

# Learning Goals

- Frequency Estimation
- The Count-Min Sketch
- Cash register and turnstile models

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- The AMS sketch estimates  $\|x\|_2$ .

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- The AMS sketch estimates  $\|x\|_2$ .
- What if we would like an estimate of each  $x_j$ ? This is called Frequency Estimation.

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- The AMS sketch estimates  $\|x\|_2$ .
- What if we would like an estimate of each  $x_j$ ? This is called Frequency Estimation.
- Recall Bloom filter: we wanted to know quickly whether an element is present, allowing mistakes.

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- The AMS sketch estimates  $\|x\|_2$ .
- What if we would like an estimate of each  $x_j$ ? This is called Frequency Estimation.
- Recall Bloom filter: we wanted to know quickly whether an element is present, allowing mistakes.
  - There we maintained many hash tables, and return YES only if there is a record in all tables

# Frequency Estimation

- Recall the streaming model: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- The AMS sketch estimates  $\|x\|_2$ .
- What if we would like an estimate of each  $x_j$ ? This is called Frequency Estimation.
- Recall Bloom filter: we wanted to know quickly whether an element is present, allowing mistakes.
  - There we maintained many hash tables, and return YES only if there is a record in all tables
  - Can we emulate the idea here?



# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .
- Clearly,  $C[h(j)]$  is an overestimate of  $x_j$  due to clashes.

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .
- Clearly,  $C[h(j)]$  is an overestimate of  $x_j$  due to clashes.
- How many clashes are there?

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .
- Clearly,  $C[h(j)]$  is an overestimate of  $x_j$  due to clashes.
- How many clashes are there?
  - If  $h$  is sampled from a universal hash family, in expectation  $C[h(j)] \leq x_j + \frac{n}{k}$ .

# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .
- Clearly,  $C[h(j)]$  is an overestimate of  $x_j$  due to clashes.
- How many clashes are there?
  - If  $h$  is sampled from a universal hash family, in expectation  $C[h(j)] \leq x_j + \frac{n}{k}$ .
  - If we set  $k = \frac{1}{\epsilon}$ , this would give an estimate with  $\epsilon n$  additive error in expectation.



# Attempt with one hash table

- Let's try a hash function  $h$  from  $[d]$  to  $[k]$  for some  $k$  that we decide later.
- Maintain counters  $C[1], \dots, C[k]$ , initialized to 0.
- When  $i_t$  arrives, increase  $C[h(i_t)]$  by 1.
- In the end, to estimate  $x_j$ , we return  $C[h(j)]$ .
- Clearly,  $C[h(j)]$  is an overestimate of  $x_j$  due to clashes.
- How many clashes are there?
  - If  $h$  is sampled from a universal hash family, in expectation  $C[h(j)] \leq x_j + \frac{n}{k}$ .
  - If we set  $k = \frac{1}{\epsilon}$ , this would give an estimate with  $\epsilon n$  additive error in expectation.
  - Let's try pushing the guarantee to "with high probability" by repetitions!

# Count-Min Sketch

The COUNT-MIN algorithm by Cormode and Muthukrishnan (2005):

- Sample  $\ell$  hash functions  $h_1, \dots, h_\ell : [d] \rightarrow [k]$  independently from a universal hash family; let  $k$  be  $\frac{2}{\epsilon}$ .

# Count-Min Sketch

The COUNT-MIN algorithm by Cormode and Muthukrishnan (2005):

- Sample  $\ell$  hash functions  $h_1, \dots, h_\ell : [d] \rightarrow [k]$  independently from a universal hash family; let  $k$  be  $\frac{2}{\epsilon}$ .
- Maintain counters  $C_1[1], \dots, C_1[k], C_2[1], \dots, C_2[k], \dots, C_\ell[1], \dots, C_\ell[k]$ , initialized to 0.

# Count-Min Sketch

The COUNT-MIN algorithm by Cormode and Muthukrishnan (2005):

- Sample  $\ell$  hash functions  $h_1, \dots, h_\ell : [d] \rightarrow [k]$  independently from a universal hash family; let  $k$  be  $\frac{2}{\epsilon}$ .
- Maintain counters  $C_1[1], \dots, C_1[k], C_2[1], \dots, C_2[k], \dots, C_\ell[1], \dots, C_\ell[k]$ , initialized to 0.
- When  $i_t$  arrives, for  $j = 1, \dots, \ell$ , increase  $C_j[h_j(i_t)]$  by one.

