Count-Min Sketch and its Variants

Potapov Yurii

NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

1 Abstract

- Various Counting Sketches are fast, memory-efficient probabilistic data structures, which used in a variety of applications where the input data stream is huge.
- Count-min sketches are much more memory efficient than unordered map or map(dictionary) because they store only the counters, not the elements themselves, and do it very efficiently. This useful property can be used in Compressed Sensing, Networking, Databases, NLP, Security and others areas.
- The accuracy of such sketches is significantly improved by an improvement that we will call a conservative update strategy. This strategy is very effective when the goal is to emphasize a relatively small number of the most frequently occurring keys in stream .
- In this work, we will model the behavior of the Count-Min and Conservative Count-Min on data of various distributions such as normal distribution, uniform distribution and Zipf distribution.
- We will most fully consider the error of sketches on a distribution of the Zipf type (or Pareto Principle, also Power Law) that have such a property that a small number of the most frequent elements represent a large part of the flow.
- We will calculate the absolute and relative error for the sketches using data obtained from the mersenne twister engine (fast random number generator).
- The main goal of the work is to show that for the count-min and conservative count min zipf type distributions, min has a small error on frequent elements, and has a large error on rare elements.

Contents

2 Basic terms and definitions

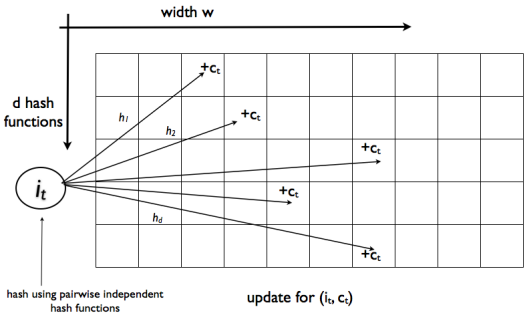
- The Count-Min (CM) sketch is a compact data structure capable of store elements (their number), with strong guarantees of accuracy. Such requirements are often needed in various areas, because it is needlessly to store all the elements everywhere. Since the data structure can easily handle updates in the form of an increase in the amount of this element, or its decrease. The CM is capable of handling high speed update streams. It is important that this is a probabilistic data structure that uses hash functions, some of which are very fast. Increasing the hit range of hash functions reduce the chance of a big error. Increasing the number of hash functions reduces average error. Therefore, to work with this data structure, it is very important to determine the size of table and the number of hashes.
- Conservative Count-Min Sketch (CCM) – is a modification of CM leading to a better accuracy, but not supporting negative updates.
- The Zipf distribution, sometimes referred to as the zeta distribution, is a discrete distribution commonly used in linguistics, insurance, and the modelling of rare events. It has probability density function is:

$$f(x) = \frac{x^{-(\rho+1)}}{\zeta(\rho+1)}$$

- Mersenne Twister is a fast pseudorandom number generator.

3 Introduction

- In General, CM Sketch – is a probabilistic data structure that can answer quesion like: "How much times there was element X in stream?". CM uses hash functions to connect events to frequencies. But not like a hash table, it does not store any value. Also, the size of the structure depends on the desired precision. The CM sketch was invented in 2003 by Graham Cormode and S. Muthu Muthukrishna and described by them in a 2005 paper.
- Very simple picture for understanding:



.

- There are currently a couple of articles that answer similar question, but they are all new. We will compare our results with the article dated 28.03.2022
- In this work we want to understand how CM and CCM version work on different data streams. Such as uniform distribution, Gaussian distribution, Zipfian distribution. Also, we want experimentally study their performance.
- The result maybe useful for industry. For example, Google uses Count-Sketch to estimate the largest k coordinates of x for their "top table". Because many datasets Google encounters (for example, the frequency of URLs on the web) are distributed as power laws.

4 Implementation

- Originally, A CM sketch is an $R \times B$ matrix A of counters where each row i is associated with a hash function $h_i(\cdot)$. Initially all entries are initialized to zero. To store counters and support their dynamic updates given in a stream, CM works as follows. To process an update (p, ℓ), we perform

$$A\left(i, h_i(p)\right) = A\left(i, h_i(p)\right) + \ell \text{ for each } 1 \leq i \leq R.$$

- If updates are only positive, there exists a modification of CM leading to a better accuracy – CCM. Under this modification, updates for each row i are made according to:

$$A\left(i, h_i(p)\right) = \max\left\{A\left(i, h_i(p)\right), \min_{1 \leq i \leq R}\left\{A\left(i, h_i(p)\right)\right\} + \ell\right\}.$$

- We will consider such a variant of the CM sketch, when all the rows of the matrix are connected into one row. Therefore, the only parameter that specifies the CM sketch is the length of the one-dimensional matrix row. Updates are made according to:

$$A\left(h_i(p)\right) = A\left(h_i(p)\right) + \ell \text{ for each } 1 \leq i \leq R.$$

