Histogramming Data Streams with Fast Per-Item Processing

Sudipto Guha [*], Piotr Indyk [**], S. Muthukrishnan[***], and Martin J. Strauss[***]

No Institute Given

Keywords: histograms, streaming algorithms
Track A, approximation algorithms

Abstract. A vector $\mathbf{A}$ of length N can be approximately represented by a histogram $\mathbf{H}$, by writing $[0, N)$ as the non-overlapping union of B intervals I_j, assigning a value b_j to I_j, and approximating $\mathbf{A}_i$ by $\mathbf{H}_i = b_j$ for $i \in I_j$. An optimal histogram representation $\mathbf{H}_{\mathrm{opt}}$ consists of the choices of I_j and b_j that minimize the sum-square-error $\|\mathbf{A} - \mathbf{H}\|_2^2 = \sum_i |\mathbf{A}_i - \mathbf{H}_i|^2$. Numerous applications in statistics, signal processing and databases rely on histograms; typically B is (significantly) smaller than N and, hence, representing $\mathbf{A}$ by $\mathbf{H}$ yields substantial compression.

We give a deterministic algorithm that approximates $\mathbf{H}_{\mathrm{opt}}$ and outputs a histogram $\mathbf{H}$ such that

$$\|\mathbf{A} - \mathbf{H}\|_2^2 \leq (1+\epsilon) \|\mathbf{A} - \mathbf{H}_{\mathrm{opt}}\|_2^2 .$$

Our algorithm considers the data items $\mathbf{A}_0, \mathbf{A}_1, \ldots$ in order, *i.e.*, in one pass, spends processing time $O(1)$ per item, uses total space $B(\log(N) \log \|\mathbf{A}\| / \epsilon)^{O(1)}$, and determines the histogram in time $O((B \log(N) \log \|\mathbf{A}\| / \epsilon)^{O(1)}$. Our algorithm is eminently suitable to emerging applications where signal is presented in a stream, size of the signal is very large, and one must construct the histogram using significantly smaller space than the signal size. In particular, our algorithm is suited to high performance needs where the per-item processing time must be minimized. Previous algorithms either used large space, i.e., $\Omega(N)$, or worked longer, *i.e.*, $N \log^{\Omega(1)}(N)$ total time over the N data items. Our algorithm is the first that simultaneously uses small space as well as runs fast, taking $O(1)$ worst case time for per-item processing. In addition, our algorithm is quite simple.

1 Introduction

We study the problem of representing signals succinctly using histograms. The *signal* is a vector $\mathbf{A}$ of length N. A *histogram* $\mathbf{H}$ on the signal is obtained by

[*] Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, `sudipto@cis.upenn.edu`

[**] MIT Laboratory for Computer Science; 545 Technology Square, NE43-373; Cambridge, Massachusetts 02139-3594; `indyk@theory.lcs.mit.edu`

[***] AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA, {`muthu`, `mstrauss`}`@research.att.com`

writing $[0, N)$ as the non-overlapping union of B intervals I_j and assigning a value b_j to I_j. The histogram $\mathbf{H}$ can be used to approximately represent the signal $\mathbf{A}$ by approximating $\mathbf{A}_i$ as $\mathbf{H}_i = b_j$ for $i \in I_j$. Equivalently, a histogram is a piecewise constant approximation to the signal. Histogram $\mathbf{H}$ takes only $O(B)$ values to store, namely, the I_j's and b_j's, in contrast to the $O(N)$ values needed to store the signal $\mathbf{A}$. Typically $B \ll N$ in applications and thus $\mathbf{H}$ is a succinct representation for $\mathbf{A}$.

Histograms must nevertheless capture the trends in the signal. Numerous measures evaluate how well a histogram achieves this; the most common form is the sum-square-error, which measures the sum of the squares of the devation from the signal. An *optimal histogram* $\mathbf{H}_{\mathrm{opt}}$, sometimes written $\mathbf{H}_{\mathrm{opt}}^B$, is a B-bucket histogram, *i.e.*, choices of I_j and b_j, that minimize the sum-square-error $\|\mathbf{A} - \mathbf{H}\|_2^2 = \sum_i |\mathbf{A}_i - \mathbf{H}_i|^2$. The problem of interest therefore is to find $\mathbf{H}_{\mathrm{opt}}$ or to approximate it. An approximation factor of α would indicate $\|\mathbf{A} - \mathbf{H}\|_2^2 \leq \alpha \|\mathbf{A} - \mathbf{H}_{\mathrm{opt}}\|^2$.

