Big-Data Algorithms: Computing the ℓ_2 Norm

Reference: http://www.sketchingbigdata.org/fall17/lec/lec3.pdf

Basic Data Stream Model

- Single pass over the data $i_1, i_2, \ldots, i_n \in [m]$.
  Here, $[m] = \{1, 2, \ldots, m\}$.

Basic Data Stream Model

- Single pass over the data $i_1, i_2, \ldots, i_n \in [m]$.
  Here, $[m] = \{1, 2, \ldots, m\}$.
- Bounded storage (often $\log^{O(1)} n$)

Basic Data Stream Model

- Single pass over the data $i_1, i_2, \ldots, i_n \in [m]$.
  Here, $[m] = \{1, 2, \ldots, m\}$.
- Bounded storage (often $\log^{O(1)} n$)
  - Units of storage: bits, words, "elements" (e.g., points, nodes/edges)

Basic Data Stream Model

- Single pass over the data $i_1, i_2, \ldots, i_n \in [m]$.
  Here, $[m] = \{1, 2, \ldots, m\}$.
- Bounded storage (often $\log^{O(1)} n$)
  - Units of storage: bits, words, "elements" (e.g., points, nodes/edges)
- Randomness and approximation OK (almost always necessary)

Basic Data Stream Model

- Single pass over the data $i_1, i_2, \ldots, i_n \in [m]$.
  Here, $[m] = \{1, 2, \ldots, m\}$.
- Bounded storage (often $\log^{O(1)} n$)
  - Units of storage: bits, words, "elements" (e.g., points, nodes/edges)
- Randomness and approximation OK (almost always necessary)
- Last lecture: estimating the number of distinct elements

$$\mathbb{P}(\text{relative error} < \varepsilon) > \frac{2}{3}$$

with space $O(\log n + (1/\varepsilon)^2)$.

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.
- Note that $a < 0$ is possible, but maybe problematic with algorithms for queries.

Generalization

- Vector $x = (x_1, x_2, \dots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.
- Note that $a < 0$ is possible, but maybe problematic with algorithms for queries.
- Number of distinct elements $=$ number of non-zero coordinates in x, denote by $\|x\|_0$.
  Algorithm from last time can also be used in this model.

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.
- Note that $a < 0$ is possible, but maybe problematic with algorithms for queries.
- Number of distinct elements = number of non-zero coordinates in x, denote by $\|x\|_0$.
  Algorithm from last time can also be used in this model.
- Will consider two methods for calculating $\|x\|_{\ell_2}^2 = \sum_{i=1}^{n} x_i^2$

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.
- Note that $a < 0$ is possible, but maybe problematic with algorithms for queries.
- Number of distinct elements = number of non-zero coordinates in x, denote by $\|x\|_0$.
  Algorithm from last time can also be used in this model.
- Will consider two methods for calculating $\|x\|_{\ell_2}^2 = \sum_{i=1}^n x_i^2$
  - Alon-Matias-Szegedy (AMS)

Generalization

- Vector $x = (x_1, x_2, \ldots, x_m)$, where each x_i is number of times $i \in [m]$ has been seen so far.
- Stream: sequence of updates (i, a), meaning $x_i \leftarrow x_i + a$.
- Basic streaming model: Corresponds to updates $(i, 1)$.
- Note that $a < 0$ is possible, but maybe problematic with algorithms for queries.
- Number of distinct elements = number of non-zero coordinates in x, denote by $\|x\|_0$.
  Algorithm from last time can also be used in this model.
- Will consider two methods for calculating $\|x\|_{\ell_2}^2 = \sum_{i=1}^n x_i^2$
  - Alon-Matias-Szegedy (AMS)
  - Johnson-Lindenstrauss

Why Calculate ℓ_2 norm?

- General ℓ_p norms:

$$\|x\|_{\ell_p} = \begin{cases} \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1 \leq i \leq n} |x_i| & \text{if } p = \infty. \end{cases}$$

Why Calculate ℓ_2 norm?

