Course : CSC591, Advanced Algorithms
Homework Assignment: H2 - Count-Min Sketches
Unity-ID: akagrawa

---

1. **Compare the Count-Min sketch with the Bloom Filter sketch. Especially compare the types of problems each aims to solve, and the space-accuracy trade-off each makes.**

   Count-Min[1] and Bloom Filters[2] both are sketch data structures that provide approximate solutions for certain common set operations. Both have a error rate vs space trade-off i.e. error rate can be controlled or minimized by increasing space. Though they belong to the same family of data structures, they differ in their implementation which provides them variant scope of providing solutions for given problems.

Bloom Filters	**Count-Min**
1. Its a bit array that stores a boolean flag for the element's presence and it is linear in the size of the set. It can perform set-membership operations. Combing two or more bloom filters can enable set union, set intersection operations also.	**1.** Its a counter array that stores a count value for the element's frequency and it is typically sublinear in the size of the frequency vector. It supports more than set-membership operations such as point-query, heavy-hitters query, top-K query, range-query, quantile query etc.
3. Bloom Filters are 1-D array of size w that stores boolean bits. It allows false positives for the above operations for the trade-off with compact space.	**3.** Count-Min sketches are 2-D array of size d x w that stores numeric count. It allows a controlled error rate ϵ with a controlled probability rate δ in its results.
4. The error probability is estimated as the function of bits per object $b = n/m$ and k set of hash functions where n is the total space and m is the total elements. Here error probability is given by $Pr = \left(1 - e^{-k/b}\right)^k$.	**4.** Since we take the minimum of counters over all the rows, the final result has error of more than $2N/w$ if all d rows give a "large" error, which happens with Probability $\leq \left(\frac{1}{2}\right)^d$.
5. From the above derived estimation, it is clear that having higher space can provides us lower error.	**5.** From the above derived estimation, it is clear that error probability reduces with increased cell depth d and similarly error rate can be controlled with increased in width w.

2. **Explain two ways in which the building of a Count-Min Sketch data structure can be parallelized, whether this parallelism scales well, and why.**

   Count-Min Sketch data structure maintains the counter in a wXd fixed size array. Here w corresponds to the width which is mapped using two pairwise-independent hash functions and d is the depth of each w cell which is used to store as a counter mapped by a hash function. Here for every row in w, the input data is mapped to one of the counter in the corresponding d cells. This type of implementation provides the flexibility to perform operation on count-min in parallel and scalable manner. Below are the points that provides support for this claim.

   (a) The key point in this data structure is that every w row has their own independence in their fixed stateless operations. In other words, row wise operations are independent of each other. This property provides the flexibility to have independent parallel operations on count-min. Using different **threads** over different rows can achieve such parallelism without affecting the state of the operation as the operations are not affected by any other row operations. Hence such parallelism can be extended in multiple nodes that enables the scalability in such operations.

   (b) Secondly, due to the additive and subtractive nature of the operations that can be done over the counter values implemented using count-min, it can work on different subsets of data simultaneously. Such independent results after performing operations on disjoint subsets can be combined in a union fashion similar to the paradigm of **map-reduce** framework. The ability to be cast its operations in map-reducible fashion strengthen the claim of its high scalability.

3. **Describe two applications of the Count-Min sketch, and what benefit(s) the Count-Min structure can offer versus an exact solution in each case.**

   Set membership and count tracking operations are the basic of many real-time applications. Such applications given the massive streaming data, provides the scope of estimation or approximation in their operations. In such scenario, count-min sketches are proved to be perfect match. Two such applications are described as follows:

   1. In a general query mode, count-min count track operations can be extended to find the **heavy hitters** i.e. the itemset that have frequency more than a given threshold. Such operations provides a wider scope of approximation because it doesn't have to deal with exact frequency count along with providing us an edge over providing exact solution by performing costly operations. In signal processing[3], recovering the heavy-hitters is a key to building the best approximation of the signal and as a result provides a compressed way to store such information.

   2. Count-min sketches providing an efficient way of compressed storing of vital information with certain error bound enables large scope in data compression applications. In Natural Language Processing where it is important to keep statistics

of the frequency of word combinations in certain sequence, Count-Min sketch provides a major improvement in compact storage when compared to exact solutions. In an experiment[4], using Count-Min sketches, the team of researchers compacted a large corpus of 90 GB data down to mere 8 GB in size.

References:

[1] Cormode, Graham, and S. Muthukrishnan. "An improved data stream summary: the count-min sketch and its applications." Journal of Algorithms 55.1 (2005): 58-75.

[2] Broder, Andrei, and Michael Mitzenmacher. "Network applications of bloom filters: A survey." Internet mathematics 1.4 (2004): 485-509.

[3] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. Proceedings of the IEEE, 98(6):937–947, June 2010.

[4] A. Goyal, J. Jagarlamudi, H. D. III, and S. Venkatasubramanian. Sketch techniques for scaling distributional similarity to the web. In Workshop on GEometrical Models of Natural Language Semantics, 2010.