Linear Sketches
– A Useful Tool in Streaming and Compressive Sensing

Qin Zhang
Random linear projection $M : \mathbb{R}^n \rightarrow \mathbb{R}^k$ that preserves properties of any $v \in \mathbb{R}^n$ with high prob. where $k \ll n$.

\[
\begin{bmatrix}
M \\
v
\end{bmatrix}
\begin{bmatrix}
Mv
\end{bmatrix} \rightarrow \text{answer}
\]
Random linear projection $M : R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high prob. where $k \ll n$.

\[
\begin{pmatrix}
M
\end{pmatrix}
\begin{pmatrix}
v
\end{pmatrix} = \begin{pmatrix}
Mv
\end{pmatrix} \rightarrow\text{answer}
\]

Simple and useful: Statistics/graph/algebraic problems in data streams, compressive sensing, ...
Random linear projection $M : R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high prob. where $k \ll n$. 

\[
\begin{bmatrix}
M \\
\end{bmatrix}
\begin{bmatrix}
v \\
\end{bmatrix}
= \begin{bmatrix}
Mv \\
\end{bmatrix} \rightarrow \text{answer}
\]

Simple and useful: Statistics/graph/algebraic problems in data streams, compressive sensing, \ldots

And rich in theory! You will see in this course.
Data streams

- **The model** *(Alon, Matias and Szegedy 1996)*
Data streams

- **The model** (Alon, Matias and Szegedy 1996)

- **Applications**

etc.
Data streams

- **The model** (Alon, Matias and Szegedy 1996)

- **Applications**

- **A list of theoretical problems**

  etc.
Why hard?

- **Game 1**: A sequence of numbers
Why hard?

- **Game 1**: A sequence of numbers

  ![Image of a number](image-url)
Why hard?

- **Game 1**: A sequence of numbers

45
Why hard?

- **Game 1**: A sequence of numbers

  18
Why hard?

- **Game 1**: A sequence of numbers

![Image of number 23]
Why hard?

- **Game 1:** A sequence of numbers

  17
Why hard?

- **Game 1**: A sequence of numbers

  41
Why hard?

- **Game 1**: A sequence of numbers

33
Why hard?

- **Game 1**: A sequence of numbers

  29
Why hard?

- **Game 1**: A sequence of numbers
Why hard?

- **Game 1:** A sequence of numbers

  12
Why hard?

- **Game 1**: A sequence of numbers

  35
Why hard?

- **Game 1**: A sequence of numbers

  **Q**: What’s the **median**?
Why hard?

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33
Why hard?

- **Game 1:** A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul
Why hard?

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Bob become friends
Why hard?

- **Game 1**: A sequence of numbers

  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Carol and Eva become friends
Game 1: A sequence of numbers

Q: What’s the median?

A: 33

Game 2: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

Eva and Bob become friends
Why hard?

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Paul become friends
Why hard?

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Paul become friends
Why hard?

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Eva and Bob unfriends
Why hard?

- **Game 1:** A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Dave become friends
Why hard?

- **Game 1:** A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Bob and Paul become friends
Why hard?

- **Game 1**: A sequence of numbers
  
  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul
  
  Dave and Paul unfriends
Why hard?

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Carol become friends
Game 1: A sequence of numbers

Q: What’s the median?

A: 33

Game 2: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

Q: Are Eva and Bob connected by friends?
Game 1: A sequence of numbers

Q: What’s the median?

A: 33

Game 2: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

Q: Are Eva and Bob connected by friends?

A: YES. Eva ↔ Carol ↔ Dave ↔ Alice ↔ Bob
Why hard?

- **Game 1**: A sequence of numbers
  
  **Q**: What’s the **median**?
  
  **A**: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul
  
  **Q**: Are Eva and Bob connected by friends?
  
  **A**: YES. Eva ⇔ Carol ⇔ Dave ⇔ Alice ⇔ Bob

- **Why hard?** Short of memory!
A simple example: distinct elements

- The problem

Q: Why linear sketch can be maintained in the streaming model?
A simple example: distinct elements

The problem

How many distinct elements?
Approximation needed.
A simple example: distinct elements

- **The problem**

  How many distinct elements?  
  Approximation needed.

- **Search version ⇒ Decision version**

  Let $D$ be $\#$ distinct elements:
  - If $D \geq T(1 + \epsilon)$, then answer YES.
  - If $D \leq T/(1 + \epsilon)$, then answer NO.

  Try $T = 1, (1 + \epsilon), (1 + \epsilon)^2, \ldots$
The algorithm

1. Select a random set \( S \subseteq \{1, 2, \ldots, n\} \), s.t. for each \( i \), independently, we have \( \Pr[i \in S] = 1/T \)

2. Make a pass over the stream, maintaining \( \text{Sum}_S(x) = \sum_{i \in S} x_i \)
   
   Note: this is a linear sketch.

