DIMACS Technical Report 2004-52
October 2004

Approximate Histogram and Wavelet Summaries of Streaming Data

by

S. Muthukrishnan[1,2]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Martin J. Strauss[3,4]
Depts. of Math and EECS
The University of Michigan
Ann Arbor, Michigan 48109

[1]Permanent Member
[2]Supported in part by NSF ITR 0220280.
[3]Permanent Member
[4]Part of this work was done while the second author was at AT&T Labs

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research and Microsoft Research. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

We consider several algorithms for building histograms that are optimal or nearly optimal, from streaming data. Specifically, we consider a vector $\mathbf{A}$ given either as a stream $\mathbf{A}[0], \mathbf{A}[1], \ldots$ of aggregate values or a stream $(i_1, x_{i_1}), (i_2, x_{i_2}), \ldots$ of updates, where (i_j, x_{i_j}) is interpreted as a request to update $\mathbf{A}[i_j]$ to $\mathbf{A}[i_j] + x_{i_j}$. We then build a histogram $\mathbf{H}$; that is, a piecewise-constant vector with a given number B of pieces. We also consider the related representation formats of piecewise-linear representations and Haar wavelet representations.

We consider first *point queries* to the data. The point query i should ideally be answered by $\mathbf{A}[i]$; instead, it will be answered by $\mathbf{H}[i]$, leading to square error $(\mathbf{A}[i] - \mathbf{H}[i])^2$. The sum-square-error is $\sum_i (\mathbf{A}[i] - \mathbf{H}[i])^2$, which we want to minimize or nearly minimize. Next, we consider *range queries*, of the form (ℓ, r), which should ideally be answered by $\sum_{\ell \le i < r} \mathbf{A}[i]$; instead, it is answered by $\sum_{\ell \le i < r} \mathbf{H}[i]$, leading to sum square error $\sum_{\ell, r} \left(\sum_{\ell \le i < r} \mathbf{A}[i] - \mathbf{H}[i] \right)^2$. Finally, we consider some of these questions with regard to two-dimensional signals; *e.g.*, signals $\mathbf{A}[i, j]$ indexed by two (or more) integers instead of just a single integer.

1 Introduction

We study a synopsis abstract data structure similar to an array abstract data type commonly seen in textbooks. Specifically, we model an array $\mathbf{A}$ of N real values, $\{i : 0 \leq i < N\} = [0, N)$. Like an array, a user poses a *point query*, *i.e.*, an index i, and expects $\mathbf{A}[i]$ in return. Compared with an array, however, the modes for updating data values are restricted. First we consider only ordered and aggregated data: the values $\mathbf{A}[0], \mathbf{A}[1], \ldots$, are presented in order, followed by point queries.

After considering the basic abstract data type, we generalize in several directions:

- The user can pose *range queries*; *i.e.*, a pair (ℓ, r) of indices, for which the ideal answer is $\sum_{\ell \leq i < r} \mathbf{A}[i]$. This is more general than standard arrays that support point queries.
- The data can be updated additively. That is, interspersed with point or range queries are update commands of the form "add x to $\mathbf{A}[i]$," where x may be positive or negative. Note that this is still less general than an array data type, since an update of the form "change $\mathbf{A}[i]$ to x," which is supported by arrays, is not supported here.
- The data may be multidimensional. That is, in dimension d, the index is a d-tuple of integers. Ideally, multidimensional and single dimensional arrays are equivalent from the perspective of point queries, but are different under range queries. Whether an array is considered to be single- or multi- dimensional also matters to the approximate implementations we introduce below.

We will study approximate versions of the abstract data structure above. The response to a point query i to instance $\mathbf{A}$ is not $\mathbf{A}[i]$ but $\mathbf{R}[i]$, where $\mathbf{R}$ is some approximate representation for $\mathbf{A}$. The error of this answer is $\mathbf{R}[i] - \mathbf{A}[i]$. We report the quality of a representation as $\sum_i |\mathbf{R}[i] - \mathbf{A}[i]|^2$, written $\|\mathbf{R} - \mathbf{A}\|^2$, where the sum is over all queries. A approximate data structure can have any internal storage format, but must lead to a vector $\mathbf{R}$ of length N—the vector of answers that it gives.

Traditionally, the advantage of implementing the approximate abstract data type instead of the ideal abstract data type is that one can store an approximate representation in much less space than the full data set. That is, the approximate data structure is a *summary.*

We will be interested in two related types of sparse summaries, *histograms* and *Haar wavelets.*

Histograms. A B-bucket *histogram* of length N is a partition of $[0, N)$ into intervals $[b_0, b_1) \cup [b_1, b_2) \cup \cdots \cup [b_{B-1}, b_B)$, where $b_0 = 0$ and $b_B = N$, together with a collection of B heights h_j, for $0 \leq j < B$, one for each bucket. On point query i, the histogram answer is h_j, where j is the index of the bucket containing i; that is, the unique j with $b_j \leq i < b_{j+1}$. In vector notation, we write χ_S for the vector that is 1 on the set S and zero elsewhere. Thus the answer vector of a histogram is $\mathbf{R} = \sum_{0 \leq j < B} h_j \chi_{[b_j, b_{j+1})}$.

Thus, in building a B-bucket histogram, we want to choose $B - 1$ boundaries b_j and B heights h_j that tend to minimize the sum square error $\left\| \mathbf{A} - \sum_{0 \leq j < B} h_j \chi_{[b_j, b_{j+1})} \right\|^2$. Once we have chosen the boundaries, the best bucket height on an interval I is the average of $\mathbf{A}$ over i. Put another way, let $\psi = \frac{1}{\sqrt{|I|}}\chi_I$. Then $\|\psi\| = 1$ and the vector resulting the best histogram that is a multiple of ψ is $\langle \mathbf{A}, \psi \rangle \psi$. ■

Haar Wavelets. We now define Haar wavelets concisely. First, we assume that N is a power of 2. This restriction is easily overcome in our algorithms.

A useful term is the *support* of a vector $\mathbf{v}$. The support of $\mathbf{v}$ denoted $\mathrm{supp}(\mathbf{v})$ is $\{i : \mathbf{v}[i] \neq 0\}$. A *dyadic* subinterval of $[0, N)$ is, intuitively, either $[0, N)$ itself or, recursively, a dyadic subinterval of $[0, N/2)$ or of $[N/2, N)$. Formally, it is an interval of the form $[k2^j, (k+1)2^j)$, where j and k must be *integers*.

