SS-ZG548: ADVANCED DATA MINING

08

Frequent Items Count Distinct

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Sept 11, 2021 ONLINE (WILP @ BITS-Pilani July-Dec 2021)

http://ktiwari.in/adm

Recap: Data streams

Consider, stream of data. Where data in arriving in rapid succession. Re-scan is NOT possible. Even storage space is insufficient to accommodate all data points.

Without storing all the data one wish to estimate

- Set of frequent items
- Number of distinct items
- Frequent itemsets
- *etc*

from first n Natural numbers, without repetition, in an arbitrary order. Can you report the missing one?

[Constraints are on memory and processing power]

Frequent items over data stream

- Let identity of items is drawn from the set $\{1, 2, 3, ..., n\}$.
- Frequency of item i be f_i
- Assume general arrival model, (i, v), $v > 0$ represents arrival and $v < 0$ is departure.
- Sum of frequencies $m = \sum_i f_i$ represent size of data stream
- **Frequent** item i, have **frequency** $f_i > m/(k+1)$ for some fixed k

Observations

- There could be at most k frequent items
  (why ? proof?) $m > k(k+1)$
- Any algorithm that finds all frequent and only frequent items requires at least $\log\binom{n}{k}$ bits
  (how? $2^s \geq \binom{n}{k}$)

Find approx frequent items

Wish to output a list of items such that

1. Every item in the list has frequency $f_i > (1 - \epsilon)\frac{m}{k+1}$
2. All the items having frequency at least $(1 + \epsilon)\frac{m}{k+1}$ is in the list

Output should satisfy above two properties with probability $(1 - \delta)$

Find approx frequent items

Wish to output a list of items such that

1. Every item in the list has frequency $f_i > (1-\epsilon)\frac{m}{k+1}$
2. All the items having frequency at least $(1+\epsilon)\frac{m}{k+1}$ is in the list

Output should satisfy above two properties with probability $(1-\delta)$

Algorithm maintains a data structure A over the stream.

Step to update an item x is as below

1. **IF** (A.ismember(x)) A[x]$++$
2. **ELSE** A.insert(x)
3. **IF** (A.size == k+1) **THEN** $\forall\, y \in A$
4. A[y]$--$,
5. **IF** (A[y] ==0) A.delete(y);

In action: Frequent items

Take $k = 4$, and consider following data stream

5	8	4	5	4	12	$\rightarrow$	
	6	5	2	8	3	5	$\rightarrow$
		4	5	4	12	6	13

Insert *x* in data structure

1. **IF** (A.ismember(x)) A[x]++
2. **ELSE** A.insert(x)
3. **IF** (A.size == k+1) **THEN** $\forall\ y \in A$
4. A[y]−−,
5. **IF** (A[y] ==0) A.delete(y);

Let us step by step execute the algorithm:

Index	Item	Frequency
1		
2		
3		
4		
5		

Count distinct over data streams (FM sketch)

Estimate number of distinct items in data stream

- If $x = ???..??1\overbrace{000...0}^{i-1}$ then L[x]=i
- Probability of L[x]=i is $p_i = \frac{2^{\log|F|-i}}{|F|} = 1/2^i$ when $x \in \{1, 2, .., F\}$
- FM sketch is a bitmap A of size $\log|F|$ with hash a function h
- Arrival of an item x, sets bit $A[L[h(x)]] \leftarrow 1$.
  Probability that $A[i] = 1$ after seeing n items is $1 - (1 - p_i)^n$
- With s independent copies of FM sketch, let $\#A[i]$ represent count of 1's at level i and $\hat{q}_i = \frac{\#A[i]}{s}$. Then choose i, such that $\hat{q}_i \geq \frac{3}{\epsilon^2} \log \frac{1}{\delta}$. By Chernoff's bound $\hat{n} \in [(1-\epsilon)E[n], (1+\epsilon)E[n]]$ with probability $(1-\delta)$

$$\hat{n} = \frac{\log(1 - \hat{q}_i)}{\log(1 - p_i)}$$

In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...

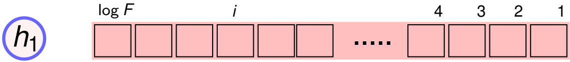
In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...

In action: Count distinct over data streams

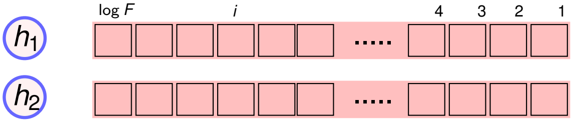
Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →

In action: Count distinct over data streams



Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... $\rightarrow$

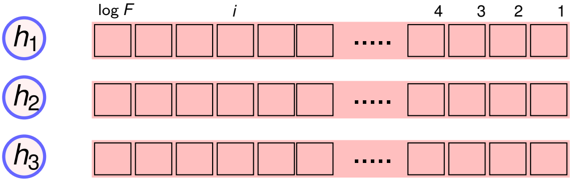
In action: Count distinct over data streams



Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →

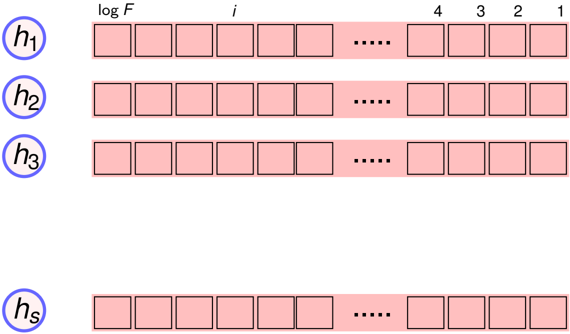
In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... $\rightarrow$



