

# Technical Summary: “An Improved Data Stream Summary: The Count-Min Sketch and its Applications” by Cormode and Muthukrishnan

Ravi Gaddipati, Matthew Ige, and Emily Wagner

December 6, 2016

## 1 Problem Statement and Overview

Consider a vector  $\vec{a}(t) = [a_1(t), \dots, a_i(t), \dots, a_n(t)]$  which evolves with time. Initially,  $\vec{a}$  is the zero vector  $\vec{0}$ . We represent the  $t$ th update as  $(i_t, c_t)$ , which modifies the vector as follows:

$$a_{i_t}(t) = a_{i_t}(t-1) + c_t, \tag{1}$$

$$a_{i'}(t) = a_{i'}(t-1), \quad i' \neq i_t \tag{2}$$

The update  $(i_t, c_t)$  modifies the  $i$ th element by adding  $c_t$  to it. For all other  $a_{i'}$ , the vector remains unchanged.

This vector and its updates represent some stream of data that evolves with time. There are 2 main models for such a stream:

1. **cash-register case:**  $c_t > 0$ , so every vector element is monotonically increasing.
2. **turnstile case:**  $c_t$  can also be negative. There are two subcases:
  - (a) **non-negative turnstile:**  $(i_t, c_t)$  will never cause vector elements  $a_{i_t}$  to dip below zero. This guarantee can be provided by the application.
  - (b) **general turnstile:** vector elements  $a_{i_t}$  may become negative.

The basic problem this work addresses is to summarize or calculate certain characteristics about the stream. There are 2 main constraints. First, the space used by such algorithms should be small; at most polylogarithmic in  $n$ . This compressed version of the data is called a **sketch**. Second, updates to the sketch should be processed quickly. Since we are sublinear in input space, we will have to approximate almost any function we want to compute over  $\vec{a}$ , but we still want to specify an approximation parameter  $\varepsilon$  and bound the probability of error by  $\delta$ .

The **count-min sketch** addresses these requirements and allows us to calculate several characteristics of a stream using a single data structure. It can be used to approximate three types of **queries**:

1. **point query:**  $Q_i(i)$  is an approximation of the vector element  $a_i(t)$

2. **range query:**  $Q_i(l, r)$  is an approximation of  $\sum_{i=l}^r a_i$ .
3. **inner product query:**  $Q_i(\vec{a}, \vec{b})$  approximates  $\vec{a} \odot \vec{b} = \sum_{i=1}^n a_i b_i$

These queries have a variety of applications: point and range queries are used in summarizing the data distribution and the inner product query can be used to estimate the join size of relations.

In addition, these queries can be used for more complex data stream functions:

1.  **$\phi$ -quantiles:** The  $\phi$ -quantiles of the cardinality

$$||\vec{a}||_1 = \sum_{i=1}^n |a_i(t)| \quad (3)$$

are the values that split a multiset of integers  $I \in [1 \dots n]$  into  $\phi$  groups based on their rank  $R$ . For example, the median of a set of data is the 2-quantile.

To obtain the  $\phi$ -quantiles, the elements are sorted and split as follows. The  $k$ th quantile is the element(s) with rank  $R = k\phi||\vec{a}||_1$  for  $k = 0, \dots, 1/\phi$ . The  $\varepsilon$ -approximation for the  $\phi$ -quantiles accepts as the  $k$ -th quantile any integer with rank  $(k\phi - \varepsilon)||\vec{a}||_1 \leq R \leq (k\phi + \varepsilon)||\vec{a}||_1$  for a given  $\varepsilon < \phi$ .

