COMPSCI 614: Randomized Algorithms with Applications to Data Science

Prof. Cameron Musco

University of Massachusetts Amherst. Spring 2024.

Lecture 9

Logistics

- Problem Set 2 is due Wednesday at 11:59pm.
- One page project proposal due Tuesday 3/12.

Summary

Last Time:

- Finish up ℓ_0 sampling analysis and applications to distributed and streaming graph connectivity.
- Start on linear sketching for frequency estimation.
- Count-sketch algorithm.

Today:

- Finish up Count-sketch analysis
- ?

Linear Sketching

- **Linear Sketching:** Compress data via a random linear function (i.e., the random matrix $\mathbf{A}$), and prove that we can still recover useful information from the compression.

$$\overset{\text{Random sketching matrix } \mathbf{A}}{\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 1 & -1 & 0 & -1 & -1 & 0 & 1 \\ 0 & -1 & -1 & -1 & 1 & 1 & 1 & 0 \end{bmatrix}} \overset{x}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ -2 \\ 0 \\ 0 \\ 3 \\ 0 \end{bmatrix}} = \overset{\mathbf{A}x}{\begin{bmatrix} 1 \\ -2 \\ 1 \\ 5 \end{bmatrix}}$$

- Linearity is useful because it lets us easily aggregate sketches in distributed settings and update sketches in streaming settings.
- May want to recover non-zero entries of x, estimate norms or other aggregate statistics, find large magnitude entries, sample entries with probabilities according to their magnitudes, etc.

Linear Sketching for ℓ_2 Heavy-Hitters

Set up: We will show how to estimate each entry of a vector $x \in \mathbb{R}^n$ up to error $\pm\epsilon \cdot \|x\|_2$ with probability at least $1 - \delta$, from a small linear sketch, of size $O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$.

- This error guarantee allows recovering the indices of all 'heavy-hitter' entries with magnitude $> 2\epsilon\|x\|_2$.
- What are some possible application of this primitive?

x(1) = 5 | x(2) = -3 | x(2) = 1 | … | x(n) = 0

random hash functions
$\boldsymbol{h}: [n] \to [m]$
$\boldsymbol{s}: [n] \to \{-1,1\}$

rand

$\boldsymbol{s}$

m length array **y**

0	0	0	0	0	0	0	0	0	0

Estimate: $x(i) \approx \mathrm{s}(i) \cdot y(\mathrm{h}(i)) = \mathrm{s}(i) \cdot \sum_{k:\mathrm{h}_j(k)=\mathrm{h}_j(i)} x(k) \cdot \mathrm{s}(k)$

$$= x(i) + \sum_{k \neq i:\mathrm{h}_j(k)=\mathrm{h}_j(i)} x(k) \cdot \mathrm{s}(k) \cdot \mathrm{s}(i)$$

View as a Linear Sketch

Random sketching matrix **A**

$$\mathbf{A} \; \begin{array}{c} x \\ \begin{bmatrix} 5 \\ -3 \\ 1 \\ -2 \\ 0 \\ 0 \\ 3 \\ 0 \end{bmatrix} \end{array} = \begin{array}{c} \mathbf{y} \\ \begin{bmatrix} 4 \\ 0 \\ 3 \\ 1 \end{bmatrix} \end{array}$$

Count Sketch Algorithm – Psuedocode

- Let $m = O(1/\epsilon^2)$ and $t = O(\log(1/\delta))$.
- Pick t random pairwise independent hash functions $\mathbf{h}_1, \ldots, \mathbf{h}_t : [n] \to [m]$.
- Pick t random pairwise independent hash functions $\mathbf{s}_1, \ldots, \mathbf{s}_t : [n] \to \{-1, 1\}$.
- Compute t independent estimates of $x(i)$ as $\tilde{\mathbf{x}}_j(i) = \mathbf{s}(i) \cdot \sum_{k:\mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathbf{s}(k)$.
- Output the median of $\{\tilde{\mathbf{x}}_1(i), \ldots, \tilde{\mathbf{x}}_t(i)\}$ as our final estimate of $x(i)$.

Concentration Analysis

Recall: $\tilde{\mathbf{x}}_j(i) = \mathbf{s}(i) \cdot \sum_{k:\mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathbf{s}(k)$.

What is $\mathbb{E}[\tilde{\mathbf{x}}_j(i)]$?

$$\begin{aligned}
\mathbb{E}[\tilde{\mathbf{x}}_j(i)] &= x(i) + \mathbb{E}\left[\sum_{k\neq i:\mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathbf{s}(k) \cdot \mathbf{s}(i)\right] \\
&= x(i) + \sum_{k\neq i:\mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathbb{E}[\mathbf{s}(k) \cdot \mathbf{s}(i)] \\
&= x(i).
\end{aligned}$$

Concentration Analysis

Recall: $\tilde{\mathrm{x}}_j(i) = \mathrm{s}(i) \cdot \sum_{k:\mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathrm{s}(k)$.

