Lectures 24 and 25: Sketching

David Woodruff
Carnegie Mellon University

Some slides from Jonathan Davis and Andrew McGregor
Outline

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

• Graph sketching
  • 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 $k \ll n$

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

• Matrix $S$ does not depend on $x$, e.g., $S$ could be a random matrix (require the entries of $S$ be $O(\log n)$ bits)
Application: Streams with Deletions

• n-dimensional vector \( x \) initialized to 0\(^n\)

• Stream of updates \( x_i \leftarrow x_i + \Delta_j \) for \( \Delta_j \) in \{-1,1\}

• One pass over the stream, have small memory

• Given \( S \cdot x \) and update \( x_i \leftarrow x_i + \Delta_j \), replace \( S \cdot x \) with \( S \cdot x + S \cdot e_i \cdot \Delta_j \)

• Memory to store \( S \cdot x \) is (# of rows of \( S \)) \cdot \log n \) bits
  • Also need to store \( S \), which typically can be stored implicitly
Outline

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

• Graph sketching
  • Boruvka’s spanning tree algorithm
  • Finding a spanning tree from a sketch
Example: Estimating the Norm of a Vector

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

\begin{align*}
\sum_{i: h(i) = 2} \sigma_i \cdot x_i
\end{align*}
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 $|Sx|^2$

\[
E[|Sx|^2] = E[\sum_j (\sum_i \delta(h(i) = j)\sigma(i) x_i)^2] \\
= \sum_j \sum_i x_i x_i E[\delta(h(i) = j)\delta(h(i') = j)\sigma(i)\sigma(i')]
= \sum_j \sum_{i,i'} x_i x_i E[\delta(h(i) = j)\delta(h(i') = j)] \cdot E[\sigma(i) \cdot \sigma(i')]
= \frac{\sum_j \sum_i x_i^2}{k} = |x|^2
\]
CountSketch

• In recitation, will show $\text{Var}[|Sx|^2] = O(|x|^4/k)$

• By Chebyshev’s inequality,

\[
\Pr \left[ \left| |Sx|^2 - |x|^2 \right| > \epsilon |x|^2 \right] \leq \frac{\text{Var}[|Sx|^2]}{\epsilon^2 |x|^4} \leq \frac{1}{10} \text{ provided } k = \Theta \left( \frac{1}{\epsilon^2} \right)
\]

• 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
  • Boruvka’s spanning tree algorithm
  • Finding a spanning tree from a sketch
A 1-Sparse Recovery Algorithm

• n-dimensional vector \( x \) initialized to \( 0^n \)

• Stream of updates \( x_i \leftarrow x_i + \Delta_j \) for \( \Delta_j \in \{-1,1\} \)
  
  • Promised 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 \( i \)

  • otherwise outputs \( \text{FAIL} \)
A 1-Sparse Recovery Algorithm

Let \( p = \text{poly}(n) \) be a random prime, and \( z \) a random integer mod \( p \)

Maintain \( A = \sum_i x_i \), \( B = \sum_i x_i \cdot i \), and \( C = \sum_i x_i \cdot z^i \mod p \)

If \( B/A \) is not in \( \{1, 2, \ldots, n\} \), output FAIL

Else if \( C = A \cdot z^{B/A} \mod p \), output \( B/A \). Otherwise output FAIL

Claim: If \( x \) is 1-sparse, we succeed. Why?

