Space-optimal Heavy Hitters with Strong Error Bounds

RADU BERINDE and PIOTR INDYK
MIT

and
GRAHAM CORMODE
AT&T Labs–Research

and
MARTIN J. STRAUSS
University of Michigan

The problem of finding heavy hitters and approximating the frequencies of items is at the heart of many problems in data stream analysis. It has been observed that several proposed solutions to this problem can outperform their worst-case guarantees on real data. This leads to the question of whether some stronger bounds can be guaranteed. We answer this in the positive by showing that a class of "counter-based algorithms" (including the popular and very space-efficient FREQUENT and SPACESAVING algorithms) provide much stronger approximation guarantees than previously known. Specifically, we show that errors in the approximation of individual elements do not depend on the frequencies of the most frequent elements, but only on the frequency of the remaining "tail." This shows that counter-based methods are the most space-efficient (in fact, space-optimal) algorithms having this strong error bound.

This tail guarantee allows these algorithms to solve the "sparse recovery" problem. Here, the goal is to recover a faithful representation of the vector of frequencies, f. We prove that using space $O(k)$, the algorithms construct an approximation f^* to the frequency vector f so that the L_1 error $\|f - f^*\|_1$ is close to the best possible error $\min_{f'} \|f' - f\|_1$, where f' ranges over all vectors with at most k non-zero entries. This improves the previously best known space bound of about $O(k \log n)$ for streams without element deletions (where n is the size of the domain from which stream elements are drawn). Other consequences of the tail guarantees are results for skewed (Zipfian) data, and guarantees for accuracy of merging multiple summarized streams.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: frequency estimation, heavy hitters, streaming algorithms

1. INTRODUCTION

In many modern applications, very large quantities of data are generated at very high rates. This data is often too large to store and index in its complete form. Instead, it is often preferable to process this data in a *streaming* fashion. The streaming paradigm is to take a single pass over the huge stream of updates, and to store only a compact summary. From this summary, the goal is to discover properties of the input, and give strong guarantees on the quality of the resulting answers. The study of streaming algorithms is hence concerned with designing methods to build such summaries which are suited for approximating certain properties of the input.

The streaming approach is relevant for many applications:

—In large scale data processing and data warehousing, streams of updates arise as large quantities of data are loaded into the system. Here, compact summaries can quickly give approximate answers to ad hoc queries much faster than retrieving the full data and computing the exact answer [Hershberger et al. 2005; Beyer and Ramakrishnan 1999; Fang et al. 1998; Han et al. 2001].

—In large ISP networks, the packets in transit across the network are never stored centrally by the service provider: the gigabit speeds of transmission mean that storing even the data for a short period requires a vast amount of storage. Instead, it is desirable to track statistics about the network usage, popular destinations, patterns of traffic and so on, based on updating a very compact summary as each data packet is seen: this gives us exactly the streaming model [Arasu et al. 2003; Cormode et al. 2003; Demaine et al. 2002; Estan and Varghese 2001].

—Within distributed sensor networks, constraints on the memory of each sensor also mean that it is impractical to retain the full set of readings made by each sensor. Instead, various statistics and summaries must be kept that can be updated as new readings are made. When queries are posed to the network, these can either be answered by sending the query to each sensor, and having the sensor return a partial answer; or by having each sensor periodically send its summary to a central location. In the latter case, this motivates the additional requirement that summaries can be combined together to give a summary of the total data observed [Bonnet et al. 2001; Shrivastava et al. 2004].

—Lastly, the *compressed sensing* paradigm that has emerged over the past few years is based on making a small number of measurements of a very high-dimensional signal, so that the results of these measurements allow an approximate version of the signal to be reconstructed. There are two ways in which streaming algorithms have impact on compressed sensing: firstly, many existing streaming algorithms have been adapted to provide measurement schemes in the compressed sensing setting; and secondly, new streaming algorithms are needed to solve streaming versions of the compressed sensing problem, where information about the signal arrives at the sensor in a streaming fashion [Gilbert et al. 2007; Candès et al. 2006].

For a broader overview of the motivating scenarios, problems, and algorithms in the streaming area, we refer the reader to the tutorials, surveys, and classes on the topic [Indyk 2007; Muthukrishnan 2005; Garofalakis et al. 2002].

The Heavy Hitters Problem. Our focus in this paper is on one of the quintessential problems in data stream algorithms, the so-called "heavy hitters" or "frequent items" problem. The problem can be stated quite simply: given a stream of items (possibly with weights attached), find those items with the greatest total weight. This is an intuitive problem, that applies to many natural questions: given a stream of search engine queries, which are the most frequently occurring terms? Given a stream of supermarket transactions and prices, which items have the highest total dollar sales? Over the past few decades, there have been a very large number of papers which address exactly this question, or variations thereof.

For more background and history, see the recent survey of Cormode and Hadjieleftheriou [2008].

Beyond the direct utility of finding the heavy hitters, it turns out that this simple question is a core subproblem of many more complex computations over data streams. For example:

—In methods for estimating the entropy of a sequence, to get an accurate answer when the entropy is low, it is necessary to determine the (approximate) empirical probability of the most frequent item(s), and consider this separately to the entropy of the remaining stream [Chakrabarti et al. 2007; Ganguly and Lakshminath 2006]. This is accomplished by finding the heavy hitters in the stream, in parallel to computing other statistics.

—In clustering geometric data that is presented as a stream of points, grids at different granularities are imposed over the data, so that the cost of a proposed clustering can be estimated based on summing the counts of all points covered by clusters of a given diameter [Indyk 2004]. The total number of points covered is estimated by using heavy hitter algorithms to estimate the weight of the cells covered in the grid representation.

—In establishing integrity constraints that hold over data that has been loaded into a data warehouse, it is necessary to find particular (antecedent, consequent) pairs that have high support in the data [Cormode et al. 2009]. The support of these pairs, as well as the support of just the antecedents, is found using heavy hitter algorithms.

Therefore, it is of high importance to design efficient algorithms for this problem, and understand the performance of existing ones.

Frequency Estimation. The heavy hitters problem can be formalized as one of estimating item frequencies. In this problem we are given a stream of N elements from some universe; the goal is to compute, for each universe element i, an estimator $\hat{f}_i$ that approximates f_i, the number of times the element i occurs in the data stream (or the sum of associated weights in a weighted version). Since the algorithms will be space limited, and will keep detailed information about only a small number of items, for many items we may have $\hat{f}_i = 0$. The goal of such estimators $\hat{f}_i$ is to provide a succinct representation of the data stream, with a controllable trade-off between description size and approximation error.

An algorithm for frequency estimation is characterized by two related parameters: the space[1] and the bounds on the error in estimating the f_is. The error bounds are typically of the "additive" form, namely we have $|f_i - \hat{f}_i| \leq \epsilon B$, for a B (as in "bound") that is a function of the stream. The bound B is equal either to the size of the whole stream - equivalently, to the quantity F_1 where $F_p = \sum_i (f_i)^p$, or to the size of the *residual* tail of the stream, given by $F_1^{res(k)}$, the sum of the frequencies of all elements other than the k most frequent ones (heavy hitters). The residual guarantee is more desirable, since it is always at least as good as the F_1 bound. More strongly, since streams from real applications often obey a very *skewed* frequency distribution, with the heavy hitters constituting the bulk of the stream, a residual guarantee can be *asymptotically* better. In particular, in the extreme case when there are only k distinct elements present in the stream, the residual error bound is zero, i.e. the frequency estimation is exact.

Prior Solutions. Algorithms for the heavy hitters problem have fallen into two main classes: (deterministic) "counter" algorithms and (randomized) "sketch" algorithms. Ta-

[1]We measure space in memory words, each consisting of a logarithmic number of bits.

Table I. Previously known bounds of frequency estimation algorithms.

Algorithm	Type	Space	Error bound
FREQUENT [Demaine et al. 2002; Misra and Gries 1982; Karp et al. 2003]	Counter	$O(1/\epsilon)$	$\|f_i - \hat{f}_i\| \leq \epsilon F_1$
FREQUENT [Bose et al. 2003]	Counter	$O(1/\epsilon)$	$\|f_i - \hat{f}_i\| \leq \epsilon F_1^{res(1)}$
LOSSYCOUNTING [Manku and Motwani 2002]	Counter	$O(\log(\epsilon F_1)/\epsilon)$	$\|f_i - \hat{f}_i\| \leq \epsilon F_1$
SPACESAVING [Metwally et al. 2005]	Counter	$O(1/\epsilon)$	$\|f_i - \hat{f}_i\| \leq \epsilon F_1$
Count-Min [Cormode and Muthukrishnan 2005]	Sketch	$O((k/\epsilon) \log n)$	$\|f_i - \hat{f}_i\| \leq \epsilon/k \cdot F_1^{res(k)}$
Count-Sketch [Charikar et al. 2002]	Sketch	$O((k/\epsilon) \log n)$	$(f_i - \hat{f}_i)^2 \leq \epsilon/k \cdot F_2^{res(k)}$
Presented result	Counter	$O(k/\epsilon)$	$\|f_i - \hat{f}_i\| \leq \epsilon/k \cdot F_1^{res(k)}$

F_1 is the sum of all frequencies; $F_1^{res(k)}$ is the sum of all but the top k frequencies; $F_2^{res(k)}$ is the sum of the squares of all but the top k frequencies; n is the size of the domain from which the stream elements are drawn.

ble I summarizes the space and error bounds of some of the main examples of such algorithms. As is evident from the table, the bounds for the counter and sketching algorithms are incomparable: counter algorithms use less space, but have worse error guarantees than sketching algorithms. The sketching algorithms are able to give so-called "tail guarantees", which depend on $F_1^{res(k)}$, the frequencies of the items in the tail of the frequency distribution, and not on the k largest frequencies in the head of the distribution. These tail guarantees can often be significantly stronger than guarantees that depend on the whole of the frequency distribution, as we explain in the next section.

In practice, however, the *actual* performance of counter-based algorithms has been observed to be appreciably better than of the sketch-based ones, given the same amount of space [Cormode and Hadjieleftheriou 2008]. The reason for this disparity has not previously been well understood or explained. This has led users to apply very conservative bounds in order to provide the desired guarantees; it has also pushed users towards sketch algorithms over counter algorithms since the latter are not perceived to offer the same types of guarantee as the former (for example, [Cormode et al. 2009] and [Ganguly and Lakshminath 2006] adopt Count-Min and Count-Sketches for estimations which require tail guarantees to provide an overall approximation bound).

1.1 Our Contributions.

In this paper we show that the good empirical performance of counter-based algorithms is not an accident: they actually *do* satisfy a much stronger error bound than previously thought. Specifically, in Section 3:

—We identify a general class of *Heavy-Tolerant Counter algorithms* (HTC), that contains the most popular FREQUENT and SPACESAVING algorithms. The class captures the essential properties of the algorithms and abstracts away from the specific mechanics of the procedures.