A *Haar wavelet* vector ψ of length N is either the anamolous vector $\frac{1}{\sqrt{N}}\chi_{[0,N)}$ or $\frac{-1}{|L \cup R|}\chi_L + \frac{1}{|L \cup R|}\chi_R$, where L and R are the left and right halves of a dyadic interval, respectively. We typically drop the term "Haar." There are N wavelet vectors in all. A wavelet term is the index j of a wavelet vector ψ_j together with a (wavelet) coefficient, c_j. A B-term wavelet representation is a collection of B wavelet terms, whose answer vector is $\sum_{j \in \Lambda} c_j \psi_j$, where $|\Lambda| = B$. In building a B-term wavelet representation, we want to choose a set Λ of B wavelet indices and coefficients c_j for $j \in \Lambda$ that tends to minimize the sum square error $\left\|\mathbf{A} - \sum_{j \in \Lambda} c_j \psi_j\right\|^2$. ■

Note that any signal is exactly represented by an N-bucket histogram, where each bucket has size 1. Similarly, any signal is exactly represented by an N-term wavelet representation. For signal $\mathbf{A}$, define $c_j = \langle \mathbf{A}, \psi_j \rangle$. Then one can show that $\mathbf{A} = \sum_j c_j \psi_j$, where the sum is over all the wavelet indexes. We therefore refer to c_j as *the* j'th wavelet coefficient of $\mathbf{A}$. One can also show (Parseval's equality) that $\|\mathbf{A}^2\| = \sum_j c_j^2$. It follows that, for any set Λ of wavelet indices, $\|\mathbf{A}\|^2 = \left\|\mathbf{A} - \sum_{j \in \Lambda} c_j \psi_j\right\|^2 + \sum_j |c_j|^2$. Since $\|\mathbf{A}^2\|$ is constant, our goal of minimizing $\left\|\mathbf{A} - \sum_{j \in \Lambda} c_j \psi_j\right\|^2$ is achieved by choosing Λ to maximize $\sum_j |c_j|^2$, provided the c_j's are optimally chosen to be the corresponding wavelet coefficients of $\mathbf{A}$. Similarly, if $\psi = \frac{1}{\sqrt{|I|}}\chi_I$ and $c = \langle \mathbf{A}, \psi \rangle$, then the error $\|\mathbf{A} - c\psi\|^2$ is $\|\mathbf{A}\|^2 - c^2$.

Note that answering a point query i from a histogram involves figuring out which bucket contains i, which can be done by binary search in time $O(\log(B))$. Answering a point query i from a wavelet representation involves summing all the wavelet terms that involve i. At most $O(\log(N))$ wavelet terms involve i, and these can be arranged in a tree in the natural way, so that the answer takes at most time $O(\log(N))$. We do not address answering time further. Instead, the problem we address is as follows:

Problem. Fix signal $\mathbf{A}$ of length N. Fix integer B and positive reals ϵ and δ. Find a B-bucket histogram (or B-term wavelet representation, respectively) $\mathbf{R}$ such that, except with probability δ,

$$\|\mathbf{A} - \mathbf{R}\|^2 \leq (1 + \epsilon) \|\mathbf{A} - \mathbf{R}_{\mathrm{opt}}\|^2,$$

where $\mathbf{R}_{\mathrm{opt}}$ is the optimal B-bucket histogram (or B-term wavelet representation, respectively). ■

Occasionally we will also discuss approximation in the size of the summary. That is, we will produce a histogram with $B' > B$ buckets and compare its error with the best B-bucket histogram. Here we will give an upper bound for B'. If B' is guaranteed to be at most αB, then we denote the overall guarantee as a $(\alpha, 1+\epsilon)$-approximation. A similar approach can be used for wavelet representations too.

The resources we use will depend quantitatively on N, B, ϵ, δ, and M, where M is a bound on the largest data item in $\mathbf{A}$, which we assume are integers (generally, they are integer multiples of some smallest resolvable quantity). Several different types of resources can be measured, depending on the nature of the data.

Ordered Aggregate Data. The data are presented in order, as $\mathbf{A}[0], \mathbf{A}[1], \mathbf{A}[2], \ldots, \mathbf{A}[N-1]$. After the last data item is presented, the algorithm outputs a histogram or wavelet representation and halts. Here the goal is to bound the total time and total space of the algorithm. We present a deterministic construction that, for constants c_1 and c_2, consumes total time bounded by $c_1 N + (B\log(N)\log(M)/\epsilon)^{c_2}$. For typical values of the parameters, the first term dominates; that is, the algorithm takes time linear in N, independent of (small changes in) the other parameters. The space is $(B\log(N)\log(M)/\epsilon)^{c_2}$, which is much less than N for typical settings of the parameters.

Dynamic Data. This is motivated by streaming data. Here we process an arbitrary sequence of updates of the form "add x to $\mathbf{A}[i]$" and "rebuild summary." The time to rebuild the histogram and the total space are both at most proportional to $(B\log(N)\log(M)\log(1/\delta)/\epsilon)^{c_2}$. In many applications, the time to process an update must be much less than the time to build a histogram. Indeed there are algorithms that make update processing much faster without significant effect on the time to build a histogram and the total space. In this article, however, we give a construction that meets only the same time bound of $(B\log(N)\log(M)\log(1/\delta)/\epsilon)^{c_2}$ as is satisfied for processing updates.

In what follows, we present techniques for various summaries. The techniques presented earlier will not only be useful in their own right for building summaries, but also be used as building blocks for more advanced techniques that are described later.

2 Wavelet Summaries for Ordered Aggregate Data

We are given the data in an ordered aggregate stream, $\mathbf{A}[0], \mathbf{A}[1], \mathbf{A}[2], \ldots, \mathbf{A}[N-1]$. Our algorithm will consist of several stages. The first stage takes in the input stream and outputs a stream of all N wavelet terms. The second stage takes a stream of all wavelet terms and outputs just the B terms with largest coefficients.

2.1 Online Wavelet Transform

Our goal is to suck in a stream of data and spit out a stream of all wavelet coefficients.

A basic *filter* Φ reads in two elements at a time. It then outputs the sum and the difference of these two elements, onto two different output streams, called the sum output stream and difference output stream. Now, form an overall filter Φ by cascading $\log(N)$ basic filters, Φ_0 to $\Phi_{\log(N)-1}$. The input to Φ_0 is the overal data input. For each $j < \log(N)$, the sum output stream of Φ_j is piped to become the input of Φ_{j+1} and the difference output stream of Φ_j is sent to the output of Φ (the output of Φ interleaves the difference output streams of the various Φ_j's arbitrarily).

Each Φ_j does $O(1)$ work per input and uses $O(1)$ space. One can check recursively that Φ_j has $N/2^j$ inputs and $N/2^{j+1}$ outputs to each stream. Thus the total work of Φ is proportional to $\sum_j N/2^j$, which is $O(N)$, and the total space of Φ is proportional to $\sum_j 1$, which is $O(\log(N))$.

It is straightforward to check that Φ computes exactly the set of non-anamolous wavelet coefficients. (In fact, the output of Φ is often taken as the definition of the non-anomolous wavelet coefficients, particularly for wavelets more general than Haar wavelets, which are defined solely in terms of the two or more output streams of the basic filters.) Finally, it is easy to get Φ also to compute the anamolous wavelet coefficient and to attach appropriate wavelet indices to the terms, thereby producing wavelet terms.

2.2 Finding the Largest Terms

Given a sequence of wavelet terms, we want to keep just the B terms having largest coeffient absolute values. This can be done as follows. Create a buffer of size $4B$, and read the first B items into it. Now we maintain the invariant that our buffer has between B and $3B$ items, including the B largest items seen so far. We proceed as follows. Fill the buffer, by reading a number of items between B and $3B$. Next, ensure that the items all have unequal rank, *e.g.*, by ordering them primarily by coefficient size and secondarily by position in the buffer. Then find an approximate median—an item whose rank is between the 25'th and 75'th percentile. (This can be done deterministically in time $O(B)$ ([3]), or probabilistically by choosing a few random elements and testing them against the entire buffer. Finally, discard buffer elements smaller than the approximate median, restoring the invariant. The total time spent on this operation is $O(B)$ and it consumes $\Omega(B)$ items, so the time cost is $O(1)$ per item, as desired.

