Distributed Summaries

Graham Cormode
graham@research.att.com

Pankaj Agarwal (Duke)
Zengfeng Huang (HKUST)
Jeff Philips (Utah)
Zhewei Wei (HKUST)
Ke Yi (HKUST)
Summaries

Summaries allow approximate computations:

- Euclidean distance (Johnson-Lindenstrauss lemma)
- Vector Inner-product, Matrix product (sketches)
- Distinct items (Flajolet-Martin onwards)
- Frequent Items (Misra-Gries onwards)
- Compressed sensing
- Subset-sums (samples)
Approximation and Parallel Computation

♦ Why use approximate when data storage is cheap?
  – Parallelize computation: partition and summarize data
    ■ Consider holistic aggregates, e.g. count-distinct
  – Faster computation (only send summaries, not full data)
    ■ Less marshalling, load balancing needed
  – Implicit in some tools (Sawzall)
Mergability

- Ideally, summaries are algebraic: associative, commutative
  - Allows arbitrary computation trees
    (see also synopsis diffusion [Nath+04], MUD model)
  - Distribution “just works”, whatever the architecture

- Summaries should have bounded size
  - Ideally, independent of base data size
  - Or sublinear in base data (logarithmic, square root)
  - Should **not** depend on number of merges
  - Rule out “trivial” solution of keeping union of input
Models of Summary Construction

♦ Offline computation: e.g. sort data, take percentiles
♦ Streaming: summary merged with one new item each step
♦ One-way merge: each summary merges into at most one
  – Single level hierarchy merge structure
  – Caterpillar graph of merges
♦ Equal-size merges: can only merge summaries of same arity
♦ Full mergeability: allow arbitrary merging schemes
  – Our main interest
Merging: sketches

♦ Example: most sketches (random projections) fully mergeable
♦ Count-Min sketch of vector \( x[1..U] \):
  - Creates a small summary as an array of \( w \times d \) in size
  - Use \( d \) hash functions \( h \) to map vector entries to \([1..w]\)
  - Estimate \( x[i] = \min_j \text{CM}[h_j(i), j] \)
♦ Trivially mergeable: \( \text{CM}(x + y) = \text{CM}(x) + \text{CM}(y) \)

Array: \( \text{CM}[i,j] \)
Merging: sketches

- **Consequence** of sketch mergability:
  - Full mergability of quantiles, heavy hitters, F0, F2, dot product...
  - Easy, widely implemented, used in practice

- **Limitations** of sketch mergeability:
  - Probabilistic guarantees
  - May require discrete domain (ints, not reals or strings)
  - Some bounds are logarithmic in domain size
Summaries for heavy hitters

- **Misra-Gries (MG) algorithm** finds up to $k$ items that occur more than $1/k$ fraction of the time in a stream.
- Keep $k$ different candidates in hand. For each item in stream:
  - If item is monitored, increase its counter.
  - Else, if $< k$ items monitored, add new item with count 1.
  - Else, decrease all counts by 1.
Streaming MG analysis

- \( N \) = total weight of input
- \( M \) = sum of counters in data structure
- Error in any estimated count at most \( \frac{(N-M)}{(k+1)} \)
  - Estimated count a lower bound on true count
  - Each decrement spread over \( (k+1) \) items: 1 new one and \( k \) in MG
  - Equivalent to deleting \( (k+1) \) distinct items from stream
  - At most \( \frac{(N-M)}{(k+1)} \) decrement operations
  - Hence, can have “deleted” \( \frac{(N-M)}{(k+1)} \) copies of any item
Merging two MG Summaries

♦ Merging alg:
  – Merge the counter sets in the obvious way
  – Take the \((k+1)\)th largest counter \(= C_{k+1}\), and subtract from all
  – Delete non-positive counters
  – Sum of remaining counters is \(M_{12}\)

♦ This alg gives full mergeability:
  – Merge subtracts at least \((k+1)C_{k+1}\) from counter sums
  – So \((k+1)C_{k+1} \leq (M_1 + M_2 - M_{12})\)
  – By induction, error is
    \[
    ((N_1 - M_1) + (N_2 - M_2) + (M_1 + M_2 - M_{12}))/\!(k+1) = ((N_1 + N_2) - M_{12})/(k+1)
    \]
Quantiles

