Data streaming algorithms (2)

He Sun

THE UNIVERSITY of EDINBURGH

Recall: streaming algorithms

- The input of a streaming algorithm is given as a data stream, which is a sequence of data

$$\mathcal{S} = s_1, s_2, \cdots, s_m, \cdots,$$

and every s_i belongs to the universe U of size n.

Recall: streaming algorithms

- The input of a streaming algorithm is given as a data stream, which is a sequence of data
$$\mathcal{S} = s_1, s_2, \cdots, s_m, \cdots,$$
and every s_i belongs to the universe U of size n.
- Constraints for streaming algorithms: the space complexity is sublinear in n, and is independent in the length of $\mathcal{S}$.

Recall: streaming algorithms

- The input of a streaming algorithm is given as a data stream, which is a sequence of data
$$\mathcal{S} = s_1, s_2, \cdots, s_m, \cdots,$$
and every s_i belongs to the universe U of size n.
- Constraints for streaming algorithms: the space complexity is sublinear in n, and is independent in the length of $\mathcal{S}$.
- Quality of the output: The algorithm needs to give a good approximate value with high probability.

Recall: streaming algorithms

- The input of a streaming algorithm is given as a data stream, which is a sequence of data
$$\mathcal{S} = s_1, s_2, \cdots, s_m, \cdots,$$
and every s_i belongs to the universe U of size n.

- Constraints for streaming algorithms: the space complexity is sublinear in n, and is independent in the length of $\mathcal{S}$.

- Quality of the output: The algorithm needs to give a good approximate value with high probability.

(ε, δ)-APPROXIMATION

For confidence parameter ε and approximation parameter δ, the algorithm's output Output and the exact answer Exact satisfies

$$\mathbb{P}\left[\text{Output} \in (1-\varepsilon, 1+\varepsilon) \cdot \text{Exact}\right] \geq 1 - \delta.$$

Recall: two models of streaming algorithms

Cash register model: every item in stream $\mathcal{S}$ is an item in U.

Recall: two models of streaming algorithms

Cash register model: every item in stream $\mathcal{S}$ is an item in U.

Turnstile model: every item s_i in $\mathcal{S}$ associates with " +" or "-", which indicates if s_i is added into or deleted from $\mathcal{S}$.

- "+" indicates that s_i is added into the dataset;
- "-" indicates that s_i is deleted from the dataset.

Why turnstile model?

- Data may be added or deleted over time, e.g. Facebook graph.
- We need *robust* algorithms to handle this situation.

Recall: F_p-Norm

F_p-NORM

Let U with $|U| = n$ be a dataset, and m_j be the number of occurrences of j in a stream. The F_p-norm is defined by

$$F_p \triangleq \sum_{i \in U} |m_i|^p .$$

Recall: F_p-Norm

F_p-NORM

Let U with $|U| = n$ be a dataset, and m_j be the number of occurrences of j in a stream. The F_p-norm is defined by

$$F_p \triangleq \sum_{i \in U} |m_i|^p \,.$$

- F_1 = total number of items in stream $\mathcal{S}$.
- F_0 = total number of distinct items in stream $\mathcal{S}$.

Recall: F_p-Norm

F_p-NORM

Let U with $|U| = n$ be a dataset, and m_j be the number of occurrences of j in a stream. The F_p-norm is defined by

$$F_p \triangleq \sum_{i \in U} |m_i|^p \,.$$

- F_1 = total number of items in stream $\mathcal{S}$.
- F_0 = total number of distinct items in stream $\mathcal{S}$.

THEOREM

The medium of the returned values from $\Theta(\log(1/\delta))$ independent copies of the BJKST algorithm gives an (ε, δ)-approximation of F_0.

Last lecture: algorithms in the cash register model

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;

Last lecture: algorithms in the cash register model

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;
2. Store the statistical information of the sampled items, or store the sampled items directly.

Last lecture: algorithms in the cash register model

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;
2. Store the statistical information of the sampled items, or store the sampled items directly.

Downside of this framework

- Sampling probability for the current item usually depends on the whole data stream that algorithm has seen so far.
  - For example, the index z in the BJKST algorithm

Last lecture: algorithms in the cash register model

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;
2. Store the statistical information of the sampled items, or store the sampled items directly.

