Tracking the ℓ_2 Norm with Constant Update Time

Chi-Ning Chou Zhixian Lei Preetum Nakkiran

Harvard University

APPROX 2019

Streaming Algorithms

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \ a_{t'} = i \} \right| .$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \; a_{t'} = i \} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\{t' \in [t] : \; a_{t'} = i\}| \, .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \; a_{t'} = i \} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

$\boxed{1}$

$$f^{(1)} = \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \dots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\{t' \in [t] : \ a_{t'} = i\}| \,.$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 **2**

$$f^{(2)} = \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \ a_{t'} = i \} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 **2** **4**

$$f^{(3)} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\{t' \in [t] : \; a_{t'} = i\}| \, .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 2 4 2

$$f^{(4)} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \; a_{t'} = i \} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 2 4 2 5

$$f^{(5)} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\{t' \in [t] : \; a_{t'} = i\}| \, .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 2 4 2 5 2

$$f^{(6)} = \begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left|\,\{t' \in [t] :\ a_{t'} = i\}\,\right|.$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 2 4 2 5 2 1

$$f^{(7)} = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\,\{t' \in [t] :\; a_{t'} = i\}\,|\,.$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1	**2**	**4**	**2**	**5**	**2**	**1**	**1**

$$f^{(8)} = \begin{bmatrix} 3 \\ 3 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\,\{t' \in [t] : \; a_{t'} = i\}\,|\,.$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1	**2**	**4**	**2**	**5**	**2**	**1**	**1**	**5**

$$f^{(9)} = \begin{bmatrix} 3 \\ 3 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{ t' \in [t] : \ a_{t'} = i \} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Example:** $n = 5$ and $m = 10$.

1 2 4 2 5 2 1 1 5 2

$$f^{(10)} = \begin{bmatrix} 3 \\ 4 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{t' \in [t] : \; a_{t'} = i\} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Applications**: Database optimization, network traffic etc.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\{t' \in [t] : \; a_{t'} = i\}| \, .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = t.

- **Applications**: Database optimization, network traffic etc.

- **Goal**: Randomized algorithms using sublinear space.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

Deterministic algorithm: $\Theta(\min\{m, n\})$ space.

- **Frequency Vector**: For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = |\,\{t' \in [t] :\ a_{t'} = i\}\,|\,.$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Applications**: Database optimization, network traffic etc.

- **Goal**: Randomized algorithms using sublinear space.

Streaming Algorithms

- **Input**: A stream of inputs from the alphabet set.

  Example: $a_1, a_2, \ldots, a_m \in [n]$.

Deterministic algorithm: $\Theta(\min\{m, n\})$ space.

- **Output**: Some statistics of the inputs.

  Example: # of distinct elements in the input stream.

- **Frequency Vector:** For each $t \in [m], i \in [n]$, define

$$f_i^{(t)} = \left| \{t' \in [t] : \ a_{t'} = i\} \right| .$$

  Example: ℓ_0 norm = # of distinct elements; ℓ_1 norm = *t*.

- **Applications**: Database optimization, network traffic etc.

- **Goal**: Randomized algorithms using sublinear space.

Faster time!?

ℓ_2 Estimation

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.
- (ϵ, δ)**-One-shot estimation**: Output σ_m s.t.

$$\Pr\left[\left|\sigma_m - \|f^{(m)}\|_2^2\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq \delta\,.$$

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.
- (ϵ, δ)**-One-shot estimation**: Output σ_m s.t.

$$\Pr\left[\left|\sigma_m - \|f^{(m)}\|_2^2\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq \delta\,.$$

- (ϵ, δ)**-Weak tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t\in[m]}\left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq \delta\,.$$

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.
- (ϵ, δ)**-One-shot estimation**: Output σ_m s.t.

$$\Pr\left[\left|\sigma_m - \|f^{(m)}\|_2^2\right| > \epsilon \|f^{(m)}\|_2^2\right] \le \delta\,.$$

- (ϵ, δ)**-Weak tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon \|f^{(m)}\|_2^2\right] \le \delta\,.$$

- (ϵ, δ)**-Strong tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon \|f^{(t)}\|_2^2\right] \le \delta\,.$$

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.
- (ϵ, δ)**-One-shot estimation**: Output σ_m s.t.

$$\Pr\left[\left|\sigma_m - \|f^{(m)}\|_2^2\right| > \epsilon \|f^{(m)}\|_2^2\right] \leq \delta\,.$$

- (ϵ, δ)**-Weak tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon \|f^{(m)}\|_2^2\right] \leq \delta\,.$$

- (ϵ, δ)**-Strong tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon \|f^{(t)}\|_2^2\right] \leq \delta\,.$$

Strong tracking => Weak tracking => One-shot

ℓ_2 Estimation

- **Goal**: Estimating the ℓ_2 norm of the frequency vector in sublinear space.
- (ϵ, δ)**-One-shot estimation**: Output σ_m s.t.

$$\Pr\left[\left|\sigma_m - \|f^{(m)}\|_2^2\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq \delta\,.$$

- (ϵ, δ)**-Weak tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t\in[m]}\left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq \delta\,.$$

- (ϵ, δ)**-Strong tracking**: Output $\sigma_1, \sigma_2, \ldots, \sigma_m$ s.t.

$$\Pr\left[\exists_{t\in[m]}\left|\sigma_t - \|f^{(t)}\|_2^2\right| > \epsilon\|f^{(t)}\|_2^2\right] \leq \delta\,.$$

Strong tracking => Weak tracking => One-shot

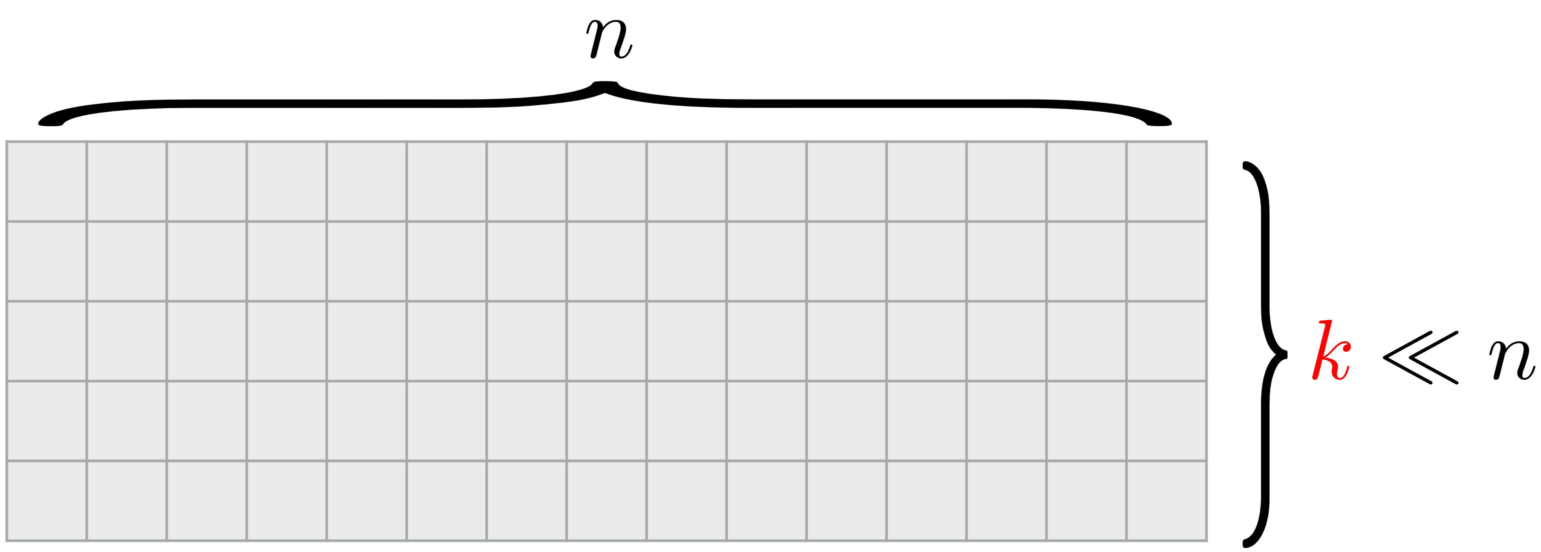
Linear Sketch

Linear Sketch

- **Linear sketch** is a special class of streaming algorithms.

Linear Sketch

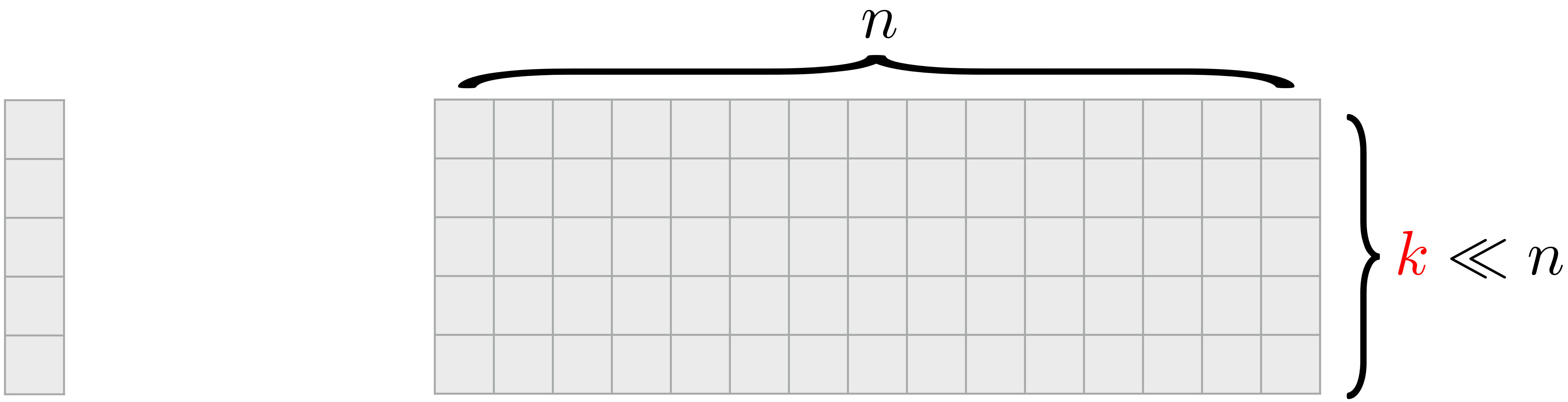
- **Linear sketch** is a special class of streaming algorithms.



