Jelani Nelson Talk
2018-11-05 (Simons Institute)
Notes by Frank Lin

Sketches - given data and a set of possible queries Q, compress data into a sketch to support queries from Q given just the sketch
Streams - maintain sketch on the fly, as the data is updated
Some benefits of sketching: less memory consumption, faster algorithms, and bandwidth reduction for distributed computation
It turns out you can do some very cool stuff with sketching

Sum of a bunch of numbers
Easiest example of a streaming algorithm would be to simply keep a running sum and increment with each number

The approximate counting problem
Suppose we have some events that occur and we want to count the number of events that happen while minimizing for space. At any given time, we want to be able to give an estimate of how many events have taken place so far.

Essentially, we want a data structure that maintains an integer n and supports two operations:
- update(), which increments n by 1
- query(), which outputs an estimate of n

The trivial algorithm can maintain the count of k events by representing it with log(k) bits
But using a sketching algorithm, we can actually do much better than this

Intuitively, let's think about the following: imagine if each time an event happened, instead of adding 1 to the counter, we instead flip a fair coin (a coin which turns up heads with probability 1/2) and if it shows heads, we increment by 1

Now after k events happen, we expect the total counter to be approximately k/2
And we can simply spit out double that for a good approximation, saving one bit

This presents some issues, for example this approach fares very poorly for small k
We can formalize the approximate counting problem by saying we want to output some approximate estimation $\tilde{n}$ such that $P(|\tilde{n} - n| > \varepsilon n) < \delta$ for some ε, δ

To achieve this bound, we can generalize this coin example by observing that instead of flipping just one fair coin, we can flip multiple fair coins and only increment the counter if all of them turn up heads, i.e. increment with probability $1/2^X$

When querying, we can output $\tilde{n} = 2^X - 1$, which makes sense as an unbiased estimator
Can use induction to show that the expectation of 2^X after n updates is n + 1

It turns out that if you run this algorithm many times and average out results, then you can bound variance using Chebyshev and Chernoff bounds to show the earlier bound

Therefore, by using this streaming algorithm, we end up with a highly improved algorithm, which for constant ε, δ is on the order of O(log(log(n))) with constant probability

This results in exponentially reduced space complexity for the counting problem (!!!)