On the formula for weighted sampling without replacement

sjhi00

Abstract

For weighted sampling without replacement, we provide a formula of a probability that a specific item is chosen. It is possible to calculate it with $\mathcal{O}\left(N \cdot 2^N\right)$ time complexity and $\mathcal{O}\left(N\right)$ space complexity, where N denotes the total number of items. Let the weight of each item is a positive integer and the sum of weights is equal to W. Then, it is possible to calculate the value of the formula with $\mathcal{O}\left(N^2W\right)$ time complexity and $\mathcal{O}\left(NW\right)$ space complexity by using dynamic programming.

1 Introduction

Consider the following weighted sampling without replacement.

- There are total N items, and the i-th item has a positive integer-valued weight w_i for $0 \leq i < N$. (For convenience, we use 0-based indexes for items.)
- Let M be a given integer such that $1 \leq M \leq N$. We repeat the following process M times.
  - Choose an item which is not chosen yet. If the i-th item is not chosen, the probability that i-th item is chosen at this time is propotional to w_i. In other words, if the $w_{i_1}, \cdots, w_{i_k}$-th items are not chosen yet (and other items are already chosen), the probability that w_{i_r}-th item is chosen at this time is $\frac{w_{i_r}}{\sum_{i=1}^{k} w_{i_k}}$ $(1 \leq r \leq k)$.
- Finally, M items were chosen among N items.

In this situation, let E_i be an event such that the i-th item is finally chosen, where $0 \leq i < N$.[1] It is possible to calculate $\mathbb{P}\left(E_0\right)$ using dynamic programming with a bitfield (bitmask), which has $\mathcal{O}\left(N \cdot 2^N\right)$ time complexity and $\mathcal{O}\left(2^N\right)$ space complexity.
Let S be a subset of $T = \{0, 1, \cdots, N-1\}$. Let $d(S)$ be a probability such that the 0-th item is finally chosen, given that all items in S are already chosen, where $0 \notin S$ and $|S| < M$. We define $d(S) = 1$ if $0 \in S$, and we define $d(S) = 0$ if $0 \notin S$ and $|S| \geq M$. Then, we have $d(\varnothing) = \mathbb{P}\left(E_0\right)$.
Let S be a subset of $T = \{0, 1, \cdots, N-1\}$, and let $T \setminus S = \{i_1, \cdots, i_k\}$. Then, we have the following formula.

$$d(S) = \frac{1}{\sum_{r=1}^{k} w_{i_r}} \cdot \sum_{r=1}^{k} w_{i_r} d\left(S \cup \{i_r\}\right)$$

Using an n-bit binary representation to express S, it is possible to calculate $d(\varnothing)$ using dynamic programming with the bitfield.
In this article, we provide a formula of $\mathbb{P}\left(E_0\right)$, which is possible to be calculated with $\mathcal{O}\left(N \cdot 2^N\right)$ time complexity and $\mathcal{O}\left(N\right)$ space complexity. Furthermore, we provide an algorithm with dynamic programming, which has $\mathcal{O}\left(N^2W\right)$ time complexity and $\mathcal{O}\left(NW\right)$ space complexity, where $W = \sum_{i=0}^{N-1} w_i$.

2 Related Work

2.1 A formula using binomial coefficients

Let $\binom{n}{r} = \begin{cases} \frac{n!}{r!(n-r)!} & (0 \leq r \leq n) \\ 0 & (\text{otherwise}) \end{cases}$ be a binomial coefficient, where $n! = \begin{cases} 1 \times 2 \times \cdots \times n & (n > 0) \\ 1 & (n = 0) \end{cases}$.

[1] In the problem 35001 in Baekjoon Online Judge, it is required to calculate $\mathbb{P}\left(E_0\right)$ modulo $10^9 + 7$, where $1 \leq M \leq N \leq 20$ and $w_i = A_i^2$, $1 \leq A_i \leq 100$.

Lemma 1. Let N, M be integers such that $1 \leq M \leq N$. Then, $\displaystyle\sum_{r=0}^{M-1} (-1)^r \binom{N-M-1+r}{N-M-1}\binom{N-1}{N-M+r} = 1$.

