Smooth Histograms for Sliding Windows

Vladimir Braverman* Rafail Ostrovsky†

Appeared in FOCS 2007: 283-293

Abstract

In the streaming model, elements arrive sequentially and can be observed only once. Maintaining statistics and aggregates is an important and non-trivial task in the model. This becomes even more challenging in the sliding windows model, where statistics must be maintained only over the most recent n elements. In their pioneering paper, Datar, Gionis, Indyk and Motwani [15] presented *exponential histograms*, an effective method for estimating statistics on sliding windows. In this paper we present a new *smooth histograms* method that improves the approximation error rate obtained via exponential histograms. Furthermore, our smooth histograms method not only captures and improves multiple previous results on sliding windows but also extends the class functions that can be approximated on sliding windows. In particular, we provide the first approximation algorithms for the following functions: L_p norms for $p \notin [1, 2]$, frequency moments, length of increasing subsequence and geometric mean.

*Dept. of Computer Science, UCLA. E-mail: vova@cs.ucla.edu. Work supported in part by NSF Cybertrust grant #0430254.

†Depts. of Computer Science and Mathematics, UCLA. E-mail: rafail@cs.ucla.edu. Work supported in part by an IBM Faculty Award, a Xerox Innovation Group Award, NSF Cybertrust grant #0430254, and a U.C. MICRO grant.

1 Introduction

Many recent applications deal with large data volumes for which methods that require multiple data passes may be infeasible. For these applications, the *data stream* model is often more appropriate. In this model, data arrives sequentially and can be observed only once. The number of data elements is unknown and may be unbounded. A typical goal is to continuously maintain statistics or aggregates over past data using minimal memory while keeping the desired precision of the answers. In this scenario, it may be challenging to maintain even simple statistics. Recently, numerous algorithms were developed for various problems in the data stream model. We refer readers to the books of Muthukrishnan [27] and Aggarwal [2] for detailed surveys on data steaming models and algorithms.

Most applications tend to discard old data and base their queries only on the recent elements. Thus, the sliding window model in which only the last n elements are taken into consideration is important in data stream processing. In this model we separate past elements into two groups. Recent elements represent a window of *active* or *non-expired* elements, and the rest are *expired*. An active element may eventually become expired, but expired elements stay in this status forever. Only active elements are relevant for statistics or queries. The window can be *sequence-based*, where every insertion corresponds to a deletion of the oldest element. In *timestamp-based* windows, there is no restriction on the number of insertions and deletions. (Typically, each element is associated with a timestamp, and the window contains all elements with active timestamps.)

1.1 Notations

We use the following notations throughout our paper. We denote D as a stream and $p_i, i \geq 0$ as its i-th element. For $0 \leq x < y$ we define $[x, y] = \{i, x \leq i \leq y\}$. *Bucket* $B(x, y)$ is the set of all stream elements between p_x and p_{y-1}: $B(x, y) = \{p_i, i \in [x, y-1]\}$. For function f that is defined on buckets we denote $f(i, j) = f(B(i, j))$. We denote N as the size of the stream and n as the size of a window. For the L_p problem, we denote m as the size of a vector that is presented by a stream. An algorithm maintains ϵ-approximation of function f on stream D if at any moment the algorithm outputs f' s.t. $(1-\epsilon)f(D) \leq f'(D) \leq (1+\epsilon)f(D)$. Similarly, an algorithm maintains (ϵ, δ)-approximation of function f if at any moment the algorithm outputs f' s.t. $(1-\epsilon)f(D) \leq f'(D) \leq (1+\epsilon)f(D)$ w.p. at least $1-\delta$. We denote $\tilde{O}(f(m)) = O(\frac{1}{\epsilon^{O(1)}}(\log m)^{O(1)}(\log n)^{O(1)}f(m))$. We use notation $B \subseteq_r A$ to indicate that bucket B is a suffix of A; i.e., it contains the most recent part of A. We denote by $A \cup C$ the union of adjacent buckets A and C.

1.2 Problems, Results and Related Work

Research on the sliding window model has a long history. In their pioneering paper, Datar, Gionis, Indyk and Motwani [15] gave memory-optimal algorithms for such fundamental statistics as count, sum of positive integers, average, $L_p, p \in [1, 2]$ etc. A further improvement to count and sum was reported by Gibbons and Tirthapura [19] who provided memory and time optimal algorithms for these problems. Lee and Ting [25] provided optimal solution for a relaxed version of the counting

problem, where the correct answer is provided only if it is comparable with the window's size. Besides these basic statistics, numerous problems were effectively solved on sliding windows. Chi, Wang, Yu and Muntz in [13] considered a problem of frequent itemsets. Algorithms for frequency counts and quantiles were proposed by Arasu and Manku [4] and by Lee and Ting [24]. Datar and Muthukrishnan [16] solved problems of rarity and similarity. Babcock, Datar, Motwani and O'Callaghan [6] provided algorithms for variance and k-medians problems. Feigenbaum, Kannan and Zhan [17] presented an efficient solution for the diameter of a data set in multidimensional space. Later, Chan and Sadjad [11] presented optimal solutions for this and other geometric problems. Agarwal, Har-Peled, and Varadarajan [1] used coresets to maintain statistics in streaming model. Babcock, Datar and Motwani [5] presented algorithms for uniform random sampling from sliding windows. For more details about recent results in the sliding windows model, we refer readers to the survey by Datar and Motwani [2].

The use of exponential histograms as a general technique for sliding windows was proposed by Datar, Gionis, Indyk and Motwani [15]. This method is widely used and numerous algorithms are based on exponential histograms or their variations (see [2] for examples of such applications). It is applicable to a wide class of "weakly additive" functions [2] with the following properties. (A and B are adjacent buckets and $\cup$ denotes concatenation.)

1. $f(A) \geq 0$.
2. $f(A) \leq poly(|A|)$.
3. $f(A \cup B) \geq f(A) + f(B)$.
4. $f(A \cup B) \leq C_f(f(A) + f(B))$ for some constant $C_f \geq 1$.
5. The function $f(A)$ admits a "sketch" which requires $g_f(|B|)$ space and is composable; i.e., the sketch for $f(A \cup B)$ can be composed efficiently from the sketches for $f(A)$ and $f(B)$.

For this class of functions, [15] presents two general results. If sketches can be computed precisely, then it is possible to maintain f with relative error $(\epsilon C_f^2 + C_f - 1)$, using $O(\frac{1}{\epsilon}\log n(g + \log n))$ bits and $O(1)$ amortized time per element. Moreover, given an algorithm that maintains $\hat{\epsilon}$-approximation of f on D, f can be approximated on sliding windows with relative error $(1+\hat{\epsilon})^2\epsilon C_f^2 + C_f - 1 + \hat{\epsilon}$, using the same space and time.

1.2.1 Summary of Our Results

In this paper we introduce a notion of *smooth function* and present technique that allows to maintain smooth functions over sliding windows. Informally, smooth functions are defined as follows. Let A be a bucket and B be its suffix; that is, B contains recent elements from A and does not contain the old elements. Consider the case when $f(B)$ is close to $f(A)$. If this closeness remains no matter what elements are added to both buckets and does not depend on A and B, we say that f is smooth (for formal definition see Section 2). To measure the closeness before and after insertions, we introduce two parameters α and β that depend only on function f and approximation parameter ϵ. Function f is (α, β)-smooth if, once $f(B)$ is β-approximation of $f(A)$, we can guarantee that

$f(B \cup C)$ is α-approximation of $f(A \cup C)$ for any portion of new elements C. Assume that there exists an algorithm that computes f precisely using g space and h time per element. Our main result states that it is possible to maintain α-approximation of f over sliding windows, using $O(\frac{1}{\beta}\log n(g+\log n))$ bits and $O(\frac{1}{\beta}h\log n)$ time. Further, $\hat{\epsilon}$-approximation of f on D results in $(\alpha+\hat{\epsilon})$-approximation of f over sliding windows.