Sketching matrix Π

Linear Sketch

- **Linear sketch** is a special class of streaming algorithms.



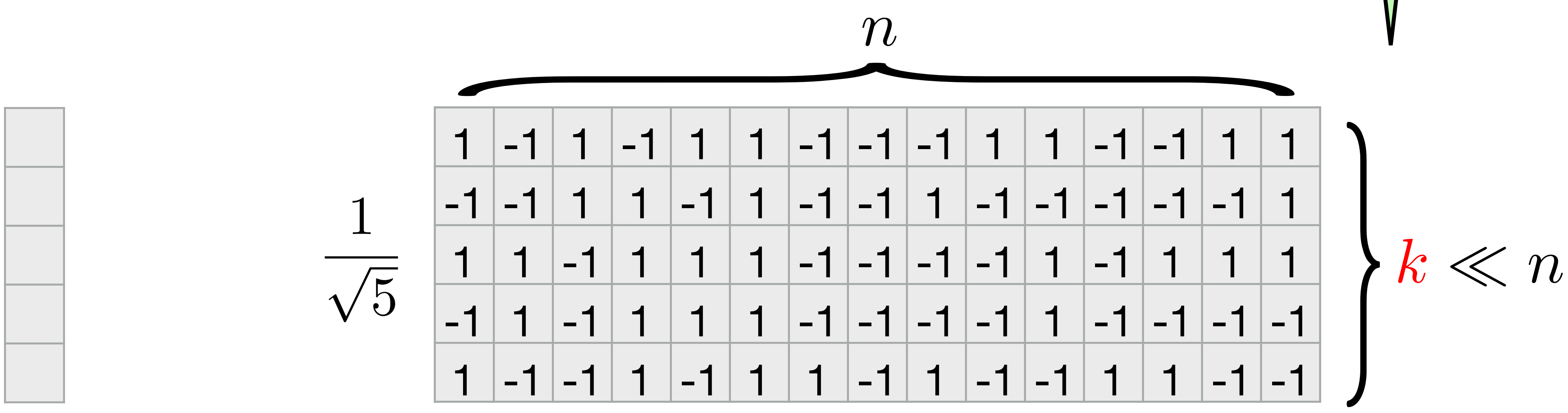
Sketching vector $\Pi f^{(t)}$

Sketching matrix Π

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\frac{1}{\sqrt{5}} \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n}\right\} k \ll n$$

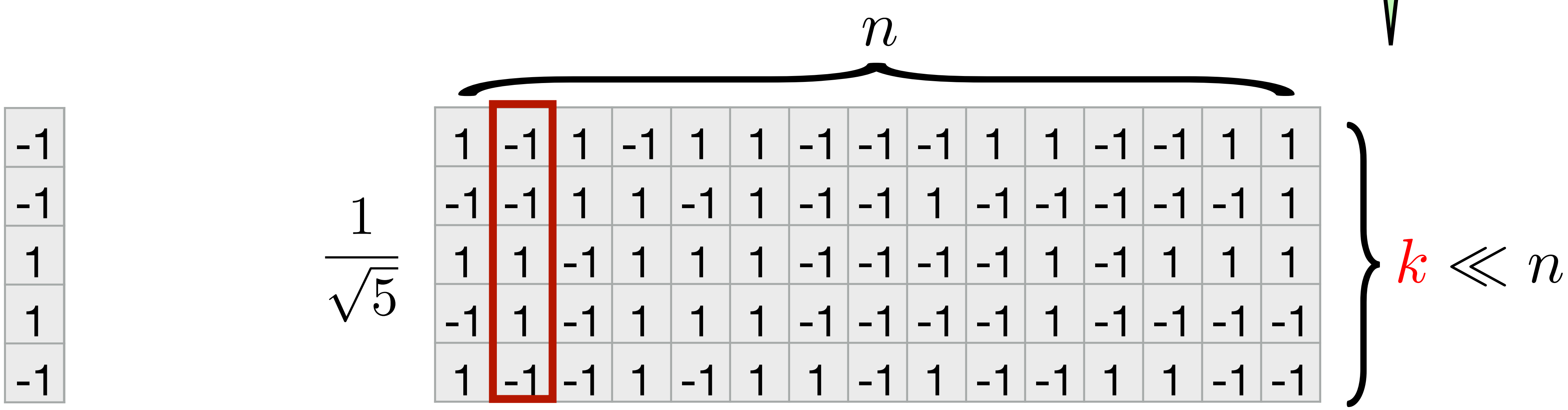
Sketching vector $\Pi f^{(t)}$

Sketching matrix Π

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} \qquad \frac{1}{\sqrt{5}} \overbrace{\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}}^{n} \Bigg\} \, k \ll n$$

Sketching vector $\Pi f^{(t)}$

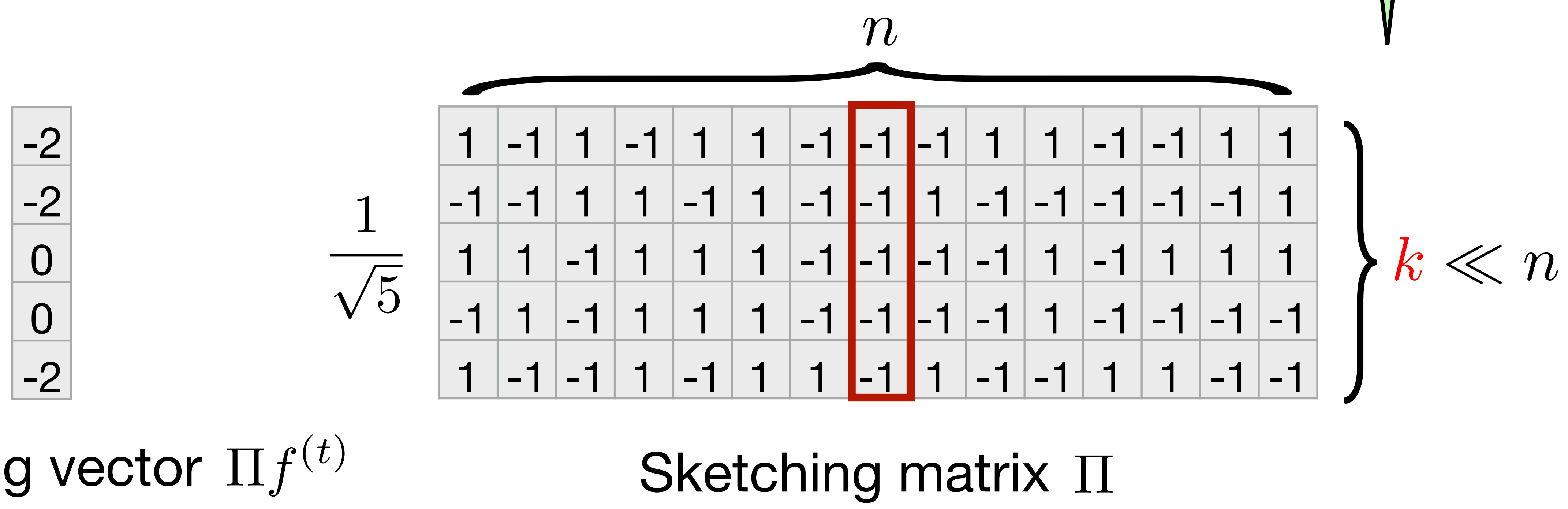
Sketching matrix Π

2

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



-2
-2
0
0
-2

Sketching vector $\Pi f^{(t)}$

$$\frac{1}{\sqrt{5}}$$

1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	1	1
-1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	1
1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	1	1	1
-1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	-1	-1
1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1

