ECEN 689
Special Topics in Data Science for Communications Networks

Nick Duffield
Department of Electrical & Computer Engineering
Texas A&M University

Lecture 8
Count-Min Sketches
Data Stream Model

• Vector $a(t) = (a_1(t), a_2(t), \ldots, a_n(t))$, initially $a_i(0) = 0$ for all $i$

• Update $(i(t), c(t))$ for each $t$

• Update rule:
  
  $a_i(t) = a_i(t-1) + c(t)$ for $i = i(t)$
  $a_i(t) = a_i(t-1)$ for $i \neq i(t)$

• Example
  – Update at $t$ is flow record arrival
    • $i(t)$ = flow key; $c(t)$ = bytes in flow;
    • $a_i(t)$ is cumulative bytes seen since 0 for key $i$
Count-Min Sketch: Structure

• Can we approximate the totals $a_i(t)$ in smaller memory?
• A Count-Min Sketch with parameters $(\epsilon, \delta)$ is a two-dimensional array with $d$ rows and $w$ columns, and entries

$$\{K[i,j]: i=1,\ldots,d, j=1,\ldots,w\},$$ all initialized to 0

where $w = \text{ceil}(\epsilon / \epsilon)$ and $d = \text{ceil}(\log(1/\delta))$.

• The Count-Min Sketch is equipped with a family of $d$ hash functions assumed pairwise independent:

$$h_1, \ldots, h_d : [1,n] \rightarrow [1,w]$$
Count-Min Update

• When \((i(t), c(t))\) arrives:

  add \(c(t)\) to the element at column \(h_j(i(t))\) in each row \(j\)

\[ K[j, h_j(i(t))] += c(t) \text{ for } j = 1, 2, \ldots, d \]
Count-Min: Basic Idea

• If there were no hash collisions then after arrival \((i(t),c(t))\) would have
  – \(K[j, h_j(i)] = a_i(t)\) for each hash \(h_j\), \(j=1,...,d\)

• If hash collisions
  – \(h_j(i(s)) = h_j(i(t))\) for some \(s<t\), \(i(s)\neq i(t)\), and hash \(h_j\)
  – Hence \(K[j, h_j(i)] > a_i(t)\)

• Hash functions are independent, unlikely to all collide.
  – For so \(w\) and \(d\) large enough, very likely that \(\min_j K[j, h_j(i)] = a_i(t)\)
  – Reminiscent of Bloom Filter
Count-Min: Approximation

• Approximate $a_i$ by $a_i' = \min_j K[j, h_j(i)]$ (omit time index)

• In row $j$: $K[j, h_j(i)] = a_i(t) + X_{i,j}$

  with $X_{i,j} = \sum a_k$ where sum is over $k$ such that $h_j(i) = h_j(k)$

• $E[X_{i,j}] = \sum_k a_k \cdot \Pr[h_j(i) = h_j(k)]$

  $\leq |a| / w$

  $\leq |a| \varepsilon / 2$  

  here $|a| = \sum_k a_k$

• $\Pr[X_{i,j} \geq \varepsilon |a|] \leq \Pr[X_{i,j} \geq 2 E[X_{i,j}]] \leq 1/2$ (Markov Inequ.)

• $\Pr[a_i' \geq a_i + \varepsilon |a|] = \Pr[X_{i,j} \geq \varepsilon |a|_1 \text{ all hash } j] \leq 1/2^d \leq \delta$
Count-Min: Summary for Point Queries

• $(\varepsilon, \delta)$ Count-Min Sketch
  – $w = \text{ceil}(\varepsilon / \varepsilon)$ rows and $d = \text{ceil}(\log(1/\delta))$ hash functions

• Bounds
  – Exact lower bound: $a'_{i} \geq a_{i}$
  – Probabilistic upper bound: $\Pr[a'_{i} \geq a_{i} + \varepsilon | a|] \leq \delta$

• Space used:
  – Size of array = $wd = O((1/\varepsilon) \log(1/\delta))$

• Computation cost for updates
  – Number of hash functions $d = O(\log(1/\delta))$

• Computation cost for estimates
  – Number of rows $d = O(\log(1/\delta))$
Count-Min: Other Queries

• Inner product queries: $a \cdot b = \Sigma_i a_i b_i$

• Range queries: $\Sigma_{r=1}^s a_i$
Count-Min: Inner Product Query

• Want $a \cdot b = \sum_i a_i b_i$

• Maintain separate Count-Min sketches $K_a$ and $K_b$

• Row j inner product $(K_a \cdot K_b)_j = \sum_{i=1}^w K_a[j , i] K_b[j , i]$

• Estimate min-wise: $(a.b)' = \min_j (K_a \cdot K_b)_j$

• Theorem: (similar to before)

\[
(a.b)' \geq a.b \quad \text{and} \quad \Pr[(a.b)' > a.b + \varepsilon|a|_1 |b|_1 ] \leq \delta
\]