2.3 Analysis

It is clear by construction that the overall algorithm meets our claims. In fact, a more careful analysis shows that the worst-case work upon reading any item is at most $O(\log(N))$ for the first phase and $O(B)$ for the second phase. By using buffers of size $O(B + \log(N))$, one can arrange that the worst-case time per item is at most $O(1)$. This entire algorithm first appeared in [6] and is also discussed and used in [8].

3 Offline Histogram Algorithms

In this section, we show how to find optimal histograms in full space. We consider histograms in one dimension that are optimal for point queries.

3.1 A Quadratic Time Exact Algorithm

We use dynamic programming. Fix $k < N$ and $\ell \leq B$. Assume that, for each $i < k$, we have found the optimal $(\ell - 1)$-bucket histogram on the prefix of data indexed by $[0, i)$. To find the best ℓ-bucket histogram on $[0, k)$, we try all possibilities $m < k$ for the final boundary, and form a histogram by joining the best $(\ell - 1)$-bucket histogram on $[0, m)$ with the 1-bucket histogram on $[m, k)$. The time for this step is therefore at most N, to try all values of m. Since this has to be repeated for all $k < N$ and all $\ell \leq B$, the total time is $O(N^2 B)$. The space is $O(BN)$, to store a table of a B-bucket histogram for each i. (Space $O(N)$ suffices if one is willing to spend more time recomputing best histograms.)

This procedure works because we can decompose the error on an overall histogram into contributions from histogram fragments, in which the error attributed to a fragment depends only on the data values local to that fragment. One can substitute other error measures for sum square error in the overall algorithm. This dynamic program explicitly appears in [5].

3.2 A Linear Time Approximation Algorithm

If $B \ll N$, one can get a much faster approximate result as follows. First, as a preprocessing step, build the *prefix array* $\mathcal{P}(\mathbf{A}^2)$ of $\mathbf{A}^2$, where $\mathcal{P}(\mathbf{A}^2)[j] = \sum_{0 \leq i < j} \mathbf{A}^2[i]$, for $0 \leq j \leq N$. This step takes time $O(N)$ and produces $\mathcal{P}(\mathbf{A}^2)$, of size N. From $\mathcal{P}(\mathbf{A}^2)$, we can, in time $O(1)$, given ℓ and r, compute $\sum_{\ell \leq i < r} \mathbf{A}^2[i]$. Thus, given a candidate histogram bucket $c\chi_I$, we can, in time $O(1)$, compute the error on I of using that bucket.

If E_{opt} denotes the error of an optimal histogram, assume that we know an E with $E_{\text{opt}} \leq E < 2E_{\text{opt}}$. Note that, if the data values are integers and $\|\mathbf{A}\|^2 \leq M$, then the error is at least $\Omega(1)$ and at most M, and there are only $O(\log(M))$ possible powers of 2 in this range. We can find the right power of 2 for E time $\log\log(M)$ by binary search.

Fix $\ell < B$ and integer $k \leq 2B/\epsilon$. Suppose for each integer $i \leq 2B/\epsilon$ we know an m such that that there is an $(\ell - 1)$-bucket histogram on $[0, m)$ with error at most $(i + \ell - 1)(\epsilon/(2B))E$ and no histogram with error at most $i(\epsilon/(2B))E$. We want to find the largest m' such that there's an ℓ-bucket histogram on $[0, m')$ with error at most $(k + \ell)(\epsilon/(2B))E$ and no histogram with error at most $k(\epsilon/(2B))E$. To do this, try all possible values of i and find, by binary search, the largest m' such that the one-bucket histogram on $[m, m')$ has error at most $(k - i + 1)(\epsilon/(2B))E$ on $[m, m')$. This combined with the best $(\ell - 1)$-bucket histogram on $[0, m)$ with error at most $(i + \ell - 1)(\epsilon/(2B))E$ is an ℓ-bucket histogram on $[0, m')$ with error at most $(k + \ell)(\epsilon/(2B))E$, as desired. On the other hand, suppose the optimal ℓ-bucket histogram on $[0, m')$ has error less than $k(\epsilon/(2B))E$. It has some last boundary, m^*. There is some error E^* on $[0, m^*)$. Find the smallest i^* such that $E^* < i^*(\epsilon/(2B))E$, so that $(i^* - 1)(\epsilon/(2B))E \leq E^*$. Then the error on $[m^*, m')$ is less than $(k - i^* + 1)(\epsilon/(2B))E$.

When we considered $(\ell-1)$-bucket histograms and $i = i^*$, we will, inductively, have found a boundary m such that $[0, m)$ has no $(\ell-1)$-bucket histogram with error $i(\epsilon/(2B))E = i^*(\epsilon/(2B))E$; it follows that $m^* \leq m$. Also, we will have a histogram on $[0, m)$ with error at most $(i^* + \ell - 1)(\epsilon/(2B))E$ and the error on $[m, m')$ is less than $(k - i^* + 1)(\epsilon/(2B))E$, since that is a strict upper bound for the error on the interval $[m^*, m') \supseteq [m, m')$. Thus, when we combine the two histograms, we get total error less than $(k + \ell)(\epsilon/(2B))E$. It follows that our search procedure will find an ℓ-bucket histogram on $[0, m)$ with error less than $(k + \ell)(\epsilon/(2B))E$.

Now, find the k such that, for $\ell = B$, we have a histogram on $[0, N)$ with error at most $(k+\ell)(\epsilon/(2B))E = (k(\epsilon/(2B))+\epsilon/2)E$ and there is none with error $k(\epsilon/(2B))E$. That is, our histogram is within the factor $(1 + B/k)$ of optimal. Also, $(k(\epsilon/(2B)) + \epsilon/2)E \geq E_{\text{opt}} \geq E$, whence $k \geq (2B/\epsilon)(1 - \epsilon/2) \geq B/\epsilon$, provided $\epsilon \leq 1$. It follows that our histogram is within the factor $(1 + \epsilon)$, as desired.

Note that, other than producing $\mathcal{P}(\mathbf{A}^2)$, the algorithm runs in time $O(\log\log(M)B^3 \log(N)/\epsilon)$, since each step above needs $O(\log(N))$ time for binary search and $O(B/\epsilon)$ time to try all i. We need to repeat the above procedure for each of B possible ℓ's and each of $O(B/\ell)$ possible k's; then, at toplevel, repeat the whole thing $\log\log(M)$ times to search for E. Similarly, other than $\mathcal{P}(\mathbf{A}^2)$, the algorithm requires space $O(B^2/\epsilon)$ to store a B-bucket histogram for each of $O(B/\ell)$ possible k's. That is, this algorithm meets our desired time bound of $c_1N + (B\log(N)\log(M)/\epsilon)^{c_2}$ but not our desired space bound for dynamic data situations.

The algorithm above is somewhat reminiscent of ones in [9, 10] where the authors use it in the ordered aggregate mode.

4 Basic Histograms from Wavelet Representations

In this section, we dicuss relationships between histograms and wavelet representations.

4.1 Wavelet Representations as Histograms

First note that any histogram element χ_I on interval I can be expressed as a $2\log(N)$-term wavelet representation. This is because $\langle \chi_I, \psi \rangle = 0$ unless $\text{supp}(\psi)$ intersects an endpoint of I. Each endpoint of I is in the support of $\log(N) + 1$ wavelet vectors, but the two wavelet vectors with support $[0, N)$ are double counted, so only $2\log(N)$ terms remain. On the other hand, any wavelet term is itself a 4-bucket histogram (*i.e.*, a histogram with 3 boundaries), so a B-term wavelet representation can be viewed as a $(3B + 1)$-bucket histogram.