2. **Heavy hitters:**

## 2 State of the art

Sketches have been commonly used to help analyze large streams of data. These sketches allow for the following types of queries:  $L_1$  and  $L_2$  norms, number of distinct items in a sequence, join size of relations, range sum queries, and more. Although these data structures have proved to be very powerful, there are a few key drawbacks that limit the effectiveness of these sketches. A few of the drawbacks are as follows:

1. Common sketches typically have a  $\Omega(\frac{1}{\epsilon^2})$  multiplicative factor. For common uses of  $\epsilon$ , like 0.1 or 0.01, this scaling can quickly become too expensive.
2. Many sketches take linear time (in the size of the sketch) to do a single update.
3. Many sketches require  $p$ -wise independent hash functions. This is not trivial, and is especially difficult in hardware applications.
4. Some sketches can only handle one particular query.
5. Many sketches use analysis that hides large constants.

The count-min sketch reduces these problems in the following ways:

1. CMS uses space proportional to  $\frac{1}{\epsilon}$
2. Update time is sublinear with respect to the size
3. CMS requires only pairwise independent hash functions

4. The sketch can answer several queries and has numerous applications
5. All constants are explicit and small.

In particular, notice that the space has been reduced from  $\frac{1}{\epsilon^2}$  to  $\frac{1}{\epsilon}$  using a CMS, and the time for an update has been reduced from  $\frac{1}{\epsilon^2}$  to 1.

### 3 How to make a count-min sketch

Note  $e$  is the base of the natural logarithm function,  $\ln$ . The count-min sketch is an array with  $w$  columns and  $d$  rows where:

$$w = \lceil e/\epsilon \rceil \quad (4)$$

$$d = \lceil \ln 1/\delta \rceil \quad (5)$$

for a given accuracy parameter  $\epsilon$  and probability guarantee  $\delta$ . We need  $d$  pair-wise independent hash functions

$$h_1 \dots h_d : \{1 \dots n\} \rightarrow \{1 \dots w\} \quad (6)$$

The sketch starts with every array entry being 0. The sketch is updated when the  $t$ th update  $\{a_{i_t}, c_t\}$  arrives as follows

$$\text{count}[j, h_j(i_t)] \leftarrow \text{count}[j, h_j(i_t)] + c_t \quad (7)$$

In other words, upon receiving an update  $(i_t, c_t)$ :

1. for each row  $1 \leq j \leq d$ 
  - (a) add  $c_t$  to the  $h_j(i)$ th column.

## 4 Approximate Point Queries using CM sketches

### 4.1 Non-Negative Case

#### 4.1.1 Algorithm

For the non-negative case (either the non-negative turnstile or cash-register case), the algorithm to estimate  $\hat{a}_i(t) = Q_i(i)$  is as follows:

$$Q_i(i) = \hat{a}_i = \min_j \text{count}[j, h_j(i)] \quad (8)$$

In other words for a given  $i_t = i$ ,  $\hat{a}_i = \text{Estimate\_Point\_Query}(\text{CM\_Sketch}(\vec{a}, t), i)$ :

```
Estimate_Point_Query ( CM_Sketch (  $\vec{a}$ , t ), i )
  make empty multiset S;
  for each row j,  $1 \leq j \leq d$ :
     $S = S \cup \text{count}[j, h_j(i)]$ 
  return min(S);
```

**Theorem 1:** The estimate  $\hat{a}_i$  has the following guarantees:

- (1)  $a_i \leq \hat{a}_i$
- (2) with probability at least  $1 - \delta$ ,

$$\hat{a}_i \leq a_i + \varepsilon \|\vec{a}\|_1 \quad (9)$$

#### 4.1.2 Proof of Theorem 1

**Define** an indicator variable  $I_{i,j,k}$  as follows:

$$I_{i,j,k} = \begin{cases} 1, & \text{if } h_j(i) = h_j(k) \text{ and } i \neq k \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

$I_{i,j,k}$  is 1 if  $i$  and  $k$ ,  $i \neq k$ , have a collision under hash function  $h_j$ .

