Weighted Reservoir Sampling Proof

Levi Viana

March 17, 2019

1 Introduction

The problem of random sampling without replacement calls for the selection of k distinct random items out of a population of size n. If all items have the same probability to be selected, the problem is known as uniform random sampling.

In weighted random sampling the items are weighted and the probability of each item to be selected is determined by its relative weight.

I wrote this short essay to provide a proof that the algorithm statated in [1] formally works, since [1] doesn't provide it. I made an implementation available at [2].

2 Weighted Sampling without replacement

Let w_i be the weights for $i \in [1, ..., n]$. We can define weighted random sampling without replacement by the following algorithm :

Algorithm A:
Input : *A population V of n weighted items.*
Output : *A subset S of V with size k.*
1 : *For $i = 1$ to k*
2 : *let $\pi_j(i) = \frac{w_j}{\sum_{sm \in V-S} w_m}$ be the probability of picking the j-th item of $V - S$ in the i-th round*
3 : *Randomly select an item v_j from $V - S$ and insert it into S*

Consider the following algorithm:

Algorithm B:
Input : *A population V of n weighted items.*
Output : *A subset S of V with size k.*
1 : *For each v_i in V, let x_i be a sample from an independent uniform distribution on the interval $[0, 1]$, and let $k_i = x_i^{1/w_i}$ be defined as its key*
2 : *S = Subset of the k items with the largest keys*

Claim: *The algorithm B performs weighted random sampling without replacement as defined by the algorithm A.*

Proof of the claim:

Let's first consider the case where $k = 1$. Let's calculate the probability that the first item will be picked. In order for this to happen, the following inequality must be satisfied:

$$x_1^{1/w_1} > max(\{x_i^{1/w_i}\}_{i=2,...,n}) \tag{1}$$

Let p_i be defined as the probability that the first item is picked whenever $i = argmax(\{x_j^{1/w_j}\}_{j=2,...,n})$. Thus, we have :

$$p_i = \int_0^1 (1 - x^{w_1/w_i}) x^{\frac{\sum_{\hat{i},\hat{1}} w_j}{w_i}} dx \tag{2}$$

Where $\sum_{\hat{i},\hat{1}} w_j$ is the sum of all weights except the i-th and the first items' weights. Thus, it follows that $p = \sum_{\hat{1}} p_i$. For convinience, the sums are over all indicies when those are not explicited.

The equation (2) can be deduced in the following way: let x be a fixed value for the sample x_i, then we would have $x_1^{1/w_1} > x^{1/w_i}$, which happens with probability $1 - x^{w_1/w_i}$ and $x_j^{1/w_j} < x^{1/w_i}$, which happens with probability x^{w_j/w_i}, for all $j \in [2, ..., \hat{i}, .., n]$. Then, we multiply all these probabilites (since the samples are drawn independently) and integrate over all the probability space of x_i (i.e. interval [0, 1] and constant density function equal to 1).

By solving the integral (2) we find that :

$$p_i = \frac{1}{1 + \frac{\sum_{\hat{i},\hat{1}} w_j}{w_i}} - \frac{1}{1 + \frac{\sum_{\hat{i}} w_j}{w_i}} \tag{3}$$

$$p_i = \frac{w_i}{\sum_{\hat{1}} w_j} - \frac{w_i}{\sum w_j} \tag{4}$$

Thus, we finally find that :

$$p = \sum_{\hat{1}} p_i = \sum_{\hat{1}} (\frac{w_i}{\sum_{\hat{1}} w_j} - \frac{w_i}{\sum w_j}) = \tag{5}$$

$$\frac{\sum_{\hat{1}} w_i}{\sum_{\hat{1}} w_j} - \frac{\sum_{\hat{1}} w_i}{\sum w_j} = 1 - \frac{\sum_{\hat{1}} w_i}{\sum w_j} = \frac{w_1}{\sum w_j} \tag{6}$$

This result matchs the 2nd step of the algorithm A straightforwardly by considering that in each round the item having largest key in $V - S$ will be picked. #

3 Weighted Reservoir Sampling

The reservoir rampling version of the algorithm B is the following :

Algorithm C:
Input : *A population V of n weighted items.*
Output : *A subset S of V with size k.*
1 : *Initialize the reservoir R with the first k items of V*
2 : *For each $v_i \in R$ calculate its key $k_i = x_i^{1/w_i}$, where x_i is a random sample of an independent uniform distribution over [0, 1]*
3 : *For each v_i in $V - R$:*
4 : *let m be the index of the item of R having the smallest key K*
5 : *calculate the key of v_i, $k_i = x_i^{1/w_i}$ (same method)*
6 : *if $k_i > K$, swap the i-th and m-th items*

The algorithms B and C are equivalent since the algorithm C garantees that the items having the k largest keys will be found inside the reservoir since either (i) they are initialized in it, or (ii) they will be eventually swapped with an item having a lower key value.

References

[1] Efraimidis, Spirakis. *Weighted Random Sampling*, 2005
`https://utopia.duth.gr/~pefraimi/research/data/2007EncOfAlg.pdf`

[2] Efficient Reservoir Sampling implementation for PyTorch,
`https://github.com/LeviViana/torch_sampling`