What is $\mathrm{Var}[\tilde{\mathrm{x}}_j(i)]$?

$$\begin{aligned}
\mathrm{Var}[\tilde{\mathrm{x}}_j(i)] &= \mathrm{Var}\left[\sum_{k \neq i : \mathbf{h}_j(k)=\mathbf{h}_j(i)} x(k) \cdot \mathrm{s}(k) \cdot \mathrm{s}(i)\right] \\
&= \mathrm{Var}\left[\sum_{k \neq i} \mathbb{I}_{\mathbf{h}_j(k)=\mathbf{h}_j(i)} \cdot x(k) \cdot \mathrm{s}(k) \cdot \mathrm{s}(i)\right] \\
&= \sum_{k \neq i} \mathrm{Var}\left[\mathbb{I}_{\mathbf{h}_j(k)=\mathbf{h}_j(i)} \cdot x(k) \cdot \mathrm{s}(k) \cdot \mathrm{s}(i)\right] \\
&= \sum_{k \neq i} \frac{1}{m} \cdot x(k)^2 \leq \frac{\|x\|_2^2}{m}.
\end{aligned}$$

Concentration Analysis

Recall: $\tilde{\mathrm{x}}_j(i) = \mathrm{s}(i) \cdot \sum_{k:\mathrm{h}_j(k)=\mathrm{h}_j(i)} x(k) \cdot \mathrm{s}(k)$.

What is an upper bound on $\Pr[|\tilde{\mathrm{x}}_j(i) - x(i)| \geq \epsilon \|x\|_2]$?

By Chebyshev's inequality:

$$\Pr[|\tilde{\mathrm{x}}_j(i) - x(i)| \geq \epsilon \|x\|_2] \leq \frac{\mathrm{Var}[\tilde{\mathrm{x}}_j(i)]}{\epsilon^2 \|x\|_2^2} \leq \frac{1}{\epsilon^2 \cdot m}$$

If we set $m = \frac{3}{\epsilon^2}$, then our estimate is good with probability $\geq 2/3$.

How large must we set m to increase our success probability to $\geq 1 - \delta$?

Median Trick for Count Sketch

To achieve $O(\log(1/\delta))$ dependence, Count Sketch uses the 'median trick'.

- Set $m = 3/\epsilon^2$ so each estimate $\tilde{\mathbf{x}}_j(i)$ is a $\pm\epsilon\|x\|_2$ approximation to $x(i)$ with probability at least $2/3$.
- If we make t such estimates independently, we expect $2/3 \cdot t$ of them to be correct.
- By a Chernoff bound, for $t = O(\log(1/\delta))$, $> 1/2$ will be correct with probability at least $1 - \delta$.
- Thus, the median estimate will be correct with probability at least $1 - \delta$.

Approximate Matrix Multiplication

Matrix Multiplication Problem

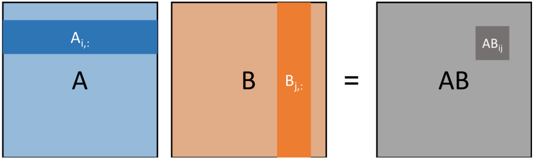
Given $A, B \in \mathbb{R}^{n \times n}$ would like to compute $C = AB$. Requires n^ω time where $\omega \approx 2.373$ in theory.

Today: We'll see how to compute an approximation in $O(n^2)$ time via a simple sampling approach.

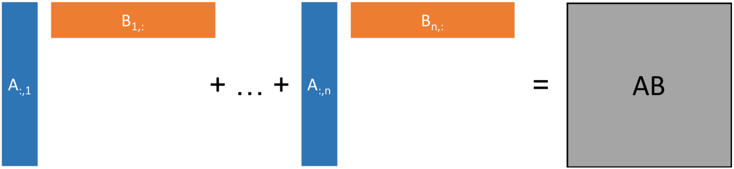
- One of the most fundamental algorithms in randomized numerical linear algebra. Forms the building block for many other algorithms.

Outer Product View of Matrix Multiplication

Inner Product View: $[AB]_{ij} = \langle A_{i,:}, B_{j,:} \rangle = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$.



Outer Product View: Observe that $C_k = A_{:,k} B_{k,:}$ is an $n \times n$ matrix with $[C_k]_{ij} = A_{jk} \cdot B_{kj}$. So $AB = \sum_{k=1}^{n} A_{:,k} B_{k,:}$



Basic Idea: Approximate **AB** by sampling terms of this sum.

Canonical AMM Algorithm

Approximate Matrix Multiplication (AMM):

- Fix sampling probabilities $p_1, \ldots, p_n$ with $p_i \geq 0$ and $\sum_{[n]} p_i = 1$.
- Select $\mathbf{i}_1, \ldots, \mathbf{i}_t \in [n]$ independently, according to the distribution $\Pr[\mathbf{i}_j = k] = p_k$.
- Let $\overline{\mathbf{C}} = \frac{1}{t} \cdot \sum_{j=1}^{t} \frac{1}{p_{\mathbf{i}_j}} \cdot A_{:,\mathbf{i}_j} B_{\mathbf{i}_j,:}$.

Claim 1: $\mathbb{E}[\overline{\mathbf{C}}] = AB$

$$\mathbb{E}[\overline{\mathbf{C}}] = \frac{1}{t} \sum_{j=1}^{t} \mathbb{E}\left[\frac{1}{p_{\mathbf{i}_j}} \cdot A_{:,\mathbf{i}_j} B_{\mathbf{i}_j,:}\right] = \frac{1}{t} \sum_{j=1}^{t} \sum_{k=1}^{n} p_k \cdot \frac{1}{p_k} \cdot A_{:,k} B_{k,:} = \frac{1}{t} \sum_{j=1}^{t} AB = AB$$

Weighting by $\frac{1}{p_{\mathbf{i}_j}}$ keeps the expectation correct. Key idea behind **importance sampling** based methods.