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 $\ell_0$ 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 $A$), and prove that we can still recover useful information from the compression.

\[
\begin{bmatrix}
1 & -1 & 0 & 0 & 1 & -1 & 0 & 1 \\
-1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\
1 & 1 & -1 & 0 & -1 & -1 & 0 & 1 \\
0 & -1 & -1 & -1 & 1 & 1 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \end{bmatrix}
= 
\begin{bmatrix}
1 \\
-2 \\
1 \\
5
\end{bmatrix}
\]

• 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.
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?
Visually Estimate:

\[ x(i) \approx s(i) \cdot y(h(i)) = s(i) \cdot \sum_{k:h_j(k)=h_j(i)} x(k) \cdot s(k) = x(i) + \sum_{k \neq i: h_j(k)=h_j(i)} x(k) \cdot s(k) \cdot s(i) \]
Random sketching matrix $A$ 

\[
\begin{bmatrix}
5 \\ -3 \\ 1 \\ -2 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0
\end{bmatrix}
= \begin{bmatrix} 4 \\ 0 \\ 3 \\ 1 \end{bmatrix}
\]
Count Sketch Algorithm – Psuedocode

- Let $m = O(1/\epsilon^2)$ and $t = O(\log(1/\delta))$.
- Pick $t$ random pairwise independent hash functions $h_1, \ldots, h_t : [n] \rightarrow [m]$.
- Pick $t$ random pairwise independent hash functions $s_1, \ldots, s_t : [n] \rightarrow \{-1, 1\}$.
- Compute $t$ independent estimates of $x(i)$ as $\tilde{x}_j(i) = s(i) \cdot \sum_{k : h_j(k) = h_j(i)} x(k) \cdot s(k)$.
- Output the median of $\{\tilde{x}_1(i), \ldots, \tilde{x}_t(i)\}$ as our final estimate of $x(i)$. 
Concentration Analysis

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

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

$$
\mathbb{E}[\tilde{x}_j(i)] = x(i) + \mathbb{E} \left[ \sum_{k \neq i: h_j(k) = h_j(i)} x(k) \cdot s(k) \cdot s(i) \right]
$$

$$
= x(i) + \sum_{k \neq i: h_j(k) = h_j(i)} x(k) \cdot \mathbb{E}[s(k) \cdot s(i)]
$$

$$
= x(i).
$$
Concentration Analysis

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

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

$$\text{Var}[\tilde{x}_j(i)] = \text{Var} \left[ \sum_{k \neq i: h_j(k)=h_j(i)} x(k) \cdot s(k) \cdot s(i) \right]$$

$$= \text{Var} \left[ \sum_{k \neq i} I_{h_j(k)=h_j(i)} \cdot x(k) \cdot s(k) \cdot s(i) \right]$$

$$= \sum_{k \neq i} \text{Var} \left[ I_{h_j(k)=h_j(i)} \cdot x(k) \cdot s(k) \cdot s(i) \right]$$

$$= \sum_{k \neq i} \frac{1}{m} \cdot x(k)^2 \leq \frac{\|x\|_2^2}{m}.$$
Recall: $\tilde{x}_j(i) = s(i) \cdot \sum_{k: h_j(k) = h_j(i)} x(k) \cdot s(k)$.

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

By Chebyshev’s inequality:

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

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 $\tilde{x}_j(i)$ 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
Given $A, B \in \mathbb{R}^{n \times n}$ would like to compute $C = AB$. Requires $n^\omega$ 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 $i_1, \ldots, i_t \in [n]$ independently, according to the distribution $\Pr[i_j = k] = p_k$.
• Let $\bar{C} = \frac{1}{t} \cdot \sum_{j=1}^{t} \frac{1}{p_{i_j}} \cdot A_{:,i_j}B_{i_j,:}$.

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

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

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