Data Streams: Where to Go?

PODS 11, Tutorial

S. *Muthu* Muthukrishnan

Familiar Puzzle: Missing Number

- A shows B numbers $1, \ldots, n$ but in a permuted order and leaves out one of the numbers.

Familiar Puzzle: Missing Number

- A shows B numbers $1, \ldots, n$ but in a permuted order and leaves out one of the numbers.
- B has to determine the missing number.

Familiar Puzzle: Missing Number

- A shows B numbers $1, \ldots, n$ but in a permuted order and leaves out one of the numbers.
- B has to determine the missing number.
- Key: B has only $2 \log n$ bits.

Familiar Puzzle: Missing Number

- A shows B numbers $1, \ldots, n$ but in a permuted order and leaves out one of the numbers.
- B has to determine the missing number.
- Key: B has only $2 \log n$ bits.
- Solution:
  B maintains the running sum s of numbers seen.
  Missing number is $\frac{n(n+1)}{2} - s$.

A New Puzzle: One Word Median

A New Puzzle: One Word Median

- A sees items $i_1, i_2, \ldots$ arrive in a stream.
- A has to maintain the median m_j of the items $i_1, \ldots, i_j$.

A New Puzzle: One Word Median

- A sees items $i_1, i_2, \ldots$ arrive in a stream.
- A has to maintain the median m_j of the items $i_1, \ldots, i_j$.
- Each i_j generated independently and randomly from some unknown distribution $\mathcal{D}$ over integers $[1, n]$.

A New Puzzle: One Word Median

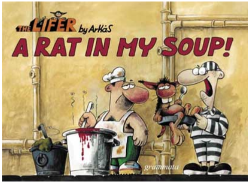
- A sees items $i_1, i_2, \ldots$ arrive in a stream.
- A has to maintain the median m_j of the items $i_1, \ldots, i_j$.
- Each i_j generated independently and randomly from some unknown distribution $\mathcal{D}$ over integers $[1, n]$.
- Key: A is allowed to store only one word of memory (of $\log n$ bits).

A New Puzzle: One Word Median

- A sees items $i_1, i_2, \ldots$ arrive in a stream.
- A has to maintain the median m_j of the items $i_1, \ldots, i_j$.

- Each i_j generated independently and randomly from some unknown distribution $\mathcal{D}$ over integers $[1, n]$.

- Key: A is allowed to store only one word of memory (of $\log n$ bits).



- Solution. Maintain μ_j.
  If $i_{j+1} > \mu_j$, $\mu_{j+1} \leftarrow \mu_j + 1$.
  If $i_{j+1} < \mu_j$, $\mu_{j+1} \leftarrow \mu_j - 1$.

A Basic Problem: Indexing

- Imagine a virtual array $F[1 \cdots n]$,

A Basic Problem: Indexing

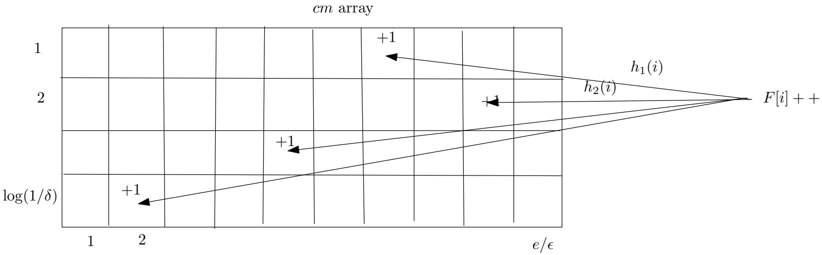
- Imagine a virtual array $F[1 \cdots n]$,
- Updates: $F[i]++$, $F[i]--$.

A Basic Problem: Indexing

- Imagine a virtual array $F[1 \cdots n]$,
- Updates: $F[i]++$, $F[i]--$.
- Query: $F[i] = ?$.

