

An Improved Data Stream Summary: The Count-Min Sketch and its Applications

Graham Cormode* and S. Muthukrishnan**

Abstract. We introduce a new *sublinear space* data structure—the *Count-Min Sketch*—for summarizing data streams. Our sketch allows fundamental queries in data stream summarization such as point, range, and inner product queries to be approximately answered very quickly; in addition, it can be applied to solve several important problems in data streams such as finding quantiles, frequent items, etc. The time and space bounds we show for using the CM sketch to solve these problems significantly improve those previously known — typically from $1/\varepsilon^2$ to $1/\varepsilon$ in factor.

1 Introduction

We consider a vector \mathbf{a} , which is presented in an implicit, incremental fashion. This vector has dimension n , and its current state at time t is $\mathbf{a}(t) = [a_1(t), \dots, a_i(t), \dots, a_n(t)]$. Initially, \mathbf{a} is the zero vector, $a_i(0) = 0$ for all i . Updates to individual entries of the vector are presented as a stream of pairs. The t th update is (i_t, c_t) , meaning that $a_{i_t}(t) = a_{i_t}(t-1) + c_t$, and $a_{i'}(t) = a_{i'}(t-1)$ for all $i' \neq i_t$. At any time t , a *query* calls for computing certain functions of interest on $\mathbf{a}(t)$.

This setup is the *data stream* scenario that has emerged recently. Algorithms for computing functions within the data stream context need to satisfy the following desiderata. First, the space used by the algorithm should be small, at most polylogarithmic in n , the space required to represent \mathbf{a} explicitly. Since the space is sublinear in data and input size, the data structures used by the algorithms to represent the input data stream is merely a summary—aka a *sketch* or synopsis [10])—of it; because of this compression, almost no function that one needs to compute on \mathbf{a} can be done precisely, so some approximation is provably needed. Second, processing an update should be fast and simple; likewise, answering queries of a given type should be fast and have usable accuracy guarantees. Typically, accuracy guarantees will be made in terms of a pair of user specified parameters, ε and δ , meaning that the error in answering a query is within a factor of ε with probability δ . The space and update time will consequently depend on ε and δ ; our goal will be limit this dependence as much as is possible.

Many applications that deal with massive data, such as Internet traffic analysis and monitoring contents of massive databases, motivate this one-pass data stream setup.

* graham@dimacs.rutgers.edu Center for Discrete Mathematics and Computer Science (DIMACS) Rutgers University, Piscataway NJ. Supported by NSF ITR 0220280 and NSF EIA 02-05116.

** muthu@cs.rutgers.edu Division of Computer and Information Systems, Rutgers University. Supported by NSF EIA 0087022, NSF ITR 0220280 and NSF EIA 02-05116.

There has been a frenzy of activity recently in the Algorithm, Database and Networking communities on such data stream problems, with multiple surveys, tutorials, workshops and research papers. See [7, 3, 16] for detailed description of the motivations driving this area.

In recent years, several different sketches have been proposed in the data stream context that allow a number of simple aggregation functions to be approximated. Quantities for which efficient sketches have been designed include the L_1 and L_2 norms of vectors [2], the number of distinct items in a sequence (ie number of non-zero entries in $\mathbf{a}(t)$) [8], join and self-join sizes of relations (representable as inner-products of vectors $\mathbf{a}(t), \mathbf{b}(t)$) [2, 1], item and range sum queries [12, 4]. These sketches are of interest not simply because they can be used to directly approximate quantities of interest, but also because they have been used considerably as “black box” devices in order to compute more sophisticated aggregates and complex quantities: quantiles [13], wavelets [12], and histograms [11]. Sketches thus far designed are typically linear functions of their input, and can be represented as projections of an underlying vector representing the data with certain randomly chosen projection matrices. This means that it is easy to compute certain functions on data that is distributed over sites, by casting them as computations on their sketches. So, they are suited for distributed applications too.

