Streaming Quantiles

Edo Liberty
Principal Scientist
Amazon Web Services
Amazon Kinesis Analytics

Get actionable insights from streaming data in real-time.
Streaming Quantiles

Manku, Rajagopalan, Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets.
Munro, Paterson. Selection and sorting with limited storage.
Greenwald, Khanna. Space-efficient online computation of quantile summaries.
Wang, Luo, Yi, Cormode. Quantiles over data streams: An experimental study.
Greenwald, Khanna. Quantiles and equidepth histograms over streams.
Agarwal, Cormode, Huang, Phillips, Wei, Yi. Mergeable summaries.
Felber, Ostrovsky. A randomized online quantile summary in $O((1/\varepsilon) \log(1/\varepsilon))$ words.
Lang, Karnin, Liberty, Optimal Quantile Approximation in Streams.
Problem Definition

Create a sketch for $R'$ such that $|R'(x) - R(x)| \leq \varepsilon n$. 

$R(\square) = 0.6 \cdot n$
Solutions

• Uniform sampling
  ✔ Fast and simple ✔ Fully mergeable ✗ Space $\tilde{O}(1/\varepsilon^2)$

• Greenwald Khanna (GK) sketch
  ✗ Slow, complex ✗ Not mergeable ✔ Space $\log(n)/\varepsilon$

• Felber-Ostrovsky, combines sampling and GK (2015)
  ✗ Slow, complex ✗ Not mergeable ✔ Space $\log(1/\varepsilon)/\varepsilon$

Previously conjectured space optimal for all algorithms.
$\log(1/\varepsilon)/\varepsilon$ lower bound for deterministic algorithms by Hung and Ting 2010.
Solutions cont’

- Uniform sampling
  - ✔ Fast, simple
  - ✔ Fully mergeable
  - ✗ Space $\tilde{O}(1/\varepsilon^2)$

- Greenwald Khanna (GK) sketch
  - ✗ Slow, complex
  - ✗ Not mergeable
  - ✔ Space $\log(n)/\varepsilon$

- Felber-Ostrovsky, combines sampling and GK (2015)
  - ✗ Slow, complex
  - ✗ Not mergeable
  - ✔ Space $\log(1/\varepsilon)/\varepsilon$

- Manku-Rajagopalan-Lindsay (MRL)
  - ✔ Fast simple
  - ✔ Fully mergeable
  - ✗ Space $\log^2(n)/\varepsilon$

- Agarwal, Cormode, Huang, Phillips, Wei, Yi
  - ✗ Fast, complex
  - ✔ Fully mergeable
  - ✔ Space $\log^{3/2}(1/\varepsilon)/\varepsilon$

Buffer based solutions
The basic buffer idea

Buffer of size \( k \)
The basic buffer idea

Stores $k$ stream entries

```
3
0
7
4
1
5
```
The basic buffer idea

The buffer sorts $k$ stream entries
The basic buffer idea

Deletes every other item
The basic buffer idea

And outputs the rest with double the weight

5 3 0
The basic buffer idea

\[ R(x) = 2 \]

\[ R' (x) = 2 \]

\[ R(x) = 5 \]

\[ R' (x) = 6 \]

\[ R' (x) = 2 \]

\[ R' (x) = 4 \]
The basic buffer idea

Repeat $n/k$ time until the end of the stream

$|R'(x) - R(x)| < n/k$
Manku-Rajagopalan-Lindsay (MRL) sketch

\[ H = \log_2(n) \text{ Buffers of size } k \]

\[ |R'(x) - R(x)| \leq \frac{n}{k} \cdot \log_2(n) \]
Manku-Rajagopalan-Lindsay (MRL) sketch

If we set $k = \log_2(n)/\varepsilon$ we get $|R'(x) - R(x)| \leq \varepsilon n$

while maintaining at most $H \cdot k \leq \log_2^2(n)/\varepsilon$ stream items.