3. If \( \text{Sum}_S(x) > 0 \), return YES, otherwise return NO.
Now, the decision problem

The algorithm

1. Select a random set $S \subseteq \{1, 2, \ldots, n\}$, s.t. for each $i$, independently, we have $\Pr[i \in S] = 1/T$

2. Make a pass over the stream, maintaining $Sum_{S}(x) = \sum_{i \in S} x_i$
   Note: this is a linear sketch.

3. If $Sum_{S}(x) > 0$, return YES, otherwise return NO.

Lemma

Let $P = \Pr[Sum_{S}(x) = 0]$. If $T$ is large enough, and $\epsilon$ is small enough, then

- If $D \geq T(1 + \epsilon)$, then $P < 1/e − \epsilon/3$.
- If $D \leq T/(1 + \epsilon)$, then $P > 1/e + \epsilon/3$.

(Introduce a few useful probabilistic basics)
Amplify the success probability

Repeat to amplify the success probability

1. Select $k$ sets $S_1, \ldots, S_k$ as in previous algorithm, for $k = C \log(1/\delta)/\epsilon^2$, $C > 0$

2. Let $Z$ be the number of values of $\text{Sum}_{S_j}(x)$ that are equal to 0, $j = 1, \ldots, k$.

3. If $Z < k/e$ then report YES, otherwise report NO.
Amplify the success probability

**Repeat** to amplify the success probability

1. Select \( k \) sets \( S_1, \ldots, S_k \) as in previous algorithm, for \( k = C \log(1/\delta)/\epsilon^2, \ C > 0 \)

2. Let \( Z \) be the number of values of \( \text{Sum}_{S_j}(x) \) that are equal to 0, \( j = 1, \ldots, k \).

3. If \( Z < k/e \) then report YES, otherwise report NO.

**Lemma**

If the constant \( C \) is large enough, then this algorithm reports a correct answer with probability \( 1 - \delta \).
Amplify the success probability

Repeat to amplify the success probability

1. Select $k$ sets $S_1, \ldots, S_k$ as in previous algorithm, for
   $k = C \log(1/\delta)/\epsilon^2$, $C > 0$

2. Let $Z$ be the number of values of $\text{Sum}_{S_j}(x)$ that are equal to 0,
   $j = 1, \ldots, k$.

3. If $Z < k/e$ then report YES, otherwise report NO.

Lemma

If the constant $C$ is large enough, then this algorithm reports a correct answer with probability $1 - \delta$.

Theorem

The number of distinct elements can be $(1 \pm \epsilon)$-approximated with probability $1 - \delta$ using $O(\log n \log(1/\delta)/\epsilon^3)$ words.
Course plan

www.cse.ust.hk/~qinzhang/HKUST-minicourse/index.html
That’s all for lecture 1.
Thank you.
**Frequency moments**:

\[ F_p = \sum_i |f_i|^p, \quad f_i: \text{frequency of item } i. \]

- \( F_0 \): number of distinct items.
- \( F_1 \): total number of items.
- \( F_2 \): size of self-join.
Frequency moments and norms

**Frequency moments:** \( F_p = \sum_i |f_i|^p \), \( f_i \): frequency of item \( i \).

- \( F_0 \): number of distinct items.
- \( F_1 \): total number of items.
- \( F_2 \): size of self-join.

A very good measurement of the skewness of the dataset.
**Frequency moments**: \( F_p = \sum_i |f_i|^p \), \( f_i \): frequency of item \( i \).

- \( F_0 \): number of distinct items.
- \( F_1 \): total number of items.
- \( F_2 \): size of self-join.

A very good measurement of the skewness of the dataset.

**Norms**: \( L_p = F_p^{1/p} \)
The sketch for $L_2$: a linear sketch $Rx = [Z_1, \ldots, Z_k]$, where each entry of $k \times n$ ($k = O(1/\epsilon^2)$) matrix $R$ has distribution $\mathcal{N}(0, 1)$.

- Each of $Z_i$ is drawn from $\mathcal{N}(0, \|x\|_2^2)$.
  Alternatively, $Z_i = \|x\|_2 G_i$, where $G_i$ drawn from $\mathcal{N}(0, 1)$. 
**$L_2$ estimation**

- **The sketch** for $L_2$: a linear sketch $Rx = [Z_1, \ldots, Z_k]$, where each entry of $k \times n$ ($k = O(1/\epsilon^2)$) matrix $R$ has distribution $\mathcal{N}(0,1)$.
  - Each of $Z_i$ is drawn from $\mathcal{N}(0, \|x\|_2^2)$.
  - Alternatively, $Z_i = \|x\|_2 G_i$, where $G_i$ drawn from $\mathcal{N}(0,1)$.

- **The estimator**: 
  
  $$Y = \frac{\text{median}\{|Z_1|, \ldots, |Z_k|\}}{\text{median}\{G\}}; \ G \sim \mathcal{N}(0,1) \quad ^a$$

  
  $^a$M is the median of a random variable $R$ if $Pr[|R| \leq M] = 1/2$
The sketch for $L_2$: a linear sketch $Rx = [Z_1, \ldots, Z_k]$, where each entry of $k \times n \ (k = O(1/\epsilon^2))$ matrix $R$ has distribution $\mathcal{N}(0, 1)$.

- Each of $Z_i$ is draw from $\mathcal{N}(0, \|x\|_2^2)$.
  Alternatively, $Z_i = \|x\|_2 \ G_i$, where $G_i$ drawn from $\mathcal{N}(0, 1)$.