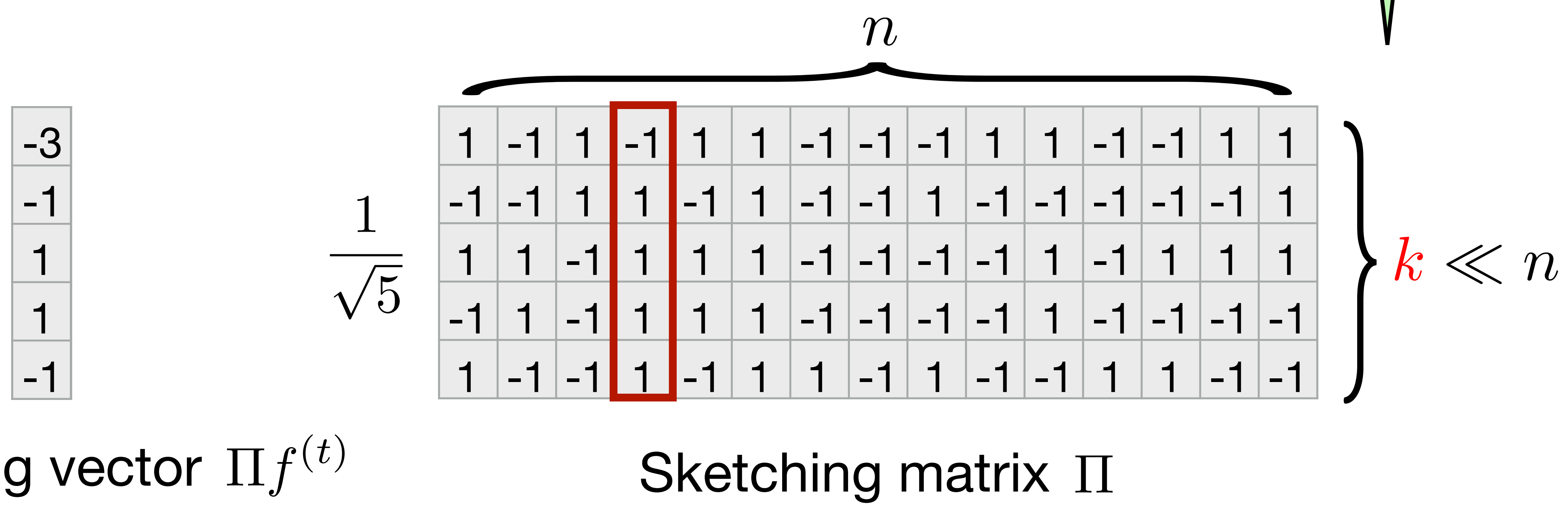
Sketching matrix Π

2 **8**

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\begin{pmatrix} -3 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} \qquad \frac{1}{\sqrt{5}} \overbrace{\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}}^{n} \left.\vphantom{\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}}\right\} k \ll n$$

Sketching vector $\Pi f^{(t)}$

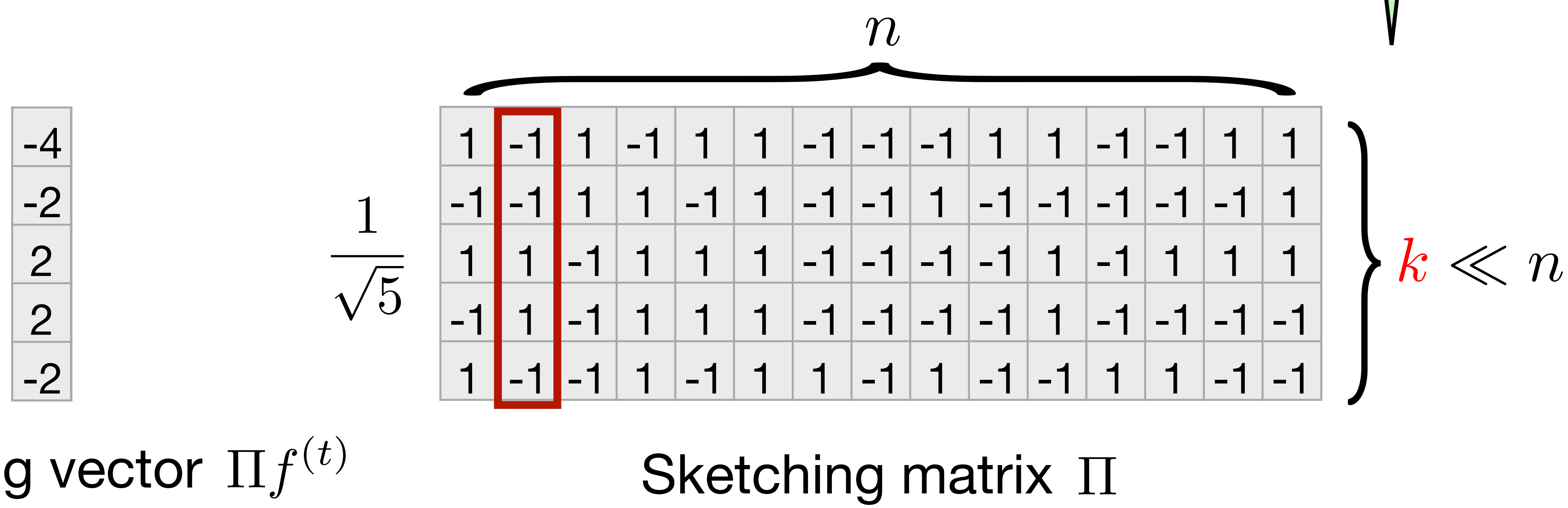
Sketching matrix Π

2 **8** **4**

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\begin{bmatrix}-4\\-2\\2\\2\\-2\end{bmatrix} \qquad \frac{1}{\sqrt{5}} \overbrace{\left[\begin{array}{c|c|ccccccccccccc} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{array}\right]}^{n} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} {\color{red}k} \ll n$$

Sketching vector $\Pi f^{(t)}$

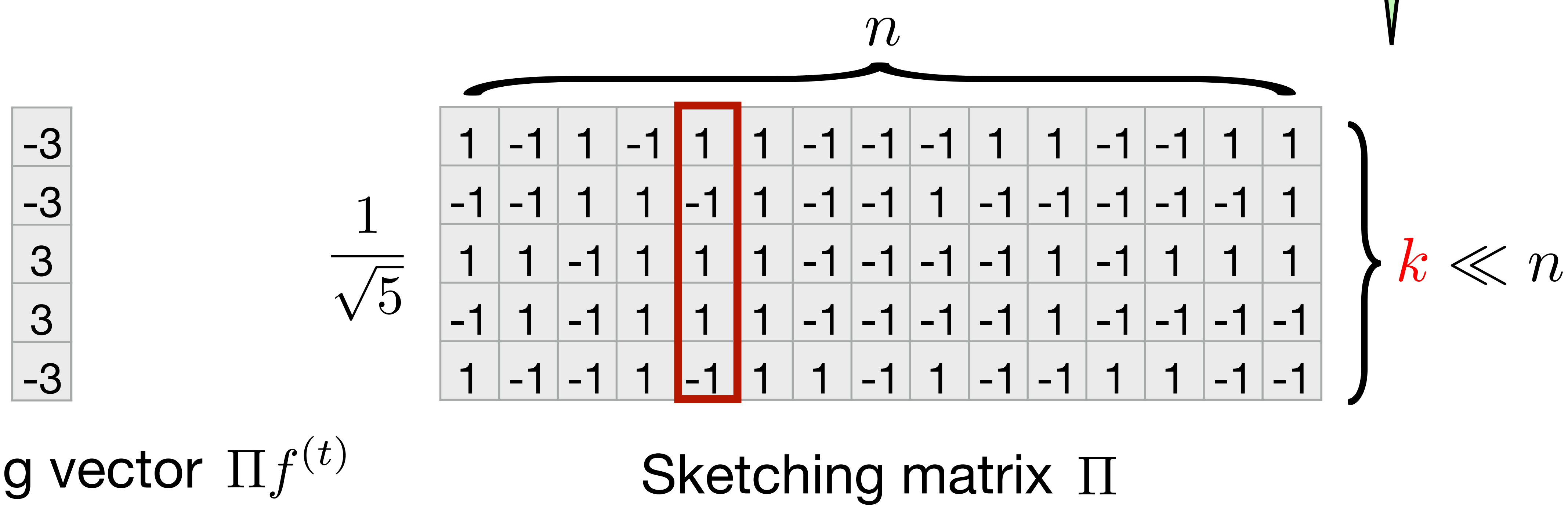
Sketching matrix Π

2 **8** **4** **2**

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\begin{bmatrix} -3 \\ -3 \\ 3 \\ 3 \\ -3 \end{bmatrix} \qquad \frac{1}{\sqrt{5}} \overbrace{\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}}^{n} \left.\vphantom{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}\right\} k \ll n$$