Downside of this framework

- Sampling probability for the current item usually depends on the whole data stream that algorithm has seen so far.
  - For example, the index z in the BJKST algorithm
- Deleting an item appeared before could potentially makes the current sampling probability useless! :(

Last lecture: algorithms in the cash register model

Common approach for designing algorithms in the cash register model:

1. Sample the data items based on hashed values;
2. Store the statistical information of the sampled items, or store the sampled items directly.

DOWNSIDE OF THIS FRAMEWORK

- Sampling probability for the current item usually depends on the whole data stream that algorithm has seen so far.
  - For example, the index z in the BJKST algorithm
- Deleting an item appeared before could potentially makes the current sampling probability useless! :(

Sampling techniques are usually non-applicable in the turnstile model.

Outline

- Approximating F_2-norm in the turnstile model
- Frequency estimation in the turnstile model

Algorithm to approximate F_2 in the turnstile model

ALGORITHM TO APPROXIMATE F_2 (THE SIMPLIFIED DESCRIPTION)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$

Algorithm to approximate F_2 in the turnstile model

ALGORITHM TO APPROXIMATE F_2 (THE SIMPLIFIED DESCRIPTION)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$

Algorithm to approximate F_2 in the turnstile model

ALGORITHM TO APPROXIMATE F_2 (THE SIMPLIFIED DESCRIPTION)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$

Algorithm to approximate F_2 in the turnstile model

ALGORITHM TO APPROXIMATE F_2 (THE SIMPLIFIED DESCRIPTION)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$
3: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (The simplified description)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$
3: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
4: if x is inserted, then $y = y + h(x)$

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (The simplified description)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$
3: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
4: if x is inserted, then $y = y + h(x)$
5: else $y = y - h(x)$

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (The simplified description)

1: Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
2: Set $y = 0$
3: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
4: if x is inserted, then $y = y + h(x)$
5: else $y = y - h(x)$
6: **return** $Z := y^2$

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (The simplified description)

Choose a **4-wise independent** hash function $h : [n] \to \{-1, 1\}$
Set $y = 0$
while item $(x, \pm)$ from stream $\mathcal{S}$ arrives
    if x is inserted, then $y = y + h(x)$
        else $y = y - h(x)$
return $Z := y^2$

The algorithm runs in the turnstile model!

Algorithm to approximate F_2 in the turnstile model

ALGORITHM TO APPROXIMATE F_2 (THE SIMPLIFIED DESCRIPTION)

```
Choose a 4-wise independent hash function h : [n] → {−1, 1}
Set y = 0
while item (x, ±) from stream S arrives
        if x is inserted, then y = y + h(x)
                               else y = y − h(x)
return Z := y^2
```

The algorithm runs in the turnstile model!

KEY LEMMA

It holds that $\mathbb{E}[Z] = F_2$ and $\mathbb{V}[Z] \leq 2 \cdot \left(\sum_{i\in\mathcal{S}} m_i^2\right)^2 = 2F_2^2$.

Algorithm to approximate F_2 in the turnstile model

Algorithm to approximate F_2 (The simplified description)

Choose a **4-wise independent** hash function $h : [n] \rightarrow \{-1, 1\}$
Set $y = 0$
while item $(x, \pm)$ from stream $\mathcal{S}$ arrives
    if x is inserted, then $y = y + h(x)$
        else $y = y - h(x)$
return $Z := y^2$

The algorithm runs in the turnstile model!

Key Lemma

It holds that $\mathbb{E}[Z] = F_2$ and $\mathbb{V}[Z] \leq 2 \cdot \left(\sum_{i \in \mathcal{S}} m_i^2\right)^2 = 2F_2^2$.

Hence, we can (ε, δ)-approximate F_2, by **running multiple copies of the algorithm in parallel and return the average value.**

Algorithm to approximate F_2 in the turnstile model

Another algorithm to approximate F_0

1: $t = \lceil 6/\varepsilon^2 \rceil$

2: Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

Algorithm to approximate F_2 in the turnstile model

ANOTHER ALGORITHM TO APPROXIMATE F_0

1: $t = \lceil 6/\varepsilon^2 \rceil$
2: Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

3: Set $y_i = 0$ for $i = 1, \ldots, t$

Algorithm to approximate F_2 in the turnstile model

ANOTHER ALGORITHM TO APPROXIMATE F_0

1: $t = \lceil 6/\varepsilon^2 \rceil$

2: Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

3: Set $y_i = 0$ for $i = 1, \ldots, t$

4: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives

Algorithm to approximate F_2 in the turnstile model

ANOTHER ALGORITHM TO APPROXIMATE F_0

$t = \lceil 6/\varepsilon^2 \rceil$
Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \rightarrow \{-1, 1\}$$