It turns out that many functions are smooth. For instance, sum, count, min, diameter are (ϵ,ϵ)-smooth that gives ϵ-approximations repeating previously known results [11, 15]. More interesting, we prove that weakly additive functions, L_p norms, frequency moments, length of longest subsequence, and geometric mean are smooth. We apply our method to these function and obtain the following results:

- We improve the general results from [15] mentioned above. For weakly additive functions that can be computed precisely on D, the relative error is improved from $\epsilon C_f^2 + C_f - 1$ to $1-\frac{1-\epsilon}{C_f}$. For $C_f = 1$ it gives ϵ-approximation, similar to [15]. For larger C_f the ratio between relative errors is approximately C_f. The space and time complexities remain unchanged. For weakly additive functions that can be approximated on D, the relative error is improved from $(1+\hat{\epsilon})^2\epsilon C_f^2 + C_f - 1 + \hat{\epsilon}$ to $1-\frac{1-\epsilon}{C_f}+\hat{\epsilon}$.

- We improve the result of [15] for $L_p, p \in [1,2]$ norms and extend it to any p. For $p \in [1,2]$ we decrease relative error from $4\epsilon(1+\hat{\epsilon})^2+1+\hat{\epsilon}$ to $\frac{1+\epsilon}{2}+\hat{\epsilon}$, preserving memory and time. We also show that adding a multiplicative factor of $\frac{1}{\epsilon^{p-1}}$ to memory can further decrease relative error to ϵ. For $p > 2$ we give an optimal $(\epsilon+\frac{\epsilon^p}{4p},\delta)$-approximation algorithm that uses $\tilde{O}(m^{1-\frac{2}{p}})$ memory, where m is a size of the estimated vector. For $p < 1$ we present a $(\epsilon+\epsilon/4,\delta)$-approximation algorithm using $O\left(\frac{1}{\epsilon}\log n(\frac{1}{\epsilon^2}\log M \log\frac{\log n}{\delta\epsilon}+\log n)\right)$ bits, where M is the maximal value of an update.

- We provide the first memory-optimal algorithm for frequency moments over sliding windows for constant $p > 2$. The algorithm maintains $(\epsilon+\frac{\epsilon^p}{p^p},\delta)$-approximation using $\tilde{O}(m^{1-\frac{2}{p}})$ space.

- We extend the results of Sun and Woodruff [29] to sliding windows and improve (in terms of space) the recent results of Chen, Yang and Yuan [12], providing ϵ-approximation of the length of longest increasing subsequence (LIS) in sliding windows. Our algorithm uses $O(\frac{1}{\epsilon}\log n(k\log\frac{L}{k}+\log n))$ bits, where k is the length of LIS and L is the number of distinct elements in the window.

- We provide the first ϵ-approximation of geometric mean on sequence-based windows using $O(\frac{1}{\epsilon}\log n(k+\log n))$ memory. Here k is the number of bits needed to store the value of geometric mean.

Below we describe these problems in detail and discuss our and previous results.

1.2.2 L_p Norms and Frequency Moments

$L_p, p \geq 0$ norm of a vector $X = \{x_1, \ldots, x_m\}$ is defined as $(\sum_{i=1}^{m} x_i^p)^{\frac{1}{p}}$. In the streaming model X is represented as follows. Each element of a stream D is a pair $(i, a), i \in [m], a \in [-M, M]$, for some positive M. The value of i-th coordinate is given by the summation $x_i = \sum_{(i,a)\in D} a$. If m is small, it is easy to calculate the L_p norm simply by maintaining the value of each coordinate. However, the usual assumption is that m is large and $\Omega(m)$ space is not allowed.

Frequency moment is a fundamental problem that is directly related to L_p norms. In this paper we use the definition that was presented by Bhuvanagiri, Ganguly, Kesh and Saha [8], $F_p = \sum_{i=1}^{m} x_i^p = L_p^p$. In many papers the simpler model is considered, where $a = 1$ for all pairs (i, a) and a's are omitted.

The first algorithms for frequency moments were proposed in the seminal paper of Alon, Matias and Szegedy [3]. For $p = 0, 2$ they provided (ϵ, δ)-approximation algorithms that use only polylogarithmic memory. For $p \geq 3$ they present an algorithm that uses $O(m^{1-\frac{1}{p}})$ memory and show a lower bound of $\Omega(m^{1-\frac{5}{p}})$. For $p > 2$, numerous improvements to lower and upper bounds were reported, including the work of Bar-Yossef, Jayram, Kumar and Sivakumar [7], Chakrabarti, Khot and Sun [10], Coppersmith and Kumar [14] and Ganguly [18]. Finally, Indyk and Woodruff [23] and later Bhuvanagiri, Ganguly, Kesh and Saha [8] presented algorithms that use $\tilde{O}(m^{1-\frac{2}{p}})$ memory and are optimal. The last algorithm can be used as well for approximation of $L_p, p > 2$ norms. For $p \in [0, 2]$ Indyk [22] presented an algorithm that maintains L_p norms using polylogarithmic space.

The extension of these problems to sliding windows is straightforward. At any moment $L_p(W) = (\sum_{i=1}^{m} x_i^p)^{\frac{1}{p}}$, where $x_i = \sum_{(i,a)\in W} a$ and W is the current window. Similar to [15], we restrict a to be positive. For negative a it was shown [15] that even for $p = 1$ and $m = 1$, the lower bound on the memory is $\Omega(n)$. The only known result for L_p norms was presented in [15] for $p \in [1, 2]$. The algorithm maintains L_p with high probability and relative error $4\epsilon(1 + \epsilon)^2 + 1 + \epsilon$ using $O(\frac{1}{\epsilon} \log N(\log N + \log M \log (1/\delta)/\epsilon^2))$ bits.

We extend this result to any p and provide a better approximation ratio for $p \in [1, 2]$. We prove that L_p is $(\epsilon, \frac{\epsilon^p}{p})$-smooth for $p \geq 1$ and (ϵ, ϵ)-smooth for $p < 1$, so our method can be applied. For $p > 2$, we apply the algorithm from [8] to show an optimal (ϵ, δ)-approximation algorithm using $\tilde{O}(m^{1-\frac{2}{p}})$ bits. For $L_p, p < 1$, we use the algorithm from [22] to construct an $(\epsilon+\epsilon/4, \delta)$-approximation algorithm using $O\left(\frac{1}{\epsilon} \log n(\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$ bits. Finally, for $L_p, 1 \leq p \leq 2$, we improve [15] by decreasing the relative error from $4(1 + \hat{\epsilon})^2\epsilon + 1 + \hat{\epsilon}$ to $\frac{1+\epsilon}{2} + \hat{\epsilon}$ using $O\left(\frac{1}{\epsilon} \log n(\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$ bits. We exploit the fact that L_p^p is weakly additive with $C_f \leq 2$ and thus $(\frac{1+\epsilon}{2}, \epsilon)$-smooth. However, it is also $(\epsilon, \frac{\epsilon^p}{p})$-smooth and, alternatively, we can achieve an ϵ-approximation using $O\left(\frac{1}{\epsilon^p} \log n(\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$ bits; i.e., increasing memory usage by the factor $\frac{1}{\epsilon^{p-1}}$.