Sketching vector $\Pi f^{(t)}$

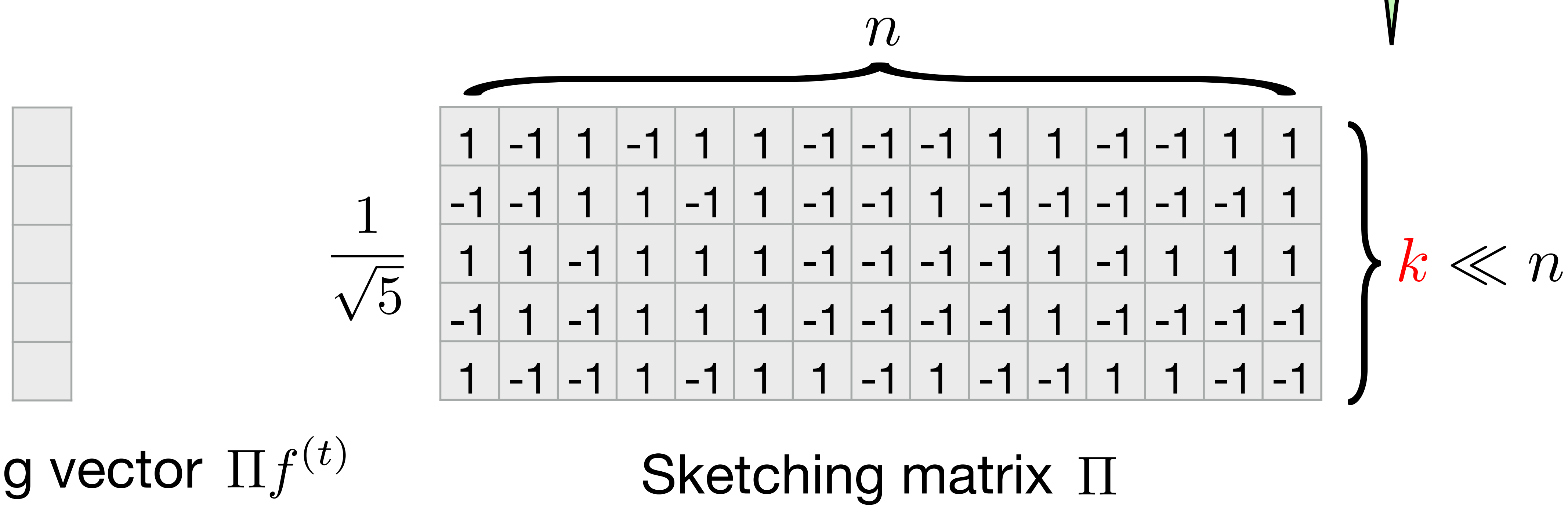
Sketching matrix Π

2	**8**	**4**	**2**	**5**

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\frac{1}{\sqrt{5}} \overbrace{\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}}^{n} \left.\vphantom{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}\right\} k \ll n$$

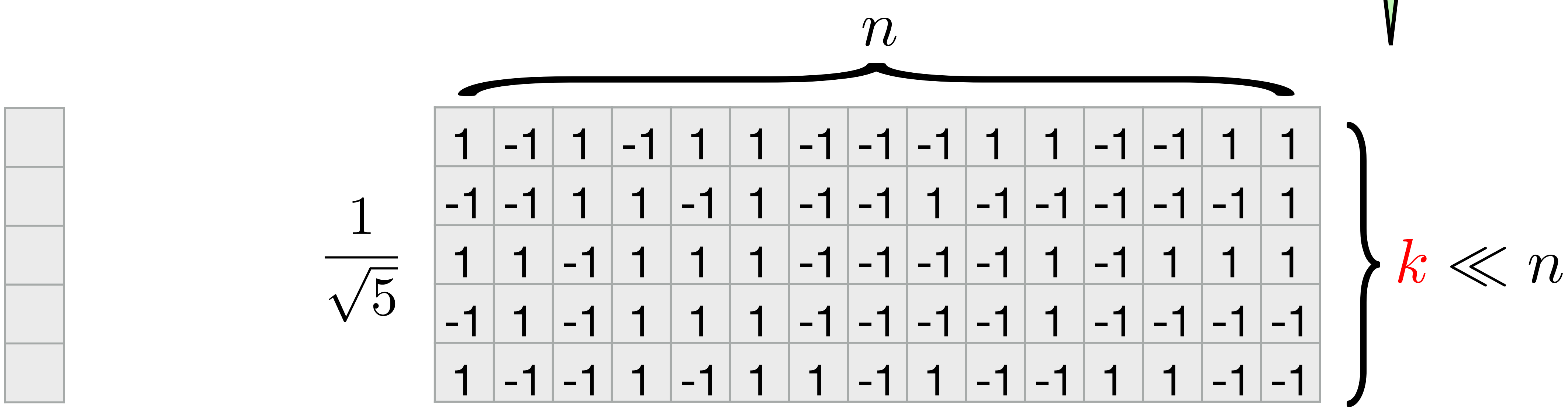
Sketching vector $\Pi f^{(t)}$

Sketching matrix Π

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\frac{1}{\sqrt{5}}\overbrace{\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}}^{n}\left.\vphantom{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}\right\} k \ll n$$

Sketching vector $\Pi f^{(t)}$

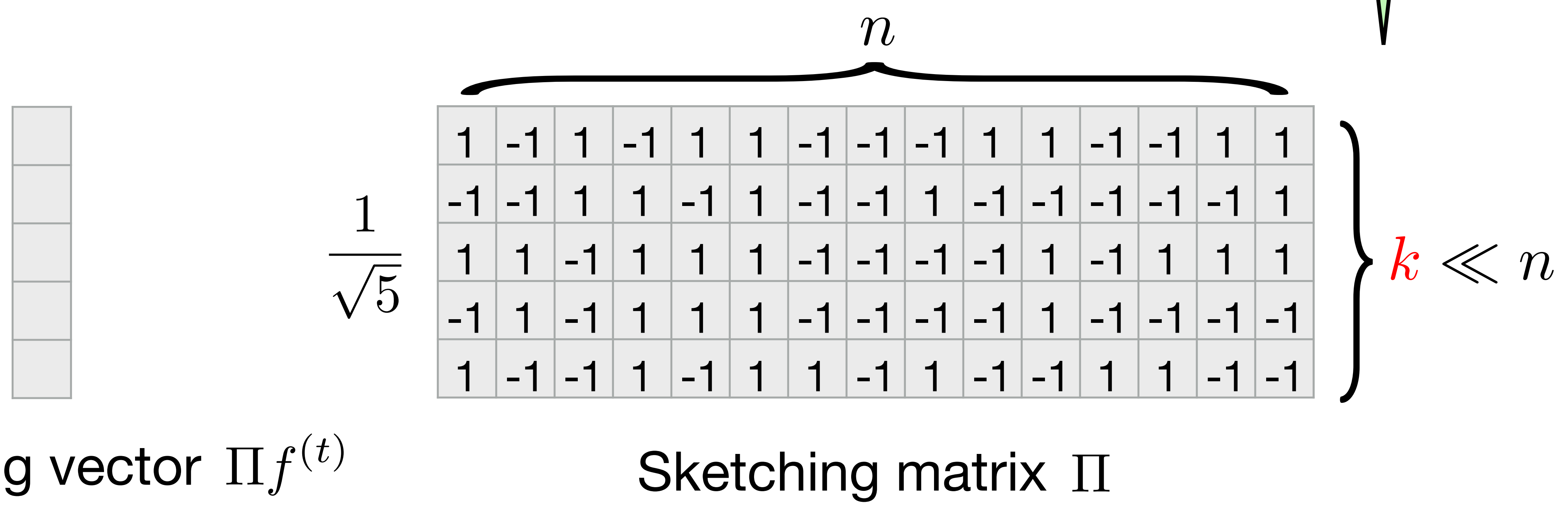
Sketching matrix Π

- **Space complexity**: $\begin{cases} O(kn) & \text{, truly random} \\ O(k \log n) & \text{, pseudo-random} \end{cases}$

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



$$\frac{1}{\sqrt{5}} \overbrace{\left.\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}\right\}}^{n} k \ll n$$

Sketching vector $\Pi f^{(t)}$

Sketching matrix Π

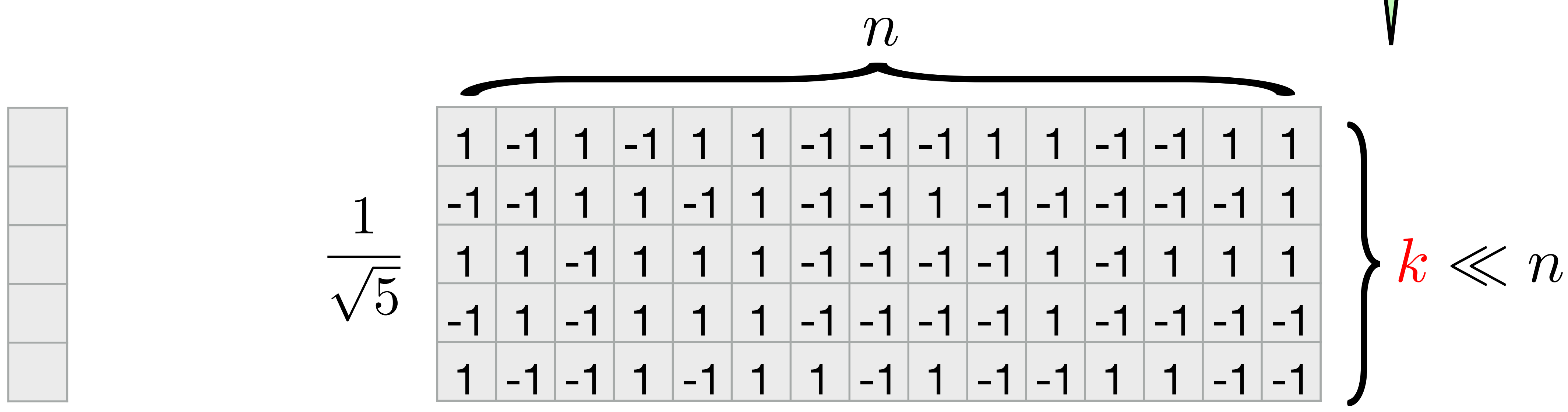
- **Space complexity**: $\begin{cases} O(kn) & \text{, truly random} \\ O(k \log n) & \text{, pseudo-random} \end{cases}$

Can be even better

Linear Sketch

AMS Sketch
[Alon-Matias-Szegedy 96]

- **Linear sketch** is a special class of streaming algorithms.



