Some Sketchy Results

Graham Cormode

graham@research.att.com
Intro to Sketches

- “Sketch” data structures are compact, randomized summaries
- Term coined by Broder in 1997
  - Exact interpretation varies
- Common sketch properties:
  - Approximate a holistic function
  - Sublinear in size of the input
  - Linear transform of input
  - Can easily merge sketches
Sketch Types

- **(Linear) Fingerprints** for equality tests (~1981)
  - Gives updatable randomized equality tests in constant space
- **Bloom filters** for set membership queries (1970)
  - Can be made linear transforms of the input
- **Min-wise hashes** for (Jaccard) similarity and sampling (~1997)
  - Not linear, but mergeable / distributable
- **Counting sketches** summarize distributions (1996, 99, 02, 03)
  - Count sketch, AMS, Count-min etc.
  - Flajolet-Martin, Gibbons-Tirthapura, BJKST etc.
Sketches in the Field

- Sketches have been widely used in many applications
- **Why** are they successful?
  - Often simple to implement
  - Solve foundational problems well
  - Can seem magical on first encounter
- Why aren’t they **more successful**?
  - **Primarily**: not yet fully mainstream
- What can we do to **promote** their success?
Count-Min Sketch

- Simple sketch idea, can be used within many different tasks
- Model input data as a vector $\mathbf{x}$ of dimension $m$
- Creates a small summary as an array of $w \times d$ in size
- Use $d$ hash function to map vector entries to $[1..w]$
- (Implicit) linear transform of input vector, so flexible
Count-Min Sketch Structure

- Each entry in vector $x$ is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Estimate $x[j]$ by taking $\min_k \text{CM}[k, h_k(j)]$
  - Guarantees error less than $\varepsilon F_1$ in size $O(1/\varepsilon \log 1/\delta)$ (Markov ineq)
  - Probability of more error is less than $1-\delta$

$w = 2/\varepsilon$

$d = \log 1/\delta$

$\text{Sketches}$
Count-Min for “Heavy Hitters”

- After sequence of items, can estimate $f_i$ for any $i$ (up to $\varepsilon N$)
- Heavy Hitters are all those $i$ s.t. $f_i > \phi N$
- Slow way: test every $i$ after creating sketch
- Faster way: test every $i$ after it is seen, and keep largest $f_i$’s
- Alternate way:
  - keep a binary tree over the domain of input items, where each node corresponds to a subset
  - keep sketches of all nodes at same level
  - descend tree to find large frequencies, discarding branches with low frequency
**F₀ Sketch**

- F₀ is the number of distinct items in a multiset
  - a fundamental quantity with many applications
- [BJKST02] Pick random hash over items, h: [m] \(\rightarrow\) [m³]

For each item i, compute h(i), and track the t distinct items achieving the smallest values of h(i)
- **Note**: whenever i occurs, h(i) is same
- Let \(v_t\) = t’th smallest value of h(i) seen.

If F₀ < t, give exact answer, else estimate \(F'_0 = tm^3/v_t\)
- \(v_t/m^3\) ≈ fraction of hash domain occupied by t smallest
- Analysis shows relative error \((1 \pm 1/\sqrt{vt})\) via Chebyshev bound
**F₀ Sketch Properties**

- **Space cost for 1 ± ε error:**
  - Store $t = 1/\varepsilon^2$ hash values, so $O(1/\varepsilon^2 \log m)$ bits
  - Can improve to $O(1/\varepsilon^2 + \log m)$ with additional tricks

- **Time cost:**
  - Hash $i$, update $v_t$ and list of $t$ smallest if necessary
  - Total time $O(\log 1/\varepsilon + \log m)$ worst case

- **Generalization [Gibbons-Tirthapura 01, Beyer-HRSG09]:**
  - Store $t$ original items with their hash values ("distinct sample")
  - Estimate number of distinct items satisfying some predicate
  - **Other extensions:** can allow (multiset) deletions
“Compressed Sensing” has been rocking the EE world since 2004
- Design a compact measurement matrix $M$
- Given product $(Mx)$, recover a good approximation of vector $x$
- Optimize: rows of $M$, density of $M$, recovery time, error prob

Sketch techniques yield compressed sensing techniques
- Very sparse binary $M$, very fast decoding, but weaker error prob

Has launched a line of research on sparse recovery
- See Gilbert-Indyk survey, wiki
Application: Stream Data Analysis

- Many “big data” applications generate large data streams
  - Network traffic analysis, web log analysis
- Sketches allow complex reports on large streaming data
  - In GS-tool (AT&T), CMON (Sprint) for telecom/network data
  - In Sawzall (Google), the only permitted tool for any log analysis
- E.g. track popular queries, number of distinct destinations
Application: Sensor Networks

- Sensor networks distribute many small, weak sensors
  - (Mergeable) sketches fit in here exactly
- Problem: no one actually does anything like this [Welsh 10]
  - Most sensor deployments have few nodes, careful placement
  - Attempt to capture all data, no in-network processing
- Hundreds of papers, but algorithms not in this field (yet)
Other Emerging Applications

- Machine learning over huge numbers of features
- Data mining: scalable anomaly/outlier detection
- Database query planning
- Password quality checking [HSM 10]
- Large linear algebra computations
- Cluster computations (MapReduce)
- Distributed Continuous Monitoring
- Privacy preserving computations
- ... [Your application here?]
**Sketch Issues**

**Strengths**
- Easy to code up and use
  - Easier than exact algs
- Small — cache-friendly
  - So can be very fast
- Open source implementations
  - (maybe barebones, rigid)
- Easily teachable
  - As intro to probabilistic analysis
- Highly parallel

**Weaknesses**
- (Still) resistance to random, approx algs
  - Less so for Bloom filter, hashes
- Memory/disk is cheap
  - Unless data is “too Big To File”
- Not yet in standard libraries
- Not yet in ugrad curricula/texts
  - “this CM sketch sounds like the bomb! (although I have not heard of it before)”
- Looking for killer parallel apps
Open Problems

- More sketches for applications
- More applications for sketches
- More outreach/PR for sketches

More info:
- Wiki: sites.google.com/site/countminsketch/
- “Sketch Techniques for Approximate Query Processing”
  www.eecs.harvard.edu/~michaelm/CS222/sketches.pdf