires $\Omega(\min\{n, m\})$ space.

We also saw that we required a randomized/approximate algorithm to do better than $\Omega(\min\{n, m\})$. Today, we discuss different query functions and approaches to streaming/sketching problems.

3 $\ell_2$-norm Estimation

We want to query $f(x) = \|x\|_2$. We have the following result, given in [1]: we cannot do better than $O(n)$ space for querying $\ell_2$-norm unless we allow for randomization and approximation.

3.1 Linear Sketching

The idea is to pick some matrix $\Pi \in \mathbb{R}^{m \times n}$, and store $\mathbb{R}^m \ni y = \Pi x$ (for space savings, we want to choose $m << n$). Let $\Pi_i$ denote the $i$'th column of $\Pi$. Then our update rule becomes

$$y \leftarrow y + v \cdot \Pi_i$$
Note that this allows us to support deletions. [2] show that any streaming algorithm (i.e., choice of query function) can be converted to a linear sketch algorithm to support deletions with a $O(\log n)$ increase in space.

### 3.2 Random Sign Matrix

How can this be used to recover $\ell_2$-norms? We will use the AMS sketch, introduced in [1]. We will choose $\Pi$ such that $\Pi_{i,j} = \frac{\sigma_{i,j}}{\sqrt{m}}$, where each $\sigma_{i,j}$ is chosen uniformly over $\{-1, +1\}$. Such a $\Pi$ is called a random-sign matrix. Then, we will compute $\|y\|_2$ as an estimate of $\|x\|_2$. Before we go into the analysis, we will mention the general idea, which is used often in the analysis of streaming algorithms. Our goal is devise a randomized algorithm that, w.h.p., does not deviate from the true value (in this case $\|x\|_2$) by more than $\epsilon$ fraction of the true value. To do this, we will bound the expectation and variance of the difference between our estimate and the true value, and apply Chebyshev’s inequality.

### 3.3 Analysis

For readability, we drop the subscript from $\|x\|_2$ as it is assumed. Let $A = (1 \pm \epsilon)B \Leftrightarrow (1 - \epsilon)B \leq A \leq (1 + \epsilon)B$.

We can apply Chebyshev’s to see

$$P(||y||^2 - ||x||^2) > \epsilon||x||^2) < \frac{\text{E}(||y||^2 - ||x||^2)^2}{\epsilon^2||x||^4}$$

Now, by linearity of expectation, we have

$$\text{E}(||y||^2) = \sum_{r=1}^{m} \text{E}(y_r^2)$$

where

$$y_r = \frac{1}{\sqrt{m}} \left( \sum_{i=1}^{n} \sigma_{r,i} \cdot x_i \right) \implies y_r^2 = \frac{1}{m} \left( \sum_{i=1}^{n} \sigma_{r,i}^2 x_i^2 + \sum_{i \neq j} \sigma_{r,i} \sigma_{r,j} x_i x_j \right)$$

Since $\sigma_{i,j} = \pm 1$, $\sigma_{i,j}^2 = 1$. Also, note that $\text{E}[\sigma_{i,j}] = 0$. Hence,

$$\text{E}(y_r^2) = \frac{1}{m} \left( \sum_{i=1}^{n} x_i^2 + \sum_{i \neq j} \text{E}(\sigma_{r,i}) \text{E}(\sigma_{r,j}) x_i x_j \right) = \frac{1}{m} ||x||^2$$

where $\text{E}(\sigma_{i,j} \sigma_{k,l}) = \text{E}(\sigma_{i,j}) \text{E}(\sigma_{k,l})$ because each entry in $\Pi$ is independent. This implies that

$$\text{E}(||y||^2) = \frac{m}{m} \sum_{r=1}^{m} ||x||^2 = ||x||^2 \implies \text{Var}(||y||^2) = \text{E}(||y||^2 - \text{E}(||y||^2)) = \text{E}(||y||^2 - ||x||^2)$$