—We show that any HTC algorithm that has an ϵF_1 error guarantee in fact satisfies the stronger residual guarantee.

We conclude that FREQUENT and SPACESAVING offer the residual bound on error, while using less space than sketching algorithms. Moreover, counter algorithms have small constants of proportionality hidden in their asymptotic cost compared to the much larger logarithmic factors of sketch algorithms, making these space savings very considerable in practice (tight bounds for the two specific algorithms are shown in Section 4). We also establish through a lower bound that the space usage of these algorithms is within a small constant factor of the space required by any counter algorithm that offers the residual bound on error.

The new bounds have several consequences beyond the immediate practical ramifications. First, we show that they provide better bounds for the *sparse recovery* problem, a streaming analog of Compressed Sensing [Donoho 2006; Candès et al. 2006; Gilbert et al. 2007; Rice DSP Group]. This problem is to find the best representation f^* of the frequency distribution, so that f^* has only k non-zero entries. Such a representation captures exact stream statistics for all but k stream elements. In Section 5 we show that using a counter algorithm to produce the k largest estimated frequencies $\hat{f}_i$ yields a good solution to this problem. Formally, let S be the set of the k largest entries in $\hat{f}$, generated by a counter algorithm with $O(k/\epsilon)$ counters. Let f^* be an n-dimensional vector such that f_i^* is equal to $\hat{f}_i$ if $i \in S$ and $f_i^* = 0$ otherwise. Then we show that, under the L_p norm, for any $p \geq 1$, we have

$$\|f - f^*\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}.$$

This is the best known result for this problem in a streaming setting; note that the error is always at least $(F_p^{res(k)})^{1/p}$. The best known sketching algorithms achieve this bound using $\Omega(k \log \frac{n}{k})$ space (see [Berinde et al. 2008; Berinde et al. 2008; Indyk and Ruzic 2008]); in contrast, our approach yields a space bound of $O(k)$. By extracting all m approximated values from a counter algorithm (as opposed to just top k), we are able to show another result. Specifically, by modifying the algorithms to ensure that they always provide an *underestimate* of the frequencies, we show that the resulting reconstruction has L_p error $(1+\epsilon)(\epsilon/k)^{1-1/p} F_1^{res(k)}$ for any $p \geq 1$.

As noted above, many common frequency distributions are naturally skewed. In Section 6 we show that, if the frequencies follow a Zipfian distribution with parameter $\alpha > 1$, then the same tail guarantee follows using only $O(\epsilon^{-1/\alpha})$ space. A set of experiments reported in Section 7 over a mixture of real and synthetic data shows that indeed these bounds are quite tight in practice. Lastly, we also discuss extensions to the cases when streams can include arbitrary weights for each occurrence of an item; and when multiple streams are summarized and need to be merged together into a single summary. We show how the algorithms considered can be generalized to handle both of these situations in Section 8.

1.2 Related Work

There is a large body of algorithms proposed in the literature for heavy hitters problems and their variants; see [Cormode and Hadjieleftheriou 2008] for a survey. Most of them can be classified as either *counter-based* or *sketch-based*. The first counter algorithm is due to Misra and Gries [1982], which we refer to as FREQUENT. Several subsequent works discussed efficient implementation and improved guarantees for this algorithm [Demaine et al. 2002; Bose et al. 2003]. In particular, Bose *et al.* [2003] showed that it offers an $F_1^{res(1)}$ guarantee. Our main result is to improve this to $F_1^{res(k)}$, for a broader class of algorithms.

A second counter algorithm is the LOSSYCOUNTING algorithm of Manku and Motwani [2002]. This has been shown to require $O(1/\epsilon)$ counters over randomly ordered streams to give an ϵF_1 guarantee, but there are adversarial order streams for which it requires $O(1/\epsilon \log \epsilon n)$ [Manku and Motwani 2002]. Our results hold over all possible stream orderings.

The most recent counter solution is the SPACESAVING algorithm due to Metwally et al. [2005]. The algorithm is shown to offer an F_1 guarantee, and also analyzed in the presence of data with Zipfian frequency distribution. Here, we show an $F_1^{res(k)}$ bound, and demonstrate similar bounds for Zipfian data for a larger class of counter algorithms. We provide uniform definitions of both FREQUENT and SPACESAVING in the next Section.

Sketch algorithms are based on linear projections of the frequency vector onto a smaller sketch vector, using compact hash functions to define the projection. Guarantees in terms of $F_1^{res(k)}$ or $F_2^{res(k)}$ follow by arguing that the items with the k largest frequencies are unlikely to (always) collide under the random choice of the hash functions, and so these items can effectively be "removed" from consideration when bounding the estimation error. Because of this random element, sketches are analyzed probabilistically, and have a probability of failure that is bounded by $1/n^c$ for a constant c (n is the size of the domain from which the stream elements are drawn). The Count-Sketch requires $O((k/\epsilon) \log n)$ counters to give guarantees on the sum of squared errors in terms of $F_2^{res(k)}$ [Charikar et al. 2002]; the Count-Min sketch uses $O((k/\epsilon) \log n)$ counters to give guarantees on the absolute error in terms of $F_1^{res(k)}$ [Cormode and Muthukrishnan 2005]. These two guarantees are incomparable in general, varying based on the distribution of frequencies. A key distinction of sketch algorithms is that they allow both positive and negative updates (where negative updates can correspond to deletions, in a transactional setting, or simply arbitrary signal values, in a signal processing environment). This, along with the fact that they are linear transforms, means that they can be used to solve problems such as designing measurements for compressed sensing systems [Gilbert et al. 2007; Candès et al. 2006]. So, although our results show that counter algorithms are strictly preferable to sketches when both are applicable, there are problems that are solved by sketches that cannot be solved using counter algorithms.

We summarize the main properties of these algorithms, along with the corresponding results based on our analysis, in Table I.

This paper represents an extended version of [Berinde et al. 2009]; it differs in the following ways:

—This version has extended introductory material, placing the results in context within the broader area of data stream algorithms, and giving more detailed examples of particular problems which can benefit from the improved guarantees.

—We provide an experimental study on a mixture of real and synthetic data, to show exactly how the tighter bounds are realized in practice (Section 7).

—We present full details of the extensions of counter algorithms to streams with arbitrary non-negative weights.

```
Algorithm 1: FREQUENT($m$)
$T \leftarrow \emptyset$;
foreach $i$ do
    if $i \in T$ then
        $c_i \leftarrow c_i + 1$;
    else if $|T| < m$ then
        $T \leftarrow T \cup \{i\}$;
        $c_i \leftarrow 1$;
    else forall $j \in T$ do
        $c_j \leftarrow c_j - 1$;
        if $c_j = 0$ then
            $T \leftarrow T \backslash \{j\}$;
```

```
Algorithm 2: SPACESAVING($m$)
$T \leftarrow \emptyset$;
foreach $i$ do
    if $i \in T$ then
        $c_i \leftarrow c_i + 1$;
    else if $|T| < m$ then
        $T \leftarrow T \cup \{i\}$;
        $c_i \leftarrow 1$;
    else
        $j \leftarrow \arg\min_{j \in T} c_j$;
        $c_i \leftarrow c_j + 1$;
        $T \leftarrow T \cup \{i\} \backslash \{j\}$;
```

Fig. 1. Pseudocode for FREQUENT and SPACESAVING algorithms

2. DEFINITIONS AND LOWER BOUNDS

We introduce the notation used throughout this paper, define the tail-guarantees more precisely and provide bounds on the best possible results that can be achieved for algorithms which offer such guarantees.

2.1 Counter-based algorithms

We consider a class of "counter-based" algorithms which maintain at most m counters. These m counters correspond to a "frequent" set of elements occurring in the input stream. The input stream contains elements, which we assume without loss of generality to be integers between 1 and n (in fact, the algorithms considered here can accept items drawn from arbitrary universes, such as strings; our restriction to bounded integer identifiers is solely for clarity of notation). We denote a stream of size N by $u_1, u_2, \ldots u_N$. We use $u_{x\ldots y}$ as a shorthand for the partial stream $u_x, u_{x+1}, \ldots, u_y$.

We denote frequencies of elements by an n-dimensional vector f. For ease of notation, we assume without loss of generality that elements are indexed in order of decreasing frequency, so that $f_1 \geq f_2 \geq \ldots \geq f_n$. When the stream is not understood from context, we specify it explicitly, e.g. $f(u_{x\ldots y})$ is the frequency vector for the partial stream $u_{x\ldots y}$. We denote the sum of the frequencies by F_1; we denote the sum of frequencies except the k largest by $F_1^{res(k)}$, and we generalize the definition to sums of the pth power of the frequencies:

$$F_p^{res(k)} = \sum_{i=k+1}^{n} f_i^p, \qquad F_p = F_p^{res(0)}$$

The algorithms considered in this paper can be thought of as adhering to the following form. The state of an algorithm is represented by an n-dimensional vector of counters c. The vector c has at most m non-zero elements. We denote the "frequent" set by $T = \{i \mid c_i \neq 0\}$, since only this set needs to be explicitly stored. The counter value of an element is an approximation for its frequency; the error vector of the approximation is denoted by δ, with $\delta_i = |f_i - c_i|$.

We demonstrate our results with reference to two known counter algorithms: FRE-

QUENT and SPACESAVING. Although similar, the two algorithms differ in the analysis and their behavior in practice. Both maintain their frequent set T, and process a stream of updates. Given a new item i in the stream which is stored in T, both simply increase the corresponding counter c_i; or, if $i \notin T$ and $|T| < m$, then i is stored with a count of 1. The algorithms differ when an unstored item is seen and $|T| = m$: FREQUENT decrements all stored (by definition, non-zero) counters by 1, and (implicitly) throws out any counters with zero count; SPACESAVING finds an item j with smallest non-zero count c_j and assigns $c_i \leftarrow c_j + 1$, followed by $c_j \leftarrow 0$, so in effect i replaces j in T. Pseudocode for these algorithms is presented in Figure 1.

These algorithms are known to provide a "heavy hitter" guarantee on the approximation errors of the counters:

Definition 1. An m-counter algorithm provides a *heavy hitter guarantee with constant* $A > 0$ if, for any stream,

$$\delta_i \leq \left\lfloor A \frac{F_1}{m} \right\rfloor \qquad \forall i$$

More precisely, they both provide this guarantee with constant $A = 1$. Our result is that they also satisfy the following stronger guarantee:

Definition 2. An m-counter algorithm provides a *k-tail guarantee with constants* (A, B)*, with* $A, B > 0$ if for any stream

$$\delta_i \leq \left\lfloor A \frac{F_1^{res(k)}}{m - Bk} \right\rfloor \qquad \forall i$$

Note that the heavy hitter guarantee is equivalent to the 0-tail guarantee. Our general proof (which can be applied to a broad class of algorithms) yields a k-tail guarantee with constants $A = 1$, $B = 2$ for both algorithms (for any $k \leq m/2$). However, by considering particular features of FREQUENT and SPACESAVING, we prove a k-tail guarantee with constants $A = B = 1$ for any $k < m$ following appropriate analysis.