Histogram representations are used extensively in signal processing, statistics and databases. In database systems for example, they are used to approximate sizes of database operations which in turn help determine efficient execution plans for complex queries (See [1] for an overview). Almost all commercial database systems use histograms; finding best histograms—and other succinct representations such as wavelets or discrete fourier coefficients of the signal—is a thriving area of research in the database community (See proceedings of recent ACM SIGMOD and VLDB conferences).

Our motivation lies in an emerging application scenario in large databases and in processing massive data in general. Signals (such as timeseries of network events, web accesses, IP traffic patterns) are huge and appear in a stream. In most cases, they are not captured in databases because they are far too volumnious. (For an overview, see [2] for sizes, description of data feeds, etc.) Nevertheless, there is a growing need (eg., for network management purposes) to summarize the signals succinctly using histograms. Even when signals are small enough to be stored in disks within databases, practitioners seek algorithms that read the signal in one pass and compute (or estimate) functions of interest. This is because multiple passes or random accesses to disk-resident data is expensive. Thus, the focus in dealing with massive data is to seek algorithms that process signals on a stream, be they available as a stream from the data source (as in IP router or server logs) or read in one pass over stored data (as in large databases). Systems researchers have extensively articulated the need for stream processing (See [3–5] for references).

In the past few years, models have been developed to design, analyze and study data stream algorithms. Specifically, algorithms are required to read each item in the stream in turn, work using some additional workspace, and compute functions of interest. No backtracking is allowed on the input data stream. There are three parameters of performance: **(1)** time to process each data stream item, **(2)** amount of workspace used, and **(3)** time to compute functions of interest. Clearly the model is of interest only when the workspace provided is smaller than the space needed to store the entire signal. Furthermore, with this restriction, almost no function can be computed exactly (for example, computing the median of signal items is hard [6]). Hence, the emphasis is on estimating functions accurately, rather than computing them exactly. Specifically, algorithms in data stream model are designed with polylogarithmic workspace and they attempt to optimize the other two parameters pertaining to speed. Data stream algorithms have been designed for estimating norms [7–9], clustering [10], wavelet and histogram estimations [11, 12, 14, 13], etc.

Our work involves designing data stream algorithms. Our departure from previous work begins with observing that not all parameters above are equally important in applications. In particular, the per-item processing time is highly critical. This is because in data streaming instances such as IP routers that generate logs of packets they forward, flows and TCP connections they maintain etc, work at blistering speeds, and process millions of items per second. It is imperative that any per-item processing be very small in order to deal with this deluge. Likewise, disk systems scan large databases at very high speeds and to keep up with the pipeline, it is desirable that data stream algorithms minimize per-item processing. Our work here is inspired by this requirement.

1.1 Our Result and Previous Work

The problem of constructing (near) optimal histograms has been investigated theoretically. Most of the solutions (including ours) can be viewed as a *streaming algorithm*, that actually consists of two algorithms: a *sketching* algorithm that preprocess the input to some data structure and a *reconstruction* algorithm that performs some computation on the structure to output the final approximation. The total time is dominated by the preprocessing time to build (and maintain) the data structure. Thus we will quote results in per-item time in context of streaming algorithms. Since we can always store a set of elements and then perform the entire computation required, the per-item time is meaningful only if the space allowed is sublinear.

In [15], an $O(N^2B)$ time and $O(NB)$ space dynamic programming based solution was presented for optimal offline histogram computation. They also presented an approximate algorithm, that, when adopted to the streaming model, use space $O(B \log \|\mathbf{A}\|)$ space and $O(\log \mathbf{A})$ per-item processing time to output a $\mathbf{H}$ of at most $3B$ intervals such that $\|\mathbf{A}-\mathbf{H}\|_2^2 \leq 3\left\|\mathbf{A}-\mathbf{H}_{\text{opt}}^B\right\|_2^2$. Gilbert et al [14] presented an algorithm that uses $O(B \log N)$ space and $O(\log N)$ per-item processing time[1] algorithm to output $\mathbf{H}$ using $O(B \log N)$ intervals with $\|\mathbf{A}-\mathbf{H}\|_2^2 \leq \|\mathbf{A}-\mathbf{H}_{\text{opt}}\|_2^2$. The essential tradeoff however has been between accuracy of the approximation and the number of buckets.

[11] presented a substantially more accurate algorithm taking $(B^2 \log(N))/\epsilon$ per-item processing and histogram computation time as well as $(B^2 \log(N))/\epsilon$ space to output a $1+\epsilon$ approximation; that is an $\mathbf{H}$ such that $\|\mathbf{A}-\mathbf{H}\|_2^2 \leq$

[1] This algorithm may be converted to use $O(1)$ time per-item time using techniques in this paper.