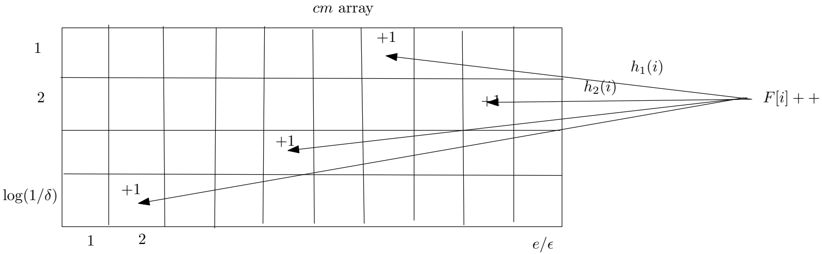
Count-Min Sketch

- For each update $F[i]++$,
  - for each $j = 1, \ldots, \log(1/\delta)$, update $cm[h_j(i)]++$.



Count-Min Sketch

- For each update $F[i]++$,
  - for each $j = 1, \ldots, \log(1/\delta)$, update $cm[h_j(i)]++$.



- Estimate $\tilde{F}(i) = \min_{j=1,\ldots,\log(1/\delta)} cm[h_j(i)]$.

Count-Min Sketch

- Claim: $F[i] \leq \tilde{F}[i]$.
- Claim: With probability at least $1 - \delta$, $\tilde{F}[i] \leq F[i] + \varepsilon \sum_{j \neq i} F[j]$.
- Space used is $O(\frac{1}{\varepsilon} \log(\frac{1}{\delta}))$.
- Time per update is $O(\log(\frac{1}{\delta}))$. Indep of n.

G. Cormode and S. Muthukrishnan: An improved data stream summary: count-min sketch and its applications. *Journal of Algorithms*.

Count-Min Sketch: The Proof

- With probability at least $1 - \delta$,

$$\tilde{F}[i] \leq F[i] + \varepsilon \sum_{j \neq i} F[j].$$

Count-Min Sketch: The Proof

- With probability at least $1 - \delta$,

$$\tilde{F}[i] \leq F[i] + \varepsilon \sum_{j \neq i} F[j].$$

- $X_{i,j}$ is the expected contribution of $F[j]$ to the bucket containing i, for any h.

$$E(X_{i,j}) = \frac{\varepsilon}{e} \sum_{j \neq i} F[j].$$

Count-Min Sketch: The Proof

- With probability at least $1 - \delta$,

$$\tilde{F}[i] \leq F[i] + \varepsilon \sum_{j \neq i} F[j].$$

- $X_{i,j}$ is the expected contribution of $F[j]$ to the bucket containing i, for any h.

$$E(X_{i,j}) = \frac{\varepsilon}{e} \sum_{j \neq i} F[j].$$

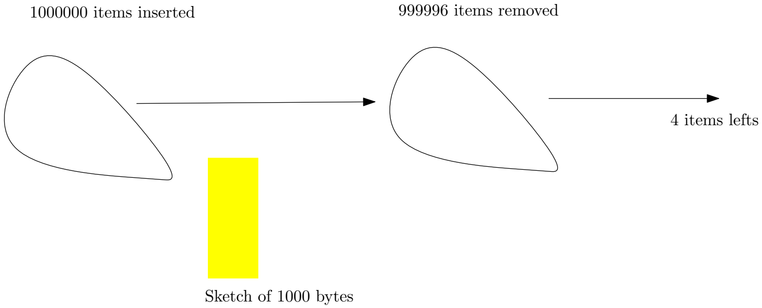
- Consider $\Pr(\tilde{F}[i] > F[i] + \varepsilon \sum_{j \neq i} F[j])$:

$$\begin{aligned} \Pr() &= \Pr(\forall j, F[i] + X_{i,j} > F[i] + \varepsilon \sum_{j \neq i} F[j]) \\ &= \Pr(\forall j, X_{i,j} \geq eE(X_{i,j})) \\ &< e^{-\log(1/\delta)} = \delta \end{aligned}$$

Improve Count-Min Sketch?