- Updates for CCM, analogically:

$$A\left(h_i(p)\right) = \max\left\{A\left(h_i(p)\right), \min_{1 \leq i \leq R}\left\{A\left(h_i(p)\right)\right\} + \ell\right\}.$$

- All data is generated randomly, so it is possible that for a some cases the error for a particular element will be extremly large. Therefore, we need the average value of the error, so experiments will run several times. On the graphs below, there is a parameter indicated as iters - this is exactly what shows how many different identical experiments were.
- The (approximate) current counter $\hat{\boldsymbol{a}}_{CM(CCM)}(p)$ associated with a key p is retrieved as

$$\hat{\boldsymbol{a}}_{CM(CCM)}(p) = \min_{1 \leq i \leq R}\left\{A(h_i(p))\right\}.$$

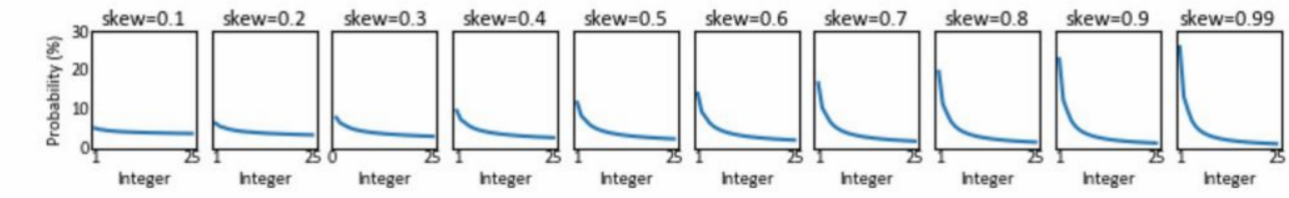
- Basically the elements of the stream will be strings of the same length.
- To consider error, it is enough to make updates like **p(l, 1), i.e. use the counting framework**.
- We will take the implementation of the Gaussian distribution and the uniform distribution from the standard library of the C ++ language
- We will take the implementation of the Zipfian distribution from open source. Clicable link
- Github repository with implementation

5 Main part

- All basic and useful theory with math explanation: article. Also article with experiments "A Formal Analysis of the Count-Min Sketch with Conservative Updates" by Younes Ben Mazziane, Sara Alouf, Giovanni Neglia.
- Error counting method: for each element in the stream, we will calculate relative error and absolte error -

$$\frac{Occurrence\,in\,CM\ -\ True\,Occurrence}{True\,Occurrence} \quad and \quad (Occurrence\,in\,CM\ -\ True\,Occurrence)$$

  Thus, we will get the relative error value for one element.
- We will draw the real occurrence of the element(True Count) from the dictionary(used C++ Unordered Map for this).
- For our purposes, three hash functions will be enough for us. These are three murmur hashes with different seeds (generated randomly on every iteration).
- It is completely clear, that relative/absolute error of CM-sketch will be bigger that the CCM-sketch, so **red line means CM-sketch and blue line means CCM-sketch**.
- The main idea of generating Zipf is make weights for elemenents and use standart library uniform distribution.
  Zipf's "Law": Integers between $1...\mathbf{N}$. Integer k gets weight proportional to $\left(\frac{1}{k}\right)^{\theta}$ where $0 < \theta < 1$ is the skew
- In the plots below we generate random integers in the range $[1, 25]$ (X axis) and plot the percentage of times they were generated (Y axis) with different skew factors. Note that the Zipfian distribution is less skewed than the self-similar one, as the percentages on the Y axis are lower.