$(1+\epsilon)\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|_2^2$. Subsequently in [12] it was modified to give a $\theta(N)$ space, $O(N)$ total time algorithm. These results capture a tradeoff between space and the time required for histogram construction.

Our result here improves both the tradeoff directions above and achieves the best possible. We provide a $1+\epsilon$ approximation in $O(1)$ per-item time. The work space we use is only $B(\log N \log \|\mathbf{A}\| / \epsilon)^{O(1)}$ which is of independent interest since previous algorithms used $\Omega(B^2)$ space, a limiting condition. The reconstruction time is polylogarithmic.

1.2 Overview of Our Techniques

Our approach uses signal processing techniques. Our overall algorithm divides neatly into two modules. The first module reads over the stream and outputs a certain number of wavelet coefficients of $\mathbf{A}$. Wavelet coefficients are inner products of the signal with dyadic basis vectors as defined in Section 2. This module is deterministic and exact, taking only $O(1)$ time for per-item processing. The second module processes these coefficients to construct $\mathbf{H}$. It has two major components. The first component involves constructing what we call a *robust* histogram of $\mathbf{A}$. A robust histogram has the property that refinement by further buckets does not decrease the overall error significantly. The notion of robust histogram was introduced in [13]; there the robust histogram had to be constructed directly from the signal via sophisticated randomized techniques. Here, we construct the robust histogram via wavelets, iteratively and deterministically, which we can do in our streaming model. The robust approximation is already a good approximation to the final histogram, but has a few too many intervals. The second major component involves culling the output histogram using dynamic programming from the robust histogram. The overall algorithm is quite simple. The crux throughout is the proof of various structural properties of intermediate histograms that yields the final histogram.

1.3 Map

In Section 2 we present formal definitions and background. In Section 3, we present the first module, namely, the streaming algorithm for computing various wavelet coefficients. In Section 4, we present the second module of constructing robust histogram and culling the final histogram from it. In Section 5, we present concluding remarks.

2 Preliminaries

We consider signals indexed on $\{0,1,\ldots,N-1\}$, where N is a power of 2. A dyadic interval is an interval of the form $[k2^j,(k+1)2^j)$, where j and k are integers. The function that equals 1 on set S and zero elsewhere is denoted χ_S.

A (Haar) wavelet is a function ψ on $[0,N)$ of one of the following forms:

- $\frac{1}{\sqrt{N}}\chi_{[0,N)}$

$$- \ 2^{-j/2}\left(-\chi_{[k2^{j-1},(k+1)2^{j-1})} + \chi_{[(k+2)2^{j-1},(k+3)2^{j-1})}\right).$$

Example wavelets of the second type are

$$\begin{array}{l} 2^{-1/2}(-1,1,0,0,0,0,\ldots,), \\ 2^{-1/2}(0,0,-1,1,0,0,0,0,\ldots,),\ldots, \\ 2^{-1}(-1,-1,1,1,0,0,0,0,\ldots,)\ldots \end{array}$$

There are N wavelets altogether, and they form an orthonormal basis, *i.e.*, $\langle\psi,\psi'\rangle$ is 1 if $\psi=\psi'$ and 0 otherwise.

Every signal can be reconstructed exactly from all its wavelet coefficients (its full *wavelet transform*, an orthonormal linear transformation), as $\mathbf{A}=\sum_j \langle\mathbf{A},\psi_j\rangle\,\psi_j$, whence a formal linear combination of distinct wavelets is its own wavelet transform.

Parsefal's equality states that the L^2 norm of a signal is invariant under orthonormal change of basis:

$$\sum_i \mathbf{A}_i^2 = \sum_j \langle\mathbf{A},\psi_j\rangle^2 .$$

It follows from Parsefal's equality that, for any set Λ of B wavelet terms, the error in using those terms to approximate $\mathbf{A}$ is the sum of the squares of the omitted terms. This is minimized when Λ contains the wavelet terms whose coefficients have the largest squares.

A simple classical wavelet algorithm computes the full wavelet transform in linear time and space. In the first of several passes, compute and output $N/2$ of the wavelets of the form $\mathbf{A}_{2i+1}-\mathbf{A}_{2i}$. Also compute $N/2$ quantities of the form $\mathbf{A}_{2i+1}+\mathbf{A}_{2i}$, and recursively compute a wavelet decomposition of that.