Proof. Let $g(x) = \left(1+x+x^2+\cdots+x^{M-1}\right)^{N-M}$ and $h(x) = (1-x)^{N-1}$. Since the number of the tuple of nonnegative integers $(x_1, x_2, \cdots, x_{N-M})$ such that $x_1 + x_2 + \cdots + x_{N-M} = r$ is equal to $\binom{N-M-1+r}{N-M-1}$ for $0 \leq r \leq M-1$, we have $g(x) = \displaystyle\sum_{r=0}^{M-1} \binom{N-M-1+r}{N-M-1} x^r + x^m P_1(x)$ for some polynomial P_1. By the binomial theorem, we have $h(x) = \displaystyle\sum_{r=0}^{N-1} \binom{N-1}{r} (-1)^r x^r$. Hence, the coefficient of x^{M-1} in $g(x)h(x)$ is equal to

$$\sum_{r=0}^{M-1} \binom{N-M-1+r}{N-M-1}\binom{N-1}{M-1-r}(-1)^{M-1-r} = (-1)^{M-1}\sum_{r=0}^{M-1} (-1)^r \binom{N-M-1+r}{N-M-1}\binom{N-1}{N-M+r}.$$

On the other hand, we have

$$g(x)h(x) = \left(\left(1+x+x^2+\cdots+x^{M-1}\right)^{N-M}(1-x)^{N-M}\right)(1-x)^{M-1} = \left(1-x^M\right)^{N-M}(1-x)^{M-1}.$$

The coefficient of x^{M-1} in $g(x)h(x)$ is equal to $(-1)^{M-1}$, since $\left(1-x^M\right)^{N-M} = 1 + x^M P_2(x)$ for some polynomial P_2. Hence we have $\displaystyle\sum_{r=0}^{M-1} (-1)^r \binom{N-M-1+r}{N-M-1}\binom{N-1}{N-M+r} = 1$. □

Theorem 1. Consider the problem of the weighted sampling without replacement, which is described in the previous section. If $M = N$, then $\mathbb{P}(E_0) = 1$. If $1 \leq M \leq N-1$, then

$$\mathbb{P}(E_0) = \sum_{r=0}^{M-1} \left((-1)^r \binom{N-M-1+r}{N-M-1}\right) \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S| = N-M+r}} \frac{w_0}{w_0 + \sum_{i\in S} w_i}. \tag{1}$$

Proof. We can use a mathematical induction on N. When $N = 1$, it is clearly true since $N = M = 1$ and the 0-th item should be always chosen.
Now let $N \geq 2$, and assume that it is true for $N-1$.
When $M = 1$, we have $\mathbb{P}(E_0) = \dfrac{w_0}{\sum_{i=0}^{N-1} w_i} = \dfrac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i}$, hence it is true. When $M = N$, then it is clearly true since the 0-th item should be always chosen.
Now let $2 \leq M \leq N-1$. Then, we have

$$\begin{aligned}
&\mathbb{P}(E_0) - \frac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i} \\
&= \sum_{k=1}^{N-1} \frac{w_k}{w_0 + \sum_{i=1}^{N-1} w_i} \left(\sum_{r=0}^{M-2} (-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\}\setminus\{k\} \\ |S| = N-M+r}} \frac{w_0}{w_0 + \sum_{i\in S} w_i} \right) \\
&= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{k=1}^{N-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\}\setminus\{k\} \\ |S| = N-M+r}} \frac{w_k}{w_0 + \sum_{i=1}^{N-1} w_i} \cdot \frac{w_0}{w_0 + \sum_{i\in S} w_i} \right) \\
&= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S| = N-M+r}} \sum_{\substack{1 \leq k \leq N-1 \\ k \notin S}} \frac{w_k}{w_0 + \sum_{i=1}^{N-1} w_i} \cdot \frac{w_0}{w_0 + \sum_{i\in S} w_i} \right) \\
&= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S| = N-M+r}} \frac{\left(\sum_{\substack{1 \leq k \leq N-1 \\ k \notin S}} w_k\right)}{w_0 + \sum_{i=1}^{N-1} w_i} \cdot \frac{w_0}{w_0 + \sum_{i\in S} w_i} \right) \\
&= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S| = N-M+r}} \frac{\left(\left(w_0 + \sum_{i=1}^{N-1} w_i\right) - \left(w_0 + \sum_{i\in S} w_i\right)\right)}{w_0 + \sum_{i=1}^{N-1} w_i} \cdot \frac{w_0}{w_0 + \sum_{i\in S} w_i} \right)
\end{aligned}$$

$$= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=N-M+r}} \left(\frac{w_0}{w_0 + \sum_{i \in S} w_i} - \frac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i} \right) \right)$$

$$= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=N-M+r}} \frac{w_0}{w_0 + \sum_{i \in S} w_i} \right)$$

$$- \frac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i} \sum_{r=0}^{M-2} (-1)^r \binom{N-M-1+r}{N-M-1} \binom{N-1}{N-M+r}$$

$$= \sum_{r=0}^{M-2} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=N-M+r}} \frac{w_0}{w_0 + \sum_{i \in S} w_i} \right)$$

$$- \frac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i} \left(1 - (-1)^{M-1} \binom{N-M-1+(M-1)}{N-M-1} \right)$$

$$= -\frac{w_0}{w_0 + \sum_{i=1}^{N-1} w_i} + \sum_{r=0}^{M-1} \left((-1)^r \binom{N-M-1+r}{N-M-1} \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=N-M+r}} \frac{w_0}{w_0 + \sum_{i \in S} w_i} \right).$$