Since, to begin with, we chose all  $h_j$  from a 2-universal hash family with range  $1 \leq h_j \leq d = \lceil \frac{e}{\varepsilon} \rceil$  we know that, for  $i \neq k$ ,

$$Pr(h_j(i) = h_j(k)) \leq \frac{1}{\text{range}(h_j)} = 1/\lceil \frac{e}{\varepsilon} \rceil = \lceil \frac{\varepsilon}{e} \rceil \quad (11)$$

by the properties of a 2-universal has function and by our choice of hash function range.

**Define**  $X_{i,j}$  as follows:

$$X_{i,j} = \sum_{k=1}^n I_{i,j,k} a_k \quad (12)$$

In other words,  $X_{i,j}$  is the sum of all corresponding vector values whose indices collide with the index of  $a_i$  under  $h_j$ . Trivially, we can then state that

$$\text{count}[j, h_j(i)] = a_i + X_{i,j} \quad (13)$$

This is clearly true as the column that  $i$  hashes to under  $h_j$  contains the real value  $a_i$  after update  $t$ , combined with all of the other  $a_k$ ,  $k \neq i$  after update  $t$  that hash to that same spot in that row  $j$  under  $h_j$ . Now, Theorem 1.a follows because we are only considering the case where vector elements are non-negative and thus

$$a_i \leq \hat{a}_i = a_i + X_{i,j} \quad (14)$$

for any possible  $X_{i,j}$

Now, to show 1.b, we evaluate:

$$E(X_{i,j}) = E\left(\sum_{k=1}^n I_{i,j,k} a_k\right) \quad (15)$$

by definition of  $X_{i,j}$ . The linearity of expectation states that

$$E(\sum_{i=1}^n c_i X_i) = \sum_{i=1}^n c_i E(X_i) \quad (16)$$

where  $X_i$ s may be dependent. Applying linearity of expectations here, we have:

$$E(\sum_{k=1}^n a_k I_{i,j,k}) = \sum_{k=1}^n a_k E(I_{i,j,k}) \quad (17)$$

We calculated  $E(I_{i,j,k}) \leq \lceil \frac{\varepsilon}{e} \rceil$  above, so we have

$$\sum_{k=1}^n a_k E(I_{i,j,k}) \leq \sum_{k=1}^n a_k \lceil \frac{\varepsilon}{e} \rceil \quad (18)$$

and finally, since  $\|\vec{a}\|_1 = \sum_{k=1}^n |a_k| = \sum_{k=1}^n a_k$  (all  $a_k$  are assumed to be non-negative)

$$\sum_{k=1}^n a_k \lceil \frac{\varepsilon}{e} \rceil = \lceil \frac{\varepsilon}{e} \rceil \|\vec{a}\|_1 \quad (19)$$

$$\implies E(X_{i,j}) \leq \lceil \frac{\varepsilon}{e} \rceil \|\vec{a}\|_1 \quad (20)$$

Now that we know that this is true in expectation, we use Markov to bound the tail bounds:

$$Pr[X \geq a] = \frac{1}{a} E(X) \quad (21)$$

Let  $X = \hat{a}_i$ , and choose  $a = a_i + \varepsilon \|\vec{a}\|_1$ . Then,

$$Pr[\hat{a}_i > a_i + \varepsilon \|\vec{a}\|_1] = Pr[\forall_j. count[j, h_j(i)] > a_i + \varepsilon \|\vec{a}\|_1] \quad (22)$$

since the probability that all values are greater than  $a_i + \varepsilon \|\vec{a}\|_1$  is equivalent to the probability that the minimum of all values is greater than  $a_i + \varepsilon \|\vec{a}\|_1$ , which is the definition of  $\hat{a}_i$ . Then, as stated in equation 13, we have

$$Pr[\forall_j. count[j, h_j(i)] > a_i + \varepsilon \|\vec{a}\|_1] = Pr[\forall_j. a_i + X_{i,j} > a_i + \varepsilon \|\vec{a}\|_1] \quad (23)$$