The estimator:

$$Y = \text{median}\{|Z_1|, \ldots, |Z_k|\} / \text{median}\{G\}; \ G \sim \mathcal{N}(0, 1)$$

\[ a \]

\[ ^aM \ is \ the \ median \ of \ a \ random \ variable \ R \ if \ Pr[|R| \leq M] = 1/2 \]

Sounds like magic? The intuition behind:

For “nice”– looking distributions (e.g., the Gaussian), the median of those samples, for large enough # samples, should converge to the median of the distribution.
Closeness in Probability

Let $U_1, \ldots, U_k$ be i.i.d. real random variables chosen from any distribution having continuous c.d.f $F$ and median $M$. Defining $U = \text{median}\{U_1, \ldots, U_k\}$, there is an absolute constant $C > 0$, $\Pr[F(U) \in (1/2 - \epsilon, 1/2 + \epsilon)] \geq 1 - e^{-Ck\epsilon^2}$
The proof

- **Closeness in Probability**
  Let $U_1, \ldots, U_k$ be i.i.d. real random variables chosen from any distribution having continuous c.d.f $F$ and median $M$. Defining $U = \text{median}\{U_1, \ldots, U_k\}$, there is an absolute constant $C > 0$, $Pr[F(U) \in (1/2 - \epsilon, 1/2 + \epsilon)] \geq 1 - e^{-Ck\epsilon^2}$

- **Closeness in Value**
  Let $F$ be a c.d.f. of a random variable $|G|$, $G$ drawn from $\mathcal{N}(0,1)$. There exists an absolute constant $C' > 0$ such that if for any $z \geq 0$ we have $F(z) \in (1/2 - \epsilon, 1/2 + \epsilon)$, then $z = M \pm C'\epsilon$. 
The proof

- **Closeness in Probability**
  Let \( U_1, \ldots, U_k \) be i.i.d. real random variables chosen from any distribution having continuous c.d.f \( F \) and median \( M \). Defining \( U = \text{median}\{U_1, \ldots, U_k\} \), there is an absolute constant \( C > 0 \),
  \[
  \Pr[F(U) \in (1/2 - \epsilon, 1/2 + \epsilon)] \geq 1 - e^{-Ck\epsilon^2}
  \]

- **Closeness in Value**
  Let \( F \) be a c.d.f. of a random variable \(|G|\), \( G \) drawn from \( \mathcal{N}(0, 1) \).
  There exists an absolute constant \( C' > 0 \) such that if for any \( z \geq 0 \) we have \( F(z) \in (1/2 - \epsilon, 1/2 + \epsilon) \), then \( z = M \pm C'\epsilon \).

**Theorem**

\[
Y = \|x\|_2 (M \pm C'\epsilon)/M = \|x\|_2 (1 \pm C''\epsilon), \text{ w.h.p.}
\]
Key property of Gaussian distribution:
If $U_1, \ldots, U_n$ and $U$ are i.i.d drawn from Gaussian distribution, then $x_1 U_1 + \ldots + x_n U_n \sim \|x\|_p U$ for $p = 2$
Key property of **Gaussian distribution**:
If $U_1, \ldots, U_n$ and $U$ are i.i.d drawn from Guassian distribution, then $x_1 U_1 + \ldots + x_n U_n \sim \|x\|_p U$ for $p = 2$

Such distributions are called “$p$-stable” [Indyk ’06]
Good news: $p$-stable distributions exist for any $p \in (0, 2]$
Generalization

- Key property of **Gaussian distribution**: If \( U_1, \ldots, U_n \) and \( U \) are i.i.d drawn from Gaussian distribution, then \( x_1 U_1 + \ldots + x_n U_n \sim \|x\|_p U \) for \( p = 2 \)

- Such distributions are called "\( p \)-stable" [Indyk ’06] 
  Good news: \( p \)-stable distributions exist for any \( p \in (0, 2] \)

  For \( p = 1 \), we get **Cauchy distribution** with density function:
  \[
  f(x) = \frac{1}{\pi(1 + x^2)}
  \]
$L_p \ (p > 2)$ (Not linear mapping but important)

- We instead approximate $F_p = \sum_{i=1}^{n} x_i^p = \|x\|_p^p$
\[ L_p \ (p > 2) \ (\text{Not linear mapping but important}) \]

- We instead approximate \( F_p = \sum_{i=1}^{n} x_i^p = \|x\|_p^p \)

- First attempt: Use two passes.
  1. Pick a random element \( i \) from the stream in 1st pass. (Q: How?)
  2. Compute \( i \)'s frequency \( x_i \) in 2nd pass
  3. Finally, return \( Y = mx_i^{p-1} \).
\textbf{$L_p$ ($p > 2$) (Not linear mapping but important)}

- We instead approximate $F_p = \sum_{i=1}^{n} x_i^p = \|x\|_p^p$

- First attempt: Use two passes.
  1. Pick a random element $i$ from the stream in 1st pass. (Q: How?)
  2. Compute $i$’s frequency $x_i$ in 2nd pass
  3. Finally, return $Y = mx_i^{p-1}$.

