Technical Details

By fanci@

Reviewer: jiayupeng@

[Link to the Google Docs version]

Overview

To evaluate the privacy of the “shuffling+flipping” algorithm for bloom filters, the “buffling” package package contains three parts: (I) signal-to-noise ratio evaluation, (II) privacy evaluation for one bloom filter, and (III) privacy evaluation for multiple BFs. Part I is included in the “signal_to_noise_ratio” module. It provides support for a simplified method for part III. Both part II and III are included in the “privacy” module. In particular, part II invokes two scientific computing helpers and generalizes to part III. In summary, Figure 1 shows the relationship among the three parts. In this document, we give the technical details for the three parts in order.

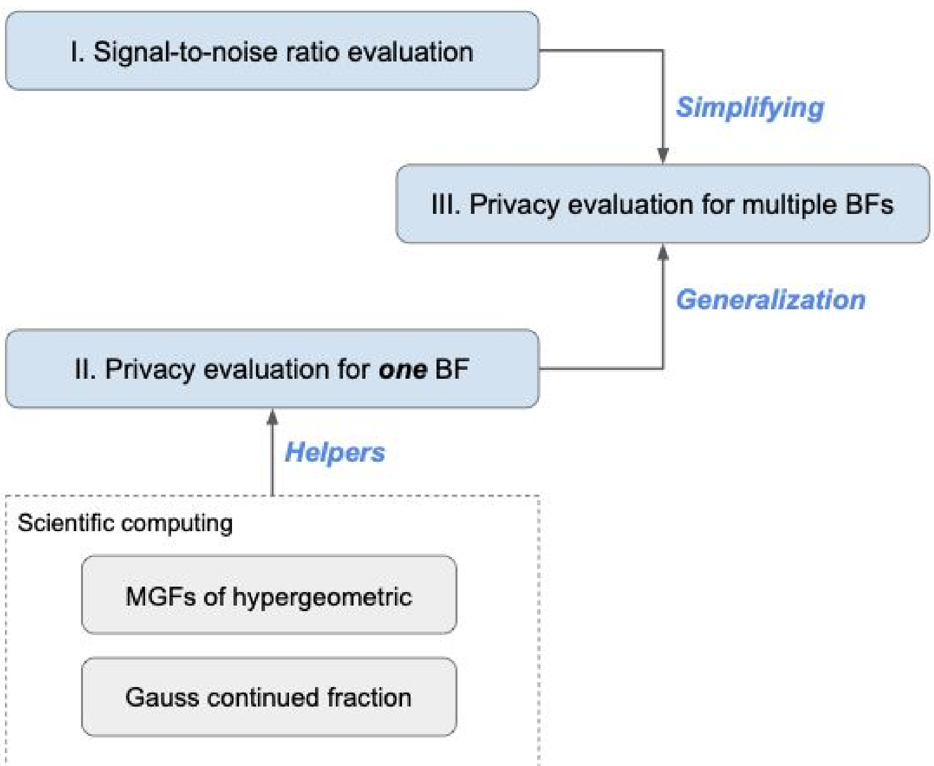

Figure 1: Organization of the “buffling” package.

I. Evaluate signal-to-noise ratio

This section describes the signal-to-noise ratio evaluation of the output logo-counts upon incremental changes.

Notations

- k BFs, each with length m and flipping probability p.
- Let $i = 0, 1, 2, ..., 2^k - 1$ index the logo of the bit matrix and let $bin(i)$ denote the binary representation of i. E.g., $bin(4) = (100)_2$.
- Let $P \in \Re^{2^k \times 2^k}$ contain the transition probability between logos, with $p_{ij} = p^{H(i,j)} q^{m-H(i,j)}$, where $H(a, b)$ is the hamming distance $bin(a)$ and $bin(b)$.
- Let $d \in \Re^{2^k}$ be the input logo-counts and let $x \in \Re^{2^k}$ be the output logo-counts.