Set $y_i = 0$ for $i = 1, \ldots, t$
while item $(x, \pm)$ from stream $\mathcal{S}$ arrives
    if x is inserted, then $y_i = y_i + h_i(x)$ for every $1 \leq i \leq t$

Algorithm to approximate F_2 in the turnstile model

ANOTHER ALGORITHM TO APPROXIMATE F_0

1: $t = \lceil 6/\varepsilon^2 \rceil$
2: Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

3: Set $y_i = 0$ for $i = 1, \ldots, t$
4: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
5: if x is inserted, then $y_i = y_i + h_i(x)$ for every $1 \leq i \leq t$
6: else $y_i = y_i - h_i(x)$ for every $1 \leq i \leq t$

Algorithm to approximate F_2 in the turnstile model

Another algorithm to approximate F_0

$t = \lceil 6/\varepsilon^2 \rceil$
Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

Set $y_i = 0$ for $i = 1, \ldots, t$
while item $(x, \pm)$ from stream $\mathcal{S}$ arrives
    if x is inserted, then $y_i = y_i + h_i(x)$ for every $1 \leq i \leq t$
        else $y_i = y_i - h_i(x)$ for every $1 \leq i \leq t$
return $\frac{1}{t} \cdot \sum_{i=1}^{t} Z_i$, where $Z_i = y_i^2$

Algorithm to approximate F_2 in the turnstile model

Another algorithm to approximate F_0

1: $t = \lceil 6/\varepsilon^2 \rceil$
2: Choose t 4-wise independent hash function $h_1, \ldots, h_t$, where

$$h_i : [n] \to \{-1, 1\}$$

3: Set $y_i = 0$ for $i = 1, \ldots, t$
4: **while** item $(x, \pm)$ from stream $\mathcal{S}$ arrives
5: if x is inserted, then $y_i = y_i + h_i(x)$ for every $1 \leq i \leq t$
6: else $y_i = y_i - h_i(x)$ for every $1 \leq i \leq t$
7: **return** $\frac{1}{t} \cdot \sum_{i=1}^{t} Z_i$, where $Z_i = y_i^2$

Theorem

With constant probability, the returned value of the algorithm is in $(1-\varepsilon, 1+\varepsilon) \cdot F_2$. Moreover, the algorithm's space complexity is $O\left((1/\varepsilon^2)\log n\right)$ bits.

Algorithm analysis: space complexity (upper bounding t)

Our current status:

- We run t independent copies in parallel and return $\left(\sum_{i=1}^{t} Z_i\right)/t$.
- The key lemma tells us that $\mathbb{E}[Z_i] = F_2$, and $\mathbb{V}[Z_i] \leq 2 \cdot F_2^2$.

Algorithm analysis: space complexity (upper bounding t)

Our current status:

- We run t independent copies in parallel and return $\left(\sum_{i=1}^{t} Z_i\right)/t$.
- The key lemma tells us that $\mathbb{E}[Z_i] = F_2$, and $\mathbb{V}[Z_i] \leq 2 \cdot F_2^2$.

To derive a upper bound on t to ensure an (ε, δ)-approximation, we apply the Law of Large Numbers:

$$\begin{aligned}\mathbb{P}\left[\left|\frac{Z_1 + \cdots + Z_t}{t} - \mathbb{E}[Z_i]\right| \geq \varepsilon \mathbb{E}[Z_i]\right] &= \mathbb{P}\left[\left|\frac{Z_1 + \cdots + Z_t}{t} - F_2\right| \geq \varepsilon F_2\right] \\ &\leq \frac{2 \cdot F_2^2}{t \cdot (\varepsilon \mathbb{E}[Z_i])^2} \\ &= \frac{2 \cdot F_2^2}{t \cdot \varepsilon^2 \cdot F_2^2}.\end{aligned}$$

Hence, choosing $t = \left\lceil 6/\varepsilon^2 \right\rceil$ suffices for our purpose.

Proving the key lemma: $\mathbb{E}[Z] = F_2$, where $Z = Z_i$

By the algorithm description, we have $y = \sum_{x \in \mathcal{S}} m_x \cdot h(x)$, where m_x is the number of occurrences of x.