- Quantiles / order statistics generalize the median:
  - Exact answer: $CDF^{-1}(\phi)$ for $0 < \phi < 1$
  - Approximate version: tolerate answer in $CDF^{-1}(\phi - \varepsilon) \ldots CDF^{-1}(\phi + \varepsilon)$

- **Hoeffding bound**: sample of size $O(1/\varepsilon^2 \log 1/\delta)$ suffices

- **Easy result**: one-way mergeability in $O(1/\varepsilon \log (\varepsilon n))$
  - Assume a streaming summary (e.g. Greenwald-Khanna)
  - Extract an approximate CDF $F$ from the summary
  - Generate corresponding distribution $f$ over $n$ items
  - Feed $f$ to summary, error is bounded
  - **Limitation**: repeatedly extracting/inserting causes error to grow

Mergeable Summaries
Equal-weight merging quantiles

♦ A classic result (Munro-Paterson ’78):
  – **Input**: two summaries of equal size $k$
  – **Base case**: fill summary with $k$ input items
  – Merge, sort summaries to get size $2k$
  – Take every other element

♦ **Deterministic bound**:
  – Error grows proportional to height of merge tree
  – Implies $O(1/\varepsilon \log^2 n)$ sized summaries (for $n$ known upfront)

♦ **Randomized twist**:
  – Randomly pick whether to take odd or even elements
Equal-size merge analysis

♦ Analyze error in range count for any interval after \( m \) merges
♦ Absolute error introduced by \( i \)'th level merge is \( 2^{i-1} \)
♦ **Unbiased**: expected error is 0 (50-50 +\( 2^{i-1} \) / -\( 2^{i-1} \))
♦ Apply Chernoff bound to sum of errors
♦ Summary size = \( O(\frac{1}{\epsilon} \log^{1/2} \frac{1}{\delta}) \) gives \( \epsilon N \) error w/prob 1-\( \delta \)
  – **Neat**: naïve sampling bound requires \( O(\frac{1}{\epsilon^2} \log \frac{1}{\delta}) \)
  – Tightens randomized result of [Suri Toth Zhou 04]
Fully mergeable quantiles

- Use equal-size merging in a standard logarithmic trick:

- Merge two summaries as binary addition

- Fully mergeable quantiles, in $O(1/\varepsilon \log (\varepsilon n) \log^{1/2} 1/\delta)$
  - $n = \text{number of items summarized, not known a priori}$

- But can we do better?
Hybrid summary

☀ **Observation**: when summary has high weight, low order blocks don’t contribute much
  – Can’t ignore them entirely, might merge with many small sets

☀ **Hybrid structure**:  
  – Keep top $O(\log 1/\varepsilon)$ levels as before  
  – Also keep a “buffer” sample of (few) items  
  – Merge/keep equal-size summaries, and sample rest into buffer

☀ **Analysis rather delicate**:  
  – Points go into/out of buffer, but always moving “up”  
  – Gives constant probability of accuracy in $O(1/\varepsilon \log^{1.5}(1/\varepsilon))$
Other Fully Mergeable Summaries

- Samples on distinct (aggregated) keys
- $\varepsilon$-approximations in constant VC-dimension $v$ in $O(\varepsilon^{-2v/(v+1)})$
- $\varepsilon$-kernels in $d$-dimensional space in $O(\varepsilon^{(1-d)/2})$
  - For “fat” pointsets: bounded ratio between extents in any direction
- Equal-weight merging for k-median implicit from streaming
  - Implies $O(poly\ n)$ fully-mergeable summary via logarithmic trick
Open Problems

- Weight-based sampling over non-aggregated data
- Fully mergeable $\varepsilon$-kernels without assumptions
- More complex functions, e.g. cascaded aggregates
- Lower bounds for mergeable summaries
- Implementation studies (e.g. in Hadoop)