Claim: If \( x \) not 1-sparse, we output FAIL with probability \( 1-1/\text{poly}(n) \)

Proof: If \( B/A \) is not in \( \{1, 2, \ldots, n\} \), we output FAIL

Else, \( B/A \) is in \( \{1, 2, \ldots, n\} \), and let \( q(y) = \sum_i x_i y^i - A \cdot y^{B/A} \mod p \)

\( q(y) \) is a degree at most \( n \) polynomial, and so has at most \( n \) roots. The chance that \( z \) is one of them is at most \( n/p = 1/\text{poly}(n) \)
Outputting a Non-Zero Coordinate of a Vector

• Maintain $A = \sum_i x_i$, $B = \sum_i x_i \cdot i$, and $C = \sum_i x_i \cdot z^i \mod p$

• $O(\log n)$ bits of space

• If $x$ is 1-sparse, output single non-zero coordinate

• Otherwise, with probability $1-1/poly(n)$, 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 $[10k]$

\[
\begin{array}{cccccccc}
  x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & \ldots & x_n \\
\end{array}
\]

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

• In each bucket, find a non-zero entry i or output FAIL, with probability $1-1/poly(n)$

• What if all 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 $1-k/(10k) = 9/10$,
  \[ h(i) \notin \{h(j) | j \neq i \text{ and } x_j \neq 0 \} \]

• 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 an algorithm which, if $x$ is $k$-sparse, outputs a non-zero item or says FAIL

• Outputs 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 to reduce the space for large $k$?
Subsampling

Uniformly sample the coordinates as nested subsets

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

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 \( x_{S_i} \)?**
  - For each non-zero entry \( j \) in \( x \), let \( Z_j = 1 \) if \( j \in S_i \), and \( Z_j = 0 \) otherwise
  - \( Z = \sum_j Z_j \)
  - \( E[Z] = k \cdot E[Z_j] = \frac{k}{2^i} \)
  - \( \text{Var}[Z] = \sum_j \text{Var}[Z_j] = k \cdot \text{Var}[Z_1] = k \left( \frac{1}{2^i} \right) \left( 1 - \frac{1}{2^i} \right) \leq \frac{k}{2^i} \)
  - If \( i = \lceil \log_2 k \rceil - 5 \), then \( 32 \leq E[Z] < 64 \) and \( \text{Var}[Z] < 64 \)
  - By Chebyshev’s inequality, \( \Pr[|Z - E[Z]| \geq 32] \leq \frac{\text{Var}[Z]}{32^2} \leq \frac{1}{16} \)
  - If we run a \( k' \)-sparse algorithm with \( k' = 96 \) on \( x_{S_i} \), we recover a non-zero item of \( x_{S_i} \) with probability at least \( 1 - 1/16 - 1/10 > 4/5 \), or output FAIL
  - **But we don’t know \( i \)?**
Algorithm for Finding a Non-Zero Item

• Run a k’=96-sparse vector algorithm on every $x_{S_i}$.

• For each $x_{S_i}$, our algorithm either returns a non-zero item of $x_{S_i}$, and hence of $x$, or outputs FAIL.

• For $i = \lceil \log_2 k \rceil - 5$, with probability at least $4/5$, we output a non-zero item of $x_{S_i}$, and hence of $x$.

• Space is $(\log_2 n) \cdot O(k' \log n) = O(\log^2 n)$ bits!
Outline

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

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

Are there sketches for graphs? \( A_G \) is the \( n \times n \) adjacency matrix of a graph \( G \)

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

\[
\begin{pmatrix}
S \\
A_G
\end{pmatrix}
\begin{pmatrix}
\end{pmatrix}
= \begin{pmatrix}
SA_G \\
\end{pmatrix}
\rightarrow \text{answer}
\]

- Is there a distribution on matrices \( S \) with a small number of 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 we see the edges of a graph $e_1, ..., e_m$ in an arbitrary order. Assume the vertices are labeled 1, 2, ..., n.