Proving the key lemma: $\mathbb{E}[Z] = F_2$, where $Z = Z_i$

By the algorithm description, we have $y = \sum_{x \in \mathcal{S}} m_x \cdot h(x)$, where m_x is the number of occurrences of x. Hence,

$$Z = \left(\sum_{x \in \mathcal{S}} m_x \cdot h(x) \right)^2 = \sum_{x \in \mathcal{S}} m_x^2 \cdot h^2(x) + \sum_{\substack{x, y \in \mathcal{S} \\ x \neq y}} m_x \cdot h(x) \cdot m_y \cdot h(y).$$

Proving the key lemma: $\mathbb{E}[Z] = F_2$, where $Z = Z_i$

By the algorithm description, we have $y = \sum_{x \in \mathcal{S}} m_x \cdot h(x)$, where m_x is the number of occurrences of x. Hence,

$$Z = \left(\sum_{x \in \mathcal{S}} m_x \cdot h(x) \right)^2 = \sum_{x \in \mathcal{S}} m_x^2 \cdot h^2(x) + \sum_{\substack{x,y \in \mathcal{S} \\ x \neq y}} m_x \cdot h(x) \cdot m_y \cdot h(y).$$

By linearity of expectation, we have

$$\mathbb{E}\left[Z\right] = \sum_{x \in \mathcal{S}} m_x^2 \cdot \mathbb{E}\left[h^2(x)\right] + \sum_{\substack{x,y \in \mathcal{S} \\ x \neq y}} m_x \cdot m_y \cdot \mathbb{E}\left[h(x)\right] \mathbb{E}\left[h(y)\right]$$

$$= \sum_{x \in \mathcal{S}} m_x^2 = F_2.$$

Here we use the fact that

$$\mathbb{E}\left[h(x)\right] = 0, \qquad \mathbb{E}\left[h^2(x)\right] = 1.$$

The key: different powers of h and $\mathbb{E}(\cdot)$ give magical cancellation!

Proving the key lemma: $\mathbb{V}[Z] \leq 2F_2^2$, where $Z = Z_i$

We have

$$\begin{aligned}\mathbb{E}\left[Z^2\right] &= \mathbb{E}\left[\left(\sum_{x\in\mathcal{S}} m_x \cdot h(x)\right)^4\right] \\ &= \sum_{x,y,u,v} m_x \cdot m_y \cdot m_u \cdot m_v \cdot \mathbb{E}\left[h(x)\cdot h(y)\cdot h(u)\cdot h(v)\right].\end{aligned}$$

Proving the key lemma: $\mathbb{V}[Z] \leq 2F_2^2$, where $Z = Z_i$

We have

$$\mathbb{E}\left[Z^2\right] = \mathbb{E}\left[\left(\sum_{x \in \mathcal{S}} m_x \cdot h(x)\right)^4\right]$$
$$= \sum_{x,y,u,v} m_x \cdot m_y \cdot m_u \cdot m_v \cdot \mathbb{E}\left[h(x) \cdot h(y) \cdot h(u) \cdot h(v)\right].$$

Since $\mathbb{E}\left[h(x)\right] = \mathbb{E}\left[h^3(x)\right] = 0$ and $\mathbb{E}\left[h^2(x)\right] = \mathbb{E}\left[h^4(x)\right] = 1$,

Proving the key lemma: $\mathbb{V}[Z] \leq 2F_2^2$, where $Z = Z_i$

We have

$$\mathbb{E}\left[Z^2\right] = \mathbb{E}\left[\left(\sum_{x\in\mathcal{S}} m_x \cdot h(x)\right)^4\right]$$
$$= \sum_{x,y,u,v} m_x \cdot m_y \cdot m_u \cdot m_v \cdot \mathbb{E}\left[h(x)\cdot h(y)\cdot h(u)\cdot h(v)\right].$$

Since $\mathbb{E}\left[h(x)\right] = \mathbb{E}\left[h^3(x)\right] = 0$ and $\mathbb{E}\left[h^2(x)\right] = \mathbb{E}\left[h^4(x)\right] = 1$, we have

$$\mathbb{E}\left[Z^2\right] = \sum_{x\in S} m_x^4 \cdot \mathbb{E}\left[h^4(x)\right] + \sum_{\substack{x,y\in\mathcal{S}\\ x\neq y}} \frac{1}{2}\cdot\binom{4}{2}\cdot m_x^2\cdot m_y^2 \cdot \mathbb{E}\left[h^2(x)\right]\mathbb{E}\left[h^2(y)\right]$$

only variables with even degrees survive!

