Outline

- Introduction to the Streaming Model
- Estimating Norms in the Streaming Model
Turnstile Streaming Model

- Underlying n-dimensional vector $x$ initialized to $0^n$

- Long stream of updates $x_i \leftarrow x_i + \Delta_j$ for $\Delta_j$ in $\{-M, -M+1, ..., M-1, M\}$
  - $M \leq \text{poly}(n)$

- Throughout the stream, $x$ is promised to be in $\{-M, -M+1, ..., M-1, M\}^n$

- Output an approximation to $f(x)$ with high probability over our coin tosses

- **Goal**: use as little space (in bits) as possible
  - Massive data: stock transactions, weather data, genomes
Testing if $x = 0^n$

- How can we test, with probability at least $9/10$, over our random coin tosses, if the underlying vector $x = 0^n$?

- Can we use $O(\log n)$ bits of space?

- We saw that for any fixed vector $x$, if $S$ is a CountSketch matrix with $O\left(\frac{1}{\epsilon^2}\right)$ rows, then $|Sx|_2^2 = (1 \pm \epsilon)|x|_2^2$ with probability at least $9/10$

- If we set $\epsilon = \frac{1}{2}$, we use $O(\log n)$ bits of space to store the $O(1)$ entries of $Sx$

- We can store the hash function and sign function defining $S$ using $O(\log n)$ bits
Testing if $x = 0^n$

- Is there a deterministic, i.e., zero-error, streaming algorithm to test if the underlying vector $x = 0^n$ with $o(n \log n)$ bits of space?

- **Theorem:** any deterministic algorithm requires $\Omega(n \log n)$ bits of space

- Suppose the first half of the stream corresponds to updates to a vector $a$ in $\{0, 1, 2, \ldots, \text{poly}(n)\}^n$

- Let $S(a)$ be the state of the algorithm after reading the first half of the stream
  - If $|S(a)| = o(n \log n)$, there exist $a \neq a'$ for which $S(a) = S(a')$

- Suppose the second half of the stream corresponds to updates to a vector $b$ in $\{0, -1, -2, \ldots, -\text{poly}(n)\}^n$

- The algorithm must output the same answer on $a+b$ and $a'+b$, so it errs in one case
Example: Recovering a k-Sparse Vector

• Suppose we are promised that $x$ has at most $k$ non-zero entries at the end of the stream

• $k$ is often small – maybe we see all coordinates of a vector $a$ followed by all coordinates of a *similar* vector $b$, and $a-b$ only has $k$ non-zero entries

• Can we recover the indices and values of the $k$ non-zero entries with high probability?

• Can we use $k \text{ poly}(\log n)$ bits of space?

• Can we do it deterministically?
Example: Recovering a k-Sparse Vector

• Suppose $A$ is an $s \times n$ matrix such that any $2k$ columns are linearly independent

• Maintain $A \cdot x$ in the stream

• Claim: from $A \cdot x$ you can recover the subset $S$ of $k$ non-zero entries and their values

• Proof: suppose there were vectors $x$ and $y$ each with at most $k$ non-zero entries and $A \cdot x = A \cdot y$

• Then $A(x-y) = 0$. But $x-y$ has at most $2k$ non-zero entries, and any $2k$ columns of $A$ are linearly independent. So $x-y = 0$, i.e., $x = y$.

• Algorithm is deterministic given $A$. But do such matrices $A$ exist with a small number $s$ of rows?
Example: Recovering a k-Sparse Vector

- Vandermonde matrix $A$ with $s = 2k$ rows and $n$ columns. $A_{i,j} = j^{i-1}$

$$
\begin{bmatrix}
1 & 1 & 1 & \ldots \\
1 & 2 & 3 & \ldots \\
1 & 4 & 9 & \ldots \\
1 & 8 & 27 & \ldots \\
\end{bmatrix}
$$

- Determinant of 2k x 2k submatrix of $A$ with set of columns equal to $\{i_1, \ldots, i_{2k}\}$ is: $\prod_j i_j \prod_{j < j'} (i_j - i_{j'}) \neq 0$, so any 2k columns of $A$ are linearly independent.

- But entries of $A$ are exponentially increasing – how to store $A$ and $A \cdot x$?

