What's Hot, What's Not, What's New and What's Next

Graham Cormode, DIMACS
graham@dimacs.rutgers.edu

Joint work with S. Muthukrishnan

Outline

- What's the problem?
- What's hot and what's not?
- What's new?
- What's next?

Data Stream Phenomenon

- Networks are sources of massive data: just metadata per hour per router is gigabytes
- Too much information to store or transmit
- So process data as it arrives: one pass, small space
- Approximate answers to most questions are OK

Network Stream Problems

Questions on networks are often simple, complexity comes from space and time restrictions.

- How many distinct host addresses?
- Destinations using most bandwidth?
- Address with biggest change in traffic overnight?

Data Stream Algorithms

- Recent interest in "data stream algorithms":
  small space, one pass approximations

- Alon, Matias, Szegedy 96: frequency moments
  Henzinger, Raghavan, Rajagopalan 98 graph streams

- In last few years:
  Counting distinct items, finding frequent items, quantiles, wavelet and Fourier representations, histograms...

The Gap

A big gap between theory and practice: good theory results aren't yet ready for primetime.

Approximate within $1\pm\varepsilon$ with probability $> 1-\delta$. Eg: AMS sketches for F_2 estimation, set $\varepsilon=1\%$, $\delta=1\%$

- Space $O(1/\varepsilon^2 \log 1/\delta)$ is approx 10^6 words = 4Mb
  Network device may have 100k-4Mb space *total*

- Each data item requires pass over whole space
  At network line speeds can afford a few dozen memory accesses, perhaps more with parallelization

Bridging the Gap

- The Count-Min sketch and change detection data structures attempt to bridge the gap
- Simple, small, fast data stream summaries which have application to a large number of problems
- Some subtlety: to beat $1/\varepsilon^2$ lower bounds, must explicitly avoid estimating frequency moments
- Applications to fundamental problems in networks, finding heavy hitters and large changes

Outline

- What's the problem?
- **What's hot and what's not?**
- What's new?
- What's next?

1. Heavy Hitters

- Focus on the Heavy Hitters problem: Find users (IP addresses) consuming more than 1% of bandwidth
- In algorithms, "Frequent Items": Find items and their counts when count more than ϕN
- Heavily studied problem (arrivals only): Charikar, Chen, Farach-Colton 02, Karp,Papadimitriou,Shenker 03, Manku, Motwani 02, Demaine, LopezOrtiz, Munro 02

Stream of Packets

- Packets arrive in a stream. Extract from header:
  Identifier, i: Source or destination IP address
  Count: connections / packets / bytes

- Stream defines a vector a[1..U], initially all 0
  Each packet increases one entry, a[i].
  In networks $U = 2^{32}$ or 2^{64}, too big to store

- Heavy Hitters are those i's where $a[i] > \phi N$
  Maintain N = sum of counts

Heavy Hitters Solution

Naive solution: keep the array a and for every item in the stream, test whether $a[i] > \phi N$, keep heap of items