$$= \sum_{x\in S} m_x^4 + \sum_{\substack{x,y\in\mathcal{S}\\ x\neq y}} \frac{1}{2}\cdot\binom{4}{2}\cdot m_x^2\cdot m_y^2$$
$$\leq 2\cdot\left(\sum_{x\in S} m_x^2\right)^2 = 2\cdot F_2^2.$$

A COMMON APPROACH

1. Construct an estimator Z (random variable). Prove that

$$Z = \text{the target value in expectation}$$

A COMMON APPROACH

1. Construct an estimator Z (random variable). Prove that

$$Z = \text{the target value in expectation}$$

   - This is probably the most elegant part. Think of the right sampling probability, and/or get the right cancellation by using hash functions and $\mathbb{E}[\cdot]$.

Streaming algorithms for approximating F_p

A COMMON APPROACH

1. Construct an estimator Z (random variable). Prove that

$$Z = \text{the target value in expectation}$$

   - This is probably the most elegant part. Think of the right sampling probability, and/or get the right cancellation by using hash functions and $\mathbb{E}[\cdot]$.

2. Upper bound $\mathbb{V}[Z]$

Streaming algorithms for approximating F_p

A COMMON APPROACH

1. Construct an estimator Z (random variable). Prove that

$$Z = \text{the target value in expectation}$$

   - This is probably the most elegant part. Think of the right sampling probability, and/or get the right cancellation by using hash functions and $\mathbb{E}[\cdot]$.

2. Upper bound $\mathbb{V}[Z]$

3. Apply Chebyshev's inequality and Chernoff bound to show the number of copies needed to run in parallel in order to have (ε, δ)-approximation.

Streaming algorithms for approximating F_p

A COMMON APPROACH

1. Construct an estimator Z (random variable). Prove that

$$Z = \text{the target value in expectation}$$

   - This is probably the most elegant part. Think of the right sampling probability, and/or get the right cancellation by using hash functions and $\mathbb{E}[\cdot]$.

2. Upper bound $\mathbb{V}[Z]$

3. Apply Chebyshev's inequality and Chernoff bound to show the number of copies needed to run in parallel in order to have (ε, δ)-approximation.
   - Sadly, applications of these inequalities always introduce a factor of $O(1/\varepsilon^2)$.
   - Is the $1/\varepsilon^2$-dependency always needed?

Outline

- Approximating F_2-norm in the turnstile model
- Frequency estimation in the turnstile model

Frequency estimation

Let S be a multiset, and S is empty initially. The data stream consists of a sequence of update operations, and each operation is one of the follows:

- Insert(x): add x into the set S;

Frequency estimation

Let S be a multiset, and S is empty initially. The data stream consists of a sequence of update operations, and each operation is one of the follows:

- Insert(x): add x into the set S;
- Delete(x): delete x from the set S;

Frequency estimation

Let S be a multiset, and S is empty initially. The data stream consists of a sequence of update operations, and each operation is one of the follows:

- Insert(x): add x into the set S;
- Delete(x): delete x from the set S;
- Query(x): return the number of occurrences of x.

Frequency estimation

Let S be a multiset, and S is empty initially. The data stream consists of a sequence of update operations, and each operation is one of the follows:

- Insert(x): add x into the set S;
- Delete(x): delete x from the set S;
- Query(x): return the number of occurrences of x.

FREQUENCY ESTIMATION

Design a streaming algorithm that supports the three operations above.

Frequency estimation

Let S be a multiset, and S is empty initially. The data stream consists of a sequence of update operations, and each operation is one of the follows:

- Insert(x): add x into the set S;
- Delete(x): delete x from the set S;
- Query(x): return the number of occurrences of x.

FREQUENCY ESTIMATION

Design a streaming algorithm that supports the three operations above.

We need to design an algorithm running in the turnstile model!

The Count-Min sketch

Cormode and Muthukrishnan (2005) introduced the Count-Min sketch for the frequency estimation problem.

The Count-Min sketch

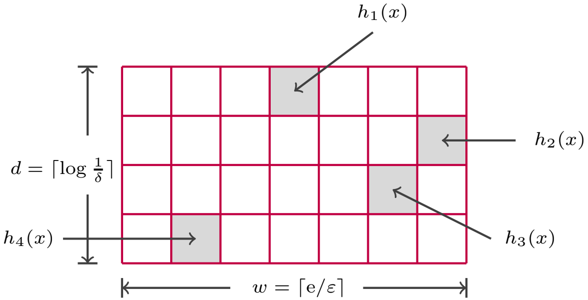
Cormode and Muthukrishnan (2005) introduced the Count-Min sketch for the frequency estimation problem.

