Tight Lower Bound for Comparison-Based Quantile Summaries

Pavel Veselý
University of Warwick

8 April 2020

Based on joint work with Graham Cormode (Warwick)
Overview of the talk

Quantiles & Distributions & Big Data Algorithms

Streaming Model

Pavel Veselý
Tight Lower Bound for Quantile Summaries
Motivation: Monitoring Latencies of Web Requests

Motivation: Monitoring Latencies of Web Requests

Millions of observations

- no need to store all observed latencies

Motivation: Monitoring Latencies of Web Requests

Millions of observations

- no need to store all observed latencies

How does the distribution look like?

What is the median latency?
Motivation: Monitoring Latencies of Web Requests


Millions of observations

- no need to store all observed latencies

How does the distribution look like?

What is the median latency?

- Average latency too high due to $\sim 2\%$ of very high latencies
Streaming Model

Motivation: monitoring latencies of requests
Streaming Model

Motivation: monitoring latencies of requests

Streaming model = one pass over data & limited memory
Streaming Model

Motivation: monitoring latencies of requests

**Streaming model = one pass over data & limited memory**

Streaming algorithm

- receives data in a *stream*, item by item
- uses memory sublinear in $N = \text{stream length}$
- at the end, computes the answer
Streaming Model

Motivation: monitoring latencies of requests

Streaming model = one pass over data & limited memory

Streaming algorithm

• receives data in a stream, item by item
• uses memory sublinear in $N = \text{stream length}$
• at the end, computes the answer

Challenges:

• $N$ very large & not known
• Data independent
• Stream ordered arbitrarily
• No random access to data
Streaming Model

Motivation: monitoring latencies of requests

Streaming model = one pass over data & limited memory

Streaming algorithm
- receives data in a stream, item by item
- uses memory sublinear in $N = \text{stream length}$
- at the end, computes the answer

Challenges:
- $N$ very large & not known
- Data independent
- Stream ordered arbitrarily
- No random access to data

Main objective: space
Streaming Model

Motivation: monitoring latencies of requests

Streaming model = one pass over data & limited memory

Streaming algorithm

- receives data in a stream, item by item
- uses memory sublinear in \( N = \) stream length
- at the end, computes the answer

Challenges:

- \( N \) very large & not known
- Data independent
- Stream ordered arbitrarily
- No random access to data

Main objective: space

How to summarize the input?
Selection Problem & Streaming

- Input: stream of $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. ’73]

Streaming restrictions:
- just one pass over the data
- limited memory: $o(N)$

No streaming algorithm for exact selection
$\Omega(N)$ space needed to find the median
[Munro & Paterson ’80, Guha & McGregor ’07]

What about finding an approximate median?
Selection Problem & Streaming

- Input: stream of $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. '73]
- Streaming restrictions:
  - just one pass over the data
  - limited memory: $o(N)$
Selection Problem & Streaming

- Input: stream of $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. ’73]
- Streaming restrictions:
  - just one pass over the data
  - limited memory: $o(N)$
- No streaming algorithm for exact selection
  - $\Omega(N)$ space needed to find the median
    [Munro & Paterson ’80, Guha & McGregor ’07]
Selection Problem & Streaming

• Input: stream of \( N \) numbers
• Goal: find the \( k \)-th smallest
  • e.g.: the median, 99th percentile
• \( \mathcal{O}(N) \) time offline algorithm [Blum et al. ’73]
• Streaming restrictions:
  • just one pass over the data
  • limited memory: \( o(N) \)

No streaming algorithm for exact selection
\( \Omega(N) \) space needed to find the median
[Munro & Paterson ’80, Guha & McGregor ’07]

What about finding an approximate median?
Approximate Median & Quantiles

How to define an approximate median?
Approximate Median & Quantiles

How to define an approximate median?

$$\phi$$-quantile = \(\lceil \phi \cdot N \rceil\)-th smallest element \(\phi \in [0, 1]\)

- Median = .5-quantile
Approximate Median & Quantiles

How to define an approximate median?

\( \phi \)-quantile = \([ \phi \cdot N ]\)-th smallest element \((\phi \in [0, 1])\)

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, ..., .99-quantiles
Approximate Median & Quantiles

How to define an approximate median?
\( \phi \)-quantile = \( \lceil \phi \cdot N \rceil \)-th smallest element (\( \phi \in [0, 1] \))

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, \ldots, .99-quantiles

\( \varepsilon \)-approximate \( \phi \)-quantile = any \( \phi' \)-quantile for \( \phi' = [\phi - \varepsilon, \phi + \varepsilon] \)

- .01-approximate medians are .49- and .51-quantiles (and items in between)
Approximate Median & Quantiles

How to define an approximate median?

