Compact data structures: data streaming

Luca Becchetti

“Sapienza” Università di Roma – Rome, Italy

November 19, 2015

Some typical queries [Muthukrishnan, 2005a]

- How many distinct IP addresses use a given link currently or anytime during the day?
- What are the top k voluminous flows currently in progress in a link?
- How many *distinct* flows were observed?
- Are traffic patterns in two routers correlated? What are (un)usual trends?

Network monitoring just one possible application

- On-line statistics on search engines' query logs
- On-line statistics on server logs
- Finding near-duplicate Web pages

Informally...

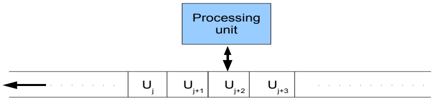
Streaming involves ([Muthukrishnan, 2005a]):

- Small number of passes over data. (Typically 1?)
- Sublinear space (sublinear in the universe or number of stream items?)

A model of computation...

- Similar to dynamic, online, approximation or randomized algorithms, but with more constraints
- Constraints impose limitations that make many "easy" problems hard (further in this lecture)
- Being poly-time/poly-space no longer sufficient

The streaming model [Muthukrishnan, 2005b]



Streaming model

Data flow

- Data item arrive over time
- U_j processed before U_{j+1} arrives
- Only one pass (or few passes, at most $O(\log)$)

The streaming model [Muthukrishnan, 2005b]

- Underlying data (signal): an n-dimensional array $\mathbf{A}$, n typically large (e.g., the size of the IP address space)
- Update arrive over time. The j-th update is a pair $U_j = (i, x)$, where i is an item (index):

$$\mathbf{A}_i = \mathbf{A}_i + x$$

- In general, x can be any
- Initially: $\mathbf{A}_i = 0, \forall i = 1, \ldots n$
- $\mathbf{A}(t)$: the state of the array after the first t updates

Goal

- Compute and maintain functions over $\mathbf{A}$ in small space, with fast updates and computation
- typically, space $<< n$

Caveats about updates, items and values

j-th update $U_j = (i, x)$

- item i is from a discrete universe of finite size
- value x
- Example
  - i = (Source IP, Dest. IP, Protocol)
  - x = packet size (in bytes)

Wlog, i can be considered an integer

Hash to an integer otherwise. Example:

- i = ("151.100.12.3", "210.15.0.2", "TCP")
- i $\rightarrow$ h(i), where h($\cdot$) maps strings to integers
- For example, the integer corresponding to the concatenation of the strings

Special cases of the model [Muthukrishnan, 2005b]

- j-th update changes $\mathbf{A}(j)$ (Time series model)
- **Cash register:** $x \geq 0$
- Turnstile model: most general model

In this lecture

- Compact summaries of data streams (Count-Min sketches [Cormode and Muthukrishnan, 2005])
- Statistics: point queries, heavy hitters, join-size, No. *distinct* items
- Only a drop in a sea of results...

Statistics and aggregates

- Point query $\mathcal{Q}(i)$: estimate $\mathbf{A}_i$
  - basic building block for more complex queries
- ϕ-heavy hitters of $\mathbf{A}$: return all $\{i : \mathbf{A}_i > \phi\|\mathbf{A}\|_1\}$

Other aggregates (see further)

- Join size of two DB relations observed in a streaming fashion
- Scalar product of two streams (viewed as two vectors of same size)
- Number of distinct items observed in $\mathbf{A}$

Markov's and Chebyshev's inequalities

Theorem (Markov's inequality)

Let X denote a random variable that assumes only non-negative values. Then, for every $a > 0$:

$$\mathbf{P}[X \geq a] \leq \frac{\mathbf{E}[X]}{a}.$$

Theorem (Chebyshev's inequality)

Let X denote a random variable. Then, for every $a > 0$:

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq a] \leq \frac{\mathbf{v}ar\,[X]}{a^2}.$$

Families of universal hash functions

We assume we have a suitably defined family $\mathcal{F}$ of hash functions, such that every member of $h \in \mathcal{F}$ is a function $h : U \to [n]$.

Definition

$\mathcal{F}$ is a 2-universal hash family if, for any $h(\cdot)$ chosen *uniformly at random* from $\mathcal{F}$ and for every $x, y \in U$ we have:

$$\mathbf{P}[h(x) = h(y)] \leq \frac{1}{n}.$$

- Definitions generalizes to k-universality [Mitzenmacher and Upfal, 2005, Section 13.3]
- **Problem:** define "compact" universal hash families

A 2-universal family

Assume $U = [m]$ and assume the range of the hash functions we use is $[n]$, where $m \geq n$ (typically, $m >> n$). We consider the family $\mathcal{F}$ defined by $h_{ab}(x) = ((ax + b) \mod p) \mod n$, where $a \in \{1, \ldots, p-1\}$, $b \in \{0, \ldots, p\}$ and p is a prime $p \geq m$.

