Introduction to Stream Computing and Reservoir Sampling

Agniva Chowdhury

Lecture 7: COMP 480/580
09/18/2025

Data streams

- Data that are continuously generated by many sources at very fast rates.
- Examples:
  - Google queries
  - Twitter feeds
  - Financial markets
  - Internet traffic
- We do not have complete information (e.g., size) on the entire dataset.
- Convenient to think about data as infinite.

How do you make critical calculations about the stream using limited amount of memory?

Applications

- Mining query streams:
  - Google wants to know what queries are more frequent today than yesterday.
- Mining click streams
  - Yahoo wants to know which of its pages are getting an unusual number of hits in the past hour.

- Mining social network news feeds:
  - Such as trending topics on Twitter, Facebook, etc.

Applications (cont'd)

- Sensor networks:
  - Many sensors feeding into a central controller.
- Telephone call records:
  - Data feeds into customer bills as well as settlements between telephone companies.
- IP packets monitored at a switch:
  - Gather information for optimal routing.
  - Detect denial-of-service attacks.

Motivations

- In massive or infinite data streams, we can't store all data.
- Still, we may need to:
  - Estimate statistics (e.g., mean, median).
  - Build models (e.g., classifiers).
  - Detect trends or anomalies.
- A uniform random sample:
  - Provides an unbiased summary of the full stream.
  - Can be used to approximate global properties.

Sampling

Random sampling is a powerful and general tool in data analysis. We'll see several variants and applications.

- Pick a small random set S from a large set.
- Estimate quantity of interest on S instead of entire data set.
- Analysis relies on sampling strategy, sample size, and estimation algorithm.

Basic sampling strategy: uniform sample of size k from set of size m

- *with replacement:* pick a uniformly random number $i \in \{1, 2, \dots, m\}$.
- *without replacement:* pick a single set uniformly from all sets of
- size k (with cardinality $\binom{m}{k}$).

Streaming model

- We are given a stream of objects/items/tokens e_1, e_2, e_3 … arriving one by one.
- The total number of items is unknown in advance.
- Let m be the number of items seen so far.
- The algorithm has limited memory, enough to store only $k \ll m$ items.
- We don't know how many points we will observe in advance.
- Want to compute interesting functions over the input data.

Basic question: how to sample from a stream

How do we pick a single uniform sample without knowing length of stream in advance?

- **Goal:** we want every item in a stream of unknown length to have equal probability of being selected, without storing the entire stream.
- We don’t know how to adjust the probability without knowing the total length in advance.

Our strategy: Reservoir sampling

How do we pick a single $(k = 1)$ uniform sample without knowing the length of stream in advance?

Uniform Sample:

$s \leftarrow \emptyset$
$m \leftarrow 0$

While (stream is not done):

- $m \leftarrow m + 1$
- e_m is the current item
- Toss a biased coin with $P(H) = \frac{1}{m}$
- If H:

    $S \leftarrow e_m$

EndWhile

Output s as the sample

Reservoir sampling: A key result

Let m be the number of items in a stream: $e_1, e_2, e_3 \dots, e_m$. Let s be the final output of the reservoir sampling algorithm with $k = 1$. Then

$$\mathbb{P}(s = e_j) = \frac{1}{m} \text{ for all } j = 1,2,\dots,m.$$

<u>Proof:</u>

1. Base case $(m = 1)$: Only one item e_1. The algorithm selects it with probability 1.
2. Suppose it works for $m - 1$ i.e., $\mathbb{P}(s = e_j) = \frac{1}{m-1}$ for each $j = 1,2,\dots,m-1$.
3. At step m:
   - The new item e_m either replaces the current sample with prob. $1/m$ or the previous sample was kept with prob. $1 - \frac{1}{m}$
   - For $j < m$, $\mathbb{P}(s = e_j) = \mathbb{P}(s = e_j\ before) \cdot \mathbb{P}(not\ replaced\ at\ m^{th}\ step) = \frac{1}{m-1} \cdot \left(1 - \frac{1}{m}\right) = \frac{1}{m}$
   - For $j = m$, $\mathbb{P}(s = e_j) = \mathbb{P}(replaced\ at\ m^{th}\ step) = \frac{1}{m}$

Reservoir sampling for size $k > 1$

- Want to pick k samples for $k > 1$. How?

- With replacement: Easy. simply run single sample algorithm independently in parallel and store the k items.

- Without replacement?

Reservoir sampling for size $k > 1$

Sampling without replacement:

$S[1 \ldots k] \leftarrow \emptyset$
$m \leftarrow 0$

While (stream is not done):

- $m \leftarrow m + 1$
- e_m is the current item
- If $m < k$: $S[m] \leftarrow e_m$
- Else:

  $r \leftarrow$ unform random number in $\{1,2,\ldots,m\}$

  if $r \leq k$: $S[r] = e_m$

EndWhile

Output S

Reservoir sampling for size $k > 1$

Let m be the number of items in a stream: $e_1, e_2, e_3 \ldots, e_m$. Let S be the final array of samples maintained by the algorithm with reservoir size $1 \leq k \leq m$. Then for any item e_j with $1 \leq j \leq m$,

$$\mathbb{P}(e_j \in S) = \frac{k}{m}$$

<u>Proof:</u>

1. Base case $(m = k)$: each is included with probability 1. Correct since $\frac{k}{k} = 1$.
2. Suppose after processing m items, each has probability $\frac{k}{m}$ of being in the reservoir.
3. Now, for $(m+1)^{th}$ item, $\mathbb{P}(r \leq k) = \frac{k}{m+1}$
4. For $j \leq m$: ??

Weighted sampling for size $k = 1$

- Uniform sampling treats all items equally.
- But in many applications, some items are more important than others:
  - Items with high frequency or importance.
  - Events with high value, risk, or uncertainty.
  - Weighted sampling gives priority to such important items.

- In streaming settings:
  - We want to sample important items proportionally to their weight.
  - But we can't store the whole stream to normalize weights.

Can we select items with probability proportional to weight
using **one pass and small memory**?

Weighted sampling for size $k = 1$

- Each item e_j in the stream has an associated weight $w_j > 0$.
- **Goal:** sample one item with probability proportional to its weight.

Weighted Sample:

For $j = 1$ to m:

- Generate $U_j \sim$Uniform(0,1)
- Compute priority $r_j = U_j^{1/w_j}$

EndFor

Output: Keep the item with largest r_j

- Due to Efraimidis and Spirakis (2006)

Weighted sampling for size $k = 1$

Let m be the number of items in a stream: $e_1, e_2, e_3 \ldots, e_m$. Each item has weight $w_j > 0$. Let s be the final sample maintained by the algorithm with size $= 1$. Then for any item e_j with $1 \leq j \leq m$,

$$\mathbb{P}(s = e_j) = \frac{w_j}{W} \text{ where } W = \sum_{j=1}^{m} w_j$$

Proof:

Extensions

- Frequent items / Heavy hitters in streams.
- Sketching & randomized summaries (Count-Min, CountSketch)
- Sublinear-time algorithms for streaming models
- Reservoir Sampling with deletion / time-decay models

⋮

⋮

Thank you

Questions?