A Randomized Online Quantile Summary in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Words

David Felber and Rafail Ostrovsky

UCLA Computer Science Department
Los Angeles, CA, USA
{dvfelber,rafail}@cs.ucla.edu

Abstract

A quantile summary is a data structure that approximates to ε-relative error the order statistics of a much larger underlying dataset.

In this paper we develop a randomized online quantile summary for the cash register data input model and comparison data domain model that uses $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words of memory. This improves upon the previous best upper bound of $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ by Agarwal et al. [1]. Further, by a lower bound of Hung and Ting [4] no deterministic summary for the comparison model can outperform our randomized summary in terms of space complexity. Lastly, our summary has the nice property that $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words suffice to ensure that the success probability is $1-e^{-\mathrm{poly}(1/\varepsilon)}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

Keywords and phrases order statistics, data stream, streaming algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.775

1 Introduction

A quantile summary S is a fundamental data structure that summarizes an underlying dataset X of size n, in space much less than n. Given a query ϕ, S returns a sample y of X such that the rank of y in X is (probably) approximately ϕn. Quantile summaries are used in sensor networks to aggregate data in an energy-efficient manner and in database query optimizers to generate query execution plans.

Quantile summaries have been developed for a variety of different models and metrics. The data input model we consider is the standard online cash register streaming model, in which a new item is added to the dataset at each new timestep, and the total number of items is not known until the end. The data domain model we consider is the comparison model, in which stream items come from an arbitrary ordered domain (and specifically, not necessarily from the integers).

Formally, our quantile summary problem is defined over a totally ordered domain $\mathcal{D}$ and by an error parameter $\varepsilon \le 1/2$. There is a dataset X that is initially empty. Time occurs in discrete steps. In timestep t, stream item x_t arrives and is then processed, and then any quantile queries ϕ in that step are received and processed. To be definite, we pick the first timestep to be 1. We write X_t or $X(t)$ for the t-item prefix stream $x_1 \dots x_t$ of X. The goal is to maintain at all times t a summary S_t of the dataset X_t that, given any query ϕ in $(0,1]$, can return a sample $y = y(\phi)$ so that $|R(y,X_t) - \phi t| \le \varepsilon t$, where $R(a,Z)$ is the *rank of item a in set Z*, defined as $|\{z \in Z : z \le a\}|$. For randomized summaries, we only require that $\forall t \forall \phi,\ P(|R(y,X_t) - \phi t| \le \varepsilon t) \ge 2/3$; that is, y's rank is only probably close to ϕt, not definitely close. In fact, it will be easier to deal with the rank directly, so we define $\rho = \phi t$ and use that in what follows.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 775–785

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Previous work

The two most directly relevant pieces of prior work ([1, 2] and [6]) are randomized online quantile summaries for the cash register/comparison model. Aside from oblivious sampling algorithms (which require storing $\Omega(1/\varepsilon^2)$ samples) the only other such work of which we are aware is an approach by Wang, Luo, Yi, and Cormode [11] that combines the methods of [1, 2] and [6] into a hybrid with the same space bound as [1, 2].

The newer of the two is that of Agarwal, Cormode, Huang, Phillips, Wei, and Yi [1, 2]. Among other results, Agarwal et al. develop a randomized online quantile summary for the cash register/comparison model that uses $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ words of memory. This summary has the nice property that any two such summaries can be combined to form a summary of the combined underlying dataset without loss of accuracy or increase in size.

The earlier such summary is that of Manku, Rajagopalan, and Lindsay [6], which uses $O(\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})$ space. At a high level, their algorithm downsamples the input stream in a non-uniform way and feeds the downsampled stream into a deterministic summary, while periodically adjusting the downsampling rate.

We note here for those familiar with the result of Manku et al. that, while our algorithm at a high level may appear similar, there are important differences. We defer a discussion of similarities and differences to Section 4 after the presentation of our algorithm in Section 3.

For the comparison model, the best deterministic online summary to date is the (GK) summary of Greenwald and Khanna [3], which uses $O(\frac{1}{\varepsilon}\log\varepsilon n)$ space. This improved upon a deterministic (MRL) summary of Manku, Rajagopalan, and Lindsay [5] and a summary implied by Munro and Paterson [7], which use $O(\frac{1}{\varepsilon}\log^2\varepsilon n)$ space.