- Second attempt: Collapse the two passes above
  1. Pick a random element $i$ from the stream, count the number of occurrences of $i$ in the rest of the stream, denoted by $r$.
  2. Now we use $r$ instead of $x_i$ to construct the estimator: $Y' = m(r^p - (r - 1)^p)$.
**Heavy hitters**

- $L_p$ heavy hitter set:

$$HH^p_\phi(x) = \{ i : |x_i| \geq \phi \|x\|_p \}$$
$L_p$ heavy hitter set:

$$HH^p_{\phi}(x) = \{ i : |x_i| \geq \phi \|x\|_p \}$$

$L_p$ Heavy Hitter Problem:

Given $\phi, \phi'$, (often $\phi' = \phi - \epsilon$), return a set $S$ such that

$$HH^p_{\phi}(x) \subseteq S \subseteq HH^p_{\phi'}(x)$$
Heavy hitters

- $L_p$ heavy hitter set:

$$HH^p_\phi(x) = \{ i : |x_i| \geq \phi \|x\|_p \}$$

- $L_p$ Heavy Hitter Problem:
  Given $\phi, \phi'$, (often $\phi' = \phi - \epsilon$), return a set $S$ such that

$$HH^p_\phi(x) \subseteq S \subseteq HH^p_{\phi'}(x)$$

- $L_p$ Point Query Problem:
  Given $\epsilon$, after reading the whole stream, given $i$, report

$$x^*_i = x_i \pm \epsilon \|x\|_p$$
$L_2$ point query

**The algorithm:**

[Gilbert, Kotidis, Muthukrishnan and Strauss ’01]

- Maintain a sketch $Rx$ such that $s = \|Rx\|_2 = (1 \pm \epsilon) \|x\|_2$
  ($R$ is a $O(1/\epsilon^2 \log(1/\delta)) \times n$ matrix, which can be constructed, e.g., by taking each cell to be $\mathcal{N}(0, 1)$)

- Estimator: $x^*_i = (1 - \|Rx/s - Re_i\|_2^2 / 2)s$
The algorithm:
[Gilbert, Kotidis, Muthukrishnan and Strauss ’01]

- Maintain a sketch $Rx$ such that $s = \|Rx\|_2 = (1 \pm \epsilon) \|x\|_2$
  ($R$ is a $O(1/\epsilon^2 \log(1/\delta)) \times n$ matrix, which can be constructed, e.g., by taking each cell to be $\mathcal{N}(0, 1)$)

- Estimator: $x_i^* = (1 - \|Rx/s - Re_i\|_2^2 / 2)s$

Johnson-Linderstrauss Lemma

$\forall x \|x\|_2 = \ell$, we have $(1 - \epsilon)\ell \leq \|Rx\|_2^2 / k \leq (1 + \epsilon)\ell$ w.p. $1 - \delta$. 
**The algorithm:**

[Gilbert, Kotidis, Muthukrishnan and Strauss ’01]

- Maintain a sketch $R x$ such that $s = \| R x \|_2 = (1 \pm \epsilon) \| x \|_2$
  
  ($R$ is a $O(1/\epsilon^2 \log(1/\delta)) \times n$ matrix, which can be constructed, e.g., by taking each cell to be $\mathcal{N}(0, 1)$)

- Estimator: $x^*_i = (1 - \| R x / s - R e_i \|_2^2 / 2) s$

**Johnson-Linderstrauss Lemma**

\[ \forall x \quad \| x \|_2 = \ell, \text{ we have } (1 - \epsilon)\ell \leq \| R x \|_2^2 / k \leq (1 + \epsilon)\ell \quad \text{w.p. } 1 - \delta. \]

**Theorem**

We can solve $L_2$ point query, with approximation $\epsilon$, and failure probability $\delta$ by storing $O(1/\epsilon^2 \log(1/\delta))$ numbers.
$L_1$ point query

The algorithm for $x \geq 0$ [Cormode and Muthu '05]

- Pick $d$ ($d = \log(1/\delta)$) random hash functions $h_1, \ldots, h_d$ where $h_i : \{1, \ldots, n\} \rightarrow \{1, \ldots, w\}$ ($w = 2/\epsilon$).

- Maintain $d$ vectors $Z^1, \ldots, Z^d$ where $Z^t = \{Z^t_1, \ldots, Z^t_w\}$ such that $Z^t_j = \sum_{i:h_t(i)=j} x_i$

- Estimator: $x^*_i = \min_t Z^t_{h_t(i)}$
The algorithm for $x \geq 0$ [Cormode and Muthu '05]

- Pick $d$ ($d = \log(1/\delta)$) random hash functions $h_1, \ldots, h_d$ where $h_i : \{1, \ldots, n\} \rightarrow \{1, \ldots, w\}$ ($w = 2/\epsilon$).
- Maintain $d$ vectors $Z^1, \ldots, Z^d$ where $Z^t = \{Z^t_1, \ldots, Z^t_w\}$ such that $Z^t_j = \sum_{i: h_t(i) = j} x_i$
- Estimator: $x^*_i = \min_t Z^t_{h_t(i)}$

Theorem

We can solve $L_1$ point query, with approximation $\epsilon$, and failure probability $\delta$ by storing $O(1/\epsilon \log(1/\delta))$ numbers.
Compressive sensing

The model (Candes-Romberg-Tao ’04; Donoho ’04)

Applications
- Medical imaging reconstruction
- Single-pixel camera
- Compressive sensor network
  etc.
Formalization