Manku-Rajagopalan-Lindsay (MRL) sketch

- ✔ Fast, Simple
- ✔ Fully mergeable
- ✗ Space
Agarwal, Cormode, Huang, Phillips, Wei, Yi (1)

\[
\log(1/\varepsilon) \text{ Buffers of size } k
\]

start sampling after \(O(1/\varepsilon^2)\) items

\[
\frac{\log^2(1/\varepsilon)}{\varepsilon}
\]

Reduces space usage to \(\frac{\log^2(1/\varepsilon)}{\varepsilon}\) items from the stream.
$R(x) = 1$

\begin{align*}
R'(x) &= 2 \\
5 &\quad 7
\end{align*}

\begin{align*}
R'(x) &= 0 \\
5 &\quad 7
\end{align*}

$E[R'(x)] = R(x)$

$R'(x)$ is a random variable now and

Reduces space usage to $\log^{3/2}(1/\varepsilon)/\varepsilon$ items from the stream.
Recap

- **Uniform sampling**
  - ✔ Fast, simple
  - ✔ Fully mergeable
  - ✗ Space $\tilde{O}(1/\epsilon^2)$

- **Greenwald Khanna (GK) sketch**
  - ✗ Slow, complex
  - ✗ Not mergeable
  - ✔ Space $\log(n)/\epsilon$

- **Felber-Ostrovsky, combines sampling and GK (2015)**
  - ✗ Slow, complex
  - ✗ Not mergeable
  - ✔ Space $\log(1/\epsilon)/\epsilon$

- **Manku-Rajagopalan-Lindsay (MRL)**
  - ✔ Fast, simple
  - ✔ Fully mergeable
  - ✔ Space $\log^2(n)/\epsilon$

- **Agarwal, Cormode, Huang, Phillips, Wei, Yi**
  - ✗ Fast, complex
  - ✔ Fully mergeable
  - ✔ Space $\log^{3/2}(1/\epsilon)/\epsilon$
Our goal

- Uniform sampling
  ✔ Fast, simple ✔ Fully mergeable ❌ Space $\tilde{O}(1/\varepsilon^2)$

- Greenwald Khanna (GK) sketch
  ❌ Slow, complex ❌ Not mergeable ✔ Space $\log(n)/\varepsilon$

- Felber-Ostrovsky, combines sampling and GK (2015)
  ❌ Slow, complex ❌ Not mergeable ✔ Space $\log(1/\varepsilon)/\varepsilon$

- Manku-Rajagopalan-Lindsay (MRL)
  ✔ Fast, simple ✔ Fully mergeable ❌ Space $\log^2(n)/\varepsilon$

- Agarwal, Cormode, Huang, Phillips, Wei, Yi
  ❌ Fast, complex ✔ Fully mergeable ✔ Space $\log^{3/2}(1/\varepsilon)/\varepsilon$

- Karnin, Lang, Liberty
  ✔ Fast, simple ✔ Fully mergeable ✔ Space $\log(1/\varepsilon)/\varepsilon$
Observation

The first buffers contribute very little to the error. They are “too good”.

\[ h = H \quad \cdots \quad h = 2 \quad h = 1 \]

\[ w_h = 2^{h-1} \]

Weight of items in the level

\[ m_h = 2^{H-h-1} \]

Number of compactions
Idea

Let buffers shrink at-most-exponentially

Let buffers shrink at-most-exponentially

\[ k_h \geq k_c^H - h \]

\[ w_h = 2^{h-1} \]

\[ H \leq \log(n/ck) + 2 \]

\[ m_h \leq (2/c)^{H-h-1} \]

Number of compactions

h = H
Analysis

\( R(h, x) \) the rank of \( x \) among

1. The items yielded by the compactor at height \( h \)
2. All the items stored in the compactors of heights \( h' \leq h \)

