Data Stream Algorithms II

Joaquim Madeira

Version 0.4 – December 2022

Overview

- Recap from last week
- Finding frequent items via Sketching

RECAP

Data Streams

- Many data generation processes can be modeled as data streams
  - Huge numbers of simple pieces of data
  - Arriving at enormous rates
  - Taken together lead to a complex whole
- Hundreds of gigabytes per day or higher !

Data Streams

- Such data may be archived and indexed within a data warehouse
- BUT it may also be important to process it "as it happens"
- Up to the minute analysis and statistics on current trends

The streaming model

- Data arrives in a streaming fashion
  - Scan the sequence in the given order
  - No random access to the data tokens !
- Must be processed on the fly !
- Accurate computations

The streaming model

- Compute some function $\Phi(\sigma)$ of a massively long input stream σ
- Make just one pass over σ !
- Goal:
  - Use resources (space and time) sublinear on the size of the input !

The basic streaming model

- The data stream:

$$\sigma = \langle a_1, a_2, \ldots, a_m \rangle$$

- Each data token a_i is drawn from a set of n elements

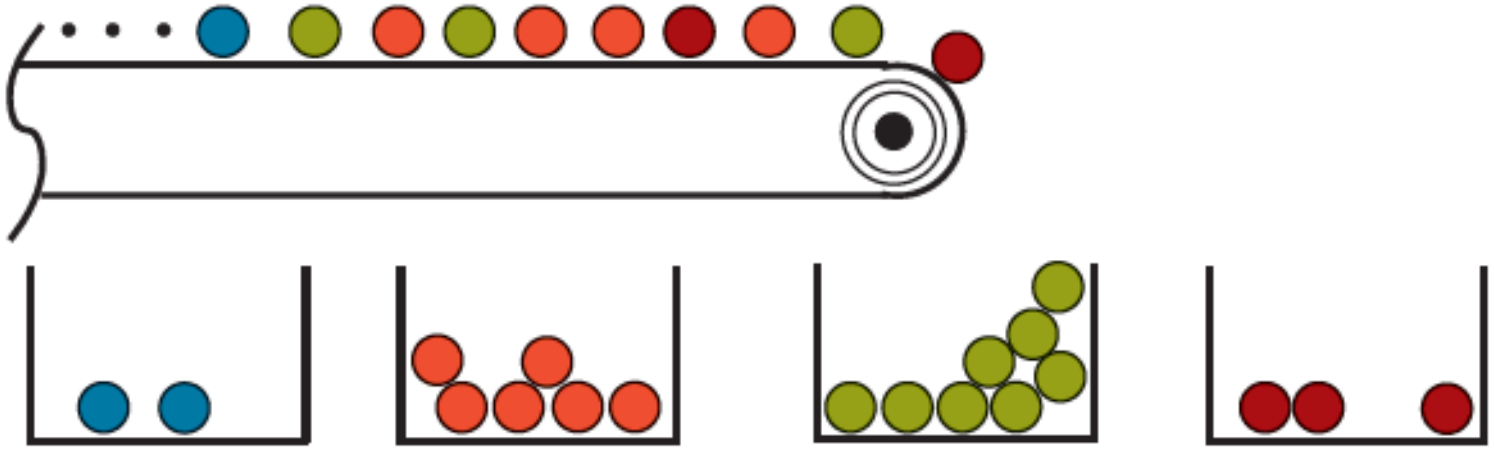
- Goal:
  - Process σ using a small amount of memory s
  - I.e., make s much smaller than m and n !

Finding frequent items

- The frequent items / "heavy-hitters" problem
- Given a sequence of items, identify those which occur most frequently
- More formally :
- Find all items whose frequency exceeds a specified fraction of the total number of items

Finding frequent items

Figure 1. A stream of items defines a frequency distribution over items. In this example, with a threshold of $\phi = 20\%$ over the 19 items grouped in bins, the problem is to find all items with frequency at least 3.8—in this case, the green and red items (middle two bins).