- Index Problem:
  - A has n long bitstring and sends messages to B who wishes to compute the ith bit.
  - Needs $\Omega(n)$ bits of communication.
- Reduction of estimating F[i] in data stream model.
  - $I[1 \cdots 1/(2\varepsilon)]$
  - $I[i] = 1 \to F[i] = 2$
  - $I[i] = 0 \to F[i] = 0; F[0] \leftarrow F[0] + 2.$
  - Estimating $F[i]$ to $\varepsilon||F|| = 1$ accuracy reveals $I[i]$.

Count-Min Sketch, The Challenge



- Recovering $F[i]$ to $\pm 0.1|F|$ accuracy will retrieve each item precisely.

Applications of Count-Min Sketch

- Solves many problems in the best possible/known bounds:
  - Data Mining: heavy hitters, heavy hitter differences.
  - Signal processing: Compressed sensing.
  - Statistics: Histograms, Wavelets, Clustering, Least squares.

Applications of Count-Min Sketch

- Solves many problems in the best possible/known bounds:
  - Data Mining: heavy hitters, heavy hitter differences.
  - Signal processing: Compressed sensing.
  - Statistics: Histograms, Wavelets, Clustering, Least squares.
- Applications to other CS/EE areas:
  - NLP, ML, Password checking.

Applications of Count-Min Sketch

- Solves many problems in the best possible/known bounds:
  - Data Mining: heavy hitters, heavy hitter differences.
  - Signal processing: Compressed sensing.
  - Statistics: Histograms, Wavelets, Clustering, Least squares.
- Applications to other CS/EE areas:
  - NLP, ML, Password checking.
- Systems, code, hardware.

Applications of Count-Min Sketch

- Solves many problems in the best possible/known bounds:
  - Data Mining: heavy hitters, heavy hitter differences.
  - Signal processing: Compressed sensing.
  - Statistics: Histograms, Wavelets, Clustering, Least squares.
- Applications to other CS/EE areas:
  - NLP, ML, Password checking.
- Systems, code, hardware.

Wiki: http://sites.google.com/site/countminsketch/

Summary

- Broken the premise that data has to be
  - captured,
  - stored,
  - communicated,

  analyzed in entirety.

Summary

- Broken the premise that data has to be
  - captured,
  - stored,
  - communicated,

  analyzed in entirety.

Polynomial time/space theory -> sublinear theory
Nyquist sampling -> SubNyquist sampling

What does this got to do with data streams?
Some My-story

- Raghu asked: what can you do with one pass?
  - Dynamic data structures, with fast update times.

What does this got to do with data streams?
Some My-story

- Raghu asked: what can you do with one pass?
  - Dynamic data structures, with fast update times.
- Gibbons and Matias abstract synopsis data structures
  - Can't simulate a stack!

What does this got to do with data streams?
Some My-story

- Raghu asked: what can you do with one pass?
  - Dynamic data structures, with fast update times.
- Gibbons and Matias abstract synopsis data structures
  - Can't simulate a stack!
- Alon, Matias and Szegedy used limited independence.
  - What does frequency moment got to do with databases?

What does this got to do with data streams? Some My-story

- Raghu asked: what can you do with one pass?
  - Dynamic data structures, with fast update times.
- Gibbons and Matias abstract synopsis data structures
  - Can't simulate a stack!
- Alon, Matias and Szegedy used limited independence.
  - What does frequency moment got to do with databases?
- George Varghese argues high speed memory is a constraint in IP packet analyses.
  - Who needs to analyze IP packet data?

What does this got to do with data streams?
Some My-story

- Raghu asked: what can you do with one pass?
  - Dynamic data structures, with fast update times.
- Gibbons and Matias abstract synopsis data structures
  - Can't simulate a stack!
- Alon, Matias and Szegedy used limited independence.
  - What does frequency moment got to do with databases?
- George Varghese argues high speed memory is a constraint in IP packet analyses.
  - Who needs to analyze IP packet data?
- Observation: $1/\varepsilon^2$ space to give ε accuracy. Prohibitive.

