Learned Interpolation for Better Streaming Quantiles with Worst Case Guarantees

by

Nicholas Schiefer

B.S., California Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Electrical Engineering and Computer Science
August 26, 2022

Certified by .....
Piotr Indyk
Thomas D. and Virginia W. Cabot Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Learned Interpolation for Better Streaming Quantiles with Worst Case Guarantees

by

Nicholas Schiefer

Submitted to the Department of Electrical Engineering and Computer Science
on August 26, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

An ε-approximate quantile sketch over a stream of n inputs approximates the rank of any query point q—that is, the number of input points less than q—up to an additive error of εn, generally with some probability of at least $1-1/\operatorname{poly}(n)$, while consuming $o(n)$ space. While the celebrated KLL sketch of Karnin, Lang, and Liberty achieves a provably optimal quantile approximation algorithm over worst-case streams, the approximations it achieves in practice are often far from optimal. Indeed, the most commonly used technique in practice is Dunning's t-digest, which often achieves much better approximations than KLL on real-world data but is known to have arbitrarily large errors in the worst case. We apply interpolation techniques to the streaming quantiles problem to attempt to achieve better approximations on real-world data sets than KLL while maintaining similar guarantees in the worst case.

Thesis Supervisor: Piotr Indyk
Title: Thomas D. and Virginia W. Cabot Professor

Acknowledgments

My path through graduate school has been unusually circuitous and so I owe a great many thanks to the people who have been with me through each bend and turn along the way.

I am immensely grateful to my advisor, Piotr Indyk, for his guidance and especially his patience as my research interests evolved over the years. I started working with Piotr in 2016 (!) thinking that I would work on theoretical computer science. He waited patiently while I spent three years in industry, learning how the real world works, and welcomed me back warmly when I decided to return. Piotr has always encouraged me to pursue my most compelling interests even when they did not necessarily align with his. Most recently, he supported me—intellectually and financially—while I started ambitious new projects very, very far from the theory work we started with. Thank you, Piotr: I know that I was not the easiest graduate student, but I hope this unusual journey was fun all the same.

I thank my other co-authors on the work presented in this thesis, Justin Chen, Shyam Narayanan, Sandeep Silwal, and Tal Wagner, for their insights and their efforts. As they know, our results were challenging to obtain despite their ultimate simplicity. It's been a pleasure working with all of you.

Thank you to the other members of Piotr's group who I've overlapped with, Anders Aamand, Arturs Backurs, Vaggos Chatziafratis, Talya Eden, Sepideh Mahabadi, Ilya Razenshteyn, Ludwig Schmidt, and Ali Vakilian, for all I've learned over Indian food and more.

I have spent the past year building a new, exciting, and confusing project that I have spent years thinking about. I owe tremendous thanks to my collaborators Daniel Jackson, Geoffrey Litt, and Johannes Schickling for going on this crazy adventure

with me. Thank you for your patience as I wrapped up stubborn projects like this one. I owe an especially hearty thanks to Geoffrey for taking a big leap into the unknown with me: I think we've dragged back enough interesting stuff to last a decade. I look forward to many more fruitful years of collaboration ahead!

This circuitous path was replete with administrative complications that I could never have navigated without help from many people at MIT. I am especially grateful to Janet Fischer, Leslie Kolodziejski, Sylvia Hiestand, Emily Cheng, Rebecca Yadegar, and Debbie Goodwin.

That I am a researcher at all is thanks to the outstanding mentors I've had throughout my career. I'd like to extend particular thanks to Piotr Indyk, Daniel Jackson, Erik Winfree, Leonard Schulman, Rob Phillips, David Stevenson, Milo Lin, Ori Herrnstadt, Scott Gray, Alex Shraer, Mike McMahon, Charles Clarke, and the late Tom Tombrello for their guidance and mentorship over the past decade.

I thank all of my friends for all that they've done, which is too extensive to list here. Thank you all for your companionship through hard times, your joy through good times, and your wisdom through confusing times. I would like to express particular thanks to Adam Jermyn for his counsel on so many decisions, big and small, consequential and not, without which I'd never have gotten so far.

Lastly, I thank my family. Thank you to my mother Nonna, my father Berni, my sister Katie, and my brother Pascal for all of their love and support going back as far as I can remember. Thank you to my late grandparents, Luba, Bruno, and Elisabeth, for their pride and belief in me: I wish you were here to celebrate with me. My deepest thanks go to my partner Sylvia. Thank you for going on this journey with me, sticking with me through thick and thin, making me laugh and smile, and holding me to my values when it mattered most. I could not have asked for a better partner for this adventure and the next one.

Contents

List of Figures

Chapter 1

Introduction

The quantile approximation problem is one of the most fundamental problems in the streaming computational model, and also one of the most important streaming problems in practice. Given a set of items $x_1, x_2, \ldots, x_n$ and a query point q, the *rank* of q, denoted $R(q)$, is the number of items in $\{x_i\}_{i=1}^n$ such that $x_i \leq q$. An ε-approximate quantile sketch is a data structure that, given access to a single pass over the stream elements, can approximate the rank of any query point with additive error at most εn.

