Random subset - Mathematical theory

Alexandre AZOR

September 2024

Contents

I Naive algorithm

A Main idea

A simple algorithm to obtain a subset of k integers from $\llbracket 1, n \rrbracket$ consists in randomly choosing integers in this interval until k different ones are yielded.

B Description

Algorithm 1 Naive random subset algorithm

```
Initialisation:
subset := $\varnothing$

Subset contruction:
while size(subset) $< k$ do
    candidateInteger := random_integer_generator(1, n)
    if candidateInteger $\notin$ subset then
        subset $\leftarrow$ candidateInteger
```

C Complexity

1 Space complexity

In the previous pseudo-code, subset is a stack which size at the end of the algorithm is k.

$$\boxed{\text{Space complexity : } \Theta(k)}$$

2 Time complexity

For $i \in \llbracket 1, k \rrbracket$,
When choosing the i-th element of the subset, $i - 1$ integers have already been chosen, hence the probability that the integer yielded by the random generator has not been chosen yet is $\frac{n+1-i}{n}$

We generate random numbers until a new one is yielded. Let R_i be the number of random number generations necessary to obtain the n-th element of the subset. R_i follows a geometric distribution with parameter $\frac{n+1-i}{n}$.

$$R_i \rightsquigarrow \mathcal{G}\left(\frac{n+1-i}{n}\right)$$

Verifying if a generated number is already in subset is an operation in $\Theta(i)$ time complexity
The average time complexity is given by the following formula:

$$\begin{aligned}
\sum_{i=1}^{k} i \times \mathbb{E}[R_i] = \sum_{i=1}^{k} i \times \frac{n}{n+1-i} &= \sum_{j=n+1-k}^{n} (n+1-j) \times \frac{n}{j} \\
&= (n+1) \times \left(\sum_{j=n+1-k}^{n} \frac{1}{j} \right) - n \times \left(\sum_{j=n+1-k}^{n} 1 \right) \\
&= n\left((n+1)\left(H_n - H_{n-k}\right) - k\right)
\end{aligned}$$

where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the n-th harmonic number

If both n and $n - k$ are big enough, we can write

$$\boxed{\text{Average time complexity : } \mathcal{O}\left(n^2 \ln\left(\frac{n}{n-k}\right)\right)}$$

If we minor each $\mathbb{E}[R_i]$ by 1, we can minor the expression by $\frac{k(k+1)}{2}$. Hence

$$\boxed{\text{Average time complexity : } \Omega\left(k^2\right)}$$

II Suggested algorithm

III Main idea

A General Idea

The principle of the suggested algorithm is to choose at each step the next smallest element of the subset. The probability to pick an element is such that each ordered subset has the same probability to be constructed. The elements are yielded one by one in the increasing order by the algorithm.

B Notations

Let D_1 denotes the first yielded element. It is an integer from $[\![1, n+1-k]\!]$ as there are at least $k-1$ integer strictly greater than D_1 but lesser than or eqal to n.
X_1 is the smallest element of the subset.

$$X_1 = D_1$$

For $i \in [\![2, k-1]\!]$, let X_i be the i-th smallest element of the subset and D_i be the difference betweeen X_i and X_{i-1}.
Instead of constructing the $(X_i)_{i\in[\![1,k]\!]}$ sequence, we will instead construct the $(D_i)_{i\in[\![1,k]\!]}$ sequence and derive $(X_i)_{i\in[\![1,k]\!]}$ from it.

$$D_i = X_i - X_{i-1} \Leftrightarrow X_i = \sum_{j=1}^{i} D_j$$

With the assertion $X_0 = 0$ almost surely, the previous formulas remain true for $i = 1$.

IV Distribution function

A Range of D_i

For $i \in [\![1, k]\!]$, there are at least $k-i$ elements in $[\![X_i + 1, n]\!]$.

$$k - i \leq \mathrm{Card}\left([\![X_i + 1, n]\!]\right)$$

$$\Leftrightarrow k - i \leq n - X_i$$

$$\Leftrightarrow k - i \leq n - X_{i-1} + X_{i-1} - X_i$$

$$\Leftrightarrow k - i \leq n - X_{i-1} - D_i$$

$$\Leftrightarrow D_i \leq n - X_{i-1} - k + i$$

Moreover, the $(X_i)_{i\in[\![0,k]\!]}$ sequence is strictly increasing by definition. Hence $1 \leq D_i$

$$\boxed{D_i \in [\![1, n - X_{i-1} - k + i]\!]}$$

B Probability density function

1 D_1

We want each oredered subset to have the same probability of being chosen. To achieve this goal, the probability of each $l \in [\![1, n-k+1]\!]$ will be proportional to the number of subsets containing l as their first element.