# Count-Min Sketch

The COUNT-MIN algorithm by Cormode and Muthukrishnan (2005):

- Sample  $\ell$  hash functions  $h_1, \dots, h_\ell : [d] \rightarrow [k]$  independently from a universal hash family; let  $k$  be  $\frac{2}{\epsilon}$ .
- Maintain counters  $C_1[1], \dots, C_1[k], C_2[1], \dots, C_2[k], \dots, C_\ell[1], \dots, C_\ell[k]$ , initialized to 0.
- When  $i_t$  arrives, for  $j = 1, \dots, \ell$ , increase  $C_j[h_j(i_t)]$  by one.
- At the end, to estimate  $x_j$ , return  $\min \{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .



# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .
  - By Markov inequality,  $\mathbf{Pr}[Y_{ij} \geq \frac{2n}{k} = \epsilon n] \leq \frac{1}{2}$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .
  - By Markov inequality,  $\mathbf{Pr}[Y_{ij} \geq \frac{2n}{k} = \epsilon n] \leq \frac{1}{2}$ .
- By independence,  $\mathbf{Pr}[Y_{ij} \geq \epsilon n, \forall i] \leq 2^{-\ell}$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .
  - By Markov inequality,  $\mathbf{Pr}[Y_{ij} \geq \frac{2n}{k} = \epsilon n] \leq \frac{1}{2}$ .
- By independence,  $\mathbf{Pr}[Y_{ij} \geq \epsilon n, \forall i] \leq 2^{-\ell}$ .
- Therefore, for some  $\ell = O(\log d)$ , with high probability our estimate is correct within additive  $\epsilon n$  error for all coordinates of  $x$ .

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .
  - By Markov inequality,  $\Pr[Y_{ij} \geq \frac{2n}{k} = \epsilon n] \leq \frac{1}{2}$ .
- By independence,  $\Pr[Y_{ij} \geq \epsilon n, \forall i] \leq 2^{-\ell}$ .
- Therefore, for some  $\ell = O(\log d)$ , with high probability our estimate is correct within additive  $\epsilon n$  error for all coordinates of  $x$ .
- Space usage:
  - Maintaining the counters: there are  $k\ell = \frac{2}{\epsilon} \log d$  counters, each taking  $O(\log n)$  space.

# Analysis of COUNT-MIN

- For each  $j \in [d]$  and  $i \in [\ell]$ ,  $x_j \leq C_i[h_i(j)]$ , so the output is never smaller than  $x_j$ .
- For each  $j \in [d]$  and  $i \in [\ell]$ , let  $Y_{i,j}$  be the number of clashes of  $j$  by  $h_i$ , then  $C_i[h_i(j)] \leq x_j + Y_{i,j}$ .
  - By universality,  $\mathbf{E}[Y_{ij}] \leq \frac{n}{k}$ .
  - By Markov inequality,  $\Pr[Y_{ij} \geq \frac{2n}{k} = \epsilon n] \leq \frac{1}{2}$ .
- By independence,  $\Pr[Y_{ij} \geq \epsilon n, \forall i] \leq 2^{-\ell}$ .
- Therefore, for some  $\ell = O(\log d)$ , with high probability our estimate is correct within additive  $\epsilon n$  error for all coordinates of  $x$ .
- Space usage:
  - Maintaining the counters: there are  $k\ell = \frac{2}{\epsilon} \log d$  counters, each taking  $O(\log n)$  space.
  - Maintaining the hash functions: there are  $\ell = O(\log d)$  of them, each taking  $O(\log d)$  space.

# More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .

## More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .

## More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- Slight generalization: each data point at time  $t$  is a pair  $(i_t, \Delta_t)$ :
  - $i_t$  is the index
  - $\Delta_t$  is the *increment* of the count at index  $i_t$
- Our streaming problems so far are special cases when all  $\Delta_t = 1$ .



## More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- Slight generalization: each data point at time  $t$  is a pair  $(i_t, \Delta_t)$ :
  - $i_t$  is the index
  - $\Delta_t$  is the *increment* of the count at index  $i_t$
- Our streaming problems so far are special cases when all  $\Delta_t = 1$ .
- If  $\Delta_t$  are positive real numbers, this is called the *cash register* model.

