COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2023.

Lecture 11

Logistics

- Problem Set 2 is due on Monday at 11:59pm.
- Midterm is in class Tuesday, 10/24. Thursday 10/19 will be devoted to midterm review.
- The grading on this week's quiz regarding the extra credit question had a bug. We will fix manually in the next few days.

Summary

Last Class:

- Locality sensitive hashing to solve the similarity search problem efficiently.
- MinHash as a locality sensitive hash function for Jaccard similarity.
- Brief look at SimHash as a locality sensitive hash function for cosine (dot product) similarity.

This Class:

- Introduce the frequent elements problem and its applications.
- Solution via the Count-Min sketch randomized data structure.

The Frequent Items Problems

k-Frequent Items (Heavy-Hitters) Problem: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_1
5	12	3	3	4	5	5	10	3	5

- What is the maximum number of items that can be returned? a) n b) k c) n/k d) $\log n$
- Trivial with $O(n)$ space – store the count for each item and return the one that appears $\geq n/k$ times.
- Can we do it with less space? I.e., without storing all n items?

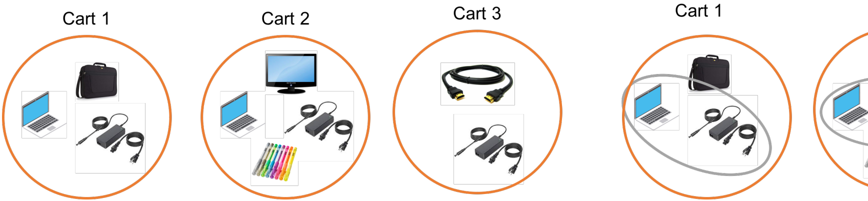
Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
- 'Iceberg queries' for all items in a database with frequency above some threshold.

Generally want very fast detection, without having to scan through database/logs. I.e., want to maintain a running list of frequent items that appear in a stream.

Frequent Itemset Mining

Association rule learning: A very common task in data mining is to identify common associations between different events.



- Identified via frequent itemset counting. Find all sets of t items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

Approximate Frequent Elements

Issue: No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency n/k (should be output) and $n/k - 1$ (should not be output).

x_1	x_2	x_3	x_4	x_5	x_6	...	$x_{n-n/k+1}$	...	x_n
3	12	9	27	4	101		3		3

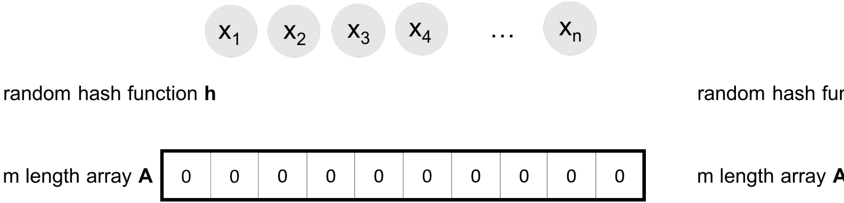
n/k-1 occurrences

(ϵ, k)-Frequent Items Problem: Consider a stream of n items $x_1, \ldots, x_n$. Return a set F of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1-\epsilon) \cdot \frac{n}{k}$ times.

- An example of relaxing to a 'promise problem': for items with frequencies in $[(1-\epsilon) \cdot \frac{n}{k}, \frac{n}{k}]$ no output guarantee.

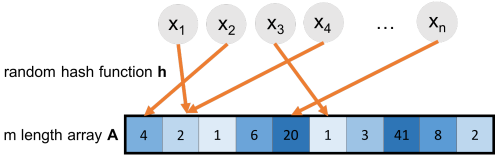
Frequent Elements with Count-Min Sketch

Today: Count-min sketch – a random hashing based method closely related to bloom filters.



Will use $A[\mathbf{h}(x)]$ to estimate $f(x)$, the frequency of x in the stream. I.e., $|\{x_i : x_i = x\}|$.

Count-Min Sketch Accuracy



Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$. Why?

- $A[\mathbf{h}(x)]$ counts the number of occurrences of any y with $\mathbf{h}(y) = \mathbf{h}(x)$, including x itself.
- $A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

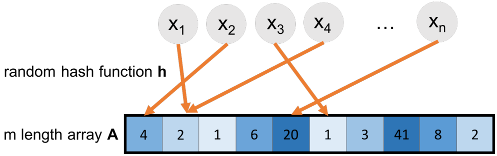
What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov's inequality: $\Pr\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2}$.

What property of **h** is required to show this bound? a) fully random b) pairwise independent c) 2-universal d) locality sensitive

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy



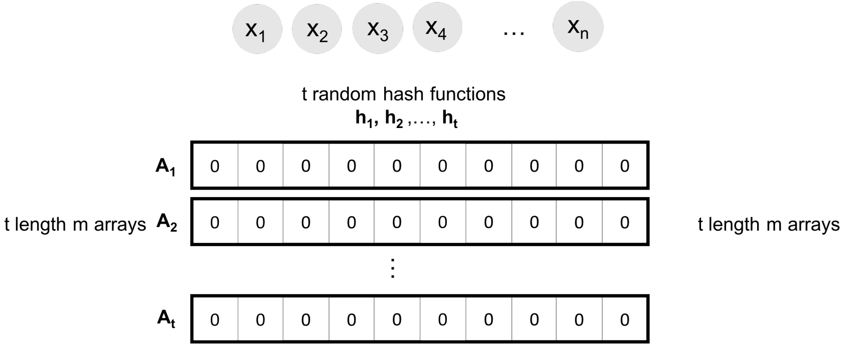
Claim: For any x, with probability at least 1/2,

$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$. How can we improve the success probability? Repetition.

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

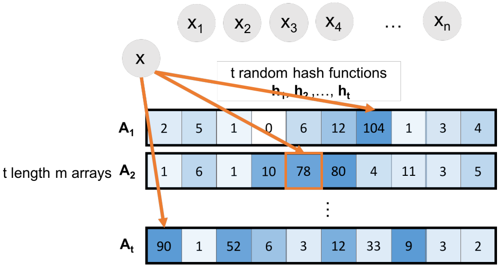
Count-Min Sketch Repetition



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!

Count-Min Sketch Analysis



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.
- To get a good estimate with probability $\geq 1 - \delta$, set $t = \log(1/\delta)$.

Count-Min Sketch

Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O\left(\log(1/\delta) \cdot k/\epsilon\right)$ space.

- Accurate enough to solve the (ϵ, k)-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.
- How should we set δ if we want a good estimate for all items at once, with 99% probability?

Identifying Frequent Elements

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

One approach:

- When a new item comes in at step i, check if its estimated frequency is $\geq i/k$ and store it if so.
- At step i remove any stored items whose estimated frequency drops below i/k.
- Store at most $O(k)$ items at once and have all items with frequency $\geq n/k$ stored at the end of the stream.