- Make 1 pass over the stream
- Trivially store stream using $O(n^2)$ bits of memory.
- Want to 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 \( S \cdot A_G \), if \( e \) is deleted, replace it with \( S \cdot A_G - S \cdot A_e = S \cdot A_{G-e} \)

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

• 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 \( O(\log^2 n) \) rows!
Parallel Computing

Input: $G=(V,E)$

$G_1=(V,E_1)$

$G_2=(V,E_2)$

$G_3=(V,E_3)$

$G_4=(V,E_4)$

$SA_{G_1}$

$SA_{G_2}$

$SA_{G_3}$

$SA_{G_4}$

$SA_G = SA_{G_1} + SA_{G_2} + SA_{G_3} + SA_{G_4}$
Outline

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

- Graph sketching
  - Boruvka’s spanning tree algorithm
  - Finding a spanning tree from a sketch
Boruvka’s Spanning Tree Algorithm (Modified)

• Assume input graph is connected

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

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

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

• Find a spanning tree among the edges in $E'$
Groups at Beginning of Round 1

List of Groups

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

Group A
Round 1

Edge A-D
Round 1

Group B
Round 1

Edge B-A
Round 1

Group D
Round 1

Group E
Round 1

Edge E-C
Round 1

Group F
Round 1

Edge F-C
Round 1

Group G
Round 1

Edge G-E
Round 1

Group H

A B C D E F G H I J

D H J

H J
Round 1

Edge H-J
Round 1

Group I
Round 1

Edge I-G
Round 1

Group J
Round 1 Ends

List of Edges Added

- A-D
- B-A
- C-F
- D-B
- E-C
- F-C
- G-E
- H-J
- I-G
- J-H
Groups at Beginning of Round 2

List of Groups

- D-A-B
- F-C-E-G-I
- H-J
Round 2: Group F-C-E-G-I
Round 2

Edge I-J
Round 2

Group H-J
Round 2

Edge J-I
Round 2 Ends

List of Edges Added

- B-C
- I-J
- J-I
Analysis

• If $G_1, G_2, ..., G_r$ are groups of vertices in an iteration, for each $G_i$, there is a $G_j, j \neq i$, and an edge $\{u,v\}$ from a vertex $u \in G_i$ to a vertex $v \in G_j$
  • Else, 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 $G_1, G_2, ..., G_r$ and $r$ edges, where edges correspond to the groups we connect
  • Number of groups now at most number of connected components in $H$. Why?

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

• $E'$ contains a spanning tree
  • Invariant: the vertices in a group are connected
Outline

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

• Graph sketching
  • 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, \( a_i[i, j] = 0 \)

\[
\begin{align*}
a_1 &= (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0) \\
a_2 &= (-1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)
\end{align*}
\]
Representing a Graph

• **Lemma:** for a subset $S$ of nodes,
  \[ \text{Support}(\sum_{i \in S} a_i) = E(S, V \setminus S) \]

• **Proof:** for edge $\{i,j\}$, if $i, j \in S$, the sum of entries on $\{i,j\}$-th column is 0
Spanning Tree Algorithm

- Compute $O(\log n)$ sketches $C_1 \cdot a_j, \ldots, C_{O(\log n)} \cdot a_j$ for each $a_j$
- Each $C_i \cdot a_j$ outputs a non-zero item of $a_j$ with probability $> 4/5$, else returns FAIL
- **Idea**: Run Boruvka’s algorithm on sketches
- For each node $j$, use $C_i \cdot a_j$ to get incident edge on $j$
- For $i = 2, \ldots, O(\log n)$
  - To get incident edge on group $G \subseteq V$, use

$$\sum_{j \in S} C_i a_j = C_i \left( \sum_{j \in S} a_j \right) \rightarrow e \in \text{support} \left( \sum_{j \in S} a_j \right) = E(S, V \setminus S)$$
Spanning Tree Wrapup

• $O(n \log n)$ sketches $C_i \cdot a_j$, as $i$ and $j$ vary, so $O(n \log^3 n)$ space

• **Note:** a 1/5 fraction of sketches fail in each iteration in expectation, but on the remaining 4/5 fraction of vertices, the number of connected components halves

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

• Since sketches are linear, can maintain with insertions and deletions of edges

• Overall, $O(n \log^3 n)$ bits of space to output a spanning tree!