How to choose u.a.r. from $\mathcal{F}$

For a given p: Simply choose a u.a.r. from $\{1, \ldots, p-1\}$ and b u.a.r. from $\{0, \ldots, p\}$

A 2-universal family/cont.

Theorem ([Carter and Wegman, 1979, Mitzenmacher and Upfal, 2005])

$\mathcal{F}$ is a 2-universal hash family. In particular, if a, b are chosen uniformly at random:

$$\mathbf{P}[h_{ab}(x) = i] = \frac{1}{n}, \forall x \in U, i \in [n].$$

$$\mathbf{P}[h_{ab}(x) = h_{ab}(y)] \leq \frac{1}{n}, \forall x, y \in U.$$

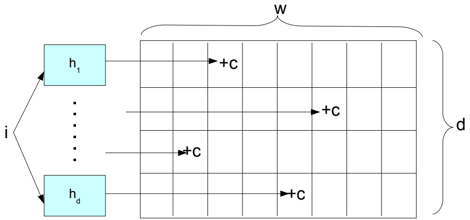
The CM sketch [Cormode and Muthukrishnan, 2005]

- *In the remainder: cash-register model*
- 2-Dimensional array whose size is determined by design parameters ϵ and δ (their meaning explained further)
- Array is $C[j, l]$, where $j = 1, \ldots, d$ and $l = 1, \ldots, w$
  - $d = \lceil \ln \frac{1}{\delta} \rceil$ (depth)
  - $w = \lceil \frac{e}{\epsilon} \rceil$ (width)
- Every entry initially 0
- d hash functions $h_1, \ldots, h_d$ chosen uniformly at random from a 2-universal (pairwise-independent) family (see first lecture)
- $h_r : \{1, \ldots, n\} \rightarrow \{1, \ldots, w\}$

Update

Pair (i, c) is observed, meaning that, *ideally*, $\mathbf{A}_i = \mathbf{A}_i + c$

CM sketch: Update procedure



CM sketch update

update(i, c)

Require: i: array index, c: value

1: **for** j : 1 . . . d **do**

2: C[j, $h_j(i)$] = C[j, $h_j(i)$] + c

3: **end for**

Point query

- Basic query, building block for the others
- $\mathcal{Q}(i)$: estimate $\mathbf{A}_i$

Point query estimate

```
PQ(i)
```

Require: `i: array index`

1: `return` $\hat{\mathbf{A}}_i = min_j$ `C[j,` $h_j(i)$`]`

Theorem ([Cormode and Muthukrishnan, 2005])

$\hat{\mathbf{A}}_i \geq \mathbf{A}_i$. *Furthermore,* $\mathbf{P}\left[\hat{\mathbf{A}}_i > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1\right] \leq \delta$, *where* $\|\mathbf{A}\|_1 = \sum_{i=1}^n |\mathbf{A}_i|$ *is the* 1*-norm of* $\mathbf{A}$.

Proof of theorem

- Define $I_{ijk} = 1$ if $(i \neq k) \bigcap (h_j(i) = h_j(k))$, 0 otherwise
- $\mathbf{P}[h_j(i) = h_j(k)] \leq \frac{1}{w} \leq \frac{\epsilon}{e}$ by pairwise independence
- Define $X_{ij} = \sum_{k=1}^{n} I_{ijk} \mathbf{A}_k$
- $X_{ij} \geq 0$ and $C[j, h_j(i)] = \mathbf{A}_i + X_{ij} \rightarrow \hat{\mathbf{A}}_i \geq \mathbf{A}_i$
- X_{ij} is the error introduced by collisions
- $\mathbf{E}[X_{ij}] = \mathbf{E}[\sum_{k=1}^{n} I_{ijk} \mathbf{A}_k] = \sum_{k=1}^{n} \mathbf{A}_k \mathbf{E}[I_{ijk}] \leq \frac{\epsilon}{e} \|\mathbf{A}\|_1$
- Notice that the only random variables are the I_{ijk}'s and the X_{ij}'s

Furthermore,

$$\mathbf{P}\left[\hat{\mathbf{A}}_i > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1\right] = \mathbf{P}[\forall j : C[j, h_j(i)] > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1]$$
$$= \mathbf{P}[\forall j : \mathbf{A}_i + X_{ij} > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1] = \mathbf{P}[\forall j : X_{ij} > e\mathbf{E}[X_{ij}]]$$
$$= \Pi_{j=1}^{d} \mathbf{P}[X_{ij} > e\mathbf{E}[X_{ij}]] < e^{-d} \leq \delta,$$

where the fifth inequality follows from Markov's inequality.