- *$L_p/L_q$ guarantee*: The goal to acquire a signal $x = [x_1, \ldots, x_n]$ (e.g., a digital image). The acquisition proceeds by computing a measurement vector $Ax$ of dimension $m \ll n$. Then, from $Ax$, we want to recover a $k$-sparse approximation $x'$ of $x$ so that

$$\|x - x'\|_q \leq C \cdot \min_{\|x''\|_0 \leq k} \|x - x''\|_p$$

(*)
\textbf{Lp/Lq guarantee}: The goal to acquire a signal \( x = [x_1, \ldots, x_n] \) (e.g., a digital image). The acquisition proceeds by computing a measurement vector \( Ax \) of dimension \( m \ll n \). Then, from \( Ax \), we want to recover a \( k \)-sparse approximation \( x' \) of \( x \) so that

\[
\| x - x' \|_q \leq C \cdot \min_{\| x'' \|_0 \leq k} \| x - x'' \|_p
\]  

\((*)\)
Formalization

- **$L_p/L_q$ guarantee**: The goal to acquire a signal $x = [x_1, \ldots, x_n]$ (e.g., a digital image). The acquisition proceeds by computing a measurement vector $Ax$ of dimension $m \ll n$. Then, from $Ax$, we want to recover a $k$-sparse approximation $x'$ of $x$ so that

$$\|x - x'\|_q \leq C \cdot \min_{\|x''\|_0 \leq k} \|x - x''\|_p \quad (\ast)$$

- Often study: $L_1/L_1$, $L_1/L_2$ and $L_2/L_2$
**Formalization**

- **Lp/Lq guarantee**: The goal to acquire a signal $x = [x_1, \ldots, x_n]$ (e.g., a digital image). The acquisition proceeds by computing a measurement vector $Ax$ of dimension $m \ll n$. Then, from $Ax$, we want to recover a $k$-sparse approximation $x'$ of $x$ so that

$$\|x - x'\|_q \leq C \cdot \min_{\|x''\|_0 \leq k} \|x - x''\|_p \quad (*)$$

- Often study: $L_1/L_1$, $L_1/L_2$ and $L_2/L_2$

- **For each**: Given a (random) matrix $A$, for each signal $x$, $(*)$ holds w.h.p.

  **For all**: One matrix $A$ for all signals $x$. Stronger.
### Results

**Scale:**
- Excellent
- Very Good
- Good
- Fair

#### Result Table

<table>
<thead>
<tr>
<th>Paper</th>
<th>Rand./Det.</th>
<th>Sketch length</th>
<th>Encode time</th>
<th>Col. sparsity/Update time</th>
<th>Recovery time</th>
<th>Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CCF'02], [CM'06]</td>
<td>R</td>
<td>$k \log n$</td>
<td>$n \log n$</td>
<td>$\log n$</td>
<td>$n \log n$</td>
<td>12/12</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$k \log^c n$</td>
<td>$n \log^c n$</td>
<td>$\log^c n$</td>
<td>$k \log^c n$</td>
<td>12/12</td>
</tr>
<tr>
<td>[CM'04]</td>
<td>R</td>
<td>$k \log n$</td>
<td>$n \log n$</td>
<td>$\log n$</td>
<td>$n \log n$</td>
<td>11/11</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$k \log^c n$</td>
<td>$n \log^c n$</td>
<td>$\log^c n$</td>
<td>$k \log^c n$</td>
<td>11/11</td>
</tr>
<tr>
<td>[CRT'04] [RJ'05]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>nk log(n/k)</td>
<td>$k \log(n/k)$</td>
<td>$n^c$</td>
<td>12/11</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>$k \log^c n$</td>
<td>$n \log n$</td>
<td>$k \log^c n$</td>
<td>$n^c$</td>
<td>12/11</td>
</tr>
<tr>
<td>[GSTV'06] [GSTV'07]</td>
<td>D</td>
<td>$k \log^c n$</td>
<td>$n \log^c n$</td>
<td>$k \log^c n$</td>
<td>$k^2 \log^c n$</td>
<td>12/11</td>
</tr>
<tr>
<td>[BGIKS'08]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>nk log(n/k)</td>
<td>$k \log(n/k)$</td>
<td>$n^c$</td>
<td>11/11</td>
</tr>
<tr>
<td>[GLR'08]</td>
<td>D</td>
<td>$k \log^{\log \log n}$</td>
<td>nk log(n/k)</td>
<td>$k \log^{\log \log n}$</td>
<td>$n^{1.4}$</td>
<td>11/11</td>
</tr>
<tr>
<td>[NV'07], [DM'08], [NT'08, BD'08]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>nk log(n/k)</td>
<td>$k \log(n/k)$</td>
<td>$nk \log(n/k)^4 T$</td>
<td>12/11</td>
</tr>
<tr>
<td>[IR'08]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>nk log(n/k)</td>
<td>$k \log(n/k)$</td>
<td>$nk \log(n/k)^4 T$</td>
<td>11/11</td>
</tr>
<tr>
<td>[BIR'08]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>nk log(n/k)</td>
<td>$k \log(n/k)$</td>
<td>$nk \log(n/k)^4 T$</td>
<td>11/11</td>
</tr>
<tr>
<td>[DIP'09]</td>
<td>D</td>
<td>$\Omega(k \log(n/k))$</td>
<td></td>
<td></td>
<td></td>
<td>11/11</td>
</tr>
<tr>
<td>[CDD'07]</td>
<td>D</td>
<td>$\Omega(n)$</td>
<td></td>
<td></td>
<td></td>
<td>12/12</td>
</tr>
</tbody>
</table>