\( \phi \)-quantile = \( \lceil \phi \cdot N \rceil \)-th smallest element (\( \phi \in [0, 1] \))

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, \ldots, .99-quantiles

\( \varepsilon \)-approximate \( \phi \)-quantile = any \( \phi' \)-quantile for \( \phi' = [\phi - \varepsilon, \phi + \varepsilon] \)

- .01-approximate medians are .49- and .51-quantiles (and items in between)

\( \varepsilon \)-approximate selection:

- query \( k \)-th smallest \( \rightarrow \) return \( k' \)-th smallest for \( k' = k \pm \varepsilon N \)
Approximate Median & Quantiles

How to define an approximate median?

\( \phi \)-quantile = \( \lceil \phi \cdot N \rceil \)-th smallest element (\( \phi \in [0, 1] \))

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, \ldots, .99-quantiles

\( \varepsilon \)-approximate \( \phi \)-quantile = any \( \phi' \)-quantile for \( \phi' = [\phi - \varepsilon, \phi + \varepsilon] \)

- .01-approximate medians are .49- and .51-quantiles (and items in between)

\( \varepsilon \)-approximate selection:

- query \( k \)-th smallest \( \Rightarrow \) return \( k' \)-th smallest for \( k' = k \pm \varepsilon N \)

Offline summary: sort data & select \( \sim \frac{1}{2\varepsilon} \) items
\(\varepsilon\)-Approximate Quantile Summaries

Data structure with two operations:

- \textbf{UPDATE}(x): \(x = \) new item from the stream
**ε-Approximate Quantile Summaries**

Data structure with two operations:

- **UPDATE(x):** \( x = \text{new item from the stream} \)
- **QUANTILE_QUERY(\( \phi \)):** For \( \phi \in [0, 1] \), return \( \varepsilon \)-approximate \( \phi \)-quantile
\( \varepsilon \)-Approximate Quantile Summaries

Data structure with two operations:

- **UPDATE**\( (x) \): \( x = \) new item from the stream
- **QUANTILE_QUERY**\( (\phi) \): For \( \phi \in [0, 1] \), return \( \varepsilon \)-approximate \( \phi \)-quantile

Additional operations:

- **RANK_QUERY**\( (x) \):
  - For item \( x \), determine its rank = position in the ordering of the input
**$\varepsilon$-Approximate Quantile Summaries**

Data structure with two operations:

- **Update($x$)**: $x =$ new item from the stream
- **Quantile Query($\phi$)**: For $\phi \in [0, 1]$, return $\varepsilon$-approximate $\phi$-quantile

Additional operations:

- **Rank Query($x$)**:
  - For item $x$, determine its rank = position in the ordering of the input
- Merge of two quantile summaries
  - Preserve space bounds, while maintaining accuracy
\( \varepsilon \)-Approximate Quantile Summaries

Data structure with two operations:

- **UPDATE**\( (x) \): \( x \) = new item from the stream
- **QUANTILE_QUERY**\( (\phi) \): For \( \phi \in [0, 1] \), return \( \varepsilon \)-approximate \( \phi \)-quantile

Additional operations:

- **RANK_QUERY**\( (x) \):
  - For item \( x \), determine its rank = position in the ordering of the input
- Merge of two quantile summaries
  - Preserve space bounds, while maintaining accuracy

Quantile summaries \( \rightarrow \) streaming algorithms for:

- Approximating distributions
- Equi-depth histograms
- Streaming Bin Packing [Cormode & V. '20]
\(\varepsilon\)-Approximate Quantile Summaries

Data structure with two operations:

- **Update**\((x)\): \(x = \text{new item from the stream}\)
- **Quantile Query**\((\phi)\): For \(\phi \in [0, 1]\), return \(\varepsilon\)-approximate \(\phi\)-quantile

Additional operations:

- **Rank Query**\((x)\):
  - For item \(x\), determine its rank = position in the ordering of the input
- Merge of two quantile summaries
  - Preserve space bounds, while maintaining accuracy

Quantile summaries \(\rightarrow\) streaming algorithms for:

- Approximating distributions
- Equi-depth histograms
- Streaming Bin Packing [Cormode & V. '20]

Bottom line: Finding \(\varepsilon\)-approximate median in data streams
Approximate Median & Quantiles: Streaming Algorithms

State-of-the-art results

space $\sim$ # of stored items
Approximate Median & Quantiles: Streaming Algorithms

State-of-the-art results

\[ \text{space} \sim \# \text{ of stored items} \]

- \( O \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \right) \) – deterministic comparison-based [Greenwald & Khanna ’01]
  maintains a subset of items + bounds on their ranks