It follows that the best $2\log(N)B$-term wavelet representation $\mathbf{R}$ is at least as good as the best B-bucket histogram. Also, $\mathbf{R}$ can be regarded as a $(6\log(N)+1)$-bucket histogram. It follows that we can find a $(O(\log(N)), 1)$-approximation to the best B-bucket histogram in linear time using previously discussed algorithm for computing the wavelet summary.

4.2 Wavelet Representations as Intermediate Summaries

Next, we consider the following algorithm: Given $\mathbf{A}$, let $\mathbf{R}$ be the best $2\log(N)B$-term wavelet representation to $\mathbf{A}$, as above. Thus, as above, $\|\mathbf{A}-\mathbf{R}\|^2 \leq \|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|^2$, where $\mathbf{H}_{\mathrm{opt}}$ is the optimal B-bucket histogram for $\mathbf{A}$. Next, let $\mathbf{H}$ be the best B-bucket histogram to $\mathbf{R}$. How good is $\mathbf{H}$ as a summary of $\mathbf{A}$?

By the triangle inequality and by optimality of $\mathbf{H}$ for $\mathbf{R}$, we have

$$\begin{aligned}\|\mathbf{A}-\mathbf{H}\| &\leq \|\mathbf{A}-\mathbf{R}\| + \|\mathbf{R}-\mathbf{H}\| \\ &\leq \|\mathbf{A}-\mathbf{R}\| + \|\mathbf{R}-\mathbf{H}_{\mathrm{opt}}\| \\ &\leq \|\mathbf{A}-\mathbf{R}\| + \|\mathbf{R}-\mathbf{A}\| + \|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| \\ &\leq 3\,\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|,\end{aligned}$$

whence $\|\mathbf{A}-\mathbf{H}\|^2 \leq 9\,\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|^2$. Thus we have a $(1,9)$-approximation algorithm to the best B-bucket histogram.

Now consider the cost to build $\mathbf{H}$. It is easy to check that the boundaries of $\mathbf{H}$ can be chosen from among the boundaries of $\mathbf{R}$. This is proved in [8]. One can then easily modify the quadratic time algorithm of Section 3.1 to run in time B times the square of the number $O(B\log(N))$ of boundaries in $\mathbf{R}$, for a total of $O(B^3\log^2(N))$. Alternatively, one can modify the algorithm of Section 3.2 by building, in time linear in $B\log(N)$, a structure that will answer $\sum_{\ell\leq i<r}\mathbf{R}[i]^2$ in time $O(1)$.

4.3 Robust Representations

In Section 4.2, we used the triangle inequality to bound $\|\mathbf{A}-\mathbf{H}\|$ by $\|\mathbf{A}-\mathbf{R}\|+\|\mathbf{R}-\mathbf{H}\|$. This is not tight in the sense that $\|\mathbf{A}-\mathbf{R}\|+\|\mathbf{R}-\mathbf{H}\|$ might actually be as big as $3\,\|\mathbf{A}-\mathbf{H}\|$. In this section, we will define and construct a *robust* representation $\mathbf{R}_{\mathrm{rob}}$ with the property that $\|\mathbf{A}-\mathbf{H}\|^2 \approx \|\mathbf{A}-\mathbf{R}_{\mathrm{rob}}\|^2 + \|\mathbf{R}_{\mathrm{rob}}^2-\mathbf{H}\|^2$ and $\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|^2 \approx \|\mathbf{A}-\mathbf{R}_{\mathrm{rob}}\|^2 + \|\mathbf{R}_{\mathrm{rob}}^2-\mathbf{H}_{\mathrm{opt}}\|^2$, where, as above, $\mathbf{H}$ is optimal for $\mathbf{R}_{\mathrm{rob}}$ and $\mathbf{H}_{\mathrm{opt}}$ is optimal for $\mathbf{A}$. This will allow us to compare $\|\mathbf{A}-\mathbf{H}\|$ with $\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|$, as above, but without giving up anything to the triangle inequality. (Instead, we give up a lot less in the approximation to the Pythagorean equality.)

The following theorem relates a wavelet representation to a histogram in a more subtle manner.

Theorem 1 *For some $T \leq O(1/(\epsilon^2\log(1/\epsilon)))$ and any signal $\mathbf{A}$, let $\mathbf{R}_{\mathrm{rob}}$ be the representation formed by greedily taking $B\log(N)$ wavelet terms at a time until either*

- *we get $TB\log(N)$ terms, or*
- *taking an additional $B\log(N)$ wavelet terms improves (reduces) the residual $\|\mathbf{A}-\mathbf{R}_{\mathrm{rob}}\|_2^2$ by a factor no better than $(1-\epsilon')$, where $\epsilon' \geq \Omega(\epsilon^2)$.*

Let $\mathbf{H}$ be the best B-bucket representation to $\mathbf{R}_{\mathrm{rob}}$. Then $\mathbf{H}$ is a $(1, 1+\epsilon)$-approximation to the best B-bucket histogram to $\mathbf{A}$.

Proof: Suppose $\mathbf{R}_{\text{rob}}$ has $TB\log(N)$ terms. Then

$$\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|_2^2 \le (1-\epsilon')^{(T-1)} \left\|\mathbf{A}-\mathbf{R}^{(B\log(N))}\right\|_2^2,$$

where $\mathbf{R}^{(B\log(N))}$ is the best representation of $B\log(N)$ terms. By choice of $T = 1 + (1/\epsilon')\log(25/\epsilon^2)$, we have $(1-\epsilon')^{(T-1)} \approx \epsilon^2/25$ and, by previous observations, $\left\|\mathbf{A}-\mathbf{R}^{(B\log(N))}\right\| \le \|\mathbf{A}-\mathbf{H}_{\text{opt}}\|$, and

$$\begin{aligned} \|\mathbf{A}-\mathbf{H}\|_2 &\le \|\mathbf{A}-\mathbf{H}_{\text{opt}}\|_2 + 2\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|_2 \\ &\le (1+2\epsilon/5)\|\mathbf{A}-\mathbf{H}_{\text{opt}}\|, \end{aligned}$$

so that $\|\mathbf{A}-\mathbf{H}\|_2^2 \le (1+\epsilon)\|\mathbf{A}-\mathbf{H}_{\text{opt}}\|_2^2$ for sufficiently small ϵ.

Now suppose $\mathbf{R}_{\text{rob}}$ has fewer than T terms, so that $\mathbf{R}_{\text{rob}}$ together with any additional $B\log(N)$ terms has error at least $(1-\epsilon')\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|_2^2$. It follows that the $B\log(N)$ terms in $\mathbf{H}$ or $\mathbf{H}_{\text{opt}}$ do not improve the square error of $\mathbf{R}_{\text{rob}}$ by more than the factor $(1-\epsilon')$. Let $\widehat{\mathbf{H}}$, regarded as a histogram or wavelet representation, be the best linear combination of $\mathbf{R}_{\text{rob}}$ and $\mathbf{H}$; our hypothesis is that $\left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 \ge (1-\epsilon')\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 = \|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 - \epsilon'\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2$. Note that, since $\widehat{\mathbf{H}}$ is the best representation on a specified line (containing $\mathbf{R}_{\text{rob}}$ and $\mathbf{H}$), it follows that there's a right angle at $\widehat{\mathbf{H}}$, so that, as in Figure 1, we have

$$\begin{aligned} \left\|\mathbf{R}_{\text{rob}}-\widehat{\mathbf{H}}\right\|^2 &= \|\mathbf{R}_{\text{rob}}-\mathbf{A}\|^2 - \left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 \\ &\le \epsilon'\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 \\ &\le (\epsilon^2/9)\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2. \end{aligned}$$

