Bloom Filters, Count Sketches and Adaptive Sketches

Rice University

Anshumali Shrivastava

anshumali@rice.edu

29th August 2016

Basics: Universal Hashing

Basic tool for shuffling and sampling from any set of objects $O = \{1, 2, ..., n\}$.

- $h : O \rightarrow \{1, 2, ..., m\}$
- $Pr(h(x) = h(y)) \leq \frac{1}{m}$ iff $x \neq y$.

Some implementations

- Pick a random number a and b, a large enough prime, return $h(x) = ax + b \text{ mod } p \text{ mod } m$
- **Fastest Trick:** Choose $m = 2^M$ to be power of 2, choose a random odd integer return $h(x) = ax >> (32 - M)$

Problems:

- Given a set O, randomly assign it to m bins.
- Randomly sample $1/m$ fraction of the data.
- **Activity:** Suppose $m >> n$ How to sample one element randomly from O

Bloom Filters Set Up

A common Task: How to know whether some event occurred (before) or not without storing the event information? The number of possible events are huge. The following list is from Wikipedia

- Akamai web servers use Bloom filters to prevent "one-hit-wonders" from being stored in its disk caches. One-hit-wonders are web objects requested by users just once.
- Google BigTable, Apache HBase and Apache Cassandra, and Postgresql use Bloom filters to reduce the disk lookups for non-existent rows or columns. Avoiding costly disk lookups considerably increases the performance of a database query operation.
- The Google Chrome web browser used to use a Bloom filter to identify malicious URLs.
- The Squid Web Proxy Cache uses Bloom filters for cache digests
- Bitcoin uses Bloom filters to speed up wallet synchronization.
- many more.

The Bloom Filter Algorithm and Analysis

A Dynamic Data Structure of m bit arrays B

- Pick k universal hash function $h_i : O \rightarrow \{1,\ 2, ...,\ m\}$ $i \in \{1,\ 2, ...,\ k\}$.
- **Insert** o_j**:** Set all the bits $B(h_i(o_j)) = 1$. $\forall\ i \in \{1,\ 2, ...,\ k\}$
- **Query** o_j**:** If $B(h_i(o_j)) = 1\ \forall\ i \in \{1,\ 2, ...,\ k\}$ RETURN True ELSE false

Properties

- If an item is present, the algorithm is always correct. No false negative.
- If an item is not present, the algorithm may return true with small probability.
- Cannot delete items easily.

Analysis On-Board

Generalized Bloom Filters: Count-Min Sketch

On a network, a lot of events keep happening. Cannot afford to store event information.

- **Bloom Filters:** Keep track of whether an given event has already happened or not.
- **Count Min Sketches (or Count Sketches):** Keep track of the frequency of the frequent events (heavy hitters).
    - Instead of bits keep Counters
    - Usually, to avoid collisions among different hashes, they are hashed into different arrays. (Hence we get Matrix)

The Classical (Non-Adaptive) Approximate Counting:

Setting:

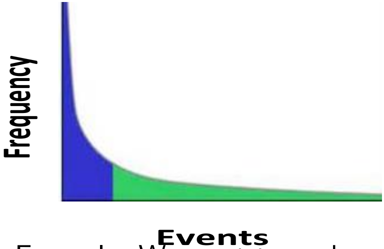
- We are given a huge number of items (co-variate) $i \in I$ to track over time $t \in \{1, 2, ..., T\}$. T can be large as well.
- We only see increments (i, t, v), the increment v to item i at time t.

Goal: In limited space (hopefully $O(\log |I| \times T)$), we want to

- **Point Queries:** Estimate the counts (increments) of item i at time t.
- **Range Queries:** Estimate the counts (increments) of item i during the given range $[t_1, t_2]$.

Classical Sketching: Count-Sketch, Count-Min Sketch (CMS), Lossy Counting, etc.