[Cormode and Hadjieleftheriou]

Finding frequent items

- Counter-based algorithms – Last Week
  - Track and maintain counts associated with a (varying) subset of stream items
- Sketch algorithms – Today !
  - Randomized approach
  - Do not explicitely store stream elements !
- Other approaches

SKETCHES

Data streams – Synopses

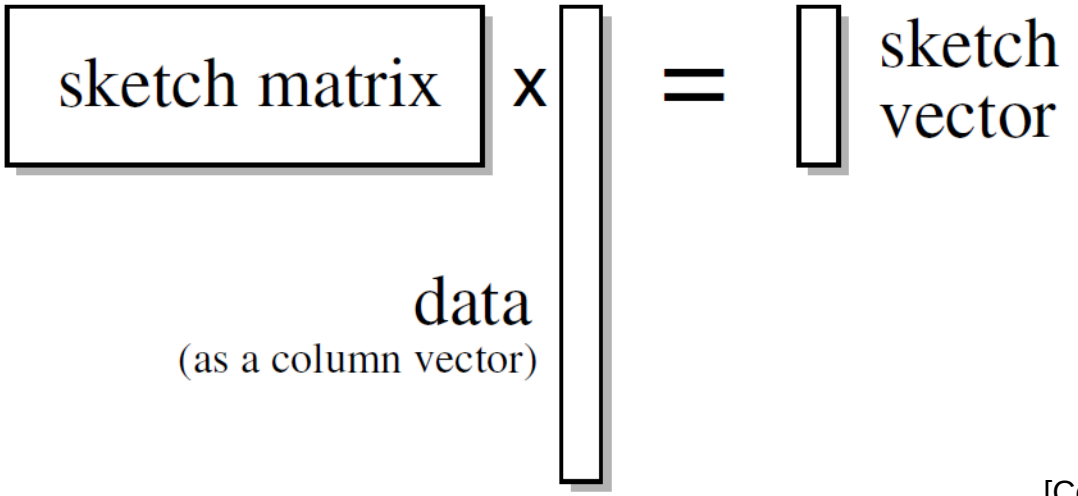
- High-speed data streams / massive data
  - Processing / exploring
- Fast / interactive response times
  - How ?
- Compute a lossy, compact synopsis of the data
  - Capturing the feature(s) of interest
- Execute queries on the synopsis
- Get accurate estimates

Data streams – Synopses

- Common synopses / summaries
  - Random samples
  - Histograms
  - Wavelets
  - Sketches
- Issues
  - Space and time efficiency
  - Accuracy and error bounds
  - …

Sketches

- Linear Sketches
- Data structures which can be represented as a *linear transform* of the input



[Cormode et al]

Sketches

- Well-suited for streaming data
  - Fast and compact + Continuous update
  - Update independent of current state
  - Easy to parallelize
- Frequency-based sketches
  - Summarize the frequency distribution
  - Heavy-hitters
  - Guaranteed accuracy

COUNTING ITEMS

Counting items

- Goal: determine the number of occurrences of items in a data stream
  - I.e., their frequencies

- BUT, no need for exact results
  - Good estimates suffice !

Counting items – Simple solution

- Simple solution
  - Construct a map / dictionary from items to counts
- Problems
  - Number of distinct items can be VERY LARGE !
  - Counting must be FAST !
- How to improve ?

Counting items – Idea

- Idea
  - Do not store pairs ($item_i$, $count_i$)
  - Store only the counters !!

- Create an array of size n, filled with zeros
- Map each item to an array index
  - Hash function
- Increment the corresponding counter

Counting items – Problem

- Query an item's count
  - Return the counter's value at its position
- Problem
  - There will be collisions !!
  - Counting too many times – Errors !
- How to improve ?

THE COUNT-MIN SKETCH

Counting items – Replicate

- Use multiple hash functions !!
  - Where have we seen that before ?
- AND multiple arrays of counters !!
  - One array for each hash function
- Question
  - When queried, what value should be returned ?

Counting items – What to return ?

- When queried, return the minimum counter value
  - The Count-Min Sketch
- Issue
  - There will still be collisions !!
  - But less…

The Count-Min Sketch – Demo

- Try the simple demo:

  http://hkorte.github.io/slides/cmsketch/#/7

- Insert several elements
- Compare the true and the sketch counts

- How to get better estimates ?