1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	1	1
-1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	1
1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	1	1	1
-1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	-1	-1
1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1

Sketching vector $\Pi f^{(t)}$

Sketching matrix Π

- **Space complexity**: $\begin{cases} O(kn) & \text{, truly random} \\ O(k \log n) & \text{, pseudo-random} \end{cases}$

Can be even better

- **AMS sketch:** $k = O(\epsilon^{-2})$ for one-shot [Alon-Matias-Szegedy 96] and for weak tracking [Braverman-Chestnut-Ivkin-Nelson-Wang-Woodruff 17].

Update Time

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline
-1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline
1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline
-1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline
\end{array}}^{n}\right\} k \ll n$$

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n}\right\} k \ll n$$

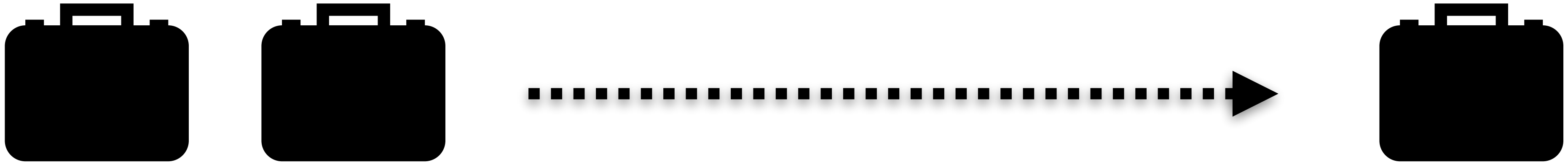
- **Application:** Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n}\right\} k \ll n$$

- **Application:** Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]

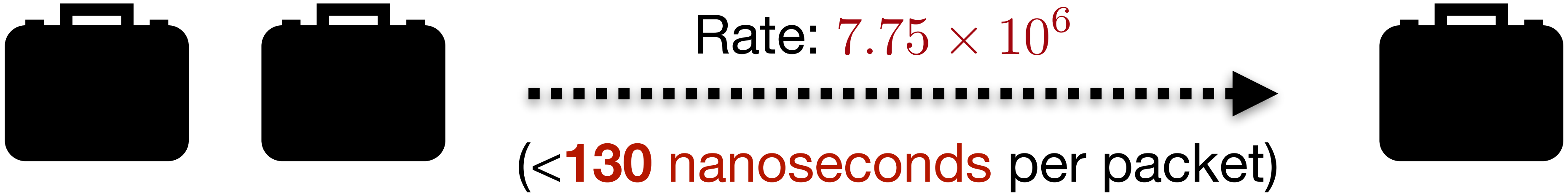


Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} k \ll n$$

- **Application:** Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]



Rate: 7.75×10^6

(<**130** nanoseconds per packet)

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n}\right\} k \ll n$$

- When ϵ is small, AMS sketch is slow. 🐌

Update Time

- **Update time complexity** for a linear sketch algorithm is the number of field operations needed in each update.
- E.g., AMS sketch has $\Theta(k) = \Theta(\epsilon^{-2})$ update time complexity.

$$\Pi = \left.\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ \hline -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ \hline 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ \hline -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \hline \end{array}}^{n}\right\} k \ll n$$

- When ϵ is small, AMS sketch is slow. 🐌

Q: Is AMS Sketch optimal in update time complexity?

Faster One-Shot Estimation

Faster One-Shot Estimation

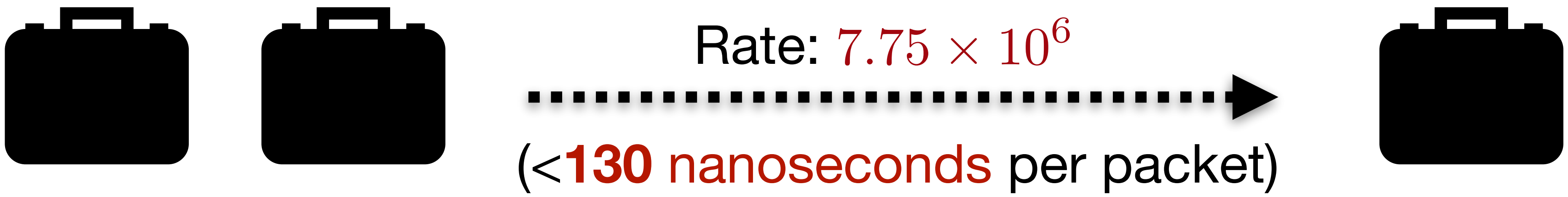
- [Dasgupta-Kumar-Sarlós 10] and [Kane-Nelson 14] showed that sparse JL achieves one-shot with $O(\epsilon^{-1})$ update time.

Faster One-Shot Estimation

- [Dasgupta-Kumar-Sarlós 10] and [Kane-Nelson 14] showed that sparse JL achieves one-shot with $O(\epsilon^{-1})$ update time.
- [Thorup-Zhang 12] showed that CountSketch (proposed by [Charikar-Chen-Farach-Colton 02]) achieves one-shot with $O(1)$ update time.

Faster One-Shot Estimation

- [Dasgupta-Kumar-Sarlós 10] and [Kane-Nelson 14] showed that sparse JL achieves one-shot with $O(\epsilon^{-1})$ update time.
- [Thorup-Zhang 12] showed that CountSketch (proposed by [Charikar-Chen-Farach-Colton 02]) achieves one-shot with $O(1)$ update time.
- Application: Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]

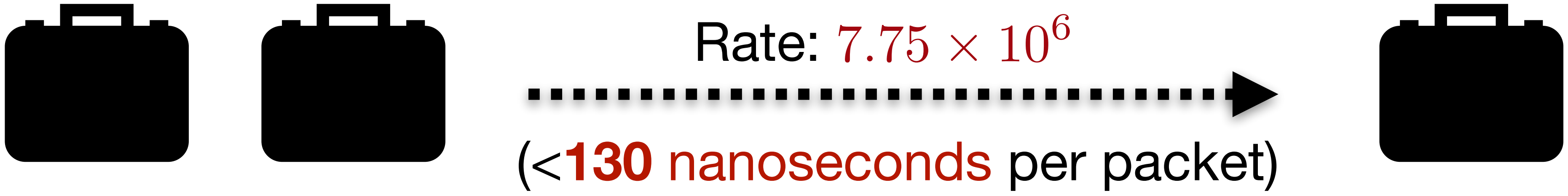


Rate: 7.75×10^6

(<**130** nanoseconds per packet)

Faster One-Shot Estimation

- [Dasgupta-Kumar-Sarlós 10] and [Kane-Nelson 14] showed that sparse JL achieves one-shot with $O(\epsilon^{-1})$ update time.
- [Thorup-Zhang 12] showed that CountSketch (proposed by [Charikar-Chen-Farach-Colton 02]) achieves one-shot with $O(1)$ update time.
- Application: Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]



Rate: 7.75×10^6

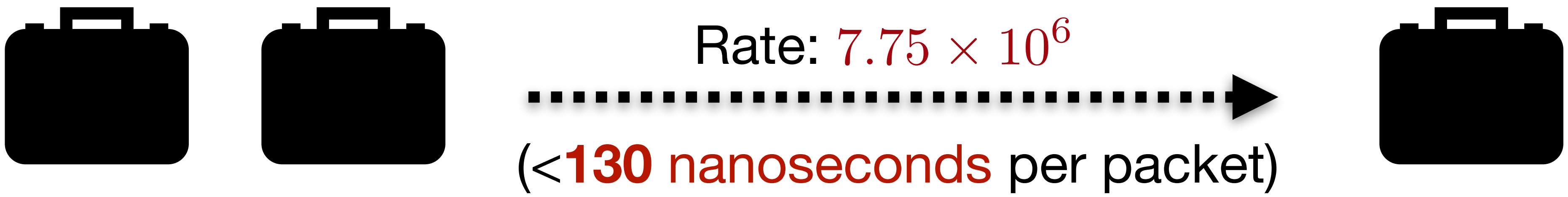
(<**130** nanoseconds per packet)

[Thorup-Zhang 12] showed that CountSketch improves AMS sketch from **182** nanoseconds to **30** nanoseconds!

Faster One-Shot Estimation

Only for **one-shot**

- [Dasgupta-Kumar-Sarlós 10] and [Kane-Nelson 14] showed that sparse JL achieves one-shot with $O(\epsilon^{-1})$ update time.
- [Thorup-Zhang 12] showed that CountSketch (proposed by [Charikar-Chen-Farach-Colton 02]) achieves one-shot with $O(1)$ update time.
- Application: Packet passing problem [Krishnamurthy-Sen-Zhang-Chen 03]



Rate: 7.75×10^6

(<**130** nanoseconds per packet)

[Thorup-Zhang 12] showed that CountSketch improves AMS sketch from **182** nanoseconds to **30** nanoseconds!

What About Faster Linear Sketch for Weak Tracking?

What About Faster Linear Sketch for Weak Tracking?

Known

- $O(1)$ time for one-shot
- $O(\epsilon^{-2})$ time for weak tracking

What About Faster Linear Sketch for Weak Tracking?

Known

- $O(1)$ time for one-shot
- $O(\epsilon^{-2})$ time for weak tracking

Unknown

- $O(1)$ time for weak tracking

CountSketch Provides Weak Tracking

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

Corollary (informal)

There is an $O(1)$ time algorithm provides $(\epsilon, 0.1)$-weak tracking.

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

- The first analysis for weak tracking with constant update time.

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

- The first analysis for weak tracking with constant update time.
- Using the median trick, there is a streaming algorithm provides (ϵ, δ)-weak tracking with $O(\log \delta^{-1})$ update time.