Because the variance of independent random variables is additive, we have

$$\text{Var}(||y||^2) = \sum_{r=1}^{m} \text{Var}(y_r^2)$$
We can expand the terms on the RHS to see

$$\text{Var}(y^2_r) = \mathbb{E}(y^4_r) - (\mathbb{E}(y^2_r))^2 = \frac{1}{m^2} \mathbb{E} \left( \left( \sum_{i=1}^n \sigma_{r,i} x_i \right)^4 \right) - \|x\|^4 / m^2$$

Now, note that $\sigma_{i,j}^2 = 1 \implies \mathbb{E}(\sigma_{i,j}^2) = 1$ and $\mathbb{E}(\sigma_{i,j}^{2k+1}) = 0$. Hence, when we expand the sum to the 4th power, we can discard all terms with odd powers of $\sigma_{i,j}$. Then, we are left with

$$\frac{1}{m^2} \mathbb{E} \left( \left( \sum_{i=1}^n \sigma_{r,i} x_i \right)^4 \right) = \frac{1}{m^2} \left( \sum_i \mathbb{E}(\sigma_{r,i}^4) x_i^4 + 3 \sum_{i \neq j} \mathbb{E}(\sigma_{r,i}^2) \mathbb{E}(\sigma_{r,j}^2) x_i^2 x_j^2 \right)$$

$$= \frac{1}{m^2} \left( \sum_i x_i^4 + 3 \sum_{i \neq j} x_i^2 x_j^2 \right)$$

$$\leq \frac{1}{m^2} \left( \left( \sum_i x_i^2 \right)^2 + 2 \sum_{i \neq j} x_i^2 x_j^2 \right)$$

$$= \frac{1}{m^2} \left( \|x\|^4 + 2 \sum_{i \neq j} x_i^2 x_j^2 \right)$$

which implies that

$$\text{Var}(y^2_r) \leq \frac{1}{m^2} \left( \|x\|^4 + 2 \sum_{i \neq j} x_i^2 x_j^2 \right) - \|x\|^4 / m^2 = \frac{2}{m^2} \sum_{i \neq j} x_i^2 x_j^2 \leq \frac{2}{m^2} \left( \sum_i x_i^2 \right)^2 = 2 \|x\|^4 / m^2$$

so

$$\text{Var}(\|y\|^2) \leq \frac{2\|x\|^4}{m}$$

Now, we can return to the original inequality (from Chebyshev’s), to see

$$P(\|\|y\|^2 - \|x\|^2\| > \epsilon \|x\|^2) < \frac{\text{Var}(\|y\|^2)}{\epsilon^2 \|x\|^4} \leq \frac{2}{m \epsilon^2}$$

Hence, if we set $m = 6/\epsilon^2$, we get failure probability at most 1/3.

We conclude that using $m = O(1/\epsilon^2)$ words of space, our estimate $\|\Pi x\| = (1 \pm \epsilon)\|x\|^2$ with probability at least 2/3. How do we get success probability at least $1 - \delta$? The idea is to run $k = \log(1/\delta)$ independent trials (i.e., have $k$ random sign matrices $\{\Pi^i\}_{i=1}^k$, $\Pi^i \in \mathbb{R}^{m \times n}$). Then, we will have $k$ estimates $\|y^i\|$, and we output their median. For our algorithm to fail, i.e., $\|y\| \neq (1 \pm \epsilon)\|x\|$, we must have at least 1/2 of our trials fail. By the Chernoff bound, this probability decays exponentially in $k$. More formally,

$$\mathbb{E}(\# \text{success}) \geq \frac{2k}{3} \implies P(\frac{k}{2} \text{ succeed}) = \exp(-\Omega(k)) = \delta$$

Hence, to get failure probability at most $(1 - \delta)$ we use $\Theta(1/\epsilon^2 \log(1/\delta))$ space. [3] shows that this is optimal.
3.4 Space

Note that we did not discuss the cost of storing $\Pi$ in the previous space analysis. If we have to store the whole matrix $\Pi \in \mathbb{R}^{m \times n}$, we would use strictly more space than just storing $x$. But note that in our analysis, all that we required was that the entries were independent. Actually, if we recall from the proof, we just needed 4-wise independence (the terms in $E(y^4)$ had at most 4 distinct $\sigma_{i,j}$, which needed to be independent so we could expand the expectations). Hence, we can use a 4-wise independent hash $h : [n] \rightarrow \{\pm 1\}$, which we can store in $\log n$ bits ($\approx 1$ word), and when we need $\sigma_{r,i}$ we just compute $h_r(i)$. Hence, we can store $\Pi$ in $\Theta(\log(1/\delta) \log n)$ space.