- Just store $A \cdot x \mod p$ for a large enough prime $p = \text{poly}(n)$.
Outline

• Quick recap of $\ell_1$-regression, and how to speed it up

• Introduction to the Streaming Model

• Estimating Norms in the Streaming Model
Example Problem: Norms

• Suppose you want $|x|_p^p = \sum_{i=1}^n |x_i|^p$

• Want $Z$ for which $(1-\epsilon) |x|_p^p \leq Z \leq (1+\epsilon) |x|_p^p$ with probability $> 9/10$

• $p = 1$ corresponds to total variation distance between distributions

• $p = 2$ useful for geometric and linear algebraic problems

• $p = \infty$ is the value of the maximum entry, useful for anomaly detection, etc.
Example Problem: Euclidean Norm

- Want \( Z \) for which \((1-\epsilon) \|x\|_2^2 \leq Z \leq (1+\epsilon) \|x\|_2^2\)

- Sample a random CountSketch matrix \( S \) with \(1/\epsilon^2\) rows

- Can store \( S \) efficiently using limited independence

- If \( x_i \leftarrow x_i + \Delta_i \) in the stream, then \( Sx \leftarrow Sx + \Delta_i S_{*,i} \)

- At end of stream, output \( \|Sx\|_2^2 \)

- With probability at least 9/10, \( \|Sx\|_2^2 = (1 \pm \epsilon)\|x\|_2^2 \)

- Space complexity is \(1/\epsilon^2\) words, each word is \(O(\log n)\) bits
Example Problem: 1-Norm

• Want Z for which \((1-\varepsilon) \|x\|_1 \leq Z \leq (1+\varepsilon) \|x\|_1\)

• Sample a random Cauchy matrix S?

• Can store S with \(\frac{1}{\varepsilon}\) words of space [Kane, Nelson, W]

• If \(x_i \leftarrow x_i + \Delta_i\) in the stream, then \(Sx \leftarrow Sx + \Delta_i S_{*,i}\)

• Space complexity is \(1/\varepsilon^2\) words, each word is \(O(\log n)\) bits

• At end of stream, output \(\|Sx\|_1\) ?

• *Cauchy random variables have no concentration...*
1-Norm Estimator

- Probability density function \( f(x) \) of \(|C|\) for a Cauchy random variable \( C \) is 
  \[
  f(x) = \frac{2}{\pi(1+x^2)}
  \]

- Cumulative distribution function \( F(z) \):
  \[
  F(z) = \int_{0}^{z} f(x)dx = \frac{2}{\pi} \arctan(z)
  \]

- Since \( \tan(\pi/4) = 1 \), \( F(1) = \frac{1}{2} \), so median(\(|C|\)) = 1

- If you take \( r = \frac{\log(\frac{1}{\delta})}{\epsilon^2} \) independent samples \( X_1, ..., X_r \) from \( F \), and \( X = \text{median}_i X_i \), then \( F(X) \) in \([1/2-\epsilon, 1/2+\epsilon]\) with probability \( 1-\delta \)

- \( F^{-1}(X) = \tan\left(\frac{X\pi}{2}\right) \in [1 - 4\epsilon, 1 + 4\epsilon] \)
p-Norm Estimator

• Can achieve $1/\epsilon^2$ words of space for p-norm estimation for any $0 < p < 2$

• Proof is similar to 1-norm estimation, and uses p-stable distributions, which exist only for $0 < p < 2$

• No closed form expression for their probability density function but they are efficiently sampleable:

  • If $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $r \in [0,1]$ are uniformly random, then

    \[
    \sin(p \theta) \left(\frac{\cos(\theta(1-p))}{\ln\left(\frac{1}{r}\right)}\right)^{\frac{1-p}{p}}
    \]

    is a sample from a p-stable distribution!

• Can discretize them and store a sketching matrix of samples from the p-stable distribution using limited independence
p-Norm Estimator for $p > 2$

- For $p > 2$, p-stable distributions do not exist!