The Count-Min Sketch – Features

- Probabilistic data structure
  - Frequency table of items for a data stream
  - Sub-linear space
- May over-estimate the true counts !
  - But not under-estimate them !
  - Biased estimator
- The relative error may be large for low frequency items

The Count-Min Sketch – Usage

- Counting items on VERY LARGE data sets
  - When small errors are acceptable
  - E.g., Natural Language Processing : keep statistics on the frequency of word combinations
- Counting items in data streams

The Count-Min Sketch

- G. Cormode & S. Muthukrishnan
  - 2003 – Conference paper
  - 2005 – Journal paper

- An improved data stream summary: the count-min sketch and its applications
  - Journal of Algorithms, 2005
  - www.sciencedirect.com/science/article/pii/S0196677403001913

The Count-Min Sketch – Links

- H. Korte, A practical introduction to the Count-Min Sketch
  - hkorte.github.io/slides/cmsketch/#/
- Count-Min Sketch Repository
  - sites.google.com/site/countminsketch/home
- …

The Count-Min Sketch – Links

- Java – Stream summarizer and cardinality estimator
  - github.com/addthis/stream-lib
- Python – A minimalistic Count-min Sketch
  - https://github.com/rafacarrascosa/countminsketch
- Python – Probabilistic string counting
  - https://github.com/AWNystrom/CoutnMinSketch

Similar idea ?

- Where have we seen a similar idea ?
- The counting Bloom filter
- But, different usages and diferente sizes !
  - Counting Bloom filter : number of elements in the multi-set
  - Count-min sketch : sub-linear number of cells
    - Relate to the desired approximation quality