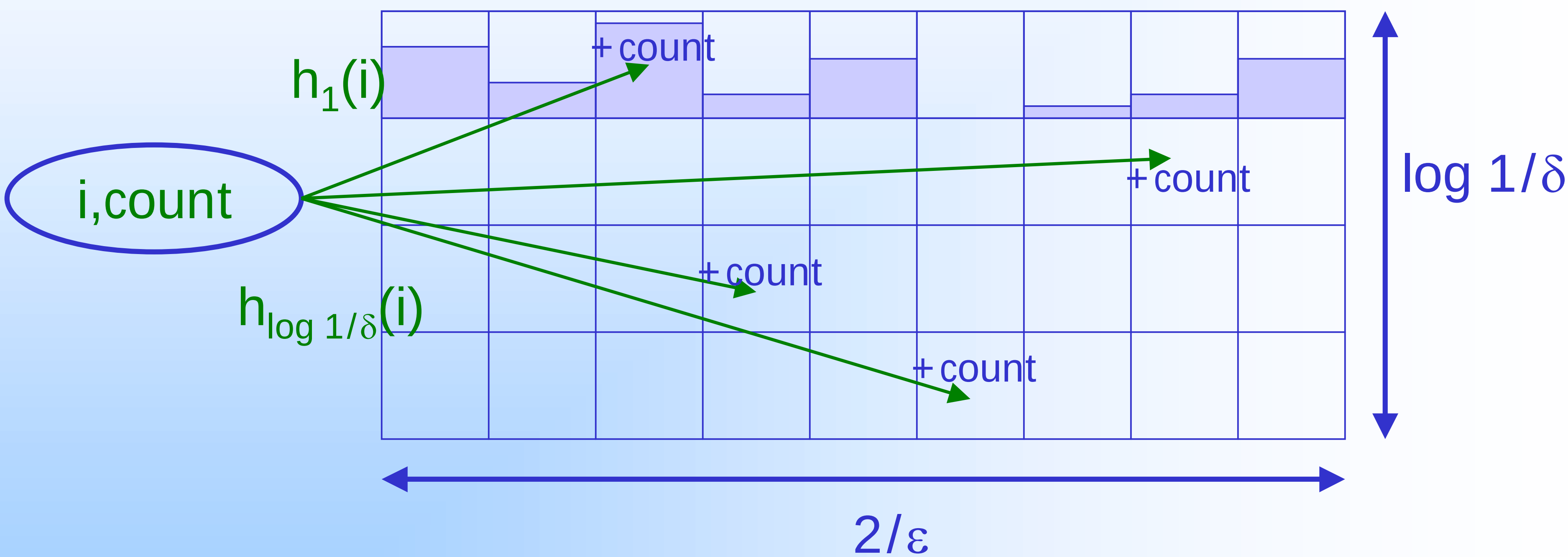
Solution here: replace a[i] with a small data structure which approximates all a[i] upto εN with prob $1-\delta$

Ingredients:

– 2-wise hash fns $h_1..h_{\log 1/\delta}$ $\{1..U\} \rightarrow \{1..2/\varepsilon\}$

– Array of counters $CM[1..2/\varepsilon, 1..\log_2 1/\delta]$

Update Algorithm



CM Sketch

Approximation

Approximate $\hat{a}[i] = \min_j CM[h_j(i),j]$

Analysis: In j'th row, $CM[h_j(i),j] = a[i] + X_{i,j}$

$$X_{i,j} = \Sigma\, a[k] \mid h_j(i) = h_j(k)$$

$$\begin{aligned} E(X_{i,j}) &= \Sigma\, a[k]*\Pr[h_j(i)=h_j(k)] \\ &\leq \Pr[h_j(i)=h_j(k)] * \Sigma\, a[k] \\ &= \varepsilon N/2 \text{ by pairwise independence of } h \end{aligned}$$

Analysis

$\Pr[X_{i,j} \geq \varepsilon N] = \Pr[X_{i,j} \geq 2E(X_{i,j})]$
$\leq 1/2$ by Markov inequality

Hence, $\Pr[\hat{a}[i] \geq a[i] + \varepsilon N] = \Pr[\forall\ j.\ X_{i,j} > \varepsilon N]$
$\leq 1/2^{\log 1/\delta} = \delta$

Final result:
with certainty $a[i] \leq \hat{a}[i]$ and
with probability at least $1-\delta$, $\hat{a}[i] < a[i] + \varepsilon N$

Results

- Every item with count $> \phi N$ is output and with prob $1-\delta$, each item in output has count $> (\phi-\varepsilon)N$
- Space $= 2/\varepsilon \log_2 1/\delta$ counters $+ \log_2 1/\delta$ hash fns
  Time per update $= \log_2 1/\delta$ hashes
  (2-wise hash functions are fast and simple)
- Fast enough and lightweight enough for use in network implementations
- Something novel: allows arbitrary fractional and negative updates to counters, so more flexible

Implementations

Implementations work pretty well, better than theory suggests: 2 or 3 hash functions suffice in practice

Running in AT&T's Gigascope, on live 2.4Gbs streams

- Each query may fire many instantiations of CM sketch, how do they scale?
- Should sketching be done at low level (close to NIC) or at high level (after aggregation)?
- Always allocate space for a sketch, or run exact algorithm until count of distinct IPs is large?