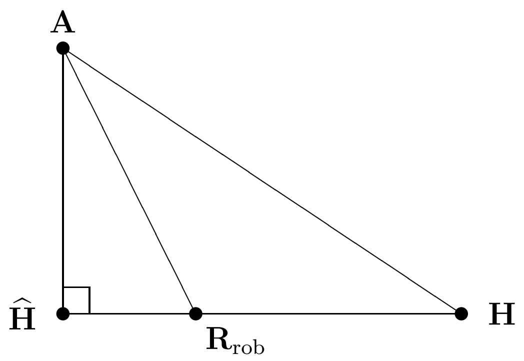
We have, using Cauchy-Schwarz in the penultimate line,

$$\begin{aligned} \|\mathbf{A}-\mathbf{H}\|^2 &= \left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 + \left\|\widehat{\mathbf{H}}-\mathbf{H}\right\|^2 \\ &= \left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 + \left(\|\mathbf{R}_{\text{rob}}-\mathbf{H}\| \pm \left\|\mathbf{R}_{\text{rob}}-\widehat{\mathbf{H}}\right\|\right)^2 \\ &= \left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2 \pm 2\|\mathbf{R}_{\text{rob}}-\mathbf{H}\|\left\|\mathbf{R}_{\text{rob}}-\widehat{\mathbf{H}}\right\| + \left\|\mathbf{R}_{\text{rob}}-\widehat{\mathbf{H}}\right\|^2 \\ &= \|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2 \pm 2\|\mathbf{R}_{\text{rob}}-\mathbf{H}\|\left\|\mathbf{R}_{\text{rob}}-\widehat{\mathbf{H}}\right\| \\ &\subseteq \|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2 \pm 2(\epsilon/3)\|\mathbf{R}_{\text{rob}}-\mathbf{H}\|\|\mathbf{A}-\mathbf{R}_{\text{rob}}\| \\ &\subseteq \|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2 \pm (\epsilon/3)\left(\|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2 + \|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2\right) \\ &= (1\pm\epsilon/3)\left(\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2\right). \end{aligned}$$

Above we used only the robustness of $\mathbf{R}_{\text{rob}}$ and the fact that $\mathbf{H}$ is a B-bucket histogram. That is, for *any* B-bucket histogram $\mathbf{H}$, robustness of $\mathbf{R}_{\text{rob}}$ insures that there is a near-right-angle at $\mathbf{R}_{\text{rob}}$, so that

$$\|\mathbf{A}-\mathbf{H}\|^2 = (1\pm\epsilon/3)\left(\|\mathbf{A}-\mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}}-\mathbf{H}\|^2\right).$$

Figure 1: Illustration of histograms in Theorem 1. The representations $\widehat{\mathbf{H}}, \mathbf{R}_{\text{rob}}$, and $\mathbf{H}$ are colinear, but $\mathbf{R}_{\text{rob}}$ may or may not be between $\widehat{\mathbf{H}}$ and $\mathbf{H}$. By optimality of $\widehat{\mathbf{H}}$, there is a right angle as indicated.



Thus, similarly,

$$\|\mathbf{A} - \mathbf{H}_{\text{opt}}\|^2 = (1 \pm \epsilon/3)(\|\mathbf{A} - \mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}} - \mathbf{H}_{\text{opt}}\|^2).$$

Note that $\mathbf{H}$ is an optimal B-bucket histogram for $\mathbf{R}_{\text{rob}}$ and $\mathbf{H}_{\text{opt}}$ is another B-bucket histogram; it follows that $\|\mathbf{R}_{\text{rob}} - \mathbf{H}\| \leq \|\mathbf{R}_{\text{rob}} - \mathbf{H}_{\text{opt}}\|$. Thus we have

$$\|\mathbf{A} - \mathbf{H}\|^2 \leq (1 + \epsilon/3) \left(\|\mathbf{A} - \mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}} - \mathbf{H}_{\text{opt}}\|^2\right)$$

and

$$\|\mathbf{A} - \mathbf{H}_{\text{opt}}\|^2 \geq (1 - \epsilon/3) \left(\|\mathbf{A} - \mathbf{R}_{\text{rob}}\|^2 + \|\mathbf{R}_{\text{rob}} - \mathbf{H}_{\text{opt}}\|^2\right).$$

Combining, we get

$$\|\mathbf{A} - \mathbf{H}\|_2^2 \leq \frac{1 + \epsilon/3}{1 - \epsilon/3} \|\mathbf{A} - \mathbf{H}_{\text{opt}}\|_2^2 \leq (1 + \epsilon) \|\mathbf{A} - \mathbf{H}_{\text{opt}}\|_2^2,$$

for sufficiently small ϵ, as desired.

∎

In the ordered aggregate data model, the properties we have proved above immediately lead to an algorithm. We analyze the cost of constructing $\mathbf{R}_{\text{rob}}$. While making a single pass over the ordered aggregate data, in parallel, we find the biggest $TB\log(N)$ terms and we compute $\|\mathbf{A}\|^2$. Both tasks can be done in time $O(N)$ and space $O(TB\log(N))$. Next we build $\mathbf{R}_{\text{rob}}$ from our collection of $TB\log(N)$ wavelet terms in the straightforward way. We will need to track $\|\mathbf{A} - \mathbf{R}_{\text{rob}}\|^2$ as $\mathbf{R}_{\text{rob}}$ changes, but this can be done using the formula $\|\mathbf{A} - \mathbf{R}_{\text{rob}}\|^2 = \|\mathbf{A}^2\| - \|\mathbf{R}_{\text{rob}}\|^2$. Thus we obtain $1 + \epsilon$ approximation to the histogram in the ordered aggregate model with space as desired. This result appears in [8].

5 Histograms and Wavelets with Dynamic Data

In this section, we briefly consider data that is subject to dynamic update. We assume that we have a synopsis data structure, called an "L^2 count sketch," parametrized by N and η, that efficiently supports the following operations on a(n unordered) set S of size N:

- UPDATES of the form "add x to a count $\mathbf{C}[i]$ for item $i \in S$;"
- Unparametrized "FIND" queries. The answer is a list containing all j where $\mathbf{C}[j]^2 \geq \eta \sum_i \mathbf{C}[i]^2$.
- "COUNT" queries about item i, for which the answer is $\widetilde{\mathbf{C}}[i]$ such that $\left|\widetilde{\mathbf{C}}[i] - \mathbf{C}[i]\right|^2 \leq \eta \|\mathbf{C}\|^2$.

There are randomized implementations of L^2 count sketches [7, 4] that need space $(\log(N)/\eta)^{O(1)}$ and time $(\log(N)/\eta)^{O(1)}$ for either query. In particular, a FIND query can only return $(\log(N)/\eta)^{O(1)}$ values. (There is also mild but necessary dependence on the range M of counts and on the failure probability of this randomized object. We do not discuss that detail further.)

Given access to an L^2 count sketch data structure with parameter η that depends polynomially on $B\log(N)/\epsilon$, we now show how to modify the previous algorithms to produce near-optimal histogram and wavelet representations.

Recall that a signal is equivalently specified by its full set of $O(\log(N))$ wavelet coefficients. Given input of the form "add x to $\mathbf{A}[i]$," we form $O(\log(N))$ additive updates to the wavelet coefficients of $\mathbf{A}$. We then use the L^2 count sketch on the wavelet coefficients. That is how we process updates.