#### Legend:
- $n = $ dimension of $x$
- $m = $ dimension of $Ax$
- $k = $ sparsity of $x^*$
- $T = $ #iterations

#### Approx guarantee:
- $12/12$: $\|x-x^*\|_2 \leq C\|x-x^*\|_2$
- $12/11$: $\|x-x^*\|_2 \leq C\|x-x^*\|_1/k^{\sqrt{2}}$
- $11/11$: $\|x-x^*\|_1 \leq C\|x-x^*\|_1$

#### Caveats:
1. Only results for general vectors $x$ are shown.
2. All bounds up to $O()$ factors.
3. Specific matrix type often matters (Fourier, sparse, etc).
4. Ignore universality, explicitness, etc.
5. Most “dominated” algorithms not shown.

---

Up to year 2009 ... copied from Indky’s talk
For each \((L_1/L_1)\)

- The algorithm for \(L_1\) point query gives a \(L_1/L_1\) sparse approximation.
For each \((L_1/L_1)\)

- The algorithm for \(L_1\) point query gives a \(L_1/L_1\) sparse approximation.

Recall \(L_1\) Point Query Problem: Given \(\epsilon\), after reading the whole stream, given \(i\), report \(x_i^* = x_i \pm \epsilon \|x\|_1\)
For each \((L_1/L_1)\)

- The algorithm for \(L_1\) point query gives a \(L_1/L_1\) sparse approximation.

Recall \(L_1\) Point Query Problem: Given \(\epsilon\), after reading the whole stream, given \(i\), report \(x_i^* = x_i \pm \epsilon \|x\|_1\)

- Set \(\epsilon = \alpha/k\) and \(\delta = 1/n^2\) in \(L_1\) point query. And then return a vector \(x'\) consisting of \(k\) largest (in magnitude) elements of \(x^*\). It gives w.p. \(1 - \delta\),

\[
\|x - x'\|_1 \leq (1 + 3\alpha) \cdot \text{Err}^k_1
\]

Total measurements: \(m = O(k/\alpha \cdot \log n)\)
A matrix $A$ satisfies $(k, \delta)$-RIP (Restricted Isometry Property) if
\[ \forall \text{ } k\text{-sparse vector } x \text{ we have } (1 - \delta) \|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta) \|x\|_2. \]
For all \((L_1/L_2)\)

A matrix \(A\) satisfies \((k, \delta)\)-RIP (Restricted Isometry Property) if \(\forall \; k\)-sparse vector \(x\) we have 

\[
(1 - \delta) \|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta) \|x\|_2.
\]

**Johnson-Linderstrauss Lemma**

\(\forall \; x \text{ with } \|x\|_2 = 1, \text{ we have } 7/8 \leq \|Ax\|_2 \leq 8/7 \text{ w.p. } 1 - e^{-O(m)}.\)
For all \((L_1/L_2)\)

A matrix \(A\) satisfies \((k, \delta)\)-RIP \((\text{Restricted Isometry Property})\) if
\[
\forall \ k\text{-sparse vector } x \text{ we have } (1 - \delta) \|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta) \|x\|_2.
\]

### Johnson-Linderstrauss Lemma

\[
\forall x \text{ with } \|x\|_2 = 1, \text{ we have } 7/8 \leq \|Ax\|_2 \leq 8/7 \text{ w.p. } 1 - e^{-O(m)}.
\]

### Theorem

If each entry of \(A\) is i.i.d. as \(\mathcal{N}(0, 1)\), and \(m = O(k\log(n/k))\), then \(A\) satisfies \((k, 1/3)\)-RIP w.h.p.
For all \((L_1/L_2)\)

A matrix \(A\) satisfies \((k, \delta)\)-RIP (Restricted Isometry Property) if \(\forall \) \(k\)-sparse vector \(x\) we have \((1 - \delta) \|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta) \|x\|_2\).

**Johnson-Linderstrauss Lemma**

\(\forall \) \(x\) with \(\|x\|_2 = 1\), we have \(7/8 \leq \|Ax\|_2 \leq 8/7\) w.p. \(1 - e^{-O(m)}\).

**Theorem**

If each entry of \(A\) is i.i.d. as \(\mathcal{N}(0, 1)\), and \(m = O(k \log(n/k))\), then \(A\) satisfies \((k, 1/3)\)-RIP w.h.p.

**Main Theorem**

If \(A\) has \((6k, 1/3)\)-RIP. Let \(x^*\) be the solution to the LP:

minimize \(\|x^*\|_1\) subject to \(Ax^* = Ax\) (\(x^*\) is \(k\)-sparse). Then

\[
\|x - x^*\|_2 \leq C/\sqrt{k} \cdot Err_k^1
\]

for any \(x\).
The lower bounds

- **What’s known:** There exists a $m \times n$ matrix $A$ with $m = O(k \log n)$ (can be improved to $m = O(k \log(n/k))$, and a $L_1/L_1$ recovery algorithm $R$ so that for each $x$, $R(Ax) = x'$ such that w.h.p.

\[
\|x - x'\|_1 \leq C \cdot \min_{\|x''\|_0 \leq k} \|x - x''\|_1.
\]
The lower bounds