Count-Min sketch: a table C of d rows and w columns, and every row j is associated with a universal hash function $h_j : [N] \to [w]$.

The Count-Min sketch

Cormode and Muthukrishnan (2005) introduced the Count-Min sketch for the frequency estimation problem.

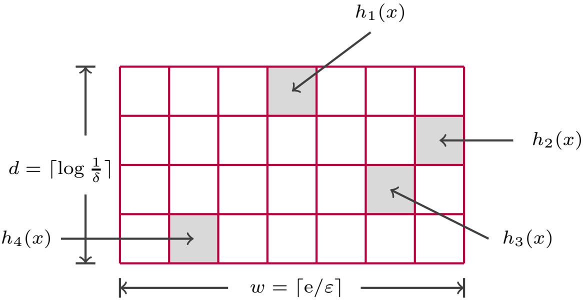
Count-Min sketch: a table C of d rows and w columns, and every row j is associated with a universal hash function $h_j : [N] \to [w]$.



The Count-Min sketch

Cormode and Muthukrishnan (2005) introduced the Count-Min sketch for the frequency estimation problem.

Count-Min sketch: a table C of d rows and w columns, and every row j is associated with a universal hash function $h_j : [N] \to [w]$.



The space complexity only depends on ε and δ.

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

1: If Insert(x) arrives

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

1: If Insert(x) arrives
2: for $j = 1$ to d do

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

```
If Insert(x) arrives
    for j = 1 to d do
        C[j, h_j(x)] = C[j, h_j(x)] + 1
```

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

```
If Insert(x) arrives
    for j = 1 to d do
        C[j, h_j(x)] = C[j, h_j(x)] + 1
If Delete(x) arrives
```

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

If Insert(x) arrives
    for $j = 1$ to d do
        $C[j, h_j(x)] = C[j, h_j(x)] + 1$
If Delete(x) arrives
    for $j = 1$ to d do

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

```
If Insert(x) arrives
    for j = 1 to d do
        C[j, h_j(x)] = C[j, h_j(x)] + 1
If Delete(x) arrives
    for j = 1 to d do
        C[j, h_j(x)] = C[j, h_j(x)] - 1
```

Update/Query operations for Count-Min

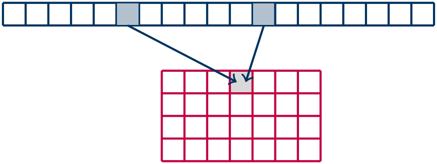
Update/Query operations for Count-Min

If Insert(x) arrives
  for $j = 1$ to d do
    $C[j, h_j(x)] = C[j, h_j(x)] + 1$
If Delete(x) arrives
  for $j = 1$ to d do
    $C[j, h_j(x)] = C[j, h_j(x)] - 1$
If Query(x) arrives, then

Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

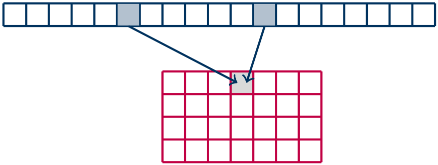
If Insert(x) arrives
    for $j = 1$ to d do
        $C[j, h_j(x)] = C[j, h_j(x)] + 1$
If Delete(x) arrives
    for $j = 1$ to d do
        $C[j, h_j(x)] = C[j, h_j(x)] - 1$
If Query(x) arrives, then
    return $m'_x \triangleq \min_{1 \leq i \leq d} C[j, h_j(x)]$



Update/Query operations for Count-Min

UPDATE/QUERY OPERATIONS FOR COUNT-MIN

If Insert(x) arrives
    for $j = 1$ to d do
        $C[j, h_j(x)] = C[j, h_j(x)] + 1$
If Delete(x) arrives
    for $j = 1$ to d do
        $C[j, h_j(x)] = C[j, h_j(x)] - 1$
If Query(x) arrives, then
    return $m'_x \triangleq \min_{1 \leq i \leq d} C[j, h_j(x)]$



Theorem: The estimate m'_x satisfies $m'_x \geq m_x$, and w. p. at least $1 - \delta$ it holds $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1 - \delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1 - \delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$.

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1 - \delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$.