While sketches have proved powerful, they have the following drawbacks.

- Although sketches use small space, the space used typically has a $\Omega(1/\varepsilon^2)$ multiplicative factor. This is discouraging because $\varepsilon = 0.1$ or 0.01 is quite reasonable and already, this factor proves expensive in space, and consequently, often, in per-update processing and function computation times as well.
- Many sketch constructions require time linear in the size of the sketch to process each update to the underlying data [2, 13]. Sketches are typically a few kilobytes up to a megabyte or so, and processing this much data for every update severely limits the update speed.
- Sketches are typically constructed using hash functions with strong independence guarantees, such as p -wise independence [2], which can be complicated to evaluate, particularly for a hardware implementation. One of the fundamental questions is to what extent such sophisticated independence properties are needed.
- Many sketches described in the literature are good for one single, pre-specified aggregate computation. Given that in data stream applications one typically monitors multiple aggregates on the same stream, this calls for using many different types of sketches, which is a prohibitive overhead.
- Known analyses of sketches hide large multiplicative constants in big-Oh notation.

Given that the area of data streams is being motivated by extremely high performance monitoring applications—eg., see [7] for response time requirements for data stream algorithms that monitor IP packet streams—these drawbacks ultimately limit the use of many known data stream algorithms within suitable applications.

We will address all these issues by proposing a new sketch construction, which we call the *Count-Min*, or CM, sketch. This sketch has the advantages that: (1) space used is proportional to $1/\varepsilon$; (2) the update time is significantly sublinear in the size of the sketch; (3) it requires only pairwise independent hash functions that are simple to construct; (4) this sketch can be used for several different queries and multiple applications;

and (5) all the constants are made explicit and are small. Thus, for the applications we discuss, our constructions strictly improve the space bounds of previous results from $1/\varepsilon^2$ to $1/\varepsilon$ and the time bounds from $1/\varepsilon^2$ to 1, which is significant.

Recently, a $\Omega(1/\varepsilon^2)$ space lower bound was shown for a number of data stream problems: approximating frequency moments $F_k(t) = \sum_k (a_i(t))^k$, estimating the number of distinct items, and computing the Hamming distance between two strings [17]. It is an interesting contrast that for a number of similar seeming problems (finding Heavy Hitters and Quantiles in the most general data stream model) we are able to give an $O(\frac{1}{\varepsilon})$ upper bound. Conceptually, CM Sketch also represents progress since it shows that pairwise independent hash functions suffice for many of the fundamental data stream applications. From a technical point of view, CM Sketch and its analyses are quite simple. We believe that this approach moves some of the fundamental data stream algorithms from the theoretical realm to the practical. Our results have some technical nuances: (1) The accuracy estimates for individual queries depend on the L_1 norm of $\mathbf{a}(t)$ in contrast to the previous works that depend on the L_2 norm. (2) Most prior sketch constructions relied on embedding into small dimensions to estimate norms. Avoiding such embeddings allows our construction to avoid $\Omega(\frac{1}{\varepsilon^2})$ lower-bounds on these embeddings.

2 Preliminaries

We consider a vector \mathbf{a} , which is presented in an implicit, incremental fashion. This vector has dimension n , and its current state at time t is $\mathbf{a}(t) = [a_1(t), \dots, a_i(t), \dots, a_n(t)]$. For convenience, we shall usually drop t and refer only to the current state of the vector. Initially, \mathbf{a} is the zero vector, $\mathbf{0}$, so $a_i(0)$ is 0 for all i . Updates to individual entries of the vector are presented as a stream of pairs. The t th update is (i_t, c_t) , meaning that

$$a_{i_t}(t) = a_{i_t}(t-1) + c_t; \quad a_{i'}(t) = a_{i'}(t-1) \quad i' \neq i_t$$

In some cases, c_t s will be strictly positive, meaning that entries only increase; in other cases, c_t s are allowed to be negative also. The former is known as the *cash register* case and the latter the *turnstile* case [16]. There are two important variations of the turnstile case to consider: whether a_i s may become negative, or whether the application generating the updates guarantees that this will never be the case. We refer to the first of these as the *general* case, and the second as the *non-negative* case. Many applications that use sketches to compute queries of interest—such as monitoring database contents, analyzing IP traffic seen in a network link—guarantee that counts will never be negative. However, the general case occurs in important scenarios too, for example in distributed settings where one considers the subtraction of one vector from another, say.