Idea: Power Law Everywhere in Practice



- Example: We want to cache answers to frequent queries on a server. All queries are just too much to keep track of.
- How to identify very frequent queries? (Note, we cannot count everything.)
- We dont even know which ones are frequent, we only see some queries within a given time set.

Counting Heavy Hitters on Data Streams

Real Problem: How to identify significant event (frequent) without having to count all of them. (sub-linear)

Counting Heavy Hitters on Data Streams

Real Problem: How to identify significant event (frequent) without having to count all of them. (sub-linear)

Classical Formalism (Turnstile Model)

- Assume we have a very long vector v (Dim D), we cannot materialize.
- We only see increments to its coordinates. E.g. co-ordinate i is incremented by 10 at time t.
- **Goal:** Find s heaviest coordinate, using space $k << D$

Counting Heavy Hitters on Data Streams

Real Problem: How to identify significant event (frequent) without having to count all of them. (sub-linear)

Classical Formalism (Turnstile Model)

- Assume we have a very long vector v (Dim D), we cannot materialize.
- We only see increments to its coordinates. E.g. co-ordinate i is incremented by 10 at time t.
- **Goal:** Find s heaviest coordinate, using space $k << D$

Seems Hopeless !

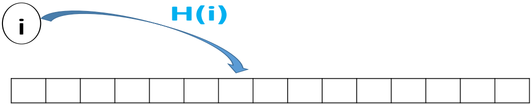
Uncertainty is the Refuge of Hope.

—Henri Frederic Amiel (1821-81)

Basic Idea behind Sketching.

Randomly assign items to a small number of counters.

- It works! AMS 85, Moody 89, Charikar 99, MuthuKrishnana 02, etc.
- If no collisions, counts exact.



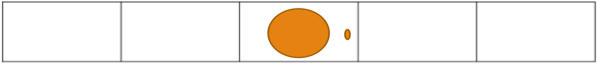
Use Random Hash Function

Handling Time:

- Treat each pair (i, t) (item, time) as different item.
- Hash pairs (i, t), instead of just items.
- Time only increases the number of items to $|I| \times T$.

What happens during Collision ?

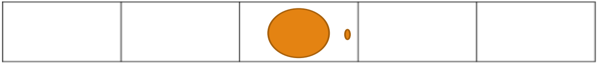
The Good



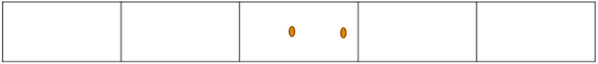
We typically care about heavy hitters.

What happens during Collision ?

The Good



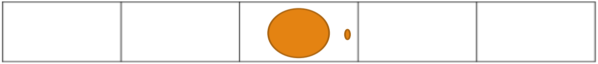
The Irrelevant



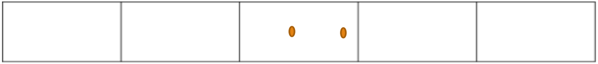
We typically care about heavy hitters.

What happens during Collision ?

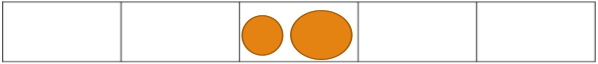
The Good



The Irrelevant



The Unlucky

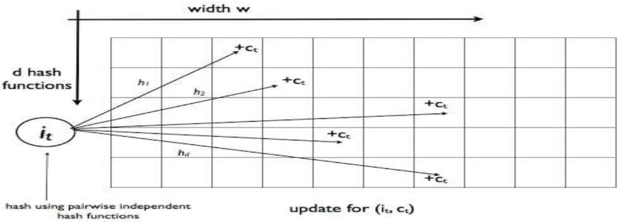


We typically care about heavy hitters.