One may argue that for $p > 2$, L_p^p is also a weakly additive function, so we can apply the exponential histograms method. While this is true, note that the relative error that can be achieved using exponential histograms is $\epsilon C_f^2 + C_f - 1$ (see [15]). For large p the relative error becomes significantly larger then 1.

We show similar results for frequency moments. In particular, we prove that F_p is $(\epsilon, \frac{\epsilon^p}{p^p})$-smooth, and thus smooth histograms can be used to approximate frequency moments with $\tilde{O}(p^p m^{1-\frac{2}{p}})$ bits. Thus, for constant $p > 2$ we obtain optimal results.

1.2.3 Length of Longest Increasing Subsequence

Let D be a stream where p_i is an integer, $p_i \in [L]$ for some L. An increasing subsequence is defined as $p_{x_1}, \ldots, p_{x_k}$ such that $x_i < x_{i+1}$ and $p_{x_i} \leq p_{x_{i+1}}$ for $i < k$. (In fact, the sequence is non-decreasing, but we follow the notations of the previous works.) The largest increasing subsequence $LIS(D)$ is an increasing subsequence with maximal size k. Correspondingly, $LIS(W)$ on window W is defined as a largest increasing subsequence on the set of last n elements. This is a well-studied statistic that is used in bioinformatics and other fields. (See works of Gusfield [21] and Pevzner [28].) Recent results in the streaming model include the papers by Liben-Nowell, Vee and Zhu [26], Gopalan, Jayram, Krauthgamer and Kumar [20] and Sun and Woodruff [29]. The last paper presents memory-optimal algorithm that uses $\Theta(k \log \frac{L}{k})$ memory. For sliding windows Chen, Yang and Yuan [12] present an algorithm that uses $\Omega(n)$ memory.

We extend the result of Sun and Woodruff [29] to sliding windows and improve the result of Chen, Yang and Yuan [12]. Our algorithm uses $O(\frac{1}{\epsilon} \log n(k \log \frac{L}{k} + \log n))$ bits and provides ϵ-approximation of the length of LIS.

1.2.4 Geometric Mean

Let D be a stream of real numbers. Geometric mean is defined as $GM(D) = \left(\prod_{i=1}^{N} p_i\right)^{\frac{1}{N}}$. For sliding windows we define geometric mean as $GM(W) = \left(\prod_{i=N-n}^{N} p_i\right)^{\frac{1}{n}}$. To the best of our knowledge, this problems was not considered in the sliding windows model. While we can use exponential histograms [15] to maintain arithmetic and harmonic means, geometric mean is not an additive function, and therefore it cannot be solved by this method. We present the first approximation algorithm for geometric mean on sequence-based sliding windows that uses $O\left(\frac{1}{\epsilon} \log n(k + \log n)\right)$ space, where k is the number of bits needed to store the answer.

1.2.5 High-Level Ideas Behind Our Approach

Let f be (α, β)-smooth for which there exists an algorithm Λ that calculates f on D using g space and h operation per element. To maintain f on sliding windows, we construct a data structure that we call *smooth histogram*. It consists of a set of indexes $x_1 < x_2 < \cdots < x_s = N$ and instances of Λ for each bucket $B(x_i, N)$. The smooth histogram ensures the following properties of the sequence. First, the last active element should be located between first and the second elements of the sequence; that is, $x_1 \leq N - n < x_2$. Since f is monotonic, this implies $f(x_2, N) \leq f(W) \leq f(x_1, N)$. Second, the values of f on successive suffixes should be close, namely $(1 - \alpha)f(x_i, N) \leq f(x_{i+1}, N)$. (There is an exception, when f drops immediately after x_i, that is $(1 - \alpha)f(x_i, N) > f(x_i + 1, N)$, in which case we require $x_{i+1} = x_i + 1$.) The first and second requirements imply that $f(x_2, N)$ is an α-approximation of f on W. Finally, we ensure that f

is decreasing, requiring $(1-\beta)f(x_i,N) > f(x_{i+2},N)$. Since f is polynomially bounded, we conclude that $s = O(\frac{1}{\beta}\log n)$. We maintain s instances of Λ and s timestamps and indexes, thus space complexity is $s(g+\log n) = O(\frac{1}{\beta}\log n(g+\log n))$. It is easy to maintain a smooth histogram, using (α,β)-smoothness of f. We ensure that an index u becomes a successor of index $v < u$ only if at some point we have $(1-\beta)f(v,N) \leq f(u,N)$. Since f is (α,β)-smooth this implies that for any $N' > N$ we have $(1-\alpha)f(v,N') \leq f(u,N')$; thus the second requirement can be maintained. To ensure the third property, we traverse the list and if $(1-\beta)f(v,N) \leq f(u,N)$ for some u that is not a successor of v, we delete all indexes between u and v. This requires $O(\frac{1}{\beta}h\log n)$ time. (The time can be reduced to an amortized $O(h)$ when Λ supports merging of buckets.) Similarly, we solve the case when Λ approximates f on D, although the structure becomes more complicated.

Our approach is similar to exponential histograms in the sense that both methods capture gradual lessening of f using a logarithmic number of Λ instances. However, there is a critical difference between these approaches that makes our results possible. Exponential histograms divide W into distinct blocks $B_1,\ldots,B_k$. This requires a strong assumption about Λ, namely the ability to merge buckets. Further, the algorithm [15] requires $f(B_i)$ to be close to $\sum_{j>i} f(B_j)$, and that limits applicability of exponential histograms to additive functions. Smooth histograms maintain f on suffixes rather then on distinct parts of the window and require closeness between these suffixes, eliminating the above restrictions. The ability to work with suffixes is due to the smoothness of f; thus it is a critical property.

1.3 Roadmap

Section 2 briefly describes smooth histograms and states our main results. In Section 3 we apply smooth histograms to approximate weakly additive functions, L_p norms, frequency moments, length of largest increasing subsequence and geometric mean. Finally, in Section 4 we discuss our future work on frequency moments.

2 Smooth Histograms

Definition 1. *Function f is* (α,β)-smooth *if it preserves the following properties:*

1. $f(A) \geq 0$.
2. $f(A) \geq f(B)$ *for* $B \subseteq_r A$.
3. $f(A) \leq poly(n)$. [1]
4. *For any* $0 < \epsilon < 1$ *there exists* $\alpha = \alpha(\epsilon, f)$ *and* $\beta = \beta(\epsilon, f)$ *such that*
   - $0 < \beta \leq \alpha < 1$.

[1] Similar to [15], we assume that $f(A)/f(p_N) \leq poly(n)$. Otherwise, exponentially decreasing sequences will require linear memory for both smooth and exponential histograms.

- *If* $B \subseteq_r A$ *and* $(1-\beta)f(A) \leq f(B)$ *then* $(1-\alpha)f(A \cup C) \leq f(B \cup C)$ *for any adjacent* C.

