Comparing Count Sketches [1,2] and Count Min Sketches [3]

1 Introduction

In the world of today, there is a lot of information we can go through, but might not have enough to store it all as is for future calculations. For example, we might want to find the frequency of some item in a long stream, or perhaps the most frequent items.

There are several algorithms that keep a "sketch" of the stream in order to answer queries in quick using the sketch, which is sublinear in space compared to the stream.

One of the algorithms is called "Counting Sketches", published by M. Charikar, K. Chen, and M. Farach-Colton. We've studied and analyzed the space complexity, query complexity and error bounds for this algorithm in class.

A second algorithm published by G. Cormode, S. Muthukrishnan is called "Count-Min Sketches" in which we are also able to create a sublinear sketch of the stream, and approximate the count of items quickly.

In this work I'll compare the Count Sketches algorithm Count Min Sketch, including the space and time required to create the sketch and answer queries, and the time required to approximate the k most frequent items.

2 The Count-Min Sketches Algorithm

Given a stream $S_t = < a_1, a_2, a_3 \dots a_t >$ of objects $o_1, o_2 \dots o_n$, each appearing $f(o_1), f(o_2), \dots f(o_n)$ times, the Count-Min Sketch data structure allows us to approximate $f(a)$, the number of occurrences of the element a at time t for any element a.

The sketch requires $O(\left(\frac{e}{\varepsilon}\right) \ln \left(\frac{1}{\delta}\right))$ space and lets us calculate $\hat{f}(a)$ such that:

$$f(a) \leq \hat{f}(a) \leq f(a) + \varepsilon t$$

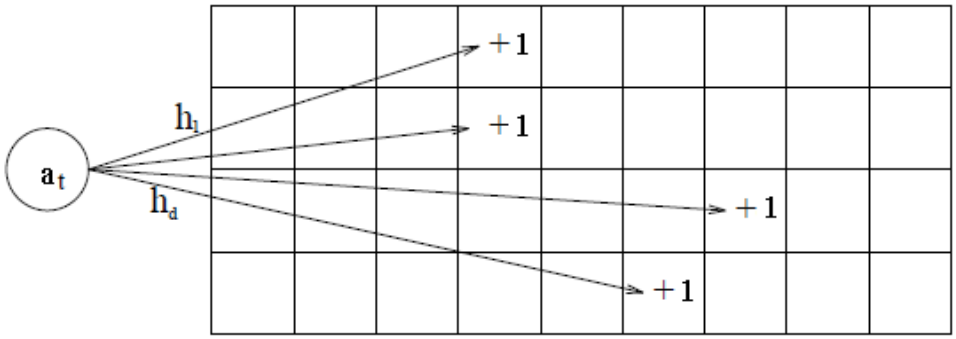
The Count-Min Sketch data structure is an $d * x$ matrix (referred to as count) where $w = \left\lceil \frac{e}{\varepsilon} \right\rceil$ and $d = \left\lceil ln(\frac{1}{\delta}) \right\rceil$.

The structure also includes d hash functions:

$$h1, h2, \dots h_d : \{1..n\} \rightarrow \{1..w\}$$

Where n is the number of different objects in our stream.

Each hash function is associated with a row in the matrix, and all hash functions are pair-wise independent.



In order to propagate the data structure, we'll execute the update procedure for each element in the stream as we encounter it.

The Update Procedure:

When we encounter an object a, we'll increment the counter in each row by 1. The counter is the determined by h_i.

Formally: $For\ each\ i \in [1..d]: count[i, h_i] = count[i, h_i] + 1$

Approximate Query:

Clearly for each i we immediately have $f(a) \leq count[i, h_i(a)]$. However, the bound may be poor. To get a better estimator, we'll take the minimum over all the rows in count.

The query function will therefore be:

$$\hat{f}(a) = \min_j count[j, h_j(a)].$$

We'll now prove a bound on the error. In order to make the proof as similar as possible to the one we gave for Counting Sketches in class, the proof is a little different than the one given by Cormode and Muthukrishnan [3].