5.1 Uniform Distribution

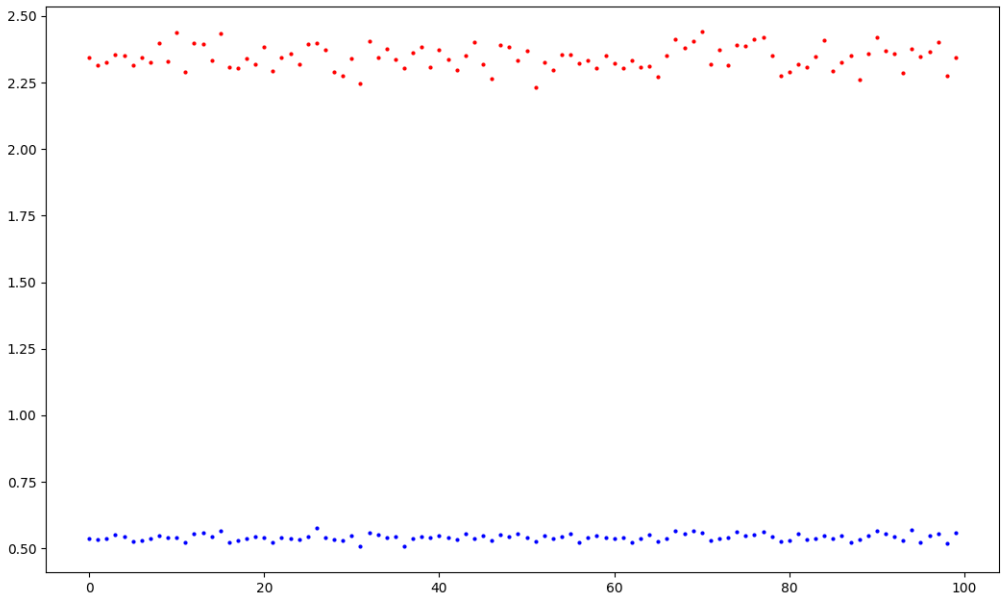


Figure 1: Absolute Error: table size = 300, size stream = 1000, different elements = 100, iters = 100. Red line CM, Blue line CCM

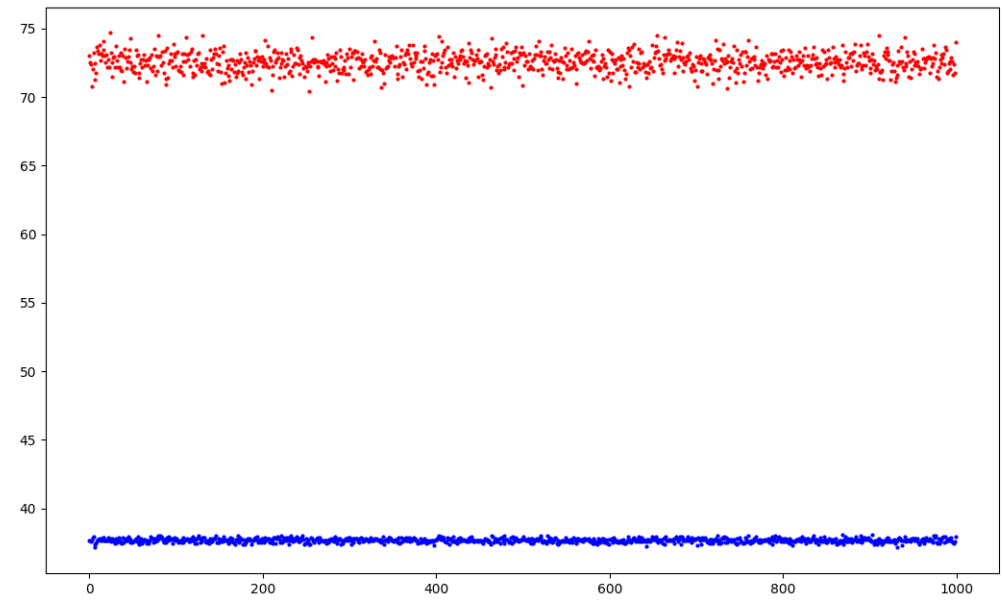


Figure 2: Absolute Error: table size = 300, size stream = 10000, different elements = 1000, iters = 100. Red line CM, Blue line CCM

▲ So we can see that error function behaves like a line. It is also clear that there is a direct correlation between the size of the stream and the size of the table, the larger table provides us smaller error.

5.2 Gauss Distribution

• Mean equal (size of different elements) / 2 and $\sigma = 35$

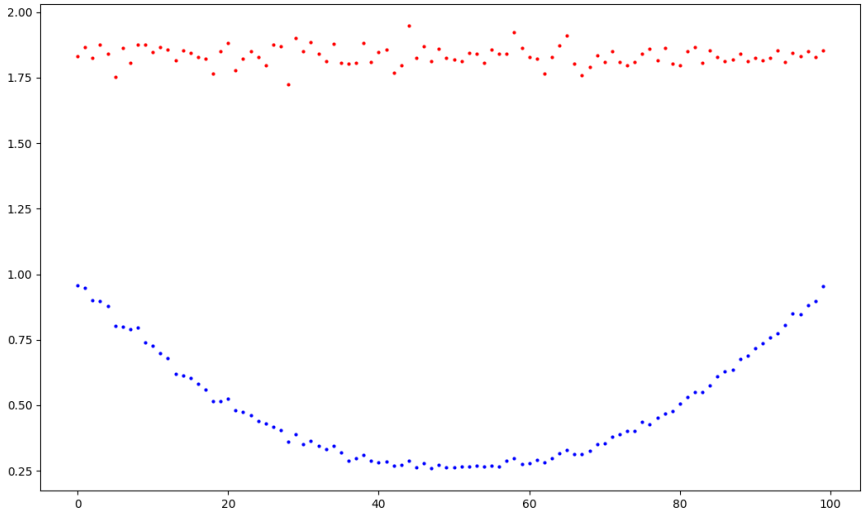


Figure 3: Absolute error: table size = 300, size stream = 1000, different elements = 100, iters = 100, Red line CM, Blue line CCM

