CS-E4600
Mining data streams II
slide set 8

Aristides Gionis
Department of Computer Science
Aalto University

reading assignment

- LRU book: chapter 4
- recent Communications of the ACM paper by Cormode [Cormode, 2017]
- optional reading
  - paper by Charikar, Chen, and Farach-Colton [Charikar et al., 2002]
  - paper by Cormode and Muthukrishnan [Cormode and Muthukrishnan, 2005]

finding frequent items in a data stream

- consider again a data stream
- $X = (x_1, x_2, \ldots, x_m)$ a data stream
- each x_i is a member of the set $N = \{1, \ldots, n\}$
- $m_i = |\{j : x_j = i\}|$ the number of occurrences of i
- $f_i = m_i/m$ the frequency of item i

- problem : estimate most frequent items in data stream

finding frequent items in a data stream

- problem formalization
- rename items $\{o_1, \ldots, o_n\}$ so that $m_1 \geq \ldots \geq m_n$
- given $k < n$ want to return top-k items $o_1, \ldots, o_k$

finding frequent items in a data stream

- problem formalization — first attempt

- problem $\textsc{FindCandidateTop}(X, k, \ell)$
  - given stream X and integers k and ℓ
  - return list of ℓ items, so that top most frequent k items of X occur in the list

- should return all most frequent items

finding frequent items in a data stream

- $\textsc{FindCandidateTop}(X, k, \ell)$ can be too hard to solve
- consider the case $m_k = m_{\ell+1} + 1$
  - i.e., number of occurences of k-th frequent item exceeds only by 1 the number of occurences of the $(\ell + 1)$-th frequent item
- almost impossible to find a list that contains the k most frequent items

finding frequent items in a data stream

- problem formalization — second attempt

- problem $\textsc{FindApproxTop}(X, k, \epsilon)$
  - given stream X, integer k, and real $\epsilon < 1$
  - return list of k items, so that for each item i in the list it is $m_i \geq (1 - \epsilon) m_k$

- no guarantee to return all most frequent items, but if return an item it should be frequent enough

finding frequent items in a data stream

- problem : $\textsc{FindApproxTop}(X, k, \epsilon)$

- algorithm : $\textsc{CountSketch}$
  - based on sketching techniques

- intuition
  - use a hash function s and a counter c
  - function s hashes objects to $\{-1, +1\}$
  - for each item o_i seen in the stream, set $c \leftarrow c + s[o_i]$
  - then $\mathbb{E}\left[c \cdot s[o_i]\right] = m_i$ (prove it!)
  - so, estimate m_i by $c \cdot s[o_i]$

the CountSketch algorithm

- problem with using one hash function and one counter
  - very high variance

- remedy 1
  use t hash functions $s_1, \ldots, s_t$ and t counters $c_1, \ldots, c_t$
  - for each item o_i seen in the stream,
    set $c_j \leftarrow c_j + s_j[o_i]$, for all $j = 1, \ldots, t$
  - to estimate m_i take median of $\{c_1 \cdot s_1[o_i], \ldots, c_t \cdot s_t[o_i]\}$
    (as before $\mathbb{E}\left[c_j \cdot s_j[o_i]\right] = m_i$ for all $j = 1, \ldots, t$)

the CountSketch algorithm

- problem with using one hash function and one counter
  - very high variance
- remedy 1

  use t hash functions $s_1, \ldots, s_t$ and t counters $c_1, \ldots, c_t$
  - for each item o_i seen in the stream,

    set $c_j \leftarrow c_j + s_j[o_i]$, for all $j = 1, \ldots, t$
  - to estimate m_i take median of $\{c_1 \cdot s_1[o_i], \ldots, c_t \cdot s_t[o_i]\}$

    (as before $\mathbb{E}[c_j \cdot s_j[o_i]] = m_i$ for all $j = 1, \ldots, t$)

the CountSketch algorithm

- problem with previous idea
  - high-frequency items (e.g., o_1) may spoil estimates of lower-frequency items (e.g., o_k)
- remedy 2
  - do not update all counters with all items
  - replace each counter with a hash table of b counters
  - items update different subsets of counters, one per hash table
  - each item gets enough high-confidence estimates (those avoiding collisions with high-frequency elements)

the CountSketch algorithm

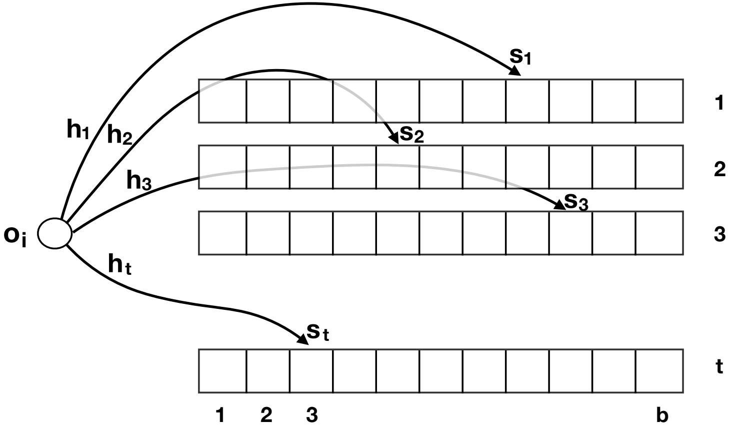
- problem with previous idea
  - high-frequency items (e.g., o_1) may spoil estimates of lower-frequency items (e.g., o_k)