A more restrictive domain model than the comparison model is the bounded universe model, in which elements are drawn from the integers $\{1,\ldots,u\}$. For this model there is a deterministic online summary by Shrivastava, Buragohain, Agrawal, and Suri [9] that uses $O(\frac{\log u}{\varepsilon})$ space.

Not much exists in the way of lower bounds for this problem. There is a simple lower bound of $\Omega(1/\varepsilon)$ which intuitively comes from the fact that no one sample can satisfy more than $2\varepsilon n$ different rank queries. For the comparison model, Hung and Ting [4] developed a deterministic $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ lower bound. Whether this bound can be extended to hold for our weaker probabilistic guarantee, and whether our algorithm can be modified to satisfy the stronger deterministic guarantee, are both open questions.

1.2 Our contributions

In the next section we describe a simple $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ streaming summary that is online except that it requires n to be given up front and that it is unable to process queries until it has seen a constant fraction of the input stream. This simple summary is not new (it is mentioned in Wang et al. [11], for example) but the discussion provides exposition for Section‘3, in which we develop this summary into a fully online summary with the same asymptotic space complexity that can answer queries at any point in time. We close in Section 4 by examining the similarities and differences between our summary and previous work and discuss a design approach for similar streaming problems.

2 A simple streaming summary

Before we describe the algorithm we must first describe its two main components in a bit more detail than was used in the introduction. The two components are Bernoulli sampling and the GK summary [3].

2.1 Bernoulli sampling

Bernoulli sampling downsamples a stream X of size n to a sample stream S by choosing to include each next item into S with independent probability m/n. (As stated this requires knowing the size of X in advance.) At the end of processing X, the expected size of S is m, and the expected rank of any sample y in S is $E(R(y,S)) = \frac{m}{n}R(y,X)$. In fact, for any times $t \leq n$ and partial streams X_t and S_t, where S_t is the sample stream of X_t, we have $E(|S_t|) = mt/n$ and $E(R(y,S_t)) = \frac{m}{n}R(y,X_t)$. To generate an estimate for $R(y,X_t)$ from S_t we use $\hat{R}(y,X_t) = \frac{n}{m}R(y,S_t)$. The following theorem bounds the probability that S is very large or that $\hat{R}(y,X_t)$ is very far from $R(y,X_t)$. A generalization of this theorem is due Vapnik and Chervonenkis [10]; the proof of this special case is a simple known application of Chernoff bounds.

▶ **Theorem 1.** *For all times $t \geq n/64$,*

$$P(|S_t| > 2tm/n) < \exp(-m/192)$$

Further, for all times $t \geq n/64$ and items y,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) < 2\exp(-\varepsilon^2 m/12288)$$

Proof. For the first part,

$$P(|S_t| > 2tm/n) < \exp(-tm/3n) < \exp(-m/192)$$

since $t \geq n/64$. For the second part,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) = P(|R(y,S_t) - E(R(y,S_t))| > \varepsilon tm/8n)$$

The Chernoff bound is

$$P(|R(y,S_t) - E(R(y,S_t))| > \delta E(R(y,S_t))) < 2\exp(-\min\{\delta,\delta^2\}E(R(y,S_t))/3)$$

Here, $\delta = \varepsilon t/8R(y,S_t)$, so

$$P < 2\exp(-\varepsilon^2 t^2 m/192nE(R(y,S_t))) \leq 2\exp(-\varepsilon^2 m/12288)$$

finishing the proof. ◀

This means that, given any $1 \leq \rho \leq t$, if we choose to return the sample $y \in S_t$ with $R(y,S_t) = \rho m/n$, then $R(y,X_t)$ is likely to be close to ρ, as long as m is $\Omega(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$.

2.2 GK summary

The GK summary is a deterministic summary that can answer queries to relative error over any portion of the received stream. Let G_t be the summary after inserting the first t items X_t from stream X into G. Greenwald and Khanna guarantee in [3] that with only $O(\frac{1}{\varepsilon}\log(\varepsilon t))$ words, given any $1 \leq \rho \leq t$, G_t can return a sample $y \in X_t$ so that $|R(y,X_t) - \rho| \leq \varepsilon t/8$. We call this the *GK guarantee*.

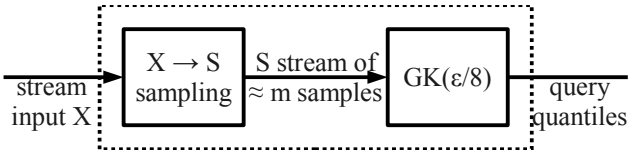

Figure 1 The big picture.