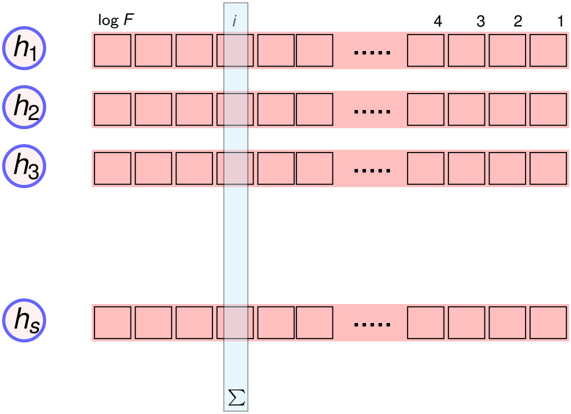
In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →



In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →



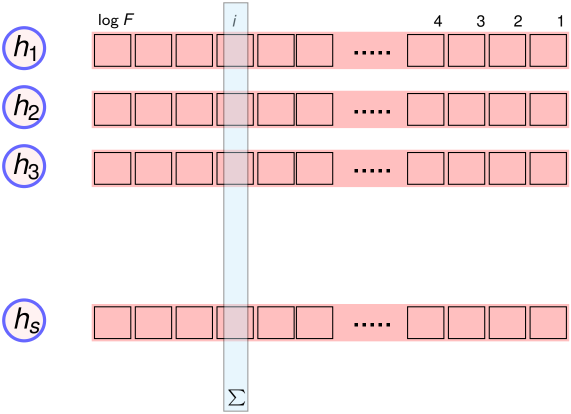
$\frac{3}{\epsilon^2} \log \frac{1}{\delta}$

$p_i = \frac{1}{2^i}$

In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →

$$\hat{q}_i = \frac{\sum}{s}$$



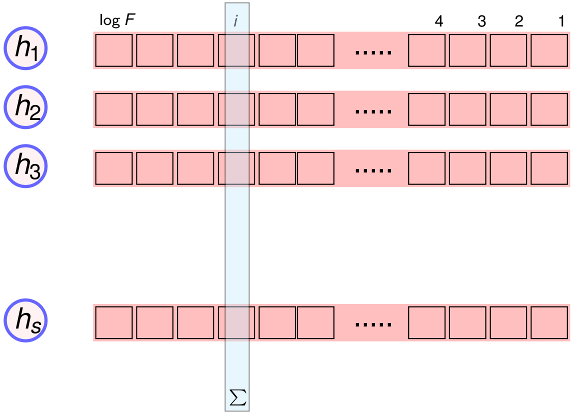
$$\frac{3}{\epsilon^2} \log \frac{1}{\delta} \qquad p_i = \frac{1}{2^i}$$

In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →

$$\hat{q}_i = \frac{\sum}{s}$$

$$\hat{n} = \frac{\log(1-\hat{q}_i)}{\log(1-p_i)}$$

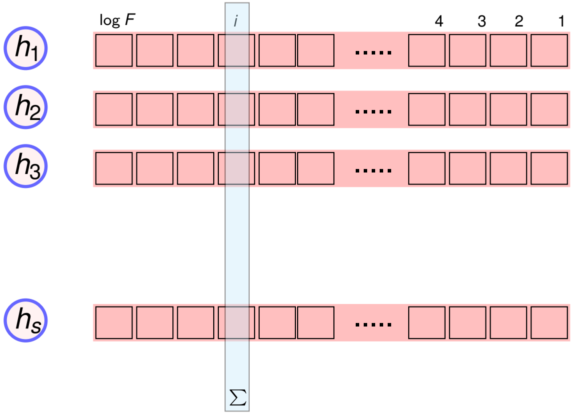


$$\frac{3}{\epsilon^2} \log \frac{1}{\delta}$$

$$p_i = \frac{1}{2^i}$$

In action: Count distinct over data streams

Stream:
25, 10 ,18 ,25, 06, 03, ...
10, 8, 2, 5, 18, 12, 9, 6, ...
12, 6, 11, 15, 5, 6, 13, 6, 8, ... →



$$\frac{3}{\epsilon^2} \log \frac{1}{\delta} \qquad p_i = \frac{1}{2^i}$$

$$\hat{q}_i = \frac{\Sigma}{s}$$

$$\hat{n} = \frac{\log(1-\hat{q}_i)}{\log(1-p_i)}$$

$\hat{n} \in [(1-\epsilon)E[n], (1+\epsilon)E[n]]$ with probability $(1-\delta)$

Frequent pattern mining over data streams

- Applications involves retail market data analysis, network monitoring, web usage mining, and stock market prediction.
- Using sliding window
- Efficiently remove the obsolete, old stream data
- Compact Pattern Stream tree (CPS-tree)
- Highly compact frequency-descending tree structure at runtime
- Efficient in terms of memory and time complexity
- Pane and window
- Insertion and restructuring

Thank You!

Thank you very much for your attention!

Queries ?