Given its central importance, the streaming quantiles problem has been studied extensively by both theoreticians and practitioners. Early work by Manku, Rajagopalan, and Lindsay [10] gave a randomized solution that used $O((1/\varepsilon)\log^2(n\varepsilon))$ space. Later, Greenwald and Khanna [4] developed a deterministic algorithm that requires only $O((1/\varepsilon)\log(n\varepsilon))$ space. More recently, Karnin, Lang, and Liberty (KLL) [7] developed the randomized KLL sketch that uses $O((1/\varepsilon)\log\log(n\varepsilon))$ space and gave a matching lower bound.

Meanwhile, streaming quantile estimation is of significant interest to practitioners

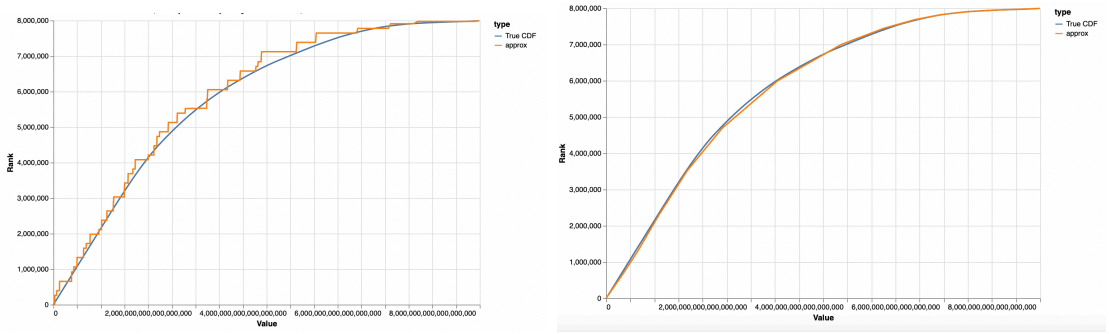
in databases, computer systems, and data science who have studied the problem as well. Most notably, Dunning [3] introduced the celebrated t-digest, a heuristic quantile estimation technique based on 1-dimensional k-means clustering that has seen adoption in numerous systems, including Influx, Apache Arrow, and Apache Spark. Although t-digest achieves remarkable accuracy on many real-world data sets, it is known to have arbitrarily bad error in the worst case [2].

To illustrate this core tradeoff, Figure 1-1a shows the rank function of the `books` dataset from the SOSD benchmark [8], along with KLL and t-digest approximations that use the same amount of space when the data set is randomly shuffled. Figure 1-1b shows the same data set streamed in an adversarial order that we found to induce especially bad performance in t-digest.

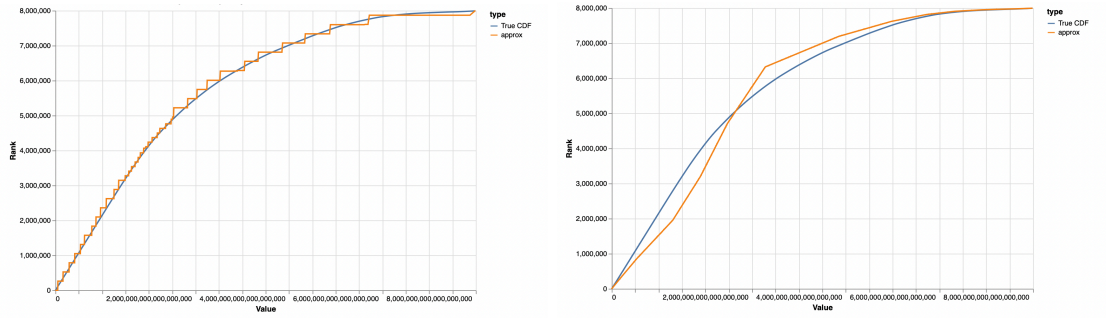
Recent advances in machine learning have led to the development of *learning-augmented algorithms* which seek to improve solutions to classical algorithms problems by exploiting empirical properties of the input distribution. Typically, a learning-augmented algorithm retains worst-case guarantees similar to those of classical algorithms while doing much better on nicely structured inputs that appear in practical applications. We might hope that a similar technique could be used for quantile estimation.

In fact, one of the seminal results in the field studied the related problem of *learned index structures*. An index is a data structure that maps a query point to its rank. Several model families have been tried for this learning problem, including neural networks and the successful recursive model index (RMI) that define a piecewise-linear approximation [9].

Although learned indexes aim to answer rank queries, they do not solve the streaming quantiles estimation problem because they do not operate on the data in a stream. For example, training a neural network or fitting an RMI model require



(a) KLL and t-digest on the SOSD `books` data set presented in a randomly shuffled order.



(b) KLL and t-digest on the SOSD `books` data set presented in an adversarial order that we discovered.

$O(n)$ of the elements in the stream to be present in memory simultaneously, or require multiple passes over the stream.

1.1 Our contributions

We present an algorithm for the streaming quantiles problem that achieves much lower error on real-world data sets than the KLL sketch while retaining similar worst case guarantees. This algorithm, which we call linear compactor sketch, uses *linear interpolation* in place of parts of the KLL sketch. Intuitively, this linear interpolation provides a better approximation to the true cumulative density function when that function is relatively smooth, as many CDFs of real world datasets are.