Then, subtracting  $a_i$  from both sides, and using equation 11 on the  $\varepsilon \|\vec{a}\|_1$  term, we have:

$$Pr[\forall_j. a_i + X_{i,j} > a_i + \varepsilon \|\vec{a}\|_1] = Pr[\forall_j. X_{i,j} > eE(X_{i,j})] \quad (24)$$

Finally, applying the bounds given by Markov's inequality, we have:

$$Pr[\forall_j. X_{i,j} > eE(X_{i,j})] \leq \frac{E(X_{i,j})}{a_i + \varepsilon \|\vec{a}\|_1} < e^{-d} < \delta \quad (25)$$

This estimate is calculated in  $O(\ln \frac{1}{\delta})$  as the minimum of a multiset can be taken in linear time. The space complexity is  $(2 + \frac{\varepsilon}{e}) \ln \frac{1}{\delta}$  words.

All previous analyses of sketch algorithm use Chebyshev in their estimation analysis, yielding a dependency on  $\frac{1}{\varepsilon^2}$  for the space complexity. Using Markov in this analysis yields a tighter bound, with a dependency  $\frac{1}{\varepsilon}$ .

Considering only the non-negative case here confers 2 advantages as we can take the minimum of the estimate, rather than the median. This allows us only one-sided error, and allows us to calculate the failure probability, instead of using Chernoff bounds, yielding smaller constants than prior work.

## 4.2 General Case

### 4.2.1 Algorithm

The algorithm is identical to the non-negative case, except that we take the median of the multi-set:

$$Q_i(i) = \hat{a}_i = \text{median}_j \text{count}[j, h_j(i)] \quad (26)$$

**Theorem 2:** With probability  $1 - \delta^{1/4}$ ,

$$a_i - 3\varepsilon \|\vec{a}\|_1 \leq \hat{a}_i \leq a_i + 3\varepsilon \|\vec{a}\|_1. \quad (27)$$

The time and space complexity are the same as the non-negative case:  $O(\ln \frac{1}{\delta})$  time and  $(2 + \frac{\varepsilon}{\delta}) \ln \frac{1}{\delta}$  words, respectively.

## 5 Inner Product Query

### 5.1 Algorithm

The estimate the inner product  $Q(a, b)$  for non-negative vectors  $a$  and  $b$  is  $\widehat{a \cdot b} = \min_j (\widehat{a \cdot b})_j$ , where  $(\widehat{a \cdot b})_j = \sum_{k=1}^w \text{count}_a[j, k] * \text{count}_b[j, k]$ . The algorithm to compute the inner product follows from point queries. **Theorem 3:** The inner product  $a \cdot b$  is less than or equal to the estimated inner product such that,

$$a \cdot b \leq \widehat{a \cdot b} \quad (28)$$

$$\Pr(\widehat{a \cdot b} \leq a \cdot b + \varepsilon \|a\|_1 \|b\|_1) = 1 - \delta \quad (29)$$

### 5.2 Proof of Theorem 3

We represent the estimated inner product as the sum of the true inner product and collisions. By the pairwise independence of  $h$ ,

$$(\widehat{a \cdot b})_j = \sum_{i=1}^n a_i b_i + \sum_{p \neq q, h_j(p)=h_j(q)} a_p b_q \quad (30)$$

$$E[(\widehat{a \cdot b})_j - a \cdot b] = E \left[ \sum_{p \neq q, h_j(p)=h_j(q)} a_p b_q \right] \quad (31)$$

$$= \Pr(h_j(p) = h_j(q)) a_p b_q \quad (32)$$

$$\leq \sum_{p \neq q} \frac{\varepsilon}{e} a_p b_q \quad (33)$$

$$\leq \frac{\varepsilon}{e} \|a\|_1 \|b\|_1 \quad (34)$$

Where this follows by the reduction of the inner product computation as a series of point queries as outlined above. Applying the markov inequality,

$$Pr(\widehat{a \cdot b} - a \cdot b > \epsilon \|a\|_1 \|b\|_1) \leq \frac{E[\widehat{a \cdot b} - a \cdot b]}{\|a\|_1 \|b\|_1} \quad (35)$$

$$\leq \frac{\frac{\epsilon}{e} \|a\|_1 \|b\|_1}{\epsilon \|a\|_1 \|b\|_1} \quad (36)$$

$$\leq 1/e \quad (37)$$

$$\leq \delta \quad (38)$$

## 6 Range Query

A range query, denoted as:  $\mathcal{Q}(l, r)$ , requests the value:

$$a[l, r] = \sum_{i=l}^r a_i$$

And we denote our estimator based on the CM sketch as:  $\hat{a}[l, r]$ .