Maximizing Luck : Count-Min Sketch (CMS)

Idea:

- We always overestimate, if unlucky, by a lot.
- Repeat independently d times and take minimum of all overestimates.
- Unless unlucky all d times, it will work. ($d = \log \frac{1}{\delta}$, $w = \frac{1}{\epsilon}$)



Theoretical Guarantee

- $c \leq \hat{c} \leq c + \epsilon \mathcal{M}^T$ with probability $1 - \delta$, where $\mathcal{M}^T$ is sum of all counts in the stream.
- Space $O(\log |I| \times T)$

New Requirement: Time Adaptability

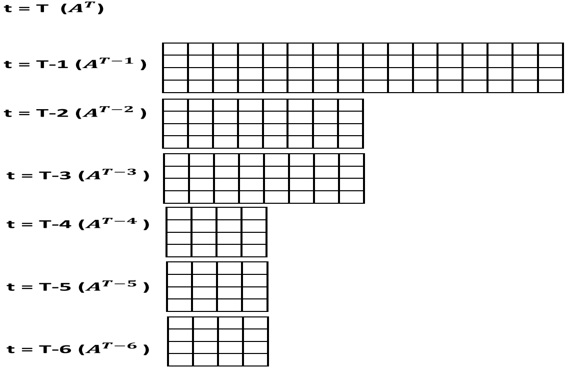
In Practice:

- Recent trends are more important.
- A burst in the number of clicks in the past few minutes more informative than similar burst last month.

Expectation: Time Adaptive Counting.

- Classical sketches do not take temporal effect into consideration.
- **Smart Tradeoff:** Given the same space, trade errors of recent counts with that of older ones.
- Like our memory, forget slowly.

Existing Solution: Hokusai[1]

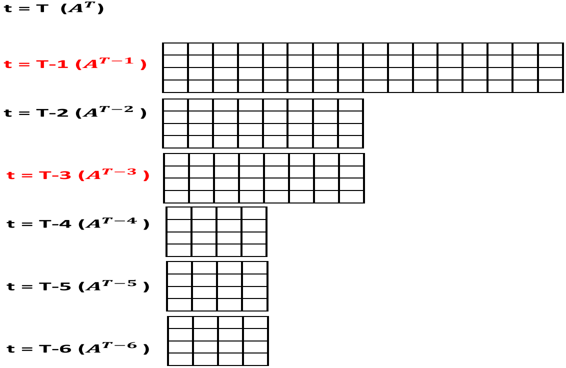


Idea: Disproportionate allocation over time.

- Accuracy of CMS dependent on memory allocated.
- More space for recent sketches and less for older.
- Keep a CMS sketch for every time. Shrink sketch size on fly.
  Clever Idea: Exploit Rollover.

[1] Matusevych, Smola and Ahmad 2012

Existing Solution: Hokusai[1]

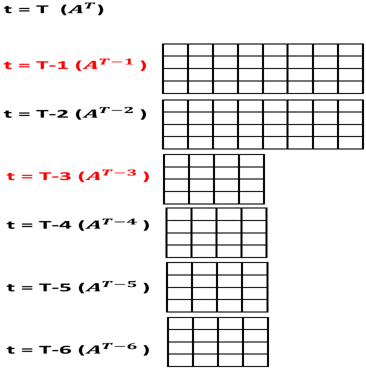


Idea: Disproportionate allocation over time.

- Accuracy of CMS dependent on memory allocated.
- More space for recent sketches and less for older.
- Keep a CMS sketch for every time. Shrink sketch size on fly.
  Clever Idea: Exploit Rollover.

[1] Matusevych, Smola and Ahmad 2012

Existing Solution: Hokusai[1]

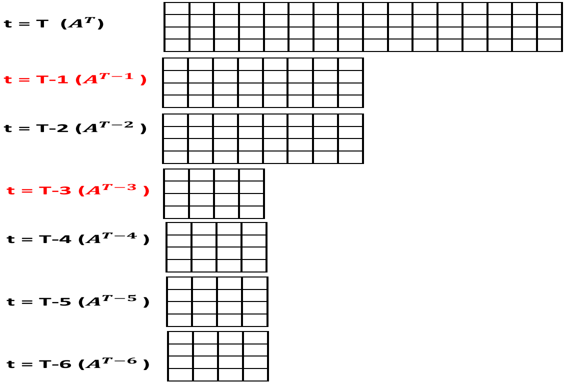


Idea: Disproportionate allocation over time.