Some Successful Streaming Systems

Specialized streaming systems:

- Gigascope at AT&T for IP traffic analysis.
  - Two level archietecture.
    Uses $\text{count-min}(A) + \text{count-min}(B) = \text{count-min}(A + B)$.

Some Successful Streaming Systems

Specialized streaming systems:

- Gigascope at AT&T for IP traffic analysis.
  - Two level archietecture.
    Uses count-min(A) + count-min(B) = count-min$(A + B)$.
- CMON at Sprint for IP traffic analysis.
  - Hash and parallelize architecture.
    Uses count-min sketch to skip over parts of the stream.

Some Successful Streaming Systems

Specialized streaming systems:

- Gigascope at AT&T for IP traffic analysis.
  - Two level archietecture.
    Uses $\text{count-min}(A) + \text{count-min}(B) = \text{count-min}(A + B)$.
- CMON at Sprint for IP traffic analysis.
  - Hash and parallelize architecture.
    Uses count-min sketch to skip over parts of the stream.
- Sawzall at Google for log data analysis
  - Mapreduce-based.
    Uses count-min sketch to decrease communication.

Some Successful Streaming Systems

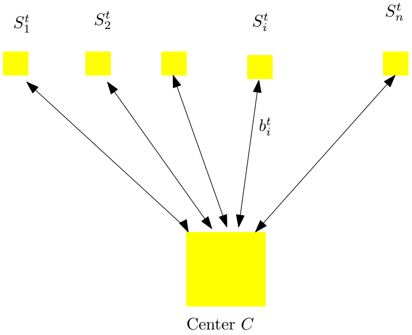
Specialized streaming systems:

- Gigascope at AT&T for IP traffic analysis.
  - Two level archietecture.
    Uses $\text{count-min}(A) + \text{count-min}(B) = \text{count-min}(A + B)$.
- CMON at Sprint for IP traffic analysis.
  - Hash and parallelize architecture.
    Uses count-min sketch to skip over parts of the stream.
- Sawzall at Google for log data analysis
  - Mapreduce-based.
    Uses count-min sketch to decrease communication.

Q: General purpose streaming systems?

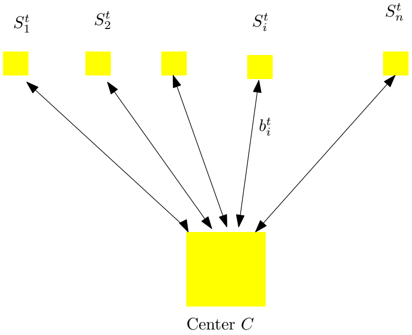
Some Research Directions

1: Distributed, continual monitoring



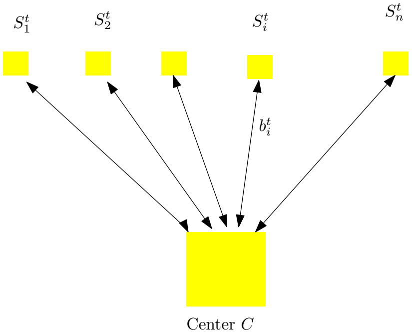
- S_i^t is the set of items seen by sensor i upto time t.
- S_t is the multiset union of S_i^t's.

1: Distributed, continual monitoring



- S_i^t is the set of items seen by sensor i upto time t.
- S_t is the multiset union of S_i^t's.
- Problem:
  - If $|S_t| > \tau$, output 1.
  - if $|S_t| < \tau - \varepsilon$, output 0.

1: Distributed, continual monitoring



- S_i^t is the set of items seen by sensor i upto time t.
- S_t is the multiset union of S_i^t's.
- Problem:
  - If $|S_t| > \tau$, output 1.
  - if $|S_t| < \tau - \varepsilon$, output 0.
- Say b_i^t is total number of bits sent b/w i and C
- Minimize $\sum_i b_i^t$.