### Procedure for estimation:

Keep  $\log_2 n$  CM sketches. A single range query can be converted into  $2 \log_2 n$  dyadic range queries, and each dyadic range query can be converted into a single point query. A CM sketch is kept for every dyadic range, and we do a single point query for every dyadic range. The sum of all of the point queries is the result,  $\hat{a}[l, r]$ .

TODO better explain dyadic range stuff? ASK LIN

**Theorem 4:**  $a[l, r] \leq \hat{a}[l, r]$ . With probability at least  $(1 - \delta)$ ,  $\hat{a}[l, r] \leq a[l, r] + 2\epsilon \log n \|a\|_1$ .

### Proof:

TODO explain why we have  $2 \log_2 n$  dyadic ranges, which are equivalently a point query for each. Then:

This proof is a simple extension of the point query proof. Because a single range query can be transformed to  $2 \log_2 n$  point queries, we can extend the error bounds of many point queries. Recall that a single point query errors by at most  $\epsilon \|a\|_1$ , with probability  $\delta$ . Because each of these point queries that we used are taken from a separate CM sketch, they are all independent. Thus, we can simply add the errors together, and therefore, we error by at most  $2 \log_2 n (\epsilon \|a\|_1)$ , and with the same probability,  $\delta$ .

## 7 Applications

### 7.1 Quantiles in the turnstile model

The count-min sketch can be used to calculate the  $\phi$ -quantiles in the turnstile model. Prior work (21) shows that  $\phi$ -quantiles can be approximated using range sums. This is done as follows:

1. For each  $k \in 1, 2, \dots, 1/\phi$ 
  - (a) Find a range sum such that  $a[1, r] = k\phi\|\vec{a}\|_1$ . Find  $r$  using a binary search on the possible range sums  $a[1, \hat{r}]$ , where  $\hat{r} \in \{1, 2, \dots, n\}$ .  $r$  is the  $\varepsilon$ -approximation for the  $k$ th  $\phi$ -quantile.

The method in [21] uses Random Subset Sums to approximate range sums. If instead a count-min sketch is used to approximate the range sums, better results follow.

To use a count-min sketch to approximate quantiles in the turnstile model,  $\log n$  sketches are kept, one for each dyadic range. Each sketch gets accuracy parameter  $\varepsilon/\log n$  for an overall accuracy bounded by  $\varepsilon$ . Each sketch gets probability guarantee  $\delta\phi/\log n$ , for an overall probability guarantee of  $\delta$  for all  $1/\phi$  quantiles. Then, we have the following:

**Theorem 5:**  $\varepsilon$ -approximate  $\phi$ -quantiles can be found with probability at least  $1 - \delta$  by keeping a data structure with space  $O\left(\frac{1}{\varepsilon} \log^2(n) \log\left(\frac{\log n}{\phi\delta}\right)\right)$ . The time for each insert or delete operation is  $O\left(\log(n) \log\left(\frac{\log n}{\phi\delta}\right)\right)$ , and the time to find each quantile on demand is  $O\left(\log(n) \log\left(\frac{\log n}{\phi\delta}\right)\right)$ .

This improves the existing query time and update by a factor of more than  $\frac{34}{\varepsilon^2} \log n$ . The space requirements are improved by a factor of at least  $\frac{34}{\varepsilon}$ .