2.2 Lower Bound.

We next demonstrate a lower bound on any deterministic counter-based algorithm that guarantees to provides an error bound of $\frac{F_1^{res(k)}}{m-k}$.

THEOREM 1. *For any deterministic counter algorithm with* m *counters, for any* k*,* $1 \leq k \leq m$*, there exists some stream in which the estimation error of an element is at least* $\frac{F_1^{res(k)}}{2m}$

PROOF. The proof is similar to that of Theorem 2 in [Bose et al. 2003]. For some integer X, consider two streams A and B. The streams share the same prefix of size $X(m + k)$, where elements $a_1 \ldots a_{m+k}$ occur X times each. After the counter algorithm runs on this first part of each stream, only m elements can have non-zero counters. Assume without loss of generality that the other k elements are $a_1 \ldots a_k$.

Then stream A continues with elements $a_1 \ldots a_k$, while stream B continues with k other elements $z_1 \ldots z_k$ distinct from $a_1 \ldots a_{m+k}$. Both streams thus have total size $X(m+k)+k$.

For both streams, after processing the prefix of size $X(m + k)$, the algorithm has no record of any of the elements in the remaining parts of either of the streams. So the two remaining parts look identical to the algorithm and will yield the same estimates. Thus, for $1 \leq i \leq k$, $c_{a_i}(A) = c_{z_i}(B)$. But $f_{a_i}(A) = X + 1$ while $f_{z_i}(B) = 1$. The counter error for one of the two streams must be at least $X/2$. Note that $F_1^{res(k)}(A) = Xm$ and $F_1^{res(k)}(B) = Xm + k$; then the error is at least

$$\frac{X}{2} \geq \frac{F_1^{res(k)}}{2m + 2k/X}$$

As $X \to \infty$, this approaches our desired bound. □

Thus an algorithm that provides an error bound of $\frac{F_1^{res(k)}}{m-k}$ must use at least $(m-k)/2$ counters. Thus the number of counters FREQUENT and SPACESAVING use is within a small factor (3 for $k \leq m/3$) of the best possible. As such, our subsequent results show that these algorithms are asymptotically *optimal* for this problem.

3. RESIDUAL ERROR BOUND

In this section we state and prove our main result on the error bound for a general class of heavy-tolerant counter algorithms. We begin by formally defining this class.

Definition 3. A value i is *x-prefix guaranteed* for the stream $u_{1\ldots s}$ if after the first $x < s$ elements of the stream have been processed, i will stay in T even if some elements are removed from the remaining stream (including occurrences of i). Formally, the value i is *x-prefix guaranteed* if $0 \leq x < s$ and $c_i(u_{1\ldots x}v_{1\ldots t}) > 0$ for all subsequences $v_{1\ldots t}$ of $u_{(x+1)\ldots s}$, $0 \leq t \leq s - x$.

Note that if i is x-prefix guaranteed, then i is also y-prefix guaranteed for all $y > x$.

Definition 4. A counter algorithm is *heavy-tolerant* if extra occurrences of guaranteed elements do not increase the estimation error. Formally, an algorithm is *heavy-tolerant* if for any stream $u_{1\ldots s}$, given any x, $1 \leq x < s$, for which element $i = u_x$ is $(x-1)$-prefix guaranteed, it holds that

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots(x-1)}u_{(x+1)\ldots s}) \qquad \forall j$$

3.1 Proof of Heavy Tolerance

First, we demonstrate that heavy tolerance is a reasonable property by proving that both the two example counter algorithms obey it. Intuitively, this property should hold because occurrences of an element already in the frequent set only affect the counter value of that element; and, as long as the element never leaves the frequent set, the value of its counter does not affect the algorithm's other choices.

THEOREM 2. *Algorithms* FREQUENT *and* SPACESAVING *are heavy-tolerant.*

PROOF. Denote $v_{1\ldots t} = u_{(x+1)\ldots(x+t)}$, with $t \leq s - x$. We prove by induction on t that for both algorithms

$$c(u_{1\ldots x}v_{1\ldots t}) = c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i$$

where $i = u_x$ and e_i is the i-th row of I_n, the $n \times n$ identity matrix; this implies that

$$\delta(u_{1\ldots x}v_{1\ldots t}) = \delta(u_{1\ldots(x-1)}v_{1\ldots t})$$

Base case at $t = 0$: By the hypothesis: $c_i(u_{1\ldots(x-1)}) \neq 0$, hence when element $u_x = i$ arrives after processing $u_{1\ldots x}$, both FREQUENT and SPACESAVING just increase i's counter:

$$c(u_{1\ldots x}) = c(u_{1\ldots(x-1)}) + e_i$$

Induction step for $t > 0$: We are given that

$$c(u_{1\ldots x}v_{1\ldots(t-1)}) = c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_i$$

Note that since i is $(x-1)$-prefix guaranteed, these vectors have the same support.
Case 1: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) > 0$. Hence
$c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) > 0$. For both streams, v_t's counter just gets incremented and thus

$$\begin{aligned} c(u_{1\ldots x}v_{1\ldots t}) &= c(u_{1\ldots x}v_{1\ldots(t-1)}) + e_{v_t} \\ &= c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_{v_t} + e_i \\ &= c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i \end{aligned}$$

Case 2: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) = 0$. Note that $v_t \neq i$ since i is x-prefix guaranteed and $c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) = 0$. By the induction hypothesis, both counter vectors have the same support (set of non-zero entries). If the support is less than m, then the algorithm adds e_{v_k} to the counters, and the analysis follows Case 1 above. Otherwise, the two algorithms differ:

—FREQUENT algorithm: In this case all non-zero counters will be decremented. Since both counter vectors have the same support, they will be decremented by the same m-sparse binary vector $\gamma = \chi(T) = \sum_{j:c_j\neq 0} e_j$.

—SPACESAVING algorithm: The minimum non-zero counter is set to zero. To avoid ambiguity, we specify that SPACESAVING will pick the counter c_j with the smallest identifier j if there are multiple counters with equal smallest non-zero value. Let

$$j = \operatorname*{argmin}_{j \in T(u_{1\ldots x}v_{1\ldots(t-1)})} c_j(u_{1\ldots x}v_{1\ldots(t-1)})$$

and

$$j' = \operatorname*{argmin}_{j' \in T(u_{1\ldots(x-1)}v_{1\ldots(t-1)})} c_{j'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)})$$

Since i is x-prefix guaranteed, its counter can never become zero, hence $j \neq i, j' \neq i$. Since

$$c_{i'}(u_{1\ldots x}v_{1\ldots(t-1)}) = c_{i'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)})$$

for all $i' \neq i$, it follows that $j = j'$ and

$$c_j(u_{1\ldots x}v_{1\ldots(t-1)}) = c_{j'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) = M.$$

Hence both streams result in updating the counters by subtracting the same difference vector $\gamma = Me_j - (M+1)e_{v_t}$

Thus each algorithm computes the same difference vector γ irrespective of which stream it is applied to, and updates the counters:

$$\begin{aligned} c(u_{1\ldots x}v_{1\ldots t}) &= c(u_{1\ldots x}v_{1\ldots(t-1)}) - \gamma \\ &= c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_i - \gamma \end{aligned}$$

$$= c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i$$

□

3.2 Proof of k-tail guarantee

Next, we show that any algorithm which is heavy-tolerant must therefore provide a k-tail guarantee. We first define an operation to reduce a stream to a simpler one, and build a sequence of lemmas which bound the error on a complex stream with that on a simpler one. Let $\texttt{Remove}(u_{1\ldots s}, i)$ be the subsequence of $u_{1\ldots s}$ with all occurrences of value i removed, i.e.

$$\texttt{Remove}(u_{1\ldots s}, i) = \begin{cases} \text{empty sequence} & \text{if } s = 0 \\ (u_1, \texttt{Remove}(u_{2\ldots s}, i)) & \text{if } u_1 \neq i \\ \texttt{Remove}(u_{2\ldots s}, i) & \text{if } u_1 = i \end{cases}$$

LEMMA 3. *If i is x-prefix guaranteed and the algorithm is heavy-tolerant, then*

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x}v_{1\ldots t}) \qquad \forall j$$

where $v_{1\ldots t} = \texttt{Remove}(u_{(x+1)\ldots s}, i)$, with $0 \leq t \leq s - x$.

PROOF. Let $x_1, x_2, \ldots, x_q$ be the positions of occurrences of i in $u_{(x+1)\ldots s}$, with $x < x_1 < x_2 < \ldots < x_q$. We apply the heavy-tolerant definition for each occurrence; for all j:

$$\begin{aligned} \delta_j(u_{1\ldots s}) &\leq \delta_j(u_{1\ldots(x_1-1)}u_{(x_1+1)\ldots s}) \\ &\leq \delta_j(u_{1\ldots(x_1-1)}u_{(x_1+1)\ldots(x_2-1)}u_{(x_2+1)\ldots s}) \\ &\leq \ldots \\ &\leq \delta_j(u_{1\ldots x}v_{1\ldots t}) \end{aligned}$$

Note in particular that $\delta_i(u_{1\ldots p})$, the error in estimating the frequency of i in the original stream, is identical to $\delta_i(u_{1\ldots x}v_{1\ldots q})$, the error of i on the derived stream, since i is x-prefix guaranteed. □

Definition 5. An *error bound* for an algorithm is a function $\Delta : \mathbb{N}^n \to \mathbb{R}^+$ such that for any stream $u_{1\ldots s}$

$$\delta_i(u_{1\ldots s}) \leq \lfloor \Delta(f(u_{1\ldots s})) \rfloor \qquad \forall i$$

In addition, we require that Δ, the absolute error, must be "increasing" in the sense that for any two frequency vectors f' and f'' such that $f'_i \leq f''_i$ for all i, it holds that $\Delta(f') \leq \Delta(f'')$.

LEMMA 4. *Let Δ be an error bound for a heavy-tolerant algorithm that provides a heavy hitter guarantee with constant A. Then the following function is also an error bound for the algorithm, for any k, $1 \leq k < m/A$:*

$$\Delta'(f) = A\frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

PROOF. Let $u_{1\ldots s}$ be any stream. Let $D = 1 + \lfloor \Delta(f(u_{1\ldots s})) \rfloor$. We assume without loss of generality that the elements are indexed in order of increasing frequency.

Let $k' = \max \{i \mid 1 \leq i \leq k \text{ and } f_i(u_{1\ldots s}) > D\}$. Recall that by convention we index the elements by decreasing frequencies ($f_1 \geq f_2 \geq \ldots$). If the there is no element i such that $f_i(u_{1\ldots s}) > D$ the argument below follows trivially (e.g. set $k' = 0$).