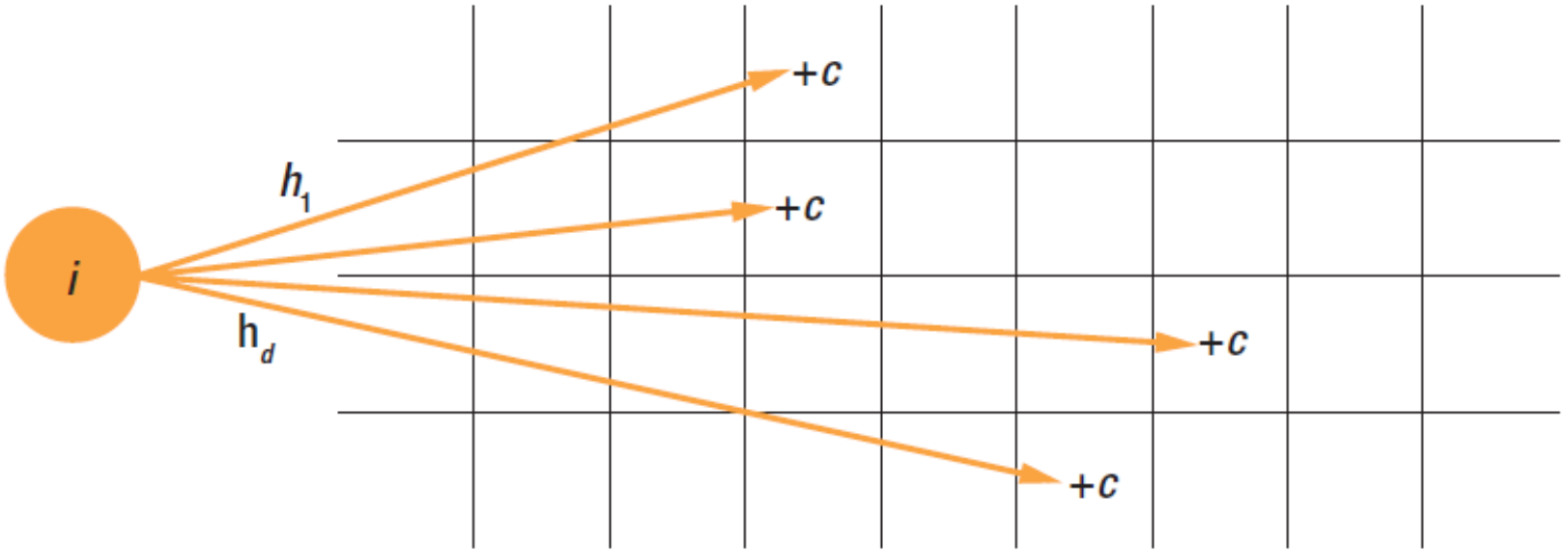
The Count-Min Sketch – Goals

- Process a data stream : one item at a time
- Count the frequency of the different items
- Query for the frequency of a particular item
- Return an estimate of that frequency
  - Within a certain distance of the true count
  - With a certain probability

The Count-Min Sketch

- 2D array : w columns and d rows
  - w : width
  - d : depth
- w and d are fixed and determine
  - The time and space needs
  - The error probability
- A separate hash function for each row
  - d pairwise independent hash functions

The Count-Min Sketch



[Cormode & Muthukrishnan]

The Count-Min Sketch

Algorithm 5: CountMin(w, d)

```
C[1,1]...C[d,w] ← 0;
for j ← 1 to d do
    Initialize g_j;
foreach i do
    n ← n + 1;
    for j ← 1 to d do
        C[j, h_j(i)] ← C[j, h_j(i)] + 1;
```

[Cormode & Hadjieleftheriou]

The Count-Min Sketch

- The hash functions do not need to be particularly strong / complex

- For items represented by integers :

$$h_k(i) = (a_k \times i + b_k) \bmod p \bmod w$$

- p is a prime number larger than the maximum *i* value
  - E.g., $2^{31} - 1$ or $2^{61} - 1$
- a_k, b_k are randomly chosen from $[1, p - 1]$

The Count-Min Sketch

```
CountMinInit(w,d,p)
    C[1,1] … C[d,w] = 0;
    for j=1 to d do
        Pick a_j,b_j from [1 … p-1];
    N = 0;
```

The Count-Min Sketch

```
CountMinUpdate(i)
     N++;
     for j=1 to d do
      h_j(i)=(a_j*i + b_j) mod p mod w;
      C[j,h_j(i)]++;
```

The Count-Min Sketch

```
CountMinEstimate(i)
    est = ∞;
    for j=1 to d do
     h_j(i)=(a_j*i + b_j) mod p mod w;
     est = min(est, C[j,h_j(i)]);
    return est;
```

Count-Min Sketch – Toy Example

- Download the file count_min_sketch.py
- Identify the available operations
- Create a CM Sketch, register and query the count of several items
- Simulate a stream of letters and experiment with sketches of different sizes
  - Check how many letters have a wrong count estimate

The Count-Min Sketch

- How to choose w and d ?
- Goal : the error in answering a query is within a factor (ε x N) with a probability (1 – δ)

$$P\left(\hat{f}_i \leq f_i + \varepsilon \times N\right) \leq 1 - \delta$$

- Number of hash functions

$$d = \left\lceil \log {}^{1}/_{\delta} \right\rceil$$

- Number of columns

$$w = \left\lceil {}^{2}/_{\varepsilon} \right\rceil$$

The Count-Min Sketch

- Required memory space is

$$\mathcal{O}\left(1/\varepsilon \times \log 1/\delta\right)$$

- Time per update is

$$\mathcal{O}\left(\log 1/\delta\right)$$

The Count-Min Sketch – Accuracy

- Each hash function spreads items uniformly over the corresponding row
- For a given i, the expected error weight due to collisions is N/w
  - It is unlikely that it will be much larger !!
- Markov inequality :
  - The probability of seeing more than twice the expected amount is at most 1/2

The Count-Min Sketch – Accuracy

- Similar in each row with different hash functions
  - Different mappings of items to counters
  - Different collections of items collide with *i*
- For each row :
  - There is (at most) a 50% chance of getting an error of more than $2N/w$
  - And (at least) a 50% chance of having an error of less than $2N/w$

The Count-Min Sketch – Accuracy

- We take the minimum counter value, over all rows, for a given column
- Final result will have an error of more than $2N/w$ only if all d rows give a "large" error !!
- Which happens with a probability of at most $(1/2)^d$!!