Frequent Items with Deletions

- When items are deleted (eg in a database relation), finding frequent items more difficult.
- Items from the past may become frequent, following a deletion, so need to be able to recover item labels.
- Impose a (binary) tree structure on the universe, nodes correspond to sum of counts of leaves.
- Keep a sketch for each level and search the tree for frequent items with divide and conquer.

Deletions - Fine Details

- Other sketches could be used but CM sketch guarantees to find all hot items, smaller space
- Binary tree costs factor of log U in update time and space, can be improved by using tree of higher branching factor, at cost of search time.
- Meta-question: do deletions really occur in Network data at the packet level?
- Meta-answer: usually no. But negative values occur when you compare streams by subtraction...

Outline

- What's the problem?
- What's hot and what's not?
- **What's new?**
- What's next?

2. Change Detection

- Find items with big change between streams x and y
  Find IP addresses with big change in traffic overnight
- "Change" could be absolute difference in counts, or large ratio, or large variance...
- Absolute difference: find large values in a(x) - a(y)
  Relative difference: find large values a(x)[i]/a(y)[i]
- CM sketch can approximate the differences, but how to find the items without testing everything? Divide and conquer will not work here!

Change Detection

- Use Non-Adaptive Group Testing: (randomized) structure of CM sketch defines groups of items
- Within each group, test for "deltoids": keep more information than just counts.
- Test depends on kind of deltoid being searched for, but same structure of groups used for all.

Group Structure

- Use a 2-wise hash function to divide the universe into $2/\varepsilon$ groups, as in CM sketch
- Repeat $\log 1/\delta$ times to amplify probability
- Keep a test for each group to determine if there is a deltoid within it.
- If there is a deltoid in the group need to identify it, so also keep tests on subsets of each group.

Group Sub-Structure

- Keep 2log U subgroups in each group based on Hamming code
- For each item i in group, include i in subgroup j if j'th bit of i is 1, else include in subgroup j'
- To find deltoids, read results of tests of subgroups: if test j is positive, bit $j = 1$, test j' positive, bit $j=0$
- If j and j' both positive, two deltoids in same group, reject the group (also if j and j' both negative)

Tests

- How to construct a test for the presence of a deltoid?
- Naively, could keep sketch for each group, but space blows up ($1/\varepsilon^2$ or worse)
- For absolute change deltoids, keeping counts of items suffices, proof similar to CM sketch
- For relative change, appropriate counts also suffice, new proof needed.

Relative Change Test

- Keep different information for each stream.
- For stream x, keep $T(x)[j] = \Sigma\, a(x)[i] \mid h(i) = j$
- For stream y, keep $T(y)[j] = \Sigma\, (1/a(y)[i]) \mid h(i) = j$
- Test: if $T(x)[j]*T(y)[j] > \phi\, \Sigma\, (a(x)[i]/a(y)[i])$
- Test has one-sided error, will always say yes if
  $(a(x)[i]/a(y)[i]) > \phi\, \Sigma\, (a(x)[i]/a(y)[i])$

Relative Change Test

- To bound false positives, and ensure true positives are not obscured by noise, need to argue that each test gives good enough estimate of (a(x)[i]/a(y)[i])
- Error variable X_{ij} = T(x)[j]*T(y)[j] - (a(x)[i]/a(y)[i]) and let p = Pr[h(i) = h(j)] = 1/#groups = ε/2