For each $i \leq k'$ let x_i be the position of the D-th occurrence of i in the stream. We claim that any $i \leq k'$ is x_i-prefix guaranteed: let $v_{1\ldots t}$ be any subsequence of $u_{(x_i+1)\ldots s}$; it holds for all j that

$$\delta_j(u_{1\ldots x_i} v_{1\ldots t}) \leq \lfloor \Delta(f(u_{1\ldots x_i} v_{1\ldots t})) \rfloor < D$$

$$\begin{aligned} \text{and so } c_j(u_{1\ldots x_i} v_{1\ldots t}) &\geq f_j(u_{1\ldots x_i} v_{1\ldots t}) - \delta_j(u_{1\ldots x_i} v_{1\ldots t}) \\ &> D - D = 0. \end{aligned}$$

Let $i_1, i_2, \ldots i_{k'}$ be the permutation of $1 \ldots k'$ so that $x_{i_1} > x_{i_2} > \ldots > x_{i_{k'}}$. We can apply Lemma 3 for i_1 which is x_{i_1}-prefix guaranteed; for all j

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x_{i_1}} v_{1\ldots s_v})$$

where $v_{1\ldots s_v} = \texttt{Remove}(u_{(x_{i_1}+1)\ldots s}, i_1)$.

Since $x_{i_2} < x_{i_1}$, i_2 is x_{i_2}-prefix guaranteed for the new stream $u_{1\ldots x_{i_1}} v_{1\ldots s_v}$ and we apply Lemma 3 again:

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x_{i_1}} v_{1\ldots s_v}) \leq \delta_j(u_{1\ldots x_{i_2}} w_{1\ldots s_w}) \qquad \forall j$$

where $w_{1\ldots s_w} = \texttt{Remove}(u_{(x_{i_2}+1)\ldots x_{i_1}} v_{1\ldots s_v}, i_2)$. Since the x_{i_j} values are decreasing, we can continue this argument for $i = 3, 4, \ldots, k'$. We obtain the following inequality for the final stream $z_{1\ldots s_z}$

$$\delta_j(u_{1\ldots s}) \leq \delta_j(z_{1\ldots s_z}) \qquad \forall j$$

where $z_{1\ldots s_z}$ is the stream $u_{1\ldots s}$ with all "extra" occurrences of elements 1 to k' removed ("extra" means after the first D occurrences). Thus

$$\|f(z_{1\ldots s_z})\|_1 = k'D + \sum_{i=k'+1}^{n} f_i(u_{1\ldots s})$$

Either $k' = k$, or $k' < k$ and $f_i(u_{1\ldots s}) \leq D$ for all $k' < i \leq k$; in both cases we can replace k' with k:

$$\|f(z_{1\ldots s_z})\|_1 \leq kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})$$

We now apply the heavy hitter guarantee for this stream; for all j:

$$\begin{aligned} \delta_j(u_{1\ldots s}) &\leq \delta_j(z_{1\ldots s_z}) \\ &\leq \left\lfloor A \frac{kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})}{m} \right\rfloor \\ &\leq \left\lfloor A \frac{k\Delta(u_{1\ldots s}) + k + F_1^{res(k)}}{m} \right\rfloor \end{aligned}$$

□

We can now prove our main Theorem on the accuracy of counter-based algorithms.

THEOREM 5. *If a heavy-tolerant algorithm provides a heavy hitter guarantee with constant A, it also provides a k-tail guarantee with constants $(A, 2A)$, for any k, $1 \leq k < m/2A$.*

PROOF. We start with the initial error bound given by the heavy hitter guarantee $\Delta(f) = A\frac{\|f\|_1}{m}$ and apply Lemma 4 to obtain another error bound Δ'. We can continue iteratively applying Lemma 4 in this way. Either we will eventually obtain a new bound which is worse than the previous one, in which case this process halts with the previous error bound; or else we can analyze the error bound obtained in the limit (in the spirit of [Bose et al. 2003]). In both cases, the following holds for the best error bound Δ:

$$\Delta(f) \leq A\frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

$$\text{and so } \Delta(f) \leq A\frac{k + F_1^{res(k)}}{m - Ak} \ .$$

We have shown that for any stream $u_{1\ldots p}$,

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A\frac{k + F_1^{res(k)}}{m - Ak} \right\rfloor \qquad \forall i$$

We show that this implies the guarantee

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A\frac{F_1^{res(k)}}{m - 2Ak} \right\rfloor \qquad \forall i$$

Case 1: $AF_1^{res(k)} < m - 2Ak$. In this case both guarantees are identical: all errors are 0.
Case 2: $AF_1^{res(k)} \geq m - 2Ak$:

$$\begin{aligned} A^2kF_1^{res(k)} &\geq Ak(m - 2Ak) \\ A(m - Ak)F_1^{res(k)} &\geq A(m - 2Ak)\left(k + F_1^{res(k)}\right) \\ A\frac{F_1^{res(k)}}{m - 2Ak} &\geq A\frac{k + F_1^{res(k)}}{m - Ak} \end{aligned}$$

□

4. SPECIFIC PROOFS

Given our new understanding of bounds which apply to all counter-based algorithms, we are now able to give tighter bounds which are specific to some of the most popular examples of the class. That is, we provide tighter k-tail guarantee proofs tailored for the FREQUENT and SPACESAVING algorithms. Specifically, we show that these algorithms, when executed with m counters, recover all element frequencies with an error of at most $F_1^{res(k)}/(m - k)$, for any $k < m$. This corresponds to the tail guarantee with $A = B = 1$.

4.1 Tail guarantee with constants $A = B = 1$ for FREQUENT

We can interpret the FREQUENT algorithm in the following way: each element in the stream results in incrementing one counter; in addition, some number of elements (call this number d) also result in decrementing $m + 1$ counters (we can think of the d elements incrementing and later decrementing their own counter). The sum of the counters at the end of the algorithm is $\|c\|_1$. We have

$$\|c\|_1 = \|f\|_1 - d(m + 1)$$

Since there were d decrement operations, and each operation decreases any given counter by at most one, it holds that the final counter value for any element is at least $f_i - d$. We restrict our attention to the k most frequent elements. Then

$$\begin{aligned}
\|c\|_1 = \|f\|_1 - d(m+1) &\geq \sum_{i=1}^{k}(f_i - d) \\
\|f\|_1 - d(m+1) &\geq -dk + \sum_{i=1}^{k} f_i \\
\sum_{i=k+1}^{n} f_i &\geq d(m+1-k) \\
d &\leq \frac{F_1^{res(k)}}{m+1-k}
\end{aligned}$$

Since the error in any counter is at most d, this implies the k-tail guarantee with $A = B = 1$.

4.2 Tail guarantee with constants $A = B = 1$ for SPACESAVING

The tail guarantee follows as a result of the following claims proven in [Metwally et al. 2005]:

LEMMA 3 IN [METWALLY ET AL. 2005]: *If the minimum non-zero counter value is Δ, then $\delta_i \leq \Delta$ for all i.*

THEOREM 2 IN [METWALLY ET AL. 2005]: *Whether or not element i (i.e. i-th most frequent element) corresponds to the i-th largest counter, the value of this counter is at least f_i, the frequency of i.*

If we restrict our attention to the k largest counters, the sum of their values is at least $\sum_{i=1}^{k} f_i$. Since in this algorithm the sum of the counters is always equal to the length of the stream, it follows that:

$$\Delta \leq \frac{\|f\|_1 - \sum_{i=1}^{k} f_i}{m-k}$$

thus by Lemma 3

$$\delta_i \leq \frac{F_1^{res(k)}}{m-k} \qquad \forall i$$

which is the k-tail guarantee with constants $A = B = 1$.

5. SPARSE RECOVERIES

The k-sparse recovery problem is to find a representation f' so that f' has only k non-zero entries ("k-sparse"), and the L_p norm $\|f - f'\|_p = (\sum_{i=1}^{n} |f_i - f_i'|^p)^{1/p}$ is minimized. A natural approach is to build f' from the heavy hitters of f, and indeed we show that this method gives strong guarantees for frequencies from heavy tolerant counter algorithms.

5.1 k-sparse recovery

To get a k-sparse recovery, we run counter algorithm that provides a k-tail guarantee with m counters and create f' using the k largest counters. These are not necessarily the k most frequent elements (with indices 1 to k in our notation), but we show that they must be "close enough".

THEOREM 6. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $m = k(\frac{3A}{\varepsilon} + B)$ counters and retain the top k counter values into the k-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}$$

PROOF. Let $K = \{1, \ldots, k\}$ be the set of the k most frequent elements. Let S be the set of elements with the k largest counters. Let $R = \{1, \ldots, n\} \setminus (S \cup K)$ be the set of all other remaining elements. Let $k' = |K \setminus S| = |S \setminus K|$.

Let $x_1 \ldots x_{k'}$ be the k' elements in $S \setminus K$, with $c_{x_1} \geq c_{x_2} \geq \ldots \geq c_{x_{k'}}$. Let $y_1 \ldots y_{k'}$ be the k' elements in $K \setminus S$, with $c_{y_1} \geq c_{y_2} \geq \ldots \geq c_{y_{k'}}$. Notice that $c_{x_i} \geq c_{y_i}$ for any i: c_{y_i} is the i^{th} largest counter in $K \setminus S$, whereas c_{x_i} is the i^{th} largest counter in $(K \cup S) \setminus (S \cap K)$, a superset of $K \setminus S$. Let Δ be an upper bound on the counter errors δ. Then for any i

$$f_{y_i} - \Delta \leq c_{y_i} \leq c_{x_i} \leq f_{x_i} + \Delta \tag{1}$$

Hence $f_{y_i} \leq f_{x_i} + 2\Delta$. Let f' be the recovered frequency vector ($f'_{x_i} = c_{x_i}$ and zero everywhere else). For any $p \geq 1$, and using the triangle inequality $\|a + b\|_p \leq \|a\|_p + \|b\|_p$ on the vector f_i restricted to $i \in R \cup S$ and the vector equal to the constant 2Δ restricted to $i \in S \setminus K$:

$$\begin{aligned}
\|f - f'\|_p &= \left(\sum_{i \in S} (c_i - f_i)^p + \sum_{i \in R \cup K \setminus S} (f_i)^p \right)^{1/p} \\
&\leq \left(\sum_{i=1}^{k} \Delta^p + \sum_{i \in K \setminus S} (f_i)^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{y_i})^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{x_i} + 2\Delta)^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq 3k^{1/p}\Delta + \left(\sum_{i \in R \cup S \setminus K} (f_i)^p \right)^{1/p} \\
&\leq 3k^{1/p}\Delta + (F_p^{res(k)})^{1/p}
\end{aligned}$$

If an algorithm has the tail guarantee with constants (A, B), by using $m = k(\frac{3A}{\varepsilon} + B)$ counters we get

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p} \tag{2}$$

□

Note that $(F_p^{res(k)})^{1/p}$ is the smallest possible L_p error of any k-sparse recovery of f. Also, if the algorithm provides one-sided error on the estimated frequencies (as is the case for FREQUENT and SPACESAVING), it is sufficient to use $m = k(\frac{2A}{\varepsilon} + B)$ counters, since now $f_{y_i} \leq f_{x_i} + \Delta$.