The Count-Min Sketch – Example

- How to set a sketch's parameters ?
- We want an error of at most 0.1% (of the sum of all frequencies)

$$2/w = 1/1000 \rightarrow w = 2000$$

- With 99.9% certainty

$$(1/2)^d = 0.001$$

$$d = \log_{1/2} 0.001 = \log 0.001 / \log 0.5 \leq 10$$

- 32-bit counters : $w \times d \times 4$ = 80 Kbytes

The Count-Min Sketch – Recap

- The number of required counters is $w \times d$
  - Usually a few thousands !
- The number of counters is independent of m
  - Throw out almost all of your data and still keep approximate counts !! – lossy compression is OK !
- 1-sided error guarantee
  - The estimate is not less than the real count !

The Count-Min Sketch – Recap

- Increase accuracy ?
  - Increase the number of columns : w
  - For each row, there will be less collisions
  - And less over-estimates !!
- Decrease the probability of bad estimates ?
  - Increase the number of rows : d
  - Take the minimum value over more independent estimates
  - BUT a few rows are usually enough !

FINDING THE HEAVY-HITTERS

Heavy-Hitters ?

- Goal: set of heavy-hitters
- Every set item occurs at least $m / k - \varepsilon \times m$
- Every item that occurs at least m / k times is in the set – OK
- How to ?

Heavy-Hitters ?

- **Case 1:** *m* is known in advance !
- Set $\varepsilon = 1 / 2k$ and initialize the CM Sketch
- Register the items
- AND remember an item once its estimated frequency is at least m / k

Heavy-Hitters ?

- **Case 2:** *m* is not known in advance !
- Use a MIN-Heap to keep the HH-candidates !
  - According to their number of occurrences
- Set $\varepsilon = 1 / 2k$ and initialize the CM Sketch
- Create an empty MIN-Heap
- Initialize the counter of registered items
  - $n = 0$

Heavy-Hitters ?

- Increase the counter *n*
- Register the next item : *x*
- Query the CM Sketch for its estimated count
- If *Count(x)* ≥ *n / k*, add *x* to the MIN-Heap or update its value, if already in the MIN-Heap
- Whenever the root element is less than *n / k, delete* it from the MIN-Heap

- At the end, output the MIN-Heap elements

THE COUNT SKETCH

The Count Sketch

- An alternative: The Count Sketch
- Similarly, it provides an estimate for the value of any individual frequency
- Main differences:
  - Estimation procedure
  - Nature of the provided accuracy guarantee