- General ℓ_p norms:

$$\|x\|_{\ell_p} = \begin{cases} \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1 \le i \le n} |x_i| & \text{if } p = \infty. \end{cases}$$

- $\|x\|_1/n$: average of $|x_1|, \ldots, |x_n|$.

Why Calculate ℓ_2 norm?

- General ℓ_p norms:

$$\|x\|_{\ell_p} = \begin{cases} \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1 \le i \le n} |x_i| & \text{if } p = \infty. \end{cases}$$

- $\|x\|_1/n$: average of $|x_1|, \ldots, |x_n|$.
- $\|x\|_{\ell_2}$ measures spikiness of x:
  If $\|x\|_1$ is fixed, then $\|x_2\|$ is minimized if
  $x_1 = x_2 = \cdots = x_n = 1/\|x\|_1$.

Why Calculate ℓ_2 norm?

- General ℓ_p norms:

$$\|x\|_{\ell_p} = \begin{cases} \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1\le i\le n} |x_i| & \text{if } p = \infty. \end{cases}$$

- $\|x\|_1/n$: average of $|x_1|, \ldots, |x_n|$.
- $\|x\|_{\ell_2}$ measures spikiness of x:
  If $\|x\|_1$ is fixed, then $\|x_2\|$ is minimized if
  $x_1 = x_2 = \cdots = x_n = 1/\|x\|_1$.
- ℓ_2-norm estimation also arises in database applications.

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \leq i \leq n)$$

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, “The Space Complexity of Approximating the Frequency Moments.” *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \leq i \leq n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

under increments to the x_i.
Since Z is linear in x:

update x by (i, a) simply via $Z \leftarrow Z + r_i a$

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \leq i \leq n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

  under increments to the x_i.
  Since Z is linear in x:

  update x by (i, a) simply via $Z \leftarrow Z + r_i a$

- Return Z^2 as our estimator for $\|x\|_{\ell_2}^2$.

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \leq i \leq n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

  under increments to the x_i.
  Since Z is linear in x:

  update x by (i, a) simply via $Z \leftarrow Z + r_i a$

- Return Z^2 as our estimator for $\|x\|_{\ell_2}^2$.

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \le i \le n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

  under increments to the x_i.
  Since Z is linear in x:

  update x by (i, a) simply via $Z \leftarrow Z + r_i a$

- Return Z^2 as our estimator for $\|x\|_{\ell_2}^2$.

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \dots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \le i \le n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

  under increments to the x_i.
  Since Z is linear in x:

  update x by (i, a) simply via $Z \leftarrow Z + r_i a$

- Return Z^2 as our estimator for $\|x\|_{\ell_2}^2$.
- Analysis?
  - Compute expectation of Z^2.

AMS Algorithm

Reference: N. Alon, Y. Matias, M. Szegedy, "The Space Complexity of Approximating the Frequency Moments." *J. Comput. Syst. Sci.*, 58(1): 137–147. 1999.

- Choose $r_1, \ldots, r_m$ to be independently identically distributed random variables, with

$$\mathbb{P}[r_i = 1] = \mathbb{P}[r_i = -1] = \tfrac{1}{2} \qquad (1 \leq i \leq n)$$

- Maintain

$$Z = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i$$

  under increments to the x_i.
  Since Z is linear in x:

  update x by (i, a) simply via $Z \leftarrow Z + r_i a$

- Return Z^2 as our estimator for $\|x\|_{\ell_2}^2$.
- Analysis?
  - Compute expectation of Z^2.
  - Bound the variance of Z^2.