Signal-to-noise ratio

We consider the linear combination of output logo-counts $c^T x$, for any $c \in \Re^{2^k}$. Consider incremental change of two logo-counts (shifting one count from logo j to logo i), i.e., $\Delta d = e_i - e_j$ for some $i \neq j$.

Definition (Signal-to-noise ratio). We define the signal-to-noise ratio of the incremental change $j \rightarrow i$ on the logo-count-array d as

$$SNR(d;\, j \rightarrow i) := \max_{c:||c||_2=1} E(\Delta c^T x)^2 \,/\, std(c^T x) = \max_{c:||c||_2=1} c^T A c \,/\, (c^T B c)\,,$$

where $A = P^T (e_i - e_j)(e_i - e_j)^T P = aa^T$ and $B = \Lambda(P^T d) - P^T \Lambda(d) P$ (see here for the derivation of the second equality), where $\Lambda(x)$ indicates the diagonal matrix with diagonal vector being x.

By Lagrange multiplier, $Bc = \lambda a a^T c = \gamma a$, γ being a scalar. For minimizing $c^T B c \,/\, c^T a a^T c$, can simply take $Bc = a$, where $a = P(e_i - e_j)$. (Note that $B\vec{1} = \vec{0}$, we would add a (location) constant to all elements in c such that $||c||_2 = 1$.)

Procedure (of SNR calculation)

(1) For $i, j = 0, 1, ..., 2^k - 1$,

   (a) Find the linear coefficient c as a solution of the linear system $Bc = a$

   (b) Normalize c to unit length (under the L_2-norm).

(2) Return the maximum SNR

Analysis of SNR

Observation 1. The SNR is increasing in the skewness of input logo-counts (see [this docs]() for more details about the skewness of input logo-counts).

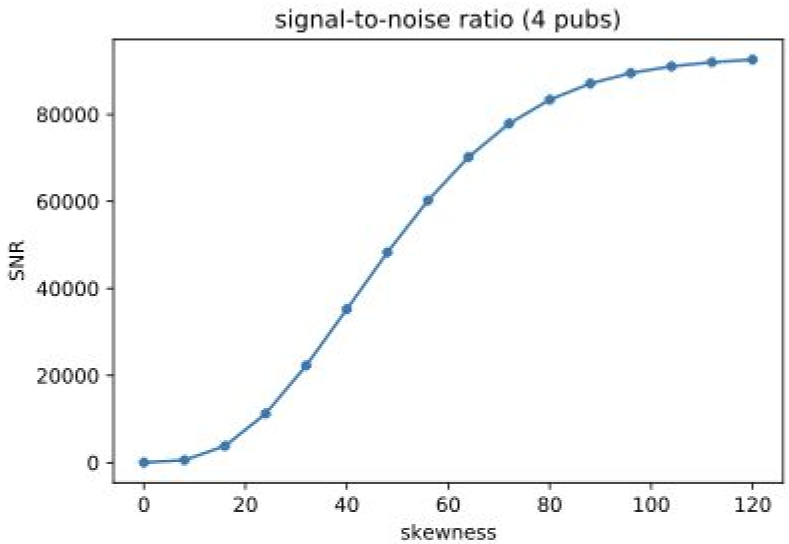

Figure 2: The relationship between the skewness of input logo-counts and the signal-to-noise ratio.

Observation 2. Fixing the input logo-counts D as described above, the SNR is maximized at the incremental change $0 \rightarrow 2^k - 1$ (Figure 3).