On the theoretical side, we prove that linear compactor sketch achieves similar worst case error to the KLL sketch. That is, linear compactor sketch computes an ε-approximation for the rank of a single item with probability $1 - \delta$ and space $O((1/\varepsilon) \log^2 \log(1/\delta)$. This is within a factor that is polylogarithmic (in $1/\delta$) of the known lower bounds and the (rather complex) version of the KLL sketch that matches it [7]. Our proof is a relatively straightforward modification of the analysis of the original KLL sketch, due to the general similarity of the algorithms. In fact, we can view our algorithm as exploiting a place in the KLL sketch analysis that left some "slack" in the algorithm design.

In our experiments, we demonstrate that linear compactor sketch achieves significantly lower error than the KLl sketch on a variety of benchmark data sets from the SOSD [8] benchmark library and for a wide variety of input orders that induce bad behaviour in other algorithms like t-digest. In many cases, linear compactor sketch achieves a space-error tradeoff that is competitive with t-digest.

Chapter 2

Understanding the KLL sketch

The complete KLL sketch and its analysis are complex. Here, we present a simplified version that closely parallels the algorithm as used in practice [5].

2.1 The basic KLL sketch

The KLL sketch is composed of a hierarchy of *compactors*. Each compactor has a capacity k, which defines the number of items that it can store. Each item is also associated with a (possibly implicit) *weight* which represents the number of points from the input stream that it represents in the sketch. All points in the same compactor have the same weight.

When a compactor reaches its capacity, it is compacted. A compaction begins by sorting the items. Then, either the even or odd elements in the compactor are chosen, and the unchosen items are discarded. The choice to discard the even or odd items is made with equal probability. The chosen items are then placed into the next compactor in the hierarchy and the points are all assigned a weight twice

what they began with. This general setup is common to many streaming quantiles sketches [10, 7].

To predict the rank of a query point q, we return the sum of the weights of all points, in all compactors, that are at most q.

A key contribution of the KLL sketch is to use different capacities for different compactors. We say that the first compactor where points arrive from the stream has a height of 0, and each successive compactor has a height one higher than the compactor below it, for a total of H compactors. In KLL, the compactor at height h has capacity $k_h = \max(kc^{H-h}, 2)$, where k is a space parameter that defines the capacity of the highest compactor and c is a scale parameter that is generally set to $c = 2/3$.

2.2 Analysis of the KLL sketch

Here, we give a somewhat simplified—to focus on the essential details—version of the analysis of the simplified KLL sketch in [7]. Consider the KLL sketch described above that terminates with H different compactors. The weight of the items at height h is $w_h = 2^h$. Let m_h be the number of compaction operations in the compactor at height h.

Consider a single compaction operation in the compactor at height h and a point x in that compactor at that time. If x was one of the even elements in the compactor, the total weight to left of it, which defines its rank, is unchanged by the compaction. If x is one of the odd elements in the compactor, the total weight either increases by w_h (if the odd items are chosen) or decreases by w_h (if the even items are chosen). For the ith compaction operation at level h, let $X_{i,h}$ be 1 if the odd items were chosen and -1 if the even items were chosen. Observe that $\mathrm{E}[X_{i,h}] = 0$ and $|X_{i,h}| \leq 1$. Then

the total error introduced by all compactions at level h is $\sum_{i=1}^{m_h} w_h X_{i,h}$. Consider any point x in the stream. The error in $R(x)$ introduced by compaction at all levels up to a fixed level H' is therefore $\sum_{i=0}^{H'-1} \sum_{i=1}^{m_h} w_h X_{i,h}$.

Applying a two-tailed Hoeffding bound to this error, we obtain that

$$\Pr[\text{error is} > \varepsilon n] = \Pr\left[\left|\sum_{i=0}^{H'-1} \sum_{i=1}^{m_h} w_h X_{i,h}\right| > \varepsilon n\right] \leq 2\exp\left(-\frac{\varepsilon^2 n^2}{2\sum_{i=0}^{H'-1} \sum_{i=1}^{m_h} w_h^2}\right).$$

This addresses the error introduced by all layers up to H'. Notice that if we set $H' = H$, then the error bound is dominated by the weight terms from the highest compactors. Instead, the analysis treats the final $s = O(\log \log n)$ compactors separately, and is assumed to contribute its worst possible error of w_h for reach compaction. This is the key lemma in the KLL analysis and the point of departure for the linear compactor sketch.

Chapter 3

The linear compactor sketch

We propose a streaming quantiles algorithm that combines our empirical and theoretical observations about how KLL might be improved. We leave the basic KLL architecture unchanged and use normal KLL compactors. However, we replace the top $t = O(1)$ compactors with a structure that we call a *linear interpolator*.

Linear interpolators. A *linear interpolator* is a sorted list of elements, each of which is a pair of an item from the stream and a weight. As in KLL, the weight represents the number of stream items that the item represents; unlike in KLL, this weight varies between elements in the list and may be an arbitrary floating point number, rather than a power of two. Like a KLL compactor, a linear interpolator has a capacity. When that capacity is exceeded, it undergoes *compaction* and only half of its elements are retained.

