Topic 11: Sketching Graphs

David Woodruff

Outline

• Sketching Model
  • Estimating the Euclidean norm of a vector
  • Finding a non-zero coordinate of a vector

• Graph sketching motivation
  • Boruvka’s spanning tree algorithm
  • Finding a spanning tree from a sketch

Sketching

• Random linear projection $S: \mathbb{R}^n \rightarrow \mathbb{R}^k$ that preserves properties of any $x \in \mathbb{R}^n$ with high probability, where $k \ll n$

\[
\begin{bmatrix}
S \\
x
\end{bmatrix}
\begin{bmatrix}
Sx
\end{bmatrix}
\rightarrow \text{answer}
\]

• The matrix $S$ does not depend on $x$, e.g., $S$ is a random matrix (typically, we require the entries of $S$ be $O(\log n)$ bits)
Estimating the Norm of a Vector

- For a vector \( x \in \mathbb{R}^n \), its (squared) Euclidean norm is \( |x|^2 = \sum x_i^2 \)
- Want to output a number \( Z \) for which \( 1 - \epsilon \leq Z \leq 1 + \epsilon \) \( |x|^2 \)
- Choose a 2-wise independent hash function \( h : [n] \rightarrow [k] \)
- Choose a 4-wise independent hash function \( \sigma : [n] \rightarrow \{-1,1\} \)

\[ \sum_{i : h(i) = 2} \sigma_i x_i \]

CountSketch

- CountSketch is a linear map \( S : \mathbb{R}^n \rightarrow \mathbb{R}^k \)
- A row \( i \) of \( S \) is a hash bucket, and \( (Sx)_i \) is the value in the bucket
- Output \( |Sx|^2 \)

\[ E[|Sx|^2] = E[\sum \delta(h(i) = i) \sigma(i) x_i^2] \]
\[ = \sum_{i,j} x_i x_j E[\delta(h(i) = j) \sigma(i) \sigma(j)] \]
\[ = \sum x_i^2 E[\sum_{i,j} \delta(h(i) = j) \sigma(i) \sigma(j)] \]
\[ = \frac{\sum x_i^2}{k} = |x|^2 \]

Estimating the Norm from CountSketch

- In recitation, you will show \( \text{Var}[|Sx|^2] = O(|x|^4/k) \)
- By Chebyshev’s inequality,

\[ \Pr \left[ |Sx|^2 - |x|^2 \geq \epsilon |x|^2 \right] \leq \frac{\text{Var}[|Sx|^2]}{\epsilon^2 |x|^4} \leq \frac{1}{10} \quad \text{provided} \quad k = \Theta\left(\frac{1}{\epsilon^2}\right) \]
- If \( S \) has \( k = \Theta\left(\frac{1}{\epsilon^2}\right) \) rows, can estimate \( |x|^2 \) from \( Sx \) up to a \((1 + \epsilon)\)-factor with probability at least 9/10

Outline

- Sketching Model
  - Estimating the Euclidean norm of a vector
  - Finding a non-zero coordinate of a vector
- Graph sketching motivation
  - Boruvka’s spanning tree algorithm
  - Finding a spanning tree from a sketch
A 1-Sparse Recovery Algorithm

- Underlying n-dimensional vector \( x \) initialized to \( 0^n \)
- Stream of updates \( x_i \leftarrow x_i + \Delta_i \) for \( \Delta_i \in \{-1,1\} \)
- Promised that at all times, \(-\text{poly}(n) \leq x_i \leq \text{poly}(n)\)
- Want a procedure which with probability \( 1 - 1/\text{poly}(n) \), if \( x \) is 1-sparse, i.e., has exactly one non-zero entry \( x_i \), it returns \((x_i, i)\)
- Otherwise output FAIL

A 1-Sparse Recovery Algorithm

- Choose 2-universal hash functions \( h_1, \ldots, h_{2\log n} \) each mapping \([n]\) to \( \{0,1\} \)
- Let \( x^{00} = x^{01} = x^{10} = 0 \) if \( h_j = 0 \), else \( x^{00} = x^{01} = x^{10} = 0 \)
- For each \( i \in \{1, 2, \ldots, 2\log n\} \), \( x = x^{10} + x^{11} \)
- Let \( S^{ab} \) be a CountSketch, and maintain \( S^{ab} \cdot x^{ab} \) for \( i \in [n] \) and \( b \in \{0,1\} \)