Note that each wavelet is a piecewise-constant function of 4 pieces (3 boundaries). Conversely, each χ_I can be written as $\chi_I=\sum_j\langle\chi_I,\psi_j\rangle\,\psi_j$, and there are only $2\log(N)$ wavelets for which the dot product is zero—those wavelets whose support intersects an endpoint of I, where the support of a vector is the set of positions where it is non-zero. Thus wavelet representations simulate histograms with at most a $O(\log(N))$ blowup in the number of buckets/terms. It follows that, using wavelets, one can easily find a $O(B\log(N))$-bucket histogram which approximates given signal as well as the best B-bucket histogram. Specifically, the $(6B\log(N)+1)$-bucket histogram representation defined as the best $2B\log N$-term wavelet representation has this property.

In what follows, we first show how to find the top B' wavelet coefficients quickly. For $B'=2B\log(N)$, this already gives an efficient construction of a $O(B\log(N))$-bucket histogram. Next, we show how to output instead a B-bucket histogram though the error of our histogram is worse than optimal by the factor $(1+\epsilon)$.

3 Efficient Computation of Wavelet Coefficients from a Stream

We will be interested in finding the largest B' coefficients of a wavelet decomposition of the stream we receive.

It is easy to modify the $\log(N)$-pass classical algorithm from the previous section to work in one pass; the order in which the coefficients are output is altered only. We need to view the $\log N$ passes as happening concurrently. The result of the first pass is the input to the second pass in a stream fashion and so forth. Thus, with space $O(\log N)$, we can output a stream of wavelet coefficients of the original stream.

Lemma 1. *There is an algorithm that reads in a stream $\mathbf{A}_0, \mathbf{A}_1, \ldots$ and outputs the N wavelet coefficients (in arbitrary order), using per-item time $O(1)$ and space $O(\log(N))$.*

We next show how to find the B' largest items in a stream by an algorithm that uses per-item time $O(1)$ and space $O(B')$.

We maintain a list of size at most $2B'$. Initially we store the first $2B'$ elements we receive. After we have $2B'$ elements, we run a selection algorithm [6] which finds the median; we then retain only the elements larger than it. The process of finding the median of $2B'$ elements and discarding the bottom half takes $O(B')$ time. At this point we have only B' elements; we store the next B' elements without any computation, then perform the reduction as above. We perform $O(N/B')$ reductions on sets of size $O(B')$; altogether, this takes $O(N)$ time with worst-case per-item time $O(B')$. By buffering the input in a buffer of size $O(B')$ we can reduce the worst-case per-item time to $O(1)$. (For example, upon reading an item that grows our list to size $2B'$, perform $O(1)$ steps of the reduction algorithm for each input, while incoming input sits in a buffer of size B'. When that buffer fills, we'll have completed the reduction, leaving a set of size B', which we combine with the input buffer of B' items.)

Lemma 2. *There is an algorithm that takes B' as input, reads in a stream $\mathbf{A}_0, \mathbf{A}_1, \ldots$ and outputs the top B' wavelet coefficients, using per-item time $O(1)$ and space $O(B')$.*

4 Wavelets to Histograms via Robust Representations

In this section, we show how to find a nearly optimal B-bucket histogram representation, using the techniques of the previous section. There are two algorithmic parts to this.

- Construction of a "robust" approximation. Given B, ϵ, M, and N, we define a particular $B' \leq (B\log(N)\log(M)/\epsilon)^{O(1)}$. We then show, for each signal $\mathbf{A}$ with $\|\mathbf{A}\|^2 \leq M$, how to construct, greedily, an approximation $\mathbf{H}_r$ from the top B' wavelet coefficients of $\mathbf{A}$ such that, if $\mathbf{H}$ refines $\mathbf{H}_r$

by an additional $B-1$ boundaries and $\mathbf{H}$ has optimal parameters, then $\|\mathbf{A}-\mathbf{H}_r\|^2 \le (1+\epsilon_r)\|\mathbf{A}-\mathbf{H}\|^2$. That is, $\mathbf{H}_r$ is not significantly improved by additional boundaries.
- Construction of our output, $\mathbf{H}$, which is defined to be the best B-term representation to $\mathbf{H}_r$. We first give an efficient construction of $\mathbf{H}$ and then we show that $\|\mathbf{A}-\mathbf{H}\|^2 \le (1+\epsilon)\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|^2$, provided ϵ_r is chosen properly given ϵ. (They are polynomially related.)