A KLL compactor C_h at height h implicitly represents a piecewise-constant function f: specifically,

$$f(q) = \sum_{x \in C_h : x \leq q} w_h.$$

This function is the contribution of this compactor to the approximated rank of a query point q. A linear interpolator implicitly represents a *piecewise-linear* function which also contributes to the rank of q. Given a linear interpolator $L = \{(y_1, w_1), (y_2, w_2), \ldots, (y_k, w_k)\}$ with $y_1 \le y_2 \le \cdots \le y_k$, the contribution of L to the the rank of q is

$$f_L(q) = \underbrace{\sum_{i=1}^{i^*} w_i}_{\text{KLL-style term}} + \underbrace{w_{i^*} \frac{q - y_{i^*-1}}{y_{i^*} - y_{i^*-1}}}_{\text{interpolation term}} \tag{3.1}$$

where i^* is the smallest index such that $y_{i^*} > q$. In effect, we spread the weight of y_{i^*} over the entire interval between y_{i^*-1} and y_{i^*}, with uniform density, rather than treating it as a point mass at y_{i^*} exactly. The resulting contribution $f_L(q)$ is a monotone, piecewise-linear function, as desired.

Adding points to a linear interpolator. Our linear interpolator receives points from the last of the KLL-style compactors, each with a fixed weight of w_{H-s-1}. These points and weights cannot be merged by merely concatenating the arrays. To see this, consider adding a single point b with unit weight to a compactor with two points a and c with $a < b < c$, and where c has weight w. The weight of c after the compaction should not be w since the weight of c before the addition should be spread uniformly over the entire interval $[a, c]$.

Instead, we add a set of new points $y_1 < y_2 < \cdots < y_m$ to an existing set of points $x_1 < x_2 < \cdots < x_n$ by merging the two lists of points into one list and sorting them into the list $z_1 < z_2 < \cdots < z_{m+n}$. Next, we set $w(z_1)$ equal to the weight of z_1 in the original list and compute the new weights recursively. Assuming that $z_i = x_i$

without loss of generality, we set

$$w(z_i) = w(x_i)\frac{x_i - z_{i-1}}{x_i - x_{i-1}} + w(y_*)\frac{x_i - z_{i-1}}{y_* - y_{*-1}}$$

where y_* is the first y_i such that $y_i > z_i$.

Equivalently, we convert each of the weight functions into a rank function using Equation 3.1, sum those, and then compute the finite differences to obtain the final weight function.

Compacting a linear interpolator. Lastly, we describe the process for compacting a linear compactor. Given a parameter $\alpha \in [0, 1]$ and a linear compactor C containing n points, we wish to obtain a new linear compactor C' with αn points with the following properties:

- The points in C' are subset of the points in C.
- The total weight of the points in both compactors is the same, so that $\sum_{x \in C} w(x) = \sum_{x \in C'} w(x')$.
- For every point $x \in C'$, the rank $f_C(x) = f_{C'}(x)$.
- The "error" introduced by the compaction is as small as possible. That is, for some loss function L, we would like $\sum_{x \in C} L(f_{C'}(x), f_C(x))$ to be as small as possible.

In general, we use $\alpha = 1/2$.

It is important that this procedure can be completed efficiently. In our experiments, we primarily use supremum (ℓ_∞) loss $L(x, x') = \sup_x |x - x'|$. This can be minimized using a dynamic programming technique introduced by [6].

Chapter 4

Analysis

We give a worst-case analysis of our algorithm that matches the worst-case analysis for the version of KLL that does not use a Greenwald-Khanna sketch as a final layer:

Theorem 1. *The linear compactor sketch described in Chapter 3 computes an ε-approximation for the rank of a single item with probability $1-\delta$ with space complexity $O((1/\varepsilon)\log^2\log(1/\delta)$.*

Our technique analyzes the error introduced by each compactor, using two techniques. To analyze the error of the KLL-style compactors of the linear compactor sketch, we prove that they introduce precisely the same error as they would in a KLL sketch run on the same stream. We then apply the two-part analysis of the KLL sketch, analyzing the first $H-s$ compactors and the $(H-s)$th through $(H-s+t)$th compactors separately. To analyze the error of the linear compactor at the top, we analyze the error introduced per compaction. We then analyze the number of compactions of the linear compactor and therefore the total error introduced by the linear compator.

Consider a stream $X = x_1, x_2, \ldots, x_n$. Let $S(X)$ be a KLL sketch computed on

this stream that terminates with H compactors and let $S_t(X)$ be the tth compactor of $S(x)$. Similarly, let $S'(X)$ be a linear compactor sketch computed on this stream with $H-s$ levels of KLL-style compactors and one linear compactor at level $H-s+1$. Let $S'_t(X)$ be the tth compactor $S'(X)$.

Following [7], let $R(S, x, h)$ be the rank of item x among all points in compactors in the sketch S at heights at most $h' \leq h$ at the end of the stream. For convenience, we set $R(x, 0)$ to be the true rank of x in the input stream. Let $\text{err}(S, x, h) = R(S, x, h) - R(S, x, h-1)$ be the total change in the approximate rank of x due to the compactor at level h. The total error decomposes into this error per compactor as $\sup_x |R(x, 0) - S'(x)| = \sum_{h=1}^{H} \text{err}(S', x, h)$.