3.5 Update Time

What is the update time? Since $y$ is dense, we need to touch every entry when we update, of which there are $m = \Theta(1/\epsilon^2)$. We repeat $\log(1/\delta)$ times, giving us a total update time of $\Theta(1/\epsilon^2 \log(1/\delta))$. But we can do better by making $\Pi$ sparse (i.e., having only one non-zero entry per column, whose position is chosen uniformly). [4] shows $E(\|y\|^2) = \|x\|^2$ and $\text{Var}(\|y\|^2) = O(1/m)\|x\|^4$ in this case, hence the analysis from before goes through. However, the update time is cut down to $O(\log(1/\delta))$ since we only need to touch one entry per column. This is called the Thorup-Zhang sketch. Note that it is strictly better than the AMS sketch. It remains an open problem if we can do better; we can show that using a linear sketch requires at least $O(\sqrt{\log(1/\delta)})$ update time.

4 Linear Regression

Recall the linear regression problem. We are given variables $\{a_i\}_{i=1}^d \in \mathbb{R}^n$, $b \in \mathbb{R}^d$. Let $A$ be the matrix formed by taking $a_i$ as its $i$’th row. We want to find a parameter vector $x \in \mathbb{R}^d$ such that $Ax \approx b$. Formally, our goal is to compute

$$x^* = \arg \min_{x \in \mathbb{R}^d} \|Ax - b\|_2$$

Recall from our problem set that we can write $x^* = (A^TA)^+A^Tb$, where $A^+$ denote the psuedo-inverse of $A$. It turns out that the bottleneck in this algorithm is computing $A^TA \in \mathbb{R}^{d \times d}$, which has $d^2$ entries, each of which require $O(n)$ time to compute (need to take the dot product of two, length $n$ columns). This gives a running time of $O(nd^2)$. We will use linear sketching to store a smaller matrix that approximately preserves norms, due to [5].

**Definition 2.** We call $\Pi$ an $\epsilon$-subspace embedding for a linear subspace $V$ if $V \in V$,

$$\|\Pi v\|^2 = (1 \pm \epsilon)\|v\|^2$$

Then, we can show the following

**Claim 3.** If $\Pi$ is an $\epsilon$-subspace embedding for span\{b, cols(A)\}, then $\tilde{x}^* = \arg \min_x \|\Pi Ax - \Pi b\|$ satisfies

$$\|A\tilde{x}^* - b\| \leq \frac{1 + \epsilon}{1 - \epsilon} \|Ax^* - b\|$$

where $x^*$ is the true OPT.
Proof.

\[(1 - \epsilon)\|A\tilde{x}^* - b\| \leq \|\Pi A\tilde{x}^* - \Pi b\| \leq \|\Pi Ax^* - \Pi b\| \leq (1 + \epsilon)\|Ax^* - b\|
\]

and rearranging gives the desired result. \(\square\)

We did not consider the time it takes to compute \(\Pi A\). Since \(\text{dim}(V) = d\), we can choose an orthonormal basis for \(V\) and make them the columns of a matrix \(U \in \mathbb{R}^{n \times d}\). Then we will have \(U^TU = I\). Moreover, every \(v \in V\) can be written as \(Ux = v\) for some \(x \in \mathbb{R}^d\). Then, \(\forall x \in \mathbb{R}^d\), we have

\[
\|\Pi Ux\| = (1 \pm \epsilon)\|Ux\|^2 = (1 \pm \epsilon)\|x^TU^TUx\| = (1 \pm \epsilon)\|x\|^2
\]

This implies that the magnitudes of all of \((\Pi U)^T(\Pi U)\)'s eigenvalues are \((1 \pm \epsilon)\). If we use the operator norm (max absolute value of eigenvalues), we have

\[
\|((\Pi U)^T(\Pi U) - I\| < \epsilon
\]