- remedy 2
  - do not update all counters with all items
  - replace each counter with a hash table of b counters
  - items update different subsets of counters, one per hash table
  - each item gets enough high-confidence estimates (those avoiding collisions with high-frequency elements)

the CountSketch algorithm

- use parameters t and b
- let $h_1, \ldots, h_t$ be hash functions from items to $1, \ldots, b$
- let $s_1, \ldots, s_t$ be hash functions from items to $\{-1, +1\}$
- consider $t \times b$ table of counters
- for each item o_i seen in the stream,
  set $h_j[o_i] \leftarrow h_j[o_i] + s_j[o_i]$, for all $j = 1, \ldots, t$
- to estimate m_i take median of
  $\{h_1[o_i] \cdot s_1[o_i], \ldots, h_t[o_i] \cdot s_t[o_i]\}$

the CountSketch data structure



an improved data stream summary

- the CountMinSketch data stream summary
- see [Cormode, 2017]
- optional reading
  [Cormode and Muthukrishnan, 2005]

the CountMinSketch data stream summary

- limitations of existing sketches
  - model limitations (a sequence of items / numbers)
  - space required is $\mathcal{O}(\frac{1}{\epsilon^2})$

    recall that quarantees are quantified by ϵ, δ parameters

    ϵ : accuracy

    δ : probability of failure
  - update time proportional to the whole sketch
  - different sketch for each summary
- CountMinSketch addresses all those limitations

incremental data-stream model

- consider a vector $\mathbf{x}(t) = \{x_1(t), \ldots, x_n(t)\}$
- number of coordinates n potentially very large
- $\mathbf{x}(t)$ the values of vector at time t
- at each time t a vector coordinate is updated
- data stream : updates (i_t, c_t) for $t = 1, \ldots$
- then

$$x_{i_t}(t) \leftarrow x_{i_t}(t-1) + c_t$$

and

$$x_j(t) \leftarrow x_j(t-1), \text{ for } j \neq i_t$$

incremental data-stream model

- generalization of previous model

  previous model was $c_t = 1$

- special cases

– cash register model : $c_t \geq 0$

– turnstile model : c_t can be negative

  – non-negative turnstile model : $x_i(t) \geq 0$

  – general turnstile model : $x_i(t)$ can be negative

the CountMinSketch data stream summary

- interesting queries that we would like to handle

– point query $\mathcal{Q}(i)$: approximate x_i

– range query $\mathcal{Q}(\ell, r)$: approximate $\sum_{i=\ell}^{r} x_i$

– inner product $\mathcal{Q}(\mathbf{x}, \mathbf{y})$: approximate $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i \, y_i$

– ϕ-quantiles

– heavy-hitters : most frequent items
given frequency threshold ϕ, find items i for which
$x_i \geq (\phi - \epsilon)||\mathbf{x}||_1$ for some $\epsilon < \phi$

the CountMinSketch data structure

- similar to CountSketch
- a table of counters C of dimension $d \times w$
- d hash functions $h_1, .., h_d$ from $\{1, .., n\}$ to $\{1, .., w\}$ chosen from a pairwise-independent family

$$C = \begin{pmatrix} C[1,1] & \cdots & C[1,w] \\ \vdots & \ddots & \vdots \\ C[d,1] & \cdots & C[d,w] \end{pmatrix}$$

- parameters d and w specify the space requirements depend on error bounds we want to achieve

CountMinSketch : update summary

- given (i_t, c_t) update one counter in each row of C, in particular

$$C[j, h_j(i_t)] \leftarrow C[j, h_j(i_t)] + c_t$$

for all $j = 1, \ldots, d$

the CountMinSketch data structure

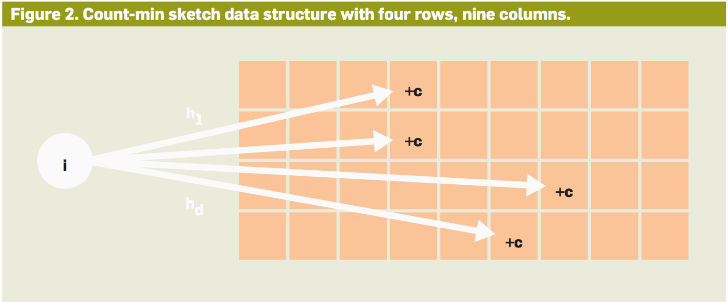


Figure from "Data Sketching", Cormode, CACM, 2017

CountMinSketch : point query

- the answer to $\mathcal{Q}(i)$ is $\hat{x}_i = \min_j C[j, h_j(i)]$
- theorem : the estimate $\hat{x}_i$ satisfies

  (i) $x_i \leq \hat{x}_i$

  (ii) $\hat{x}_i \leq x_i + \epsilon||\mathbf{x}||_1$ with prob at least $1 - \delta$

CountMinSketch

- similar type of estimates for other queries
  - range, inner product, etc.

- parameters are set to

$$d = \left\lceil \log \frac{1}{\delta} \right\rceil \quad \text{and} \quad w = \left\lceil \frac{1}{\epsilon} \right\rceil$$

  - improved space ; instead of usual $\mathcal{O}(\frac{1}{\epsilon^2})$
  - improved update time : access only d counters

references I

Charikar, M., Chen, K., and Farach-Colton, M. (2002).
Finding frequent items in data streams.
In *International Colloquium on Automata, Languages, and Programming*, pages 693–703.

Cormode, G. (2017).
Data sketching.
Communications of the ACM, 80(9):48–55.

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