In this section we assume that f is (α, β)-smooth and there exists an algorithm Λ that calculates f (precisely or approximately) on the whole stream D. We construct α-approximation algorithms for such f on sliding windows. Typically, $\alpha = \epsilon$, so we obtain ϵ-approximation; however, for weakly additive functions, we put $\alpha = 1 - \frac{1-\epsilon}{C_f}$. First, we assume that Λ calculates f precisely using g space and h operations per element. Λ applied on bucket $B(i,j)$ is denoted by $\Lambda(i,j)$.

Definition 2. *A* smooth histogram *is a structure that consists of an increasing set of indexes* $X_N = \{x_1, \ldots, x_s = N\}$ *and* s *instances of algorithm* Λ*, namely* $\Lambda_1, \ldots, \Lambda_s$ *with the following properties*

1. p_{x_1} *is expired,* p_{x_2} *is active or* $x_1 = 0$ *and* p_0 *is active.*
2. *For all* $i < s$ *one of the following holds*
   (a) $x_{i+1} = x_i + 1$ *and* $f(x_{i+1}, N) < (1-\beta)f(x_i, N)$.
   (b) $(1-\alpha)f(x_i, N) \leq f(x_{i+1}, N)$ *and if* $i+2 \leq s$ *then* $f(x_{i+2}, N) < (1-\beta)f(x_i, N)$.
3. $\Lambda_i = \Lambda(x_i, N)$ *maintains* $f(x_i, N)$.

Lemma 1. *It is possible to maintain a smooth histogram using* $O\left(\frac{1}{\beta}(g + \log n)\log n\right)$ *bits and* $O\left(\frac{1}{\beta}h \log n\right)$ *operations per element.*

Proof. Note that the conditions $2.(a)$ and $2.(b)$ may overlap, so it is possible that both conditions are true for some x_i. For $N = 1$, we put $x_1 = 1, s = 1$ and initiate Λ with p_{x_1}. Given a smooth histogram at step N, and the new element p_{N+1}, we execute the following update procedure.

I. For all i, calculate $f(x_i, N+1)$ using $\Lambda_i = \Lambda(x_i, N)$ and p_{N+1}.

II. Put $s = s + 1$ and $x_s = N + 1$ and initiate new algorithm instance $\Lambda(N+1, N+1)$.

III. For $i = 1 \ldots s-2$ do

   (a) Find the largest $j > i$ such that $f(x_j, N+1) \geq (1-\beta)f(x_i, N+1)$.

   (b) Delete all $x_t, i < t < j$ and all instances $\Lambda(x_t, N)$, and shift the list accordingly.

IV. Find the smallest i such that p_{x_i} is expired and $p_{x_{i+1}}$ is active. Delete all $x_j, j < i$ and Λ_j structures and change the enumeration accordingly.

Below we prove that the update procedure maintains a smooth histogram. It follows from the last operation that property 1 is preserved. Property 3 follows from the first two steps. To prove property 2, let $v < N+1$ be a fixed index from X_N that was not deleted during the update procedure. Let v' be the successor of v in the sequence X_N at step N; i.e., for some i we had $x_i = v, x_{i+1} = v'$.

If $v' \notin X_{N+1}$, let u and w be two successors of v in X_{N+1}. By the update procedure, it must be the case that $f(u, N+1) \geq (1-\beta)f(v, N+1) \geq (1-\alpha)f(v, N+1)$ and $f(w, N+1) < (1-\beta)f(v, N+1)$. Thus, $2.(b)$ is correct for v.

If $v' \in X_{N+1}$ and $v' > v+1$, let $N' \leq N$ be the step when v' became the successor of v. An update procedure implies that $f(v', N') \geq (1-\beta)f(v, N')$ and since f is (α, β)-smooth we have $f(v', N+1) \geq (1-\alpha)f(v, N+1)$. Let u be the successor (if one exists) of v' in X_{N+1}. Since v' was not deleted, we have $f(u, N+1) < (1-\beta)f(v, N+1)$. Thus, $2.(b)$ is correct for v.

Finally, if $v' \in X_{N+1}$ and $v' = v+1$, we have two cases. If $f(v', N+1) < (1-\beta)f(v, N+1)$, then $2.(a)$ is true. Otherwise, we have $f(v', N+1) \geq (1-\beta)f(v, N+1) \geq (1-\alpha)f(v, N+1)$. Let u be the successor (if one exists) of v' in X_{N+1}. Similarly, $f(u, N+1) < (1-\beta)f(v, N+1)$. Thus, $2.(b)$ is correct for v, and property 2 is preserved for any $v < N+1$.

Let us estimate size s of the sequence X_N. By the properties above, we have that for any i either $f(x_{i+2}, N)$ or $f(x_{i+1}, N)$ are less then $(1-\beta)f(x_i, N)$. This and the fact that f is polynomially bounded imply $s = O\left(\frac{1}{\beta}\log n\right)$. Since we maintain exactly s instances of algorithm Λ and timestamps, the space complexity is $O(s(g+\log n)) = O\left(\frac{1}{\beta}(g+\log n)\log n\right)$, and the time complexity per element is $O(sh) = O\left(\frac{1}{\beta}h\log n\right)$. □

Theorem 1. *Let f be a (α, β)-smooth function. If there exists an algorithm Λ that calculates precisely f on streams, uses space g and performs h operations per each element, then there exists an algorithm Λ' that calculates α-approximation of f on sliding windows and uses $O\left(\frac{1}{\beta}(g+\log n)\log n\right)$ bits and $O\left(\frac{1}{\beta}h\log n\right)$ operations per element.*

Proof. The algorithm maintains a smooth histogram and outputs $f(x_2, N)$ as an approximation of f on the window. To prove that this is a α-approximation, let j be the index of the last active element, so the precise value is $f(j, N)$. If $2.(a)$ is correct for x_1, then by property 1, $j = x_2$ and the answer is precise. Otherwise $2.(b)$ is correct for x_1, and we have, since f is monotonic: $f(j, N) \geq f(x_2, N) \geq (1-\alpha)f(x_1, N) \geq (1-\alpha)f(j, N)$. □

In many cases it is impossible to calculate f precisely. Below we show how to adapt approximation algorithms to sliding windows. We assume that Λ maintains $\hat{\epsilon}$-approximation of f on D, $\hat{\epsilon} \leq \frac{\beta}{4}$, and uses $g(\hat{\epsilon})$ space and $h(\hat{\epsilon})$ operations per element. We call such approximation f'.

Definition 3. *The approximate smooth histogram is a structure that consists of an increasing set of indexes $X_N = \{x_1, \ldots, x_s = N\}$ and s instances of algorithm Λ, namely $\Lambda_1, \ldots, \Lambda_s$ with the following properties*

1. *p_{x_1} is expired, p_{x_2} is active or $x_1 = 0$.*
2. *For all $i < s$, one of the following holds*
   - *(a) $x_{i+1} = x_i + 1$ and $f'(x_{i+1}, N) < (1-\frac{\beta}{2})f'(x_i, N)$.*
   - *(b) $(1-\alpha)f(x_i, N) \leq f(x_{i+1}, N)$ and if $i+2 \leq s$ then $f'(x_{i+2}, N) < (1-\frac{\beta}{2})f'(x_i, N)$.*

3. $\Lambda_i = \Lambda(x_i, N)$ *maintains* $f'(x_i, N)$.

Lemma 2. *It is possible to maintain an approximate smooth histogram using* $O\left(\frac{1}{\beta}(g(\hat{\epsilon}) + \log n)\log n\right)$ *bits and* $O\left(\frac{1}{\beta}h(\hat{\epsilon})\log n\right)$ *operations per element.*

Proof. The update procedure repeats Lemma 1. For $N = 1$, we put $x_1 = 1, s = 1$ and initiate Λ with p_{x_1}. Given an approximate smooth histogram at step N, and the new element p_{N+1}, we execute the following update procedure.