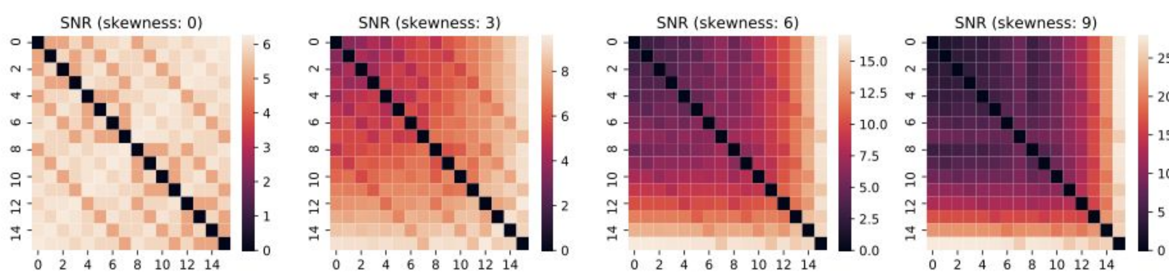

Figure 3: Heatmap of SNR. Each panel corresponds to one setting of a (which characterizes the skewness) with fixed input logo-counts D. For each setting, we calculate the SNR for all incremental change $j \rightarrow i$, for $i, j = 0, 1, ..., 2^k - 1$. Here, we have $k = 3$ publishers.

Observation 3. Fixing the input logo-counts D, $i = 0$ and $j = 2^k - 1$, the SNR maximizing coefficient c "concentrates" proportional to $(s,\ s,\ ...,\ s,\ -1)$. for some small $s > 0$.

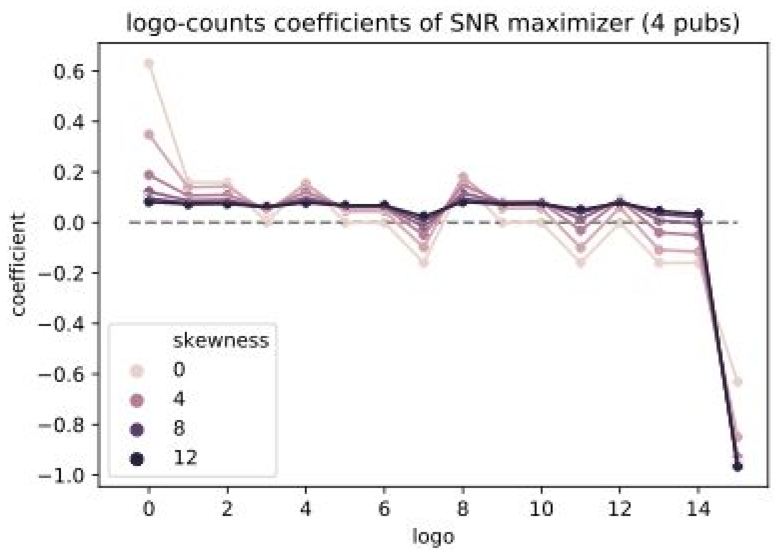

Figure 4: SNR maximizing coefficients of output logo-counts for different skewness of input logo-counts.

Conclusion. The maximal SNR occurs for the following two differentiate sets:

Dataset D has logo_count_dict = {bin(0): m, bin(i): 0 for any $1 \le i \le 2^k - 1$ };

Dataset D' has logo_count_dict = {bin(0): $m - 1$, bin($2^k - 1$): 1, bin(i): 0 for any other i }.

II. Evaluate the privacy for one bloom filter

This section describes the evaluation of privacy parameters for one bloom filter.

1. Main function: evaluate_privacy_for_one_bloom_filter

This function takes the following as input: the count of 1s in the input, the size of bloom filter, and the flipping probability. The output is an estimate of the privacy parameter of the “shuffling+flipping” algorithm.

Notations

Consider input bloom filter (BF) d (or d') of size m (i.e., the number of bits), and the output BF x. We denote the count of 1s in the input as y_d, and in the output as y_x. The flipping probability is p, and $q = 1 - p$.

Procedure

(1) Simulate the count of output 1s, $y_x = U + V$ (for `n_simu` times), where $U \sim Binomial(y_d,\ 1-p)$ and $V \sim Binomial(m - y_d,\ p)$ are independent.