Illegible Equations Slide

$$
\begin{aligned}
E(X_{ij}) &= E(T(x)[j]*T(y)[j] - (a(x)[i]/a(y)[i])) \\
&= (a(x)[i] + a(x)[j] \mid h(j) = h(i))* \\
&\qquad (1/a(y)[i] + 1/a(y)[j] \mid h(j) = h(i)) \\
&\qquad\quad - (a(x)[i]/a(y)[i])
\end{aligned}
$$

$$
\begin{aligned}
&\le a(x)[i]*p*\Sigma\, 1/a(y)[j] + 1/a(y)[i]*p*\Sigma\, a(x)[j] \\
&\qquad + p*(\Sigma_{j\neq i}\, a(x)[j])*(\Sigma_{j\neq i}\, 1/a(y)[j])
\end{aligned}
$$

$$
\le p(\Sigma a(x)[i])*(\Sigma 1/a(y)[i]) = \varepsilon \|a(x)\|_1 \, \|1/a(y)\|_1/2
$$

Consequences

- Expected error is 1/2 of $\varepsilon \|a(x)\|_1 \|1/a(y)\|_1$
- By Markov again, constant probability that there is error at most $\varepsilon \|a(x)\|_1 \|1/a(y)\|_1$ for each test, amplify to probability $1-\delta$ with $\log 1/\delta$ tests
- Can argue that if this condition is met, and $\varepsilon < \phi$, then will find relative change deltoid with probability at least $1-\delta$
- With probability $1-\delta$, every item output has change at least $\phi \Sigma (a(x)[i]/a(y)[i]) - \varepsilon \|a(x)\|_1 \|1/a(y)\|_1$

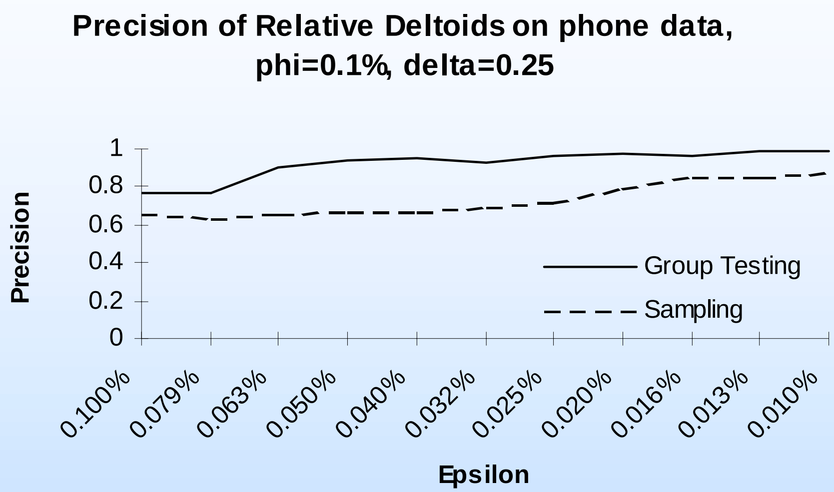
Nuances

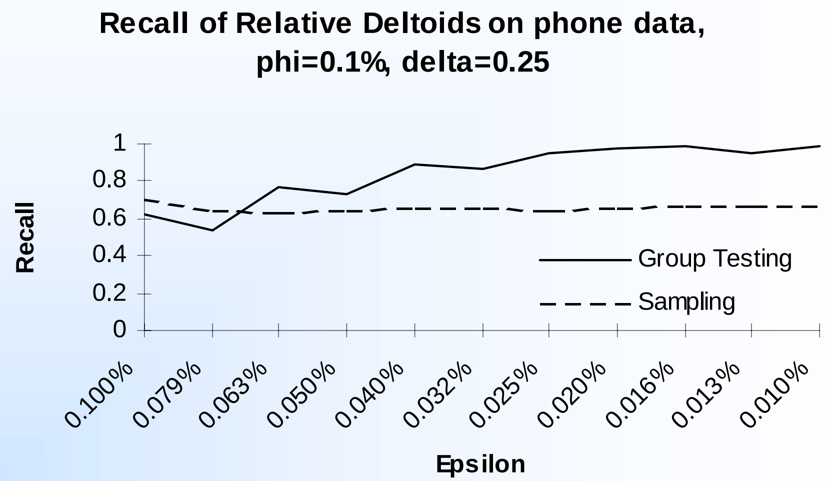
- Error term is $\varepsilon\|a(x)\|_1 \|1/a(y)\|_1$ not Σ (a(x)[i]/a(y)[i]) — but the latter is not possible in small space
- Requires one of the streams to be aggregated and reformatted, to compute 1/a(y).
- No problem if streams are naturally aggregated (eg SNMP data)
- Scenario: enough space to capture one stream, then "compress" into Group Testing data structure for later comparison and analysis with new streams