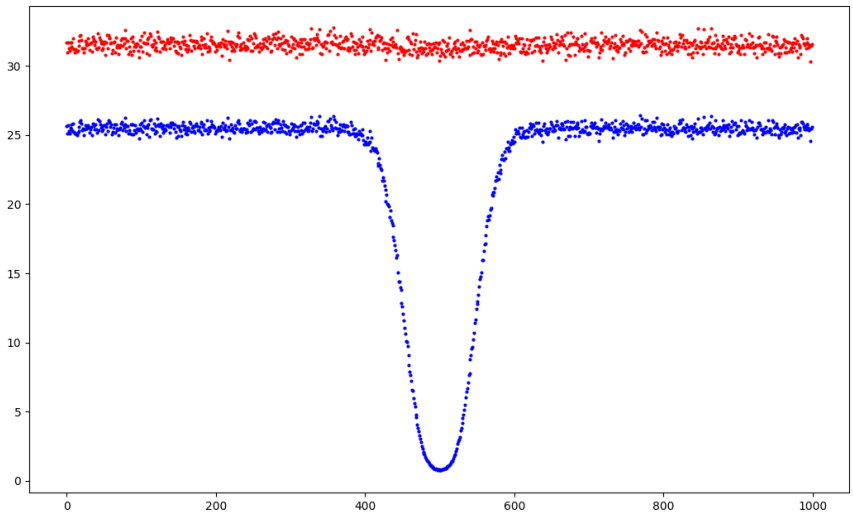


Figure 4: Absolute error: table size = 300, size stream = 10000, different elements = 1000, iters = 100, Red line CM, BLue line CCM

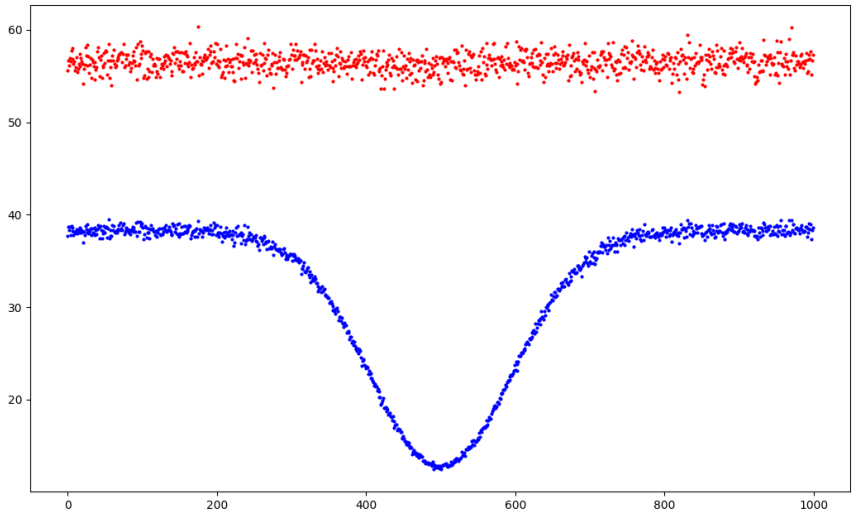


Figure 5: Absolute Error: table size $=$ 300, size stream $=$ 10000, different elements $=$ 1000, iters $=$ 100, $\sigma = 100$

▲ All graphs are quite clear - as mentioned earlier, the most frequent elements will have a smaller error than the rare ones. Also, we could consider only one half of the graph, the results will be the same.

5.3 Zipf Distribution

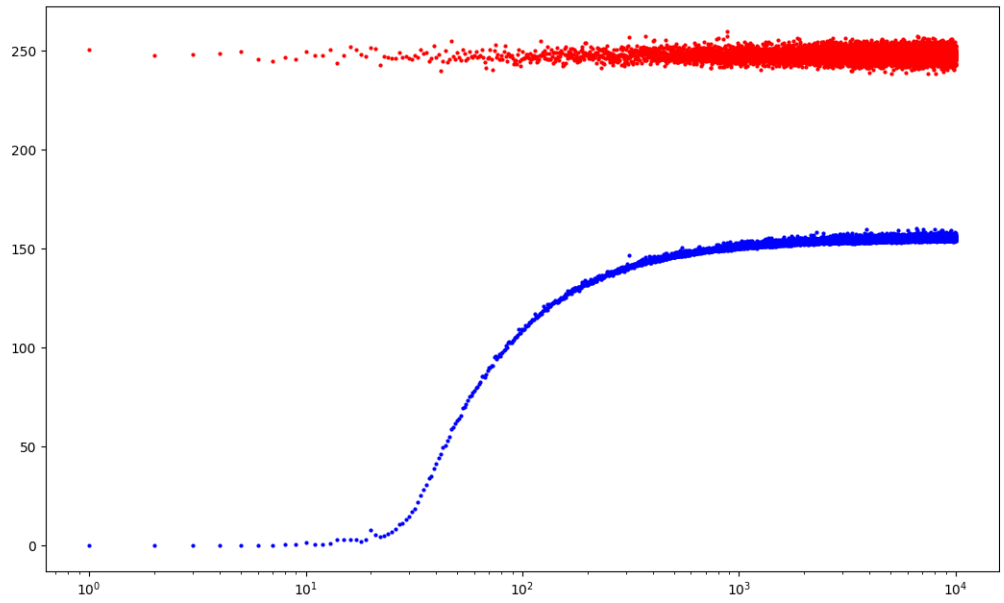


Figure 6: Big stream, Absolute error, $\theta = 0.99$, table size = 300, size stream = 50000, different elements = 10000, iters = 100

