DS-563 / CS-543 2025-01-29

Today:

- Nice properties of linear sketching algorithms
- Streaming algorithms
- Frequency moments F_p
- AMS sketch for F_2

Reminder: Linear sketching algorithms (such as Count-Min Sketch)

Can be seen as:

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \text{randomized} \\ \text{matrix} \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

maintained low-dimensional sketch

what the algorithm does

high dimensional data / input

In CMS:

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \text{randomized} \\ \text{matrix} \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

kt counts corresponding to buckets

0/1 - matrix describing how values get assigned to buckets by hash functions

frequency counts $f(\dots)$ interpreted as a vector

Nice properties of linear sketching algorithms:
(follow from basic linear algebra properties)

- can handle both insertions & deletions

$$\underset{\text{sketch}}{s} = \underset{\text{randomized matrix}}{M} \, d \leftarrow \text{data}$$

updated sketch

change in the vector corresponding to data insertion

Insertion handling: $s' = M(d + \Delta) = Md + M\Delta = s + M\Delta$

Deletion handling: compute $M\Delta$ but subtract from sketch

- can handle multiples of updates easily:
  Instead adding $M\Delta$ to the sketch add $\alpha M \Delta$, where α is how many times the operation has to be applied.

Example: adding 1,000,000 copies of x in CMS takes essentially the same amount of time as adding one copy of x

- can handle distributed data:

$$d = d_1 + d_2 + \dots + d_k$$

each part in a different location

  - no need to send all data to a single location
  - each site computes $s_i \stackrel{def}{=} M d_i$ and sends it to a central location
  - central location can now add sketches to get sketch

for all data: $s \stackrel{\text{def}}{=} \sum s_i =$

$$= \sum M d_i$$

$$= M \sum d_i$$

$$= M d$$

Important concept: Streaming algorithms

big stream of data

$$\boxed{\text{Streaming algorithm}} \longleftarrow a_1, a_2, a_3, \dots, a_n$$

individual items,
e.g., database records,
graph edges, numbers, ...

- The algorithm reads and processes the input stream items one by one
- The algorithm should use much less space than the input size
- Natural question: How much space needed to solve a specific problem?

Comments:

- CMS can be seen as a streaming algorithm
- Insertion-only vs. insertions & deletions:

  In the simplest version, items just arrive, but in the more general version (including CMS), each input item is either "insert x" or "delete x" for some x.
- One pass vs. multiple passes:

  As presented, the algorithm gets to read each element once, but in some scenarios, it makes sense to assume that multiple passes are allowed. Example: If the data is stored on an external storage device, then sequential reading may maximize throughput (i.e., the rate at which data is read and processed). In this case, perhaps a small number of passes over the data is achievable.

Next problem: Frequency moments

Setting (same as in CMS for simplicity):

Our data is a set of items in X.

$f(x)$, for $x \in X$, is the number of copies of x.

p-th moment:

$$F_p(\text{our data}) = \sum_{x \in X} |f(x)|^p$$

$p \in (0, \infty)$

absolute value

because in some settings it makes sense to consider negative values

In the limit, if we assume that $0^0 = 0$ (and $x^0 = 1$ for $x \neq 0$), then F_0 is the number of distinct elements in our set.

Plan for next lectures:

- Approximating F_2 (AMS sketch)
- Approximating $F_0 = \#$ distinct elements