(2) Compute the privacy loss $R(y_x) = (p/q) \cdot E[(q/p)^{2\xi_1}] / E[(q/p)^{2\xi_2}]$ or $R(y_x')$, where $\xi_1 \sim Hypergeometric(m, y_d + 1, y_x)]$ and $\xi_2 \sim Hypergeometric(m, y_d, y_x)]$. The ratio of two expectations is computed by calling ratio_of_hypergeometric_mgf.

(3) Output the max of (1) δ quantile of $\ln R(y_x)$ and (2) the $1-\delta$ quantile of $\ln R(y_x')$.

Derivation

Given two differentiated BFs d and d' (i.e., d and d' differ by exactly one bit), define the privacy loss as $R(y_x) = P(A(d') = x) / P(A(d) = x) = f(y_x \mid x = A(d')) / f(y_x \mid x = A(d))$.

Remark:

- $R(y_x \mid y_d)$ is a function of y_x, y_d
- Let $\varepsilon(y_d) = \max_{y_x} |\ln R(y_x|y_d)|$, then the final ε in ε-DP equals $\max_{y_d} \varepsilon(y_d)$.
- Considering δ, $\varepsilon(y_d;\ \delta)$ is defined as the max of (1) δ quantile of $\ln R(y_x|y_d)$ and (2) the $1-\delta$ quantile of $\ln R(y_x'|y_d')$.

Next, we obtain pmf from the moment-generating function (MGF). Note that $y_x = U + V$, where $U \sim Binomial(y_d,\ 1-p)$ and $V \sim Binomial(m - y_d,\ p)$ are independent. The MGF of y_x is $(p+qe^t)^{y_d} \times (q+pe^t)^{m-y_d} = \sum_{i=0}^{y_d} \binom{y_d}{i} p^{y_d-i} q^i e^{it} \times \sum_{j=0}^{m-y_d} \binom{m-y_d}{j} q^{m-y_d-j} p^j e^{jt}$. It follows that

$$f(y_x \mid x = A(d)) = Coefficient\ of\ the\ term\ e^{y_x t} = \sum_{i+j=y_x} \binom{y_d}{i} p^{y_d-i} q^i \binom{m-y_d}{j} q^{m-y_d-j} p^j$$

$$= \sum_i \binom{y_d}{i}\binom{m-y_d}{y_x-i} p^{y_d+y_x-2i} q^{m-y_d-y_x+2i} = p^{y_d+y_x} q^{m-y_d-y_x} \sum_{i=\max(0,\ y_d+y_x-m)}^{\min(y_d,\ y_x)} \binom{y_d}{i}\binom{m-y_d}{y_x-i} (q/p)^{2i}$$

$$= \binom{m}{y_x} p^{y_d+y_x} q^{m-y_d-y_x} \sum_{i=\max(0,\ y_d+y_x-m)}^{\min(y_d,\ y_x)} \left[\binom{y_d}{i}\binom{m-y_d}{y_x-i} \Big/ \binom{m}{y_x}\right] (q/p)^{2i}.$$

Hence, $f(y_x \mid x = A(d)) = \binom{m}{y_x} p^{y_d+y_x} q^{m-y_d-y_x} E[(q/p)^{2\xi} \mid \xi \sim Hypergeometric(m,\ y_d,\ y_x)]$. Similarly, $f(y_x \mid x = A(d))$ is obtained by replacing y_d with $y_d + 1$,
$f(y_x \mid x = A(d')) = \binom{m}{y_x} p^{y_d+1+y_x} q^{m-y_d-1-y_x} E[(q/p)^{2\xi} \mid \xi \sim Hypergeometric(m,\ y_d + 1,\ y_x)]$