Hence we have $\mathbb{P}(E_0) = \sum_{r=0}^{M-1} \left((-1)^r \binom{N-M-1+r}{N-M-1} \right) \sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=N-M+r}} \frac{w_0}{w_0 + \sum_{i \in S} w_i}$.

By the mathematical induction, we can conclude that the given statement is true. □

It is possible to calculate the value of the equation (1) with $\mathcal{O}\left(N \cdot 2^N\right)$ time complexity and $\mathcal{O}(N)$ space complexity, since $\sum_{r=0}^{M-1} \binom{N-1}{N-M+r} \leq \sum_{k=0}^{N-1} \binom{N-1}{k} = 2^{N-1}$. To calculate binomial coefficients, it is enough to calculate $0!, 1!, 2!, \cdots, N!$ and their inverses, and store in an array. [2]

Here are some examples with fixed values of N and M.

When $M = 1$, $\mathbb{P}(E_0) = \frac{w_0}{\sum_{i=0}^{N-1} w_i}$.

When $N = 3$ and $M = 2$, $\mathbb{P}(E_0) = -\frac{w_0}{w_0 + w_1 + w_2} + \left(\frac{w_0}{w_0 + w_1} + \frac{w_0}{w_0 + w_2} \right)$.

When $N = 4$ and $M = 2$, $\mathbb{P}(E_0) = -\frac{2w_0}{w_0 + w_1 + w_2 + w_3} + \left(\frac{w_0}{w_0 + w_1 + w_2} + \frac{w_0}{w_0 + w_1 + w_3} + \frac{w_0}{w_0 + w_2 + w_3} \right)$.

When $N = 4$ and $M = 3$,

$$\mathbb{P}(E_0) = \frac{w_0}{w_0 + w_1 + w_2 + w_3} - \left(\frac{w_0}{w_0 + w_1 + w_2} + \frac{w_0}{w_0 + w_1 + w_3} + \frac{w_0}{w_0 + w_2 + w_3} \right) + \left(\frac{w_0}{w_0 + w_1} + \frac{w_0}{w_0 + w_2} + \frac{w_0}{w_0 + w_3} \right).$$

2.2 Dynamic Programming using a Polynomial

Let $W = \sum_{i=0}^{N-1} w_i$. To calculate the value of the equation (1), it is enough to calculate $\sum_{\substack{S \subseteq \{1,2,\cdots,N-1\} \\ |S|=k}} \frac{1}{w_0 + \sum_{i \in S} w_i}$ for each k with $N - M \leq k \leq N - 1$.

Hence, for each k and j with $N - M \leq k \leq N - 1$ and $0 \leq j \leq W$, it is enough to find the number of elements of $S_{k,j} = \left\{ S \subseteq \{1, 2, \cdots, N-1\} \mid |S| = k, \quad w_0 + \sum_{i \in S} w_i = j \right\}$.

Consider a polynomial $f(t, x) = t \cdot x^{w_0} \cdot \prod_{i=1}^{N-1} (1 + t \cdot x^{w_i})$. Then the coefficient of $t^k x^j$ in $f(t, x)$ is equal to $|S_{k,j}|$.

It is enough to calculate $\sum_{j=1}^{W} \frac{1}{j} |S_{k,j}|$, using $f(t, x) = \sum_{k=1}^{N} \sum_{j=1}^{W} |S_{k,j}| \, t^k x^j$.

[2] The implementation of this algorithm with $\mathcal{O}\left(N \cdot 2^N\right)$ time complexity and $\mathcal{O}(N)$ space complexity can be found in this link.