# More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- Slight generalization: each data point at time  $t$  is a pair  $(i_t, \Delta_t)$ :
  - $i_t$  is the index
  - $\Delta_t$  is the *increment* of the count at index  $i_t$
- Our streaming problems so far are special cases when all  $\Delta_t = 1$ .
- If  $\Delta_t$  are positive real numbers, this is called the *cash register* model.
- If  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time, this is called the *strict turnstile* model.

## More General Streaming Models

- Our streaming model so far: the stream  $i_1, \dots, i_n \in [d] := \{1, \dots, d\}$ .
- The frequency vector  $x \in \mathbb{Z}^d$ :  $x_j = |\{t : i_t = j\}|$ .
- Slight generalization: each data point at time  $t$  is a pair  $(i_t, \Delta_t)$ :
  - $i_t$  is the index
  - $\Delta_t$  is the *increment* of the count at index  $i_t$
- Our streaming problems so far are special cases when all  $\Delta_t = 1$ .
- If  $\Delta_t$  are positive real numbers, this is called the *cash register* model.
- If  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time, this is called the *strict turnstile* model.
- If  $\Delta_t$  can be negative, and  $x_j$ 's can be negative as well, this is called the *turnstile* model.

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .
    - Recall  $\|x\|_1 = \sum_i |x_i|$ .

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .
    - Recall  $\|x\|_1 = \sum_i |x_i|$ .
- In the strict turnstile model,  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time.

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .
    - Recall  $\|x\|_1 = \sum_i |x_i|$ .
- In the strict turnstile model,  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time.
  - COUNT-MIN still works.



# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .
    - Recall  $\|x\|_1 = \sum_i |x_i|$ .
- In the strict turnstile model,  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time.
  - COUNT-MIN still works.
- In the turnstile model,  $\Delta_t$  can be negative, and  $x_j$ 's can be negative as well.

# Frequency Estimation in Turnstile Model

Does COUNT-MIN still work in these more general settings?

- In cash register model:  $\Delta_t$  are positive real numbers.
  - COUNT-MIN still works, just increase the counters by  $\Delta_t$ ; the error term is relaxed to  $\epsilon \|x\|_1$ .
    - Recall  $\|x\|_1 = \sum_i |x_i|$ .
- In the strict turnstile model,  $\Delta_t$  are allowed to be negative, but every frequency counter  $x_j$  is guaranteed to be non-negative at all time.
  - COUNT-MIN still works.
- In the turnstile model,  $\Delta_t$  can be negative, and  $x_j$ 's can be negative as well.
  - The analysis of COUNT-MIN is problematic in this setting. Markov inequality needs nonnegativity!

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.
- $C_i[h_i(j)] = x_j + \text{error}$ .

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.
- $C_i[h_i(j)] = x_j + \text{error}$ .
  - When “error” is all nonnegative, we can take minimum among  $C_i[h_i(j)]$ . But when error can be negative, taking the minimum may seriously underestimate  $x_j$ .

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.
- $C_i[h_i(j)] = x_j + \text{error}$ .
  - When “error” is all nonnegative, we can take minimum among  $C_i[h_i(j)]$ . But when error can be negative, taking the minimum may seriously underestimate  $x_j$ .
  - Similarly, taking the maximum may overestimate  $x_j$ .

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.
- $C_i[h_i(j)] = x_j + \text{error}$ .
  - When “error” is all nonnegative, we can take minimum among  $C_i[h_i(j)]$ . But when error can be negative, taking the minimum may seriously underestimate  $x_j$ .
  - Similarly, taking the maximum may overestimate  $x_j$ .
  - It is natural to try the *median*.

# Frequency Estimation for the Turnstile Model

- What goes wrong with the analysis of COUNT-MIN is that the error term caused by clashes can be negative.
- $C_i[h_i(j)] = x_j + \text{error}$ .
  - When “error” is all nonnegative, we can take minimum among  $C_i[h_i(j)]$ . But when error can be negative, taking the minimum may seriously underestimate  $x_j$ .
  - Similarly, taking the maximum may overestimate  $x_j$ .
  - It is natural to try the *median*.

## Claim (Problem 1 in Problem Set 3)

Let  $Z_1, \dots, Z_n$  be i.i.d. random variables. Let  $M$  be a median. There is a constant  $c > 0$  such that:

- If  $\Pr[Z_i \geq t] \leq p < \frac{1}{4}$ , then  $\Pr[M \geq t] \leq e^{-cn}$ .
- If  $\Pr[Z_i \leq t] \leq p < \frac{1}{4}$ , then  $\Pr[M \leq t] \leq e^{-cn}$ .



- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .
  - Similarly, indices that may cause negative error are  $N := \{j' : x_{j'} < 0\}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .
  - Similarly, indices that may cause negative error are  $N := \{j' : x_{j'} < 0\}$ .
- For any counter  $C_i$ , the expected error caused by indices in  $P$  is  $\leq \frac{\|x\|_1}{k}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .
  - Similarly, indices that may cause negative error are  $N := \{j' : x_{j'} < 0\}$ .
- For any counter  $C_i$ , the expected error caused by indices in  $P$  is  $\leq \frac{\|x\|_1}{k}$ .
- By Markov inequality,  $\Pr[\sum_{j' \in P \setminus \{j\}} x_{j'} \mathbb{1}_{h_i(j)=h_i(j')} \geq \frac{4\|x\|_1}{k}] \leq \frac{1}{4}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .
  - Similarly, indices that may cause negative error are  $N := \{j' : x_{j'} < 0\}$ .
- For any counter  $C_i$ , the expected error caused by indices in  $P$  is  $\leq \frac{\|x\|_1}{k}$ .
- By Markov inequality,  $\Pr[\sum_{j' \in P \setminus \{j\}} x_{j'} \mathbb{1}_{h_i(j)=h_i(j')} \geq \frac{4\|x\|_1}{k}] \leq \frac{1}{4}$ .
- Similarly,  $\Pr[\sum_{j' \in N \setminus \{j\}} |x_{j'}| \mathbb{1}_{h_i(j)=h_i(j')} \geq \frac{4\|x\|_1}{k}] \leq \frac{1}{4}$ .

- Algorithm: same setup and initialization as before
  - At input  $(i_t, \Delta_t)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += \Delta_t$
  - In the end, as an estimate of  $x_j$ , output a median of  $\{C_1[h_1(j)], \dots, C_\ell[h_\ell(j)]\}$ .
- Consider any fixed index  $j \in [d]$ .
  - Which indices may cause positive error?  $P := \{j' : x_{j'} > 0\}$ .
  - Similarly, indices that may cause negative error are  $N := \{j' : x_{j'} < 0\}$ .
- For any counter  $C_i$ , the expected error caused by indices in  $P$  is  $\leq \frac{\|x\|_1}{k}$ .
- By Markov inequality,  $\Pr[\sum_{j' \in P \setminus \{j\}} x_{j'} \mathbb{1}_{h_i(j)=h_i(j')} \geq \frac{4\|x\|_1}{k}] \leq \frac{1}{4}$ .
- Similarly,  $\Pr[\sum_{j' \in N \setminus \{j\}} |x_{j'}| \mathbb{1}_{h_i(j)=h_i(j')} \geq \frac{4\|x\|_1}{k}] \leq \frac{1}{4}$ .
- Setting  $k = \frac{4}{\epsilon}$ ,  $\ell = O(\log d)$ , with high probability our output for every coordinate is correct within  $\epsilon \|x\|_1$  additive error.



# COUNT-SKETCH

- We can do a bit better to control the error term to within  $\epsilon \|x\|_2$ 
  - Recall that  $\|x\|_p = (\sum_i x_i^p)^{1/p}$  decreases with  $p$  for  $p \in (0, \infty)$ .

## COUNT-SKETCH

- We can do a bit better to control the error term to within  $\epsilon \|x\|_2$ 
  - Recall that  $\|x\|_p = (\sum_i x_i^p)^{1/p}$  decreases with  $p$  for  $p \in (0, \infty)$ .
  - Common bound:  $\|x\|_2 \geq \frac{1}{\sqrt{n}} \|x\|_1$  by Cauchy-Schwartz

## COUNT-SKETCH

- We can do a bit better to control the error term to within  $\epsilon \|x\|_2$ 
  - Recall that  $\|x\|_p = (\sum_i x_i^p)^{1/p}$  decreases with  $p$  for  $p \in (0, \infty)$ .
  - Common bound:  $\|x\|_2 \geq \frac{1}{\sqrt{n}} \|x\|_1$  by Cauchy-Schwartz
- COUNT-SKETCH due to Charikar, Chen, Farach-Colton (2004)
  - Same setup and initialization as before; in addition, maintain hash functions  $g_1, \dots, g_\ell : [d] \rightarrow \{+1, -1\}$ , each drawn independently from a 2-wise universal hash family