I. For all i, calculate $f'(x_i, N+1)$ using $\Lambda_i = \Lambda(x_i, N)$ and p_{N+1}.

II. Put $s = s + 1$ and $x_s = N + 1$, and initiate a new algorithm instance $\Lambda(N+1, N+1)$.

III. For $i = 1 \dots s-2$ do

   (a) Find the largest $j > i$ such that $f'(x_j, N+1) \geq (1 - \frac{\beta}{2})f'(x_i, N+1)$.

   (b) Delete all $x_t, i < t < j$ and all instances $\Lambda(x_t, N)$, and shift the list accordingly.

IV. Find the smallest i such that p_{x_i} is expired and $p_{x_{i+1}}$ is active. Delete all $x_j, j < i$ and Λ_j structures, and change the enumeration accordingly.

It follows from the last operation that property 1 is preserved. Property 3 follows from the first two steps. To prove property 2, let $v < N+1$ be a fixed index from X_N that was not deleted during the update procedure. Let v' be the successor of v in the sequence X_N at step N; i.e., for some i we had $x_i = v, x_{i+1} = v'$.

If $v' \notin X_{N+1}$, let u and w be two successors of v in X_{N+1}. By update procedure, it must be the case that $f'(u, N+1) \geq (1 - \frac{\beta}{2})f'(v, N+1)$, thus

$$(1-\alpha)f(v, N+1) \leq (1-\beta)f(v, N+1) \leq$$

$$(1 - \frac{\beta}{2})\frac{1 - \frac{\beta}{4}}{1 + \frac{\beta}{4}}f(v, N+1) \leq \frac{1 - \frac{\beta}{2}}{1 + \frac{\beta}{4}}f'(v, N+1) \leq$$

$$\frac{f'(u, N+1)}{1 + \frac{\beta}{4}} \leq f(u, N+1).$$

Also, it must be the case that $f'(w, N+1) < (1 - \frac{\beta}{2})f'(v, N+1)$, thus, $2.(b)$ is correct for v.

If $v' \in X_{N+1}$ and $v' > v + 1$, let $N' \leq N$ be the step when v' became the successor of v. The update procedure implies that $f'(v', N') \geq (1 - \frac{\beta}{2})f'(v, N')$ and thus $(1-\beta)f(v, N') \leq f(v', N')$. Since f is (α, β)-smooth we have $f(v', N+1) \geq (1-\alpha)f(v, N+1)$. Let u be the successor (if one exists) of v' in X_{N+1}. Since v' was not deleted, we have $f'(u, N+1) < (1 - \frac{\beta}{2})f(v, N+1)$. Thus, $2.(b)$ is correct for v.

Finally, if $v' \in X_{N+1}$ and $v' = v+1$, we have two cases. If $f'(v', N+1) < (1 - \frac{\beta}{2})f'(v, N+1)$, then $2.(a)$ is true. Otherwise, we have $f(v', N+1) \geq (1-\alpha)f(v, N+1)$, repeating the calculations

above. Let u be the successor of v' in X_{N+1}. Similarly, $f'(u, N+1) < (1-\frac{\beta}{2})f'(v, N+1)$. Thus, $2.(b)$ is correct for v. Thus property 2 is preserved for any $v < N+1$.

Properties of a histogram imply that for any i, either $f'(x_{i+2}, N)$ or $f'(x_{i+1}, N)$ are less then $(1-\frac{\beta}{2})f'(x_i, N)$. This property, the fact that f' is at least a $\frac{\beta}{4}$-approximation of f and the fact that f is polynomially bounded, imply $s = O\left(\frac{1}{\beta}\log n\right)$. Since we maintain exactly s instances of algorithm Λ, the space complexity is $O(sg) = O\left(\frac{1}{\beta}(g(\hat{\epsilon}) + \log n)\log n\right)$, and the time complexity per element is $O(sh) = O\left(\frac{1}{\beta}h\log n\right)$. □

Theorem 2. *Let f be a (α, β)-smooth function. If there exists an algorithms Λ that maintains an $\hat{\epsilon}$-approximation of f on D, using space $g(\hat{\epsilon})$ and performing $h(\hat{\epsilon})$ operations per stream element, then there exists an algorithm Λ' that maintains a $(\alpha+\hat{\epsilon})$-approximation of f on sliding windows and uses $O\left(\frac{1}{\beta}(g(\hat{\epsilon}) + \log n)\log n\right)$ bits and $O\left(\frac{1}{\beta}h(\hat{\epsilon})\log n\right)$ operations per element.*

Proof. The algorithm maintains an approximate smooth histogram and outputs $f'(x_2, N)$ as an approximation of f on the window. Let j be the index of the last active element, so the precise value is $f(j, N)$. If $2.(a)$ is correct for x_1, then by property 1, $j = x_2$ and the answer is a $\hat{\epsilon}$-approximation of $f(j, N)$. Otherwise $2.(b)$ is correct for x_1, and we have

$$(1+\alpha+\hat{\epsilon})f(j, N) \geq \frac{1+\alpha+\hat{\epsilon}}{1+\frac{\beta}{4}}f'(x_2, N) \geq f'(x_2, N).$$

Also,

$$(1-\alpha-\hat{\epsilon})f(j, N) \leq (1-\alpha-\hat{\epsilon})f(x_1, N) \leq$$
$$(1-\hat{\epsilon})f(x_2, N) \leq f'(x_2, N).$$

□

Similarly, we can approximate functions for which there exists algorithm Λ that maintains a $(\hat{\epsilon}, \delta)$-approximation on D. The proof remains the same, we only need to ensure that probability of failure is at most δ. Recall that the smooth histogram uses $O(\frac{1}{\beta}\log n)$ instances of Λ. Thus, if for each instance we limit the probability of failure by $\frac{\delta\beta}{\log n}$ then, by union bound, the total probability of failure will be at most δ. We obtain the following theorem.

Theorem 3. *Let f be a (α, β)-smooth function. If there exists an algorithm Λ that maintains an $(\hat{\epsilon}, \delta)$-approximation of f on D, using space $g(\hat{\epsilon}, \delta)$ and performing $h(\hat{\epsilon}, \delta)$ operations per stream element, then there exists an algorithm Λ' that maintains a $(\alpha+\hat{\epsilon})$-approximation of f on sliding windows and uses $O\left(\frac{1}{\beta}(g(\hat{\epsilon}, \frac{\delta\beta}{\log n}) + \log n)\log n\right)$ bits and $O\left(\frac{1}{\beta}h(\hat{\epsilon}, \frac{\delta\beta}{\log n})\log n\right)$ operations per element.*

Note that the proofs above are correct for sequence-based and timestamp-based widows.

3 Applications

3.1 Weakly Additive Functions

Let f be a weakly additive that can be precisely computed on D using space g and time h. It was proved in [15] that f can be approximated on sliding windows with relative error $\epsilon C_f^2 + C_f - 1$, space $O(\frac{1}{\epsilon}(g + \log n) \log n)$, and amortized time $O(h)$. Smooth histograms improve the relative error, preserving space and time complexities.