4.1 Robust Histograms

Definition 1 (see [13]). *Fix a signal,* $\mathbf{A}$. *A representation* $\mathbf{H}_r$ *is called a* (B_r,ϵ_r)-robust approximation to $\mathbf{A}$ *if, for any representation* $\mathbf{H}$ *on the boundaries of* $\mathbf{H}_r$ *and any other* B_r-1 *boundaries, with optimal parameters, we have*

$$(1-\epsilon_r)\|\mathbf{A}-\mathbf{H}_r\|^2 \le \|\mathbf{A}-\mathbf{H}\|^2 .$$

In [13], an algorithm for constructing a robust approximation was given. We obtain the same result here, but in a significantly simpler way, which is possible in our model. We include a sketch of the construction here. In this paper, we will need $B_r = B\log n$ and $\epsilon_r = \epsilon^4$ to get a B-bucket histogram with error $(1+\epsilon)$ times the optimal.

Lemma 3. *Given* B_r, N, ϵ_r, *and* M, *for any integer-valued signal* $\mathbf{A}$ *with* $\|\mathbf{A}\| \le M$, *there exists a* $B' \le (B_r\log(N)\log(M)/\epsilon_r)^{O(1)}$ *and a* $(B')^{O(1)}$*-time algorithm to find a* (B_r,ϵ_r)*-robust approximation to* $\mathbf{A}$ *from among the top* B' *wavelet coefficients for* $\mathbf{A}$.

Proof. First note that the characteristic function χ_I can be written as the sum of $O(\log(N))$ wavelets—corresponding to those wavelets whose support intersects an endpoint of I. Thus a B-bucket histogram can be viewed as a $O(B\log(N))$-term wavelet representation, refinenment of a histogram by $B-1$ boundaries can be simulated by refining a wavelet representation by $O(B\log(N))$ terms, and it suffices to find a wavelet representation that is not much improved by $O(B_r\log(N))$ additional wavelet terms.

The algorithm is as follows. Start with the zero representation $\mathbf{R}$. If $\mathbf{R}$ is not already robust, then some $B_r\log(N)$ wavelet terms improve it. The terms giving the best improvement are, by Parsefal, those with the largest coefficients not already in $\mathbf{R}$. It follows that the largest single coefficient gives some improvement, namely,

$$\|\mathbf{A}-(\mathbf{R}+\langle\mathbf{A},\psi\rangle\psi)\|^2 \le \left(1+\Omega\left(\frac{\epsilon_r}{B_r\log(N)}\right)\right)^{-1}\|\mathbf{A}-\mathbf{R}\|^2 .$$

Replace $\mathbf{R} \leftarrow \mathbf{R}+\langle\mathbf{A},\psi\rangle\psi$ and remove ψ from the list of available terms. Repeat this procedure until the representation is robust. Observe that after $O\left(\frac{B_r\log(N)\log(M)}{\epsilon_r}\right)$ iterations, the value of $\|\mathbf{A}-\mathbf{R}\|^2$ has been reduced from

$\|\mathbf{A}\|^2$ by the factor $\frac{1}{4M^2}$. Since we assumed that $\mathbf{A}$ is integer-valued and $\|\mathbf{A}\| \leq M$, it follows that a rounding of $\mathbf{R}$ to integer coefficients (a legitimate wavelet representation) equals $\mathbf{A}$. Thus, at this iteration, one can choose $\mathbf{A}$ itself as a robust representation.

Thus we put $B' = \Theta\left(\frac{B_r \log(N) \log(M)}{\epsilon_r}\right)$. We construct a $(O(B_r \log(N)), \epsilon_r)$-robust wavelet represntation (generalizing the definition of robustness from histograms to wavelets in the obvious way), for which we need the top B' wavelet terms. The result can be viewed as a (B_r, ϵ_r)-robust histogram approximation with at most $O(B')$ buckets.

4.2 Approximating the Robust Histogram

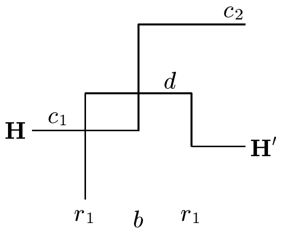
In this section, we show how to find the best B-bucket approximation $\mathbf{H}$ to $\mathbf{H}_r$. (Later we will argue that $\mathbf{H}$ is, in fact, a good approximation to $\mathbf{A}$.) We will show that $\mathbf{H}$ only needs to use the boundaries in $\mathbf{H}_r$, of which there are at most $O(B')$. Thus a dynamic programming algorithm [15] will find $\mathbf{H}$ in time polynomial in B' using space $O(B')$.

Lemma 4. *Given a (sparsely presented) B'-bucket histogram $\mathbf{H}'$ on $N \geq B'$ numbers, the best B-bucket approximation $\mathbf{H}$ to $\mathbf{H}'$ uses only the boundaries of $\mathbf{H}'$.*

Proof. Suppose not. Let $r_1 < b < r_2$, where b is a boundary in $\mathbf{H}$ and r_1 and r_2 are consecutive boundaries in $\mathbf{H}'$. Let c_1 and c_2 be the coefficients in buckets to the left and right of b and let d be the coefficient in bucket $[r_1, r_2)$. (See Figure 1.)

