Lecture 7: Dynamic sampling

Dimension Reduction
Plan

• Admin:
  – PSet 2 released later today, due next Wed
  – Alex office hours: Tue 2:30-4:30

• Plan:
  – Dynamic streaming graph algorithms
  – S2: Dimension Reduction & Sketching

• Scriber?
Sub-Problem: dynamic sampling

- **Stream**: general updates to a vector $x \in \{-1,0,1\}^n$

- **Goal**: 
  - Output $i$ with probability $\frac{|x_i|}{\sum_j |x_j|}$
Dynamic Sampling

- **Goal:** output \( i \) with probability \( \frac{|x_i|}{\sum_j |x_j|} \)
- **Let** \( D = \{ i \text{ s.t. } x_i \neq 0 \} \)
- **Intuition:**
  - Suppose \( |D| = 10 \)
    - How can we sample \( i \) with \( x_i \neq 0 \)?
    - Each \( x_i \neq 0 \) is a 1/10-heavy hitter
    - Use CountSketch \( \Rightarrow \) recover all of them
    - \( O(\log n) \) space total
  - Suppose \( |D| = 10\sqrt{n} \)
    - Downsample: pick a random set \( I \subset [n] \) s.t. \( \Pr[i \in I] = \frac{1}{\sqrt{n}} \)
    - Focus on substream on \( i \in I \) only (ignore the rest)
    - What’s \( |D \cap I| \)?
      - In expectation = 10
    - Use CountSketch on the downsampling stream \( I \)...
  - In general: prepare for all levels
Basic Sketch

- **Hash function** $g: [n] \rightarrow [n]$
- **Let** $h(i) = \# \text{tail zeros in } g(i)$
  - $\Pr[h(i) = j] = 2^{-j-1}$ for $j = 0 \ldots L - 1$ and $L = \log_2 n$
- **Partition stream into substreams** $I_0, I_1, \ldots, I_L$
  - Substream $I_j$ focuses on elements with $h(i) = j$
  - $E[|D \cap I_j|] = |D| \cdot 2^{-j-1}$
- **Sketch:** for each $j = 0, \ldots, L$,
  - Store $CS_j$: CountSketch for $\phi = 0.01$
  - Store $DC_j$: distinct count sketch for approx=1.1
    - $F_2$ would be sufficient here!
  - Both for success probability $1 - 1/n$
Estimation

- Find a substream $I_j$ s.t. $DC_j$ output $\in [1,20]$
  - If no such stream, then FAIL
- Recover all $i \in I_j$ with $x_i \neq 0$ (using $CS_j$)
- Pick any of them at random

Algorithm DynSampleBasic:

Initialize:
- hash function $g:[n] \rightarrow [n]$
  - $h(i) =$ # tail zeros in $g(i)$
- CountSketch sketches $CS_j$, $j \in [L]$
- DistinctCount sketches $DC_j$, $j \in [L]$

Process(int $i$, real $\delta_i$):
- Let $j = h(i)$
- Add $(i, \delta_i)$ to $CS_j$ and $DC_j$

Estimator:
- Let $j$ be s.t. $DC_j \in [1,20]$
  - If no such $j$, FAIL
  - $i =$ random heavy hitter from $CS_j$
- Return $i$
Analysis

• If $|D| < 10$
  - then $|D \cap I_j| \in [1,10]$ for some $j$
• Suppose $D \geq 10$
  - Let $k$ be such that $|D| \in [10 \cdot 2^k, 10 \cdot 2^{k+1}]
  \quad E[|D \cap I_k|] = |D| \cdot 2^{-k-1}
  \in [5,10]
  \quad Var[|D \cap I_k|] \leq |D| \cdot 2^{-k-1} \leq 10$
• Chebyshev: $|D \cap I_k|$ deviates from expectation by $4 > \sqrt{1.5Var}$ with probability at most $\frac{1}{1.5} < 0.7$
  - i.e., probability of FAIL is at most 0.7

Algorithm DynSampleBasic:

Initialize:
  hash function $g:[n] \to [n]
  h(i) = $ # tail zeros in $g(i)$
  CountSketch sketches $CS_j$, $j \in [L]$
  DistinctCount sketches $DC_j$, $j \in [L]$

Process(int $i$, real $\delta_i$):
  Let $j = h(i)$
  Add $(i, \delta_i)$ to $CS_j$ and $DC_j$

Estimator:
  Let $j$ be s.t. $DC_j \in [1,20]$
  If no such $j$, FAIL
  $i =$ random heavy hitter from $CS_j$
  Return $i$
Analysis (cont)

• Let $j$ with $DC_j \in [1, 20]$
  – All heavy hitters = $D \cap I_j$
  – $CS_j$ will recover a heavy hitter, i.e., $i \in D \cap I_j$

• By symmetry, once we output some $i$, it is random over $D$

• Randomness?
  – We just used Chebyshev
  \[ \Rightarrow \text{pairwise } g \text{ is OK !} \]