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

- The first analysis for weak tracking with constant update time.
- Using the median trick, there is a streaming algorithm provides (ϵ, δ)-weak tracking with $O(\log \delta^{-1})$ update time.
- The packet passing problem now has tracking guarantee.

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

- The first analysis for weak tracking with constant update time.
- Using the median trick, there is a streaming algorithm provides (ϵ, δ)-weak tracking with $O(\log \delta^{-1})$ update time.
- The packet passing problem now has tracking guarantee.

The rest of the talk will focus on the proof

CountSketch Provides Weak Tracking

Theorem (informal)

CountSketch with $O(\epsilon^{-2})$ rows provides $(\epsilon, 0.1)$-weak tracking.

- The first analysis for weak tracking with constant update time.
- Using the median trick, there is a streaming algorithm provides (ϵ, δ)-weak tracking with $O(\log \delta^{-1})$ update time.
- The packet passing problem now has tracking guarantee.

The rest of the talk will focus on the proof **sketch**.

CountSketch [Charikar-Chen-Farach-Colton 02]

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$$\Pi = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 \\
\hline
0 & \mathbf{-1} & \mathbf{1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
\hline
\mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} \\
\hline
\end{array}$$

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$$\Pi = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 \\
\hline
0 & \mathbf{-1} & \mathbf{1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
\hline
\mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} \\
\hline
\end{array}$$

[Thorup-Zhang 12] showed that CountSketch with $O(\epsilon^{-2})$ rows achieve one-shot estimation.

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$$\Pi = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 \\
\hline
0 & \mathbf{-1} & \mathbf{1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
\hline
\mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} \\
\hline
\end{array}$$

[Thorup-Zhang 12] showed that CountSketch with $O(\epsilon^{-2})$ rows achieve one-shot estimation.

- **Analysis**:

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$$\Pi = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 \\
\hline
0 & \mathbf{-1} & \mathbf{1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
\hline
\mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} \\
\hline
\end{array}$$

[Thorup-Zhang 12] showed that CountSketch with $O(\epsilon^{-2})$ rows achieve one-shot estimation.

- **Analysis:**
  - Obs: $\mathbb{E}\left[(\Pi^\top \Pi)_{ij}\right] = \mathbf{1}_{i=j}$.

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$\Pi =$

0	0	0	0	0	**1**	0	0	0	0	**1**	**-1**	0	0	0
0	**-1**	**1**	0	**-1**	0	0	0	0	0	0	0	0	**-1**	0
0	0	0	0	0	0	0	**-1**	**-1**	**-1**	0	0	**1**	0	0
-1	0	0	**1**	0	0	**-1**	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**-1**

[Thorup-Zhang 12] showed that CountSketch with $O(\epsilon^{-2})$ rows achieve one-shot estimation.

- **Analysis**:
  - Obs: $\mathbb{E}\left[(\Pi^\top\Pi)_{ij}\right] = \mathbf{1}_{i=j}$.
  - Expectation: $\mathbb{E}\left[\|\Pi f^{(m)}\|_2^2\right] = \mathbb{E}\left[\sum_{i,j\in[n]} (\Pi^\top\Pi)_{ij} f_i^{(m)} f_j^{(m)}\right] = \|f^{(m)}\|_2^2$.

CountSketch [Charikar-Chen-Farach-Colton 02]

- **Idea**: Exactly one non-zero entry in each column.

$$\Pi = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 \\ 0 & \mathbf{-1} & \mathbf{1} & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 \\ \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} \end{bmatrix}$$

[Thorup-Zhang 12] showed that CountSketch with $O(\epsilon^{-2})$ rows achieve one-shot estimation.

- **Analysis**:
  - Obs: $\mathbb{E}\left[(\Pi^\top \Pi)_{ij}\right] = \mathbf{1}_{i=j}$.
  - Expectation: $\mathbb{E}\left[\|\Pi f^{(m)}\|_2^2\right] = \mathbb{E}\left[\sum_{i,j\in[n]} (\Pi^\top \Pi)_{ij} f_i^{(m)} f_j^{(m)}\right] = \|f^{(m)}\|_2^2$.
  - Apply Chebyshev's inequality.

Intuition for Weak Tracking

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

$\left(\epsilon, \frac{\delta}{m}\right)$-one-shot

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

$$\left(\epsilon, \frac{\delta}{m}\right)\text{-one-shot} \rightarrow (\epsilon, \delta)\text{-weak tracking}$$

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

$$\left(\epsilon, \frac{\delta}{m}\right)\text{-one-shot} \quad \rightarrow \quad (\epsilon, \delta)\text{-weak tracking}$$

  - Using $O\left(\epsilon^{-2}\delta^{-1}m\right)$ rows (or $O\left(\epsilon^{-2}\delta^{-1}\log m\right)$ rows after median trick).

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

$\left(\epsilon, \frac{\delta}{m}\right)$-one-shot → (ϵ, δ)-weak tracking

  - Using $O\left(\epsilon^{-2}\delta^{-1}m\right)$ rows (or $O\left(\epsilon^{-2}\delta^{-1}\log m\right)$ rows after median trick).

- **Idea**: Using chaining argument [Braverman-Chestnut-Ivkin-Nelson-Wang-Woodruff 17] to get a fancier (and tighter) union bound.

Intuition for Weak Tracking

- **First attempt**: Apply union bound on one-shot analysis.

$\left(\epsilon, \frac{\delta}{m}\right)$-one-shot → (ϵ, δ)-weak tracking

  - Using $O\left(\epsilon^{-2}\delta^{-1}m\right)$ rows (or $O\left(\epsilon^{-2}\delta^{-1}\log m\right)$ rows after median trick).

- **Idea**: Using chaining argument [Braverman-Chestnut-Ivkin-Nelson-Wang-Woodruff 17] to get a fancier (and tighter) union bound.

We can get rid of the m dependency!

Step 1: Extracting the Correlation

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking:** Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left|\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon\|f^{(m)}\|_2^2\right|\right] \leq \delta$$

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left|\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon\|f^{(m)}\|_2^2\right|\right] \leq \delta$$

- Rewrite the error as:

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

- Rewrite the error as:

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \dots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

- Rewrite the error as:

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and
- $\tilde{B}_{\eta, f^{(t)}}$ depends on Π and $f^{(t)}$.

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t \in [m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

- Rewrite the error as:

Highly correlated

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and
- $\tilde{B}_{\eta, f^{(t)}}$ depends on Π and $f^{(t)}$.

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

- Rewrite the error as:

Highly correlated

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and
- $\tilde{B}_{\eta, f^{(t)}}$ depends on Π and $f^{(t)}$.

- The bad event becomes:

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right| \right] \leq \delta$$

- Rewrite the error as:

Highly correlated

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and
- $\tilde{B}_{\eta, f^{(t)}}$ depends on Π and $f^{(t)}$.

- The bad event becomes:

$$\sup_{t\in[m]} \left| \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma \right| > \epsilon \|f^{(m)}\|_2^2$$

Step 1: Extracting the Correlation

(ϵ, δ)-**Weak tracking**: Output $\|\Pi f^{(1)}\|_2^2, \ldots, \|\Pi f^{(m)}\|_2^2$ s.t.

$$\Pr\left[\exists_{t\in[m]} \left| \|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 > \epsilon \|f^{(m)}\|_2^2 \right|\right] \leq \delta$$

- Rewrite the error as:

Highly correlated

$$\|\Pi f^{(t)}\|_2^2 - \|f^{(t)}\|_2^2 = \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma$$

where

- $\sigma \in \{-1, 1\}^n$ and
- $\tilde{B}_{\eta, f^{(t)}}$ depends on Π and $f^{(t)}$.

Different from [BCINWW17]

- The bad event becomes:

$$\sup_{t\in[m]} \left| \sigma^\top \tilde{B}_{\eta, f^{(t)}} \sigma \right| > \epsilon \|f^{(m)}\|_2^2$$

Step 2: ϵ-Net

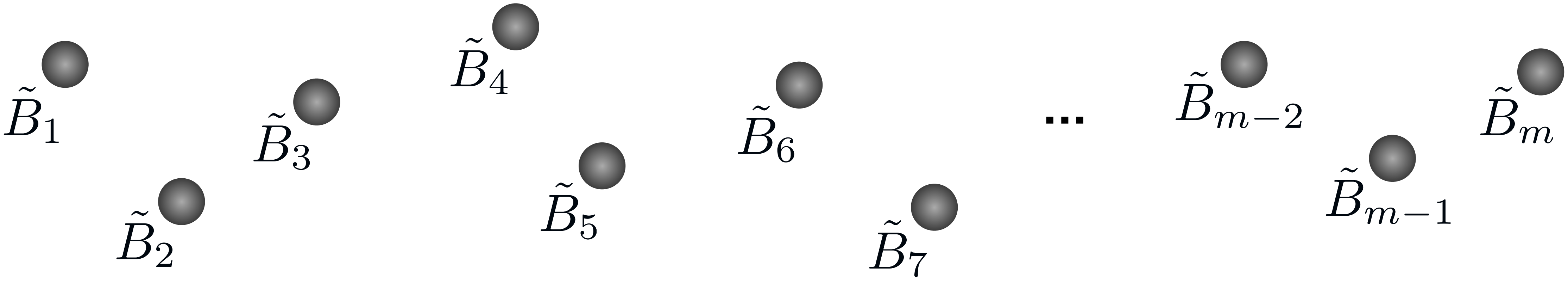
Step 2: ϵ-Net

Goal:
$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$

Step 2: ϵ-Net

Goal:

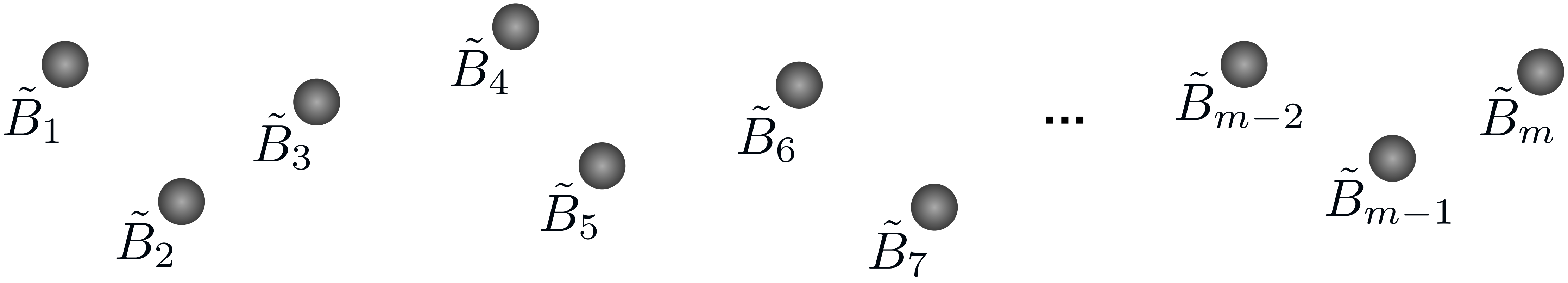
$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$



Step 2: ϵ-Net

Goal:

$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \le 0.1\,.$$

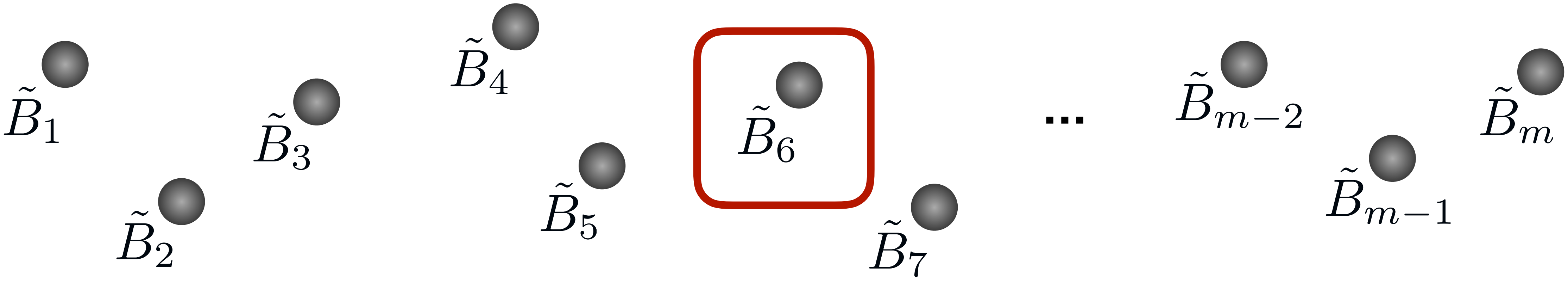


- A sequence of nets $T_0, T_1, \ldots$ such that

Step 2: ϵ-Net

Goal:

$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$

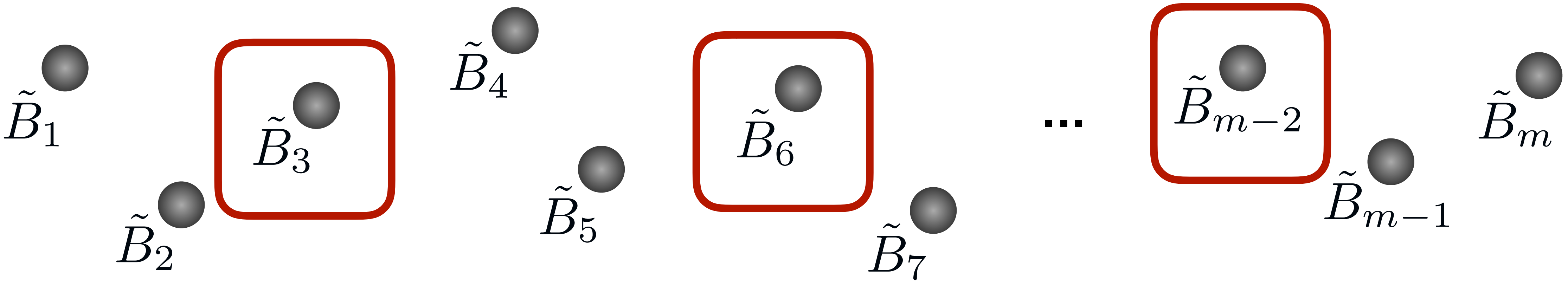


- A sequence of nets $T_0, T_1, \ldots$ such that

Step 2: ϵ-Net

Goal:

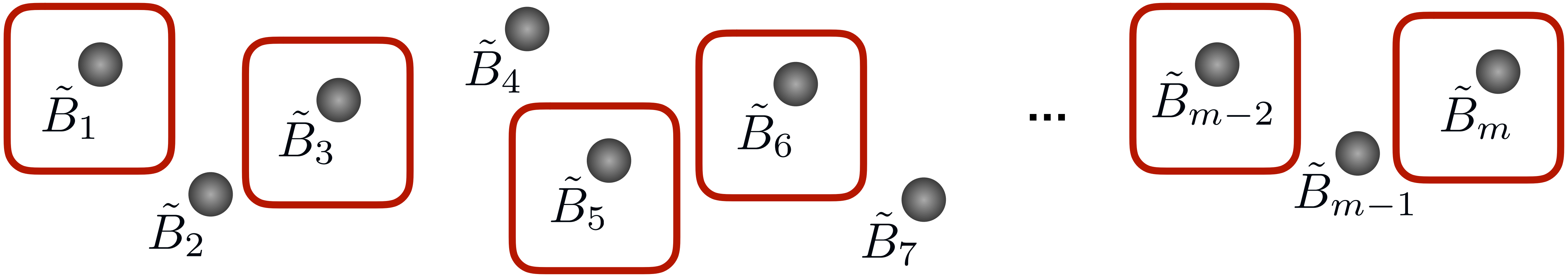
$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$



- A sequence of nets $T_0, T_1, \ldots$ such that

Step 2: ϵ-Net

Goal: $$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$

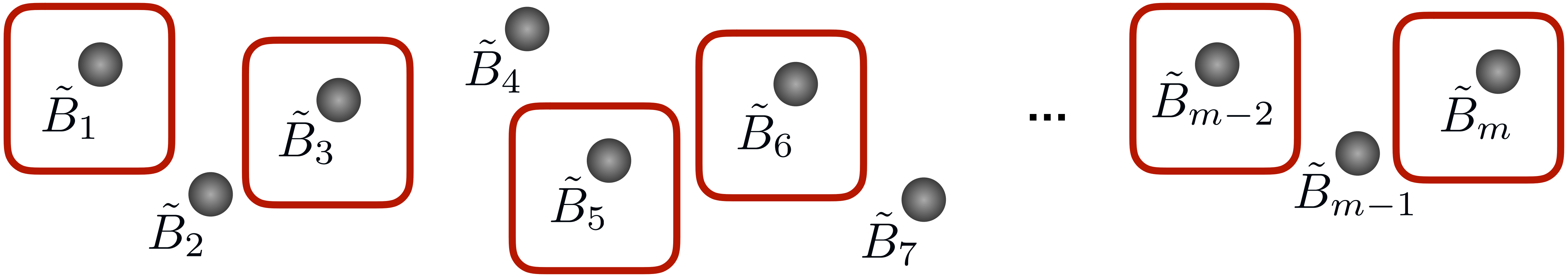


- A sequence of nets $T_0, T_1, \ldots$ such that

Step 2: ϵ-Net

Goal:

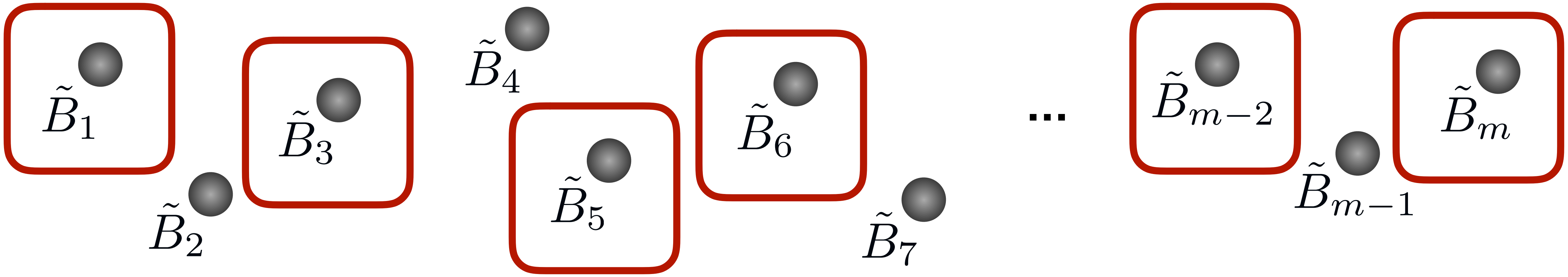
$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \le 0.1\,.$$



- A sequence of nets $T_0, T_1, \ldots$ such that
  - The coarser the net is, the smaller it is.