Lemma 3. *Weakly additive function f with parameter C_f is $(1 - \frac{1-\epsilon}{C_f}, \epsilon)$-smooth.*

Proof. Let A be a bucket and B be its suffix such that $(1 - \epsilon)f(A) \leq f(B)$. For any adjacent bucket C we have

$$(1 - (1 - \frac{1 - \epsilon}{C_f}))f(A \cup C) \leq (1 - \epsilon)(f(A) + f(C)) \leq$$

$$f(B) + f(C) \leq f(B \cup C).$$

□

Corollary 1. *There exists an algorithm that maintains a $(1 - \frac{1-\epsilon}{C_f})$-approximation f on sliding windows using $O(\frac{1}{\epsilon}(g + \log n) \log n)$ space and $O(h)$ amortized time per element.*

Proof. By applying Theorem 1, we almost obtain the result. The only problem is the logarithmic number of operations per element. This can be reduced using the fact that sketches are composable. Instead of recalculating f on buckets for each new element, we do it for every $\log n$-th element, collecting all new elements in an auxiliary buffer. Let v be index of the first collected point. Since sketches are composable, we can compute $f(u, N)$ using sketch for $f(u + 1, N)$ and p_u for any $v \leq u < N$. Using sketch for $f(u, N)$, we compute $f(x_i, N)$ for all i. The rest of the algorithm remains unchanged. □

Also, it is shown in [15] that given an algorithm that maintains $\hat{\epsilon}$-approximation of f on D, f can be approximated on sliding windows with relative error $(1 + \hat{\epsilon})^2 \epsilon C_f^2 + C_f - 1 + \hat{\epsilon}$, using the same space and time. Using Theorem 2 and repeating the arguments from Corollary 1, we obtain:

Corollary 2. *There exists an algorithm that maintains a $\left(1 - \frac{1-\epsilon}{C_f} + \hat{\epsilon}\right)$-approximation f on sliding windows using $O(\frac{1}{\epsilon}(g + \log n) \log n)$ space and $O(h)$ amortized time per element.*

Note that $1 - \frac{1-\epsilon}{C_f} \leq \epsilon C_f^2 + C_f - 1$ and $1 - \frac{1-\epsilon}{C_f} + \hat{\epsilon} \leq (1 + \hat{\epsilon})^2 \epsilon C_f^2 + C_f - 1 + \hat{\epsilon}$. For large C_f, the improvement becomes significant (relative improvement is comparable with C_f).

3.2 L_p Norms

We use smooth histograms to approximate L_p norms on sliding windows. Recall that for this problem D represents a vector $X = \langle x_1, \ldots, x_m \rangle$. Each element is a pair $(i, a), i \in [m], a \in [M]$ for some positive $M \leq n^{O(1)}$. The value of the i-th coordinate is given by $x_i = \sum_{(i,a) \in W} a$, and $L_p(W)$ is defined as $(\sum_{i=1}^{m} x_i^p)^{\frac{1}{p}}$. For this model Datar, Gionis, Indyk and Motwani [15] showed a $(4\epsilon(1+\epsilon)^2 + 1 + \epsilon, \delta)$-approximation for $p \in [1, 2]$. We extend this result to any p and improve the approximation parameter for $p \in [1, 2]$. Below we prove that L_p is a smooth function.

Lemma 4. *For $p > 1$ L_p is a $(\epsilon, \frac{\epsilon^p}{p})$-smooth function. For $p < 1$ L_p is a (ϵ, ϵ)-smooth function.*

Proof. It is easy to see that in this model L_p satisfied the properties of function f; i.e., it is monotonic on buckets, polynomially bounded and positive. Let $p > 1$ and X, Y be vectors that are represented by buckets A, B ($B \subseteq_r A$) such that $(1 - \frac{\epsilon^p}{p}) L_p(A) \leq L_p(B)$. Recall that in our model, $x_i \geq y_i$ for all $i \in [m]$. We have

$$(1 - \epsilon^p)\|X\|_p^p \leq (1 - \frac{\epsilon^p}{p})^p \|X\|_p^p \leq \|Y\|_p^p$$

Since $\|X\|_p^p \geq \|Y\|_p^p + \|X - Y\|_p^p$, we obtain $\|X - Y\|_p \leq \epsilon \|X\|_p \leq \epsilon \|X + Z\|_p$ for any Z that is represented by adjacent bucket C (recall that $z_i \geq 0$). By triangle inequality

$$\|X + Z\|_p \leq \|Y + Z\|_p + \|X - Y\|_p \leq \|Y + Z\|_p + \epsilon \|X + Z\|_p.$$

That concludes the lemma for $p > 1$. For $p < 1$, let X, Y be vectors such that $(1 - \epsilon)\|X\| \leq \|Y\|$. We have for any Z:

$$\|Y + Z\|_p^p = \|Y\|_p^p + \sum_{i=1}^{m} ((y_i + z_i)^p - y_i^p) \geq$$

$$(1 - \epsilon)^p \|X\|_p^p + (1 - \epsilon)^p \sum_{i=1}^{m} ((x_i + z_i)^p - x_i^p) =$$

$$(1 - \epsilon)^p \|X + Z\|_p^p.$$

The inequality follows from the assumption that $\|Y\|_p \geq (1 - \epsilon)\|X\|_p$ and the fact that function $f(x) = (x + a)^p - x^p$ is decreasing for $x \geq 0$ and $p < 1, a > 0$.

□

For $p > 2$ we apply the algorithm of Bhuvanagiri, Ganguly, Kesh and Saha [8] for frequency moments. In our model, frequency moments are simply $L_p(x) = F_p(x)^{\frac{1}{p}}$. Thus, for $p > 1$, the ϵ-approximation for F_p is also the ϵ-approximation for L_p. Recall their result.

Theorem 4. *[8] There exists an algorithm that computes a $(\epsilon, \frac{3}{4})$-approximation of $L_p, p > 2$ using $O\left(\frac{p^2}{\epsilon^{2+\frac{4}{p}}} m^{1-\frac{2}{p}} \log^2 N (\log m + \log N)\right)$ bits.*

Corollary 3. *There exists an algorithm that uses $\tilde{O}(m^{1-\frac{2}{p}})$ memory and calculates a $(\epsilon + \frac{\epsilon^p}{4p}, \delta)$-approximation of $L_p, p > 2$ norm over sliding windows.*

Proof. We apply Theorem 3 and the algorithm from [8] (after amplification). The resulting space complexity is $O\left(\frac{p^3}{\epsilon^{2p+4}} m^{1-\frac{2}{p}} \log^3 n (\log m + \log n) \log \frac{\log n}{\delta\epsilon}\right)$. □

For $p < 1$, similar to [15] we can apply the algorithm of Indyk [22]. Recall that this algorithm provides a (ϵ, δ)-approximation of $L_p(D), p \in [1, 2]$ and uses $O(\frac{1}{\epsilon^2} \log M \log \frac{1}{\delta})$ bits for each sketch. Additionally, $O(\frac{1}{\epsilon^2} \log M \log \frac{m}{\delta} \log \frac{1}{\delta})$ bits are required and are common for all sketches. Since $L_p(W)$ is a (ϵ, ϵ)-smooth function, we can apply Theorem 3 using the algorithm above.

Corollary 4. *There exists an algorithm that calculates a $(\epsilon + \frac{\epsilon}{4}, \delta)$-approximation of $L_p, p < 1$ norm on sliding windows and uses space $O\left(\frac{1}{\epsilon} \log n (\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$.*

For $p \in [1, 2]$ we present two solutions. We can repeat the arguments above or we can apply Corollary 1 noting that $L_p^p, p \in [1, 2]$ is a weakly additive with $C_f \leq 2$.