Output:

- If there’s an \( i \) with \( S^{10} \cdot x^{10} > 0 \) and \( S^{11} \cdot x^{11} > 0 \), output FAIL
- If for all \( i \), \( S^{10} \cdot x^{10} = S^{11} \cdot x^{11} = 0 \), output FAIL
- For each \( i \), if there’s a \( b \in \{0,1\} \) with \( S^{ab} \cdot x^{ab} > 0 \), let \( T^i = h^{-1}(b) \), else \( T^i = [n] \)
- If \( \cap_i T^i = 1 \), output the item in \( \cap_i T^i \), else output FAIL

A 1-Sparse Recovery Algorithm

- If there’s an \( i \) with \( S^{10} \cdot x^{10} > 0 \) and \( S^{11} \cdot x^{11} > 0 \), output FAIL
- If for all \( i \), \( S^{10} \cdot x^{10} = S^{11} \cdot x^{11} = 0 \), output FAIL
- For each \( i \), if there’s a \( b \in \{0,1\} \) with \( S^{1b} \cdot x^{1b} > 0 \), let \( T^i = h^{-1}(b) \), else \( T^i = [n] \)
- If \( \cap_i T^i = 1 \), output the item in \( \cap_i T^i \), else output FAIL

Outputting a Non-Zero Coordinate of a Vector

- Maintain \( S^{ab} \cdot x^{ab} \) for \( i \in [n] \) and \( b \in \{0,1\} \) in the stream
- Easy to maintain with positive and negative updates to coordinates
- \( O(\log^2 n) \) bits of space

- With probability \( 1 - 1/n \), if \( x \) is 1-sparse, correctly output single non-zero coordinate \( j \)
- Otherwise, correctly output FAIL

- Call this algorithm 1-Sparse-Finder

  - Can we use 1-Sparse-Finder to find a non-zero item of \( x \) if \( x \) is not 1-sparse?
Outputting a Non-Zero Coordinate of a k-Sparse Vector

- If $x$ is k-sparse, i.e., has k non-zero entries, use hashing
- Let $h$ be a 2-universal hash function from $[n]$ to $[10k]$

\[
\begin{array}{cccccccc}
\text{x}_1 & \text{x}_2 & \text{x}_3 & \text{x}_4 & \text{x}_5 & \text{x}_6 & \text{x}_7 & \text{x}_8 & \cdots & \text{x}_n \\
\end{array}
\]

- In the $j$-th hash bucket, run 1-Sparse Finder

---

k-Sparse Algorithm Analysis

- In each bucket, we find a non-zero entry $i$ or output FAIL, with probability $1 - 1/poly(n)$
  - What if all the non-zero items of $x$ collide in a bucket?
    - Consider a non-zero entry $i$ of $x$
    - Since $h$ is 2-universal, with probability at least $1 - k/(10k) = 9/10$, $h(i) \neq h(j)$ for all $j \neq i$ for which $x_j \neq 0$
    - With $9/10$ probability, we output a non-zero entry $i$ of $x$
    - We know when we fail to output a non-zero entry (except with probability $1/poly(n)$)

---

Reducing the Space

- Have a procedure which, if $x$ is k-sparse, either outputs a non-zero item $i$ of $x$ or says FAIL
- Output a non-zero item with probability at least $9/10$
- Use $O(k \log n)$ bits of space
- Good if $k$ is small, but how can we reduce the space for large $k$?

---

Subsampling

Uniformly sample the coordinates as nested subsets

$[n] = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_{\log_2 n}$

Include each item from $S_{i-1}$ in $S_i$ independently with probability $1/2$

$x_{S_i}$ is $x$ restricted to coordinates in $S_i$
Algorithm for Finding a Non-Zero Item

- If \( x \) has \( k \) non-zero entries, what's the expected number of non-zero entries in \( x_{S_{1}} \)?
  - For each non-zero entry \( j \), let \( Z_{j} = 1 \) if \( j \in S_{1} \), and \( Z_{j} = 0 \) otherwise
  - \( Z = \sum Z_{j} \), and so \( E[Z] = k \cdot E[Z_{1}] = \frac{k}{2} \)