We know that $f(t,x)$ is a polynomial with NW or fewer monomial terms. If $t \cdot x^{w_0} \cdot \prod_{i=1}^{k-1} (1 + t \cdot x^{w_i})$ was calculated earlier, it is possible to calculate $t \cdot x^{w_0} \cdot \prod_{i=1}^{k} (1 + t \cdot x^{w_i}) = \left(t \cdot x^{w_0} \cdot \prod_{i=1}^{k-1} (1 + t \cdot x^{w_i}) \right) (1 + t \cdot x^{w_k})$ with $\mathcal{O}(NW)$ operations. Hence, it is possible to calculate all coefficients in $f(t,x)$ with $\mathcal{O}\left(N^2 W\right)$ time complexity and $\mathcal{O}\left(NW\right)$ space complexity by using dynamic programming. [3] Let $W_{\max} = \max_{0 \le i < n} w_i$. Then the time complexity can be also written as $\mathcal{O}\left(N^3 W_{\max}\right)$, anad the space complexity can be also written as $\mathcal{O}\left(N^2 W_{\max}\right)$. [4]
This algorithm is not useful when W is too big or w_i are not integers.

2.3 Miscellaneous

Let p be a prime number greater than $W = \sum_{i=0}^{N-1} w_i$.

For $S \subseteq \{1, 2, \cdots, N-1\}$, we have $0 < w_0 \le w_0 + \sum_{i \in S} w_i \le \sum_{i=0}^{N-1} w_i = W < p$. Hence the modular multiplicative inverse $\left(w_0 + \sum_{i \in S} w_i \right)^{-1}$ exists for each S.
To calculate the value of the equation (1) modulo p in section 2.1, the time complexity may be $\mathcal{O}\left(N \cdot 2^N \cdot \log p\right)$ if we calculate the modular multiplicative inverses without precomputation.
To calculate $j^{-1} \pmod p$ for each $j = 1, \cdots, W$, it is possible to calculate with time complexity $\mathcal{O}\left(W \log p\right)$. If we calculate $1!, 2!, \cdots, W!$ by using $k! = (k-1)! \cdot k$, it is possible to calculate $(W!)^{-1}$ with time complexity $\mathcal{O}\left(\log p\right)$, and calculate $((W-1)!)^{-1}, ((W-2)!)^{-1}, \cdots, (1!)^{-1}, (0!)^{-1}$ by using $((k-1)!)^{-1} = (k!)^{-1} \cdot k$. Using these values, it is possible to calculate $1^{-1}, 2^{-1}, \cdots, W^{-1}$ with time complexity $\mathcal{O}\left(W + \log p\right)$ using $k^{-1} = (k!)^{-1} \cdot (k-1)!$.
If the precomputation of $1^{-1}, 2^{-1}, \cdots, W^{-1}$ is allowed, the value of the equation (1) modulo p can be calculated with time complexity $\mathcal{O}\left(N \cdot 2^N + W + \log p\right)$ and space complexity $\mathcal{O}\left(N + W\right)$.

3 Conclusion

In this article, we provide a formula of the probability that a specific item is chosen in weighted sampling without replacement. If the weights are positive integers and the sum of weight is small enough, it is possible to use the algorithm in section 2.2.
Suppose that a programming contest have K problems with $1 \le K \le 20$, and there are N contestants with $1 \le N \le 100$. Suppose that a staff of this contest choose M contestants to give some gifts, where $1 \le M \le N$. Assume that the main progress of choosing contestants is the weighted sampling without replacement, which is introduced in this article. Suppose that the weight for each contestant is proportional to the square of solved problems. Then we have $W_{\max} \le K^2 \le 400$, thus $N^3 \cdot W_{\max} \le 4 \times 10^8$. Assume that there are few people who solve all problems, and most of people solve less than half of the K problems. Then the sum of weights may be not too big, and this algorithm in section 2.2 may be useful.

4 References

[1] Ben-Hamou, A., Peres, Y., & Salez, J. (2018). Weighted sampling without replacement. Brazilian Journal of Probability and Statistics, 32(3), 657-669.
[2] Güngör, M. (2024). On inclusion probabilities for weighted random sampling without replacement. Sigma Journal of Engineering and Natural Sciences, 42(6), 1780-1785.
[3] Efraimidis, P. S., & Spirakis, P. G. (2006). Weighted random sampling with a reservoir. Information processing letters, 97(5), 181-185.

[3] The implementation of this algorithm with $\mathcal{O}\left(N^2 W\right)$ time complexity and $\mathcal{O}\left(NW\right)$ space complexity can be found in this link.
[4] In the problem 35001 in Baekjoon Online Judge, since $1 \le N \le 20$ and $W_{\max} \le 10^4$, we have $N^3 W_{\max} \le 8 \times 10^7$. Hence it is possible to calculate using this algorithm.