At any time t , a *query* calls for computing certain functions of interest on $\mathbf{a}(t)$. We focus on approximating answers to three types of query based on vectors \mathbf{a} and \mathbf{b} .

- A *point query*, denoted $\mathcal{Q}(i)$, is to return an approximation of a_i .
- A *range query* $\mathcal{Q}(l, r)$ is to return an approximation of $\sum_{i=l}^r a_i$.
- An *inner product query*, denoted $\mathcal{Q}(\mathbf{a}, \mathbf{b})$ is to approximate $\mathbf{a} \odot \mathbf{b} = \sum_{i=1}^n a_i b_i$.

These queries are related: a range query is a sum of point queries; both point and range queries are specific inner product queries. However, in terms of approximations to these queries, results will vary. These are the queries that are fundamental to many applications in data stream algorithms, and have been extensively studied. In addition, they are of interest in non-data stream context. For example, in databases, the point and range queries are of interest in summarizing the data distribution approximately; and inner-product queries allow approximation of join size of relations. Fuller discussion of these aspects can be found in [9, 16].

We will also study use of these queries to compute more complex functions on data streams. As examples, we will focus on the two following problems. Recall that $\|\mathbf{a}\|_1 = \sum_{i=1}^n |a_i(t)|$; more generally, $\|\mathbf{a}\|_p = (\sum_{i=1}^n |a_i(t)|^p)^{1/p}$.

- (*ϕ -Quantiles*) The ϕ -quantiles of the cardinality $\|\mathbf{a}\|_1$ multiset of (integer) values each in the range $1 \dots n$ consist of those items with rank $k\phi\|\mathbf{a}\|_1$ for $k = 0 \dots 1/\phi$ after sorting the values. Approximation comes by accepting any integer that is between the item with rank $(k\phi - \epsilon)\|\mathbf{a}\|_1$ and the one with rank $(k\phi + \epsilon)\|\mathbf{a}\|_1$ for some specified $\epsilon < \phi$.
- (*Heavy Hitters*) The ϕ -heavy hitters of a multiset of $\|\mathbf{a}\|_1$ (integer) values each in the range $1 \dots n$, consist of those items whose multiplicity exceeds the fraction ϕ of the total cardinality, i.e., $a_i \geq \phi\|\mathbf{a}\|_1$. There can be between 0 and $\frac{1}{\phi}$ heavy hitters in any given sequence of items. Approximation comes by accepting any i such that $a_i \geq (\phi - \epsilon)\|\mathbf{a}\|_1$ for some specified $\epsilon < \phi$.

Our goal is to solve the queries and the problems above using a sketch data structure, that is using space and time significantly sublinear—polylogarithmic—in input size n and $\|\mathbf{a}\|_1$. All our algorithms will be approximate and probabilistic; they need two parameters, ϵ and δ , meaning that the error in answering a query is within a factor of ϵ with probability δ . Both these parameters will affect the space and time needed by our solutions. Each of these queries and problems has a rich history of work in the data stream area. We refer the readers to surveys [16, 3], tutorials [9], as well as the general literature.

3 Count-Min Sketches

We now introduce our data structure, the Count-Min, or CM, sketch. It is named after the two basic operations used to answer point queries, counting first and computing the minimum next. We use e to denote the base of the natural logarithm function, \ln .

