CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 10 (Oct 7, 2019): Weighted and Priority Sampling

Lecturer: Mohammad R. Salavatipour *Scribe: Ramin Mousavi*

10.1 Introduction

Last lecture we discussed how to sample an item uniformly at random from a stream[1] when all items have a unit weight. For sampling $k > 1$ elements with replacement we can also run our 1-sampling routine for k times. In this lecture we give an algorithm for sampling $k \geq 1$ elements without replacement in a stream that elements might have different weights. This problem is called *weighted random sampling with a reservoir*. Then, we talk about *priority sampling* which we are given a stream where each item might have different weights and we want to select a representative sample of items so that we can answer subset sum queries. At the end we give a l_0-sampling and in the next lecture we will analyze it.

10.2 Weighted Sampling with a Reservoir

In the weighted sampling without replacement in the most general case, we have a stream $x_1, ..., x_n$ with positive weights $w_1, ..., w_m$ and we want to sample $k \geq 1$ distinct items such that the probability that the first item that is chosen is x_{i_1}, second item is x_{i_2} ... , the k-th item is x_{i_k} is

$$\frac{w_{i_1}}{W} \cdot \frac{w_{i_2}}{W - w_{i_1}} \cdot ... \cdot \frac{w_{i_k}}{W - w_{i_1} - ... - w_{i_{k-1}}}, \tag{10.1}$$

where W is the total weights of the items, i.e., $W = \sum_{i=1}^{n} w_i$.

Note that for $k = 1$ we can easily adapt the 1-sampling with replacement's algorithm from the last lecture in the following way: when we see an element x_i, pick x_i with probability $\frac{w_i}{\sum_{j=1}^{i} w_j}$. The following algorithms is due to [ES06].

[1] Note that we do not know the number of elements in the stream in advance.

Weighted Random Sampling with a Reservoir

1. Let $S[1...k] \leftarrow \emptyset$
2. $i \leftarrow 0$
3. While there is a new element in the stream do

   $i \leftarrow i + 1$

   Pick r_i uniformly at random from $(0, 1)$

   $w'_i \leftarrow r_i^{\frac{1}{w_i}}$

   If $i \leq k$

   add (x_i, w'_i) to S

   Else

   Update S by keeping the largest k values respect to w'_is.

4. Return S in a non-increasing manner, i.e., the item with the largest w'_i in S comes first and so on (this will be helpful in the analysis later).

We show that the above algorithm works correctly for the case that there are two items in the stream $x_1,\ x_2$ and $k = 1$. The argument for the general case is similar and is left as an exercise.

Lemma 1 *Let r_1, r_2 be two numbers drawn independently from uniform distribution over $(0, 1)$. Let $X_1 = r_1^{\frac{1}{w_1}}$ and $X_2 = r_2^{\frac{1}{w_2}}$ for $w_1, w_2 > 0$. Then,*

$$\Pr[X_1 \leq X_2] = \frac{w_2}{w_1 + w_2}.$$

Proof. Let $X = r^{\frac{1}{w}}$ for any $w > 0$ where $r \sim (0, 1)$. Then, cumulative distribution function for X is

$$F_X(t) := \Pr[X \leq t] = \Pr[r^{\frac{1}{w}} \leq t] = \Pr[r \leq t^w] = t^w.$$

So the probability density function for X is $f_X(t) = \frac{dF_X(t)}{dt} = wt^{w-1}$.

$$\begin{aligned}
\Pr[X_1 \leq X_2] &= \int_{t_2=0}^{1} \int_{t_1=0}^{t_2} f_{X_1}(t_1) f_{X_2}(t_2) dt_1 dt_2 \\
&= \int_{t_2=0}^{1} f_{X_2}(t_2) dt_2 \int_{t_1=0}^{t_2} f_{X_1}(t_1) dt_1 \\
&= \int_{t_2=0}^{1} f_{X_2}(t_2) F_{X_2}(t_2) dt_2 \\
&= \int_{0}^{1} w_2 t^{w_2 - 1} t^{w_1} dt \\
&= \frac{w_2}{w_1 + w_2}.
\end{aligned}$$

■

In the case that the stream is x_1, x_2 and $k = 1$, Lemma 1 implies that $\Pr[x_i \textit{ is chosen}] = \frac{w_i}{w_1 + w_2}$ for $i = 1, 2$, as desired.

Now we state the more general lemma related to our sampling algorithm.