2.3 A simple streaming summary

We combine Bernoulli sampling with the GK summary by downsampling the input data stream X to a sample stream S and then feeding S into a GK summary G. It looks like in Figure 1.

The key reason this gives us a small summary is that we never need to store S; each time we sample an item into S we immediately feed it into G. Therefore, we only use as much space as $G(S(X_t))$ uses. In particular, for $m = O(\text{poly}(1/\varepsilon))$, we use only $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words. To answer a query ρ for X_t, we scale ρ by m/n, ask $G(S(X_t))$ for that, and return the resulting sample y.

We formalize this intuition in the following lemma, which combines the ideas in the proof of Theorem 1 with the GK guarantee to yield approximation and correctness guarantees.

▶ **Lemma 2.** *Fix some time $t \geq n/64$ and some rank $\rho \leq t$, and consider querying $G(S(X_t))$ with $q = \min\{\rho m/n, |S|\}$, obtaining y as the result.*

Say that $S = S(X_t)$ is good *if $||S| - mt/n| \leq \varepsilon mt/8n$ and if none of the first $\leq mt/n$ samples z in S has $|R(z,S) - \frac{m}{n}R(z,X_t)| > \varepsilon mt/8n$.*

If S is good then $|R(y,X_t) - \rho| \leq \varepsilon t/2$.

Further, if $m \geq \frac{400000\ln 1/\varepsilon}{\varepsilon^3}$ then $P(S$ is not good$) \leq \varepsilon^3 e^{-1/\varepsilon}/8$.

Proof. First, by the GK guarantee, $G(S)$ returns some item y with $|R(y,S) - q| \leq \varepsilon t/8$. If S is good, then $|q - \rho m/n| \leq \varepsilon mt/8n$, and also $|R(y,S) - \frac{m}{n}R(y,X_t)| \leq \varepsilon mt/8n$. By the triangle inequality, $|\frac{m}{n}R(y,X_t) - \rho m/n| \leq 3\varepsilon mt/8n$. Equivalently, $|R(y,X_t) - \rho| \leq 3\varepsilon t/8$.

Now, following the proof of Theorem 1, we have that

$$P(||S_t| - mt/n| > \varepsilon mt/8n) \;<\; 2\exp(-\varepsilon^2 m/12288)$$

and also for each of the first $\leq m$ samples z that

$$P(|R(z,S) - \tfrac{m}{n}R(z,X_t)| > \varepsilon mt/8n) \;<\; 2\exp(-\varepsilon^2 m/12288)$$

By the union bound, $P(S$ is not good$) \leq 4m\exp(-\varepsilon^2 m/12288)$. Choosing $m \geq \frac{400000\ln 1/\varepsilon}{\varepsilon^3}$ suffices to bound this quantity by $\varepsilon^3 e^{-1/\varepsilon}/8$. ◀

2.4 Caveats

There are two serious issues with this summary. The first is that it requires us to know the value of n in advance to perform the sampling. Also, as a byproduct of the sampling, we can only obtain approximation guarantees after we have seen at least 1/64 (or at least some constant fraction) of the items. This means that while the algorithm is sufficient for approximating order statistics over streams stored on disk, more is needed to get it to work for online streaming applications, in which (1) the stream size n is not known in advance, and (2) queries can be answered approximately at all times $t \leq n$ and not just when $t \geq n/64$.

Adapting this basic streaming summary idea to work online constitutes the next section and the bulk of our contribution. We start with a high-level overview of our online summary algorithm. In Section 3.1 we formally define an initial version of our algorithm whose expected size at any given time is $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words. In Section 3.2 we show that our algorithm guarantees that $\forall n \forall \rho,\ P(|R(y, X_n) - \rho| \leq \varepsilon n) \geq 1 - \exp(-1/\varepsilon)$. In Section 3.3 we discuss the slight modifications necessary to get a deterministic $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ space complexity, and also perform a time complexity analysis.