For $l \in [\![1, n-k+1]\!]$, the number of subsets of $[\![1, n]\!]$ containing k elements and having l as their smallest elements is exactly the number of ways to choose $k-1$ different numbers from $[\![l+1, n]\!]$.

$$\exists \alpha \in \mathbb{R}_+^*, \forall l \in [\![1, n-k+1]\!], \quad \mathbb{P}(D_1 = l) = \frac{\binom{n-l}{k-1}}{\alpha}$$

$$\begin{aligned} & D_1 \in [\![1, n-k+1]\!] \text{ a.s} \\ \Leftrightarrow\ & \mathbb{P}(D_1 \in [\![1, n-k+1]\!]) = 1 \\ \Leftrightarrow\ & \sum_{l=1}^{n-k+1} \mathbb{P}(D_1 = l) = 1 \\ \Leftrightarrow\ & \sum_{l=1}^{n-k+1} \frac{\binom{n-l}{k-1}}{\alpha} = 1 \\ \Leftrightarrow\ & \sum_{l=1}^{n-k+1} \binom{n-l}{k-1} = \alpha \\ \Leftrightarrow\ & \sum_{m=k-1}^{n-1} \binom{m}{k-1} = \alpha \\ \Leftrightarrow\ & \binom{n}{k} = \alpha \\ \Leftrightarrow\ & \alpha = \binom{n}{k} \end{aligned}$$

$$\boxed{\forall l \in [\![1, n-k+1]\!], \quad \mathbb{P}(D_1 = l) = \frac{\binom{n-l}{k-1}}{\binom{n}{k}}}$$

2 D_i

Let $\mathcal{D}(n,k)$ be the probability distribution followed by D_1.

For $i \in [\![2, k]\!]$, D_i actually follows a law similar to D_1 but with different parameters. Let n_i and k_i be those parameters
As there are $k-i+1$ integers left to choose, we can easily see why $k_i = k-i+1$
The maximum potential value for D_i should be $n - X_{i-1} - k + i$ and if $D_i \rightsquigarrow \mathcal{D}(n_i, k_i)$, the maximum potential value is $n_i - k_i + 1$.

$$\begin{aligned} & n - X_{i-1} - k + i = n_i - k_i + 1 \\ \Leftrightarrow\ & n - X_{i-1} - k + i = n_i - k + i - 1 + 1 \\ \Leftrightarrow\ & n_i = n - X_{i-1} \end{aligned}$$

$$\boxed{D_i \rightsquigarrow \mathcal{D}(n - X_{i-1}, k-i+1)}$$

3 Verification

Let us check that having $D_i \rightsquigarrow \mathcal{D}(n - X_{i-1}, k-i+1)$ for all $i \in [\![1, k]\!]$ leads to each ordered subset to be picked equiprobably.

Let $(x_1, ..., x_k)$ be an ordered subset of $[\![1, n]\!]$ $(x_1 < ... < x_k)$

$$\begin{aligned} \mathbb{P}(X_1 = x_1, ..., X_k = x_k) &= \mathbb{P}(X_1 = x_1) \frac{\mathbb{P}(X_1 = x_1, ..., X_k = x_k)}{\mathbb{P}(X_1 = x_1)} \\ &= \mathbb{P}(X_1 = x_1) \prod_{i=2}^{k} \frac{\mathbb{P}(X_1 = x_1, ..., X_i = x_i)}{\mathbb{P}(X_1 = x_1, ..., X_{i-1} = x_{i-1})} \\ &= \mathbb{P}(X_1 = x_1) \prod_{i=2}^{k} \mathbb{P}\left(X_i = x_i | X_1 = x_1, ..., X_{i-1} = x_{i-1}\right) \end{aligned}$$