- **What’s known:** There exists a \( m \times n \) matrix \( A \) with \( m = O(k \log n) \) (can be improved to \( m = O(k \log(n/k)) \)), and a \( L_1/L_1 \) recovery algorithm \( R \) so that for each \( x \), 
  \( R(Ax) = x' \) such that w.h.p.

  \[
  \|x - x'\|_1 \leq C \cdot \min_{\|x''\|_0 \leq k} \|x - x''\|_1.
  \]

- We are going to show that this is optimal. That is, \( m = \Omega(k \log(n/k)) \). [Do Ba et. al. SODA ’10]
  To show this we need
  - Communication complexity
  - Coding theory
They want to jointly compute some function $f(x, y)$
They want to jointly compute some function $f(x, y)$

We would like to minimize

- Total bits of communication
- # rounds of communication

(today we only consider 1-round protocol)
**Promise Input:**
Alice gets $x = \{x_1, x_2, \ldots, x_d\} \in \{0, 1\}^d$

Bob gets $y = \{y_1, y_2, \ldots, y_d\} \in \{0, 1, \bot\}^d$
such that for some (unique) $i$:

1. $y_i \in \{0, 1\}$
2. $y_k = x_k$ for all $k > i$
3. $y_1 = y_2 = \ldots = y_{i-1} = \bot$

**Output:**
Does $x_i = y_i$ (YES/NO)?
Augmented indexing

Promise Input:
Alice gets \( x = \{ x_1, x_2, \ldots, x_d \} \in \{0, 1\}^d \)
Bob gets \( y = \{ y_1, y_2, \ldots, y_d \} \in \{0, 1, \bot\}^d \)
such that for some (unique) \( i \):

1. \( y_i \in \{0, 1\} \)
2. \( y_k = x_k \) for all \( k > i \)
3. \( y_1 = y_2 = \ldots = y_{i-1} = \bot \)

Output:
Does \( x_i = y_i \) (YES/NO)?

Theorem
Any 1-round protocol for Augmented-Indexing that succeeds w.p. \( 1 - \delta \) for some small const \( \delta \) has communication complexity \( \Omega(d) \).
The proof

Let $X$ be the maximal set of $k$-sparse $n$-dimensional binary vectors with minimum Hamming distance $k$. We have $\log |X| = \Omega(k \log(n/k))$. 
The proof

- Let $X$ be the maximal set of $k$-sparse $n$-dimensional binary vectors with minimum Hamming distance $k$. We have $\log |X| = \Omega(k \log(n/k))$.

- Restate of the input for Augmented-Indexing (AI): Alice is given $y \in \{0, 1\}^d$ and Bob is given $i \in d$ and $y_{i+1}, y_{i+2}, \ldots, y_d$. Goal: Bob wants to learn $y_i$. 
Let $X$ be the maximal set of $k$-sparse $n$-dimensional binary vectors with minimum Hamming distance $k$. We have $\log |X| = \Omega(k \log(n/k))$.

Restate of the input for Augmented-Indexing (AI): Alice is given $y \in \{0, 1\}^d$ and Bob is given $i \in d$ and $y_{i+1}, y_{i+2}, \ldots, y_d$. Goal: Bob wants to learn $y_i$.

A protocol using $L_1/L_1$ recovery for AI:
Set $d = \log |X| \log n$. Let $D = 2C + 3$ ($C$ is the constant in the sparse recovery)
Protocol in next slides
The proof (cont.)

A protocol using $L_1/L_1$ recovery for AI:

1. Alice splits her string $y$ into $\log n$ contiguous chunks $y^1, \ldots, y^{\log n}$, each containing $\log |X|$ bits. She uses $y^j$ as an index into $X$ to choose $x_j$. Alice define: $x = D^1x_1 + D^2x_2 + \ldots + D^{\log n}x_{\log n}$.

2. Alice and Bob use shared randomness to choose a random matrix $A$ with orthonormal rows, and round it to $A'$ with $b = O(\log n)$ bits per entry. Alice computes $A'x$ and send to Bob.

3. Bob uses $i$ to compute $j = j(i)$ for which the bit $y_i$ occurs in $y^j$. Bob also uses $y_{i+1}, \ldots, y_d$ to compute $x_{j+1}, \ldots, x_{\log n}$, and he can compute $z = D^{j+1}x_{j+1} + D^{j+2}x_{j+2} + \ldots + D^{\log n}x_{\log n}$.

4. Set $w = x - z$ Bob then computes $A'w$ using $A'z$ and $A'x$.

5. From $w$ Bob can recover $w'$ such that $\|w - u - w'\|_1 \leq C \cdot \min_{\|x''\|_0 \leq k} \|w - u - w''\|_1$, where $u \in_R B^n_1(k)$ (the $L_1$ ball of radius $k$).