3 An online summary

Our algorithm works in *rows*, which are illustrated in figure 2. Row r is a summary of the first $2^r 32m$ stream items. Since we don't know how many items will actually be in the stream, we can't start all of these rows running at the outset. Therefore, we start each row $r \geq 1$ once we have seen $1/64$ of its total items. However, since we can't save these items for every row we start, we need to construct an approximation of this fraction of the stream, which we do by using the summary of the previous row, and join this approximating stream with the new items that arrive while the row is live. We then wait until the row has seen a full half of its items before we permit it to start answering queries; this dilutes the influence of approximating the $1/64$ of its input that we couldn't store.

Operation within a row is very much like the operation of our fixed-n streaming summary. We feed the joint approximate prefix + new item stream through a Bernoulli sampler to get a sample stream, which is then fed into a GK summary (which is stored). After row r has seen half of its items, its GK summary becomes the one used to answer quantile queries. When row $r+1$ has seen $1/64$ of *its* total items, row r generates an approximation of those items from its GK summary and feeds them as a stream into row $r+1$.

Row 0 is slightly different in order to bootstrap the algorithm. There is no join step since there is no previous row to join. Also, row 0 is active from the start. Lastly, we get rid of the sampling step so that we can answer queries over timesteps $1 \ldots m/2$.

After the first $32m$ items, row 0 is no longer needed, so we can clean up the space used by its GK summary. Similarly, after the first $2^r 32m$ items, row r is no longer needed. The upshot of this is that we never need storage for more than six rows at a time. Since each GK summary uses $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words, the six live GK summaries also only use $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words.

Our error analysis, on the other hand, will require us to look back as many as $\Theta(\log 1/\varepsilon)$ rows to ensure our approximation guarantee. We stress that we will not need to actually *store* these $\Theta(\log 1/\varepsilon)$ rows for our guarantee to hold; we will only need that they didn't have any bad events (as will be defined) when they *were* alive.

3.1 Algorithm description

Our algorithm works in rows. Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. For each row r we define several streams: A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r by querying it with ranks $\rho_1 \ldots \rho_{8/\varepsilon}$, where $\rho_q = q(\varepsilon/8)(m/32)$ for $r \geq 1$ and $\rho_q = q\varepsilon m/8$ for $r = 0$.

The prefix stream $A_r = X(2^{r-1}m)$ for row $r \geq 1$, importantly, is not directly received by row r. Instead, at the end of timestep $2^{r-1}m$, row $r-1$ generates Q_{r-1} and duplicates each of those $8/\varepsilon$ items $2^{r-1}\varepsilon m/8$ times to get the replacement prefix R_r, which is then immediately fed into row r before timestep $2^{r-1}m+1$ begins.