Algorithm DynSampleBasic:

Initialize:
  hash function $g: [n] \rightarrow [n]$
  $h(i) = \# \text{tail zeros in } g(i)$
  CountSketch sketches $CS_j$, $j \in [L]$
  DistinctCount sketches $DC_j$, $j \in [L]$

Process(int $i$, real $\delta_i$):
  Let $j = h(i)$
  Add $(i, \delta_i)$ to $CS_j$ and $DC_j$

Estimator:
  Let $j$ be s.t. $DC_j \in [1, 20]$
  If no such $j$, FAIL
  $i = \text{random heavy hitter from } CS_j$
  Return $i$
Dynamic Sampling: overall

- **DynSampleBasic guarantee:**
  - **FAIL:** with probability \( \leq 0.7 \)
  - Otherwise, output a random \( i \in D \)
    - Modulo a negligible probability of \( CS/DC \) failing

- **Reduce FAIL probability?**

- **DynSample-Full:**
  - Take \( k = O(\log n) \) independent DynSampleBasic
  - Will not FAIL in at least one with probability at least \( 1 - 0.7^k \geq 1 - 1/n \)
  - **Space:** \( O(\log^4 n) \) words for:
    - \( k = O(\log n) \) repetitions
    - \( O(\log n) \) substreams
    - \( O(\log^2 n) \) for each \( CS_j, DC_j \)
Back to Dynamic Graphs

• Graph $G$ with edges inserted/deleted
• Define node-edge incidence vectors:
  – For node $v$, we have vector:
    • $x_v \in \mathbb{R}^p$ where $p = \binom{n}{2}$
    • For $j > v$: $x_v(v, j) = +1$ if edge $(v, j)$ exists
    • For $j < v$: $x_v(j, v) = -1$ if edge $(j, v)$ exists

• Idea:
  – Use Dynamic-Sample-Full to sample an edge from each vertex $v$
  – Collapse edges
  – How to iterate?

• Property:
  – For a set $Q$ of nodes
  – Consider: $\sum_{v \in Q} x_v$
  – **Claim:** has non-zero in coordinate $(i, j)$ iff edge $(i, j)$ crosses from $Q$ to outside (i.e., $|Q \cap \{i, j\}| = 1$)

• Sketch enough for: for any set $Q$, can sample an edge from $Q$!
Dynamic Connectivity

• Sketching algorithm:
  – Dynamic-Sample-Full for each $x_v$

• Check connectivity:
  – Sample an edge from each node $v$
  – Contract all sampled edges
  – $\Rightarrow$ partitioned the graph into a bunch of components $Q_1, \ldots, Q_l$ (each is connected)
  – Iterate on the components $Q_1, \ldots, Q_l$

• How many iterations?
  – $O(\log n)$ - each time we reduce the number of components by a factor $\geq 2$

• Issue: iterations not independent!
  – Can use a fresh Dynamic-Sampling-Full for each of the $O(\log n)$ iterations

$x_v \in \mathbb{R}^p$ where $p = \binom{n}{2}$
for $j > v$: $x_v(v, j) = +1$ if $\exists (v, j)$
for $j < v$: $x_v(j, v) = -1$ if $\exists (j, v)$
A little history

- [Ahn-Guha-McGregor’12]: the above streaming algorithm
  - Overall $O(n \cdot \log^4 n)$ space

- [Kapron-King-Mountjoy’13]:
  - Data structure for maintaining graph connectivity under edge inserts/deletes
    - First algorithm with $(\log n)^{O(1)}$ time for update/connectivity!
    - Open since ‘80s
Section 2:

Dimension Reduction & Sketching
Why?

• Application: Nearest Neighbor Search in high dimensions

• **Preprocess**: a set $D$ of points

• **Query**: given a query point $q$, report a point $p \in D$ with the smallest distance to $q$
Motivation

• Generic setup:
  – Points model objects (e.g. images)
  – Distance models (dis)similarity measure

• Application areas:
  – machine learning: k-NN rule
  – speech/image/video/music recognition, vector quantization, bioinformatics, etc...

• Distance can be:
  – Euclidean, Hamming
Low-dimensional: easy

• Compute Voronoi diagram
• Given query $q$, perform point location
• Performance:
  – Space: $O(n)$
  – Query time: $O(\log n)$
High-dimensional case

- All exact algorithms degrade rapidly with the dimension $d$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Query time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full indexing</td>
<td>$O(\log n \cdot d)$</td>
<td>$n^{0(d)}$ (Voronoi diagram size)</td>
</tr>
<tr>
<td>No indexing – linear scan</td>
<td>$O(n \cdot d)$</td>
<td>$O(n \cdot d)$</td>
</tr>
</tbody>
</table>
Dimension Reduction

• Reduce high dimension?!
  – “flatten” dimension $d$ into dimension $k \ll d$

• Not possible in general: packing bound

• But can if: for a fixed subset of $\mathbb{R}^d$