- We will see later that $\Omega(n^{1-2/p})$ bits of space needed to approximate $p$-norms, $p > 2$, up to a constant factor with constant probability

- To achieve an $\widetilde{O}(n^{1-2/p})$ bits of space algorithm, we will use exponential random variables. We will focus on constant approximation parameter $\epsilon$

- Our sketch will be $P \cdot D$:

$$
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
1/E_1^{1/p} \\
1/E_2^{1/p} \\
\vdots \\
1/E_n^{1/p}
\end{bmatrix}
$$
Stability of Exponential Random Variables

- Exponential random variable $E$ with parameter $\lambda$
  - (PDF) probability density function: $f(x) = \lambda e^{-\lambda x}$ if $x \geq 0$, and 0 otherwise
  - (CDF) cumulative density function: $F(x) = 1 - e^{-\lambda x}$ for $x \geq 0$
- $t \cdot E$ for scalar $t \geq 0$ has CDF $F_x = 1 - e^{-\frac{\lambda}{t}x}$

- Stability: consider independent exponential random variables $E_1, \ldots, E_n$ and scalars $|y_1|, \ldots, |y_n|$, let $q = \min(\frac{E_1}{|y_1|^p}, \ldots, \frac{E_n}{|y_n|^p})$

- $\Pr[q > x] = \Pr\left[\forall i, \frac{E_i}{|y_i|^p} \geq x\right] = \prod_i e^{-x|y_i|^p} = e^{-x|y|^p}$

- So $q$ is an exponential random variable with $\lambda = |y|^p$, that is,

  $q \equiv \left(\frac{1}{|y|^p}\right)E$ for a standard exponential random variable $E$
Stability of Exponential Random Variables

- Recall our sketch $P \cdot D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/E_1^{1/p} \\ 1/E_2^{1/p} \\ \vdots \\ 1/E_n^{1/p} \end{bmatrix}$

- What does $|Dy|_\infty$ look like for an arbitrary $y$?

- $|Dy|_\infty^p = \max_i \left( \frac{|y_i|^p}{E_i} \right) = \frac{1}{\min_i \frac{E_i}{|y_i|^p}} \equiv \frac{1}{\frac{1}{E} \cdot \frac{|y|^p}{p}} = \frac{|y|^p}{p}$

- $\Pr[E \in \left[ \frac{1}{10}, 10 \right]] = (1 - e^{-10}) - (1 - e^{-\frac{1}{10}}) = e^{-\frac{1}{10}} - e^{-10} > \frac{4}{5}$
Stability of Exponential Random Variables

- We know $|Dy|_\infty \in \left[ \frac{|y|_p}{10^{1/p}}, 10^{1/p} |y|_p \right]$ with probability at least $\frac{4}{5}$.

- So $|Dy|_\infty$ is a good estimate of $|y|_p$, but $Dy$ is an n-dimensional vector!

- Recall our sketch $P*D =$

$\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} \cdot \begin{bmatrix}
1/E_1^{1/p} \\
1/E_2^{1/p} \\
\vdots \\
1/E_n^{1/p}
\end{bmatrix}$

- What can we say about $|PDy|_\infty$ if $P$ has $s$ rows?

- Intuitively $P$ is hashing coordinates of $Dy$ into buckets and taking a signed sum of the entries. Expect everything to cancel out and $|PDy|_\infty \approx |Dy|_\infty \smile$
Understanding $|PDy|_\infty$

- Let $s$ be the number of rows of $P$, which we can think of as hash buckets.

- $P$ is a CountSketch matrix with hash functions $h$ and $\sigma$:
  - $h: [n] \rightarrow [s]$,
  - $\sigma: [n] \rightarrow \{-1, 1\}$
  - Let’s assume $h$ and $\sigma$ are truly random (can be derandomized).

- We know $|Dy|_\infty \in \left[ \frac{|y|_p}{10^{1/p}}, 10^{1/p} |y|_p \right]$ with probability at least $4/5$.