- What's the variance of \( Z \)?
  - \( \text{Var}[Z] = \sum \text{Var}[Z_{j}] = k \cdot \text{Var}[Z_{1}] = k \left( \frac{1}{2} \right) \left( 1 - \frac{1}{2} \right) \leq \frac{k}{2} \)
  - If \( i = \lfloor \log_{2} k \rfloor - 5 \), then \( 32 \leq E[Z] < 64 \) and \( \text{Var}[Z] < 64 \)
  - By Chebyshev, \( \Pr[Z - E[Z] \geq 32] \leq \frac{\text{Var}[Z]}{32^2} \leq \frac{1}{16} \)
  - If we run a \( k' \)-sparse algorithm with \( k' = 96 \) on \( x_{S_{1}} \), we recover a non-zero item of \( x_{S_{1}} \) with probability at least \( 4/5 \), or output FAIL

- But we don't know \( i \)?

Outline

- Sketching Model
  - Estimating the Euclidean norm of a vector
  - Finding a non-zero coordinate of a vector

- Graph sketching motivation
  - Boruvka's spanning tree algorithm
  - Finding a spanning tree from a sketch

Algorithm for Finding a Non-Zero Item

- Run a \( k' = 96 \)-sparse vector algorithm on every \( x_{S_{1}} \)!

- For each \( x_{S_{1}} \), our algorithm either returns a non-zero item of \( x_{S_{1}} \), and hence of \( x \), or outputs FAIL

- For \( i = \lfloor \log_{2} k \rfloor - 5 \), with probability at least \( 4/5 \), we output a non-zero item of \( x_{S_{1}} \), and hence of \( x \)

- Space is \( (\log_{2} n) \cdot O(k' \log n) = O(\log^{3} n) \) bits!

Sketching Graphs

Are there sketches for graphs? Here \( A_{G} \) is the \( n \times n \) adjacency matrix

- \( (A_{G})_{ij} = 1 \) if \( (i,j) \) is an edge, and \( (A_{G})_{ij} = 0 \) otherwise

\[
\begin{pmatrix}
S \\
A_{G}
\end{pmatrix}
= 
\begin{pmatrix}
S A_{G} \\
\end{pmatrix}
\rightarrow \text{answer}
\]

- Is there a distribution on random matrices \( S \) with \( \text{poly}(\log n) \) rows so that you can output a spanning tree of \( G \) given \( SA_{G} \), with high probability?
Application: Graph Streams

• Want to process a graph stream, where you see the edges of a graph \( e_1, \ldots, e_m \) one at a time, in an arbitrary order. Assume the vertices are labeled 1, 2, ..., \( n \).

\[ e_3 \quad e_{10} \quad e_1 \quad e_2 \quad e_5 \quad e_7 \quad e_6 \quad \ldots \]

• Can only make 1 pass over the stream
• Trivially store stream using \( O(n^2) \) bits of memory.
• Want to instead use \( n \cdot \text{poly}(\log n) \) bits of memory

• How would you compute a spanning forest?

Computing a Spanning Forest

• For each edge \( e \) in the stream
  • If ____________________, store edge \( e \)
  • ____________________ is “doesn’t form a cycle”

  • Store at most \( n-1 \) edges, so \( O(n \log n) \) bits of memory

  • But what if you are allowed to delete edges? This is called a dynamic stream

Handling Deletions with Sketching

• Given \( S \cdot A_G \), replace it with \( S \cdot A_{G-e} = S \cdot A_G - S \cdot A_e \)

• Memory required to store \( S \cdot A_G \) is (# of rows of \( S \)) \cdot n \cdot \log n \) bits
  • Also need to store \( S \), which is (# of rows of \( S \)) \cdot n \cdot \log n \)

• Goal: find a distribution on matrices \( S \) with a small # of rows so that given \( S \cdot A_G \), can output a spanning tree of \( G \) with high probability

• Theorem: there is a distribution on \( S \) with only \( \text{poly}(\log n) \) rows!