Suppose $|c_1 - d| \leq |c_2 - d|$. Then the result of moving b to the right to r_2 is no worse, since, in $\mathbf{H}$, now more of $[r_1, r_2)$ gets the value c_1 and less gets c_2.

Fig. 1. Illustration of histograms in Lemma 4.



4.3 Correctness of the Output

We now show that if $\mathbf{H}_r$ is a (B, ϵ_r)-robust approximation to $\mathbf{A}$ for suitable B and ϵ_r, then the best B-bucket approximation to $\mathbf{H}_r$ is a nearly optimal representation for $\mathbf{A}$. We will have $\epsilon_r = \epsilon^4/8 < \epsilon/8$. Let $\mathbf{H}^*$ denote the best linear combination of $\mathbf{H}_r$ with $\mathbf{H}_{\mathrm{opt}}$. Since this is a refinement of $\mathbf{H}_r$ by $B-1$ boundaries,

$$(1-\epsilon_r)\left\|\mathbf{A}-\mathbf{H}_r\right\|^2 \leq \left\|\mathbf{A}-\mathbf{H}^*\right\|^2 \leq \left\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\right\|^2 \tag{1}$$

Lemma 5. *Either* $\|\mathbf{H}-\mathbf{H}_r\| > \frac{\epsilon}{2}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| \geq \epsilon_r\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|$ *or* $\|\mathbf{H}-\mathbf{A}\|^2 \leq (1+O(\epsilon))\|\mathbf{H}_{\mathrm{opt}}-\mathbf{A}\|^2$.

Proof. Assume $\|\mathbf{H}-\mathbf{H}_r\| \leq \frac{\epsilon}{2}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|$. Then, by the triangle inequality,

$$\begin{aligned}
\|\mathbf{H}-\mathbf{A}\| &\leq \|\mathbf{H}-\mathbf{H}_r\| + \|\mathbf{H}_r-\mathbf{A}\| \\
&\leq \frac{\epsilon}{2}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| + \|\mathbf{H}_r-\mathbf{A}\| \\
&\leq \frac{\epsilon}{2}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| + \frac{1}{\sqrt{1-\epsilon_r}}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| \\
&\leq (1+\epsilon)\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|.
\end{aligned}$$

Lemma 6. *Fix a signal* $\mathbf{A}$*, and let* $\mathbf{H}_r$ *be a* (B,ϵ_r)*-robust approximation to* $\mathbf{A}$*. Let* $\mathbf{H}$ *be the best* B*-bucket approximation to* $\mathbf{H}_r$*. Then*

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

where $\epsilon_r = \Theta(\epsilon^4)$.

Proof. Assume that the inequality $\frac{\epsilon}{2}\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\| < \|\mathbf{H}-\mathbf{H}_r\|$ in Lemma 5 holds. Let $\widehat{\mathbf{H}}$ be the best linear combination of $\mathbf{H}$ and $\mathbf{H}_r$ (see Figure 2). Thus there are right angles at $\mathbf{H}^*$ and at $\widehat{\mathbf{H}}$. In Equations (2) to (5), we show that $\mathbf{H}^*, \widehat{\mathbf{H}}$, and $\mathbf{H}_r$ are all close together, so that, in that sense, the angles $\mathbf{A}$-$\mathbf{H}_r$-$\mathbf{H}$ and $\mathbf{A}$-$\mathbf{H}_r$-$\mathbf{H}$ are close to right angles. We then give the conclusion.

Note that each of $\mathbf{H}^*$ and $\widehat{\mathbf{H}}$ refines $\mathbf{H}_r$ by B buckets, so, it follows that

$$\begin{aligned}
\left\|\widehat{\mathbf{H}}-\mathbf{H}_r\right\|^2 &= \|\mathbf{A}-\mathbf{H}_r\|^2 - \left\|\mathbf{A}-\widehat{\mathbf{H}}\right\|^2 \\
&\leq \epsilon_r\|\mathbf{A}-\mathbf{H}_r\|^2, \quad \text{by robustness} && (2)\\
&\leq \epsilon_r(1+O(\epsilon_r))\|\mathbf{A}-\mathbf{H}_{\mathrm{opt}}\|^2, \quad \text{by (1)} \\
&\leq \frac{4\epsilon_r(1+O(\epsilon_r))}{\epsilon^2}\|\mathbf{H}-\mathbf{H}_r\|^2 \\
&\leq \epsilon^2\|\mathbf{H}-\mathbf{H}_r\|^2, \quad \text{by Lemma 5.} && (3)
\end{aligned}$$