- To achieve $|PDy|_\infty \approx |Dy|_\infty$ with good probability, we want:
  - (1) in each bucket $i$ not containing the coordinate $j$ for which $|(Dy)_j| = |Dy|_\infty$, we have $|(PDy)_i| \leq \frac{|y|_p}{100}$
  - (2) in the bucket $i$ containing the coordinate $j$ for which $|(Dy)_j| = |Dy|_\infty$, we have $|||(PDy)_i| - |Dy|_\infty| \leq |y|_p/100$
Analyzing $|PDy|_\infty$

- Let $\delta(E) = 1$ if event $E$ holds, and $\delta(E) = 0$ otherwise.
- What does the $i$-th bucket value $(PDy)_i$ look like?
- $(PDy)_i = \sum_j \delta(h(j) = i) \sigma_j(Dy)_j$
- $E[(PDy)_i] = 0$
- What about the variance of $(PDy)_i$?
Understanding $|PDy|_\infty$

- $E_P[(PDy)_i^2] = \sum_{i,j} E[\delta(h(j) = i)\delta(h(j') = i)\sigma_j \sigma_{j'}](Dy)_j(Dy)_{j'} = \left(\frac{1}{s}\right) |Dy|_2^2$

- $E_D[|Dy|_2^2] = \sum_i y_i^2 \cdot E[D_{i,i}^2]$

- $E[D_{i,i}^2] = \int_{t\geq0} t^{-2/p} e^{-t} \, dt$
  
  $= \int_{t\in[0,1]} t^{-2/p} e^{-t} \, dt + \int_{t>1} t^{-2/p} e^{-t} \, dt$

  $\leq \int_{t\in[0,1]} t^{-2/p} \, dt + \int_{t>1} e^{-t} \, dt$

  $= \left(\frac{1}{1 - \frac{2}{p}}\right) \cdot t^{1-2/p}\big|_0^1 - e^{-t}\big|_1^\infty$

  $= O(1)$

- So, $E[(PDy)_i^2] = O\left(\frac{1}{s}\right) |y|_2^2 = O\left(\frac{1}{s}\right) (n^{1-2/p} |y|_p^2)$. Why?
Understanding $|PDy|_\infty$

- $E[(PDy)_i] = 0$ for each hash bucket $i$, and $E \left[ (PDy)_i^2 \right] = O \left( \frac{1}{s} \right) \left( n^{1-\frac{2}{p}} |y|_p^2 \right)$

- Bernstein’s bound: Suppose $R_1, \ldots, R_n$ are independent, and for all $j$, $|R_j| \leq K$, and $\text{Var} \left[ \sum_j R_j \right] = \sigma^2$. There are constants $C, c$, so that for all $t > 0$,
  \[
  \Pr \left[ \left| \sum_j R_j - E \left[ \sum_j R_j \right] \right| > t \right] \leq C \left( e^{-\frac{ct^2}{\sigma^2}} + e^{-\frac{ct}{K}} \right)
  \]

- Recall $(PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$, and set $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$

- Want $|PDy|_\infty \approx |Dy|_\infty$, where $|Dy|_\infty \in \left[ \frac{|y|_p}{10^{1/p}}, 10^{1/p} |y|_p \right]$ with probability $> 4/5$

- Set $t = \frac{|y|_p}{100}$ and $s = \Theta(n^{1-\frac{2}{p}} \log n)$, to get $\frac{1}{n^2}$ error probability in Bernstein’s bound

- But what is $K = \max_j |R_j|$?
Understanding the Large Elements

- Recall \( (PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j \), and set \( R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j \)

- We will separately handle those \( R_j \) for which \(|R_j| > \frac{\alpha|y|_p}{\log n}\), for a sufficiently small constant \( \alpha > 0 \). If \(|R_j| > \frac{\alpha|y|_p}{\log n}\), then necessarily \(|(Dy)_j| \geq \frac{\alpha|y|_p}{\log n}\)
  - We call such a \( j \) large if \(|(Dy)_j| \geq \frac{\alpha|y|_p}{\log n}\), otherwise \( j \) is small. How many indices \( j \) are large?

- Recall: \(|(Dy)_j| = |y_j|/E_j^{1/p}\)

- \( \Pr[|(Dy)_j| \text{ is large}] = \Pr \left[ \frac{|y_j|}{E_j^{1/p}} \geq \frac{\alpha|y|_p}{\log n} \right] = \Pr \left[ \frac{|y_j|^p}{\alpha^p|y|_p} \left( \log^p n \right) \geq E_j \right] \)

\[
= 1 - e^{-\frac{|y_j|^p \left( \log^p n \right)}{\alpha^p|y|_p}} \leq \frac{|y_j|^p \left( \log^p n \right)}{\alpha^p|y|_p},
\]
so the expected number of large \( j \) is \( O(\log^p n) \)
Understanding the Large Elements