5.2 m-sparse recovery

When the counter algorithm uses m counters, it stores approximate values for m elements. It seems intuitive that by using all m of these counter values, the recovery should be even better. This turns out not to be true in general. Instead, we show that it is possible to derive a better result given an algorithm which always *underestimates* the frequencies ($c_i \leq f_i$). For example, this is true in the case of FREQUENT.

As described so far, SPACESAVING always overestimates, but can be modified to underestimate the frequencies. In particular, the algorithm has the property that error is bounded by the smallest counter value, i.e. $\Delta = \min\{c_j | c_j \neq 0\}$. So setting $c'_i = \max\{0, c_i - \Delta\}$ ensures that $c'_i \leq f_i$. Because $f_i + \Delta \geq c_i \geq f_i$, $f_i - c'_i \leq \Delta$ and thus c' satisfies the same k-tail bounds with $A = B = 1$ (as per Section 4.2). Note that in practice, slightly improved per-item guarantees follow by storing ϵ_i for each non-zero counter c_i as the value of Δ when i last entered the frequent set, and using $c_i - \epsilon_i$ as the estimated value (as described in [Metwally et al. 2005]).

THEOREM 7. *If we run an underestimating counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the counter values into the m-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq (1+\varepsilon)\left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}$$

PROOF. Set $m = k(\frac{A}{\varepsilon} + B)$ in Definition 2 to obtain

$$\begin{aligned}
\|f - f'\|_p &= \left(\sum_{i=1}^{k}(f_i - c_i)^p + \sum_{i=k+1}^{n}(f_i - c_i)^p\right)^{1/p} \\
&\leq \left(k\frac{\varepsilon^p}{k^p}(F_1^{res(k)})^p + \sum_{i=k+1}^{n}(f_i - c_i)\frac{\varepsilon^{p-1}}{k^{p-1}}(F_1^{res(k)})^{p-1}\right)^{1/p} \\
&\leq \left(\frac{\varepsilon^p}{k^{p-1}}(F_1^{res(k)})^p + \frac{\varepsilon^{p-1}}{k^{p-1}}(F_1^{res(k)})^p\right)^{1/p} \\
&\leq (1+\varepsilon)\left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}
\end{aligned}$$

□

5.3 Estimating $F_1^{res(k)}$

Since our algorithms give guarantees in terms of $F_1^{res(k)}$, a natural related problem is to estimate the value of this quantity.

THEOREM 8. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the largest k counter values as the*

k-sparse vector f', then:

$$F_1^{res(k)}(1-\varepsilon) \leq F_1 - \|f'\|_1 \leq F_1^{res(k)}(1+\varepsilon)$$

PROOF. To show this result, we rely on the definitions and properties of sets S and K from the proof of Theorem 6. By construction of sets S and K, $f_{x_i} \leq f_{y_i}$ for any i. Using equation (1) it follows that

$$f_{y_i} - \Delta \leq c_{x_i} \leq f_{y_i} + \Delta$$

So the norm of f' must be close to the norm of the best k-sparse representative of f, i.e. $(F_1 - F_1^{res(k)})$. Summing over each of the k counters yields

$$\begin{aligned} F_1 - F_1^{res(k)} - k\Delta \leq \quad \|f'\|_1 \quad &\leq F_1 - F_1^{res(k)} + k\Delta \\ F_1^{res(k)} - k\Delta \leq\ F_1 - \|f'\|_1\ &\leq F_1^{res(k)} + k\Delta \end{aligned}$$

The result follows when setting $m = k(\frac{Ak}{\varepsilon} + B)$ so the upper bound ensures $\Delta \leq \frac{\varepsilon}{k} F_1^{res(k)}$. □

6. ZIPFIAN DISTRIBUTIONS

Realistic data can often be approximated with a Zipfian distribution. Prior work has argued that such distributions fit a wide variety of data, including sizes of cities and word frequencies in text [Zipf 1949]; citations of papers [Redner 1998]; web page access frequencies [Breslau et al. 1999]. and file transfer size and duration [Bestavros et al. 1999]. A stream of length $F_1 = N$, with n distinct elements, distributed (exactly) according to the Zipfian distribution with parameter α has frequencies

$$f_i = N\frac{1}{i^\alpha \zeta(\alpha)} \quad \text{where} \quad \zeta(\alpha) = \sum_{i=1}^{n} \frac{1}{i^\alpha}$$

The value $\zeta(\alpha)$ converges to a small constant when $\alpha > 1$. Although data rarely obeys this distribution exactly, our first result requires only that the "tail" of the distribution can be bounded by a (small constant multiple of) a Zipfian distribution. Note that this requires that the total frequencies follow this distribution, but the order of items in the stream can be arbitrary, that is, it is not necessary for the each item to be drawn i.i.d. from the frequency distribution. We first show that to obtain the weaker heavy hitter guarantee over Zipfian distributions, considerably less space is needed.

THEOREM 9. *Given Zipfian data with parameter $\alpha \geq 1$, if a counter algorithm that provides a k-tail guarantee with constants (A, B) for $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ is used with $m = (A+B)\left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ counters, the counter errors are at most εF_1.*

PROOF. The k-tail guarantee with constants (A, B) means

$$\Delta = A\frac{F_1^{res(k)}}{m - Bk} \leq A\frac{N}{\zeta(\alpha)} \frac{\sum_{i=k+1}^{n} i^{-\alpha}}{m - Bk}$$

Then

$$\sum_{i=k+1}^{n} \frac{1}{i^\alpha} \leq \int_k^n \frac{1}{x^\alpha} dx = \frac{1}{k^{\alpha-1}} \int_1^{n/k} \frac{1}{x^\alpha} dx \leq \frac{\zeta(\alpha)}{k^{\alpha-1}}$$

$$\Delta \leq A \frac{\zeta(\alpha)}{k^{\alpha-1}} \frac{N}{\zeta(\alpha)(m - Bk)} = \frac{N}{k^\alpha} A \frac{k}{m - Bk}$$

by setting $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$, $m = (A + B)k$,

$$\Delta \leq \frac{N}{k^\alpha} = \varepsilon N$$

□

A similar result is proved for SPACESAVING in [Metwally et al. 2005] under the stronger assumption that the frequencies are exactly as defined by the Zipfian distribution.

6.1 Top-k

In this section we analyze the algorithms in the context of the problem of finding top k elements, when the input is Zipf distributed.

THEOREM 10. *Assuming Zipfian data with parameter $\alpha > 1$, a counter algorithm that provides a k'-tail guarantee for $k' = \Theta\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ can retrieve the top k elements in correct order using $O\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ counters. For Zipfian data with parameter $\alpha = 1$, an algorithm with k'-tail guarantee for $k' = \Theta(k^2 \ln n)$ can retrieve the top k elements in correct order using $O(k^2 \ln n)$ counters.*

PROOF. To get the top k elements in the correct order we need

$$\Delta < \frac{f_k - f_{k+1}}{2}.$$

If we assume that the data follows a Zipfian distribution, then this difference is given by:

$$\begin{aligned} f_k - f_{k+1} &= \frac{N}{\zeta(\alpha)} \left(\frac{1}{k^\alpha} - \frac{1}{(k+1)^\alpha} \right) \\ &= \frac{N}{\zeta(\alpha)} \frac{(k+1)^\alpha - k^\alpha}{(k+1)^\alpha k^\alpha} \\ &< \frac{N}{\zeta(\alpha)} \frac{\alpha k^{\alpha-1}}{(k+1)^\alpha k^\alpha} = \frac{N}{\zeta(\alpha)} \frac{\alpha}{(k+1)^\alpha k} \end{aligned}$$

Thus we need error rate

$$\varepsilon = \frac{\alpha}{2\zeta(\alpha)(k+1)^\alpha k} = \begin{cases} \Theta(\alpha/k^{1+\alpha}) & \text{for } \alpha > 1 \\ \Theta(1/(k^2 \ln n)) & \text{for } \alpha = 1 \end{cases}$$

The result then follows from Theorem 9. □

7. EXPERIMENTAL EVALUATION

We have shown that FREQUENT and SPACESAVING provide tail guarantees with $A = B = 1$. In this section we present experiments which show that (i) these bounds hold in practice; (ii) the tail guarantee is useful in practical datasets, in that it can provide significantly tighter bounds that the usual heavy-hitter guarantee; and (iii) the theoretical bounds are in practice very close to the observed errors, confirming that these bounds cannot be improved further, and that the constants derived are quite tight.

The method of testing was as follows. Each dataset consists of a stream of integers, each representing the occurrence of a single element. An exact algorithm which uses large amounts of memory was used to compute the exact frequencies of all elements. Our implementations of the algorithms FREQUENT, SPACESAVING, as well as the underestimating variant of SPACESAVING (described in section 5.2) were performed on the stream. We tried a variety of choices for the number of counters m, corresponding to $1, 2$ and 5 times 10, 100, 1000 and 10,000 – this covers five orders of magnitude, to demonstrate the effect over a wide range of parameter values. For each run, we record the maximum error $\|c - f\|_\infty$, i.e. the maximum difference between a recovered frequency and the corresponding exact frequency. The recovered frequency of an element is given by the value of the counter assigned to that element, or is set to 0 if the element is not in the frequent set maintained by the algorithm. In addition, for the two algorithms that never overestimate the element frequencies and estimate zero for items not explicitly stored (FREQUENT and the modified version of SPACESAVING), we also record the square root of the sum of the squares of all errors (for all items); this is the L_2 recovery error $\|c - f\|_2$ of the m-sparse approximation c induced by the counters, as discussed in Section 5.2.

Our experiments concentrate on understanding the accuracy of these algorithms with a given space budget in the light of tail-guarantees. We do not study their time cost or other parameters; for this comparison, see other experimental studies such as [Cormode and Hadjieleftheriou 2008; Manerikar and Palpanas 2009].