$$
\begin{aligned}
&= \mathbb{P}(X_1 = x_1) \prod_{i=2}^{k} \mathbb{P}\left(D_i = x_i - x_{i-1} | X_1 = x_1, ..., X_{i-1} = x_{i-1}\right) \\
&= \mathbb{P}(X_1 = x_1) \prod_{i=2}^{k} \mathbb{P}\left(D_i = x_i - x_{i-1} | X_{i-1} = x_{i-1}\right) \\
&= \frac{\binom{n-x_1}{k-1}}{\binom{n}{k}} \prod_{i=2}^{k} \frac{\binom{n-x_{i-1}-(x_i-x_{i-1})}{k-i}}{\binom{n-x_{i-1}}{k-i+1}} \\
&= \frac{\binom{n-x_1}{k-1}}{\binom{n}{k}} \prod_{i=2}^{k} \frac{\binom{n-x_i}{k-i}}{\binom{n-x_{i-1}}{k-(i-1)}} \\
&= \frac{\binom{n-x_1}{k-1}}{\binom{n}{k}} \frac{\binom{n-x_k}{k-k}}{\binom{n-x_{2-1}}{k-2+1}} \\
&= \frac{\binom{n-x_1}{k-1}}{\binom{n}{k}} \frac{\binom{n-x_k}{0}}{\binom{n-x_1}{k-1}} \\
&= \frac{1}{\binom{n}{k}} \frac{\binom{n-x_1}{k-1}}{\binom{n-x_1}{k-1}} \\
&= \frac{1}{\binom{n}{k}}
\end{aligned}
$$

We have the expected result

C Cumulative distribution function

$$
\begin{aligned}
\forall l \in [\![1, n-k+1]\!] \quad \mathbb{P}(D_1 \leq l) &= 1 - \mathbb{P}(D_1 > l) \\
&= 1 - \sum_{q=l+1}^{n-k+1} \mathbb{P}(D_1 = q) \\
&= 1 - \frac{\sum_{q=l+1}^{n-k+1} \binom{n-q}{k-1}}{\binom{n}{k}} \\
&= 1 - \frac{\sum_{m=k-1}^{n-l-1} \binom{m}{k-1}}{\binom{n}{k}} \\
&= 1 - \frac{\binom{n-l}{k}}{\binom{n}{k}}
\end{aligned}
$$

$$
\boxed{\forall l \in [\![1, n-k+1]\!] \quad \mathbb{P}(D_1 \leq l) = 1 - \frac{\binom{n-l}{k}}{\binom{n}{k}}}
$$

D Computation

1 *Naive version*

Let F denote the cumulative distribution function associated to the $\mathcal{D}(n,k)$ distribution.
Let U be a random variable following a continuous uniform distribution.
Let D be a random variable such that

$$
D = l \Leftrightarrow F(l-1) \leq U < F(l)
$$

As $U \in [0,1]$ and F is strictly increasing, D is always properly defined.

$$
\begin{aligned}
\forall l \in [\![1, n-k+1]\!], \quad \mathbb{P}(D = l) &== \mathbb{P}\left(F(l-1) \leq U < F(l)\right) \\
&= F(l) - F(l-1) \\
&= \mathbb{P}(D_1 \leq l) - \mathbb{P}(D_1 \leq l-1)
\end{aligned}
$$

$$\begin{aligned} &= \mathbb{P}(l-1 < D_1 \leq l) \\ &= \mathbb{P}(D_1 = l) \end{aligned}$$

$$\boxed{D \rightsquigarrow \mathcal{D}(n,k)}$$

Unfortunately, this method requires the computation of binomial coefficients which can hardly be done in an efficient way, especially when n is large.

2 *Actual method*

To counter the problem due to the size of the values to compute, we use the ln function to only deal with values of reasonable size.
To facilitate computation, we will manipulate $1 - F(l)$ instead of $F(l)$.
With the same notations as previously,

$$\begin{aligned} D = l &\Leftrightarrow F(l-1) \leq U < F(l) \\ &\Leftrightarrow 1 - F(l) \leq 1 - U < 1 - F(l-1) \\ &\Leftrightarrow \ln(1 - F(l)) \leq \ln(1-U) < \ln(1 - F(l-1)) \\ &\Leftrightarrow -\ln(1 - F(l-1)) \leq -\ln(1-U) < -\ln(1 - F(l)) \end{aligned}$$

Let $V = -\ln(1-U)$

$$\begin{aligned} &\forall v \in \mathbb{R}_+ \\ \mathbb{P}(V \leq v) &= \mathbb{P}(-\ln(1-U) \leq v) \\ &= \mathbb{P}(-v \leq \ln(1-U)) \\ &= \mathbb{P}(e^{-v} \leq 1-U) \\ &= \mathbb{P}(U \leq 1 - e^{-v}) \\ &= 1 - e^{-v} \end{aligned}$$