The Count Sketch

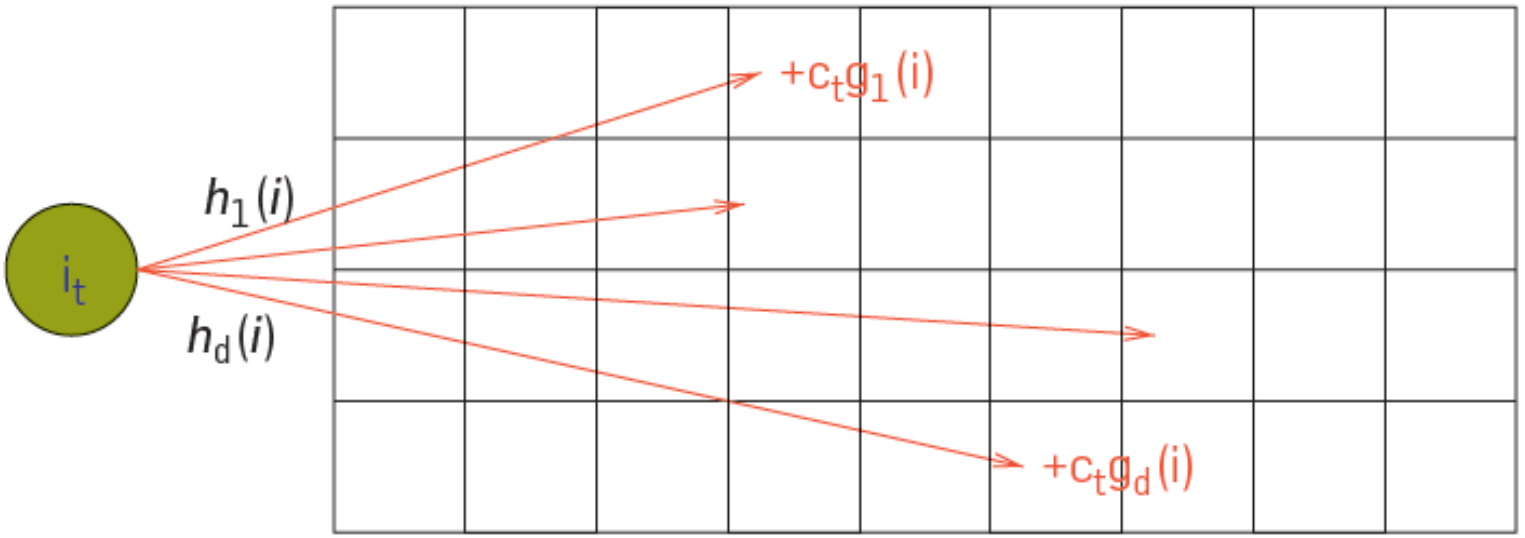
- M. Charikar & K. Chen & M. Farach-Colton
  - 2004 – Journal paper
  - Most of the work was done while the authors were at Google Inc.

- Finding frequent items in data streams
  - Theoretical Computer Science, 2004
  - http://www.sciencedirect.com/science/article/pii/S0304397503004006

The Count Sketch

- 2D array : w columns and d rows
- Two hash functions for each row
  - $h_j(i)$ which maps input items onto $[w]$
  - $g_j(i)$ which maps input items onto $\{-1, +1\}$
- Each input item i causes $g_j(i)$ to be added on to entry $C[j, h_j(i)]$ in row j, for $1 \leq j \leq d$

The Count Sketch



[Cormode & Hadjieleftheriou]

The Count Sketch

Algorithm 4: CountSketch(w, d)

$C[1,1] \ldots C[d,w] \leftarrow 0$;
for $j \leftarrow 1$ **to** d **do**
  Initialize g_j, h_j;
foreach i **do**
  $n \leftarrow n + 1$;
  for $j \leftarrow 1$ **to** d **do**
    $C[j, h_j(i)] \leftarrow C[j, h_j(i), j] + g_j(i)$;

[Cormode & Hadjieleftheriou]

The Count Sketch

- For any row j, the value $g_j(i) \times C[\, j, h_j\,(i)]$ is an unbiased estimator for f_i
- The count estimate is the median of the estimates over the d rows

The Count Sketch

- Setting

$$d = \log {}^{4}/_{\delta} \text{ and } w = \mathcal{O}\left({}^{1}/_{\varepsilon^2}\right)$$

- Ensures that $\widehat{f_i}$ has error at most

$$\varepsilon \times {F_2}^{1/2} \leq \varepsilon \times N$$

- With probability of at least $(1 - \delta)$

- The hash functions must be chosen randomly from "fourwise independent" family

The Count Sketch

- Required memory space is

$$\mathcal{O}\left({}^{1}/_{\varepsilon^2} \times \log {}^{1}/_{\delta}\right)$$

- Time per update is

$$\mathcal{O}\left(\log {}^{1}/_{\delta}\right)$$

Count Sketch vs Count-Min Sketch

- The Count-Min Sketch is a particular case of the Count Sketch, for which $g_j(i) = +1$
- See the paper by Cormode & Hadjieleftheriou for a performance comparison

SOME RECENT PAPERS

2016 – Augmented Sketch

Augmented Sketch: Faster and More Accurate Stream Processing

Pratanu Roy* Systems Group Computer

Arijit Khan† School of Computer

Gustavo Alonso Systems Group Computer

Approximated algorithms are often used to estimate the frequency of items on high volume, fast data streams. The most common ones are variations of Count-Min sketch, which use sub-linear space for the count, but can produce errors in the counts of the most frequent items and can misclassify low-frequency items. In this paper, we improve the accuracy of sketch-based algorithms by increasing the frequency estimation accuracy of the most frequent items and reducing the possible misclassification of low-frequency items, while also improving the overall throughput.