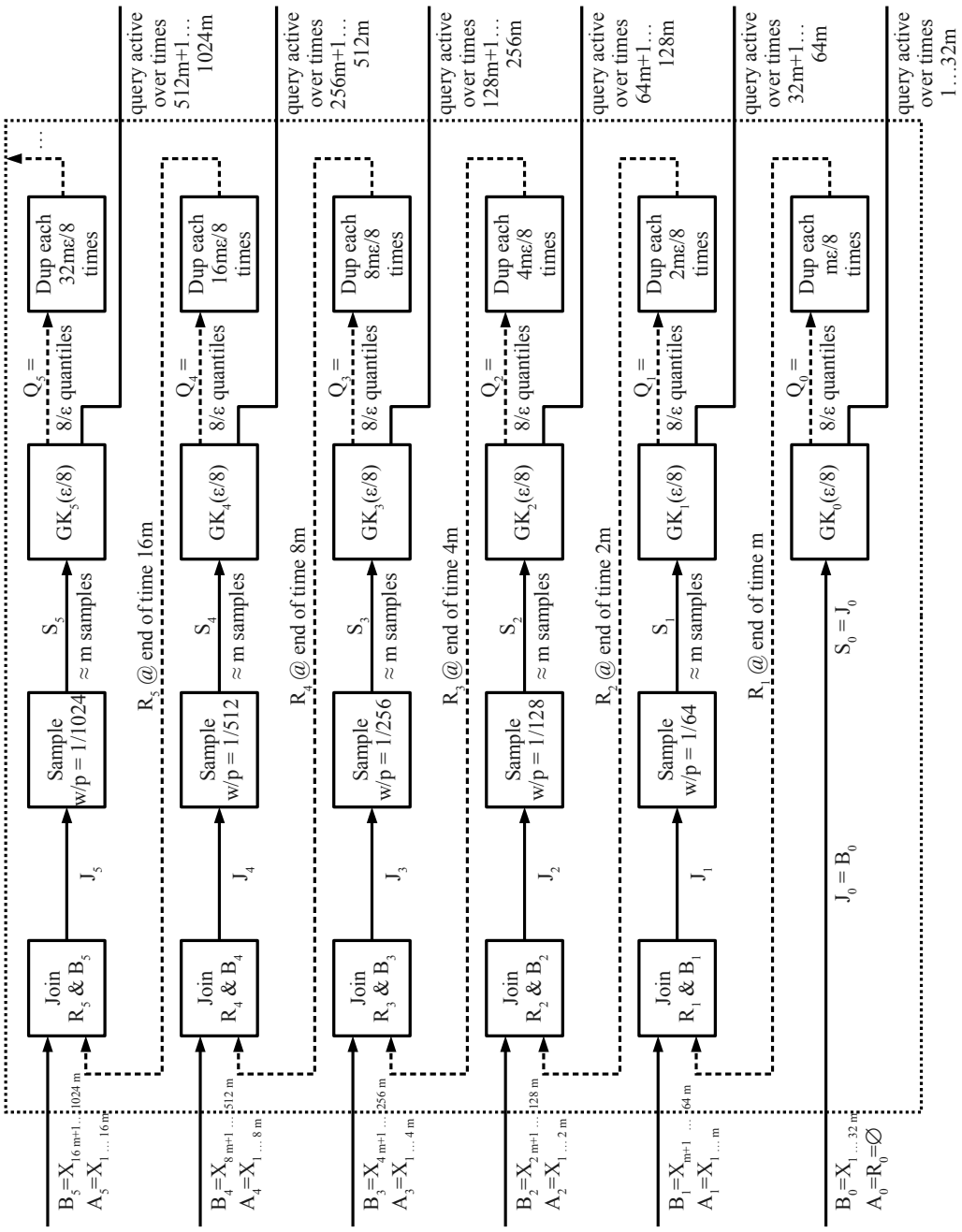

Figure 2 Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r at time $2^r m$ to get the replacement prefix R_{r+1}.

Each row can be *live* or not and *active* or not. Row 0 is live in timesteps $1 \dots 32m$ and row $r \geq 1$ is live in timesteps $2^{r-1}m+1 \dots 2^r 32m$. Live rows require space; once a row is no longer live we can free up the space it used. Row 0 is active in timesteps $1 \dots 32m$ and row $r \geq 1$ is active in timesteps $2^r 16m+1 \dots 2^r 32m$. This definition means that exactly one row $r(t)$ is active in any given timestep t. Any queries that are asked in timestep t are answered by $G_{r(t)}$. Given query ρ, we ask $G_{r(t)}$ for $\rho/2^{r(t)}32$ (if $r \geq 1$) or for ρ (if $r = 0$) and return the result.

At each timestep t, when item x_t arrives, it is fed as the next item in the suffix stream B_r for each live row r. B_r joined with R_r defines the joined input stream J_r. For $r \geq 1$, J_r is downsampled to the sample stream S_r by sampling each item independently with probability $1/2^r 32$. For row 0, no downsampling is performed, so $S_0 = J_0$. Lastly, S_r is fed into G_r.

Figure 2 shows the operation of and the communication between the first six rows. Solid arrows indicate continuous streams and dashed arrows indicate one-time messages. Algorithm 1 is a pseudocode listing of the algorithm.

```
Initially, allocate space for G_0. Mark row 0 as live and active.
for t = 1, 2, ... do
    foreach live row r ≥ 0 do
        with probability 1/2^r 32 do
            Insert x_t into G_r.
    if t = 2^{r-1}m for some r ≥ 1 then
        Allocate space for G_r. Mark row r as live.
        Query G_{r-1} with ρ_1 ... ρ_{8/ε} to get y_1 ... y_{8/ε}.
        for q = 1 ... 8/ε do
            for 1 ... 2^{r-1}εm/8 do
                with probability 1/2^r 32 do
                    Insert y_q into G_r.
    if t = 2^r 16m for some r ≥ 1 then
        Mark row r as active. Unmark row r−1 as active.
    if t = 2^r 32m for some r ≥ 0 then
        Unmark row r as live. Free space for G_r.
on query ρ do
    Let r = r(t) be the active row.
    Query G_r for rank ρ/2^r 32 (if r ≥ 1) or for rank ρ (if r = 0).
    Return the result.
```

Algorithm 1. Procedural listing of the algorithm in Section 3.1.