Analyzing the KLL compactors. In both S and S', stream elements only move from lower compactors to higher ones, and the compactor at level t at any point while processing the stream is defined entirely by the compactors at *lower* levels up to that point. Therefore, for all $t < H - s$, $S'_t(X) = S(X)$. We restate a theorem from [7] that we will use as a key lemma:

Theorem 2 (Theorem 3 in [7])**.** *Consider the KLL sketch $S(X)$ with height H, and where the compactor at level h has capacity $k_h \geq kc^{H-h}$. Let H'' be the height at which the compactors have size greater than 2 (i.e., where the compactors do not just perform sampling). For any $H' > H''$, we have*

$$\Pr\left[\sum_{h=1}^{H} \text{err}(S, x, h) > 2\varepsilon n\right] \leq 2\exp\left(-c\varepsilon^2 k 2^{H-H''}/32\right) + 2\exp\left(-C\varepsilon^2 k^2 2^{2(H-H')}\right)$$

Analyzing the linear compactor As mentioned, we will analyze the error introduced by the linear compactor compaction-by-compaction. Specifically, we analyze the linear compaction sketch between the end of one compaction and the end of the

following compaction. During this interval, a total of d items of weight 2^{H-t} are added to the linear compactor, where either $d = sk$ if the linear compactor has never compacted or $d = sk/2$ if it has.

Let f be the piecewise linear rank function of the full linear compactor right before the compaction with endpoint set Z comprising $z_1 < z_2 < \cdots < z_{sk}$ and weight function w. Let f' be the piecewise linear rank function of the compactor immediately after the compaction, with endpoint set $Z' \subset Z$, weight function w', and $|Z'| = |Z|/2$.

The linear compaction procedure removes some of the items in the linear compactor. A *run* is a sequence of removed elements that are adjacent in sorted order. We show that the error introduced by a linear compactor is bounded by the greatest run of displaced weight.

Lemma 1. *Organize $Z \setminus Z'$ into continuous runs of adjacent removed elements, and let F_i be the total weight of the ith run. Then* $\sup_{z \in Z} |f(z) - f'(z)| \leq \max_i F_i$.

Proof. Fix a run with endpoints a and b and let its total weight be $F = \sum_{i=a+1}^{b-1} w(z_i)$. Consider any point z_j in that run, so that $a < j < b$. Its original rank was $f(z_j) = \sum_{i=1}^{j} w(z_i)$ while its new rank is, by construction, $f'(z_j) = \sum_{i=1}^{a} w(z_i) + \frac{F + w(z_b)}{z_b - z_a}(z_j -$

z_a). Therefore,

$$\begin{aligned}
|f(z_j) - f'(z_j)| &= \left| \sum_{i=a+1}^{j} w(z_i) - \frac{F + w(z_b)}{z_b - z_a}(z_j - z_a) \right| \\
&= \left| \sum_{i=a+1}^{j} w(z_i) - \sum_{i=a+1}^{b} w(z_i) \frac{z_j - z_a}{z_b - z_a} \right| \\
&\leq \sum_{i=a+1}^{b-1} w(z_i) \\
&= F
\end{aligned}$$

□

Next, we show that the greater error introduced by a linear compaction step occurs *at one of the discarded endpoints*:

Lemma 2. *There is some* $z_i \in Z \setminus Z'$ *such that* $\sup_{x \in [z_1, z_{sk}]} |f(x) - f'(x)| = |f(z_i) - f'(z_i)|$.

Proof. Consider any point $x \in [z_1, z_d]$. If x is one of the endpoints retained after compaction $z_j \in Z'$, then by construction $f(x) = \sum_{i \leq j} w(z_i) = f'(x)$. Our claim does not depend on the error if x is one of the endpoints in $Z \setminus Z'$.

Suppose then that x is not in the original endpoint set Z. Let z_a and z_b be the left and right neighbours of x in Z. By the definition of the linear compactor,

$$f(z_a) = \sum_{i=1}^{a} w(z_i), \quad f(x) = \sum_{i=1}^{a} w(z_i) + \frac{w(z_b)}{z_b - z_a}(x - z_a), \quad f(z_b) = \sum_{i=1}^{b} w(z_b)$$

Let $z_{a'}$ and $z_{b'}$ be the left and right neighbours of x in Z'. By definition, the weight

$W := w'(z_{b'}) = \sum_{i=a'+1}^{b'} w(z_i)$ and so we have

$$f'(x) = \sum_{i=1}^{a'} w(z_i) + \frac{W}{z_{b'} - z_{a'}}(x - z_{a'}).$$

Therefore,

$$f(x) - f(x') = \sum_{i=a'+1}^{a} w(z_i) + \left(\frac{w(z_b)}{z_b - z_a} - \frac{W}{z_{b'} - z_{a'}}\right)(x - z_{a'})$$

Observe that this expression obtains its extrema on the interval $[z_a, z_b] \subset [z_{a'}, z_{b'}]$ at either z_a or z_b, depending on the sign of $D = \frac{w(z_b)}{z_b - z_a} - \frac{W}{z_{b'} - z_{a'}}$. In either case, $|f(x) - f(x')|$ achieve its maximum at one of the endpoints z_a or z_b, completing the proof. □

We use a simple counting argument to bound the size of the majority of the weights in a linear compactor:

Lemma 3. *Consider a linear compactor that has just completed its cth compaction. At least half of the endpoints Z in the linear compactor have weight at most* $(2c + 3)2^{H-t}$.

Proof. Every point enters the linear compactor with weight 2^{H-t}. After c compactions, a total of $(2+c)sk/2$ such points have entered the compactor. A compaction operation conserves the total weight of points so the total weight of the compactor is $(2 + c)2^{H-t}sk/2$.