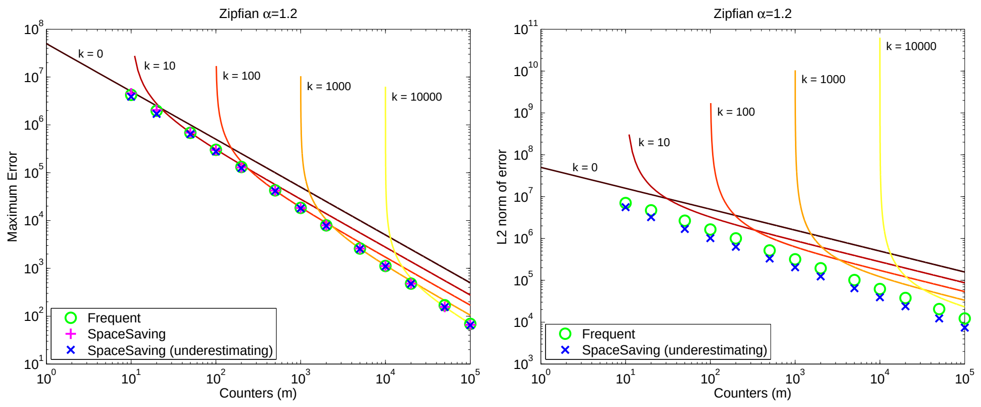
7.1 Synthetic datasets

For the synthetic datasets, we generated random streams according to the Zipfian distribution. Such distributions are commonly used to test the behavior of streaming algorithms. Each generated stream has a length of 50 million items, large enough to dominate the size of the summaries (ranging from about a kilobyte for $m = 100$ to about a megabyte for $m = 10^4$. We used the following values for the skew parameter: $\alpha = 1.2$ (light to moderate skew), $\alpha = 1.5$, and $\alpha = 2$ (quite skewed). Note that experiments on real data sets such as network traffic sizes, degree distributions of power law graphs and so on, have fitted Zipfian distributions with parameters in the range 1 to 2.2, consistent with our experimental parameters. The results are shown in Figure 2.

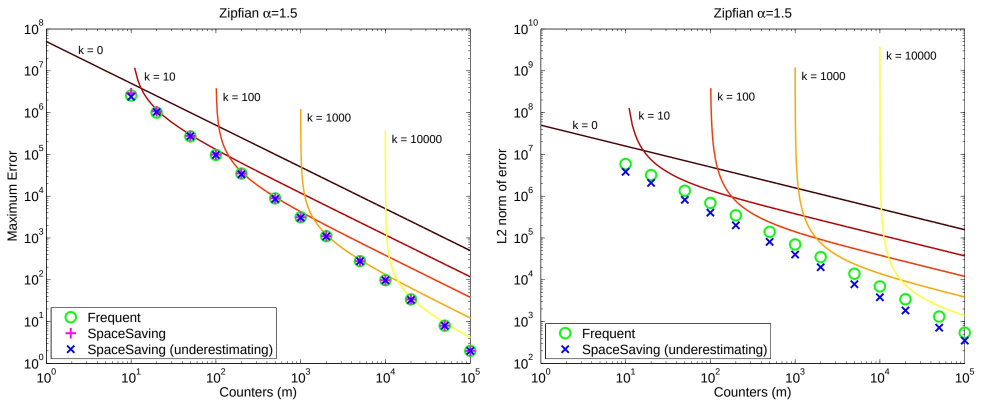
The left-hand plots shows how the observed errors as m was varied. For a given choice of m, our prior analysis argues that the error should be bounded by all k-tail guarantees with $k < m$. For a given k, the k-tail bound can be represented as the curve $y(m) = F_1^{res(k)}/(m - k)$; all data points with $m > k$ must be under this curve for the bound to be satisfied. We plot these (data-dependent) curves for a few choices of k showing how each choice provides the tightest bound over a range of m values.

The right-hand plots show the L_2 error of the obtained m-sparse approximation as m was varied. Theorem 7 establishes that if $m = k + k/\epsilon$ then the L_2 norm of the error vector $\|c - f\|_2$ is bounded by $F_1^{res(k)}(1+\epsilon)\sqrt{\frac{\epsilon}{k}}$. As before, for a given choice of m, the bound must hold for all $k < m$. For a given k, we represent this m-sparse recovery bound as the curve $y(m) = F_1^{res(k)}\left(1 + \frac{k}{m-k}\right)\sqrt{\frac{1}{m-k}}$, obtained by substituting $\epsilon = k/(m-k)$ in the bound above.

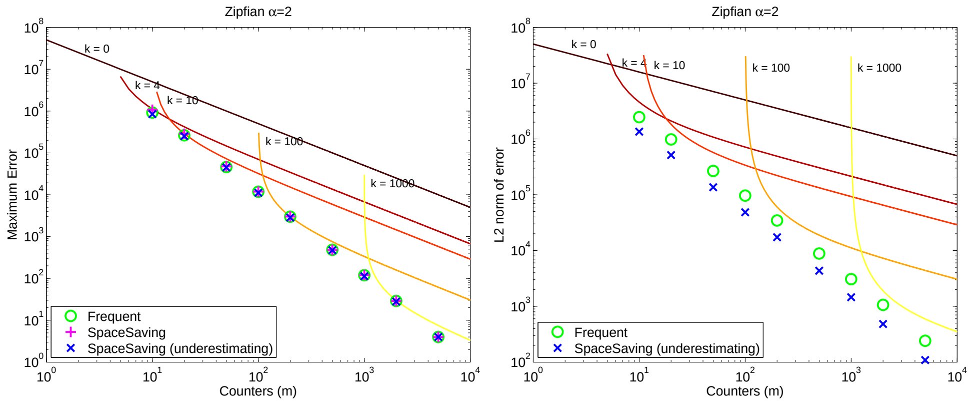
The plots in Figure 2 confirm the correctness of our analysis: it is indeed the case that the worst case error of both algorithms is never greater than the bound given by the curves. In some cases, the error is less, but not by a great deal and not consistently so, suggesting



(a) Zipfian Distribution with $\alpha = 1.2$



(b) Zipfian Distribution with $\alpha = 1.5$



(c) Zipfian Distribution with $\alpha = 2.0$

Fig. 2. Experimental results on synthetic Zipfian data. The curves in the left-hand plots show k-tail guarantees; the curves in the right-hand plots show m-sparse recovery L_2 error bounds established by theorem 7.

that over all possible streams these bounds are quite accurate. The maximum error plots also demonstrate the power of a tail bound guarantee compared to the simple heavy hitter (F_1) guarantee. Consider the left plot in Figure 2(b). The straight $k = 0$ line indicates the worst case F_1 guarantee as a function of m. The observed points are significantly below this on the log-scale plot, and the increase in accuracy grows as m increases. For $m = 10^5$, the benefit is over two orders of magnitude: in other words, the accuracy is over 100 times better than the simple F_1/m guarantee would suggest. This error continues to decrease towards zero; it reaches zero only when k is large enough to record the frequencies of all items with non-zero counts. This effect on the maximum error becomes more pronounced with greater skew (Figure 2(c)), and diminished with lesser skew (Figure 2(a)).

For the right-hand plots in Figure 2, we see a similar effect for the L_2 error of the m-sparse approximations. However, there is a consistent separation between the two algorithms: the SPACESAVING algorithm achieves uniformly better results, by factors approaching 2. Again, the theoretical bounds are respected, and seem somewhat tight.

7.2 Real datasets

We performed similar experiments using several real datasets; some have been widely used in similar experiments (see, for example, [Manerikar and Palpanas 2009]).

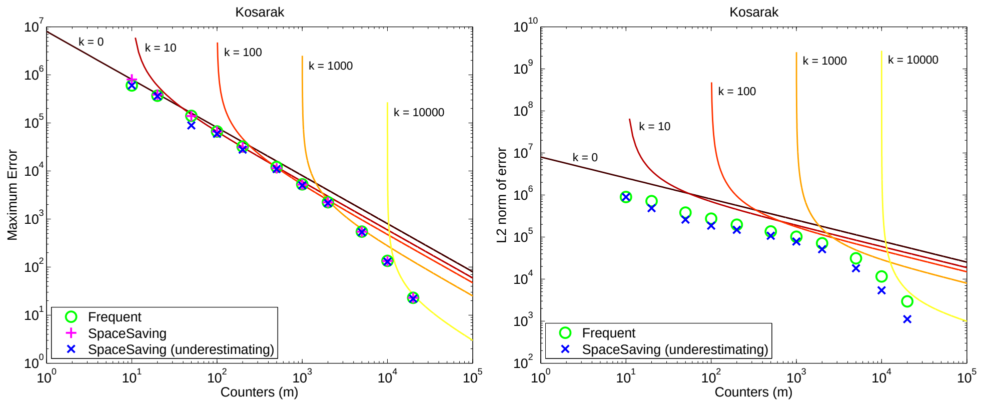
—The first dataset, `Kosarak`, contains (anonymized) click-stream dataset of a Hungarian on-line news portal [FIMI Repository 2008]. The dataset consists of multi-item transactions; we consider every single item in serial order (as in [Manerikar and Palpanas 2009]). The stream contains over 8M items, drawn from a domain of size 41K. The results are shown in Figure 3(a).

—The `Q148` dataset is also identical to one used in [Manerikar and Palpanas 2009]. It was derived from the KDD Cup 2000 data [Kohavi et al. 2000], provided by Blue Martini. The stream consists of the attribute number 148 (“Request Processing Time Sum”) from the `clicks` dataset. The stream contains 234954 items drawn from a domain of size approximately 12K. The results are shown in Figure 3(b).

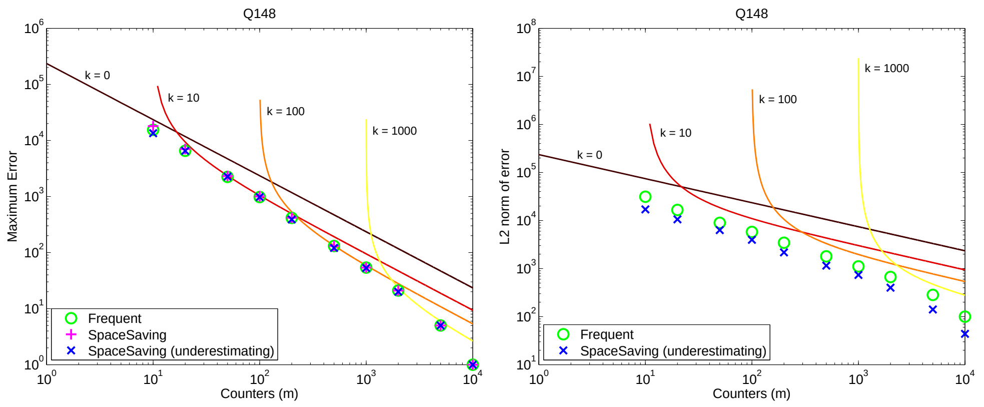
—`Webdocs`, the last dataset, was built from a spidered collection of web HTML documents. It is also available at [FIMI Repository 2008]; see [Lucchese et al. 2004] for a more detailed description. As before, we consider the transaction items in serial order, obtaining a stream of approximately 300M elements drawn from a domain of over 5M. Figure 3(c) shows the results.