3.2 Error analysis

Define $C_r = x(2^r 32m+1), x(2^r 32m+2), \dots$ and Y_r to be R_r followed by B_r and then C_r. That is, Y_r is just the continuation of J_r for the entire length of the input stream.

Fix some time t. All of our claims will be relative to time t; that is, if we write S_r we mean $S_r(t)$. Our error analysis proceeds as follows. We start by proving that $R(y, Y_r)$ is a good approximation of $R(y, Y_{r-1})$ when certain conditions hold for S_{r-1}. By induction, this means that $R(y, Y_r)$ is a good approximation of $R(y, X = Y_0)$ when the conditions hold for all of $S_0 \dots S_{r-1}$, and actually it's enough for the conditions to hold for just $S_{r-\log 1/\varepsilon} \dots S_{r-1}$ to get a good approximation. Having proven this claim, we then prove that the result $y = y(\rho)$

of a query to our summary has $R(y, X)$ close to ρ. Lastly, we show that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that the conditions hold for $S_{r-\log 1/\varepsilon} \dots S_{r-1}$ with very high probability $(1 - e^{-1/\varepsilon})$.

▶ **Lemma 3.** *Let α_r be the event that $|S_r| > 2m$ and let β_r be the event that any of the first $\leq 2m$ samples z in S_r has $|2^r 32R(z, S_r) - R(z, Y_r)| > \varepsilon t/8$. Say that S_r is* good *if neither α_r nor β_r occur (or if $r = 0$).*

For all rows $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, and all for all items y, if S_{r-1} is good then we have that $|R(y, Y_r) - R(y, Y_{r-1})| \leq 2^r \varepsilon m$.

Proof. At the end of time t_r we have $Y_r(t_r) = R_r(t_r)$, which is each item $y(\rho_q)$ in Q_{r-1} duplicated $\varepsilon t_r/8$ times. If $S_{r-1}(t_r)$ is good then $|R(y(\rho_q), Y_{r-1}(t_r)) - 2^{r-1}32\rho_q| \leq \varepsilon t_r/2$ following Lemma 2.

Fix q so that $y(\rho_q) \leq y < y(\rho_{q+1})$, where $y(\rho_0)$ and $y(\rho_{1+8/\varepsilon})$ are defined to be $\inf \mathcal{D}$ and $\sup \mathcal{D}$ for completeness. Fixing q this way implies that $R(y, Y_r(t_r)) = 2^{r-1}32\rho_q$. By the above bound on $R(y(\rho_q), Y_{r-1}(t_r))$ we also have that

$$2^{r-1}32\rho_q - \varepsilon t_r/2 \leq R(y, Y_{r-1}(t_r)) < 2^{r-1}32\rho_{q+1} + \varepsilon t_r/2$$

Recalling that $\rho_q = q\varepsilon m/256$, these bounds imply that

$$|R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \leq 2^r \varepsilon m$$

For each time t after t_r, the new item x_t changes the rank of y in both streams Y_r and Y_{r-1} by the same additive offset, so

$$|R(y, Y_r) - R(y, Y_{r-1})| = |R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \leq 2^r \varepsilon m$$

yielding the lemma. ◀

By applying this lemma inductively we can bound the difference between Y_r and $X = Y_0$:

▶ **Corollary 4.** *For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_0(t_1), S_1(t_2), \dots, S_{r-1}(t_r)$ are good, then $|R(y, Y_r) - R(y, X)| \leq 2 \cdot 2^r \varepsilon m$.*

To ensure that all of these S_i are good would require m to grow with n, which would be bad. Happily, it is enough to require only the last $\log_2 1/\varepsilon$ sample summaries to be good, since the other items we disregard constitute only a small fraction of the total stream.

▶ **Corollary 5.** *Let $d = \log_2 1/\varepsilon$. For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_{r-1}(t_r), \dots, S_{r-d}(t_{r-d+1})$ are good, then $|R(y, Y_r) - R(y, X)| \leq 2^{r+2}\varepsilon m$.*

Proof. By Lemma 3 we have $|R(y, Y_r) - R(y, Y_{r-d})| \leq 2^{r+1}\varepsilon m$. At time $t \geq t_{r-d}$, Y_{r-d} and X share all except possibly the first $2^{(r-d)-1}m = 2^{r-1}m/2^d = 2^{r-1}\varepsilon m$ items. Thus

$$|R(y, Y_r) - R(y, X)| \leq |R(y, Y_r) - R(y, Y_{r-d})| + |R(y, Y_{r-d}) - R(y, X)| \leq 2^{r+1}\varepsilon m + 2^r \varepsilon m$$

proving the corollary. ◀

We now prove that if the last several sample streams were good then querying our summary will give us a good result.