First define the "inverse" of h_i. Let $h_i^{-1}(a)$ to be the set of all elements that h_i sets in the same place as a (this is similar to the inverse h we defined in class for Counting Sketches).

Formally:

$$h_i^{-1}(a) = \{b | h_i(a) = h_i(b)\}$$

We'll denote

$$\hat{f}_i(a) = count[i][h_i(a)] = \sum_{b \in h_i^{-1}(a)} f(b) =$$

$$f(a) + \sum_{b \in h_i^{-1}(a) \setminus \{a\}} f(b)$$

The expected difference between the i'th estimator for $a - \hat{f}_i(a)$ and $f(a)$ itself can be bounded:

$$E\left[\hat{f}_i(a) - f(a)\right] = E\left[\sum\nolimits_{b \in h_i^{-1}(a) \setminus \{a\}} f(b)\right] =$$

$$\frac{1}{w}\sum_{b \neq a} f(b) = \frac{\varepsilon}{e}\sum_{b \neq a} f(b)$$

$$\leq \frac{\varepsilon}{e}t = \frac{\varepsilon}{e}\left|\left|f\right|\right|_1$$

We get this from the assumption that h_i is a perfect hash function, and therefore the probability for a collision is $\frac{1}{w}$ which is set to $\frac{\varepsilon}{e}$.

Using the Markov inequality:

$$Pr[\hat{f}_i(a) - f(a) \geq \varepsilon t] = \Pr\left[\hat{f}_i(a) - f(a) \geq \frac{\varepsilon t}{e} e\right] \leq \frac{1}{e}$$

From the pair-wise independence of the hash functions, we get:

$$Pr\left[\hat{f}(a) - f(a) \geq \varepsilon t\right] \leq \frac{1}{e^d} = \frac{1}{e^{\ln\left(\frac{1}{\delta}\right)}} = \delta$$

Therefore,

$$Pr\left[\hat{f}(a) \leq f(a) + \varepsilon t\right] \geq 1 - \delta$$

Finding the frequent items:

Similarly to what we've done in class, the algorithm for finding the most frequent items is to go over the stream, and update a Count-Min Sketch with all the elements seen so far. When we process (update) an element, we also approximate its frequency (using the query method above) and keep the top k most frequent items.

3 Analysis of Count Sketches

We'll now give a quick analysis of the Count Sketches algorithm. This analysis is similar to the one presented in class but we'll bound the error chance for each point query. This will allow us to better compare the two algorithms later on.

We'll start by analyzing the error for a fixed item a. We'll take a look at a's counter in row i, that is count[i][$h_i(a)$], and analyze $X = f(a) - C[i][h_i(a)] \cdot s_i(a)$ (i.e. X denotes the error of the approximation in row i). For $o_j ! = a$, denote Y_j the contribution of o_j to the error. Note that this happens only when $h_i(a) = h_i(o_j)$, with probability $\frac{1}{b}$. In such situation, it contributes $f(o_j) \cdot s_i(o_j)$ to X.

$$Y_j = \begin{cases} f(o_j) & w.p \frac{1}{2b} \\ -f(o_j) & w.p \frac{1}{2b} \\ 0 & w.p\ 1 - \frac{1}{b} \end{cases}$$

Therefore $E[Y_j] = 0$.

$$Var[Y_j] = E[Y_j^2] - E^2[Y_j] = \frac{f(o_j)^2}{b}$$

We'll assume WLOG that $a = o_1$. The error is then the sum of all other Y_j's:

$$X = Y_2 + Y_3 + \cdots + Y_n$$

All Y_j's are pair-wise independent because the hash functions are pair-wise independent:

$$Var[X] = Var[Y_2] + \cdots + Var[Y_n] =$$

$$\frac{f(o_2)^2}{b} + \cdots + \frac{f(o_n)^2}{b} = \frac{1}{b}\sum_{j=2}^{n} f(o_j)^2 = ||f||_2^2 - f(o_1)$$