Similarly,

$$\begin{aligned}
\|\mathbf{H}^*-\mathbf{H}_r\|^2 &= \|\mathbf{A}-\mathbf{H}_r\|^2 - \|\mathbf{A}-\mathbf{H}^*\|^2 \\
&\leq \epsilon^2\|\mathbf{H}-\mathbf{H}_r\|^2 \\
&\leq \epsilon^2\|\mathbf{H}_{\mathrm{opt}}-\mathbf{H}_r\|^2, \quad \text{by optimality of } \mathbf{H} \text{ for } \mathbf{H}_r && (4)
\end{aligned}$$

Furthermore, it also follows that

$$\begin{aligned}
\left\|\widehat{\mathbf{H}}-\mathbf{H}^*\right\| &\leq \left\|\widehat{\mathbf{H}}-\mathbf{H}_r\right\| + \left\|\mathbf{H}_r-\mathbf{H}^*\right\| \\
&\leq 2\sqrt{\epsilon_r}\left\|\mathbf{A}-\mathbf{H}_r\right\|, \quad \text{by (2)} \\
&\leq \frac{2\sqrt{\epsilon_r}}{\sqrt{1-\epsilon_r}}\left\|\mathbf{A}-\mathbf{H}^*\right\| \quad \text{by (1)} \\
&\leq \epsilon\left\|\mathbf{A}-\mathbf{H}^*\right\|. \qquad (5)
\end{aligned}$$

See Figure 2. Finally, we have:

$$\begin{aligned}
\left\|\mathbf{H}-\mathbf{A}\right\|^2 &= \left\|\mathbf{H}-\widehat{\mathbf{H}}\right\|^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2 \\
&\leq \left(\left\|\mathbf{H}-\mathbf{H}_r\right\| + \left\|\mathbf{H}_r-\widehat{\mathbf{H}}\right\|\right)^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2 \\
&\leq (1+\epsilon)^2\left\|\mathbf{H}-\mathbf{H}_r\right\|^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2, \quad \text{by (3)} \\
&\leq (1+\epsilon)^2\left\|\mathbf{H}_{\text{opt}}-\mathbf{H}_r\right\|^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2, \quad \text{since } \mathbf{H} \text{ is optimal for } \mathbf{H}_r \\
&\leq (1+\epsilon)^2\left(\left\|\mathbf{H}_{\text{opt}}-\mathbf{H}^*\right\| + \left\|\mathbf{H}^*-\mathbf{H}_r\right\|\right)^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2 \\
&\leq (1+\epsilon)^4\left\|\mathbf{H}_{\text{opt}}-\mathbf{H}^*\right\|^2 + \left\|\widehat{\mathbf{H}}-\mathbf{A}\right\|^2, \quad \text{by (4)} \\
&\leq (1+O(\epsilon))\left\|\mathbf{H}_{\text{opt}}-\mathbf{H}^*\right\|^2 + \left(\left\|\widehat{\mathbf{H}}-\mathbf{H}^*\right\| + \left\|\mathbf{H}^*-\mathbf{A}\right\|\right)^2 \\
&\leq (1+\epsilon)^4\left\|\mathbf{H}_{\text{opt}}-\mathbf{H}^*\right\|^2 + (1+\epsilon)^2\left\|\mathbf{H}^*-\mathbf{A}\right\|^2, \quad \text{by (5)} \\
&\leq (1+\epsilon)^4\left\|\mathbf{H}_{\text{opt}}-\mathbf{A}\right\|^2, \quad \text{by the Pythagorean theorem} \\
&\leq (1+O(\epsilon))\left\|\mathbf{H}_{\text{opt}}-\mathbf{A}\right\|^2.
\end{aligned}$$

4.4 Main Theorem

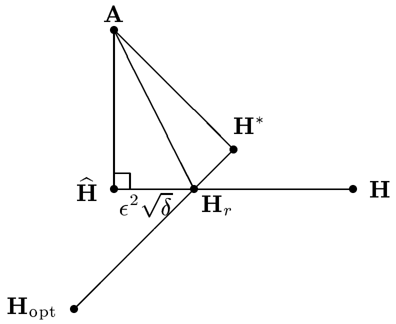
Combining the above results, we have

Theorem 1. *There exists an algorithm that, given B, N and ϵ, on input the N values of an integer-valued signal $\mathbf{A}$ with $\|\mathbf{A}\| \leq M$, outputs a B-bucket histogram $\mathbf{H}$ with*

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

where $\mathbf{H}_{\text{opt}}$ is the best possible B-bucket histogram representation to $\mathbf{A}$. The algorithm uses space $B(\log(N)\log(M)/\epsilon)^{O(1)}$. The algorithm requires two modules. The first (sketching) uses time $O(N)$ and the second (reconstruction) uses time in $(B\log(M)\log(N)/\epsilon)^{O(1)}$.