Data Structure. A *Count-Min (CM) sketch* with parameters (ϵ, δ) is represented by a two-dimensional array counts with width w and depth d : $count[1, 1] \dots count[d, w]$. Given parameters (ϵ, δ) , set $w = \lceil \frac{e}{\epsilon} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$. Each entry of the array is initially zero. Additionally, d hash functions $h_1 \dots h_d : \{1 \dots n\} \rightarrow \{1 \dots w\}$ are chosen uniformly at random from a pairwise-independent family.

Update Procedure. When an update (i_t, c_t) arrives, meaning that item a_{i_t} is updated by a quantity of c_t , then c_t is added to one count in each row; the counter is determined by h_j . Formally, set $\forall 1 \leq j \leq d : \text{count}[j, h_j(i_t)] \leftarrow \text{count}[j, h_j(i_t)] + c_t$. The space used by Count-Min sketches is the array of wd counts, which takes wd words, and d hash functions, each of which can be stored using 2 words when using the pairwise functions described in [15].

4 Approximate Query Answering Using CM Sketches

For each of the three queries introduced in Section 2: Point, Range, and Inner Product queries, we show how they can be answered using Count-Min sketches.

4.1 Point Query

We first show the analysis for point queries for the non-negative case.

Estimation Procedure. The answer to $\mathcal{Q}(i)$ is given by $\hat{a}_i = \min_j \text{count}[j, h_j(i)]$.

Theorem 1. *The estimate \hat{a}_i has the following guarantees: $a_i \leq \hat{a}_i$; and, with probability at least $1 - \delta$, $\hat{a}_i \leq a_i + \epsilon \|\mathbf{a}\|_1$.*

Proof. We introduce indicator variables $I_{i,j,k}$, which are 1 if $(i \neq k) \wedge (h_j(i) = h_j(k))$, and 0 otherwise. By pairwise independence of the hash functions, then

$$\mathbb{E}(I_{i,j,k}) = \Pr[h_j(i) = h_j(k)] \leq 1/\text{range}(h_j) = \frac{\epsilon}{e}.$$

Define the variable $X_{i,j}$ (random over the choices of h_i) to be $X_{i,j} = \sum_{k=1}^n I_{i,j,k} a_k$. Since all a_i are non-negative in this case, $X_{i,j}$ is a non-negative variable. By construction, $\text{count}[j, h_j(i)] = a_i + X_{i,j}$. So, clearly, $\min \text{count}[j, h_j(i)] \geq a_i$. For the other direction, observe that

$$\mathbb{E}(X_{i,j}) = \mathbb{E}\left(\sum_{k=1}^n I_{i,j,k} a_k\right) \leq \sum_{k=1}^n a_k \mathbb{E}(I_{i,j,k}) \leq \frac{\epsilon}{e} \|\mathbf{a}\|_1$$

by pairwise independence of h_j , and linearity of expectation. By the Markov inequality,

$$\begin{aligned} \Pr[\hat{a}_i > a_i + \epsilon \|\mathbf{a}\|_1] &= \Pr[\forall j. \text{count}[j, h_j(i)] > a_i + \epsilon \|\mathbf{a}\|_1] \\ &= \Pr[\forall j. a_i + X_{i,j} > a_i + \epsilon \|\mathbf{a}\|_1] \\ &= \Pr[\forall j. X_{i,j} > \epsilon \|\mathbf{a}\|_1] < e^{-d} \leq \delta \quad \blacksquare \blacksquare \end{aligned}$$

The time to produce the estimate is $O(\ln \frac{1}{\delta})$ since finding the minimum count can be done in linear time; the same time bound holds for updates. The constant e is used here to minimize the space used: more generally, we can set $w = \epsilon/b$ and $d = \log_b \frac{1}{\delta}$ for any $b > 1$ to get the same accuracy guarantee. Choosing $b = e$ minimizes the space used, since this solves $\frac{d(wd)}{db} = 0$, giving a cost of $(2 + \frac{\epsilon}{e}) \ln \frac{1}{\delta}$ words. For implementations, it may be preferable to use other (integer) values of b for simpler computations or faster updates.