▶ **Lemma 6.** *Let $d = \log_2 \frac{1}{\varepsilon}$ and $r = r(t)$. If all $S_r(t), S_{r-1}(t_r), \dots, S_{r-d}(t_{r-d+1})$ are good, then querying our summary with rank ρ (= querying the active GK summary G_r with $\rho/2^r 32$ if $r \geq 1$, or with ρ if $r = 0$) returns $y = y(\rho)$ such that $|R(y, X) - \rho| \leq \varepsilon t$.*

Proof. For $r \geq 1$ we have by Corollary 5 that $|R(y, Y_r) - R(y, X)| \leq 2^{r+2}\varepsilon m \leq \varepsilon t/2$. We apply Lemma 2 once more at row r, which tells us that $|R(y, Y_r) - \rho| \leq \varepsilon t/2$, and combine these bounds with the triangle inequality.

For $r = 0$, the GK guarantee alone proves the lemma. ◀

Lastly, we prove that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that all of $S_r(t)$, $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.

▶ **Lemma 7.** *Let $d = \log_2 1/\varepsilon$ and $r = r(t)$. If $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^3}$ then all $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.*

Proof. There are at most $1+\log_2 1/\varepsilon \leq 4/\varepsilon$ of these summary streams total. Lemma 2 and the union bound give us

$$P(\text{some } S_r \text{ is bad}) \leq \tfrac{4}{\varepsilon}\tfrac{\varepsilon^3}{8}e^{-1/\varepsilon} \leq e^{-1/\varepsilon}$$

which implies our claim. ◀

3.3 Space and time complexity

A minor issue with the algorithm is that, as written in section 3.1, we do not actually have a bound on the worst-case space complexity of the algorithm; we only have a bound on the space needed at any given point in time. This issue is due to the fact that there are low probability events in which $|S_r|$ can get arbitrarily large and the fact that over n items there are a total of $\Theta(\log n)$ sample streams. The space complexity of the algorithm is $O(\max |S_r|)$, and to bound this value with constant probability using the Chernoff bound appears to require that $\max |S_r| = \Omega(\log \log n)$, which is too big.

Fortunately, fixing this problem is simple. Instead of feeding every sample of S_r into the GK summary G_r, we only feed each next sample if G_r has seen $< 2m$ samples so far. That is, we deterministically restrict G_r to receiving only $2m$ samples. Lemmas 3 through 6 condition on the goodness of the sample streams S_r, which ensures that the G_r receive at most $2m$ samples each, and the claim of Lemma 7 is independent of the operation of G_r. Therefore, by restricting each G_r to receive at most $2m$ inputs we can ensure that the space complexity is deterministically $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ without breaking our error guarantees.

From a practical perspective, the assumption in the streaming setting is that new items arrive over the input stream X at a high rate, so both the worst-case per-item processing time as well as the amortized time to process n items are important. For our per-item time complexity, the limiting factor is the duplication step that occurs at the end of each time $t_r = 2^{r-1}m$, which makes the worst-case per-item processing time as large as $\Theta(n)$. Instead, at time t_r we could generate Q_{r-1} and store it in $O(1/\varepsilon)$ words, and then on each arrival $t = 2^{r-1}m+1 \ldots 2^r m$ we could insert both x_t and also the next item in R_r. By the time $t_{r+1} = 2t_r$ that we generate Q_r, all items in R_r will have been inserted into J_r. Thus the worst-case per-item time complexity is $O(\frac{1}{\varepsilon}T_{\text{GK}}^{\max})$, where $T_{\text{GK}}^{\max}$ is the worst-case per-item time to query or insert into one of our GK summaries. Over $2^r 32m$ items there are at most $2m$ insertions into any one GK summary, so the amortized time over n items in either case is $O(\frac{m\log(n/m)}{n}T_{\text{GK}})$, where T_{GK} is the amortized per-item time to query or insert into one of our GK summaries. Algorithm 2 includes the changes of this section.