Suppose that more than half of the sk points currently in the compactor have weight at most T. These points have a total weight greater than $Tsk/4$ while the remaining point s each have weight at least 2^{H-t} and so have total weight at least

$2^{H-t}sk/4$. The total weight is therefore $(T+2^{H-t})sk/4$. This weight must not exceed the total conserved weight $(2+c)2^{H-t}sk/2$, and so we have

$$\frac{(T+2^{H-t})sk}{4} \leq \frac{(2+c)2^{H-t}sk}{2}$$

Rearranging, we obtain that our result holds for any $T \leq (2c+3)2^{H-t}$. □

Combining these lemmas, we obtain a bound on the error introduced during a single compaction step.

Theorem 3. *Suppose that the compaction being studied is the* $(c+1)$*th compaction. The error introduced during this compaction step is* $\sup_x |f(x) - f'(x)| \leq (c+2)2^{H-t+1}$.

Proof. We construct a particular post-compaction distribution of weights as follows. Let f'' be the rank function for that post-compaction state. During this interval, there were $tk/2$ points with weight 2^{H-t} that we added to the linear compactor for the first time. In addition, there were $tk/2$ points remaining from a previous linear compaction. We sort the $tk/2$ new points and keep every fourth point, discarding the rest and reallocting their weight to the next highest retained point (of either type). By Lemma 3, there exists at least $tk/4$ of the existing points in the linear compactor with weight at most 2^{H-t+c}. We sort these points and discard every other point. In total, we discard the required $tk/2$ points.

Observe that the longest possible run in this compaction consists of one of the existing points and three (out of a sequence of four) of the new points that were discarded. By Lemma 1, the error introduced on any of the original endpoints by this compaction is bounded by the sum of the weights of the points in the run: in

this case, that sum is $2^{H-t} + 3 \cdot (2c+3)2^{H-t} \leq (c+2)2^{H-t+1}$. By Lemma 2, we find that the error introduced by f'' is $\sup_x |f(x) - f''(x) \leq (c+2)2^{H-t+1}$.

We have exhibited a particular feasible solution to the optimization problem in the linear compaction. Our actual algorithm finds, among all such feasible solutions, the one that minimizes this error function; it follows that

$$\sup_x |f(x) - f'(x) \leq \sup_x |f(x) - f''(x) \leq (c+2)2^{H-t+1}. \qquad \square$$

Combining KLL and linear compactors. Lastly, we combine our analysis of the KLL and linear compactors to obtain an overall error bound and prove Theorem 1. Our analysis closely follows the form of the proof of Theorem 4 in [7].

Proof of Theorem 1. First, we analyze the compactors with height at most $H - t$, including the sampling compactors. These are all KLL-style compactors; by Theorem 2 these compactors will contribute error at most εn with probability $1 - \delta$ so long as $\varepsilon k 2^s \geq c' \sqrt{\log(2/\delta)}$ for a sufficiently small c'. Second, we analyze the top $s - t$ compactors. The error introduced by these compactors is bounded by the error of the equivalent KLL sketch where we have a full s equal-size compactors at the top. This error is in turn bounded by $\sum_{h=H-s+1}^{h} m_h w_h = \sum_{h=H-s+1}^{H} n/k = sn/k$, where m_h is the number of times that the KLL compactor at level h is compacted and $w_h = 2^h$ is the weight associated with that compactor; this is at most εn so long as $s \leq k\varepsilon$. Taking $k = O(1/\varepsilon \log\log(1/\delta))$ and $s = O(\log\log(1/\delta))$ as in KLL, we satisfy both of these conditions.

Lastly, we analyze the single linear compactor with size tk that replaces the top $t < s$ KLL compactors. Let M be the number of compactions of the linear compactor. Observe that since between each compaction of the linear compactor we add $tk/2$ entries, each with weight 2^{H-t} to the compactor, and so $M \leq 2n/(tk2^{H-t})$. Applying

Theorem 3, and summing the error introduced per compaction, we obtain a total error of

$$\sum_{c=1}^{M}(c+2)2^{H-t+1} = M2^{H-t} + 2^{H-t}M(M+1) \leq 2^{H-t+1}M^2 \leq \frac{8n^2}{t^2k^2 2^{H-t}}$$

Our compactors are sized at each level in the same was a KLL. As in the KLL analysis, we have $H \leq \log(n/ck) + 2$ for a constant $0 < c < 1$. Therefore, our error is bounded by

$$\frac{8n^2}{t^2k^2 2^{\log(n/ck)+2-t}} \leq \frac{8ckn^2}{t^2k^2n2^{2-t}} = \frac{cn2^{t+1}}{t^2k}$$

For constant t and any $k = O(1/\varepsilon \log\log(1/\delta))$ as in KLL, this is at most εn. Therefore, the total error of the sketch is $O(\varepsilon n)$ as required.

Each part of the sketch contributes some space. The KLL compactors increase geometrically in size, so the space used by the KLL portion of the sketch is dominated by the top $s - t$ compactors and uses $O(sk) = O(1/\varepsilon \log^2 \log(1/\delta))$ space. The linear compactor uses twice as much space per element as a KLL compactor, for a total of $O(tk) = O(k)$ space, so the total space usage is $O(1/\varepsilon \log^2 \log(1/\delta)^2)$. □

Chapter 5

Experiments

We wrote a performant implementation of our algorithm and evaluated its empirical error over a wide range of space parameters k and several linear compactor heights t. Our experiments were conducted on the recent SOSD [8] benchmarking suite for learned index structures. Each SOSD benchmark consists of a large number (generally 200 to 800 million) of 64-bit unsigned integer values. Of particular interest were the `books`, `osm_cellid`, and `wiki_ts` data sets, since the rank functions of these three data sets have distinctly different shapes, as shown in Figure 5-1.