Pavel Veselý
Tight Lower Bound for Quantile Summaries
Approximate Median & Quantiles: Streaming Algorithms

State-of-the-art results

space $\sim$ # of stored items

- $O\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ – deterministic comparison-based [Greenwald & Khanna '01]
  maintains a subset of items + bounds on their ranks

- $O\left(\frac{1}{\varepsilon} \cdot \log M\right)$ – deterministic for integers $\{1, \ldots, M\}$ [Shrivastava et al. '04]
  not for floats or strings

Many more papers: [Munro & Paterson '80, Manku et al. '98, Manku et al. '99, Hung & Ting '10, Agarwal et al. '12, Wang et al. '13, Felber & Ostrovsky '15, ...]
Approximate Median & Quantiles: Streaming Algorithms

State-of-the-art results

\[ \text{space } \sim \# \text{ of stored items} \]

\[ \mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right) \] – deterministic comparison-based \cite{GreenwaldKhanna}

\[ \text{maintains a subset of items + bounds on their ranks} \]

\[ \mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log M\right) \] – deterministic for integers \( \{1, \ldots, M\} \) \cite{Shrivastavaetal}

\[ \text{not for floats or strings} \]

\[ \mathcal{O}\left(\frac{1}{\varepsilon}\right) \] – randomized \cite{Karninetal}

\[ \text{const. probability of violating } \pm \varepsilon N \text{ error guarantee} \]
Approximate Median & Quantiles: Streaming Algorithms

State-of-the-art results

space $\sim$ # of stored items

- $O\left(\frac{1}{\varepsilon} \log \varepsilon N\right)$ – deterministic comparison-based [Greenwald & Khanna '01]
  
  maintains a subset of items + bounds on their ranks

- $O\left(\frac{1}{\varepsilon} \log M\right)$ – deterministic for integers \(\{1, \ldots, M\}\) [Shrivastava et al. '04]
  
  not for floats or strings

- $O\left(\frac{1}{\varepsilon}\right)$ – randomized [Karnin et al. '16]
  
  const. probability of violating $\pm \varepsilon N$ error guarantee

Many more papers: [Munro & Paterson '80, Manku et al. '98, Manku et al. '99]

[Hung & Ting '10, Agarwal et al. '12, Wang et al. '13, Felber & Ostrovsky '15, ...]
Approx. Median & Quantiles: Is There a “Perfect” Algorithm?

What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic
- constant space for fixed $\varepsilon$
- ideally $O\left(\frac{1}{\varepsilon}\right)$; or e.g. $O\left(\frac{1}{\varepsilon^2}\right)$
- no additional knowledge about items
- comparison-based

Theorem (Cormode, V. ’20)
There is no perfect streaming algorithm for $\varepsilon$-approximate median

- Optimal space lower bound $\Omega\left(\frac{1}{\varepsilon}\cdot \log \varepsilon N\right)$
- Matches the result in [Greenwald & Khanna ’01]
Approx. Median & Quantiles: Is There a “Perfect” Algorithm?

What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic

---

Theorem (Cormode, V. '20)

There is no perfect streaming algorithm for $\varepsilon$-approximate median

- Optimal space lower bound $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$
- Matches the result in [Greenwald & Khanna '01]
Approx. Median & Quantiles: Is There a “Perfect” Algorithm?

What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic
- constant space for fixed $\varepsilon$
  - ideally $O\left(\frac{1}{\varepsilon}\right)$; or e.g. $O\left(\frac{1}{\varepsilon^2}\right)$
What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic
- constant space for fixed $\varepsilon$
  - ideally $O\left(\frac{1}{\varepsilon}\right)$; or e.g. $O\left(\frac{1}{\varepsilon^2}\right)$
- no additional knowledge about items
  - comparison-based
What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic
- constant space for fixed $\varepsilon$
  - ideally $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$; or e.g. $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$
- no additional knowledge about items
  - comparison-based

**Theorem** (Cormode, V. ’20)

There is **no** perfect streaming algorithm for $\varepsilon$-approximate median
Approx. Median & Quantiles: Is There a “Perfect” Algorithm?

What would be a “perfect” streaming algorithm?

- finds $\varepsilon$-approximate median
- deterministic
- constant space for fixed $\varepsilon$
  - ideally $O\left(\frac{1}{\varepsilon}\right)$; or e.g. $O\left(\frac{1}{\varepsilon^2}\right)$
- no additional knowledge about items
  - comparison-based

**Theorem** (Cormode, V. ’20)

There is no perfect streaming algorithm for $\varepsilon$-approximate median

- Optimal space lower bound $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$
  - Matches the result in [Greenwald & Khanna ’01]
Approx. Median & Quantiles: Lower Bound Idea

Comparison-based algorithm

⇒ cannot compare with items deleted from the memory
Approx. Median & Quantiles: Lower Bound Idea

Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

How does 30 compare to discarded items between 10 and 50?