AMS Algorithm: Compute Expectation

- We have

$$\mathbb{E}[Z^2] = \mathbb{E}\left(\sum_{i=1}^{n} r_i x_i\right)^2 = \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right)$$

$$= \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right) = \sum_{i,j=1}^{n} x_i x_j \mathbb{E}[r_i r_j]$$

AMS Algorithm: Compute Expectation

- We have

$$\mathbb{E}[Z^2] = \mathbb{E}\left(\sum_{i=1}^{n} r_i x_i\right)^2 = \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right)$$
$$= \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right) = \sum_{i,j=1}^{n} x_i x_j \mathbb{E}[r_i r_j]$$

- But

AMS Algorithm: Compute Expectation

- We have

$$\mathbb{E}[Z^2] = \mathbb{E}\left(\sum_{i=1}^{n} r_i x_i\right)^2 = \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right)$$
$$= \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right) = \sum_{i,j=1}^{n} x_i x_j \mathbb{E}[r_i r_j]$$

- But
  - For $i \neq j$, we have $\mathbb{E}[r_i r_j] = \mathbb{E}[r_i]\mathbb{E}[r_j] = 0$, and so term disappears.

AMS Algorithm: Compute Expectation

- We have

$$\mathbb{E}[Z^2] = \mathbb{E}\left(\sum_{i=1}^{n} r_i x_i\right)^2 = \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right)$$

$$= \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right) = \sum_{i,j=1}^{n} x_i x_j \mathbb{E}[r_i r_j]$$

- But
  - For $i \neq j$, we have $\mathbb{E}[r_i r_j] = \mathbb{E}[r_i]\mathbb{E}[r_j] = 0$, and so term disappears.
  - For $i = j$, we have $\mathbb{E}[r_i r_j] = \mathbb{E}[r_i^2] = \mathbb{E}[1] = 1$.

AMS Algorithm: Compute Expectation

- We have

$$\mathbb{E}[Z^2] = \mathbb{E}\left(\sum_{i=1}^{n} r_i x_i\right)^2 = \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right)$$

$$= \mathbb{E}\left(\sum_{i,j=1}^{n} r_i x_i r_j x_j\right) = \sum_{i,j=1}^{n} x_i x_j \mathbb{E}[r_i r_j]$$

- But
  - For $i \neq j$, we have $\mathbb{E}[r_i r_j] = \mathbb{E}[r_i]\mathbb{E}[r_j] = 0$, and so term disappears.
  - For $i = j$, we have $\mathbb{E}[r_i r_j] = \mathbb{E}[r_i^2] = \mathbb{E}[1] = 1$.
- So

$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} x_i^2 = \|x\|_{\ell_2}^2$$

and hence Z^2 is an unbiased estimator of $\|x\|_{\ell_2}^2$.

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

- Decompose into a sum of

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

- Decompose into a sum of
  - $\sum_{i=1}^{n}(r_i x_i)^4$, with expectation $\sum_{i=1}^{n} x_i^4$.

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

- Decompose into a sum of
  - $\sum_{i=1}^{n} (r_i x_i)^4$, with expectation $\sum_{i=1}^{n} x_i^4$.
  - $6 \sum_{i,j=1}^{n} (r_i r_j x_i x_j)^2$, with expectation $6 \sum_{1 \le i < j \le n} x_i^2 x_j^2$

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

- Decompose into a sum of
  - $\sum_{i=1}^{n}(r_i x_i)^4$, with expectation $\sum_{i=1}^{n} x_i^4$.
  - $6\sum_{i,j=1}^{n}(r_i r_j x_i x_j)^2$, with expectation $6\sum_{1\le i<j\le n} x_i^2 x_j^2$
  - Terms involving no repeated multipliers (such as $r_1 x_1 r_2 x_2 r_3 x_3 r_4 x_4$), with expectation 0.

AMS Algorithm: Bounding the Variance

- We have

$$\mathrm{Var}[Z^2] = \mathbb{E}[(Z^2)^2] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4$$

- Now

$$Z^4 = \left(\sum_{i=1}^{n} r_i x_i\right)^4$$

- Decompose into a sum of
  - $\sum_{i=1}^{n}(r_i x_i)^4$, with expectation $\sum_{i=1}^{n} x_i^4$.
  - $6\sum_{i,j=1}^{n}(r_i r_j x_i x_j)^2$, with expectation $6\sum_{1\le i<j\le n} x_i^2 x_j^2$
  - Terms involving no repeated multipliers (such as $r_1 x_1 r_2 x_2 r_3 x_3 r_4 x_4$), with expectation 0.
- So

$$\mathbb{E}[Z^4] = \sum_{i=1}^{n} x_i^4 + 6 \sum_{1\le i<j\le n}^{n} x_i^2 x_j^2$$