Now, using Chebychev's inequality:

$$\Pr\left[|X| \geq \varepsilon\sqrt{\left(||f||_2^2 - f(o_1)\right)}\right] \leq \frac{1}{\varepsilon^2 b}$$

We'll demand $\frac{1}{\varepsilon^2 b} \leq \frac{1}{e} \rightarrow b = O(\frac{e}{\varepsilon^2})$

Now with taking the median of $s_i(o)count[i][h_i(a)]\ for\ i = 1\ to\ t$, and using Chernoff's inequality we get

$$\left|\hat{f}(a) - f(a)\right| \leq \varepsilon\sqrt{\left(||f||_2^2 - f(o_1)\right)} \leq \varepsilon||f||_2$$

With probability $1 - \delta$ for $t = O(\ln\left(\frac{1}{\delta}\right))$.

4 Comparing Count Sketches and Count Min Sketches

We'll now compare the two algorithms, by means of space required by their data structures, the running time of the update operation and the running time of the query operation.

Count-Min:

Space: The space required by the Count-Min Sketches data structure is $O(dw + d)$, because we keep a count matrix sized $d \cdot w$ and d hash functions (each of them can be stored in $O(1)$ space).

Overall, the sketch requires $O(dw) = O\left(\left(\frac{e}{\varepsilon}\right) \cdot \ln\left(\frac{1}{\delta}\right)\right)$ space.

Update: The update time for each element is $O(d) = O(\ln\left(\frac{1}{\delta}\right))$

Query: The minimum of d elements can be found in linear time, and therefore the query time is $O(d) = O\left(\frac{1}{\delta}\right)$

Error measurement: The error is measured as a ratio to the norm 1 of the stream's frequencies, in our case, the number of elements in the stream.

Count-Sketches:

Space: The space required by the Count Sketches algorithm we've seen in class is $O(bt + 2t)$, because we keep a count matrix sized $b \cdot t$ and 2 hash functions per row.

Overall, the sketch requires $(bt) = O((\frac{e}{\varepsilon^2}) \cdot \ln\left(\frac{1}{\delta}\right))$ space.

Update: The update time for each element is $O(t) = O(\ln\left(\frac{1}{\delta}\right))$.

Query: Finding the median of t elements can be done in linear time using some selection algorithm, and therefore the query time is $O(t) = O(\ln(\frac{1}{\delta}))$.

Error measurement: The error is measured as a ratio to the norm 2 of f.

Norm 2 of a vector is in general bounded by the first norm. From this perspective Counting Sketches might give a better guarantee. This is especially true when the items' frequencies are evenly distributed.

The following table sums up some of the important differences of the algorithms:

Algorithm	Count-Min Sketches	Count Sketches (from class)	Comments
Data Structure Space	$O\left(\left(\frac{e}{\varepsilon}\right) \cdot \ln\left(\frac{1}{\delta}\right)\right)$	$O(\left(\frac{e}{\varepsilon^2}\right) \cdot \ln\left(\frac{1}{\delta}\right))$	Count-Min requires space that is proportional to $\frac{1}{\varepsilon}$ while Count-Sketches requires space that is proportional to $\frac{1}{\varepsilon^2}$
Time per update	$O(\ln\left(\frac{1}{\delta}\right))$	$O(\ln\left(\frac{1}{\delta}\right))$	Update and query time are similar in both algorithms
Time per query	$O(\ln\left(\frac{1}{\delta}\right))$	$O(\ln\left(\frac{1}{\delta}\right))$	
Epsilon error is a ratio of	$\lvert\lvert f \rvert\rvert_1$ – norm-1 of the frequency vector.	$\lvert\lvert f \rvert\rvert_2$ - norm-2 of the frequency vector.	When setting same value for ε in both algorithms, the Count Sketches gives a better error guarantee.
Hash function independence	Pairwise	Pairwise	

5 Some “real world” results

In order to compare the two algorithms on the “battlefield”, we’ve implemented both of them in Java, using very similar data structures and objects (source code in Java 1.6 is attached).

