CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture 18 — October 13, 2023

Prof. Samson Zhou *Scribe: Anmol Anand*

1 L_0 Sampling

Sampling distinct elements with similar probability

Given a set S of m elements from $[n]$, such that S has N distinct elements. The goal is to return a random sample, such that each unique item from S is chosen with probability $\frac{1}{N} \pm \frac{1}{poly(N)}$, where $poly(N)$ is an arbitrarily large polynomial in N (for example: N^{1000}).

The motivation of this problem is summarizing data and to deal with the unique elements in a stream regardless of their frequency.

1.1 Reservoir Sampling

Can we use reservoir sampling to solve this problem? — No, because, reservoir sampling would sample every unique element x with the probability $\frac{f_x}{m}$, where f_x is the frequency of x in the stream and m is the length of the stream. For example, reservoir sampling cannot sample the unique elements $\{1, 2\}$ in the following stream with similar probability. The probability of sampling 2 will be considerably higher if we use reservoir sampling.

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

1.2 Distinct Elements (F_0 Estimation)

Let's recall that the algorithm to find the number of distinct elements is as follows:

Algorithm 1: Number of distinct elements (F_0 estimation)

1 Set $U_0 = [n]$ and $i = 1$
2 Sample each element of U_{i-1} into U_i with probability $\frac{1}{2}$
3 If $|S \cap U_i| > \frac{2000}{\varepsilon^2} \log n$, then increment $i \leftarrow i + 1$ and repeat from Step 2
4 At the end of the stream, output $2^i \cdot |S \cap U_i|$

Can we use the distinct elements (F_0 Estimation) algorithm to sample from distinct elements in a stream with arbitrarily similar probability? — Yes. Run the distinct elements algorithm and at the end of the stream, output a random element from $S \cap U_i$.

Algorithm 2: Sampling distinct elements with arbitrarily similar probability using F_0 estimation

---

1 Set $U_0 = [n]$ and $i = 1$
2 Sample each element of U_{i-1} into U_i with probability $\frac{1}{2}$
3 If $|S \cap U_i| > \frac{2000}{\varepsilon^2} \log n$, then increment $i \leftarrow i + 1$ and repeat from Step 2
4 At the end of the stream, output a random elements from $S \cap U_i$

---

2 Distinct Elements (F_0 Estimation)

Apart from Algorithm 1, another simpler algorithm to estimate the number of distinct elements for insertion-only streams uses hash functions and is as follows:

Algorithm 3:

---

1 Let $h : [n] \to [0, 1]$ be a random hash function with a real-valued output
2 Initialize s = 1
3 For $x_1, x_2, ..., x_m$: $s \leftarrow min\left(s, h\left(x_i\right)\right)$
4 Return $Z = \frac{1}{s} - 1$

The intuition behind this algorithm is that the larger the value of N, the smaller we expect s to be.

There are other results that follow from this algorithm:

- It can be shown that $E[s] = \frac{1}{N+1}$ – however this is not the same as $E[Z] = N$ (which is not true!)
- It can be shown that $|s - E[s]| \le \varepsilon \cdot E[s] \implies (1 - 2\varepsilon)N \le Z \le (1 + 4\varepsilon)N$
- It can be shown that $Var[s] \le \frac{1}{(N+1)^2}$
- It can be shown that by taking the mean of $O\left(\frac{1}{\varepsilon^2}\right)$ independent instances , we get $|s - E[s]| \le \varepsilon \cdot E[s]$ with probability $\frac{2}{3}$

Note that $(1 - \varepsilon)s \le E[s] \le (1 + \varepsilon)s$ implies $(1 - O(\varepsilon)) \cdot N \le Z \le (1 + O(\varepsilon)) \cdot N$.

The space complexity of Algorithm 3 is $O(1)$ words. If we run $O\left(\frac{1}{\varepsilon^2}\right)$ independent instances, the space complexity of this algorithm is $O\left(\frac{1}{\varepsilon^2}\right)$ as we only need to store the s value for each of these instances.