As before, we see that the tail guarantees tightly bound the observed errors (left-hand plots). These data sets exhibit moderate skew, so the benefits are not as pronounced as the most skewed of the synthetic data. Nevertheless, the (data dependent) curves show that the theoretical benefit is usually a constant factor, and becomes up to an order of magnitude for m large enough. The data points from the algorithms also show that in most cases these theoretical bounds are quite tight for this data. The right-hand plots verify that the m-sparse recovery bounds hold in practice, with the modified SPACESAVING usually resulting in somewhat better approximations compared to FREQUENT.

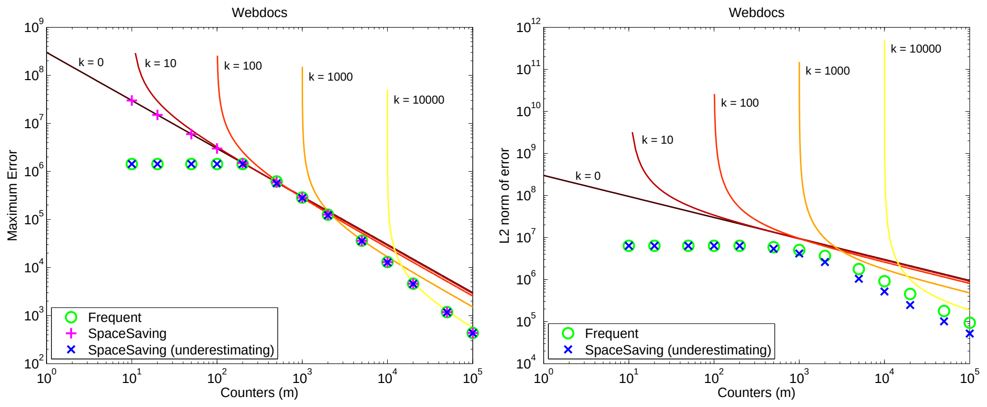
Note that for the `Kosarak` and `q148` data sets (Figures 3(a) and 3(b)), we stop before $m = 10^5$, since here the total number of distinct elements is less than m, and hence the counter algorithms obtain the exact count for each item; i.e., there exist $k < m$ such that $F_1^{res(k)}$ is 0. However, the naive F_1 guarantee still predicts a worst case error of the order



(a) `kosarak` data set



(b) `q148` data set



(c) `webdocs` data set

Fig. 3. Experimental results on real datasets. The curves in the left-hand plots show k-tail guarantees; the curves in the right-hand plots show m-sparse recovery L_2 error bounds established by theorem 7.

Algorithm 3: FREQUENT$\mathbb{R}(m)$

```
$T \leftarrow \emptyset$;
foreach $(i, w)$ do
    if $i \in T$ then $c_i \leftarrow c_i + w$;
    else
        $T \leftarrow T \cup \{i\}$; $c_i \leftarrow w$;
        if $|T| > m$ then
            $c_{min} = \min_{j \in T} c_j$;
            forall $j \in T$ do
                $c_j \leftarrow c_j - c_{min}$;
                if $c_j = 0$ then
                    $T \leftarrow T \backslash \{j\}$;
```

Algorithm 4: SPACESAVING$\mathbb{R}(m)$

```
$T \leftarrow \emptyset$;
foreach $(i, w)$ do
    if $i \in T$ then
        $c_i \leftarrow c_i + w$;
    else if $|T| < m$ then
        $T \leftarrow T \cup \{i\}$;
        $c_i \leftarrow w$;
    else
        $j \leftarrow \arg\min_{j \in T} c_j$;
        $c_i \leftarrow c_j + w$;
        $T \leftarrow T \cup \{i\} \backslash \{j\}$;
```

Fig. 4. Pseudocode for FREQUENT$\mathbb{R}$ and SPACESAVING$\mathbb{R}$ algorithms

of 10^2 at this point. On the Webdocs data (Figure 3(c)), the data is more uniform: not until the several hundred largest items have been removed is the tail noticeably smaller than the overall F_1. Nevertheless, for k large enough, the benefits are palpable. The visible disparity between SPACESAVING and the other two algorithms in this figure for values of m of 100 or less is due to how the algorithms behave when the data set is insufficiently skewed: SPACESAVING always increments a counter and thus when the top-m items do not amount to a significant fraction of the whole stream, the m counters will all be approximately F_1/m, whereas in the other algorithms the m counters will be close to 0.

8. EXTENSIONS

8.1 Real-Valued Update Streams

So far, we have considered a model of streams where each stream token indicates an arrival of an item with (implicit) unit weight. More generally, streams often include a weight for each arrival: a size in bytes or round-trip time in seconds for Internet packets; a unit price for transactional data, and so on. When these weights are large, or not necessarily integral, it is still desirable to solve heavy hitters and related problems on such streams.

In this section, we make the observation that the two counter algorithms FREQUENT and SPACESAVING naturally extend to streams in which each update includes a positive real valued weight to apply to the given item. That is, the stream consists of tuples t, The jth tuple t_j is a pair (i, w) representing the arrival of an amount of weight w to be counted towards element i where $w \in \mathbb{R}^+$ is a positive real value.

We show how to extend the two algorithms to correctly process such streams. For SPACESAVING, observe that when processing each new item i, the algorithm identifies a counter corresponding to i and increments it by 1. We simply change this to incrementing the appropriate counter by w to generate an algorithm we denote SPACESAVING$\mathbb{R}$. This generalizes SPACESAVING, since when every w is 1, then the two algorithms behave identically. The pseudocode for SPACESAVING$\mathbb{R}$ is shown in Algorithm 4.

THEOREM 11. *Algorithm* SPACESAVING$\mathbb{R}$ *provides a k-tail guarantees with $A = B = 1$ over real-valued non-negative update streams.*

PROOF. We first prove the weaker result, that SPACESAVING$\mathbb{R}$ gives a standard heavy hitters guarantee. This is a generalization of the analysis of [Metwally et al. 2005] to demonstrate that SPACESAVING$\mathbb{R}$ achieves the basic Heavy Hitters guarantee (Definition 1).

The following invariant holds by induction over updates: $\sum_{j=1}^{n} c_j = \sum_{t_j=(i,w)} w = \sum_{j=1}^{n} f_j$: in other words, the (generalization of) the F_1 of the count vector c is equal to the F_1 of the frequencies of items. Also, since there are k counters, the smallest non-zero counter, $\Delta = \min_{1\leq j\leq n: c_j>0} c_j$, is at most F_1/k. The true count of any element i which is not recorded in the data structure is at most Δ, by induction over the sequence of updates (it is true initially, and remains true over each operation). This gives us the result that δ_i, the error in the estimate of any frequency f_i, is at most Δ. Thus, when an uncounted item is stored in the data structure, we associate it with the current value of Δ, which is always an overestimate of the true count. Hence, every approximate count is an overestimate. Suppose we do not store some item i whose true decayed count is above F_1/k. Then we must have overwritten i with another item e when the minimum count was Δ_e. But since our estimate of the count of i at any instant is guaranteed to be an overestimate, this gives a contradiction, since the true count of i is at least $F_1/k \geq \Delta \geq \Delta_e$; consequently, we would not have overwritten i. Therefore, we can conclude that the algorithm retains information about all items with count at least F_1/k. The estimated counts are off by at most F_1/k, so we have the heavy hitter guarantee.

Following the same argument as in [Metwally et al. 2005], we also have that the ith largest counter is at least f_i, the ith largest frequency. This gives us the same guarantees that we had in Section 4.2. Since the argument there did not require that the f_is were integral, it follows by the same steps that $\delta_i \leq F_1^{res(k)}.(m-k)$, i.e. we have the k-tail guarantee with parameters $A = B = 1$ for SPACESAVING$\mathbb{R}$. □

Defining FREQUENT$\mathbb{R}$ is a little more complex. If the new item i is present in T, then we can simply increase c_i by w; and if there are fewer than $m-1$ counters then one can be allocated to i and set to w. But, if i is not stored, then the next step depends on the size of $c_{\min}$, the smallest counter value stored in T. If $w \leq c_{\min}$, then all stored counters are reduced by w. Otherwise, all counters are reduced by $c_{\min}$, and some counter with zero count (there must be at least one now) is assigned to i and given count $w - c_{\min}$. Following this, items with zero count are removed from T. Equivalently, we can view this as always adding w onto the corresponding counter c_i, but, if the resulting set of "active" counters exceeds k in quantity, then we subtract the value of the smallest non-zero counter from *all* $k+1$ non-zero counters, which necessarily leaves at most k non-zero counters remaining. This algorithm is given in pseudocode by Algorithm 3. We can also show a k-tail guarantee for this generalized algorithm.

THEOREM 12. *Algorithm* FREQUENT$\mathbb{R}$ *provides the k-tail guarantees with $A = B = 1$ over real-valued non-negative update streams.*

PROOF. The fact that FREQUENT$\mathbb{R}$ achieves the basic Heavy Hitter guarantee follows almost immediately, by observing that every subtraction of counter values for a given item coincides with the same subtraction to m others, and all counter increments correspond to some w of a particular item. Therefore, by a charging argument, the error in the count of any item is at most F_1/m, since any reduction to any particular c_j is charged against the frequencies of $m-1$ other items.