The best known previous result using sketches was in [4]: there sketches were used to approximate point queries. Results were stated in terms of the frequencies of individual items. For arbitrary distributions, the space used is $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$, and the dependency on ε is $\frac{1}{\varepsilon^2}$ in every case considered.

In the full version of this paper¹ we describe how all existing sketch constructions can be viewed as variations of a common construction. This emphasizes the importance of our attempt to find the simplest sketch construction which has the best guarantees and smallest constants. A similar result holds when entries of the implicit vector \mathbf{a} may be negative, which is the general case. Details of this appear in the full version of this paper.

4.2 Inner Product Query

Estimation Procedure. Set $(\widehat{\mathbf{a} \odot \mathbf{b}})_j = \sum_{k=1}^w \text{count}_{\mathbf{a}}[j, k] * \text{count}_{\mathbf{b}}[j, k]$. Our estimation of $\mathcal{Q}(\mathbf{a}, \mathbf{b})$ for non-negative vectors \mathbf{a} and \mathbf{b} is $\widehat{\mathbf{a} \odot \mathbf{b}} = \min_j (\widehat{\mathbf{a} \odot \mathbf{b}})_j$.

Theorem 2. $\mathbf{a} \odot \mathbf{b} \leq \widehat{\mathbf{a} \odot \mathbf{b}}$ and, with probability $1 - \delta$, $\widehat{\mathbf{a} \odot \mathbf{b}} \leq \mathbf{a} \odot \mathbf{b} + \varepsilon \|\mathbf{a}\|_1 \|\mathbf{b}\|_1$.

Proof.

$$(\widehat{\mathbf{a} \odot \mathbf{b}})_j = \sum_{i=1}^n a_i b_i + \sum_{p \neq q, h_j(p)=h_j(q)} a_p b_q$$

Clearly, $\mathbf{a} \odot \mathbf{b} \leq \widehat{\mathbf{a} \odot \mathbf{b}}$ for non-negative vectors. By pairwise independence of h ,

$$\mathbb{E}(\widehat{\mathbf{a} \odot \mathbf{b}}_j - \mathbf{a} \odot \mathbf{b}) = \sum_{p \neq q} \Pr[h_j(p) = h_j(q)] a_p b_q \leq \sum_{p \neq q} \frac{\varepsilon a_p b_q}{e} \leq \frac{\varepsilon \|\mathbf{a}\|_1 \|\mathbf{b}\|_1}{e}$$

So, by the Markov inequality, $\Pr[\widehat{\mathbf{a} \odot \mathbf{b}} - \mathbf{a} \odot \mathbf{b} > \varepsilon \|\mathbf{a}\|_1 \|\mathbf{b}\|_1] \leq \delta$, as required. ■

The space and time to produce the estimate is $O(\frac{1}{\varepsilon} \log \frac{1}{\delta})$. Updates are performed in time $O(\log \frac{1}{\delta})$.

Join size estimation is important in database query planners in order to determine the best order in which to evaluate queries. The *join size* of two database relations on a particular attribute is the number of items in the cartesian product of the two relations which agree the value of that attribute. We assume without loss of generality that attribute values in the relation are integers in the range $1 \dots n$. We represent the relations being joined as vectors \mathbf{a} and \mathbf{b} so that the values a_i represents the number of tuples which have value i in the first relation, and b_i similarly for the second relation. Then clearly $\mathbf{a} \odot \mathbf{b}$ is the join size of the two relations. Using sketches allows estimates to be made in the presence of items being inserted to and deleted from relations. The following corollary follows from the above theorem.