To build a summary in our algorithms above, we need to find the top few wavelet terms. Using the L^2 count sketch, we proceed as follows:

- Do a FIND query to find one or more wavelet index j with potentially large coefficient.
- Do COUNT queries to get an approximation $\widetilde{c}_j$ to the coefficient(s).
- Do an update of $-c_j\psi_j$; that is subtract c_j from the count of wavelet index j.

In general, we need to repeat this process several times to get enough wavelet terms ($TB\log(N)$ iterations will certainly suffice, though fewer iterations will also work). If we find a coefficient approximation $\widetilde{c}_j$ whose square is relatively large, after we subtract it off, other coefficients (as well as the residual coefficient $c_j - \widetilde{c}_j$ will appear relatively larger, since the sum of squares of coefficients has shrank.

Using just the L^2 count sketch, we cannot get exact values for coefficients and therefore we cannot compare terms whose coefficients are close in magnitude to choose the indices with largest coefficients. Nevertheless, one can show that the error in coefficients is dominated by the unavoidable error $\|\mathbf{A} - \mathbf{R}_{\text{opt}}\|$ and we will only choose a suboptimal index j if it displaces another index j' whose coefficient, $c_{j'}$, is only marginally better than c_j.

It follows that we can still build an $\mathbf{R}_{\text{rob}}$ with the property that

- taking an additional $B\log(N)$ wavelet terms and changing all the coefficients to their ideal values reduces the residual $\|\mathbf{A}-\mathbf{R}_{\mathrm{rob}}\|_2^2$ by a factor no better than $(1-\epsilon')$, where $\epsilon' \geq \Omega(\epsilon^2)$.

This property suffices for the rest of the constructions—we can either take the top B terms for a wavelet representation or take the best B-term histogram to $\mathbf{R}_{\mathrm{rob}}$. The cost in space, update time, and query time for building a representation this way is as claimed. This follows from the cost guarantees assumed about the L^2 count sketch. What we have outlined is the procedure in [7].

6 Generalized Histograms

In this section, we discuss generalizing histograms slightly. First, we consider representations that are piecewise *linear* instead of piecewise *constant*. Next, we show how to find piecewise constant histograms that are optimal for *range* queries (defined below) instead of point queries. Finally, we will discuss *multidimensional* histograms.

6.1 Piecewise-Linear Representations

A B-piece piecewise-linear representation of a vector on $[0,N)$ consists of a partition of $[0,N)$ into B intervals and, for each interval, an average height h_j and a slope s_j. Thus a histogram is a piecewise-linear representation in which all slopes are zero. The answer vector associated with a piecewise-linear representation is as follows. To answer a query at position i, find the bucket j such that $b_j \leq i < b_{j+1}$. Let $m_j = (b_j + b_{j+1} - 1)/2$, an integer or half-integer, be the midpoint of the bucket. Return the value $h_j + s_j(i - m_j)$. (For example, the midpoint of the 4-point bucket $[0,4) = \{0,1,2,3\}$ is 1.5.)

To build a near-optimal piecewise-linear representation, for ordered aggregate or dynamic data, we follow the above outline but replace "Haar wavelets" with "linear multiwavelets," that we define next. Like the set of Haar wavelet vectors of length N, the linear multiwavelet vectors form an orthonormal family of N vectors. They consist of the appropriate scaling of the following:

- An anomolous linear multiwavelet vector $\chi_{[0,N)}$. (This is called a *constant*. The constant is also the one anomalous Haar wavelet vector.)

- An anomolous linear multiwavelet vector that takes the value $i-(N-1)/2$ at position i. (That is, the positive-slope normalized linear function of i that is orthogonal to $\chi_{[0,N)}$. This is called a *slope*.)

- On each dyadic interval $I = L \cup R$ of size at least 4 consisting of left half L and right half R with midpoint m, there is a linear multiwavelet that is a linear function on each of L and R and has even symmetry with respect to m. By symmetry, this linear multiwavelet is orthogonal to the slopes; the height is chosen so that it is orthogonal to the constants, too. We call this a *vee*.

- On each dyadic interval I as above, find a vector that is orthogonal to the constants, slopes, and the vee on I. This one is called an *odd*. (It has odd symmetry about the midpoint of I and is linear on each of L and R.)

By construction, the constant and slope are orthogonal, each vee and each odd are orthogonal to each constant and each slope and each pair of vee and odd on the *same* dyadic interval are orthogonal. Furthermore, if two linear multiwavelet vectors are supported on disjoint intervals, then they are obviously orthogonal. Finally, suppose two dyadic intervals I_1 and I_2 intersect and are unequal. Then, by properties of the dyadic intervals, one interval properly contains the other; say $I_1 \subseteq I_2$. Suppose ψ_1 is a vee or odd associated with I_1 and ψ_2 is a vee or odd associated with I_2. Then ψ_2 restricted to I_1 is a linear function, *i.e.*, a linear combination of a constant and a slope. It follows that ψ_1 and ψ_2 are orthogonal. It is easy to verify that the family consists of N vectors, whence it is an orthogonal basis.

We now confirm that the above algorithms can be modified in a straightforward way to produce piecewise-linear representations. We need to check each of the following easy facts.

- Each data index is in $O(\log(N))$ dyadic intervals of length at least 4, so is in the support of $O(\log(N))$ linear multiwavelets.

- Each multiwavelet vector is a piecewise-linear vector of $O(1)$ pieces.

- Each element of a piecewise linear representation (*i.e.*, for each bucket j, the vector that results from setting to zero all heights and slopes *other* than the j'th height and slope) is the linear combination of the $O(\log(N))$ linear multiwavelets whose support intersects an endpoint of the bucket.

- The transform that takes N data points to the $N-2$ non-anomalous linear multiwavelet coefficients can be expressed as a filter Φ that consists of cascaded constant-work filters Φ_j at level j, $0 \leq j < \log(N)$, and the anomolous coefficients can also be computed from ordered aggregate data.

- To build the best B-bucket piecewise-linear representation $\mathbf{H}$ to a sparse linear multiwavelet representation $\mathbf{R}_{\text{rob}}$, use the technique of Section 3.2. (Alternatively, one could use the technique of Section 3.1, with the modification that boundaries of $\mathbf{H}$ may be chosen from a small set S that includes not only the boundaries of $\mathbf{R}_{\text{rob}}$, but also additional boundaries, so that, for each bucket $[b_j, b_j+1)$ and each pair of consecutive points $x, y \in S$, either $y = x+1$ or $(y-b_j) \leq (1+\epsilon')(x-b_j)$ and $(b_{j+1}-x) \leq (1+\epsilon')(b_{j+1}-y)$, where ϵ' is proportional to $\epsilon/\log(N)$.)

Finally, we note that these techniques can be used more directly to produce linear multiwavelet representations. While these are less common than Haar wavelets, histograms, or piecewise-linear representations, they are a natural variant in our context. The preceding discussion was technically developed in [12].