Lemma 2 *Let $r_1, ..., r_n$ be the numbers drawn independently from uniform distribution over $(0,1)$. Let $X_i = r_i^{\frac{1}{w_i}}$ for $1 \le i \le n$. Then, for any $\alpha \in [0,1]$ we have*

$$\Pr[X_1 \le X_2 \le ... \le X_n \le \alpha] = \alpha^{w_1+...+w_n} \prod_{i=1}^{n} \frac{w_i}{w_1 + ... + w_i}.$$

Proof. Apply induction on n and use Lemma 1 as the based case. ■

Lemma 2 immediately implies the following fact.

Corollary 1 *The probability that X_j is the largest value among $X_1, ..., X_n$ is $\frac{w_j}{w_1+...+w_n}$.*

Proof. WLOG, we can assume $j = n$. Let $\mathcal{S}_{n-1}$ be the set of all permutations for $1, ..., n-1$. So

$$\begin{aligned}
\Pr[X_n \textit{ is the largest}] &= \sum_{\sigma \in \mathcal{S}_{n-1}} \Pr[X_{\sigma(1)} \le ... \le X_{\sigma(n-1)} \le X_n] \\
&= \frac{w_n}{w_1 + ... + w_n} \Big(\sum_{\sigma \in \mathcal{S}_{n-1}} \Pr[X_{\sigma(1)} \le ... \le X_{\sigma(n-1)}] \Big) \\
&= \frac{w_n}{w_1 + ... + w_n},
\end{aligned}$$

where the second equality holds by applying Lemma 2 with $\alpha = 1$, and the last equality follows because the sum inside the parenthesis is 1. ■

Theorem 1 *The Weighted Random Sampling with a Reservoir algorithm outputs a set S with k distinct items and the probability of choosing x_{i_1} first, x_{i_2} second .... and x_{i_k} in the k-th round is equal to (10.1).*

Proof. The probability that X_{i_1} is the largest (it should be in order to be output first) by Corollary 1 is $\frac{w_{i_1}}{w_1+...+w_n}$. Then, we can condition on X_{i_1} to be the largest and use the fact that X_is are independent (since r_is are independent). ■

10.3 Priority Sampling

In the priority sampling problem, we are given a stream $\sigma = x_1, ..., x_n$ with non-negative weights $w_1, ..., w_n$. We want to answer the following query: given a subset of indices $I \subseteq [n]$, output $\sum_{i \in I} w_i$. For a given k, we describe an algorithm that finds a sample $S \subseteq [n]$ of size at most k such that for a given $I \subseteq [n]$ it approximates the value of $\sum_{i \in I} w_i$. For a given k, we describe an algorithm that finds a sample $S \subseteq [n]$ with high probability.

The following is the priority sampling algorithm due to [DLT07].

Priority Sampling

1. For each item $i \in [n]$ we see in the stream, pick uniformly at random $u_i \in (0, 1]$.
2. Compute priority q_i of item i which is $q_i := \frac{w_i}{u_i}$.
3. Always keep the items with the largest k priorities and call this set S. Also keep the $(k+1)$-th largest priority τ.
4. Given a set $I \subseteq [n]$, return $\hat{W}_I = \sum_{j \in I \cap S} \max\{\tau, w_j\}$.

Define $\hat{w}_i$ as follows:

$$\hat{w}_i := \begin{cases} \max\{\tau, w_i\}, & \text{if } i \in S \\ 0, & \text{otherwise,} \end{cases}$$

where τ is the $k+1$-th largest priority obtained from the algorithm. In the next lemma, we show that in expectation we get what we wanted.

Lemma 3 $\mathbb{E}[\hat{w}_i] = w_i$.

Proof. Let $A(\tau')$ be the event that the $(k+1)$-th largest priority is τ'. Then, for all $i \in S$ we have $q_i \geq \tau'$ and $\hat{w}_i = \max\{\tau', w_i\}$. For $i \notin S$, we have $q_i \leq \tau'$ and $\hat{w}_i = 0$.

We consider two cases:

Case 1 (When $w_i \geq \tau'$): $\Pr[i \in S |\ A(\tau')] = 1$ and $\hat{w}_i = w_i$, since $q_i = \frac{w_i}{u_i} > \tau'$ so it is selected in S. So $\mathbb{E}[\hat{w}_i] = 1 \cdot w_i = w_i$.

Case 2 (When $w_i < \tau'$): $\Pr[i \in S |\ A(\tau')] = \Pr[\frac{w_i}{u_i} \geq \tau'] = \Pr[u_i \leq \frac{w_i}{\tau'}] = \frac{w_i}{\tau'}$, and $\hat{w}_i = \tau'$. So $\mathbb{E}[\hat{w}_i] = \frac{w_i}{\tau'} \cdot \tau' = w_i$.