- Recall \( (PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j \), and set \( R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j \)

- We have shown the expected number of large \( j \) is \( O(\log^p n) \), so by a Markov bound we have \( O(\log^p n) \) large \( j \) with constant probability and we condition on \( D \) satisfying this

- We also condition on \( |Dy|_\infty \in \left[ \frac{|y|_p}{10^p}, \frac{1}{10^p} |y|_p \right] \), which held with probability \( > \frac{4}{5} \)

- All the large \( j \) get perfectly hashed into separate hash buckets by \( P \)
  - We are throwing \( O(\log^p n) \) balls into \( s \geq n^{1-2/p} \) bins

- We apply Bernstein for each hash bucket separately
  - We apply Bernstein on the small indices \( j \) inside a hash bucket!
Understanding the Large Elements

- $E[(PDy)_i] = 0$ for each hash bucket $i$, and $E \left[ (PDy)_i^2 \right] = O \left( \frac{1}{s} \right) \left( n^{1-\frac{2}{p}} |y|^2 \right)$

- Bernstein’s bound: Suppose $R_1, \ldots, R_n$ are independent, and for all $j$, $|R_j| \leq K$, and $\text{Var}[\sum R_j] = \sigma^2$. There are constants $C, c$, so that for all $t > 0$,
  - $\Pr[|\sum R_j - E[\sum R_j]| > t \leq C \left( e^{-\frac{ct^2}{\sigma^2}} + e^{-\frac{ct}{K}} \right)$

- $(PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$, and $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$

- Can assume $K = \max_j |R_j| \leq \frac{\alpha |y|_p}{\log n}$, since there is at most one large $j$ in any hash bucket $(PDy)_i$

- Set $t = \frac{|y|_p}{100}$, and $s = \Theta(n^{1-\frac{2}{p}} \log n)$ in Bernstein’s bound, to get for a bucket $(PDy)_i$:
  $$\Pr \left[ \left| \sum_{\text{small } j} \delta(h(j) = i) \sigma_j (Dy)_j \right| > \frac{|y|_p}{100} \right] \leq C \left( e^{-\theta(\log n)} + e^{-\frac{c(\log n)}{100\alpha}} \right) \leq \frac{1}{n^2}$$

- By a union bound over all the $s$ buckets, the “signed sum” of small $j$ in every bucket will be at most $\frac{|y|_p}{100}$
Wrapping Up

- For all $i$,
  - $|(PDy)_i| \leq \frac{|y|_p}{100}$ if no large indices in $i$-th bucket
  - $|(PDy)_i| = |\sigma_j(Dy)_j| \pm \frac{|y|_p}{100}$ if exactly one large index $j$ in $i$-th bucket
  - No bucket contains more than 1 large index $j$

- We conditioned on $|Dy|_\infty \in \left[\frac{|y|_p}{10^p}, \frac{1}{10p}|y|_p\right]$

- What is $|PDy|_\infty$?
  - $|PDy|_\infty \leq 10^p|y|_p + \frac{|y|_p}{100}$ and $|PDy|_\infty \geq \frac{|y|_p}{10^p} - \frac{|y|_p}{100}$

- So just output $|PDy|_\infty$ as your estimate to $|y|_p$

- Total space is $s = O(n^{1-\frac{2}{p}} \log n)$ words, which is $O(n^{1-\frac{2}{p}} \log^2 n)$ bits
Outline

- Quick recap of $\ell_1$-regression, and how to speed it up
- Introduction to the Streaming Model
- Estimating Norms in the Streaming Model
- Heavy Hitters in a Stream
- Estimating Number of Non-Zero Entries ($\ell_0$)
Heavy Hitter Guarantees

- **$l_1$ – guarantee**
  - output a set containing all items $j$ for which $|x_j| \geq \phi |x|_1$
  - the set should not contain any $j$ with $|x_j| \leq (\phi - \epsilon) |x|_1$

- **$l_2$ – guarantee**
  - output a set containing all items $j$ for which $x_j^2 \geq \phi |x|_2$
  - the set should not contain any $j$ with $x_j^2 \leq (\phi - \epsilon)|x|_2$

- **$l_2$ – guarantee can be much stronger than the $l_1$ – guarantee**
  - Suppose $x = (\sqrt{n}, 1, 1, 1, \ldots, 1)$
  - Item 1 is an $l_2$-heavy hitter for constant $\phi$, $\epsilon$, but not an $l_1$-heavy hitter
  - If $|x_j| \geq \phi |x|_1$, then $x_j^2 \geq \phi^2 |x|_1^2 \geq \phi^2 |x|_2^2$
Heavy Hitter Intuition

- Suppose you are promised at the end of the stream, \( x_i = n \), and \( x_j \in \{0,1\} \) for \( j \in \{1, 2, \ldots, n\} \) with \( j \neq i \)

- How would you find the identity \( i \)?