We’ve used the bible as our dataset (King James Bible – free version available at http://printkjv.ifbweb.com/).

As mentioned before, the algorithms differ in their error measurement, which comparing the results difficult. In order to make the comparison more applicable, we’ve decided to use the error bound from both algorithms as a word count.

This was computed as follows:

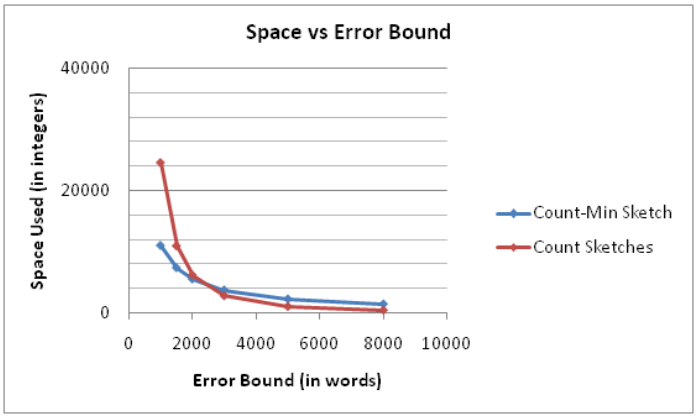
- Count how many times each word appears and put the results into a vector – f.
- Compute norm-1 of the vector.
- Compute norm-2 of the vector.
- Compute ε for the Count-Min Sketch algorithm using:

$$\varepsilon \cdot ||f||_1 \leq WordCountError$$

- Compute ε for the Count-Min Sketch algorithm using:

$$\varepsilon \cdot ||f||_2 \leq WordCountError$$

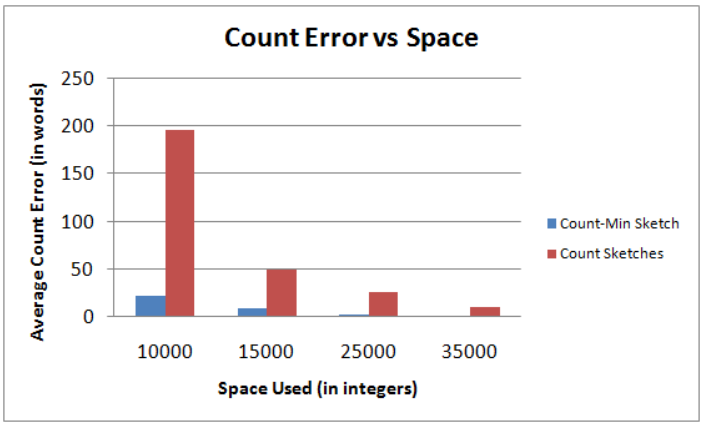
When using $\delta = 0.01$, these were the results:



It's possible to see that as the word count error bound decreases, the Count-Min Sketch algorithm performs better. This is due to the $O(\frac{1}{\varepsilon^2})$ space required in the Count Sketches algorithms versus the $O(\frac{1}{\varepsilon})$ in Count-Min sketch while the norm values stays the same (the dataset stays the same). This matches our expectations.

Timing the results gives similar results between the two algorithms, and only depends on the δ parameter as expected (Since both update and query operations depend only on δ.
It's possible to see the Count Sketches algorithm is a little bit slower in practice, probably because it runs twice as many hash calculations than the Count-Min algorithm.

Another interesting thing to look at is the algorithms error count in practice, when we limit their space usage. In the following graph we can see the average error count (in words) of each of the algorithms, when they are limited to some space (in all cases δ=0.01).



We can see that using this dataset (King James Bible), the Count-Min sketch algorithm gives better average error than the Count Sketches when using the space bound.

References

[1] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.

[2] Edo Liberty. Lecture Notes, Data Mining 2012, Tel Aviv University.

[3] G. Cormode, S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and its Applications.