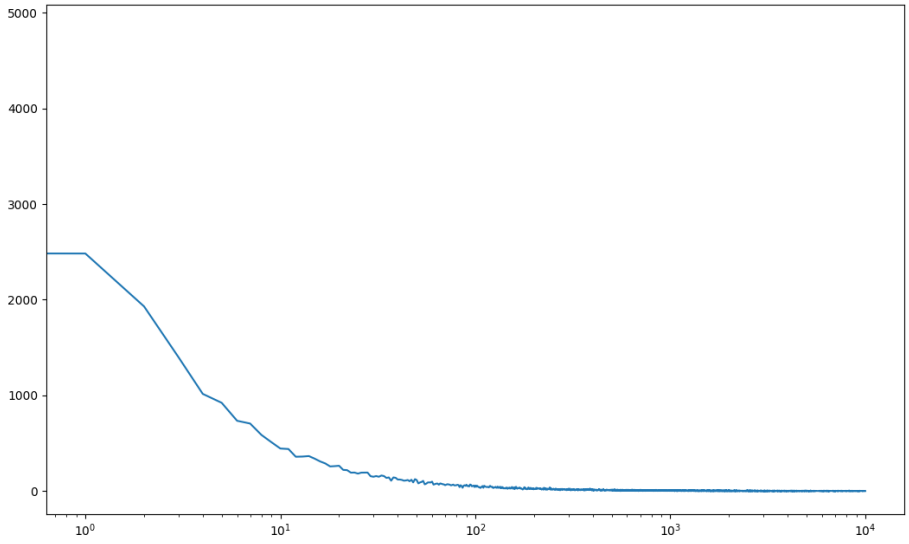


Figure 7: Distribution looks like(for one iteration)

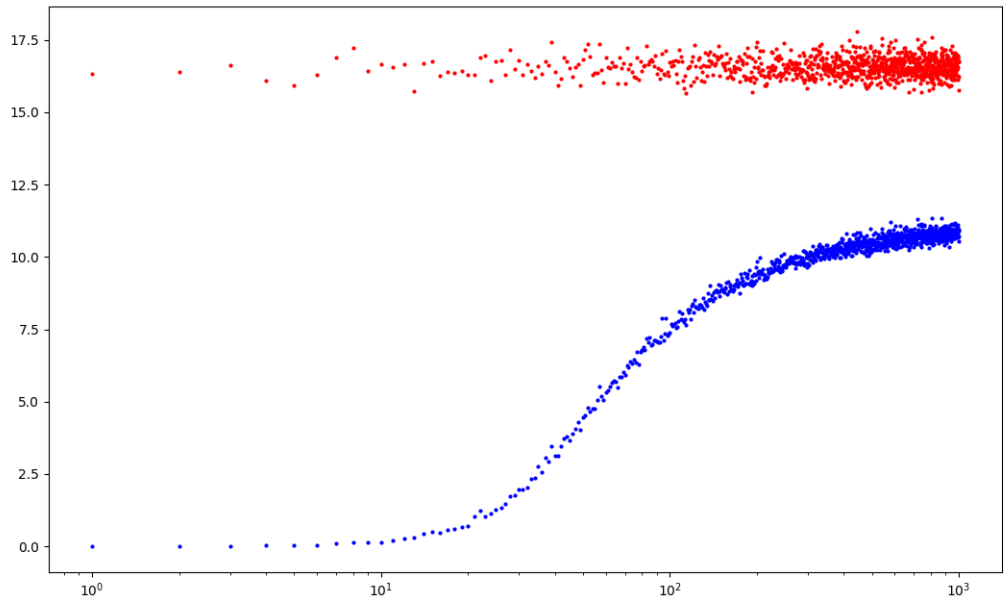


Figure 8: Smaller stream, Absolute error, $\theta = 0.99$, table size = 300, size stream = 5000, different elements = 1000, iters = 100