Corollary 1. *The Join size of two relations on a particular attribute can be approximated up to $\varepsilon \|\mathbf{a}\|_1 \|\mathbf{b}\|_1$ with probability $1 - \delta$, by keeping space $O(\frac{1}{\varepsilon} \log \frac{1}{\delta})$.*

¹ To appear in Journal of Algorithms

Previous results have used the “tug-of-war” sketches [1]. However, here some care is needed in the comparison of the two methods: the prior work gives guarantees in terms of the L_2 norm of the underlying vectors, with additive error of $\varepsilon \|\mathbf{a}\|_2 \|\mathbf{b}\|_2$; here, the result is in terms of the L_1 norm. In some cases, the L_2 norm can be quadratically smaller than the L_1 norm. However, when the distribution of items is non-uniform, for example when certain items contribute a large amount to the join size, then the two norms are closer, and the guarantees of the CM sketch method is closer to the existing method. As before, the space cost of previous methods was $\Omega(\frac{1}{\varepsilon^2})$, so there is a significant space saving to be had with CM sketches.

4.3 Range Query

Estimation Procedure. We will adopt the use of *dyadic ranges* from [13]: a dyadic range is a range of the form $[x2^y + 1 \dots (x+1)2^y]$ for parameters x and y . Keep $\log_2 n$ CM sketches, in order to answer range queries $Q(l, r)$ approximately. Any range query can be reduced to at most $2 \log_2 n$ dyadic range queries, which in turn can each be reduced to a single point query. Each point in the range $[1 \dots n]$ is a member of $\log_2 n$ dyadic ranges, one for each y in the range $0 \dots \log_2(n) - 1$. A sketch is kept for each set of dyadic ranges of length 2^y , and update each of these for every update that arrives. Then, given a range query $Q(l, r)$, compute the at most $2 \log_2 n$ dyadic ranges which canonically cover the range, and pose that many point queries to the sketches, returning the sum of the queries as the estimate.

Theorem 3. $a[l, r] \leq \hat{a}[l, r]$ and with probability at least $1 - \delta$,

$$\hat{a}[l, r] \leq a[l, r] + 2\varepsilon \log n \|\mathbf{a}\|_1.$$

Proof. Applying the inequality of Theorem 1, then $a[l, r] \leq \hat{a}[l, r]$. Consider each estimator used to form $\hat{a}[l, r]$; the expectation of the additive error for any of these is $2 \log n \frac{\varepsilon}{e} \|\mathbf{a}\|_1$, by linearity of expectation of the errors of each point estimate. Applying the same Markov inequality argument as before, the probability that this additive error is more than $2\varepsilon \log n \|\mathbf{a}\|_1$ for any estimator is less than $\frac{1}{e}$; hence, for all of them the probability is at most δ . ■

The time to compute the estimate or to make an update is $O(\log(n) \log \frac{1}{\delta})$. The space used is $O(\frac{\log(n)}{\varepsilon} \log \frac{1}{\delta})$.

The above theorem states the bound for the standard CM sketch size. The guarantee will be more useful when stated without terms of $\log n$ in the approximation bound. This can be changed by increasing the size of the sketch, which is equivalent to rescaling ε . In particular, if we want to estimate a range sum correct up to $\varepsilon' \|\mathbf{a}\|_1$ with probability $1 - \delta$ then set $\varepsilon = \frac{\varepsilon'}{2 \log n}$. The space used is $O(\frac{\log^2(n)}{\varepsilon'} \log \frac{1}{\delta})$. An obvious improvement of this technique in practice is to keep exact counts for the first few levels of the hierarchy, where there are only a small number of dyadic ranges. This improves the space, time and accuracy of the algorithm in practice, although the asymptotic bounds are unaffected.

The best previous bounds for this problem in the turnstile model are given in [13], where range queries are answered by keeping $O(\log n)$ sketches, each of size

$O(\frac{1}{\varepsilon'^2} \log(n) \log \frac{\log n}{\delta})$ to give approximations with additive error $\varepsilon \|\mathbf{a}\|_1$ with probability $1 - \delta'$. Thus the space used there is $O(\frac{\log^2 n}{\varepsilon'^2} \log \frac{\log n}{\delta})$ and the time for updates is linear in the space used. The CM sketch improves the space and time bounds; it improves the constant factors as well as the asymptotic behavior. The time to process an update is significantly improved, since only a few entries in the sketch are modified, rather than a linear number.