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1 - \delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w}$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Hence,

$$\mathbb{P}\left[m'_x \geq m_x + \varepsilon \cdot F_1\right] = \mathbb{P}\left[\forall j : C[j, h_j(x)] \geq m_x + \varepsilon \cdot F_1\right]$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Hence,

$$\begin{aligned}\mathbb{P}\left[m'_x \geq m_x + \varepsilon \cdot F_1\right] &= \mathbb{P}\left[\forall j : C[j, h_j(x)] \geq m_x + \varepsilon \cdot F_1\right] \\ &= \mathbb{P}\left[\forall j : m_x + Z_{j,x} \geq m_x + \varepsilon \cdot F_1\right]\end{aligned}$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1 - \delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Hence,

$$\begin{aligned}\mathbb{P}\left[m'_x \geq m_x + \varepsilon \cdot F_1\right] &= \mathbb{P}\left[\forall j : C[j, h_j(x)] \geq m_x + \varepsilon \cdot F_1\right] \\ &= \mathbb{P}\left[\forall j : m_x + Z_{j,x} \geq m_x + \varepsilon \cdot F_1\right] = \mathbb{P}\left[\forall j : Z_{j,x} \geq \varepsilon \cdot F_1\right]\end{aligned}$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Hence,

$$\begin{aligned}\mathbb{P}\left[m'_x \geq m_x + \varepsilon \cdot F_1\right] &= \mathbb{P}\left[\forall j : C[j, h_j(x)] \geq m_x + \varepsilon \cdot F_1\right] \\ &= \mathbb{P}\left[\forall j : m_x + Z_{j,x} \geq m_x + \varepsilon \cdot F_1\right] = \mathbb{P}\left[\forall j : Z_{j,x} \geq \varepsilon \cdot F_1\right] \\ &\leq \mathbb{P}\left[\forall j : Z_{j,x} \geq \mathrm{e} \cdot \mathbb{E}[Z_{j,x}]\right]\end{aligned}$$

Analysis of Count-Min Sketch

Theorem

The estimate m'_x satisfies $m'_x \geq m_x$, and with probability at least $1-\delta$ it holds that $m'_x \leq m_x + \varepsilon \cdot F_1$, where F_1 is the first moment of the multiset S.

Proof: Clearly, for any x and j it holds that $C[j, h_j(x)] \geq m_x$, so $m'_x \geq m_x$.

Now for the second statement. Let $Z_{j,x}$ be the number of items $y \in [N] \setminus \{x\}$ such that $h_j(x) = h_j(y)$. Then we have that $C[j, h_j(x)] = m_x + Z_{j,x}$. Since we use a universal family of hash functions, it holds that

$$\mathbb{P}[h_j(x) = h_j(y)] \leq \frac{1}{w} \leq \frac{\varepsilon}{\mathrm{e}} \qquad \Rightarrow \qquad \mathbb{E}[Z_{j,x}] \leq \frac{\varepsilon}{\mathrm{e}} \cdot F_1.$$

Hence,

$$\begin{aligned}
\mathbb{P}\left[m'_x \geq m_x + \varepsilon \cdot F_1\right] &= \mathbb{P}\left[\forall j : C[j, h_j(x)] \geq m_x + \varepsilon \cdot F_1\right] \\
&= \mathbb{P}\left[\forall j : m_x + Z_{j,x} \geq m_x + \varepsilon \cdot F_1\right] = \mathbb{P}\left[\forall j : Z_{j,x} \geq \varepsilon \cdot F_1\right] \\
&\leq \mathbb{P}\left[\forall j : Z_{j,x} \geq \mathrm{e} \cdot \mathbb{E}[Z_{j,x}]\right] \leq \mathrm{e}^{-d} \leq \delta.
\end{aligned}$$

Markov inequality

Distributed frequency estimation with the CM sketch

Setup: Dataset is arbitrarily allocated in different servers.

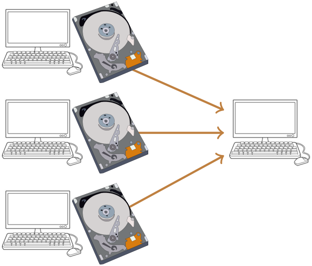
Objective: Design a communication-efficient algorithm for frequency estimation.

Distributed frequency estimation with the CM sketch

Setup: Dataset is arbitrarily allocated in different servers.

Objective: Design a communication-efficient algorithm for frequency estimation.