Our solution, called Augmented Sketch (ASketch), is based on a pre-filtering stage that dynamically identifies and aggregates the most frequent items. Items overflowing the pre-filtering stage are

Aug 2018 – Comparative Analysis

2018 Sixth International Conference on Advanced Cloud and Big Data

Comparative Analysis of Different Sketch Methods in Practical Use

Yuan Zhang*, Haiting Zhu*, Nan Bao* and Lu Zhang†

***Abstract*—Network measurement is widely used in many aspects of network management and security area. Accurate estimation with limited resource is the basic requirement for good measurement methods. Our work aims to compare the accuracy between different sketch methods under the real network environment traffic.**

Sketch is a compact data structure used to summarize data stream. At present, there are some based-sketch measurement methods such as Count-Min Sketch, Deltoid, Reversible Sketch and UnivMon. We first introduce the working principles of these sketch methods and their theoretically performance. Then we implement the Count-Min Sketch and Deltoid in heavy hitter detection under real network traffic. Based on the results, comparisons in accuracy and resource consumption were done

Aug 2018 – CMS Optimal Estimation

Count-Min: Optimal Estimation and Tight Error Bounds using Empirical Error Distributions

Daniel Ting
Tableau Software
Seattle, WA

The Count-Min sketch is an important and well-studied data summarization method. It can estimate the count of any item in a stream using a small, fixed size data sketch. However, the accuracy of the Count-Min sketch depends on characteristics of the underlying data. This has led to a number of count estimation procedures which work well in one scenario but perform poorly in others. A practitioner is faced with two basic, unanswered questions. Given an estimate, what is its error? Which estimation procedure should be chosen when the data is unknown?

We provide answers to these questions. We derive new count

Aug 2018 – Elastic Sketch

Elastic Sketch: Adaptive and Fast Network-wide Measurements

Tong Yang
Peking University

Jie Jiang
Peking University

Peng Liu
Peking University

ABSTRACT

When network is undergoing problems such as congestion, scan attack, DDoS attack, *etc.*, measurements are much more important than usual. In this case, traffic characteristics including available bandwidth, packet rate, and flow size distribution vary drastically, significantly degrading the performance of measurements. To address this issue, we propose the Elastic sketch. It is adaptive to currently traffic characteristics. Besides, it is generic to measurement tasks and platforms. We implement the Elastic sketch on six platforms: P4, FPGA, GPU, CPU, multi-core CPU, and OVS, to process six typical measurement tasks. Experimental results and the-

Oct 2018 – CountMax Sketch

CountMax: A Lightweight and Cooperative Sketch Measurement for Software-Defined Networks

Xiwen Yu, Hongli Xu, *Member, IEEE*, Da Yao, Haibo Wang, and Liusheng Huang, *Member, IEEE*

***Abstract*— In a software-defined network (SDN), statistics information is of vital importance for different applications, such as traffic engineering, flow rerouting, and attack detection. Since some resources, e.g., ternary content addressable memory, SRAM, and computing capacity, are often limited on SDN switches, traffic measurements based on flow tables or sampling become infeasible. In fact, sketches provide a promising building block for filling this void by monitoring every packet with fixed-size memory. Although many efficient sketches have been designed, our analysis shows that existing sketch-based measurement solutions may suffer from severe computing overhead on switches especially under high traffic load that significantly interferes with switch's basic functions, such as flow rule setup and modification. In this paper, we present CountMax,**