Parameterization. The algorithm is parameterized by the KLL space parameter k, which determines the size of the largest compactors and the linear compactor, and t, the number of KLL compactors that are replaced by the linear compactor. Our worst-case bound holds for any constant t but this bound is exponential in t. In practice, we experimented with a variety of small but non-zero values ($t = 1, 2, 3$). We see t as a parameter that is tunable based on the desired empirical performance and desired worst-case guarantees and expect that it will be selected on an application-by-application basis.

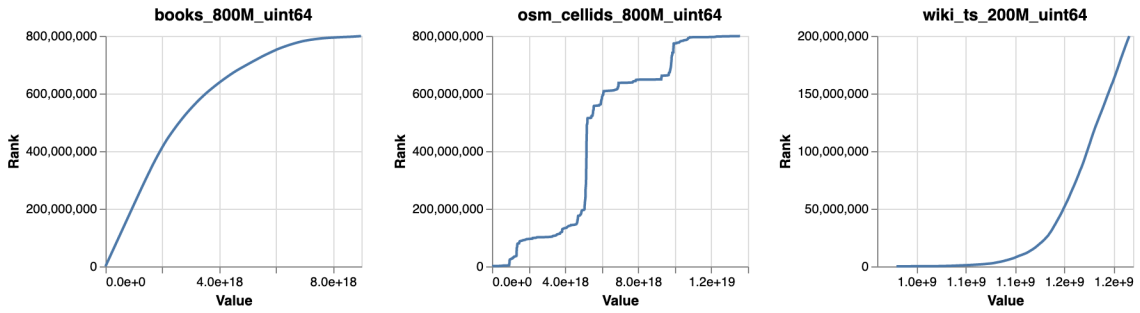


Figure 5-1: The rank functions for the three SOSD data sets used in our experiments. The three data sets have rank functions with distinctive shapes, allowing us to compare the algorithms in a variety of settings

Implementation details. We implemented our algorithms in C++ with Python bindings for experiment management and data analysis. Our implementation is reasonably performant: in informal experiments, it achieves a throughput that is only about three times less than that of highly-optimized, production-quality KLL implementations. This performant implementation allowed us to work with the entire SOSD data sets; in our preliminary work, we found that many promising algorithms would only show improvements over KLL on moderately-sized data sets of less than a million points. Our implementation supports any integer $t \geq 0$: when $t = 0$, our implementation is identical to the commonly implement variant of KLL without the Greenwald-Khanna sketch.

Baselines. Our algorithm is most naturally compared to KLL since the KLL sketch can be seen as an instance of the linear compactor sketch with no linear compactor. We ran our experiments on our implementation of KLL (by setting $t = 0$) and validated those results with an open-source implementation from Facebook's Folly library [1]. Like most implemented version of the KLL sketch, neither of these include the final Greenwald-Khanna sketch that is required to achieve space-optimality.

In addition to the KLL sketch, which offers worst-case guarantees, we ran experiments on the t-digest [3], which is commonly used in practice but is known to have arbitrarily bad worst-case performance [2]. We used the C++ implementation of t-digest in the `digestible` library [11].

Stream order. We found that many streaming quantiles algorithms without worst-case guarantees achieve very low error compared to the KLL sketch if they are given an input stream in a particular order but high error on other input orders. For example, Figures 1-1a and 1-1b show that, even for a fixed set of inputs with a smooth rank function (`books`), there exists an adversarial order that makes the t-digest approximation have high error. This observation might be of independent interest.

As a result, we evaluated the linear compactor sketch and the baselines on a variety of input orders for each data set:

- Random: the data are shuffled with a fixed seed.
- Sorted: the data are presented in a sorted order.
- First half sorted, second half reverse-sorted: the first half of the stream has the first half of the sorted data, in that order. The second half of the stream has the second half of the sorted data presented in *reverse-sorted order*.
- Flip flop: the stream has the smallest element, then the largest element, then the second smallest element, then the second largest element, and so on.