6.2 Range Queries

To this point, we have discussed only building histograms that are nearly optimal for point queries. We now consider range queries. That is, at the user level, a query is a pair of points (ℓ, r) and the ideal answer is $\mathbf{A}[\ell, r) = \sum_{\ell \le i < r} \mathbf{A}[i]$. An answer from a (piecewise-constant) histogram $\mathbf{H}$ would be $\mathbf{H}[\ell, r) = \sum_{\ell \le i < r} \mathbf{H}[i]$. The sum square error of a histogram is

$$\sum_{\ell, r} \left| \mathbf{A}[\ell, r) - \mathbf{H}[\ell, r) \right|^2 .$$

Note that, for any approximate summary, the empty query $[\ell, \ell)$ is always answered perfectly as 0. We take the convention that $\mathbf{A}[r, \ell) = -\mathbf{A}[\ell, r)$. It then follows that we may equivalently take the sum over all queries (ℓ, r) or require that $\ell < r$ or $\ell \le r$; the sum square error is exactly twice that in the latter two cases.

Building histograms for range queries is less straightforward than building histograms for point queries. For point or range queries, note that the data can essentially fix the bucketing. For example, if the data alternates between runs of values between 0 and 10 and runs of values between 1,000,000 and 1,000,010, then the bucketing must respect the runs of data—anything else would be awful. For range quereis, even for a fixed bucketing, it is not obvious how to choose the bucket heights. This is because the optimal height for bucket j depends on other bucket heights and data outside the bucket—for example, if buckets $j - 1$ and $j + 1$ underestimate the data, we may want bucket j to overestimate the data, to give net error contribution zero of the three buckets to overflying queries. Within a single bucket, there is a tension between making the bucket height equal to the average of the data in the bucket (to avoid contributing error to overflying queries) and setting the height to optimize within-bucket queries. For example, if the data in a bucket of size s is all zero except for a -1 near an endpoint and a $+1$ near the center, then the average data value in the bucket is 0. But roughly half the intra-bucket range queries have answer $+1$, roughly half have answer 0, and $O(s)$ of the $\Theta(s^2)$ intra-bucket queries have answer -1, so, from the perspective of intra-bucket queries, the ideal height will be positive. Unlike the situation for point queries, we know of no polynomial time algorithm, even using full space, that finds the optimal B-bucket histogram for range queries.

Nevertheless, we can easily transform this problem so that the previously described approximation algorithms will work on it. Specifically, we take the original data, $\mathbf{A}$, and transform to the prefix array, $\mathcal{P}(\mathbf{A})$. Under this transform, a piecewise-constant representation transforms to a piecewise linear connected representation and the square error summed over all range queries (ℓ, r) transforms to N times the sum square error over all point queries. That is, to get a piecewise-constant representation that minimizes the sum square error of range queries to $\mathbf{A}$, we only need to find a piecewise linear *connected* representation $\mathbf{H}$ that minimizes the sum square error of point queries to $\mathcal{P}(\mathbf{A})$, then output $\Delta(\mathbf{H})$, where $\Delta(\mathbf{H})[i] = \mathbf{H}[i + 1] - \mathbf{H}[i]$. Thus Δ and $\mathcal{P}$ are opposite operations. Also note that, if the length of $\mathbf{A}$ is N, then the length of $\mathcal{P}(\mathbf{A})$ is $N + 1$ and $\mathcal{P}(\mathbf{A})[0]$ is always 0.

Unfortunately, we do not know how to find optimal piecewise linear *connected* representations—the transformed problem is completely equivalent to the original. But we do know how to find nearly optimal representations from the larger class general piecewise-linear B-bucket representations, not necessarily connected. Formally, recall that, if $|\mathbf{A}| = N$, then $|\mathcal{P}(\mathbf{A})| = N+1$ and $\mathcal{P}(\mathbf{A})[0] = 0$; we find a piecewise-linear representation only for the N values $\mathcal{P}(\mathbf{A})[1, N]$. Furthermore, one can show that if the bucket heights are optimal in the piecewise-linear representation (it is easy to make both the heights and slopes optimal), then the correspondence is preserved between the sum square error over point queries to piecewise-linear $\mathbf{H}$ and the sum square error over range queries to $\Delta(\mathbf{H})$. Finally, we consider $\Delta(\mathbf{H})$. Since $\mathbf{H}$ is not connected, $\Delta(\mathbf{H})$ is not necessarily a B-bucket piecewise-constant representation, but it is a $(2B-1)$-bucket piecewise-constant representation, where the discontinuities in $\mathbf{H}$ each give rise to a bucket of size one in $\Delta(\mathbf{H})$.

Next, we consider the changes necessary to our algorithms in order to realize the desired transform. It is easy to perform the Δ operation in post-processing on a small object like $\mathbf{H}$. As for pre-processing, in the ordered aggregate model, we can easily convert $\mathbf{A}$ to $\mathcal{P}(\mathbf{A})$ in time $O(1)$ per item by a pipe before proceeding to compute linear multiwavelet coefficients. In the dynamic model, note that an update to $\mathbf{A}[i]$ transforms to an update to each $\mathcal{P}(\mathbf{A})[j]$ for $j > i$; *i.e.*, to the function $\chi_{[i+1,N+1)}$. In turn, this causes an update to several linear multiwavelets, but only those whose support intersects an endpoint of $[i+1, N+1]$; *i.e.*, the $O(\log(N))$ of the linear multiwavelets whose support contains $i+1$. The rest of the algorithms are unchanged.

It follows that we can get a

1. $(2, 1)$ approximation in polynomial time and full space using the techniques of Section 3.1;

2. $(2, 1+\epsilon)$-approximation for ordered aggregate data in time $c_1 N + (B \log(N) \log(M)/\epsilon)^{c_2}$ and space $(B \log(N) \log(M)/\epsilon)^{c_2}$ using the techniques of Section 4.3; and a

3. $(2, 1+\epsilon)$-approximation for dynamic data in time and space $(B \log(N) \log(M)/\epsilon)^{c_2}$ using the techniques of Section 5.

Finally, note that, because the resulting $2B$-bucket histogram has alternate buckets of size 1, we only need to store B boundaries and $2B$ heights. Arguably, the storage requirement is only 1.5 times that of a general B-bucket histogram.

We now describe briefly a $(1, 1+\epsilon)$-approximation in polynomial time and full space using an offline algorithm. The dependence on N is approximately quadratic for time and linear for space. First we build a robust representation $\mathbf{R}_{\text{rob}}$ for $\mathcal{P}(\mathbf{A})$ in linear time, as above. Then we want to find the best piecewise-constant histogram for range queries to $\Delta(\mathbf{R}_{\text{rob}})$. We now show how to do this, modifying the techniques of Section 3.2.

We can regard the dynamic program of Section 3.2 as asking, "What is a prefix $[0, m)$ that has a representation with ℓ buckets and error budget approximately k (in appropriate units)?" Knowing the answer for a given ℓ and all k lets us answer the question for $\ell + 1$ and all k. The cost depended on the range and precision needed for the error budget; in Section 3.2, the error could be at most $2E_{\text{opt}}$ and units of $(\epsilon/(2B))E_{\text{opt}}$ suffice. To generalize for range queries, we will need to track an additional quantity. Specifically, define the raw suffix error of a partial histogram $\mathbf{H}$ on $[0, m)$ to be $\sum_{\ell<m} \mathbf{A}[\ell, m) - \mathbf{H}[\ell, m)$. Observe that if we know the raw suffix error σ and the sum square error E for a prefix $[0, m)$, that, together with local information in a potential new bucket $[m, m')$, suffices to compute the two quantities for the new prefix $[0, m')$. Specifically, the new suffix raw error is a linear combination of σ, the overflight error $\mathbf{A}[m, m') - \mathbf{H}[m, m')$ and the suffix error σ_1' within the bucket $[m, m')$; and the new sum square error is a linear combination of E, the sum square error of queries within $[m, m')$, and $\sigma_1\pi_1$, where π_1 is the analogously-defined prefix error within $[m, m')$. To see this, one can trace the contribution to the three quantities of a single range query, whose endpoints may be in $[0, m), [m, m')$, or $[m', N)$. The coefficients in the linear combinations mentioned above depend only on m, m', and N; they account for the number of ranges making contributions.