Heavy hitters [Cormode and Muthukrishnan, 2005]

- ϕ-heavy hitters of $\mathbf{A}$: $\{i : \mathbf{A}_i > \phi\|\mathbf{A}\|_1\}$
- **Fact:** No. ϕ-heavy hitters between 0 and $1/\phi$
- *Approximate* heavy hitters: accept i such that $\mathbf{A}_i \geq (\phi - \epsilon)\|\mathbf{A}\|_1$ for some specified $\epsilon < \phi$
- We consider the cash-register model

Heavy hitters algorithm: ingredients

- CM sketch and point query basic building blocks
- Return items whose estimate exceeds $\phi\|\mathbf{A}\|_1$
- Assume c_s is the s-th update $\rightarrow \|\mathbf{A}\|_1 = \sum_{s=1}^{t} c_s \rightarrow \|\mathbf{A}\|_1$ can be easily maintained and updated in small space

Heavy hitters: update and query

update(i, c, S, H)

Require: i: array index, $c \geq 0$, S: CM sketch, H: heap

update(i, c, S) {Update CM sketch}

$\hat{\mathbf{A}}_i$ = PQ(i)

if $\hat{\mathbf{A}}_i > \phi \|\mathbf{A}\|_1$ **then**

  HeapUpdate(i, $\hat{\mathbf{A}}_i$, H) {Insert if $i \notin H$}

end if

s = HeapMin(H)

while PQ(s) $\leq \phi \|\mathbf{A}\|_1$ do

  HeapDelete(s)

  s = HeapMin(H)

end while

Heap and query

- Generic heap element: pair $(i, \hat{\mathbf{A}}_i)$ ordered by $\hat{\mathbf{A}}_i$
- Heavy hitters: return all elements i in H such that $\hat{\mathbf{A}}_i > \phi \|\mathbf{A}\|_1$

Heavy hitters: performance

Theorem ([Cormode and Muthukrishnan, 2005])

Assume Inserts only (cash register model). With CM sketches using space $O\left(\frac{1}{\epsilon}\log\frac{\|\mathbf{A}\|_1}{\delta}\right)$ *and update time* $O\left(\log\frac{\|\mathbf{A}\|_1}{\delta}\right)$ *per item:*

- *Every heavy hitter is output*
- *With probability at least* $1-\delta$ *: i) no item whose* real *count is* $\leq (\phi-\epsilon)\|\mathbf{A}\|_1$ *is output and ii) the number of items in the heap is* $O\left(\frac{1}{\phi-\epsilon}\right)$

Question

Assume $d = \left\lceil\frac{e}{\epsilon}\right\rceil$ and $w = \left\lceil\ln\frac{n}{\delta}\right\rceil$ and let $\mathcal{T}$ be the estimated set of heavy hitters. Recall that $\hat{\mathbf{A}}_i \geq \mathbf{A}_i$. After any number t of insertions, define $S_\epsilon = \{i : \mathbf{A}_i < (\phi-\epsilon)\|\mathbf{A}\|_1\}$. Prove that

$$\mathbf{P}[S_\epsilon \cap \mathcal{T} \neq \emptyset] \leq \delta.$$

Solution

Consider $i \in S_\epsilon$. If $i \in \mathcal{T}$,
$\hat{\mathbf{A}}_i > \phi\|\mathbf{A}\|_1 \rightarrow \forall j : C[j, h_j(i)] > \phi\|\mathbf{A}\|_1$. Hence:

$$\mathbf{A}_i < (\phi - \epsilon)\|\mathbf{A}\|_1 \rightarrow C[j, h_j(i)] - \mathbf{A}_i > \epsilon\|\mathbf{A}\|_1, \forall j.$$

This implies:

$$\mathbf{P}[i \in \mathcal{T}] = \mathbf{P}\left[\hat{\mathbf{A}}_i > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1\right]$$
$$= \mathbf{P}[\forall j : C[j, h_j(i)] > \mathbf{A}_i + \epsilon\|\mathbf{A}\|_1] < e^{-d} = \frac{\delta}{n},$$

where the third inequality follows from the general result seen for `PQ(i)`. Finally, since $|S_\epsilon| \leq n$:

$$\mathbf{P}[S_\epsilon \cap \mathcal{T} \neq \emptyset] = \mathbf{P}[\cup_{i \in S_\epsilon}(i \in \mathcal{T})] \leq \frac{\delta}{n} \cdot |S_\epsilon| \leq \delta.$$

Carter, J. L. and Wegman, M. N. (1979).
Universal classes of hash functions.
Journal of Computer and System Sciences, 18(2):143–154.

Cormode, G. and Muthukrishnan, S. (2005).
An improved data stream summary: the count-min sketch and its applications.
J. Algorithms, 55(1):58–75.

Mitzenmacher, M. and Upfal, E. (2005).
Probability and Computing : Randomized Algorithms and Probabilistic Analysis.
Cambridge University Press.

Muthukrishnan, S. (2005a).
Data stream algorithms. URL: http://www.cs.rutgers.edu/˜muthu/str05.html.

Muthukrishnan, S. (2005b).

Data streams: Algorithms and applications.

In *Foundations and Trends in Theoretical Computer Science, Now Publishers or World Scientific*, volume 1.

Draft at authors homepage: http://www.cs.rutgers.edu/~muthu/stream-1-1.ps.