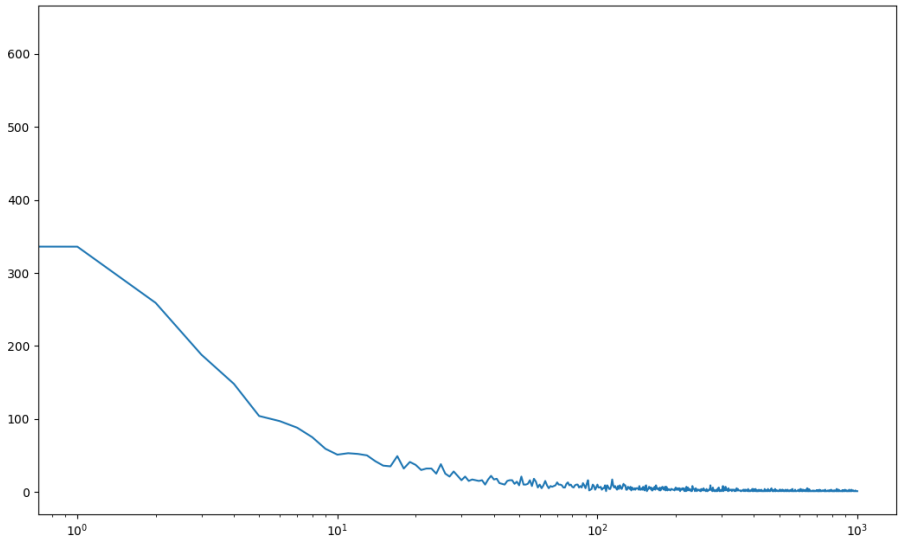


Figure 9: Distribution looks like(for one iteration)

▲ We can see that for this type of distribution CCM gives us very small error for frequent elements, which makes it very convenient to use if your distribution looks like this. Importantly, our results coincided with the article, but we used a smaller scale due to resources.

5.4 Zipf experiments with table size

Let's experiment with the size of the sketch - the following plots show the error function. On the x-axis, we show the element number (for zipf-distribution, the lower the number, the more common the element), and on the y-axis, the average error. In all experiments, 3 hash functions are used, the stream size is 5000, the number of different elements is 1000. The size of the table is 50, 100, 300, 500, 1000, 2500, 5000, respectively.

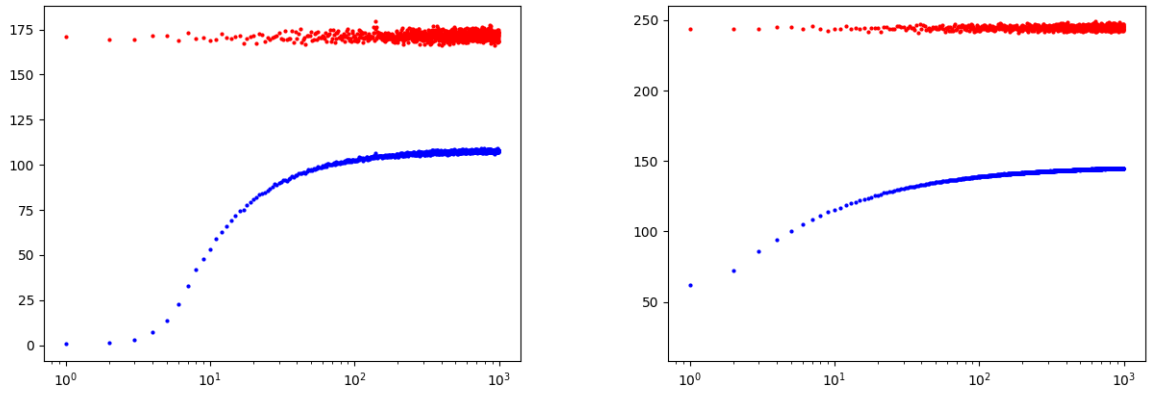


Figure 10: table size = 50, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

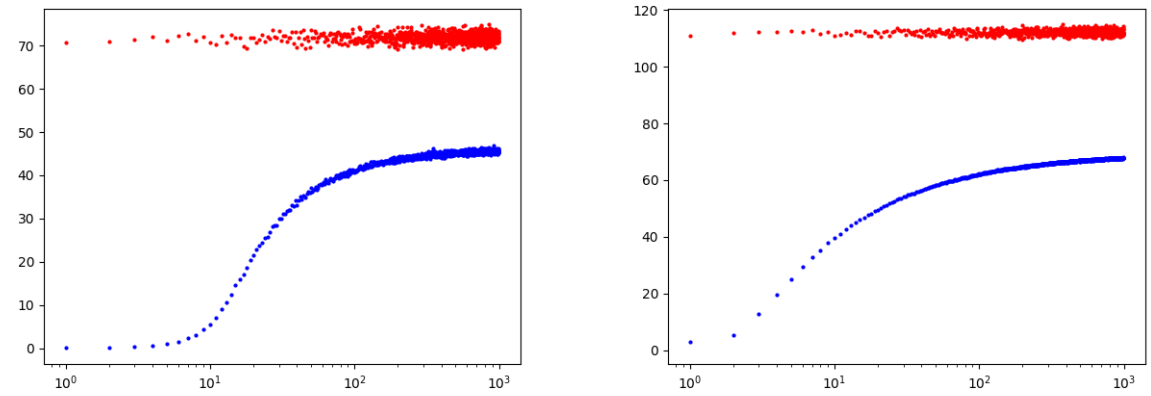


Figure 11: table size = 100, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