- For each \( j \) in \( \{1, 2, 3, \ldots, \log n\} \), let \( A_j \subset [n] \) be the set of indices with \( j \)-th bit in their binary representation equal to 0, and \( B_j \) be the set with \( j \)-th bit equal to 1

- Compute \( a_j = \sum_{i \in A_j} x_i \) and \( b_j = \sum_{i \in B_j} x_i \) for each \( j \) in \( \{1, 2, \ldots, \log n\} \)

- Read off the identity of item \( i \)
Suppose you are promised at the end of the stream, $x_i = 100 \sqrt{n \log(n)}$, and $x_j \in \{0,1\}$ for $j \in \{1, 2, \ldots, n\}$ with $j \neq i$

How would you find the identity $i$?

For each $j$ in $\{1, 2, 3, \ldots, \log n\}$, let $A_j \subset [n]$ be the set of indices with $j$-th bit in their binary representation equal to 0, and $B_j$ be the set with $j$-th bit equal to 1

Compute $a_j = \sum_{i \in A_j} \sigma_i \cdot x_i$ and $b_j = \sum_{i \in B_j} \sigma_i \cdot x_i$ for each $j$ in $\{1, 2, \ldots, \log n\}$

Read off the identity of item $i$?

Additive Chernoff bound implies magnitude of “noise” in a count is at most $\sqrt{n \log(n)}$ w.h.p.

Remove assumptions: (1) $x_i = 100 \sqrt{n \log(n)}$ and (2) and $x_j \in \{0,1\}$ for $j \in \{1, 2, \ldots, n\}$ with $j \neq i^\circ$
CountSketch achieves the $l_2$–guarantee

• Assign each coordinate $i$ a random sign $\sigma_i \in \{-1, 1\}$

• Randomly partition coordinates into $B$ buckets, maintain $c_j = \sum_{i: h(i) = j} x_i \cdot \sigma_i$ in the $j$-th bucket

• Estimate $x_i$ as $\sigma_i \cdot c_{h(i)}$
Why Does CountSketch Work?

• $E[\sigma_i c_{h(i)}] = \sigma_i \sum_{i' : h(i) = h(i')} \sigma_{i'} x_{i'} = x_i$

• Suppose we independently repeat this hashing scheme $O(\log n)$ times

• Output the median of the estimates across the $\log n$ repetitions

• “Noise” in a bucket is $\sigma_i \cdot \sum_{i' \neq i, h(i') = h(i)} \sigma_{i'} \cdot x_{i'}$

• What is the variance of the noise?

• $E\left[\left(\sigma_i \cdot \sum_{i' \neq i, h(i') = h(i)} \sigma_{i'} \cdot x_{i'}\right)^2\right] = \frac{|x|^2}{B}$

• So with constant probability, the noise in a bucket is $O\left(\frac{|x|^2}{\sqrt{B}}\right)$ in magnitude

• Since the $\log n$ repetitions are independent, this ensures that our estimate $\sigma_i c_{h(i)}$ will equal $x_i \pm O\left(\frac{|x|^2}{\sqrt{B}}\right)$ with probability $1-1/poly(n)$

• Hence, we approximate every $x_i$ simultaneously up to additive error $O\left(\frac{|x|^2}{\sqrt{B}}\right)$
Tail Guarantee

- CountSketch approximates every $x_i$ simultaneously up to additive error $O\left(\frac{|x|_2}{\sqrt{B}}\right)$

- But what if $x_1$ is a super large poly(n), and $x_2 = n$ and $x_3 = \ldots = x_n = 1$?