AMS Algorithm: Bounding the Variance (con'td)

- Recall that

$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} x_i^2 = \|x\|_{\ell_2}^2$$

and

$$\mathbb{E}[Z^4] = \sum_{i=1}^{n} x_i^4 + 6 \sum_{1 \leq i < j \leq n} x_i^2 x_j^2$$

AMS Algorithm: Bounding the Variance (con'td)

- Recall that
$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} x_i^2 = \|x\|_{\ell_2}^2$$
and
$$\mathbb{E}[Z^4] = \sum_{i=1}^{n} x_i^4 + 6 \sum_{1 \le i < j \le n} x_i^2 x_j^2$$
- So
$$\begin{aligned}
\mathrm{Var}[Z^2] &= \mathbb{E}[Z^4] - \left(\mathbb{E}[Z^2]\right)^2 = \mathbb{E}[Z^4] - \|x\|_{\ell_2}^4 \\
&= \sum_{i=1}^{n} x_i^4 + 6 \sum_{1 \le i < j \le n} x_i^2 x_j^2 - \left(\sum_{i=1}^{n} x_i^2\right)^2 \\
&= \sum_{i=1}^{n} x_i^4 + 6 \sum_{1 \le i < j \le n} x_i^2 x_j^2 - \left(\sum_{i=1}^{n} x_i^4 + 2 \sum_{1 \le i < j \le n} x_i^2 x_j^2\right) \\
&= 4 \sum_{1 \le i < j \le n} x_i^2 x_j^2 \le 2 \left(\sum_{i=1}^{n} x_i^2\right)^2 = 2\|x\|_{\ell_2}^2.
\end{aligned}$$

AMS Algorithm: Completing the Analysis

- We have an estimator $Z^2 \approx \|x\|_{\ell_2}^2$, with $E[Z^2] = \|x\|_{\ell_2}^2$ and $\sigma = \mathrm{Var}[Z^2] \leq 2\|x\|_{\ell_2}^2$.

AMS Algorithm: Completing the Analysis

- We have an estimator $Z^2 \approx \|x\|_{\ell_2}^2$, with $E[Z^2] = \|x\|_{\ell_2}^2$ and $\sigma = \mathrm{Var}[Z^2] \leq 2\|x\|_{\ell_2}^2$.
- Apply Chebyshev inequality

$$\mathbb{P}[|\mathbb{E}[Y] - Y| \geq c\sigma] \leq \frac{1}{c^2} \qquad \forall\, c > 0$$

to find

$$\mathbb{P}\left[\left|\mathbb{E}[Z^2] - \|x\|_{\ell_2}^2\right| \geq c\sqrt{2}\|x\|_{\ell_2}^2\right] \leq \frac{1}{c^2}$$

AMS Algorithm: Completing the Analysis

- We have an estimator $Z^2 \approx \|x\|_{\ell_2}^2$, with $E[Z^2] = \|x\|_{\ell_2}^2$ and $\sigma = \text{Var}[Z^2] \leq 2\|x\|_{\ell_2}^2$.
- Apply Chebyshev inequality

$$\mathbb{P}[|\mathbb{E}[Y] - Y| \geq c\sigma] \leq \frac{1}{c^2} \qquad \forall\, c > 0$$

to find

$$\mathbb{P}\left[\left|\mathbb{E}[Z^2] - \|x\|_{\ell_2}^2\right| \geq c\sqrt{2}\|x\|_{\ell_2}^2\right] \leq \frac{1}{c^2}$$

- Problem: This gives a lousy estimator if failure probability δ is small. For instance, if $\delta = \frac{1}{3}$, must choose $c = 3$, finding

$$\mathbb{P}\left[\left|\mathbb{E}[Z^2] - \|x\|_{\ell_2}^2\right| \geq 3\sqrt{2}\|x\|_{\ell_2}^2\right] \leq \frac{1}{9}$$

But $\mathbb{E}[Z^2] \geq 0$, so this is worse than natural bound.

