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 5, 2016

1 Problem Statement and Overview

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

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

The update (i_t, c_t) modifies the ith 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 we address is to summarize or calculate certain characteristics about the stream. We have 2 main requirements. 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, we want to be able to process updates to the sketch 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 ε and bound the probability of error by δ.

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. **ϕ-quantiles:** The ϕ-quantiles of the cardinality

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

   are the values that split a multiset of integers $\in [1 \dots n]$ into ϕ groups based on their rank R. For example, the median of a set of data is the 2-quantile. The elements are sorted and split as follows. The kth quantile is the element(s) with rank $R = k\phi||\vec{a}||_1$ for $k = 0, \dots, 1/\phi$. The ε-approximation for the ϕ-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 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:

$$w = \lceil e/\varepsilon \rceil \tag{4}$$

$$d = \lceil \ln 1/\delta \rceil \tag{5}$$

for a given accuracy parameter ε and probability guarantee δ. We need d pair-wise independent hash functions

$$h_1 \dots h_d : \{1 \dots n\} \to \{1 \dots w\} \tag{6}$$

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

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

In other words, when you get 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.

3 Approximate Point Queries using CM sketches

3.1 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:

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

In other words for a given $i_t = i$, $\hat{a}_i =$ `Estimate_Point_Query(CM_Sketch(`$\vec{a}, t$`), i )`:

```
Estimate_Point_Query(CM_Sketch(a, t), i)
    make empty multiset S;
    for each row j, 1 ≤ j ≤ d:
        S = S ∪ 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 $1 - \delta$,

$$\hat{a}_i \leq a_i + \varepsilon ||\vec{a}||_1 \tag{9}$$

3.2 Proof of Theorem 1

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

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

In other words, $I_{i,j,k}$ 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 $i \neq k$,

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

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

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

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

In other words, it 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

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

Here, we're just saying that 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. This is just how the data structure is made. Now, 1.a follows because we are only considering the case where vector elements are non-negative and thus

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

for any possible $X_{i,j}$

Now, to show 1.b, we evaluate:

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

by definition of $X_{i,j}$. The linearity of expectation gives us

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

where X_is may be dependent. Applying this here, we have:

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

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

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

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)

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

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

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

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

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

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

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

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

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

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

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

$$Pr[\forall_j.X_{i,j} > eE(X_{i,j})] \leq \frac{E(X_{i,j})}{a_i + \varepsilon||a||_1} \tag{25}$$

4 Applications

4.1 Quantiles in the turnstile model

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

1. For each $k \in 1, 2, \ldots, 1/\phi$
   (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 $\hat{r} \in \{1, 2, \ldots, n\}$. r is the ε-approximation for the kth ϕ-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 ε. Each sketch gets probability guarantee $\delta\phi/\log n$, for an overall probability guarantee of δ for all $1/\phi$ quantiles. Then, we have the following:

Theorem 5: ε-approximate ϕ-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 $\frac{34}{\varepsilon}$.