Results

- Show that with probability $1-\delta$, all deltoids are found, no items which are far from being deltoids
- Space is $O(1/\varepsilon \log U \log 1/\delta)$
  Update time is $O(\log U \log 1/\delta)$
  Time to search is linear in the space used
- First one pass solution for absolute change deltoids, and first result on relative change deltoids

Experiments

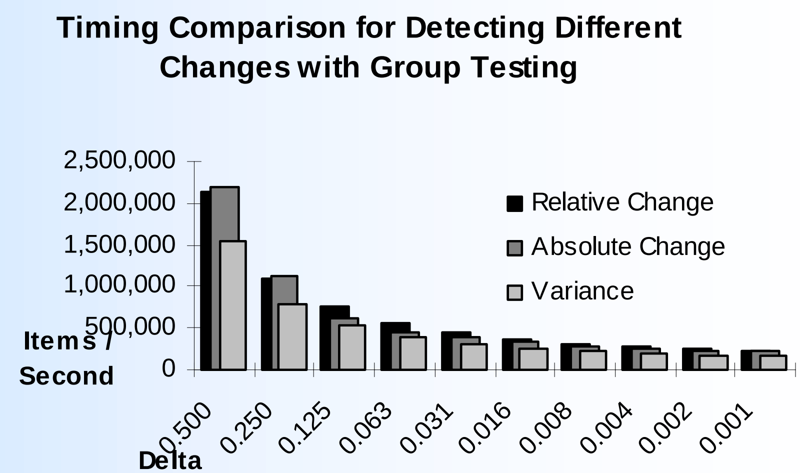




Recall = fraction of deltoids found

Precision = fraction of returned items that are deltoids

Full details to appear in INFOCOM ‘04



Improvements

- Can keep additional tests (CM sketches) to verify the candidate items, reduces space for identification
- log U factor can be painful for high speed data, can decrease this at the cost of more space...
- Instead of reading off one bit at a time, read off one nibble (4x speed, 4x space),
  or one byte (8x speed, 32x space)

Outline

- What's the problem?
- What's hot and what's not?
- What's new?
- **What's next?**

Other Applications

These techniques can be applied to several other fundamental stream problems:

- Range Sum Estimation
- Inner Product Estimation
- Approximate Quantiles Finding
- Hierarchical Heavy Hitters (HHH) etc.
- Wavelets and Histograms…

Pairwise independence sufficient for all

Group testing paradigm approach is fundamental

Ongoing Work

- Agenda: Move other stream algorithms from the theoretical to the practical
- More implementations and experiments with existing and developing work
- Other problems: eg Burst detection on text streams
- Other scenarios: Items in hierarchies, eg IP addresses (HHH in VLDB 03, HHHH in progress)

Other Directions

- Massive geometric data — streams of points from mobile clients. Massive Graphs — streams of edges
- Some problems can be solved by turning them into vector style problems and using sketches etc.
- More satisfying to find new solutions. Eg, Radial Histogram: a division space allowing approximation of geometric aggregates, join size estimation.

Questions

- Why do ghouls and demons hang out together?
- Because demons are a ghouls best friend.