Corollary 5. *It is possible to maintain a $(\epsilon + \frac{\epsilon^p}{p}, \delta)$-approximation of $L_p, p \in [1, 2]$ over sliding windows using $O\left(\frac{1}{\epsilon^p} \log n (\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$ bits. Also, it is possible to maintain a $(\frac{1+\epsilon}{2} + \hat{\epsilon}, \delta)$-approximation using $O\left(\frac{1}{\epsilon} \log n (\frac{1}{\epsilon^2} \log M \log \frac{\log n}{\delta\epsilon} + \log n)\right)$ bits.*

The second result strictly improves [15], while the first one significantly improves relative error and increases memory requirements by $\frac{1}{\epsilon^{p-1}}$.

3.3 Frequency Moments

The observations above are valid for frequency moments, for constant p.

Lemma 5. *For $p \geq 1$, F_p is a $(\epsilon, \frac{\epsilon^p}{p^p})$-smooth function. For $p < 1$, F_p is a (ϵ, ϵ)-smooth function.*

Proof. Let $p \geq 1$ and X, Y be two vectors such that

$$\left(1 - \frac{\epsilon^p}{p^p}\right) \sum_{i=1}^{m} x_i^p \leq \sum_{i=1}^{m} y_i^p.$$

We have

$$\sum_{i=1}^{m} (x_y - y_i)^p \leq \frac{\epsilon^p}{p^p} \sum_{i=1}^{m} x_i^p$$

Thus for any vector Z that is added to X and Y, we have

$$(\sum_{i=1}^{m} (x_y - y_i)^p)^{\frac{1}{p}} \leq \frac{\epsilon}{p} (\sum_{i=1}^{m} (x_y + z_i)^p)^{\frac{1}{p}}$$

and by triangle inequality

$$(\sum_{i=1}^{m} (y_i + z_i)^p)^{\frac{1}{p}} \geq$$

$$(\sum_{i=1}^{m}(x_y + z_i)^p)^{\frac{1}{p}} - (\sum_{i=1}^{m}(x_i - y_i)^p)^{\frac{1}{p}} \geq$$

$$(1 - \frac{\epsilon}{p})(\sum_{i=1}^{m}(x_y + z_i)^p)^{\frac{1}{p}}.$$

Thus,

$$\sum_{i=1}^{m}(y_i + z_i)^p \geq (1 - \frac{\epsilon}{p})^p \sum_{i=1}^{m}(x_y + z_i)^p \geq$$

$$(1 - \epsilon)\sum_{i=1}^{m}(x_y + z_i)^p.$$

If $p < 1$ we have for X, Y, Z as above, assuming $(1 - \epsilon)F_p(X) \leq F_p(Y)$:

$$\sum_{i=1}^{m}(y_i + z_i)^p = \sum_{i=1}^{m} y_i^p + \sum_{i=1}^{m}((y_i + z_i)^p - y_i^p) \geq$$

$$(1 - \epsilon)\sum_{i=1}^{m} x_i^p + (1 - \epsilon)\sum_{i=1}^{m}((x_i + z_i)^p - x_i^p) =$$

$$(1 - \epsilon)\sum_{i=1}^{m}(x_i + z_i)^p.$$

The inequality follows from the assumption above and the fact that function $(x + a)^p - x^p$ is decreasing for $p < 1, a > 0$.

□

Now, we can apply the algorithm of Bhuvanagiri, Ganguly, Kesh and Saha [8].

Corollary 6. *For constant $p > 2$, there exists an algorithm that uses $\tilde{O}(m^{1-\frac{2}{p}})$ memory and calculates a (ϵ, δ)-approximation of p-th frequency moment over sliding windows.*

Proof. The proof is identical to the proof of Corollary 3. □

3.4 Length of Longest Increasing Subsequence

First, we show that our technique is applicable.

Lemma 6. *$LIS - length$ is a (ϵ, ϵ)-smooth function.*

Proof. Let $Y \subseteq_r X$ be two buckets such that $(1 - \epsilon)|LIS(X)| \leq |LIS(Y)|$. It is enough to show that for any new element z we guarantee $(1 - \epsilon)|LIS(X \cup \{z\})| \leq |LIS(Y \cup \{z\})|$. If $|LIS(Y \cup \{z\})| > |LIS(Y)|$ or $|LIS(X \cup \{z\})| = |LIS(X)|$, the inequality obviously holds. We prove below that these are the only possible cases. Assume, in contradiction, that $|LIS(Y \cup \{z\})| =$

$|LIS(Y)|$ and $|LIS(X \cup \{z\})| > |LIS(X)|$. Let $x_1, \ldots, x_i$ and $y_1, \ldots, y_j$ be indexes of some longest increasing subsequences of X, Y, respectfully.

Under the above assumptions, $x_i \neq y_j$, since otherwise we can add z to $LIS(Y)$. We prove that $x_i > y_j$ and $p_{x_i} < p_{y_j}$. Suppose, in contradiction, that $x_i \leq y_j$. If $p_{x_i} > p_{y_j}$, then $|LIS(Y \cup \{z\})| > |LIS(Y)|$ since $z > p_{x_i} > p_{y_j}$. Otherwise if $p_{x_i} \leq p_{y_j}$ then we can add p_{y_j} to $LIS(X)$ and increase its length. Both cases contradict the assumptions. If $x_i > y_j$ and $p_{x_i} \geq p_{y_j}$, then we can increase $|LIS(Y)|$ by adding p_{x_i}.

Obviously $x_1 \notin Y$, otherwise we can increase $|LIS(Y)| = |LIS(X)|$ by adding z. Let x_l be the largest index that does not belong to Y. If $j < i - l$, we violate the maximality of $y_1, \ldots, y_j$ and if $j = i - l$, we violate the assumption that $LIS(Y)$ cannot be increased by adding z; thus $j > i - l$. In this case, it must be that $p_{x_l} > p_{y_1}$, since otherwise we can replace $x_{l+1}, \ldots, x_i$ with $y_1, \ldots, y_j$ and increase $LIS(X)$. Thus we proved that $p_{x_l} > p_{y_1}$.

Let s be the maximal number for which there exists $x_t > y_s$, and for every t s.t. $y_s < x_t$ we have $p_{y_s} < p_{x_t}$. There exists at least one such number, namely y_1. Indeed, we have, for any $x_t > y_1$, $p_{y_1} < p_{x_l} \leq p_{x_t}$. There exists at least one such t, since $x_i > y_j > y_1$.

Let t be the minimal number such that $y_s < x_t$. Note that $x_{t-1} < y_s$ and $p_{x_{t-1}} \leq p_{y_{s+1}}$ (otherwise for every $x_{t'} > x_{s+1}$ we have $p_{x_{t'}} \geq p_{x_{t-1}} > p_{y_{s+1}}$ that contradicts the maximality of s). Therefore the following sequences are increasing: $y_1, \ldots, y_s, x_t, \ldots, x_i$ and $x_1, \ldots, x_{t-1}, y_{s+1}, \ldots, y_j$. So, it must be that $j - s - 1 = i - t$. But in this case we can increase $LIS(Y)$ by adding z to the first sequence. Therefore, the lemma is correct. □

Therefore, we can apply Theorem 1 using the algorithm of Sun and Woodruff [29].