A naive approach:



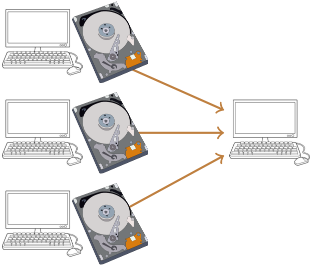
- Every site sends all the received data to a host server;
- The host maintains the CM sketch;

Distributed frequency estimation with the CM sketch

Setup: Dataset is arbitrarily allocated in different servers.

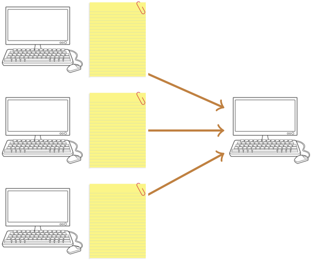
Objective: Design a communication-efficient algorithm for frequency estimation.

A naive approach:



- Every site sends all the received data to a host server;
- The host maintains the CM sketch;

A communication-efficient way:



- The sites communicate initially to use the same hash functions.
- All the sites maintains their own CM sketch;
- The sites send their CM sketches to the host.

Discussions of the CM Sketch

- The analysis is only based on Markov inequality (no Chebyshev inequality, no variance calculation). This gives us a very simple proof, and the space complexity proportional to $1/\varepsilon$.

Discussions of the CM Sketch

- The analysis is only based on Markov inequality (no Chebyshev inequality, no variance calculation). This gives us a very simple proof, and the space complexity proportional to $1/\varepsilon$. Notice that the space usage of our previous streaming algorithms is proportional to $1/\varepsilon^2$.
  - Think of $\varepsilon = 0.01$. This improvement from $1/\varepsilon^2$ to $1/\varepsilon$ represents reducing the space usage by 100 times!
- Since each entry is non-negative, the CM Sketch returns the ***minimum*** value instead of the ***medium*** value.

Discussions of the CM Sketch

- The analysis is only based on Markov inequality (no Chebyshev inequality, no variance calculation). This gives us a very simple proof, and the space complexity proportional to $1/\varepsilon$. Notice that the space usage of our previous streaming algorithms is proportional to $1/\varepsilon^2$.
  - Think of $\varepsilon = 0.01$. This improvement from $1/\varepsilon^2$ to $1/\varepsilon$ represents reducing the space usage by 100 times!
- Since each entry is non-negative, the CM Sketch returns the ***minimum*** value instead of the ***medium*** value.
- The error bound is **one-sided**. This feature is crucial for many applications.

Discussions of the CM Sketch

- The analysis is only based on Markov inequality (no Chebyshev inequality, no variance calculation). This gives us a very simple proof, and the space complexity proportional to $1/\varepsilon$. Notice that the space usage of our previous streaming algorithms is proportional to $1/\varepsilon^2$.
  - Think of $\varepsilon = 0.01$. This improvement from $1/\varepsilon^2$ to $1/\varepsilon$ represents reducing the space usage by 100 times!
- Since each entry is non-negative, the CM Sketch returns the ***minimum*** value instead of the ***medium*** value.
- The error bound is **one-sided**. This feature is crucial for many applications.
- The paper introducing the CM sketch has received more than 1,100 citations (checked in October 2018), which is very unusual for a theory paper.
- For further discussion, see https://sites.google.com/site/countminsketch/home

Summary

- Key features of streaming algorithms:

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.
  - A streaming algorithm for frequency estimation.

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.
  - A streaming algorithm for frequency estimation.
- Other problems investigated in the setting of streaming algorithms:

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.
  - A streaming algorithm for frequency estimation.
- Other problems investigated in the setting of streaming algorithms:
  - Approximating certain norms of matrices

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.
  - A streaming algorithm for frequency estimation.
- Other problems investigated in the setting of streaming algorithms:
  - Approximating certain norms of matrices
  - Counting the number of certain subgraphs in a graph

Summary

- Key features of streaming algorithms:
  - It is required to read the dataset only once with a certain order.
  - Algorithm's working space is sublinear in the size of the dataset, so storing an entire input is impossible.
  - Algorithm is required to output a good approximate answer with high probability.
- What have we seen:
  - Streaming algorithms for F_p-norm approximation.
  - A streaming algorithm for frequency estimation.
- Other problems investigated in the setting of streaming algorithms:
  - Approximating certain norms of matrices
  - Counting the number of certain subgraphs in a graph
  - and much more...