Finally, the privacy loss take the form

$$R(y_x \mid y_d)$$

$$= f(y_x \mid x = A(d')) / f(y_x \mid x = A(d))$$

$$= (p/q) \times E[(q/p)^{2\xi} \mid \xi \sim HG(m,\ y_d+1,\ y_x)] / E[(q/p)^{2\xi} \mid \xi \sim HG(m,\ y_d,\ y_x)].$$

2. Helper function: ratio_of_hypergeometric_mgf

This function computes the ratio of two expectations:

$$\frac{E[(q/p)^{2\xi_1}]}{E[(q/p)^{2\xi_2}]}$$

where $\xi_1 \sim Hypergeometric(m,\ y_d+1,\ y_x)$ and $\xi_2 \sim Hypergeometric(m,\ y_d,\ y_x)$. The input includes the four parameters y_d (or count_input_ones), y_x (or count_output_ones), p, m (or n_bit).

Procedure

(1) Output $\frac{m-y_d-y_x}{m-y_d}\frac{{}_2F_1(a,b;c;z)}{{}_2F_1(a+1,b;c+1;z)}$,
where $\frac{{}_2F_1(a,b;c;z)}{{}_2F_1(a+1,b;c+1;z)}$ is obtained by invoking evaluate_privacy_of_one_bloom_filter,
$a = -y_d - 1$, $b = -y_x$, $c = m - y_x - y_d$, and $z = (q/p)^2$.

Derivation

Note that a hypergeometric distribution, $Hypergeometric(N, K, n)$, has the moment-generating function

$$\frac{\binom{N-K}{n}\,{}_2F_1(-n,-K;N-K-n+1;e^t)}{\binom{N}{n}}$$

where ${}_2F_1(a,b;c;z) = \sum_{n=0}^{\infty}\frac{(a)_n(b)_n}{(c)_n}\frac{z^n}{n!}$ is the hypergeometric function. Here $(q)_n$ is the (rising) Pochhammer symbol, which is defined by: $(q)_n = q(q+1)\cdots(q+n-1)$ if $n>0$ or 1 if $n=0$. We also have that ${}_2F_1(a,b;c;z) = {}_2F_1(b,a;c;z)$.

Hence, $E_{\xi_1}[(q/p)^{2\xi_1}] = \frac{(m-y_d-1)!(m-y_x)!}{m!(m-y_x-y_d-1)!}\,{}_2F_1(-y_d-1,\ -y_x;\ m-y_x-y_d;\ (q/p)^2)$ and
$E_{\xi_2}[(q/p)^{2\xi_2}] = \frac{(m-y_d)!(m-y_x)!}{m!(m-y_x-y_d)!}\,{}_2F_1(-y_d,\ -y_x;\ m-y_x-y_d+1;\ (q/p)^2)$.

Then $E_{\xi}[(q/p)^{2\xi_1}]\ /\ E_{\xi}[(q/p)^{2\xi_2}] = \frac{(m-y_d-y_x)\,{}_2F_1(a,b;c;z)}{(m-y_d)\,{}_2F_1(a+1,b;c+1;z)}$,
where $a = -y_d - 1$, $b = -y_x$, $c = m - y_x - y_d$, and $z = (q/p)^2$.

3. Helper function: evaluate_gauss_continued_fraction

This function approximates the Gauss' continued fraction,

$$\frac{{}_2F_1(a+1,b;c+1;z)}{{}_2F_1(a,b;c;z)}$$

Procedure

(1) Initialize $f_T = 1$ for some large enough number T (e.g. 1000)

(2) For i in $0, 1, \ldots, T$:

(a) If i is odd, $k_i = \frac{(b - c - (i+1)/2)\,(a + (i+1)/2)}{(c + i)\,(c + i + 1)}$