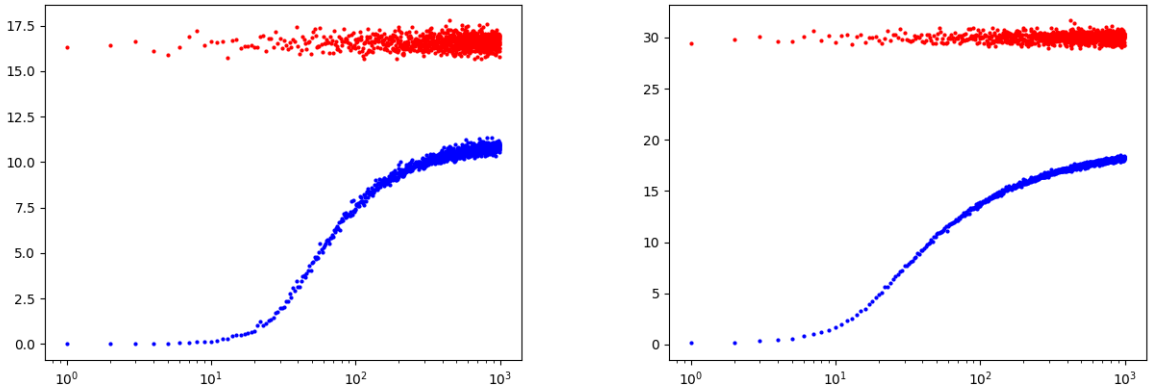


Figure 12: table size = 300, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

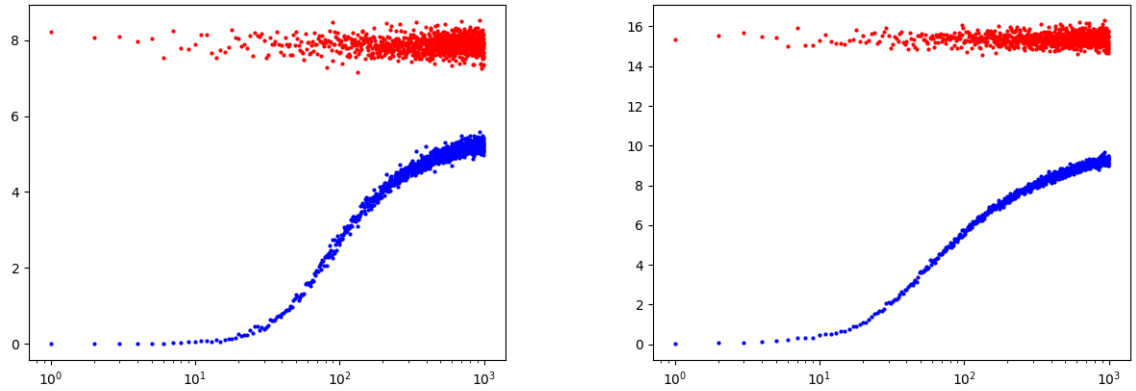


Figure 13: table size = 500, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

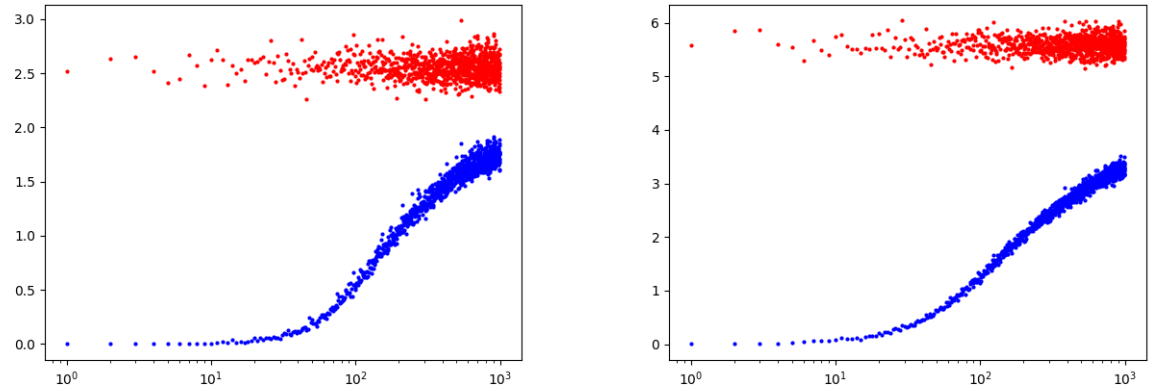


Figure 14: table size = 1000, size stream = 5000, different elements = 1000, iters = 100