Idea: Introduce uncertainty

• too high uncertainty ⇒ not accurate enough answers

• need to show: low uncertainty ⇒ many items stored ⇒ large space needed

→ recursive construction of worst-case stream
→ lower bound $\Omega \left( \epsilon \cdot \log \epsilon N \right)$
Approx. Median & Quantiles: Lower Bound Idea

Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

\[ \leq \]

\[ 10 \quad 50 \quad \mathbb{R} \]

How does 30 compare to discarded items between 10 and 50?

Idea: Introduce uncertainty

• too high uncertainty ⇒ not accurate-enough answers

• need to show: low uncertainty ⇒ many items stored ⇒ large space needed

→ recursive construction of worst-case stream → lower bound \( \Omega \left( 1 \cdot \log \varepsilon \cdot N \right) \)
Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

new item: 30

How does 30 compare to discarded items between 10 and 50?
Approx. Median & Quantiles: Lower Bound Idea

Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

How does 30 compare to discarded items between 10 and 50?

Idea: Introduce uncertainty

- too high uncertainty ⇒ not accurate-enough answers
Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

new item: 30

How does 30 compare to discarded items between 10 and 50?

Idea: Introduce uncertainty

• too high uncertainty ⇒ not accurate-enough answers
• need to show: low uncertainty ⇒ many items stored ⇒ large space needed
Approx. Median & Quantiles: Lower Bound Idea

Comparison-based algorithm

⇒ cannot compare with items deleted from the memory

How does 30 compare to discarded items between 10 and 50?

Idea: Introduce uncertainty

• too high uncertainty ⇒ not accurate-enough answers
• need to show: low uncertainty ⇒ many items stored ⇒ large space needed

→ recursive construction of worst-case stream → lower bound \( \Omega \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \right) \)
Problem solved: ≤

- Deterministic algorithms: space $\Theta \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \right)$ optimal  
  [Greenwald & Khanna '01]
  [Cormode, V. '20]

- Randomized algorithms: space $\Theta \left( \frac{1}{\varepsilon} \right)$ optimal  
  (const. probability of too high error)  
  [Karnin et al. '16]
Approximating Median & Quantiles: Conclusions & Open Problems

Problem solved:  
- Deterministic algorithms: space $\Theta\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ optimal  
  [Greenwald & Khanna '01]  
  [Cormode, V. '20]

- Randomized algorithms: space $\Theta\left(\frac{1}{\varepsilon}\right)$ optimal  
  (const. probability of too high error)  
  [Karnin et al. '16]

Future work:  
- Figure out constant factors
Approximating Median & Quantiles: Conclusions & Open Problems

Problem solved:

- Deterministic algorithms: space $\Theta \left( \frac{1}{\epsilon} \cdot \log \epsilon N \right)$ optimal  \[\text{[Greenwald & Khanna '01]}\]
  \[\text{[Cormode, V. '20]}\]

- Randomized algorithms: space $\Theta \left( \frac{1}{\epsilon} \right)$ optimal (const. probability of too high error) \[\text{[Karnin et al. '16]}\]

Future work:

- Figure out constant factors
- Randomized algorithm with good expected space, but guaranteed $\pm \epsilon N$ error
Approximating Median & Quantiles: Conclusions & Open Problems

Problem solved:

- Deterministic algorithms: space $\Theta\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ optimal [Greenwald & Khanna ’01]
  
- Randomized algorithms: space $\Theta\left(\frac{1}{\varepsilon}\right)$ optimal (const. probability of too high error) [Karnin et al. ’16]

Future work:

- Figure out constant factors
- Randomized algorithm with good expected space, but guaranteed $\pm \varepsilon N$ error
- A non-trivial lower bound for integers $\{1, \ldots, M\}$?
  - Or can we do better than $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log M\right)$?
Problem solved:

- Deterministic algorithms: space $\Theta \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \right)$ optimal [Greenwald & Khanna '01] [Cormode, V. '20]

- Randomized algorithms: space $\Theta \left( \frac{1}{\varepsilon} \right)$ optimal (const. probability of too high error) [Karnin et al. '16]

Future work:

- Figure out constant factors
- Randomized algorithm with good expected space, but guaranteed $\pm \varepsilon N$ error
- A non-trivial lower bound for integers $\{1, \ldots, M\}$?
  - Or can we do better than $O \left( \frac{1}{\varepsilon} \cdot \log M \right)$?
- Dynamic streams w/ insertions and deletions of items
Thank You!