(b) If i is even, $k_i = \frac{(a - c - i/2)\,(b + i/2)}{(c + i)\,(c + i + 1)}$

(3) For i in $T - 1, \ ..., \ 1, \ 0 : f_i = 1 + k_i z \,/\, f_{i+1}$

(4) Output $1/f_0$

Derivation

$g_0(z) = {}_2F_1(a, b; c; z)$

$g_1(z) = {}_2F_1(a + 1, b; c + 1; z)$

$g_2(z) = {}_2F_1(a + 1, b + 1; c + 2; z)$

$g_3(z) = {}_2F_1(a + 2, b + 1; c + 3; z)$

$g_3(z) = {}_2F_1(a + 2, b + 2; c + 4; z)$

Then $g_{i-1} - g_i = k_i \, z \, g_{i+1}$, where

$d_1 = \frac{(a-c)b}{c(c+1)}$

$d_2 = \frac{(b-c-1)(a+1)}{(c+1)(c+2)}$

$d_3 = \frac{(a-c-1)(b+1)}{(c+2)(c+3)}$

$d_4 = \frac{(b-c-2)(a+2)}{(c+3)(c+4)}$

Hence, $\frac{{}_2F_1(a+1,\, b;\, c+1;\, z)}{{}_2F_1(a,b;c;z)} = \frac{g_1}{g_0} = \cfrac{1}{1+\cfrac{d_1 z}{1+\cfrac{d_2 z}{1+\cfrac{d_3 z}{1+\cfrac{d_4 z}{1+\cdots}}}}}$

III. Evaluate the privacy for multiple BF

This section describes the simplified evaluation of privacy parameters for multiple bloom filters.

1. Main function: evaluate_privacy_for_bloom_filter

Notations

- *Privacy loss* is defined as $R(X, D, D') = P(A(D') = X) \,/\, P(A(D) = X)$, where X is any output of the algorithm, and D and D' are two inputs that differ by one element (remarks).

- In (ε, δ)-differential privacy, we refer to ε as the *privacy parameter* and δ as the *approximation parameter* (see Dword et al. “The Algorithmic Foundations of Differential Privacy” (2014) for more further reference).

Procedure

(1) Precalculate $P_0 = stats.binom.pmf(k,\ p,\ y)$ and $P_k = stats.binom.pmf(k,\ q,\ y)$, for $y = 0, 1, ..., k$.

(2) Sample $g \sim Multinomial(m,\ P_0)$ and $g' \sim Multinomial(1,\ P_k) \oplus Multinomial(m-1,\ P_0)$.

(3) Estimate the PLD using $R(g) = \frac{1}{m} \sum_{y=0}^{k} (\frac{q}{p})^{2y-k} g_y$ and $R(g')$

(4) Output the max of (1) δ quantile of $\ln R(g)$ and (2) $1-\delta$ quantile of $\ln R(g')$.

Derivation

From the analysis of SNR, we conclude that the maximal SNR occurs for the following two differentiate sets:

- Dataset D has logo_count_dict = {bin(0): m, bin(i): 0 for any $1 \le i \le 2^k - 1$};
- Dataset D' has logo_count_dict = {bin(0): $m-1$, bin($2^k - 1$): 1, bin(i): 0 for any other i}.

For input logo-count distribution D and D', we defined the privacy loss as $P(A(D') = X) / P(A(D) = X)$ for any possible output X of the algorithm (remarks).

With these input logo-counts, the privacy loss is simple enough to write. Specifically, for any output logo_count_dictionnary $f = \{bin(i) :\ f_i\}$ subject to $\sum_{i=0}^{2^k-1} f_i = m$,

$$\Pr[A(D) = f]$$

$$= pmf\{f_0, f_1,\ \cdots, f_{2^k-1} \mid Multinomial(m,\ [p_{0\to 0},\ \cdots,\ p_{0\to 2^k-1}])\}$$

$$= \binom{m}{f_0, f_1, \cdots, f_{2^k-1}} \prod_{i=0}^{2^k-1} (p_{0\to i})^{f_i}.$$