So, by analogy with Section 3.2, we ask, "What is a prefix $[0, m)$ that has a representation with ℓ buckets, sum raw suffix error budget approximately i, and sum square error budget approximately k?" As above, k needs to take values up to $O(E_{\text{opt}})$ in units of $\Omega((\epsilon/B)E_{\text{opt}})$. By the Cauchy-Schwartz inequality, one can show that the sum raw suffix error is at most $\sqrt{2NE_{\text{opt}}}$, since, otherwise, these at-most-N suffixes would contribute more than $2E_{\text{opt}}$ to the total sum square error. Observe that the sum raw suffix error ultimately contributes to the sum square error only by multiplying a sum raw prefix error, which is also bounded by $\sqrt{2NE_{\text{opt}}}$. It follows that we only need precision $\Omega((\epsilon/B)\sqrt{E_{\text{opt}}/N})$ for sum raw suffix errors in order that the B quantization errors total $O(B \cdot ((\epsilon/B)\sqrt{E_{\text{opt}}/N}) \cdot \sqrt{NE_{\text{opt}}}) = O(\epsilon E_{\text{opt}})$. Thus there are $O(\sqrt{NE_{\text{opt}}}/((\epsilon/B)\sqrt{E_{\text{opt}}/N})) = O(BN/\epsilon)$ possible values for i, a number polynomial in BN/ϵ, as desired. Since there are B/ϵ possibilities for k and B possibilities for ℓ, the total space is the product, B^3N/ϵ^2. As for the time, we need to try all values of $i' < i$ and $k' < k$ to compute the (i, k)'th value, we need to do binary search to find m, and binary search to find a factor 2 approximation to E_{opt}, so the time is $B^5N^2 \log(N) \log\log(M)/\epsilon^4$. Improvements to the running time will be of great interest. These results for rangesum appears in [12].

6.3 Multidimensional Histograms

Now we generalize the basic problem to multi, say two, dimensional arrays. The first issue that arises is what is a two dimensional histogram summary? One can think of partitions into rectangles, squares and other shapes, or covers (ie., allow shapes to overlap) in which case we have a variety of ways to intrepret the approximation in overlapped areas. Covering-basd histograms are discussed in [14] where the authors present polylogarithmic space algorithms, albeit with $\Omega(N^2)$ time. Here, we focus on partition-based histograms. Among these, there are a number of variations depending upon how the partition is done. The natural and general class is one of arbitrary partitioning into B axis-parallel rectangles.

Such two (and higher) dimensional problems are known to be NP-Hard, but, in two dimensions, a $(O(1), 1)$-approximation can be computed in polynomial time and space [11]. For example, the authors in [11] give an algorithm to produce the best *hierarchical* histogram. A *hierarchical* histogram is one with a hierarchical bucketing, i.e., in which the bucketing is obtained by repeatedly partitioning one of the existing buckets into two pieces. Since any bucketing of B buckets in two dimensions can be refined to a hierarchical bucketing of $4B$ buckets [2], a $(4,1)$-approximation to the general problem ensues.

In streaming context, consider $N \times N$ integer-valued signal $\mathbf{A}$. We can solve the B-bucket, $(4, 1+\epsilon)$-approximate two dimensional histogram problem using $(B \log \|\mathbf{A}\| \log(N)/\epsilon)^{O(1)}$ space, per-item time, and time per hist query. For static data, for constants c_1 and c_2, we can $(4, 1+\epsilon)$-approximate the best B-bucket histogram in time $c_1 N^2 + (B \log \|\mathbf{A}\| \log(N)/\epsilon)^{c_2}$ and space $(B \log \|\mathbf{A}\| \log(N)/\epsilon)^{c_2}$, making one pass over the data, with the order we specify to lay out $\mathbf{A}$. Both the results above are achieved by using techniques that have been developed in the context one dimensional histograms [7, 8], combining them with techniques known for offline multidimensional histogramming [11], and extending them. The extensions are proving that a hierarchical histogram approximation to robust multidimensional representation is in fact a good approximation to the best histogram and generalizing the wavelet based approach to tensor products, The technical details are in [13].

7 Concluding Remarks

Motivated by the streaming context, a few interesting techniques (eg., robustness, finding heavy coefficients) have been developed for histogram and wavelet summary computation, and in particular, their relationship has proved useful. Simpler algorithms for dynamic data will be of interest, in particular, for two dimensional histograms, even if for additions only (i.e., $x > 0$ in the updates). For demanding IP network applications, the current update times are challenging. There are two major problems that need new ideas. In one which we have a *workload* of π that for each i, gives weight $\pi[i]$ based on the point query being $\mathbf{A}[i]$. Then one would like to minimize the error weighted by π. The open problem is when both π and $\mathbf{A}$ are dynamic because updates modify $\mathbf{A}$ but interspersed queries modify π. In the other, the updates are *resets*, that is, $\mathbf{A}[i] \leftarrow x$. The problem is to maintain a suitable summary under resets. This is motivated by self-tuning database query processors [1].

References

[1] A. Aboulnaga, S. Chaudhuri: Self-tuning Histograms: Building Histograms Without Looking at Data. *Proc. ACM SIGMOD Conference*, 1999, 181-192.

[2] F. d'Amore and P. G. Franciosa. On the Optimal Binary Plane Partition for Sets of Isothetic Rectangles. *Information Processing Letters* 44(5): pp. 255–259, 1992.

[3] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *J. Computer and System Sciences*, 7:448-461, 1972.

[4] G. Cormode, S. Muthukrishnan: An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. LATIN 2004: 29-38.

[5] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal histograms with quality guarantees. *Proc. of the 1998 Intl. Conf. on Very Large Data Bases (VLDB)*, 275-286, 1998.

[6] A. Gilbert, Y. Kotidis, S. Muthukrishnan and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. *Proc. of the 2001 Intl. Conf. on Very Large Data Bases (VLDB)*, 79-88, 2001.

[7] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. *Proc. STOC*, 2002: 389-398

[8] S. Guha, P. Indyk, S. Muthukrishnan and M. Strauss. Histogramming data streams with fast per-item processing. *Proc. ICALP*, 2002. 681-692.

[9] S. Guha, N. Koudas and K. Shim. Data-streams and histograms. *Proc. of the 2001 Annual ACM Symp. on Theory of Computing (STOC)*, 471-475, 2001.

[10] S. Guha and N. Koudas. Approximating a data stream for querying and estimation: Algorithms and performance evaluation. *Proc. of the 2002 Intl. Conf. on Data Engineering (ICDE)*, 567-576, 2002.

[11] S. Muthukrishnan, V. Poosala, T. Suel. On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications. *Proc. ICDT*, 1999, 236-256.

[12] S. Muthukrishnan, M. Strauss. Rangesum histograms. *Proc. ACM-SIAM SODA*, 2003: 233-242.

[13] S. Muthukrishnan, M. Strauss. Maintenance of Multidimensional Histograms. *Proc. FSTTCS*, 2003, 352-362.

[14] N. Thaper, S. Guha, P. Indyk, N. Koudas. Dynamic multidimensional histograms. *Proc. ACM SIGMOD Conference*, 2002, 428-439.