1: Distributed, continual monitoring

- When sensor sees $O(\frac{\varepsilon^2 \tau}{k})$ elements, sends a bit w.p $\frac{1}{k}$ to center.

1: Distributed, continual monitoring

- When sensor sees $O(\frac{\varepsilon^2 \tau}{k})$ elements, sends a bit w.p $\frac{1}{k}$ to center.
- Center outputs 1 when $O(1/\varepsilon^2)$ bits received.

1: Distributed, continual monitoring

- When sensor sees $O(\frac{\varepsilon^2 \tau}{k})$ elements, sends a bit w.p $\frac{1}{k}$ to center.
- Center outputs 1 when $O(1/\varepsilon^2)$ bits received.
- $O(\frac{1}{\varepsilon^2} \log(\frac{1}{\delta}))$ bits suffice with prob of success $1 - \delta$.
- Independent of k.

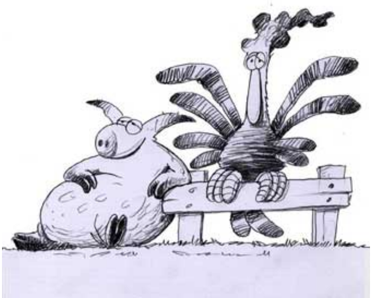
Algorithms for distributed functional monitoring. Cormode, Muthukrishnan, Yi. SODA 08.

1. Distributed, Continual Monitoring: Summary

- Statistics: Frequency moments, Distinct counts.
- Optimization: Clustering.
- Signal processing: Compressed sensing.

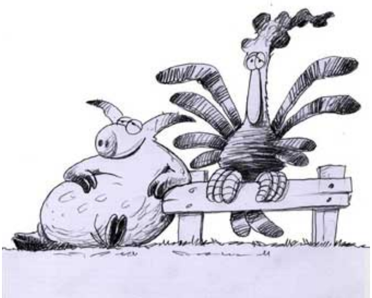
Need a fuller theory.
Connections to Slepian-Wolf, network coding.

2. Probabilistic Streams



- Each stream update is a random variable X_i, $1 \leq i \leq n$, $X_i \in \{0, 1\}$, identically distributed.

2. Probabilistic Streams



- Each stream update is a random variable X_i, $1 \leq i \leq n$, $X_i \in \{0, 1\}$, identically distributed.
- The query is to estimate $\Pr[\sum_i X_i \leq c]$.

Probabilistic Streams Contd

Berry-Esseen Theorem

Let $X_1, \ldots, X_n$ be i.i.d. random variables with

- $E(X_i) = 0, E(X^2) = \sigma^2$, and $E(|X|^3) = \rho$.

Let $Y_n = \sum_i X_i / n$ with

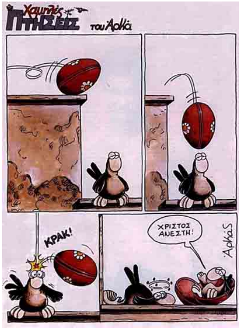
- F_n the cdf of $Y_n \sqrt{n} / \sigma$

ϕ the cdf of the std normal dist.

Then there exists a positive C such that for all x and n,

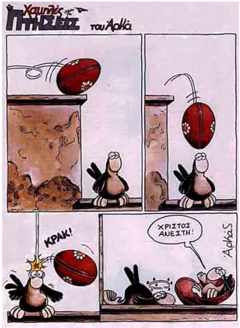
$$|F_n(x) - \phi(x)| \leq \frac{C\rho}{\sigma^3 \sqrt{n}}.$$

Probabilistic Streams Contd



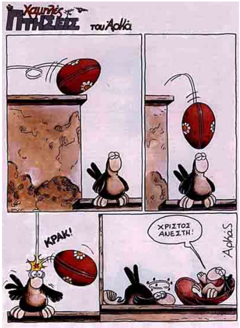
- We have $\sum_i X_i \leq c$ implies $Y_n = \sum_i X_i/n \leq c/n$.

