More Stream Mining

Bloom Filters
Sampling Streams
Counting Distinct Items
Computing Moments

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Filtering Stream Content

- To motivate the Bloom-filter idea, consider a web crawler.
- It keeps, centrally, a list of all the URL's it has found so far.
- It assigns these URL's to any of a number of parallel tasks; these tasks stream back the URL's they find in the links they discover on a page.
- It needs to filter out those URL's it has seen before.

Role of the Bloom Filter

- A Bloom filter placed on the stream of URL's will declare that certain URL's have been seen before.
- Others will be declared new, and will be added to the list of URL's that need to be crawled.
- Unfortunately, the Bloom filter can have false positives.
  - It can declare a URL has been seen before when it hasn't.
  - But if it says "never seen," then it is truly new.

How a Bloom Filter Works

- A *Bloom filter* is an array of bits, together with a number of hash functions.
- The argument of each hash function is a stream element, and it returns a position in the array.
- Initially, all bits are 0.
- When input x arrives, we set to 1 the bits h(x), for each hash function h.

Example: Bloom Filter

- Use N = 11 bits for our filter.
- Stream elements = integers.
- Use two hash functions:
  - $h_1(x)$ =
    - Take odd-numbered bits from the right in the binary representation of x.
    - Treat it as an integer i.
    - Result is i modulo 11.
  - $h_2(x)$ = same, but take even-numbered bits.

Example – Continued

Stream element	h_1	h_2	Filter contents
			00000000000
25 = 11001	5	2	00100100000
159 = 10011111	7	0	10100101000
585 = 1001001001	9	7	10100101010

Bloom Filter Lookup

- Suppose element y appears in the stream, and we want to know if we have seen y before.
- Compute h(y) for each hash function y.
- If all the resulting bit positions are 1, say we have seen y before.
- If at least one of these positions is 0, say we have not seen y before.

Example: Lookup

- Suppose we have the same Bloom filter as before, and we have set the filter to 10100101010.
- Lookup element y = 118 = 1110110 (binary).
- $h_1(y) = 14$ modulo 11 = 3.
- $h_2(y) = 5$ modulo 11 = 5.
- Bit 5 is 1, but bit 3 is 0, so we are sure y is not in the set.

Performance of Bloom Filters

- Probability of a false positive depends on the density of 1's in the array and the number of hash functions.
  - = (fraction of 1's)$^{\#\text{ of hash functions}}$.
- The number of 1's is approximately the number of elements inserted times the number of hash functions.
  - But collisions lower that number slightly.

Throwing Darts

- Turning random bits from 0 to 1 is like throwing *d* darts at *t* targets, at random.
- How many targets are hit by at least one dart?
- Probability a given target is hit by a given dart = 1/t.
- Probability none of d darts hit a given target is $(1-1/t)^d$.
- Rewrite as $(1-1/t)^{t(d/t)} \sim= e^{-d/t}$.

Example: Throwing Darts

- Suppose we use an array of 1 billion bits, 5 hash functions, and we insert 100 million elements.
- That is, $t = 10^9$, and $d = 5*10^8$.
- The fraction of 0's that remain will be $e^{-1/2} = 0.607$.
- Density of 1's = 0.393.
- Probability of a false positive = $(0.393)^5 = 0.00937$.