June 2021 – Clock-Sketch

Out of Many We are One: Measuring Item Batch with Clock-Sketch

Peiqing Chen*
Peking Univ
Beijing

Dong Chen*

Lingxiao Zheng*
University

ABSTRACT

[1] Item batch denotes a consecutive sequence of identical items that are close in time in a data stream. It is a useful data stream pattern in cache, burst detection, APT detection ,*etc.* Basic item batch measurement tasks include membership, cardinality, time span and size. Currently, there is no algorithm tailored for item batch measurement. The greatest challenge lies in accurately estimating the time gap between two consecutive identical items. In this paper, we propose Clock-sketch, a framework that introduces the well-known CLOCK algorithm into item batch measurement. The methodology of Clock-sketch is to clean outdated information as much as possible, while guaranteeing that the information of all items visited within the time window $\mathcal{T}$ is preserved. We conduct experiments on three real-world datasets that feature in item batch

Dec 2021 – Current Trends

Current Trends in Data Summaries

Graham Cormode*
Meta AI

ABSTRACT

The research area of data summarization seeks to find small data structures that can be updated flexibly, and answer certain queries on the input accurately. Summaries are widely used across the area of data management, and are studied from both theoretical and practical perspectives. They are the subject of ongoing research to improve their performance and broaden their applicability. In this column, recent developments in data summarization are surveyed, with the intent of inspiring further advances.

Jan 2022 – Survey

ELSEVIER

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Applications of sketches in network traffic measurement: A survey

Hui Han [a], Zheng Yan [a,b,*], Xuyang Jing [c], Witold Pedrycz [d]

ABSTRACT

Accurate and timely network traffic measurement is essential for network status monitoring, network fault analysis, network intrusion detection, and network security management. With the rapid development of the network, massive network traffic brings severe challenges to network traffic measurement. However, existing measurement methods suffer from many limitations for effectively recording and accurately analyzing big-volume traffic. Recently, sketches, a family of probabilistic data structures that employ hashing technology for summarizing traffic data, have been widely used to solve these problems. However, current literature still lacks a thorough review on sketch-based traffic measurement methods to offer a comprehensive insight on how to apply sketches for fulfilling various traffic measurement tasks. In this paper, we provide a detailed and comprehensive review on the applications of sketches in network traffic measurement. To this end, we classify

2022 - LUSketch

LUSketch: A Fast and Precise Sketch for top-k Finding in Data Streams

Jie Lu
PLA Information Engineering University
Zheng Zhou, China

Hongchang Chen
National Digital Switching System Engineering and Technological Research and Development Center

Zhen Zhang
Network Communication and Security Purple Mountain Laboratory
Nanjing, China

***Abstract*—Finding top-*k* flows in data streams is a fundamental task in network management. As the line rates continue to increase in a network, it becomes increasingly challenging to find the top-*k* flows precisely and quickly in real time. Existing algorithms that can achieve high precision suffer from a slow speed. In this paper, we propose a novel sketch, LUSketch, which is much faster than existing algorithms. LUSketch adopts a new strategy called limited-and-imperative-update to significantly improve the insertion speed. The key idea is to significantly reduce the number of update operations by establishing a connection between the sketch and heap part when tracking the top-*k* flows. Our**

Aug 2022 – CMS-CU Analysis

ELSEVIER

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Analyzing Count Min Sketch with Conservative Updates

Younes Ben Mazziane *, Sara Alouf, Giovanni Neglia

Université Côte d'Azur, Inria, Sophia Antipolis, France

Count-Min Sketch with Conservative Updates (CMS-CU) is a popular algorithm to approximately count items' appearances in a data stream. Despite CMS-CU's widespread adoption, the theoretical analysis of its performance is still wanting because of its inherent difficulty. In this paper, we propose a novel approach to study CMS-CU and derive new upper bounds on both the expected value and the CCDF of the estimation error under an i.i.d. request process. Our formulas can be successfully employed to derive improved estimates for the precision of heavy-hitter detection methods and improved configuration rules for CMS-CU. The bounds are evaluated both on synthetic and real traces.

REFERENCES

References

- G. Cormode & M. Hadjieleftheriou, Finding the frequent items in streams of data, *Commun. ACM*, Vol. 52, N. 10, 2009
- G. Cormode & S. Muthukrishnan, Approximating data with the Count-Min Sketch, *IEEE Software*, Vol. 29, N. 1, 2012
- G. Cormode et al., Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches, Foundations and Trends in Databases, Vol. 4, 2012