Probabilistic Streams Contd



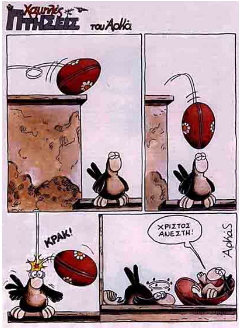
- We have $\sum_i X_i \leq c$ implies $Y_n = \sum_i X_i / n \leq c/n$.
- Then $\Pr(\sum_i X_i \leq c) = F_n(c/\sigma\sqrt{n})$.

Probabilistic Streams Contd



- We have $\sum_i X_i \leq c$ implies $Y_n = \sum_i X_i / n \leq c/n$.
- Then $\Pr(\sum_i X_i \leq c) = F_n(c/\sigma\sqrt{n})$.
- This can be approximated by $\phi(c/\sigma\sqrt{n})$.

Probabilistic Streams Contd



- We have $\sum_i X_i \leq c$ implies $Y_n = \sum_i X_i/n \leq c/n$.
- Then $\Pr(\sum_i X_i \leq c) = F_n(c/\sigma\sqrt{n})$.
- This can be approximated by $\phi(c/\sigma\sqrt{n})$.
- To finish up. Estimate σ and its impact on overall error. Extend to more general X_i's.

3. Stochastic Streams

3. Stochastic Streams

- Input is a stochastic stream $X_1, \ldots, X_n$, each X_i is drawn from known distribution D. n is known.
- Problem: Stop at input t and output X_t.

3. Stochastic Streams

- Input is a stochastic stream $X_1, \ldots, X_n$, each X_i is drawn from known distribution D. n is known.
- Problem: Stop at input t and output X_t.
- Goal: maximize X_t. Formally,

$$\max \frac{E(X_t)}{E(OPT) = E(\max_i X_i)}$$

3. Stochastic Streams

- Input is a stochastic stream $X_1, \ldots, X_n$, each X_i is drawn from known distribution D. n is known.
- Problem: Stop at input t and output X_t.
- Goal: maximize X_t. Formally,

$$\max \frac{E(X_t)}{E(OPT) = E(\max_i X_i)}$$

- Observe:
  - Can *a priori* look at the dist of $\max_i X_i$.
  - Not the same as finding $\max_i X_i$.

3. Stochastic Streams Contd

- Algorithm:
  - $X^* = \max_i X_i$.
  - m: median of X^*, ie., $\Pr(X^* < m) \approx 1/2$.
  - τ is the smallest t such that $X_t > m$.

    τ is the answer.

3. Stochastic Streams Contd

- Algorithm:
  - $X^* = \max_i X_i$.
  - m: median of X^*, ie., $\Pr(X^* < m) \approx 1/2$.
  - τ is the smallest t such that $X_t > m$.

    τ is the answer.
- Algorithm finds t such that $E(X_t)/E(OPT) \geq 1/2$.

  Prophet inequality.

Many basic problems on stochastic streams still open.

Conclusions

- Talk summary:
  - Indexing problem.
  - count-min sketch and applications.
  - classical streaming.
- New directions:
  - Distributed, continual.
  - Probabilistic.
  - Stochastic.

Conclusions

- Talk summary:
  - Indexing problem.
  - count-min sketch and applications.
  - classical streaming.
- New directions:
  - Distributed, continual.
  - Probabilistic.
  - Stochastic.

- Comments:
  - Need convincing systems and applications to motivate new directions.

Conclusions

- Talk summary:
  - Indexing problem.
  - count-min sketch and applications.
  - classical streaming.
- New directions:
  - Distributed, continual.
  - Probabilistic.
  - Stochastic.

- Comments:
  - Need convincing systems and applications to motivate new directions.
  - Left out: window streams, rich queries and data, MapReduce