Claim, for \( C = c^2(2c - 1) \)

\[
\Pr \left[ |R(x, H') - R(x)| \geq \varepsilon n \right] \leq 2 \exp \left( -C \varepsilon^2 k^2 2^{2(H-H')} \right)
\]

Proof

Use Hoeffding’s inequality on \( \sum_{h=1}^{H} [R(x, h) - R(x, h - 1)] \)
Solution 1

Set $c = 2/3$ and $k_h = \lceil kc^{H-h} \rceil + 1$

- Karnin, Lang, Liberty (1)
  - Fast, simple
  - Fully mergeable
  - Space $\sqrt{\log(1/\varepsilon)}/\varepsilon$

Better than previously conjectured optimal!

- $\log(n)$ exponentially decreasing capacity buffers
- Sampler replaces all buffers of size 2
Solution 2 (KLL + MRL)

Set $c = 2/3$ and $k_h = \left\lfloor kc^H - h \right\rfloor + 1$ except that the top $\log \log(1/\varepsilon)$ buffers all have capacity $k$.

- Karnin, Lang, Liberty (2)
  - ✅ Fast, simple
  - ✅ Fully mergeable
  - ✅ Space $\log^2 \log(1/\varepsilon) / \varepsilon$

Buffers of capacity $k$

$\log \log(1/\varepsilon)$

$\log(n)$ exponentially decreasing capacity buffers

Sampler replaces all buffers of size 2
Solution 3 (KLL + GK)

Set $c = 2/3$ and $k_h = \lceil kc^{H-h} \rceil + 1$ replace the top $\log \log (1/\varepsilon)$ with a GK sketch. $k$

- Karnin, Lang, Liberty (3)
- Fast, simple ✗
- Fully mergeable ✗
- Space $\log \log (1/\varepsilon) / \varepsilon$

GK sketch replaces top $\log \log (1/\varepsilon)$ levels

$log(n)$ exponentially decreasing capacity buffers

Sampler replaces all buffers of size 2
Count Distinct (Demo Only)


**sketches-core**
Core Sketch Library.

- Java
- 415
- 119
- Updated a day ago

---

[Yahoo!](https://www.yahoo.com)  [Druid](https://druid.io)  [Splice Machine](https://splice-machine.com)  [Amazon Web Services](https://aws.amazon.com)
Assume you need to estimate the distribution of numbers in a file

```
$ head data.csv
0
1
0
0
3
0
2
3
7
3
3
2
```

In this one, row $i$ tasks a value from $[0,i]$ uniformly at random.
Some stats: there are 10,000,000 such numbers in this ~76Mb file.

```
$ time wc -lc data.csv
 10000000 76046666 data.csv
```

real 0m0.101s
user 0m0.072s
sys 0m0.021s

Reading the file take ~1/10 seconds. We don’t foresee IO being an issue.
In python it looks like this:

```python
$ cat quantiles.py
import sys
ints = sorted([int(x) for x in sys.stdin])
for i in range(0, len(ints), int(len(ints)/100)):
    print(str(ints[i]))
```

$ time cat data.csv | python quantiles.py > /dev/null

real 0m13.406s
user 0m12.937s
sys 0m0.407s
This is the way to do this with the sketching library

```bash
$ time cat data.csv | sketch rank
```

```bash
$ time cat data.csv | sketch rank > /dev/null
real 0m1.495s
user 0m1.878s
sys 0m0.141s
```

Too fast to use the system monitor UI...

It uses ~ 4k of memory!
exact and approximate quantiles

exact vs approximate quantiles

- approximate quantiles
- exact quantiles
Some experimental results

Lazy KLL versus (Sketch Library and Two Variants)

Error

Sketch Library
Variant 1
Variant 2
Lazy KLL

Number of Items in Randomly Permuted Stream

Space Used For Storing Samples

Sketch Library
Variant 1
Variant 2
Lazy KLL

Number of Items in Randomly Permuted Stream
Thank you!