## 7.2 Heavy Hitters

A potential application of the count min sketch is the heavy hitters problem. Given an input  $A$  of length  $n$  and a parameter  $k$  we wish to return values that occur in the input at least  $n/k$  times. There may be at most  $k$  such heavy hitters, though there also may be none. This sketch provides a convenient way to find the heavy hitters of some data stream with a large input size such that maintaining a counter for each unique element is not feasible. We look at two cases:

1. The cash register case: elements are only added
2. The turnstile case: elements can be added or removed

### 7.2.1 Cash Register Case

Recall that the cash register case is for positive updates only. We can easily obtain  $\|\mathbf{a}\|$  at any given point in time because it is simply:  $\sum_{i=1}^t c_i$ . We define a  $\phi$  heavy hitter if the estimation of the point query,  $Q(i_t) = \hat{a}_{it} \geq \phi\|\mathbf{a}(t)\|$ . We can do this by maintaining a heap to store the items above the  $\phi\|\mathbf{a}(t)\|$  threshold. On any update, we check the lowest value in the heap, and if the update would be greater than the lowest item, we replace it in the heap. When we are finished with the stream, we do a final scan of all items in the heap, and return the ones which have a value over  $\phi\|\mathbf{a}\|_1$ .

**Theorem 6:** We can identify the heavy hitters of a sequence of length  $\|\mathbf{a}_i\|_1$  using space  $O\left(\frac{1}{\varepsilon} \log\left(\frac{\|\mathbf{a}\|}{\delta}\right)\right)$  and time  $O\left(\log\left(\frac{\|\mathbf{a}\|}{\delta}\right)\right)$ . Every item which occurs with count  $\phi\|\mathbf{a}\|_1$  is output, and with probability  $(1 - \delta)$ , no items whose count is less than  $(\phi - \varepsilon)\|\mathbf{a}\|_1$  is output.



**Proof idea:**

Because we have only positive updates, and a CM sketch will only over estimate, it is not possible to miss any heavy hitter. Thus, we will never omit any heavy hitters. The bound on the error of outputting a non heavy hitter comes from the fact that a single point query outputs the estimate count of an item within  $\epsilon$  of the actual value, with probability  $(1 - \delta)$ . Because the heavy hitter relies purely on the point query, the error bound is the same, and thus, is as above.

**7.2.2 Turnstile Case**

The method outlined by Graham Cormode et. al.<sup>1</sup> is adapted using the count min sketch. In this scheme, a sketch is maintained for each of the  $\log n$  dyadic ranges. When an element is inserted or removed, a binary search is performed while updating the counter at each level of the hierarchy. To find the heavy hitters, a parallel binary search is performed recursively searching each range for which the counter weight exceeds  $(\phi + \epsilon) \|a\|_1$ . Since at each level there are at most  $1/\phi$  items with frequency greater than  $\phi$ , the probability of failure for each sketch is set at  $\delta\phi/(2 \log n)$  to limit the number of output items whose true frequency is less than  $\phi$ . By doing this the total probability that there is any overestimated by more than  $\epsilon$  is bounded by  $\delta$  by the union bound.

Since the sketch at minimum reports the true count, we are guaranteed that the true heavy hitters, if they exist, are reported. With probability  $1 - \delta$ , the algorithm will report a false heavy hitter.

**Theorem 7:**

The algorithm uses space  $O((1/\epsilon) \log n \log(2 \log n / (\delta\phi)))$  and time  $O(\log n \times \log(2 \log n / (\delta\phi)))$  per update. Every item with frequency at least  $(\phi + \epsilon) \|a\|_1$  is reported, and with probability  $1 - \delta$  no item whose frequency is less than  $\phi \|a\|_1$  is output.

---

<sup>1</sup>Graham Cormode and S. Muthukrishnan. 2005. What's hot and what's not: tracking most frequent items dynamically. ACM Trans. Database Syst. 30, 1 (March 2005), 249-278. DOI=<http://dx.doi.org/10.1145/1061318.106132>