5 Applications of Count-Min Sketches

By using CM sketches, we show how to improve best known time and space bounds for the two problems from Section 2.

5.1 Quantiles in the Turnstile Model

In [13] the authors showed that finding the approximate ϕ -quantiles of the data subject to insertions and deletions can be reduced to the problem of computing range sums. Put simply, the algorithm is to do binary searches for ranges $1 \dots r$ whose range sum $a[1, r]$ is $k\phi \|\mathbf{a}\|_1$ for $1 \leq k \leq \frac{1}{\phi} - 1$. The method of [13] uses *Random Subset Sums* to compute range sums. By replacing this structure with Count-Min sketches, the improved results follow immediately. By keeping $\log n$ sketches, one for each dyadic range and setting the accuracy parameter for each to be $\varepsilon / \log n$ and the probability guarantee to $\delta\phi / \log(n)$, the overall probability guarantee for all $1/\phi$ quantiles is achieved.

Theorem 4. *ε -approximate ϕ -quantiles can be found with probability at least $1 - \delta$ by keeping a data structure with space $O(\frac{1}{\varepsilon} \log^2(n) \log(\frac{\log n}{\phi\delta}))$. The time for each insert or delete operation is $O(\log(n) \log(\frac{\log n}{\phi\delta}))$, and the time to find each quantile on demand is $O(\log(n) \log(\frac{\log n}{\phi\delta}))$.*

Choosing CM sketches over Random Subset Sums improves both the query time and the update time from $O(\frac{1}{\varepsilon^2} \log^2(n) \log \frac{\log n}{\varepsilon\delta})$, by a factor of more than $\frac{34}{\varepsilon^2} \log n$. The space requirements are also improved by a factor of at least $\frac{34}{\varepsilon}$.

It is illustrative to contrast our bounds with those for the problem in the weaker Cash Register Model where items are only inserted (recall that in our stronger Turnstile model, items are deleted as well). The previously best known space bounds for finding approximate quantiles is $O(\frac{1}{\varepsilon} (\log^2 \frac{1}{\varepsilon} + \log^2 \log \frac{1}{\delta}))$ space for a randomized sampling and $O(\frac{1}{\varepsilon} \log(\varepsilon \|\mathbf{a}\|_1))$ space for a deterministic solution [14]. These bounds are not completely comparable, but our result is the first on the more powerful Turnstile model to be comparable to the Cash Register model bounds in the leading $1/\varepsilon$ term.

5.2 Heavy Hitters in the Turnstile Model

We adopt the solution given in [5], which describes a divide and conquer procedure to find the heavy hitters. This keeps sketches for computing range sums: $\log n$ different sketches, one for each different dyadic range. When an update (i_t, c_t) arrives, then each of these is updated as before. In order to find all the heavy hitters, a parallel binary

search is performed, descending one level of the hierarchy at each step. Nodes in the hierarchy (corresponding to dyadic ranges) whose estimated weight exceeds the threshold of $(\phi + \varepsilon)\|\mathbf{a}\|_1$ are split into two ranges, and investigated recursively. All single items found in this way whose approximated count exceeds the threshold are output.