Parallel Computing
Outline

• Sketching Model
  • Estimating the Euclidean norm of a vector
  • Finding a non-zero coordinate of a vector

• Graph sketching motivation
  • Boruvka’s spanning tree algorithm
  • Finding a spanning tree from a sketch

Boruvka’s Spanning Tree Algorithm (Modified)

• For simplicity, assume the input graph is connected

• Initialize edgeset $E'$ to $\emptyset$

• Create a list $L$ of $n$ groups of vertices, each initialized to a single vertex

• While the list has more than one group
  • For each group $G$, put an edge $e$ from a vertex in $G$ to a vertex not in $G$ into $E'$
  • Merge any groups connected by an edge in the previous step

• Find a spanning tree among the edges in $E'$

Input Graph

Groups at Beginning of Round 1

List of Groups

• A
• B
• C
• D
• E
• F
• G
• H
• I
• J
Round 1

Group C

Round 1

Edge C-F

Round 1

Group D

Round 1

Edge D-A
Round 1

Group I

Round 1

Edge I-G

Group J

Round 1

Edge J-H
Round 1 Ends

List of Edges Added

- A-D
- B-A
- C-F
- D-B
- E-C
- F-C
- G-E
- H-J

List of Groups

Groups at Beginning of Round 2

List of Groups

- D-A-B
- F-C-E-G-I
- H-J

Round 2

Group D-A-B

Round 2

Edge B-C
Round 2

Group F-C-E-G-I

Round 2

Edge I-J

Round 2

Group H-J

Round 2

Edge J-I
Analysis

- If $G_1, G_2, \ldots, G_r$ are the groups of vertices in an iteration, for each $G_i$, there is a $G_j$, $i \neq j$, and an edge $(u,v)$ from a vertex $u \in G_i$ to a vertex $v \in G_j$
  - Otherwise, graph is disconnected

- If $t$ groups at start of an iteration, at most $t/2$ groups at end of iteration
  - Consider graph $H$ with vertex set $G_1, G_2, \ldots, G_r$ and $r$ edges, where the edges correspond to the groups we connect
  - Number of groups at most number of connected components in $H$. Why?

- After $\log_2 n$ iterations, one group left
  - At most $n + n/2 + n/4 + \ldots + 1 \leq 2n$ edges chosen in $E'$

- $E'$ contains a spanning tree
  - Invariant: the vertices in each group in each iteration are connected

Outline

- Sketching Model
  - Estimating the Euclidean norm of a vector
  - Finding a non-zero coordinate of a vector

- Graph sketching motivation
  - Boruvka’s spanning tree algorithm
  - Finding a spanning tree from a sketch
Representing a Graph

• For node $i$, let $a_i$ be a vector indexed by node pairs

• If $\{i, j\}$ is an edge, $a_i[i, j] = 1$ if $j > i$, and $a_i[i, j] = -1$ if $j < i$

• If $\{i, j\}$ is not an edge, $a_i[i, j] = 0$

\[
a_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}
\]

\[
a_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\]

Spanning Tree Algorithm

• Compute $O(\log n)$ sketches $C_i(a_j)$, ..., $C_{O(\log n)}(a_j)$ for each $a_j$

• Each sketch $C_i(a_j)$ can output a non-zero item of $a_j$ with probability at least $4/5$, otherwise returns FAIL

• Idea: Run Boruvka’s algorithm in sketch space

• For each node $j$, use $C_j(a_j)$ to get incident edge on each node $j$

• For $i = 2$, ..., $O(\log n)$
  • To get incident edge on group $G \subseteq V$, use

\[
\sum_{j \in S} C_i a_j = C_i \left( \sum_{j \in S} a_j \right) \rightarrow e \in \text{support}(\sum_{j \in S} a_j) = E(S, V \setminus S)
\]

Spanning Tree Wrapup

• $O(n \log n)$ total sketches $C_i(a_j)$, as $i$ and $j$ vary, so $O(n \log^4 n)$ space

• Note: a $1/5$ fraction of sketches fail in each iteration in expectation, but Boruvka’s algorithm guarantees that on the remaining $4/5$ fraction of vertices, the number of connected components halves

• Expected number of iterations still $O(\log n)$

• Since sketches are linear, they can be maintained with insertions and deletions of edges in a stream

• Overall $O(n \log^4 n)$ bits of space to output a spanning tree!
  • Can be improved to $O(n \log^5 n)$ bits