This algorithm FREQUENT$\mathbb{R}$ shares many properties in common with FREQUENT. In particular, we can adapt the argument of Section 4.1: now, let d denote the sum of all subtractions made for any arriving item. That is, whenever all k counters are occupied and they are all reduced by $\min(w, c_{min})$, we count this quantity $\min(w, c_{min})$ towards d. Then we have a relationship between $\|c\|_1$, the sum of all counters, and $\|f\|_1$, the sum of all frequencies, as

$$\|c\|_1 = \|f\|_1 - d(m+1)$$

From here, we can follow the same chain of argument given in Section 4.1, since there is no assumption that the counters or frequencies are integers, and conclude that

$$d \leq \frac{F_1^{res(k)}}{m-k+1}.$$

That is, we have the k-tail guarantee with $A = B = 1$. □

8.2 Merging Multiple Summaries

A consequence of sparse recovery is the fact that multiple summaries of separate streams can be merged together to create a summary of the union of the streams. More formally, consider ℓ streams, defining frequency distributions $f^{(1)} \ldots f^{(\ell)}$ respectively. Given a summary of each stream produced by the (same) algorithm with m counters, the aim is to construct an accurate summary of $f = \sum_{j=1}^{\ell} f^{(j)}$. For the analysis, we require the following bound:

LEMMA 13. *For any n-dimensional vectors x and y,*

$$|F_1^{res(k)}(x) - F_1^{res(k)}(y)| \leq \|x - y\|_1$$

PROOF. Let X denote the set of k largest entries of x, and Y the set of k largest entries of y. Let $\pi(i)$ determine any bijection from $i \in Y \backslash X$ to $\pi(i) \in X \backslash Y$. Then

$$\begin{aligned} F_1^{res(k)}(x) - F_1^{res(k)}(y) &= \sum_{i \notin X} x_i - \sum_{i \notin Y} y_i \\ &\leq \sum_{i \in Y \backslash X} x_{\pi(i)} - \sum_{i \in X \backslash Y} y_i + \sum_{i \notin (X \cup Y)} |x_i - y_i| \\ &= \sum_{i \notin Y} |x_i - y_i| \leq \sum_i |x_i - y_i| \leq \|x - y\|_1 \end{aligned}$$

Interchanging the roles of x and y gives the final result. □

We now give the main Theorem in this section:

THEOREM 14. *Given summaries of each $f^{(j)}$ produced by a counter algorithm that provides a k-tail guarantee with constants (A, B), a summary of f can be obtained with a k-tail guarantee with constants $(3A, B + A)$.*

PROOF. We construct a summary by first building a k-sparse vector $f'^{(j)}$ from the summary of $f^{(j)}$, with the guarantee of equation (2). By generating a stream corresponding to this vector for each stream, and feeding this into the counter algorithm, we obtain a summary of the distribution $f' = \sum_{j=1}^{\ell} f'^{(j)}$. Now observe that from this we have an estimated frequency for any item i as c_i so that

$$|c_i - f_i| \leq \Delta = \Delta_{f'} + \sum_{j=1}^{\ell} \Delta_j$$

where each Δ_j is the error from summarizing $f^{(j)}$ by $f'^{(j)}$, while $\Delta_{f'}$ is the error from summarizing f'.

Using Lemma 13 lets us place an upper bound on the first component of the error:

$$\Delta_{f'} \leq \frac{A}{m - Bk} F_1^{res(k)}(f') \leq \frac{A}{m - Bk}(F_1^{res(k)}(f) + \|f - f'\|_1)$$

where, by the triangle inequality and the proof of Theorem 6,

$$\|f - f'\|_1 \leq \sum_{j=1}^{\ell} \|f^{(j)} - f'^{(j)}\|_1 \leq \sum_{j=1}^{\ell}(3k\Delta_j + F_1^{res(k)}(f^{(j)}))$$

Since $\Delta_j \leq AF_1^{res(k)}(f^{(j)})/(m - Bk)$, the total error obeys

$$\Delta \leq \frac{A}{m - Bk} \left(F_1^{res(k)}(f) + \sum_{j=1}^{\ell}(3k\Delta_j + 2F_1^{res(k)}(f^{(j)})) \right)$$

We observe that

$$\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq F_1^{res(k)} \left(\sum_{j=1}^{\ell} f^{(j)} \right) = F_1^{res(k)}(f)$$

since $\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq \sum_{j=1}^{\ell} \sum_{i \notin T} f^{(j)}$ for any T such that $|T| = k$. So

$$\begin{aligned}\Delta \leq & \frac{A}{m - Bk} \left(3F_1^{res(k)}(f) + 3k \frac{A}{m - Bk}(F_1^{res(k)}(f)) \right) \\ = & \frac{3A}{m - Bk} \left(1 + \frac{Ak}{m - Bk} \right) F_1^{res(k)}(f))\end{aligned}$$

This can be analyzed as follows:

$$\begin{aligned}(m - Bk)^2 - (Ak)^2 \leq & (m - Bk)^2 \\ (m - Bk + Ak)(m - Bk - Ak) \leq & (m - Bk)^2 \\ 1 + \frac{Ak}{m - Bk} \leq & \frac{(m - Bk)}{m - (A + B)k} \\ \frac{3A}{m - Bk}\left(1 + \frac{Ak}{m - Bk}\right) \leq & \frac{3A}{m - (A + B)k}\end{aligned}$$

Hence, we have a $(3A, A + B)$ guarantee for the k-tail estimation. $\square$

In particular, since the two counter algorithms analyzed have k tail guarantees with constants $(1, 1)$, their summaries can be merged in this way to obtain k tail summaries with constants $(3, 2)$. Equivalently, this means to obtain a desired error Δ, we need to pick the number of counters m to be at most a constant factor (three) times larger to give the same bound on merging multiple summaries as for a single summary.

REFERENCES

ARASU, A., BABU, S., AND WIDOM, J. 2003. CQL: A language for continuous queries over streams and relations. In *Proceedings of the 9th DBPL International Confenrence on Data Base and Programming Languages*. 1–11.

BERINDE, R., CORMODE, G., INDYK, P., AND STRAUSS, M. 2009. Space-optimal heavy hitters with strong error bounds. *ACM Symposion on Principles of Database Systems*.

BERINDE, R., GILBERT, A., INDYK, P., KARLOFF, H., AND STRAUSS, M. 2008. Combining geometry and combinatorics: a unified approach to sparse signal recovery. In *Allerton Conference*.

BERINDE, R., INDYK, P., AND RUZIC, M. 2008. Practical near-optimal sparse recovery in the l_1 norm. In *Allerton Conference*.

BESTAVROS, A., CROVELLA, M., AND TAQQU, T. 1999. *Heavy-Tailed Probability Distributions in the World Wide Web*. Birkhäuser, 3–25.

BEYER, K. AND RAMAKRISHNAN, R. 1999. Bottom-up computation of sparse and iceberg cubes. In *ACM SIGMOD International Conference on Management of Data*. 359–370.

BONNET, P., GEHRKE, J., AND SESHADRI, P. 2001. Towards sensor database systems. In *Proceedings of the 2nd IEEE MDM International Conference on Mobile Data Management*. 3–14.

BOSE, P., KRANAKIS, E., MORIN, P., AND TANG, Y. 2003. Bounds for frequency estimation of packet streams. In *Proceedings of the 10th International Colloquium on Structural Information and Communication Complexity*. 33–42.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. 1999. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM*. 126–134.

CANDÈS, E. J., ROMBERG, J., AND TAO, T. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics 59,* 8, 1208–1223.

CHAKRABARTI, A., CORMODE, G., AND MCGREGOR, A. 2007. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*.

CHARIKAR, M., CHEN, K., AND FARACH-COLTON, M. 2002. Finding frequent items in data streams. In *Proceedings of the 29th ICALP International Colloqium on Automata, Languages and Programming*. 693–703.

CORMODE, G., GOLAB, L., KORN, F., MCGREGOR, A., SRIVASTAVA, D., AND ZHANG, X. 2009. Estimating the confidence of conditional functional dependencies. In *Proceedings of ACM-SIGMOD*.

CORMODE, G. AND HADJIELEFTHERIOU, M. 2008. Finding frequent items in data streams. *PVLDB 1,* 2, 1530–1541.

CORMODE, G., KORN, F., MUTHUKRISHNAN, S., AND SRIVASTAVA, D. 2003. Finding hierarchical heavy hitters in data streams. In *International Conference on Very Large Data Bases*. 464–475.

CORMODE, G. AND MUTHUKRISHNAN, S. 2005. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms 55,* 1, 58–75.

DEMAINE, E., ORTIZ, A. L., AND MUNRO, J. 2002. Frequency estimation of internet packet streams with limited space. In *Proceedings of the 10th ESA Annual European Symposium on Algorithms*. 348–360.

DONOHO, D. L. 2006. Compressed Sensing. *IEEE Trans. Info. Theory 52,* 4 (Apr.), 1289–1306.

ESTAN, C. AND VARGHESE, G. 2001. New directions in traffic measurement and accounting. In *ACM SIGCOMM Internet Measurement Workshop*.

FANG, M., SHIVAKUMAR, N., GARCIA-MOLINA, H., MOTWANI, R., AND ULLMAN, J. 1998. Computing iceberg queries efficiently. In *International Conference on Very Large Data Bases*. 299–310.

FIMI Repository 2008. Frequent itemset mining dataset repository, University of Helsinki. Available at `http://fimi.cs.helsinki.fi/data`.

GANGULY, S. AND LAKSHMINATH, B. 2006. Estimating entropy over data streams. In *Proceedings of European Symposium on Algorithms (ESA)*.

GAROFALAKIS, M., GEHRKE, J., AND RASTOGI, R. 2002. Querying and mining data streams: You only get one look. In *Proceedings of ACM-SIGMOD*.

GILBERT, A. C., STRAUSS, M. J., TROPP, J. A., AND VERSHYNIN, R. 2007. One sketch for all: fast algorithms for compressed sensing. In *ACM Symposium on Theory of Computing*. 237–246.

HAN, J., PEI, J., DONG, G., AND WANG, K. 2001. Efficient computation of iceberg cubes with complex measures. In *ACM SIGMOD International Conference on Management of Data*. 1–12.

HERSHBERGER, J., SHRIVASTAVA, N., SURI, S., AND TÓTH, C. D. 2005. Space complexity of hierarchical heavy hitters in multi-dimensional streams. In *ACM Principles of Database Systems*. 338–347.

INDYK, P. 2004. Algorithms for dynamic geometric problems over data streams. In *ACM Symposium on Theory of Computing*.

INDYK, P. 2007. Sketching, streaming and sublinear-space algorithms. Graduate course notes, available at `http://stellar.mit.edu/S/course/6/fa07/6.895/`.

INDYK, P. AND RUZIC, M. 2008. Near-optimal sparse recovery in the l_1 norm. In *IEEE Conference on Foundations of Computer Science*.

KARP, R. M., SHENKER, S., AND PAPADIMITRIOU, C. H. 2003. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS) 28,* 1, 51–55.

KOHAVI, R., BRODLEY, C., FRASCA, B., MASON, L., AND ZHENG, Z. 2000. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations 2,* 2, 86–98.

LUCCHESE, C., ORLANDO, S., PEREGO, R., AND SILVESTRI, F. 2004. Webdocs: a real-life huge transactional dataset. In *FIMI*.

MANERIKAR, N. AND PALPANAS, T. 2009. Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *Data and Knowledge Engineering 68,* 4.

MANKU, G. AND MOTWANI, R. 2002. Approximate frequency counts over data streams. In *International Conference on Very Large Data Bases*. 346–357.

METWALLY, A., AGRAWAL, D., AND ABBABI, A. 2005. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*. 398–412.

MISRA, J. AND GRIES, D. 1982. Finding repeated elements. *Science of Computer Programming 2*, 142–152.

MUTHUKRISHNAN, S. 2005. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science.

REDNER, S. 1998. How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B*, 131–134.

Rice DSP Group. Compressed sensing resources. Available at `http://www.dsp.ece.rice.edu/cs`.

SHRIVASTAVA, N., BURAGOHAIN, C., AGRAWAL, D., AND SURI, S. 2004. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Network Sensor Systems*. 239–249.

ZIPF, G. 1949. *Human Behavior and The Principle of Least Effort*. Addison-Wesley.