V follows an exponential distribution with parameter 1.

$$V \rightsquigarrow \mathcal{E}(1)$$

$$\boxed{D = l \quad \Leftrightarrow \quad -\ln(1 - F(l-1)) \leq V < -\ln(1 - F(l))}$$

3 *Rewriting the condition*

$$\begin{aligned} \forall l \in [\![1, n-k+1]\!], \quad F(l) &= 1 - \frac{\binom{n-l}{k}}{\binom{n}{k}} \\ \Leftrightarrow 1 - F(l) &= \frac{\binom{n-l}{k}}{\binom{n}{k}} \\ &= \frac{(n-l)!}{k!(n-l-k)!}\frac{k!(n-k)!}{n!} \\ &= \frac{(n-l)!}{(n-l-k)!}\frac{(n-k)!}{n!} \\ \Leftrightarrow -\ln(1-F(l)) &= -\ln((n-l)!) + \ln((n-l-k)!) - \ln((n-k)!) + \ln(n!) \end{aligned}$$

$$\text{Let } g : m \mapsto \ln(m!) - \ln((m-k)!)$$

Then

$$\boxed{-\ln(1 - F(l)) = -g(n-l) + g(n)}$$

$$\begin{aligned} D = l \quad &\Leftrightarrow \quad -g(n-l+1) + g(n) \leq V < -g(n-l) + g(n) \\ &\Leftrightarrow \quad -g(n-l+1) \leq V - g(n) < -g(n-l) \\ &\Leftrightarrow \quad g(n-l) \leq g(n) - V < g(n-l+1) \end{aligned}$$

$$\boxed{D = l \quad \Leftrightarrow \quad g(n-l) \leq g(n-k) - V < g(n-l+1)}$$

4 Approximations

If $m \in [\![1, 20]\!]$ we compute $\ln(m!)$ with the following formula:

$$\ln(m!) = \ln\left(\prod_{q=1}^{m} q\right) = \sum_{q=1}^{m} \ln(q)$$

If $m > 20$, we use the Striling's approximation

$$\ln(m!) \approx \left(m + \frac{1}{2}\right)\ln(m) - m + \frac{1}{2}\ln(2\pi) + \frac{1}{12m} - \frac{1}{360m^3}$$

5 Retrieve l

$$l \mapsto F(l) \text{ is strictly increasing}$$
$$l \mapsto 1 - F(l) \text{ is strictly decreasing}$$
$$l \mapsto -\ln\left(1 - F(l)\right) \text{ is strictly increasing}$$
$$l \mapsto g(n) - g(n-l) \text{ is strictly increasing}$$
$$l \mapsto -g(n-l) \text{ is strictly increasing}$$
$$l \mapsto g(n-l) \text{ is strictly decreasing}$$
$$l \mapsto g(l) \text{ is strictly increasing}$$

Hence, a dichotomy allows us to retrieve the value of l verifiying

$$g(n-l) \leq g(n) - V < g(n-l+1)$$

V Description

Algorithm 2 Random subset algorithm

```
Initialisation:
X := 0
n := $N$
nOrig := $N$
k := $K$
kOrig := $K$

Subset contruction:
for $i = 1 \to k$ do
    d := random_distance_generator(n, k)
    X := X + d
    Print X
    n := n - d
    k := k - 1

Return to initial state
X := 0
n := nOrig
k := kOrig
```

VI Complexity

A Space complexity

Only `n, nOrig, k, kOrig` and `X` are kept in memory.

$$\boxed{\text{Space complexity : } \mathcal{O}(1)}$$

B Time complexity

To generate a realisation of D_i, we need to find the value of d that verifies

$$g(n-d) \leq g(n) - V < g(n-d+1)$$

With the use of the Stirling approximation, the computation of the g function can be done in a $\mathcal{O}(1)$ complexity. As g is stricly increasing, a dichotomy algorithm can be use for that matter. As the value of d is within $[\![1, n]\!]$, for each generation of D_i, the generation of each one of them can be done in a $\mathcal{O}(\ln(n))$. Hence:

$$\boxed{\text{Time complexity : } \mathcal{O}(k \ln(n))}$$