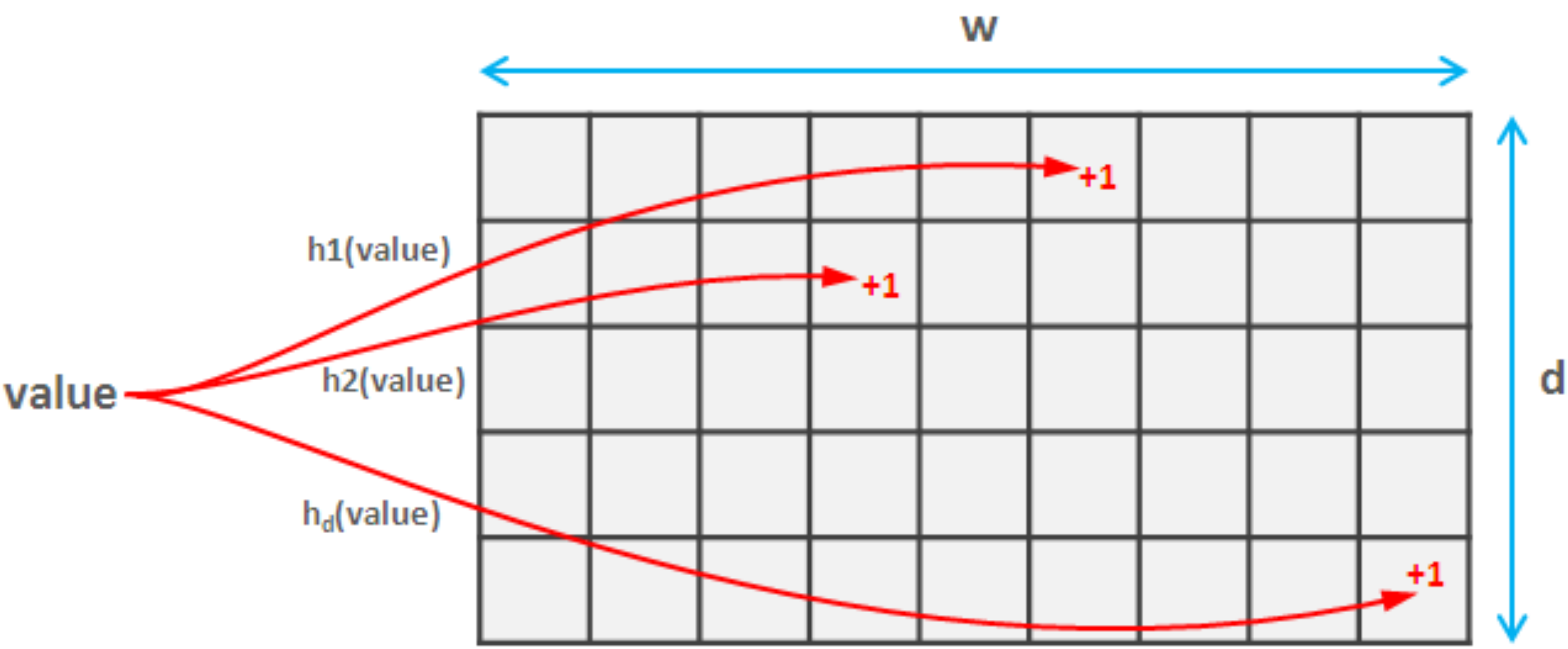
Data stream statistics

Filippo Geraci, CNR, Pisa

Cache and Bloom filters

- Consider you have a proxy that caches web pages. You may want not to cache a page that will be visited only once
  - Solution: use a bloom filter. Once you have a request first check whether it has already be seen. If YES cache the page, otherwise NO. ANYWAY add the page to the Bloom filter.

Estimation of the number of occurrences



What about computing distributions?

- Given highly skewed data I want to measure the frequency at least of the top elements
- Facts:
  - Counters are expected to be higher because of the contribution of other elements
  - CM returns the counter with less noise
- Idea
  - Estimate the contribution of noise for a specific counter

CMM – Count Mean-Min sketch

```
class CountMeanMinSketch {
    // initialization and addition procedures as in CountMinSketch
    // n is total number of added elements

    long estimateFrequency(value) {
        long e[] = new long[d]
        for(i = 0; i < d; i++) {
            sketchCounter = estimators[i][ hash(value, i) ]
            noiseEstimation = (n - sketchCounter) / (w - 1)
            e[i] = sketchCounter – noiseEstimator
        }
        return median(e)
    }
}
```

Heavy hitters

- All the above data structures allow counting or membership evaluation.
- How to know the most represented keys in a stream?
- Until now:
  - I can count how many keys exist,
  - I can check if a particular key is present
  - I can count the number of its occurrences
  - …but I can’t do anything if I don’t know it

Bad news

- Naïve solution:
  - Sort data

- There is no algorithm that solves the Heavy Hitters problems in one pass while using a sublinear amount of auxiliary space

A simple algorithm

- Problem: find the elements that occur more than N/k times (N is the stream length, k is a free parameter)
- Solution:
  - Maintain a CM and a max-heap (with k elements) of the top elements
- Process:
  1. Add the element in the CM and estimate its frequency
  2. If frequency >= N/k insert the element in the heap
  3. Note: the number of elements in the heap must be at most k

The Space saving algorithm - build

```
Algorithm: Space-Saving(m counters, stream S)
begin
  for each element, e, in S{
    If e is monitored,
      increment the counter of e;
    else{
      let e_m be the element with least hits, min
      Replace e_m with e;
      Increment count_m;
      Assign ε_m the value min;
    }
  }// end for
end;
```

The Space saving algorithm - query

Algorithm: QueryFrequent(m counters, support ϕ)

```
begin
  Bool guaranteed = true;
  Integer i = 1;
  while (count_i > φN AND i ≤ m){
    output e_i;
    If ((count_i − ε_i) < φN)
      guaranteed = false;
    i++;
  }// end while
  return( guaranteed )
end;
```

Note that counts are sorted in descending order in this implementation

References

- New Estimation Algorithms for Streaming Data: Count-min Can Do More
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.449&rep=rep1&type=pdf
- Efficient Computation of Frequent and Top-k Elements in Data Streams
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.8360&rep=rep1&type=pdf
- PROBABILISTIC DATA STRUCTURES FOR WEB ANALYTICS AND DATA MINING
  - https://highlyscalable.wordpress.com/2012/05/01/probabilistic-structures-web-analytics-data-mining/

Datasets

- Free Twitter datasets
  - http://followthehashtag.com/datasets/
- Stackexchange Q&A website
  - https://archive.org/download/stackexchange