Fig. 2. Illustration of histograms, which are not necessarily coplanar. The histograms indicated as colinear are, in fact colinear; the order of three histograms in a line is not necessarily the order indicated. The points $\widehat{\mathbf{H}}, \mathbf{H}_r$, and $\mathbf{H}^*$ are all close can can be roughly collapsed. Then, since $\|\mathbf{H} - \mathbf{H}_r\| \leq \|\mathbf{H}_{\text{opt}} - \mathbf{H}_r\|$, it follows that $\|\mathbf{H} - \mathbf{A}\|^2 \leq (1+\epsilon)\|\mathbf{H}_{\text{opt}} - \mathbf{A}\|^2$.



Proof. Use the classic wavelet algorithm to produce all coefficients, reading in a stream and spitting out a stream. Use the median algorithm to select the B' coefficients with largest square. Use the greedy algorithm to produce a $(O(B\log(N)), \Theta(\epsilon^4))$-robust wavelet approximation $\mathbf{H}_r$, which is a $(B, \Theta(\epsilon^4))$-robust histogram representation. Use dynamic programming to find the best B-bucket histogram $\mathbf{H}$ to $\mathbf{H}_r$, by observing that the boundaries of $\mathbf{H}$ are among the boundaries of $\mathbf{H}_r$. By Lemma 6, this output is correct.

5 Concluding Remarks

We have presented a simple algorithm that takes $O(1)$ time to process each item of a data stream and using $B(\log(N)\log(M)/\epsilon)^{O(1)}$ space and polylogarithmic reconstruction time, obtains an $1+\epsilon$ approximation to the optimal histogram.

A more general streaming model is one in which we are allowed to observe the signal only through updates (not necessarily left-to-right pass). Under this model, the best known algorithm for histogram constructing takes $(B\log(N)\log(M)/\epsilon)^{O(1)}$ resources for all three parameters [13]. Some of the exponents involved are rather large, *e.g.*, at least 6. It is an open problem to see if per-item processing time can be reduced further in this model.

Our approach of first obtaining large wavelet coefficients, then obtaining a robust histogram from which the final histogram is culled, is of interest. For example, in practice, one may experimentally compare each of these intermediate and final representations in terms of their overall accuracy in capturing signal trends. Our algorithm is simple and implementable, so this is a feasible study for the future.

References

1. V. Poosala. Histogram techniques for databases. Ph. D Thesis, Univ. Wisconsin, Madison, 1996.
2. Anja Feldmann, Albert G. Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, Fred True. Deriving traffic demands for operational IP networks: methodology and experience. SIGCOMM 2000: 257-270
3. http://www-db.stanford.edu/stream/
4. http://www.cs.cornell.edu/database/cougar/index.htm.
5. Fjording the Stream: An Architecture for Queries over Streaming Sensor Data. Sam Madden and Michael J. Franklin, ICDE Conference, February, 2002, San Jose.
6. Selection and sorting with limited storage. J. I. Munro and M. S. Paterson. Theoretical Computer Science, pages 315-323, 1980.
7. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. Piotr Indyk: FOCS 2000: 189-197
8. The Space Complexity of Approximating the Frequency Moments. Noga Alon, Yossi Matias, Mario Szegedy. STOC 1996: 20-29
9. An Approximate L1-Difference Algorithm for Massive Data Streams. Joan Feigenbaum, Sampath Kannan, Martin Strauss, Mahesh Viswanathan. FOCS 1999: 501-511
10. Clustering Data Streams. Sudipto Guha, Nina Mishra, Rajeev Motwani, Liadan O'Callaghan. FOCS 2000: 359-366
11. Data-streams and histograms. Sudipto Guha, Nick Koudas, Kyuseok Shim. STOC 2001: 471-475
12. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. Sudipto Guha, Nick Koudas. ICDE 2002
13. Dynamic maintenance of histograms. A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan and M. Strauss. To appear in STOC 2002.
14. Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, Martin Strauss. VLDB 2001: 79-88
15. H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, Torsten Suel. Optimal Histograms with Quality Guarantees. VLDB 1998: 275-286.