So we have $\mathbb{E}[\hat{w}_i] = w_i$. Also in expectation we satisfy the subset sum query because of the linearity of expectation. ■

Next we compute the variance of $\hat{w}_i$. Let $\hat{v}_i$ be

$$\hat{v}_i := \begin{cases} \tau \max\{0, \tau - w_i\}, & \text{if } i \in S \\ 0, & \text{otherwise.} \end{cases}$$

Lemma 4 $\text{Var}[\hat{w}_i] = \mathbb{E}[\hat{v}_i]$.

Proof. Omitted! ■

We can also show that $\text{Cov}(\hat{w}_i, \hat{w}_j) = 0$. In fact we can show that

$$\mathbb{E}[\prod_{i \in I} \hat{w}_i] = \begin{cases} \prod_{i \in I} w_i, & \text{if } |I| \in k \\ 0, & \text{otherwise.} \end{cases}$$

This implies the following fact about the variance.

Lemma 5 $\mathrm{Var}[\sum_{i \in I} \hat{w}_i] = \sum_{i \in I} \mathrm{Var}[\hat{w}_i]$.

So once we know τ, we can compute the variance of $\sum_{i \in I} \hat{w}_i$. Then, we can apply Chebyshev's inequality to bound the error in the estimation.

10.4 l_0-Sampling

In l_p-sampling, we are given a non-zero vector $a = (a_1, ..., a_n) \in \mathbb{R}^n$ and we want to sample a random element $r \in [n]$ such that $\Pr[r = i] = \frac{|a|_i^p}{\sum_{j \in [n]} |a_j|^p}$. For example, the reservoir sampling (with replacement) is an l_1-sampling. For a given error parameters $\epsilon, \delta > 0$, our goal is to sample an item i such that

$$(1+\epsilon)\frac{|a_i|^p}{\sum_{j \in [n]} |a_j|^p},$$

and we want the probability of failure be bounded by δ. Here we give an algorithm for l_0-sampling due to [CF14]. We first give a non-stream version of the algorithm and then we give the streaming version.

l_0-Sampling (non-streaming version) with the error parameters $\epsilon, \delta > 0$

1. Let $s = O(\max\{\log \frac{1}{\epsilon}, \log \frac{1}{\delta}\}$.
2. Let $m = O(\log n)$.
3. Let $h : [n] \to [n^3]$ be a hash function that is chosen uniformly at random from a $O(s)$-universal hash family.
4. For $0 \leq j \leq m$ define the vectors $a[j]$ as follows:
$$a[j]_i = \begin{cases} a_i, & \text{if } h(i) \leq \frac{n^3}{2^j} \\ 0, & \text{otherwise.} \end{cases}$$
5. Feed each $a[j]$ as input to an s-sparse recovery algorithm.
6. Pick the smallest j such that $a[j]$ is s-sparse. Note that the s-sparse algorithm returns a vector. sample randomly a coordinate of this sparse vector.

Note that $a[0]$ is the same as a and $a[1]$ is a vector that roughly half of the coordinates of a are zeroed and so on so forth.

Now we give the streaming algorithm for l_0-sampling.

l_0-Sampling (streaming version) with the error parameters $\epsilon, \delta > 0$

1. Let $s = O(\max\{\log \frac{1}{\epsilon}, \log \frac{1}{\delta}\}$.
2. Let $m = O(\log n)$.
3. Let $D_1, ..., D_{\log n}$ be independent s-sparse recovery algorithms.
4. Let $h : [n] \to [n^3]$ be a hash function that is chosen uniformly at random from a $O(s)$-universal hash family.
5. While there is a token (i, c) do

   For all $0 \leq j \leq m$ do

   if $h(i) \leq \frac{n^3}{2^j}$

   Feed (i, c) to D_j
6. Find the smallest j such that D_j returns an s-sparse vector, then sample uniformly at random one of the coordinates of this sparse vector.

We will see the analysis of this algorithm in the next lecture.

References

ES06 S. Efraimidis and Paul G. Spirakis, Weighted random sampling with a reservoir. *Inf. Process. Lett*, 97, 5 (March 2006), 181-185.

DLT07 N. Duffield, C. Lund, and M. Thorup, Priority sampling for estimation of arbitrary subset sums. *J. ACM 54, 6, Article 32*, 2007.

CF14 G. Cormode, and D. Firmani, A unifying framework for 0-sampling algorithms. *Distributed and Parallel Databases*, 32.3 (2014): 315-335.