AMS+ Algorithm

- Run AMS k times, getting $Z_1, Z_2, \ldots, Z_k$.

AMS+ Algorithm

- Run AMS k times, getting $Z_1, Z_2, \ldots, Z_k$.
- Estimator is

$$Y = \frac{1}{k}\sum_{j=1}^{k} Z_j^2$$

AMS+ Algorithm

- Run AMS k times, getting $Z_1, Z_2, \ldots, Z_k$.
- Estimator is

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

- We then have

$$\mathbb{E}[Y] = \frac{1}{k} \sum_{j=1}^{k} \mathbb{E}[Z_j^2] = \frac{1}{k} \cdot k \|x\|_{\ell_2}^2 = \|x\|_{\ell_2}^2.$$

AMS+ Algorithm

- Run AMS k times, getting $Z_1, Z_2, \ldots, Z_k$.
- Estimator is

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

- We then have

$$\mathbb{E}[Y] = \frac{1}{k} \sum_{j=1}^{k} \mathbb{E}[Z_j^2] = \frac{1}{k} \cdot k \|x\|_{\ell_2}^2 = \|x\|_{\ell_2}^2.$$

- Moreover,

$$\mathrm{Var}[Y] = \frac{1}{k^2} \sum_{j=1}^{k} \mathrm{Var}[Z_j^2] \leq \frac{2}{k} \|x\|_{\ell_2}^2,$$

so that Chebyshev's inequality yields

$$\mathbb{P}\left[\left| \mathbb{E}[Y] - \|x\|_{\ell_2}^2 \right| \leq c\sqrt{2/k} \|x\|_{\ell_2}^2 \right] < 1/c^2$$

AMS+Algorithm (cont'd)

- Run AMS k times, getting

$$Z_j = \sum_{i=1}^{n} r_{j,i} x_i \qquad (1 \leq j \leq k)$$

AMS+Algorithm (cont'd)

- Run AMS k times, getting

$$Z_j = \sum_{i=1}^{n} r_{j,i} x_i \qquad (1 \le j \le k)$$

- Use estimator

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

AMS+Algorithm (cont'd)

- Run AMS k times, getting

$$Z_j = \sum_{i=1}^{n} r_{j,i} x_i \qquad (1 \le j \le k)$$

- Use estimator

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

- We then have

$$\mathbb{E}[Y] = \|x\|_{\ell_2}^2.$$

  and

$$\mathbb{P}\left[\left|\mathbb{E}[Y] - \|x\|_{\ell_2}^2\right| \le c\sqrt{2/k}\|x\|_{\ell_2}^2\right] < 1/c^2$$

AMS+Algorithm (cont'd)

- Run AMS k times, getting

$$Z_j = \sum_{i=1}^{n} r_{j,i} x_i \qquad (1 \leq j \leq k)$$

- Use estimator

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

- We then have

$$\mathbb{E}[Y] = \|x\|_{\ell_2}^2.$$

and

$$\mathbb{P}\left[\left|\mathbb{E}[Y] - \|x\|_{\ell_2}^2\right| \leq c\sqrt{2/k}\|x\|_{\ell_2}^2\right] < 1/c^2$$

- Set c to be a constant and $k = O(1/\varepsilon^2)$, get a $(1 \pm \varepsilon)$-bit approximation with constant probability.