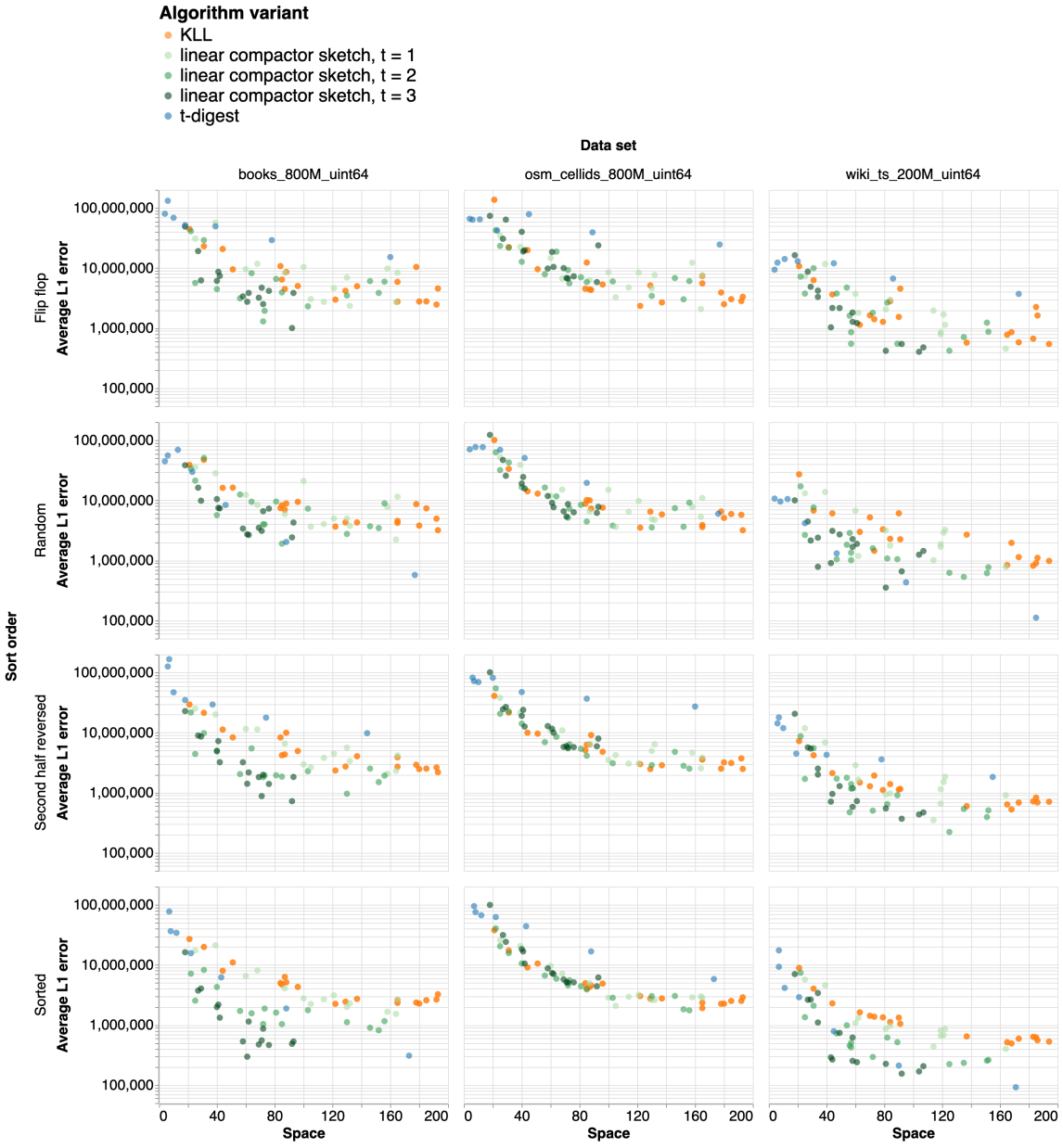


Figure 5-2: Space-error tradeoff curves for the baselines and linear compactor sketch on three different data sets from SOSD and four different sort order, described above. A better algorithm has a curve that is further down and to the left, indicating lower error at a given amount of space.

5.1 Experiment results and discussion

Our primary tool for insight into our experiments is the space-error tradeoff curve that shows how the total space needed for the sketch compares to the empirical error between the exact rank function and the approximation defined by the sketch. We obtain these curves for three different data sets from SOSD, four different sort orders, and four different algorithms; these curves are shown in Figure 5-2. We use average L1 error, defined for a data set X as $\sum_{x \in X} |f(x) - f'(x)|$.

The linear compactor sketch is never significantly worse than KLL: with adversarial input orders like flip-flop it roughly matches KLL performance. In contrast, when the data are randomly shuffled or sorted, linear compactor sketch is competitive with t-digest in many cases. The differences are most pronounced on the `books` dataset; this is easy to understand when looking at the true rank function and approximate rank function for the three different data sets.

Bibliography

[1] Folly: Facebook open-source library. `https://github.com/facebook/folly`.

[2] Graham Cormode, Abhinav Mishra, Joseph Ross, and Pavel Veselỳ. Theory meets practice at the median: a worst case comparison of relative error quantile algorithms. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2722–2731, 2021.

[3] Ted Dunning. The t-digest: Efficient estimates of distributions. *Software Impacts*, 7:100049, 2021.

[4] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

[5] Nikita Ivkin, Edo Liberty, Kevin Lang, Zohar Karnin, and Vladimir Braverman. Streaming quantiles algorithms with small space and update time. *arXiv preprint arXiv:1907.00236*, 2019.

[6] Hosagrahar Visvesvaraya Jagadish, Nick Koudas, S Muthukrishnan, Viswanath Poosala, Kenneth C Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In *VLDB*, volume 98, pages 24–27, 1998.

[7] Zohar Karnin, Kevin Lang, and Edo Liberty. Optimal quantile approximation in streams. In *2016 ieee 57th annual symposium on foundations of computer science (focs)*, pages 71–78. IEEE, 2016.

[8] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. Sosd: A benchmark for learned indexes. *arXiv preprint arXiv:1911.13014*, 2019.

[9] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pages 489–504, 2018.

[10] 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.

[11] Dan Morton. digestible: A modern c++ implementation of a merging t-digest data structure. `https://github.com/SpirentOrion/digestible`.