## COUNT-SKETCH

- We can do a bit better to control the error term to within  $\epsilon \|x\|_2$ 
  - Recall that  $\|x\|_p = (\sum_i x_i^p)^{1/p}$  decreases with  $p$  for  $p \in (0, \infty)$ .
  - Common bound:  $\|x\|_2 \geq \frac{1}{\sqrt{n}} \|x\|_1$  by Cauchy-Schwartz
- COUNT-SKETCH due to Charikar, Chen, Farach-Colton (2004)
  - Same setup and initialization as before; in addition, maintain hash functions  $g_1, \dots, g_\ell : [d] \rightarrow \{+1, -1\}$ , each drawn independently from a 2-wise universal hash family
  - At input  $(i_t, \Delta)$ , for  $i = 1, \dots, \ell$ ,  $C_i[h_i(i_t)] += g_i \Delta_t$ .
  - In the end, for index  $j \in [d]$ , output a median  $M$  among  $g_1(j)C_1[h_1(j)], \dots, g_\ell(j)C_\ell[h_\ell(j)]$ .

# Analysis of COUNT-SKETCH

- For any  $i \in [\ell]$  and  $j \in [d]$ . By pairwise independence of  $g_i(\cdot)$ 's,

$$\mathbf{E} [C_i[h_i(j)]g_i(j)] = x_j + \mathbf{E} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] = x_j.$$

# Analysis of COUNT-SKETCH

- For any  $i \in [\ell]$  and  $j \in [d]$ . By pairwise independence of  $g_i(\cdot)$ 's,

$$\mathbf{E} [C_i[h_i(j)]g_i(j)] = x_j + \mathbf{E} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] = x_j.$$

- We bound the deviation by Chebyshev inequality:

# Analysis of COUNT-SKETCH

- For any  $i \in [\ell]$  and  $j \in [d]$ . By pairwise independence of  $g_i(\cdot)$ 's,

$$\mathbf{E} [C_i[h_i(j)]g_i(j)] = x_j + \mathbf{E} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] = x_j.$$

- We bound the deviation by Chebyshev inequality:

$$\text{Var} [C_i[h_i(j)]g_i(j)] = \text{Var} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] \leq \frac{\|x\|_2^2}{k}.$$

# Analysis of COUNT-SKETCH

- For any  $i \in [\ell]$  and  $j \in [d]$ . By pairwise independence of  $g_i(\cdot)$ 's,

$$\mathbf{E} [C_i[h_i(j)]g_i(j)] = x_j + \mathbf{E} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] = x_j.$$

- We bound the deviation by Chebyshev inequality:

$$\text{Var} [C_i[h_i(j)]g_i(j)] = \text{Var} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] \leq \frac{\|\mathbf{x}\|_2^2}{k}.$$

$$\Pr [|g_i(j)C_i[h_i(j)] - x_j| \geq \epsilon \|\mathbf{x}\|_2] \leq \frac{\|\mathbf{x}\|_2^2}{k\epsilon^2 \|\mathbf{x}\|_2^2} = \frac{1}{k\epsilon^2}$$



# Analysis of COUNT-SKETCH

- For any  $i \in [\ell]$  and  $j \in [d]$ . By pairwise independence of  $g_i(\cdot)$ 's,

$$\mathbf{E} [C_i[h_i(j)]g_i(j)] = x_j + \mathbf{E} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] = x_j.$$

- We bound the deviation by Chebyshev inequality:

$$\text{Var} [C_i[h_i(j)]g_i(j)] = \text{Var} \left[ \sum_{j' \neq j} g_i(j)g_i(j')x_{j'} \mathbb{1}_{h_i(j')=h_i(j)} \right] \leq \frac{\|\mathbf{x}\|_2^2}{k}.$$

$$\Pr [|g_i(j)C_i[h_i(j)] - x_j| \geq \epsilon \|\mathbf{x}\|_2] \leq \frac{\|\mathbf{x}\|_2^2}{k\epsilon^2 \|\mathbf{x}\|_2^2} = \frac{1}{k\epsilon^2}$$

We can take  $k = O(\frac{1}{\epsilon^2})$ .