On the other hand,

$$\Pr[A(D') = f]$$

$$= pmf\{f_0, f_1,\ \cdots, f_{2^k-1} \mid Multinomial(m-1,\ [p_{0\to 0},\ \cdots,\ p_{0\to 2^k-1}]) \oplus Multinomial(1,\ [p_{(2^k-1)\to 0},\ \cdots,\ p_{(2^k-1)\to 2^k-1}])\}$$

$$= \sum_{j=0}^{m} [p_{(2^k-1)\to j} \times \binom{m-1}{f_0, f_1, \cdots, f_{2^k-1}} \times f_j \times \prod_{i=0}^{2^k-1} (p_{0\to i})^{f_i} \times (p_{0\to j})^{-1}]$$

Thus, the privacy loss (PL) is

$$\Pr[A(D') = f] \,/\, \Pr[A(D) = f] = \sum_{j=0}^{m} (p_{(2^k-1)\to j} \,/\, p_{0\to j}) \times (f_j/m)\,.$$

Let $y(i)$ denote the number of ones in $bin(i)$, for any $0 \le i \le 2^k - 1$. Then,

$$p_{(2^k-1)\to j} = q^{y(j)} p^{k-y(j)} \text{ and } p_{0\to j} = q^{k-y(j)} p^{y(j)}\,,$$

under blipping with probability p. Hence, the PL is

$$\Pr[A(D') = f] \,/\, \Pr[A(D) = f]$$

$$= \sum_{j=0}^{m} (q/p)^{2y(j)-k} \times (f_j/m)$$

$$= (1/m) \sum_{y=0}^{k} [(q/p)^{2y-k} \times \sum_{j: y(j)=y} f_j] \text{ (grouping columns (or } j \text{ s) with the same \# of 1s)}$$

$$= \tfrac{1}{m} \sum_{y=0}^{k} (\tfrac{q}{p})^{2y-k} g_y\,,$$

where $g_y := \sum_{j: y(j)=y} f_j$, for $y = 0, 1, ..., k$, is the sufficient statistics for $f \in \Re^{2^k}$.

To simulate PLD, we need the prob dist of g. Generally, define $\pi \in \Re^{(k+1)\times(k+1)}$ with

$\pi_{y_1\to y_2} := \sum_{i:\, y(i)=y_1} \sum_{j:\, y(j)=y_2} p_{i\to j} = q^k \sum_{i:\, y(i)=y_1} \sum_{j:\, y(j)=y_2} (p/q)^{H(i,j)}$ for $y_1,\, y_2 = 0, 1, ..., k$ and $i,j = 0, 1, 2, ..., 2^k - 1$, where $H(i,j)$ is the hamming distance between $bin(i)$ and $bin(j)$. Let ξ be the number of positions in $bin(i)$ and $bin(j)$ that are both 1s. Then, $H(i,j) = y_1 + y_2 - 2\xi$. Note that $\xi \mid y_1, y_2$ follows Hypergeometric distribution with parameters $(k,\, y_1,\, y_2)$. It follows that

$$\sum_{i:\, y(i)=y_1} \sum_{j:\, y(j)=y_2} (q/p)^{-H(i,j)} \mid y_1, y_2 = (k-1)! \times (q/p)^{-y_1-y_2} \times E_\xi[(q/p)^{2\xi}]\,.$$

In particular, since each $p_{0\to j} = p^j q^{k-j}$, $\pi_{0\to y} = \sum_{j:\, y(j)=y} p_{0\to y} = \binom{k}{y} p^y q^{k-y} = stats.binom.pmf(k,\ p,\ y)$.

Similarly, $\pi_{k\to y} = \sum_{j:\, y(j)=y} p_{(2^k-1)\to j} = \binom{k}{y} q^y p^{k-y} = stats.binom.pmf(k,\ q,\ y)$.

2. Main function: estimate_flip_prob

Given the target privacy parameter ε, this function searches for the needed flipping probability in the "shuffling+blipping" algorithm, using a binary search schema.