Corollary 7. *Let D be a steam of integers such that $p_i \in [L]$. There exists an algorithm that computes a ϵ-approximation of $|LIS(W)|$, where W is the current window. The algorithm uses space*

$$O(\frac{1}{\epsilon} \log n(|LIS(W)| \log \frac{L}{|LIS(W)|} + \log n)).$$

3.5 Geometric Mean

Let D be a stream of real numbers. Recall that $GM(W) = \left(\prod_{i=N}^{N-n} p_i\right)^{\frac{1}{n}}$. We assume that we need k bits to store the results.

Corollary 8. *There exists an algorithm that computes a ϵ-approximation of geometric mean over sequence-based sliding windows using $O\left(\frac{1}{\epsilon}(k + \log n) \log n\right)$ space.*

Proof. We divide D into two sub-streams, $D_{<1} = \{p_i | p_i \leq 1\}$ and $D_{>1} = \{p_i | p_i \geq 1\}$. The active window W is correspondingly separated into $W_{<1}$ and $W_{>1}$. We define two auxiliary functions on buckets $h(B) = \left(\prod_{i \in B} p_i\right)^{\frac{1}{n}}$ and $g(B) = \left(\prod_{i \in B} \frac{1}{p_i}\right)^{\frac{1}{n}}$. We apply our method on these two sub-streams and then combine the results to obtain the approximation. Note that n is fixed, but $W_{>1}, W_{<1}$ are timestamp-based windows.

It is easy to see that h is (ϵ, ϵ)-smooth on $D_{>1}$. Indeed, its is monotonic (for fixed n), polynomially bounded, and new elements do not change the ratio $\frac{h(A)}{h(B)}$ for any buckets A, B. Thus,

we can maintain a ϵ-approximation of $h(W) = GM(W_{>1})$ using $O\left(\frac{1}{\epsilon}(k + \log n)\log n\right)$ memory. Similarly, g is a (ϵ, ϵ)-smooth function on $D_{<1}$, and thus we obtain a ϵ-approximation of $g(W) = \frac{1}{GM(W_{<1})}$. Dividing h by g, we obtain a 2ϵ-approximation of the geometrical mean. $\square$

4 Future Work on Frequency Moments

In the current paper we prove that F_p is $(\epsilon, \frac{\epsilon^p}{p^p})$-smooth, and thus smooth histograms can be used to approximate frequency moments with $\tilde{O}(p^p m^{1-\frac{2}{p}})$ bits. Thus, for constant $p > 2$ we obtain optimal results. However, the additional factor of p^p makes the smooth-histogram approach infeasible for large p, which current paper does not handle. In a follow-up work [9], we show how to handle frequency moments for arbitrary p. In particular, we show how to compute F_p using $\tilde{O}(m^{1-\frac{1}{p}})$ bits for any $p > 2$.

References

[1] P. K. Agarwal, S. Har-Peled, and K. Varadarajan, "Geometric approximation via coresets", *Combinatorial and Computational Geometry* (J. E. Goodman, J. Pach, and E. Welzl, eds.), Math. Sci. Research Inst. Pub., Cambridge, 2005.

[2] C. Aggarwal (editor), "Data Streams: Models and Algorithms", *Springer Verlag*, 2007.

[3] N. Alon, Y. Matias, M.Szegedy, "The space complexity of approximating the frequency moments". *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 20–29, 1996.

[4] A. Arasu, G. S. Manku, "Approximate counts and quantiles over sliding windows", *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2004.

[5] B. Babcock, M. Datar, R. Motwani, "Sampling from a moving window over streaming data", *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 633–634, 2002.

[6] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan, "Maintaining variance and k-medians over data stream windows", *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 234–243, 2003.

[7] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, "An Information Statistics Approach to Data Stream and Communication Complexity", *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pp. 209–218, 2002.

[8] L. Bhuvanagiri, S. Ganguly, D. Kesh, C. Saha, "Simpler algorithm for estimating frequency moments of data streams", *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 708–713, 2006.

[9] V. Braverman, R. Ostrovsky, C. Zaniolo, "Succinct sampling on streams", *http://arxiv.org/abs/cs.DS/0702151*, 2007.

[10] A. Chakrabarti, S. Khot, X. Sun, "Near-optimal lower bounds on the multi-party communication complexity of set-disjointness", *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, 2003.

[11] T. Chan, B. Sadjad, "Geometric Optimization Problems Over Sliding Windows", *Proc 15th Annual International Symposium on Algorithms and Computation*, pp. 246–258, 2004.

[12] E. Chen, L. Yang, H. Yuan, "Longest Increasing Subsequences in Windows based on Canonical Antichain Partition",*Proceedings of 16th Annual International Symposium on Algorithms and Computation*, pp. 1153–1162, 2005.

[13] Y. Chi, H. Wang, P. S. Yu, R. R. Muntz, "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window", *Fourth IEEE International Conference on Data Mining* (ICDM'04), pp. 59–66, 2004.

[14] D. Coppersmith , R. Kumar, "An improved data stream algorithm for frequency moments", *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp.151–156 , 2004.

[15] M. Datar, A. Gionis, P. Indyk, R. Motwani, "Maintaining stream statistics over sliding windows", *SIAM J. Comput. 31, 6*, pp.1794–1813, 2002.

[16] M. Datar, S. Muthukrishnan, "Estimating Rarity and Similarity over Data Stream Windows", *Proceedings of the 10th Annual European Symposium on Algorithms*, pp.323–334, 2002.

[17] J. Feigenbaum, S. Kannan, J. Zhan, "Computing diameter in the streaming sliding-window models", *Algorithmica 41*, pp. 25–41, 2004.

[18] S. Ganguly. "Estimating Frequency Moments of Update Streams using Random Linear Combinations". *Proceedings of the 8th International Workshop on Randomized Algorithms*, pp. 369-380, 2004.

[19] P. B. Gibbons, S. Tirthapura, "Distributed streams algorithms for sliding windows", *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pp.10–13, 2002.

[20] P. Gopalan, T. S. Jayram, R. Krauthgamer, R. Kumar, "Estimating the Sortedness of a Data Stream", *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[21] D. Gusfield, "Algorithms on Strings, Trees, and Sequences", *Cambridge University Press*, 1997.

[22] P. Indyk, "Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation", *Proc. IEEE Symp. Foundations of Computer Science*, pp. 189–197, 2000.

[23] P. Indyk, D. Woodruff, "Optimal approximations of the frequency moments of data streams", *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp.202–208, 2005.

[24] L. K. Lee, H. F. Ting, "Frequency counting and aggregation: A simpler and more efficient deterministic scheme for finding frequent items over sliding windows", *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (PODS '06), pp. 290–297, 2006.

[25] L. K. Lee, H. F. Ting, "Maintaining significant stream statistics over sliding windows", *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp.724–732, 2006.

[26] D. Liben-Nowell, E. Vee, A. Zhu, "Finding Longest Increasing and Common Subsequences in Streaming Data", *11th International Computing and Combinatorics Conference*, 2005.

[27] S. Muthukrishnan, "Data Streams: Algorithms And Applications" *Foundations and Trends in Theoretical Computer Science*, Volume 1, Issue 2.

[28] P. Pevzner, "Computational Molecular Biology", *Elsevier Science Ltd.*, 2003.

[29] X. Sun, D. Woodruff, "The Communication and Streaming Complexity of Computing the Longest Common and Increasing Subsequences", *SODA*, 2007.