- We get a pretty bad approximation to $x_2$

- **Tail Guarantee**: CountSketch approximates every $x_i$ simultaneously up to additive error $O\left(\frac{|x_{-B/4}|_2}{\sqrt{B}}\right)$, where $x_{-B/4}$ is $x$ after zero-ing out its top $B/4$ coordinates in magnitude

- Proof: with probability at least $3/4$, in each repetition the top $B/4$ coordinates of $x$ in magnitude do not land in the same hash bucket as $x_i$
  - Do we need a lot of independence for this?

- What happens if $x$ is $B/4$-sparse?
How to Find the Top k Heavy Hitters Quickly

• There are $2^i$ nodes in i-th level of tree
  • Start at the level with $2k$ nodes

• Each node corresponds to a subset of $[n]$ of size $n/2^i$ with the same i-bit prefix

• In i-th level, for each i, hash to $O(k)$ buckets repeat $O(\log k)$ times. Like CountSketch, but in each bucket we run an approximation algorithm to the 2-norm

• In top level our universe has only $2k$ nodes, so we find top k just by computing estimate for all of them

**Main idea:** in next level, we only need to consider the left and right child of each of the k nodes we found at the previous level. So only $2k \ll n$ nodes to consider.
Why Care About the $\ell_1$-Guarantee?

• $\ell_1$ – guarantee
  • output a set containing all items $j$ for which $|x_j| \geq \phi|x|_1$
  • the set should not contain any $j$ with $|x|_j \leq (\phi-\epsilon)|x|_1$

• $\ell_2$ – guarantee
  • output a set containing all items $j$ for which $x_j^2 \geq \phi|x|_2^2$
  • the set should not contain any $j$ with $x_j^2 \leq (\phi - \epsilon)|x|_2^2$

• $\ell_2$ – guarantee implies the $\ell_1$ – guarantee

• So why care about the $\ell_1$ – guarantee?

• A nice thing about the $\ell_1$-guarantee is that it can be solved deterministically!
Deterministic $\ell_1$ Heavy Hitters

- An $s \times n$ matrix $S$ is $\epsilon$-incoherent if
  - for all columns $S_i$, $|S_i|_2 = 1$
  - for all pairs of columns $S_i$ and $S_j$, $|\langle S_i, S_j \rangle| \leq \epsilon$
  - entries can be specified with $O(\log n)$ bits of space

- Compute $S \cdot x$ in a stream using $O(s \log n)$ bits of space

- Estimate $\hat{x}_i = S_i^T S x$
  - $\hat{x}_i = \sum_{j=1, \ldots, n} \langle S_i, S_j \rangle x_j = |S_i|_2^2 x_i \pm \max_{i,j} |\langle S_i, S_j \rangle| |x|_1 = x_i \pm \epsilon |x|_1$
  - Can figure out which $|x_i| \geq \phi |x|_1$ and which $|x_i| \leq (\phi - \epsilon) |x|_1$

- But do $\epsilon$-incoherent matrices exist?
\(\epsilon\)-Incoherent Matrices

- Consider a prime \(q = \Theta((\log n)/\epsilon)\). Let \(d = \epsilon \cdot q = O(\log n)\)

- Consider \(n\) distinct non-zero polynomials \(p_1, \ldots, p_n\) each of degree less than \(d\).
  - \(q^d - 1 > n\)

- Associate \(p_i\) with the \(i\)-th column of \(S\)

- Let \(s = q^2\) and group the rows of \(S\) into \(q\) groups of size \(q\)
  - In \(j\)-th group, the \(i\)-th column has a single non-zero on the \(p_i(j)\)-th entry
  - \(p_{i(j)}\)-th entry is equal to \(1/q^{1/2}\)

- Each column \(S_i\) has \(|S_i|_2 = 1\)
- \(S_i\) and \(S_j\) each have the same non-zero in the \(k\)-th group iff \(p_i(k) = p_j(k)\)
- Number of such groups \(k\) is at most \(d \leq \epsilon q\), so \(|\langle S_i, S_j \rangle| \leq \epsilon\)
Outline

• Quick recap of $\ell_1$-regression, and how to speed it up

• Introduction to the Streaming Model

• Estimating Norms in the Streaming Model

• Heavy Hitters in a Stream

• Estimating Number of Non-Zero Entries ($\ell_0$)