- Accuracy of CMS dependent on memory allocated.
- More space for recent sketches and less for older.
- Keep a CMS sketch for every time. Shrink sketch size on fly.
  Clever Idea: Exploit Rollover.

[1] Matusevych, Smola and Ahmad 2012

Existing Solution: Hokusai[1]



Idea: Disproportionate allocation over time.

- Accuracy of CMS dependent on memory allocated.
- More space for recent sketches and less for older.
- Keep a CMS sketch for every time. Shrink sketch size on fly.
  Clever Idea: Exploit Rollover.

[1] Matusevych, Smola and Ahmad 2012

Problems with Hokusai

Issues:

- Discontinuity. If time t is empty, we still have to shrink sketch size for older times.
- O(T) memory. One for each t.
- Shrinking overhead. Shrink $\log t$ sketches for every transition from t to $t + 1$.
- No flexibility.

Detour: Dolby Noise Reduction (1960s)

High Level View

- In digital recording, the music signal compete with tape hiss (background noise).
- if Signal to Noise (SNR) ratio is high, the recording is noise free.
- While recording the frequencies in the music is artificially inflated (Pre-Emphasis).
- During playback a reverse transformation is applied which cancels pre-emphasis. (De-Emphasis)
- Overall effect of noise is minimized.

Proposal: (Adaptive)Ada-Sketches

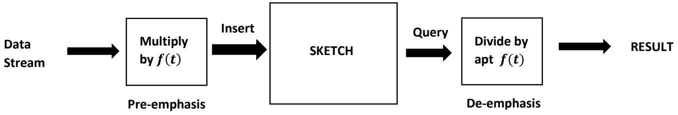
Analogy with Dolby Noise Reduction:

- Sketches preserves heavier counts more accurately.
- Artificially inflate recent counts (Pre-emphasis).
- Inflated counts will be preserved with less error.
- Deflate by exact same amount during estimation. (De-emphasis)

Proposal: (Adaptive)Ada-Sketches

Analogy with Dolby Noise Reduction:

- Sketches preserves heavier counts more accurately.
- Artificially inflate recent counts (Pre-emphasis).
- Inflated counts will be preserved with less error.
- Deflate by exact same amount during estimation. (De-emphasis)



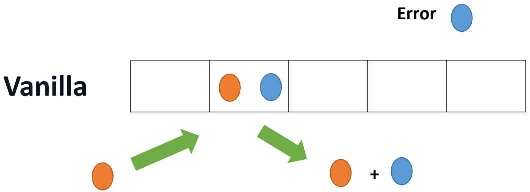
Proposal

- Let $f(t)$ be any (pre-defined) monotonically increasing sequence. ($f(t)$ can be chosen wisely)
- Multiply the count of (i, t) with $f(t)$ and then add to the sketch.
- While querying (i,t), get the estimate and divide by $f(t)$

Why it works ?

Observation

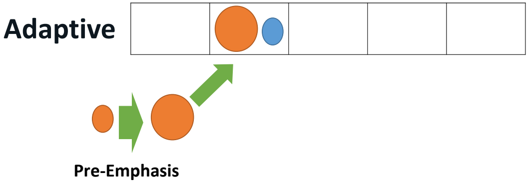
- If no collision then exact.
- During collision, errors or recent counts decrease due to greater de-emphasis.



Why it works ?

Observation

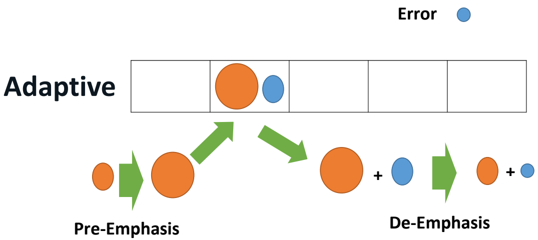
- If no collision then exact.
- During collision, errors or recent counts decrease due to greater de-emphasis.