We instead must limit the number of items output whose true frequency is less than the fraction ϕ . This is achieved by setting the probability of failure for each sketch to be $\frac{\delta\phi}{2^{\log n}}$. This is because, at each level there are at most $1/\phi$ items with frequency more than ϕ . At most twice this number of queries are made at each level, for all of the $\log n$ levels. By scaling δ like this and applying the union bound ensures that, over all the queries, the total probability that any one (or more) of them overestimated by more than a fraction ε is bounded by δ , and so the probability that every query succeeds is $1 - \delta$. It follows that

Theorem 5. *The algorithm uses space $O(\frac{1}{\varepsilon} \log(n) \log(\frac{2 \log(n)}{\delta\phi}))$, and time $O(\log(n) \log(\frac{2 \log n}{\delta\phi}))$ per update. Every item with frequency at least $(\phi + \varepsilon)\|\mathbf{a}\|_1$ is output, and with probability $1 - \delta$ no item whose frequency is less than $\phi\|\mathbf{a}\|_1$ is output.*

The previous best known bound appears in [5], where a non-adaptive group testing approach was described. Here, the space bounds agree asymptotically but have been improved in constant factors; a further improvement is in the nature of the guarantee: previous methods gave probabilistic guarantees about outputting the heavy hitters. Here, there is absolute certainty that this procedure will find and output every heavy hitter, because the CM sketches never underestimate counts, and strong guarantees are given that no non-heavy hitters will be output. This is often desirable.

In some situations in practice, it is vital that updates are as fast as possible, and here update time can be played off against search time: ranges based on powers of two can be replaced with an arbitrary branching factor k , which reduces the number of levels to $\log_k n$, at the expense of costlier queries and weaker guarantees on outputting non-heavy hitters.

6 Conclusions

We have introduced the Count-Min sketch, and shown how to estimate fundamental queries such as point, range or inner product queries as well as solve more sophisticated problems such as quantiles and heavy hitters. The space and/or time bounds of our solutions improve previously best known bounds for these problems. Typically the improvement is from $1/\varepsilon^2$ factor to $1/\varepsilon$ which is significant in real applications. Our CM sketch is quite simple, and is likely to find many applications, including in hardware solutions for these problems.

We have recently applied these ideas to the problem of change detection on data streams [6], and we also believe that it can be applied to improve the time and space bounds for constructing approximate wavelet and histogram representations of data streams [11]. Also, the CM Sketch can also be naturally extended to solve problems

on streams that describe multidimensional arrays rather than the unidimensional array problems we have discussed so far.

Our CM sketch is not effective when one wants to compute the norms of data stream inputs. These have applications to computing correlations between data streams and tracking the number of distinct elements in streams, both of which are of great interest. It is an open problem to design extremely simple, practical sketches such as our CM Sketch for estimating such correlations and more complex data stream applications.

References

1. N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems (PODS '99)*, pages 10–20, 1999.
2. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996. Journal version in *Journal of Computer and System Sciences*, 58:137–147, 1999.
3. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of Symposium on Principles of Database Systems (PODS)*, pages 1–16, 2002.
4. M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.
5. G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proceedings of ACM Principles of Database Systems*, pages 296–306, 2003.
6. G. Cormode and S. Muthukrishnan. What's new: Finding significant differences in network data streams. In *Proceedings of IEEE Infocom*, 2004.
7. C. Estan and G. Varghese. Data streaming in computer networks. In *Proceedings of Workshop on Management and Processing of Data Streams*, <http://www.research.att.com/conf/mpds2003/schedule/estanV.ps>, 2003.
8. P. Flajolet and G. N. Martin. Probabilistic counting. In *24th Annual Symposium on Foundations of Computer Science*, pages 76–82, 1983. Journal version in *Journal of Computer and System Sciences*, 31:182–209, 1985.
9. M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.
10. P. Gibbons and Y. Matias. Synopsis structures for massive data sets. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, A, 1999.
11. A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 389–398, 2002.
12. A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 79–88, 2001. Journal version in *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, 2003.
13. A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 454–465, 2002.

14. M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):58–66, 2001.
15. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
16. S. Muthukrishnan. Data streams: Algorithms and applications. In *ACM-SIAM Symposium on Discrete Algorithms*, <http://athos.rutgers.edu/~muthu/stream-1-1.ps>, 2003.
17. D. Woodruff. Optimal space lower bounds for all frequency moments. In *ACM-SIAM Symposium on Discrete Algorithms*, 2004.