COMP 480/580 — Probabilistic Algorithms and Data Structure Sept 15, 2022

Lecture 8: Introduction to Stream Computer and Reservoir Sampling

Lecturer: Anshumali Shrivastava, Zhaozhuo Xu

Scribe By: David Zhao, Meng-Chi Yu, Kristen Curry

1 Data Streams

A data stream is a continuous flow of information. Some characteristics that are important to take into account include:

- data is *continuously* generated at a *fast* rate
- data set is too large to store in its entirety
- complete information (e.g size) is unknown
- data can be viewed as *infinite*

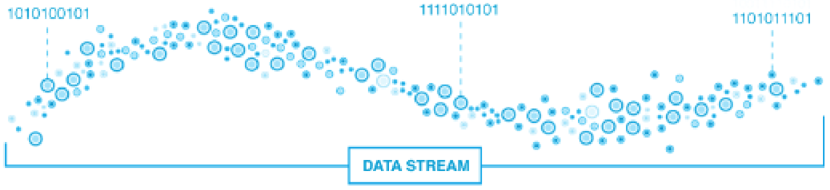

Figure 1: Continuous data stream visualization [1].

A data stream is visualized in Figure 1. The *infinite* aspect of the data is such that conventional approaches of extracting information from the data cannot be applied. Yet, essentially everything is transmitted over the internet. Thus, **how do we perform critical calculations from the stream using limited memory?**

1.1 Examples and applications

Some examples of data streams on the internet include: Google queries and Twitter feeds. Specifically, Google may want to extract information regarding queries made today as opposed to yesterday. Or Twitter may want the current information on which topics are trending. Data streams are also extended in other areas of technology such as sensory networks, telephone call records, and IP packets monitored at a switch. One sensory example is an air conditioner. An air conditioner set to maintain a given temperature takes the current temperature as a constant stream of information and acts accordingly.

1.2 Formulation: one pass model

The main complication with data streams is that we cannot store all of the information. Thus, we do a single pass on on the stream, keeping only what is desired from the data. Once the data has passed, it can no longer be recovered. This problem is formulated as a one pass model, where n, or the amount of data, is unknown.

$$D_n = x_1, x_2, ...x_n \tag{1}$$

Here, D_n is the stream of information where we observe x_t at time t. A common scenario would be to compute some function of D_t ($f(D_t)$) for a given time t.

1.2.1 One pass model example

Let's say we want to find the average of D_n for a given time t. We can achieve this by storing two values: the sum of seen values s_t and the current number of counts c_t. We can then find the average at any time t with $f(t) = \frac{s_t}{c_t}$ with memory O(1).

2 Data stream sampling

We can also estimate values by extracting and storing a representative *sample* of the data stream, then performing our function on just the subset. The question then becomes: **how do we take an unbiased representative sample of the data stream?**

2.1 Naive sampling method

A naive approach to getting a representative sample is to keep every jth element. For example, say we have n elements, yet our memory only allows for the storage size for m elements. We could then store every $j = \frac{n}{m}$ element. This has 2 issues:

1. n is not defined at the start, thus we cannot guarantee we will not run out of memory
2. this structured method of sampling can cause bias

2.2 Naive random sampling method

In attempt to combat the bias issue, we can instead randomly sample $\frac{m}{n}$ of the elements by randomly generating a number from 1 to $\frac{n}{m}$ for each element. If the random selected number is 1, then we keep the element in the sample. However, we can demonstrate that a bias still occurs through the following example:

Say we have the following data set:

- U = number of of unique elements
- $2D$ = number of of pairwise duplicate elements
- $N = U + 2D$ = total number of of elements in set
- $\alpha = \frac{2D}{U+2D}$ = true fraction of duplicate

Now we want to estimate α, the fraction of duplicates in our data set. We would do this by counting the number of duplicate and unique elements in the random sample and apply the formula accordingly. Since duplicate items are in sequence in the data set, the likelihood that the two identical elements are in the random sample is low. Thus, we would expect to underestimate the value of α. This means a bias still occurs.