AMS+Algorithm (cont'd)

- Run AMS k times, getting

$$Z_j = \sum_{i=1}^{n} r_{j,i} x_i \qquad (1 \leq j \leq k)$$

- Use estimator

$$Y = \frac{1}{k} \sum_{j=1}^{k} Z_j^2$$

- We then have

$$\mathbb{E}[Y] = \|x\|_{\ell_2}^2.$$

  and

$$\mathbb{P}\left[\left|\mathbb{E}[Y] - \|x\|_{\ell_2}^2\right| \leq c\sqrt{2/k}\|x\|_{\ell_2}^2\right] < 1/c^2$$

- Set c to be a constant and $k = O(1/\varepsilon^2)$, get a $(1 \pm \varepsilon)$-bit approximation with constant probability.
- Space usage: $O(\log(mn)/\varepsilon^2)$ bits (not counting the r_i)

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:
  - Maintain a "linear sketch" vector $Z = Rx$, where $R = [r_{j,i}]$

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:
  - Maintain a "linear sketch" vector $Z = Rx$, where $R = [r_{j,i}]$
  - Easily handles (say) two data streams, since $R(x + \tilde{x}) = Rx + R\tilde{x}$.

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:
  - Maintain a "linear sketch" vector $Z = Rx$, where $R = [r_{j,i}]$
  - Easily handles (say) two data streams, since $R(x + \tilde{x}) = Rx + R\tilde{x}$.
  - Estimate $\|x\|_{\ell_2}^2$ by $\|Rx\|_{\ell_2}^2 / k$.

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:
  - Maintain a “linear sketch” vector $Z = Rx$, where $R = [r_{j,i}]$
  - Easily handles (say) two data streams, since $R(x + \tilde{x}) = Rx + R\tilde{x}$.
  - Estimate $\|x\|_{\ell_2}^2$ by $\|Rx\|_{\ell_2}^2 / k$.
  - Reduction of dimension, use Rx (with k elements) instead of x (with n elements)

AMS: Final Comments

- Only needed 4-wise independence of $r_1, \ldots, r_n$.
  Can generate using $O(\log m)$ random bits, using 4-wise independent hash functions.
- What we did:
  - Maintain a "linear sketch" vector $Z = Rx$, where $R = [r_{j,i}]$
  - Easily handles (say) two data streams, since $R(x + \tilde{x}) = Rx + R\tilde{x}$.
  - Estimate $\|x\|_{\ell_2}^2$ by $\|Rx\|_{\ell_2}^2 / k$.
  - Reduction of dimension, use Rx (with k elements) instead of x (with n elements)
- Error bound too loose to be useful for small δ:
  For $c = O(1/\sqrt{\delta})$, need $k = O\big(1/(\delta\varepsilon^2)\big)$, linear in δ.
  That's because we only used second moment.

JL Algorithm

Reference: W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space." *Contemporary Mathematics*, 26(1): 189–206, 1984.

- Normal distribution $\mathcal{N}(\mu, \sigma)$ has density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \forall\, x \in \mathbb{R}$$

JL Algorithm

Reference: W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space." *Contemporary Mathematics*, 26(1): 189–206, 1984.

- Normal distribution $\mathcal{N}(\mu, \sigma)$ has density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \forall x \in \mathbb{R}$$

- Basic facts for normal distribution:

JL Algorithm

Reference: W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space." *Contemporary Mathematics*, 26(1): 189–206, 1984.

- Normal distribution $\mathcal{N}(\mu, \sigma)$ has density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \forall\, x \in \mathbb{R}$$

- Basic facts for normal distribution:
  - Mean μ, variance σ^2.

JL Algorithm

Reference: W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space." *Contemporary Mathematics*, 26(1): 189–206, 1984.

- Normal distribution $\mathcal{N}(\mu, \sigma)$ has density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \forall x \in \mathbb{R}$$

- Basic facts for normal distribution:
  - Mean μ, variance σ^2.
  - If X and Y are independent identically distributed random variables, then $X + Y$ is normally distributed, with $\mathrm{Var}[X] + \mathrm{Var}[Y]$

JL Algorithm

Reference: W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space." *Contemporary Mathematics*, 26(1): 189–206, 1984.

- Normal distribution $\mathcal{N}(\mu, \sigma)$ has density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \forall x \in \mathbb{R}$$

- Basic facts for normal distribution:
  - Mean μ, variance σ^2.
  - If X and Y are independent identically distributed random variables, then $X + Y$ is normally distributed, with $\text{Var}[X] + \text{Var}[Y]$
  - $\text{Var}[cX] = c^2 \, \text{Var}[X]$

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0, 1)$.