Step 2: ϵ-Net

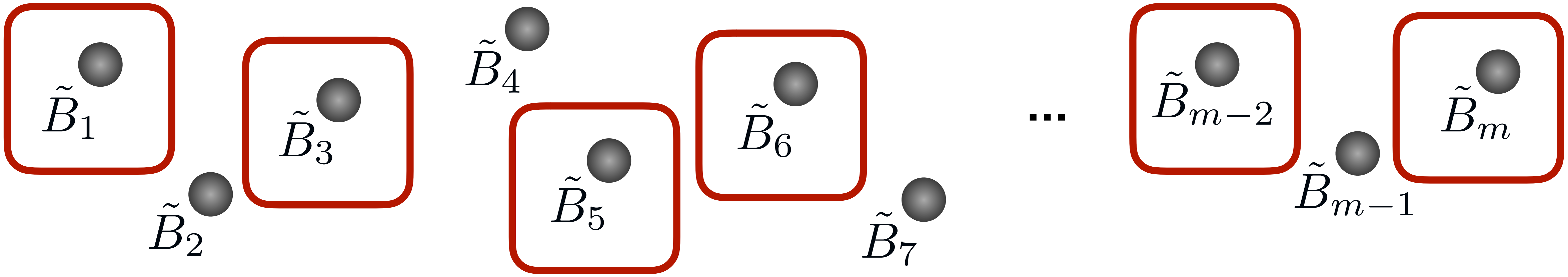
Goal: $$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \le 0.1\,.$$



- A sequence of nets $T_0, T_1, \ldots$ such that
  - The coarser the net is, the smaller it is.
- Telescoping $\tilde{B}_{\eta,f^{(t)}}$ using these nets

Step 2: ϵ-Net

Goal: $$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$

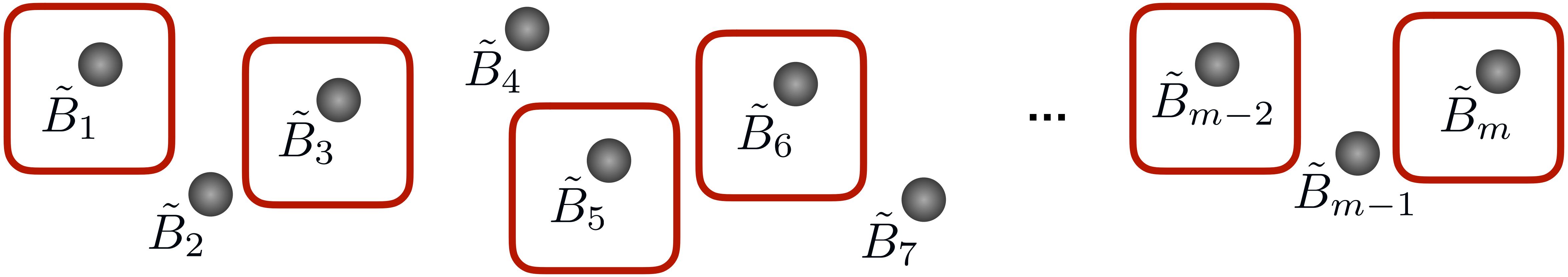


- A sequence of nets $T_0, T_1, \ldots$ such that
  - The coarser the net is, the smaller it is.
- Telescoping $\tilde{B}_{\eta,f^{(t)}}$ using these nets
  - $\sup_{t\in[m]}\gamma\left(\tilde{B}_{\eta,f^{(t)}}\right) \leq \sup_{t\in[m]}\gamma\left(B_{\eta,0}^{(t)}\right) + \sum_{\ell=1}^{\infty}\sup_{t\in[m]}\gamma\left(B_{\eta,\ell}^{(t)} - B_{\eta,\ell-1}^{(t)}\right)$

Step 2: ϵ-Net

Goal:

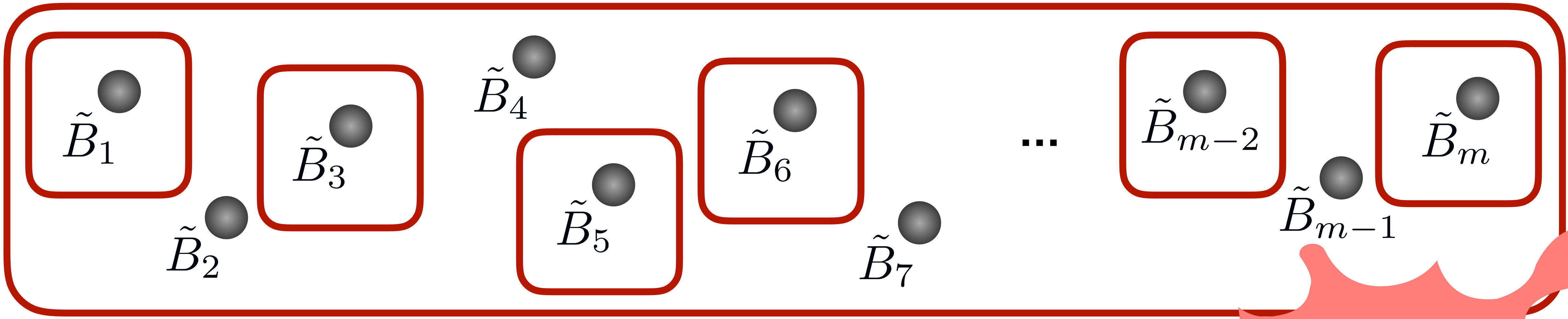
$$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \leq 0.1\,.$$



- A sequence of nets $T_0, T_1, \ldots$ such that
  - The coarser the net is, the smaller it is.
- Telescoping $\tilde{B}_{\eta,f^{(t)}}$ using these nets
  - $$\sup_{t\in[m]}\gamma\left(\tilde{B}_{\eta,f^{(t)}}\right) \leq \sup_{\boxed{T_0}}\gamma\left(B^{(t)}_{\eta,0}\right) + \sum_{\ell=1}^{\infty}\sup_{\boxed{T_\ell}}\gamma\left(B^{(t)}_{\eta,\ell} - B^{(t)}_{\eta,\ell-1}\right)$$

Step 2: ϵ-Net

Goal: $$\Pr\left[\sup_{t\in[m]}\left|\sigma^\top \tilde{B}_{\eta,f^{(t)}}\sigma\right| > \epsilon\|f^{(m)}\|_2^2\right] \le 0.1\,.$$



Different from [BCINWW17]

- A sequence of nets $T_0, T_1, \ldots$ such that
  - The coarser the net is, the smaller it is.
- Telescoping $\tilde{B}_{\eta,f^{(t)}}$ using these nets
  - $$\sup_{t\in[m]}\gamma\left(\tilde{B}_{\eta,f^{(t)}}\right) \le \sup_{\boxed{T_0}}\gamma\left(B^{(t)}_{\eta,0}\right) + \sum_{\ell=1}^{\infty}\sup_{\boxed{T_\ell}}\gamma\left(B^{(t)}_{\eta,\ell} - B^{(t)}_{\eta,\ell-1}\right)$$

What’s Missing…

What's Missing…

- Dudley's inequality.

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta, f^{(t)}}\}_{t \in [m]}$?

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta, f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta, f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.
- How to bound the error magnitude?

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.
- How to bound the error magnitude?
  - Using the Hansen-Wright inequality for the moments of $\sigma^\top B\sigma - \mathbb{E}[\sigma^\top B\sigma]$.

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.
- How to bound the error magnitude?
  - Using the Hansen-Wright inequality for the moments of $\sigma^\top B\sigma - \mathbb{E}[\sigma^\top B\sigma]$.
- High probability regime?

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.
- How to bound the error magnitude?
  - Using the Hansen-Wright inequality for the moments of $\sigma^\top B\sigma - \mathbb{E}[\sigma^\top B\sigma]$.
- High probability regime?
  - Median trick.

What's Missing…

- Dudley's inequality.
- How to bound the size of ϵ-net for $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$?
  - Greedily pick from $\{\tilde{B}_{\eta,f^{(t)}}\}_{t\in[m]}$ and analyze by the insertion-only structure of the input stream.
- How to bound the error magnitude?
  - Using the Hansen-Wright inequality for the moments of $\sigma^\top B\sigma - \mathbb{E}[\sigma^\top B\sigma]$.
- High probability regime?
  - Median trick.

Ask me offline for more details!

Conclusion

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
  - CountSketch and packet passing problem now have tracking guarantee!

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
  - CountSketch and packet passing problem now have tracking guarantee!
- Future directions:

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
  - CountSketch and packet passing problem now have tracking guarantee!
- Future directions:
  - Empirical performance of CountSketch?

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
    - CountSketch and packet passing problem now have tracking guarantee!
- Future directions:
    - Empirical performance of CountSketch?
    - Weak tracking for ℓ_p norm with faster update time?

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
  - CountSketch and packet passing problem now have tracking guarantee!
- Future directions:
  - Empirical performance of CountSketch?
  - Weak tracking for ℓ_p norm with faster update time?
  - Other applications of charing technique?

Conclusion

- We show the first streaming algorithm achieving weak tracking for ℓ_2 estimation with constant update time.
  - CountSketch and packet passing problem now have tracking guarantee!
- Future directions:
  - Empirical performance of CountSketch?
  - Weak tracking for ℓ_p norm with faster update time?
  - Other applications of charing technique?

Thanks for your attention, questions?