2.3 Reservoir sampling

Our goal of obtaining an unbiased sample can be rewritten as obtaining 2 goals:

1. s elements are in the sample
2. every element in the sample x_t has probability $\frac{s}{t}$ of being selected

Our two naive approaches cannot achieve this since the memory is unbounded. However, this can be achieved with reservoir sampling. The algorithm works by keeping the first s elements in data stream D. For each of the following elements, with probability $\frac{s}{t}$, the new element x_t uniformly selects an element from the current reservoir sample S_{t-1} to replace. The algorithm is shown in Algorithm 1.

Algorithm 1 Reservoir Sampling

```
    Input $x_t \leftarrow$ element observed at time $t$, $s \leftarrow$ desired sample size, $S_{t-1} \leftarrow$ uniform sample at time $t-1$
    Output $S_t$ uniform sample at time $t$
  if $t < s$ then
      $S_t \leftarrow S_{t-1} \cup \{x_t\}$
  else
      if with probability $\frac{s}{t}$ then
          $v \leftarrow$ uniformly selected from $S_{t-1}$
          $S_t \leftarrow S_{t-1} - \{v\} \cup \{x_t\}$
      else
          $S_t \leftarrow S_{t-1}$
      end if
  end if
  return $S_t$
```

We now must prove our two conditions. First, it is clear that the sample is of size s. In order to prove that each element in the stream has an equal probability of being selected for the sample, we can form a proof by induction. Our inductive hypothesis is that after observing t elements, each element in the reservoir was sampled with probability $\frac{s}{t}$.

Basis: The first $t = s$ elements in the data stream are samples with probability $1 = \frac{s}{t}$

Inductive Step: We must consider both the new additional element and the elements already in the reservoir maintain selection probability $\frac{s}{t}$.

- The new additional element x_t is in the sample with probability $\frac{s}{t}$ as this is the probability it gets selected.

- To prove the probability that an element x that is already in the sample will stay in the sample, we start by first evaluating the probability that x stays in the reservoir after the new stream element x_t is seen, given that x is already in the reservoir.

$$\begin{aligned} P(x\,stays) &= P(x_t\,rejected) + P(x_t\,accept)P(x\,not\,selected) \\ &= (1-\frac{s}{t}) + \frac{s}{t}\frac{s-1}{s} \\ &= \frac{t-1}{t} \end{aligned}$$

  We can now find the probability x is in the reservoir sample, given that it has already been seen in the data stream D.

$$\begin{aligned} P(x \in S_t) &= P(x \in S_{t-1})P(x\,stays) \\ &= \frac{s}{t-1}\frac{t-1}{t} \\ &= \frac{s}{t} \end{aligned}$$

This shows that each element x in the data stream has the probability $\frac{s}{t}$ of being in the sample S and is therefore uniformly selected.

2.4 Weighted reservoir sampling

Weighted reservoir sampling is a generalized version of reservoir sampling. Now let's say we have a data stream where each element has a weight corresponding to the importance that the element is maintained in the sample. As before, we would like to sample elements from the stream such that at time n, we always have s elements sampled. However, we would now like to base the sample probability off of the elements weights, so that each element x_i in the sample has the sample probability

$$P\Big[x_i \text{ being sampled}\Big] = \frac{w_i}{\sum_{j=1}^{n} w_i} \tag{2}$$

Some guidelines were given by Pavlos S Efraimidis and Paul G Spirakis in 2006. They suggested the sampling algorithm should follow:

- Observe x_i
- Generate r_i uniformly in [0, 1]
- Set $score_i = r_i^{\frac{1}{w_i}}$
- Keep the s highest scores in the sampling pool

In order to maintain a list of top s scores, we can take advantage of a heap of element-score tuples $(x_i, score_i)$ so that we can find and lowest score tuple in $O(log(s))$ time. Note that Algorithm 1 enjoys $O(1)$ insertion and deletion, but we should not worried, as it's logarithmic and s does not have to be large.

References

[1] What is Data Stream: Definition — Blog OnAudience.com, April 2019.