6. From $w'$ he can recover $x_j$, thus $y^j$, thus bit $y_i$. 

Next topic:

Graph Algorithms
Goal: sample an element from the support of $a \in \mathbb{R}^n$
Goal: sample an element from the support of $a \in \mathbb{R}^n$

Algorithm

- Maintain $\tilde{F}_0$, an $(1 \pm 0.1)$-approximation to $F_0$.
- Hash items using $h_j : [n] \rightarrow [0, 2^j - 1]$ for $j \in [\log n]$.
- For each $j$, maintain:
  - $D_j = (1 \pm 0.1) |\{ t \mid h_j(t) = 0\}|$
  - $S_j = \sum_{t, h_j(t) = 0} f_t i_t$
  - $C_j = \sum_{t, h_j(t) = 0} f_t$
**Goal:** sample an element from the support of \( a \in \mathbb{R}^n \)

**Algorithm**
- Maintain \( \tilde{F}_0 \), an \((1 \pm 0.1)\)-approximation to \( F_0 \).
- Hash items using \( h_j : [n] \rightarrow [0, 2^j - 1] \) for \( j \in [\log n] \).
- For each \( j \), maintain:
  - \( D_j = (1 \pm 0.1) |\{ t \mid h_j(t) = 0 \}| \)
  - \( S_j = \sum_{t, h_j(t) = 0} f_t i_t \)
  - \( C_j = \sum_{t, h_j(t) = 0} f_t \)

**Lemma**
At level \( j = 2 + \lceil \log \tilde{F}_0 \rceil \), there is a *unique* element in the stream that maps to 0 with constant probability.
**$L_0$ sampling**

**Goal:** sample an element from the support of $a \in \mathbb{R}^n$

**Algorithm**
- Maintain $\tilde{F}_0$, an $(1 \pm 0.1)$-approximation to $F_0$.
- Hash items using $h_j : [n] \to [0, 2^j - 1]$ for $j \in [\log n]$.
- For each $j$, maintain:
  - $D_j = (1 \pm 0.1) |\{ t \mid h_j(t) = 0\}|$
  - $S_j = \sum_{t,h_j(t)=0} f_t i_t$
  - $C_j = \sum_{t,h_j(t)=0} f_t$

**Lemma**
At level $j = 2 + \lceil \log \tilde{F}_0 \rceil$, there is a unique element in the stream that maps to 0 with constant probability.

Uniqueness is verified if $D_j = 1 \pm 0.1$. If unique, then $S_j = C_j$ gives identity of the element and $C_j$ is the count.
Graphs

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.
Graphs

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

- For example: Connectivity is easy with $\tilde{O}(n)$ space if edges are only inserted. But what if edges get deleted?
In semi-streaming, want to process graph defined by edges \( e_1, \ldots, e_m \) with \( \tilde{O}(n) \) memory and reading sequence in order.

For example: Connectivity is easy with \( \tilde{O}(n) \) space if edges are only inserted. But what if edges get deleted?

A sketch matrix with dimension \( \tilde{O}(n) \times n^2 \) suffice!

To delete \( e \) from \( G \): update
\[
MA_G \rightarrow MA_G - MA_e = MA_{G-e},
\]
where \( A_G \) is the adjacency matrix of \( G \).
In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

For example: Connectivity is easy with $\tilde{O}(n)$ space if edges are only inserted. But what if edges get deleted?

A sketch matrix with dimension $\tilde{O}(n) \times n^2$ suffice!

To delete $e$ from $G$: update 

$MA_G \rightarrow MA_G - MA_e = MA_{G - e}$,

where $A_G$ is the adjacency matrix of $G$.

Magic? Mmm, the information of connectivity is $\tilde{O}(n)$ :-(
**Basic algorithm** (Spanning Forest):
1. For each node, sample a random incident edge
2. Contract selected edges. Repeat until no edges.
**Basic algorithm** (Spanning Forest):
1. For each node, sample a random incident edge
2. Contract selected edges. Repeat until no edges.

**Lemma**

Takes $O(\log n)$ steps and selected edges include spanning forest.
**Basic algorithm** (Spanning Forest):
1. For each node, sample a random incident edge
2. Contract selected edges. Repeat until no edges.

**Lemma**
Takes $O(\log n)$ steps and selected edges include spanning forest.

**Graph Representation** For node $i$, let $a_i$ be vector indexed by node pairs. Non-zero entries: $a_i[i,j] = 1$ if $j > i$ and $a_i[i,j] = -1$ if $j < i$. 
Basic algorithm (Spanning Forest):
1. For each node, sample a random incident edge
2. Contract selected edges. Repeat until no edges.

**Lemma**
Takes $O(\log n)$ steps and selected edges include spanning forest.

**Graph Representation** For node $i$, let $a_i$ be vector indexed by node pairs. Non-zero entries: $a_i[i, j] = 1$ if $j > i$ and $a_i[i, j] = -1$ if $j < i$.

**Lemma**
For any subset of nodes $S \subset V$,
$$\text{support}(\sum_{i \in S} a_i) = E[S, V \setminus S]$$
Sketch: Apply $\log n$ sketches $C_i$ to each $a_j$
Sketch: Apply log \( n \) sketches \( C_i \) to each \( a_j \)

Run previous algorithm in sketch space:
1. Use \( C_1 a_j \) to get incident edge on each node \( j \)
2. For \( i = 2 \) to \( t \):
   - To get an incident edge on supernode \( S \subset V \) use:
     \[
     \sum_{j \in S} C_i a_j = C_i(\sum_{j \in S} a_j)
     \]
   - Use \( L_0 \) sampling algorithm to sample an edge
     \[
     e \in \text{support}(\sum_{i \in S} a_i) = E[S, V \setminus S]
     \]