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0,1)$.
- Maintain $Z = Rx$ instead of x.

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0,1)$.
- Maintain $Z = Rx$ instead of x.
- Estimate $\|x\|_{\ell_2}^2$ by $Y = \|Rx\|_{\ell_2}^2 / k$.

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0,1)$.
- Maintain $Z = Rx$ instead of x.
- Estimate $\|x\|_{\ell_2}^2$ by $Y = \|Rx\|_{\ell_2}^2 / k$.
- As before, find $\mathbb{E}[Y] = \|x\|_{\ell_2}^2$, since

$$\mathbb{E}\left[\frac{1}{k}\|Rx\|_{\ell_2}^2\right] = \frac{1}{k}\mathbb{E}[x^T R^T Rx] = \frac{1}{k}x^T \mathbb{E}[R^T R]x$$
$$= \frac{1}{k}x^T \operatorname{diag}[k, k, \ldots, k]x = \|x\|_{\ell_2}^2$$

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0,1)$.
- Maintain $Z = Rx$ instead of x.
- Estimate $\|x\|_{\ell_2}^2$ by $Y = \|Rx\|_{\ell_2}^2/k$.
- As before, find $\mathbb{E}[Y] = \|x\|_{\ell_2}^2$, since

$$\mathbb{E}\left[\frac{1}{k}\|Rx\|_{\ell_2}^2\right] = \frac{1}{k}\mathbb{E}[x^T R^T R x] = \frac{1}{k}x^T \mathbb{E}[R^T R]x$$
$$= \frac{1}{k}x^T \operatorname{diag}[k, k, \ldots, k]x = \|x\|_{\ell_2}^2$$

- Moreover, there exists $C > 0$ such that

$$\mathbb{P}[|Y - \|x\|_{\ell_2}^2| > \varepsilon\|x\|_{\ell_2}^2] \leq \exp(-C\varepsilon^2 k) \qquad \text{for small } \varepsilon > 0$$

JL Algorithm (cont'd)

Basic idea:

- Let $R \in \mathbb{R}^{k \times m}$, where each $r_{i,j}$ is i.i.d. random variable from $\mathcal{N}(0,1)$.
- Maintain $Z = Rx$ instead of x.
- Estimate $\|x\|_{\ell_2}^2$ by $Y = \|Rx\|_{\ell_2}^2/k$.
- As before, find $\mathbb{E}[Y] = \|x\|_{\ell_2}^2$, since

$$\mathbb{E}\left[\frac{1}{k}\|Rx\|_{\ell_2}^2\right] = \frac{1}{k}\mathbb{E}[x^T R^T Rx] = \frac{1}{k}x^T \mathbb{E}[R^T R]x$$
$$= \frac{1}{k}x^T \operatorname{diag}[k, k, \ldots, k]x = \|x\|_{\ell_2}^2$$

- Moreover, there exists $C > 0$ such that

$$\mathbb{P}[|Y - \|x\|_{\ell_2}^2| > \varepsilon\|x\|_{\ell_2}^2] \leq \exp(-C\varepsilon^2 k) \qquad \text{for small } \varepsilon > 0$$

- Set $k = O(1/\varepsilon^2 \log(1/\delta))$ to get $1 \pm \varepsilon$ approximation with probability $1 - \delta$.

JL Algorithm: Final Comments

- Can use k-wise independence to generate r_is, but much messier than for AMS

JL Algorithm: Final Comments

- Can use k-wise independence to generate r_is, but much messier than for AMS
- Time to compute sketch vector Z from x is $O(k)$, bad if k is large.

JL Algorithm: Final Comments

- Can use k-wise independence to generate r_is, but much messier than for AMS
- Time to compute sketch vector Z from x is $O(k)$, bad if k is large.
- Fast JL, sparse JL: reduce updating time