4 Discussion

Our starting point is a very natural idea used in Manku et al. [6]: downsample the input stream and feed the resulting sample stream into a deterministic summary data structure

Initially, allocate space for G_0. Mark row 0 as live and active.
for $t = 1, 2, \ldots$ **do**
  foreach *live row* $r \geq 0$ **do**
    with probability $1/2^r 32$ **do**
      Insert x_t into G_r if G_r has seen $< 2m$ insertions.
    if $r \geq 1$ *and* $2^{r-1}m < t \leq 2^r m$ *and* G_r *has seen* $< 2m$ *insertions* **then**
      with probability $1/2^r 32$ **do**
        Also insert item $t - 2^{r-1}m$ of R_r into G_r.
  if $t = 2^{r-1}m$ *for some* $r \geq 1$ **then**
    Allocate space for G_r. Mark row r as live.
    Query G_{r-1} with $\rho_1 \ldots \rho_{8/\varepsilon}$ to get $Q_{r-1} = y_1 \ldots y_{8/\varepsilon}$.
    Store Q_{r-1}, to implicitly define R_r.
  if $t = 2^r 16m$ *for some* $r \geq 1$ **then**
    Mark row r as active. Unmark row $r-1$ as active.
  if $t = 2^r 32m$ *for some* $r \geq 0$ **then**
    Unmark row r as live. Free space for G_r.
on *query* ρ **do**
  Let $r = r(t)$ be the active row.
  Query G_r for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank ρ (if $r = 0$).
  Return the result.

Algorithm 2. Procedural listing of the algorithm in Section 3.3. The changes between Sections 3.1 and 3.3 are that G_r never has more than $2m$ insertions and that stream R_r is paired with items in B_r.

(compare our Figure 1 with figure 1 on page 254 of [6]). At a very high level, we are simply replacing their deterministic $O(\frac{1}{\varepsilon}\log^2 \varepsilon n)$ MRL summary [5] with the deterministic $O(\frac{1}{\varepsilon}\log \varepsilon n)$ GK summary [3].

However, our implementation of this idea differs conceptually from the implementation of Manku et al. in two important ways. First, we use the GK algorithm strictly as a black box, whereas Manku et al. peek into the internals of their MRL algorithm, using its algorithm-specific interface (New, Collapse, Output) rather than the more generic interface (Insert, Query). At an equivalent level, dealing with the GK algorithm is already unpleasant—the space complexity analysis in [3] is quite involved, and in fact a simpler analysis of the GK algorithm is an open problem [8]. Using the generic interface, our implementation could just as easily replace the GK boxes in the diagram in Figure 2 with MRL boxes; or, for the bounded universe model, with boxes running the q-digest summary of Shrivastava et al. [9].

The second way in which our algorithm differs critically from that of Manku et. al. is that we operate on *streams* rather than on stream *items*. We use this approach in our proof strategy too; the key step in our error analysis, Lemma 3, is a statement about (what to us are) static objects, so we can trade out the complexity of dealing with time-varying data structures for a simple induction. We believe that developing streaming algorithms with analyses that hinge on analyzing streams rather than just stream items is likely to be a useful design approach for many problems.

Acknowledgements. We thank the anonymous reviewers especially for suggesting that we clarify our usage of Theorem 1 in Section 3 and for providing historical context and further insight into previous work, among the many useful comments and suggestions they provided.

Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS'12, pages 23–34, New York, NY, USA, 2012. ACM.
2. Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):26, 2013.
3. Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.
4. Regant YS Hung and Hingfung F Ting. An\ omega (\ frac {1}{\ varepsilon}\ log\ frac {1}{\ varepsilon}) space lower bound for finding ε-approximate quantiles in a data stream. In *Frontiers in Algorithmics*, pages 89–100. Springer, 2010.
5. Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Approximate medians and other quantiles in one pass and with limited memory. *ACM SIGMOD Record*, 27(2):426–435, 1998.
6. Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. *ACM SIGMOD Record*, 28(2):251–262, 1999.
7. J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS'78, pages 253–258, Washington, DC, USA, 1978. IEEE Computer Society.
8. Problem 2: Quantiles – open problems in sublinear algorithms (suggested by graham cormode at kanpur 2006), 2006. `http://sublinear.info/2`.
9. Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 239–249. ACM, 2004.
10. Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.
11. Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. Quantiles over data streams: an experimental study. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 737–748. ACM, 2013.