• Space and costs same as for point query
Count-Min: Range Queries

- Range queries: \( a[r,s] = \sum_{r=1}^{s} a_i \)

- Possible approach: estimate \( \sum_{r=1}^{s} a_i \) by \( \sum_{r=1}^{s} a'_i \)

- Problem: errors add linearly with number of terms
Dyadic Partitions on $[1,n]$

- Partition $n$ into **dyadic ranges** in $O(\log n)$ ways

  - $\{\{1\}, \{2\}, \{3\}, \{4\}, \ldots, \{n\}\}$ Partition into singletons
  - $\{\{1,2\}, \{3,4\}, \{5,6\}, \ldots, \{n-1,n\}\}$ Partition into pairs
  - $\{\{1,2,3,4\}, \{5,6,7,8\}, \ldots, \{n-3,n-2,n-1,n\}\}$
  - etc…
  - $\{1,2,3,\ldots,n\}$

Any subinterval $[r,s]$ on $[1,n]$ can be expressed as a disjoint union of at most $2 \log n$ dyadic ranges
Count-Min: Range Queries in [1,n]

• Maintain log n separate Count-Min Sketches
  – One for each dyadic partition
  – Columns of sketch correspond to the dyadic ranges in the partition

• Update accordingly for each arrival \((i(t),c(t))\)
  – For each dyadic partition
  – For each hash \(h\)
  – Add \(c(t)\) to the count for the dyadic interval containing \(h(i(t))\)

• Suppose \([r,s]\) is union of dyadic ranges \(\{D_1, D_2, \ldots, D_k\}\)

• Estimate \(a[r,s]\) by \(a'[r,s] = \sum_{i=1}^{k} a'(D_i)\) where \(a'(D_i)\) is Count-Min estimator for dyadic range \(D_i\)
Count-Min: Range Queries: Bounds

- Exact lower bound $a'[r,s] \geq a[r,s]$
- Probabilistic upper bound
  \[ Pr[a'[r,s] \geq a[r,s] + 2\varepsilon \log n |a|_1] \leq \delta \]
- Basic idea:
  - $E[\text{total error}] = \log(n) E[\text{error per interval}]$; use Markov as before
- Note with of upper bound increasing with $n$

- Space: $O(\log(n)/\varepsilon \log(1/\delta))$
- Update Cost: $O(\log(n) \log(1/\delta))$
- Estimate Cost: $O(\log(n) \log(1/\delta))$
- All increase by factor $\log(n)$: number of sketches
Count-Min Sketch: Quantile Queries

- Consider $a_i$ as weights of distribution over \{1,2,\ldots,n\}

- $\phi$-quantile: $j : a_1,\ldots,a_j$ have fraction $\phi$ of total weight $|a|_1$

  $$\sum_{i=1,..,j} a_i \leq \phi |a|_1 \leq \sum_{i=1,..,j+1} a_i$$

- Example: median, with $\phi = \frac{1}{2}$

- Find approximate quantiles: tolerate error $\epsilon |a|_1$

- Method: binary search on range sums
  - Exploit binary tree structure of dyadic sums
Count-Min Sketch Heavy Hitters

- **ϕ-heavy hitters**: \( i \) that have at least fraction \( \phi \) of total weight:
  \[
a_i \geq \phi |a|_1
  \]

- Find using search on dyadic tree:
  \[
  \{1,2,3,4,5,6,7,8\}
  \]
  \[
  \{1,2,3,4\} \quad \{5,6,7,8\}
  \]
  \[
  \{1,2\} \quad \{3,4\} \quad \{5,6\} \quad \{7,8\}
  \]
  \[
  \{1\} \quad \{2\} \quad \{3\} \quad \{4\} \quad \{5\} \quad \{6\} \quad \{7\} \quad \{8\}
  \]

- If \( a_i \) is a \( \phi \)-heavy hitter; all ancestors \( D \) of \( \{i\} \) have \( a[D] \geq \phi |a|_1 \)
Heavy Hitter Application

- Internet traffic
- Want to find dominant IP addresses
  - Originating or receiving large proportion of total traffic
  - Large website, DDOS victim
- Exploit natural dyadic tree structure of IP addresses based on prefixes
Reference

• Count-Min Sketch, Graham Cormode