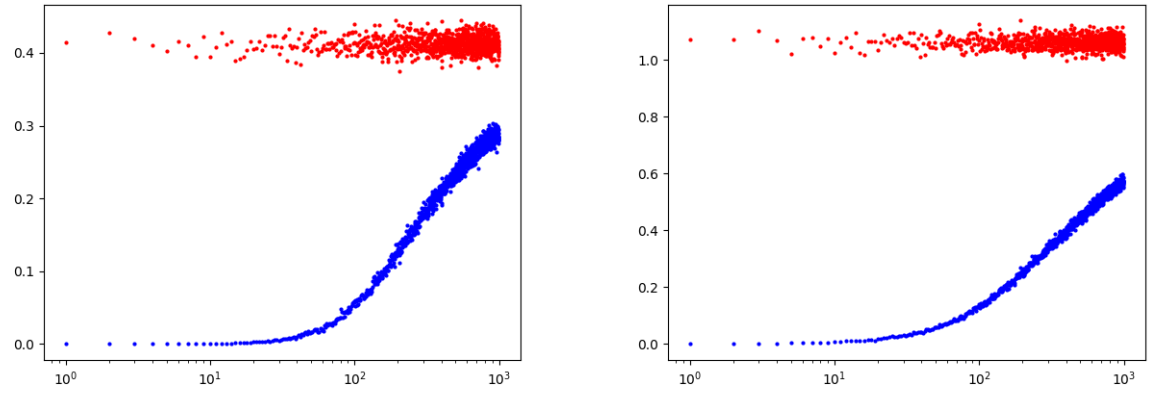


Figure 15: table size = 2500, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

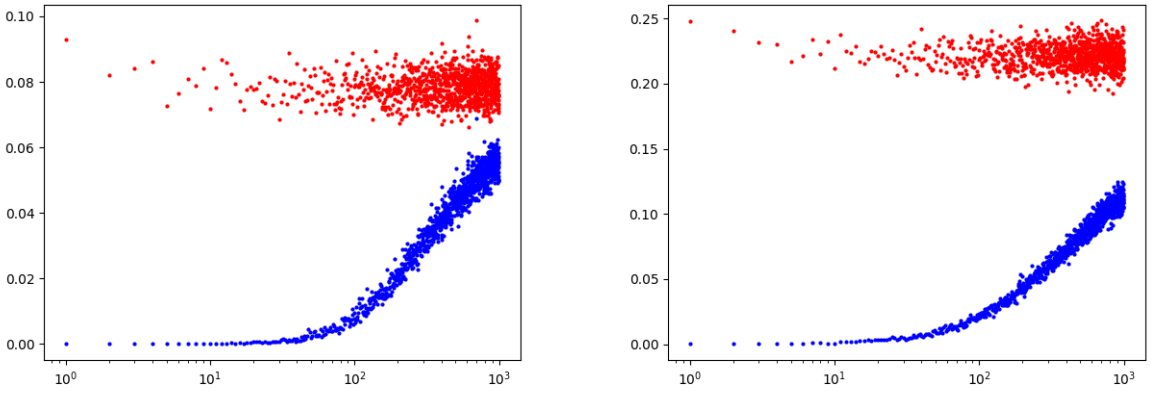


Figure 16: table size = 5000, size stream = 5000, different elements = 1000, iters = 100, left $\theta = 0.99$, right $\theta = 0.6$

6 Conclusion

Finally, CM and CCM are powerful probabilistic data structures that allow you to answer a request for the number of elements in a stream in O(1) and which support adding an element in $O(1) = O(number\ of\ hash\ functions)$. Even with three hash functions and a small table, we achieved relatively small errors. Additionally, we can to say that CCM, having a small overhead, above CM provides a much smaller error, for special types of distributions. With the right selection of parameters, a insignificant error can be achieved by CCM.

7 Bibliography

- Graham Cormode and S. Muthukrishnan. An improved data stream summary: the Count-Min sketch and its applications. Journal of Algorithms, $55(1) : 58 - 75$, April 2005.
- Graham Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. pages $44 - 55, 2005.5$ th SIAM International Conference on Data Mining, SDM 2005; Conference date: 21-04-2005 Through 23-04-2005.
- Graham Cormode. Count-min sketch. In Ling Liu and M. Tamer Özsu, editors, Encyclopedia of Database Systems, Second Edition. Springer, 2018.
- Lada A. Adamic. "Zipf, Power-laws, and Pareto - a ranking tutorial". Glottometrics 3, pages $143 - 150$. 2002.
- Younes Ben Mazziane, Sara Alouf "A Formal Analysis of the Count-Min Sketch with Conservative Updates"; Conference: "The Second IEEE INFOCOM Workshop on Networking Algorithms" London, United Kingdom May, 2022.
- Giuseppe Bianchi, Ken Duffy, Douglas Leith, Vsevolod Shneer. "Modeling Conservative Updates in Multi-Hash Approximate Count Sketches"
- Gregory T. Minton, Eric Price. "Improved Concentration Bounds for Count-Sketch"