Why it works ?

Observation

- If no collision then exact.
- During collision, errors or recent counts decrease due to greater de-emphasis.



Advantages

- No Discontinuity. If time t is empty, no addition, no extra collisions, no extra errors.
- $O(\log |I| \times T)$ memory just like CMS.
- No shrinking overhead. Minimum modification to CMS. (Strict Generalization)

Provable Time Adaptive Guarantees

Theorem

For $w = \lceil \frac{e}{\epsilon} \rceil$ *and* $d = \log \frac{1}{\delta}$*, given any* (i, t) *we have*

$$c_i^t \leq \widehat{c_i^t} \leq c_i^t + \epsilon \beta^t \sqrt{\mathcal{M}_2^T}$$

with probability $1 - \delta$*. Here* $\beta^t = \frac{\sqrt{\sum_{t'=0}^{T} (f(t'))^2}}{f(t)}$ *is the time adaptive factor monotonically decreasing with* t*.*

More..

Works with any Sketching Algorithm

- Adaptive Count Sketches, Adaptive Lossy Counting etc.
- Provable Time Adaptive Guarantees for all of them.

More..

Works with any Sketching Algorithm

- Adaptive Count Sketches, Adaptive Lossy Counting etc.
- Provable Time Adaptive Guarantees for all of them.

Flexibility in Choice of $f(t)$

- Any monotonic $f(t)$ works. Can be tailored
- Upper bound dependent on $\beta^t = \frac{\sqrt{\sum_{t'=0}^{T}(f(t'))^2}}{f(t)}$.
- Fine control over the error distributions.

Experiments: Accuracy for a given Memory

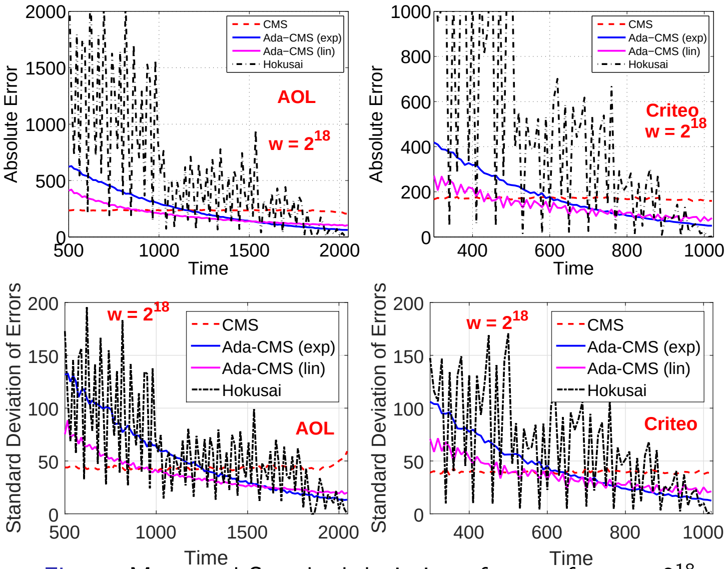


Figure: Mean and Standard deviation of errors for $w = 2^{18}$.

Scalability: Throughput

Table: Time in sec to summarize AOL dataset

	2^{20}	2^{22}	2^{25}	2^{27}	2^{30}
CMS	44.62	44.80	48.40	50.81	52.67
Hoku	68.46	94.07	360.23	1206.71	9244.17
ACMS (lin)	44.57	44.62	49.95	52.21	52.87
ACMS (exp)	68.32	73.96	76.23	82.73	76.82

Table: Time in sec to summarize Criteo Dataset

	2^{20}	2^{22}	2^{25}	2^{27}	2^{30}
CMS	40.79	42.29	45.81	45.92	46.17
Hoku	55.19	90.32	335.04	1134.07	8522.12
ACMS (lin)	39.07	42.00	44.54	45.32	46.24
ACMS (exp)	69.21	69.31	71.23	72.01	72.85