Intuitively, under the assumptions here, \(\Pi\) approximately preserves orthonormal bases. Compare this with the previous section, where \(\Pi\) approximately preserved vectors. Now, we can compute

\[
P(\|\Pi U\|^2 - I\| > \epsilon) < \frac{\mathbb{E}(\|((\Pi U)^T(\Pi U) - I\|^2)}{\epsilon^2} \\
< \frac{\mathbb{E}(\|((\Pi U)^T(\Pi U) - I)^2}{\epsilon^2} \\
= \frac{1}{\epsilon^2}O \left( \frac{d^2}{m} \right)
\]

where the second step follows because the first numerator is the largest eigenvalue squared, and the second numerator is the sum of all eigenvalues squared. If we use the Thorup-Zhang sketch, and choose \(m = d^2/\epsilon^2\), we can make the failure probability at most a constant. We can decrease this probability by performing independent trials, as before. Now, since every column in \(\Pi\) has only 1 non-zero, to compute \(\Pi A\), we spend constant time adding unique non-zero in the corresponding column of \(\Pi\) for every non-zero entry in \(A\). Hence, computing \(\Pi A\) takes \(|\text{supp}(A)|\) time. See [8] and [9] for these results/analysis.

5 Dimensionality Reduction

The following theorem, known as the Johnson-Lindenstrauss lemma, was shown in [6].

**Theorem 4.** \(\forall X \subset \mathbb{R}^n, |X| = N, \exists \Pi \in \mathbb{R}^{m \times n}\) with \(m = O(1/\epsilon^2(\log N))\), such that

\[
\forall x, y \in X, \quad \|\Pi x - \Pi y\| = (1 \pm \epsilon)\|x - y\|
\]

Recently, [7] showed that \(m = \Omega(\min\{n, (\log N)/\epsilon^2\})\) is necessary. Note we take the min with \(n\) because we can always choose the identity matrix, which trivially satisfies the property.

We give an outline of the proof. Design a distribution \(D\) over \(\mathbb{R}^{m \times n}\) such that \(\forall z \in \mathbb{R}^n, \|z\| = 1,\) we have, for \(\Pi \sim D,\)

\[
P(|\|\Pi z\|^2 - 1| > \epsilon) < \delta
\]

5
Then, we can set $\delta = 1/\binom{N}{2}$, $z = \frac{x-y}{\|x-y\|}$, and union bound over all of the points in $X$. How do we get such a $D$? As usual, take $D$ to be uniform over $\{1/\sqrt{m}, -1/\sqrt{m}\}^{m \times n}$ with $m = \Theta(1/\epsilon^2 \log(1/\delta))$. Then, instead of looking at the second moment (Chebyshev’s) as we did for streaming, we look at a larger moment. We won’t discuss the details, but they are conceptually similar to problem 2 of problem set 2.

6 Point Query

Now, we return to the turnstile model of updates $x_i \leftarrow x_i + v$. Our query function will be $f(x) = x_i$. Actually, we will have a lot of query function (one for each component of $x$), so let $f_i(x) = x_i$. Our goal is to produce an estimate of $x_i$. Specifically, we want $f_i(x) = x_i \pm \epsilon \|x\|^2$. Suppose we have some $\Pi \in \mathbb{R}^{m \times n}$ such that $\|\Pi_i\| = 1$ and $\langle \Pi_i, \Pi_j \rangle < \epsilon$. We will store $\text{sketch}(x) = \Pi x = y$ and we will output the estimate $\tilde{x}_i = (\Pi^T y)_i = (\Pi^T \Pi x)$. We can expand the matrix product to see

$$\tilde{x}_i = x_i + \sum_{j \neq i} \langle \Pi_i, \Pi_j \rangle x_j$$

But since the dot products are all within $\pm \epsilon$, we have $\tilde{x}_i = x_i \pm \epsilon \|x\|_1$. How do we construct such a $\Pi$? We can apply the JL lemma to $X = \{0, e_1, e_2, \ldots e_n\}$, which gives $m = O((1/\epsilon^2)(\log n))$.

References


