Lecture notes: exercise session 2

October 11, 2018

1 Problem 1

Given:

- a data stream $X = (x_1, x_2, ..., x_m)$, $x_j \in [1, n]$, for all $j = 1 \ldots, m$
- Frequency of number i in X is $m_i = |\{j : x_j = i\}|$.

Question 1.1. How to compute m_i, for $i = 1, \ldots, n$ with one pass over X and available memory $\mathcal{O}(n \log m)$.

Question 1.2. Consider the CountSketch algorithm, which uses a hash function s that maps each number of $[1..n]$ to $\{+1, -1\}$ independently and uniformly at random.

1.1 Computing m_i with $\mathcal{O}(n \log m)$ memory

The idea is similar with counting sort: we create an array of the number of times each number between 1 and n occurs within the data stream. Once the number is read from data stream, the needed counter will be updated. The value of the counter cannot exceed the size of the data stream m, that's why we need $\mathcal{O}(\log m)$ space per counter and $\mathcal{O}(n \log m)$ in total.

1.2 Prove that $E[c \times s[i]] = m_i$ after CountSketch algorithm

Let's expand the expression $c \times s[i]$: $cs[i] = s[i] \sum_{j=1}^{m} s[x_j]$. It is known that i appears m_i times in the stream, therefore

$$s[i]\sum_{j=1}^{m} s[x_j] = s[i](m_i s[i] + \sum_{\substack{j=1 \\ x_j \neq i}}^{m} s[x_j])$$

$$= m_i + s[i] \sum_{\substack{j=1 \\ x_j \neq i}}^{m} s[x_j]$$

$$\Rightarrow E[c \times s[i]] = m_i + E\left[s[i] \sum_{\substack{j=1 \\ x_j \neq i}}^{m} s[x_j] \right]$$

For the 2nd term, we have:

$$E\left[s[i] \sum_{\substack{j=1 \\ x_j \neq i}}^{m} s[x_j] \right] = s[i] E\left[\sum_{\substack{j=1 \\ x_j \neq i}}^{m} s[x_j] \right]$$

$$= s[i] \sum_{\substack{j=1 \\ x_j \neq i}}^{m} E\left[s[x_j]\right] \text{ (by linearity of expectation)}$$

$$= 0 \text{ (since } E[s[j]] = 0 \text{ for each } j.)$$

Thus, $E[c \times s[i]] = m_i$.

2 Problem 2

Consider $X_1, \ldots, X_k$ independent and identically distributed random variables with $E[X_i] = \mu$ and $Var[X_i] = \sigma^2$, and their average $Y = \frac{1}{k}\sum_i X_i$.

Show that

$$E[Y] = \mu \quad \text{and} \quad Var[Y] = \frac{1}{k}\sigma^2.$$

Explain how you can use this result when you are designing randomized algorithms.

Solution:

$$E[Y] = \frac{1}{k}\sum_{i=1}^{k} E[X_i] = \frac{1}{k} \times k\mu = \mu$$

$$Var[Y] = \frac{1}{k^2}\sum_{i=1}^{k} Var[X_i] = \frac{1}{k^2} \times k\sigma^2 = \frac{\sigma^2}{k}$$

Let's imagine that we have sample $X_1, X_2, \ldots$ and $E[X_i] = \mu$, where μ is unknown, then the sample mean can estimate the mean. Moreover, the variance converges to 0, when $k \to \infty$. Therefore, we can understand the real value μ with high accuracy by calculating the mean for big sample.

3 Problem 3

Given:

- an algorithm `A` that can be used to solve a decision problem **P**, whose answer is in the set $\{\texttt{Y}, \texttt{N}\}$.
- `A` is *randomized*, and it has a probability of error $\frac{1}{8}$. The probability here is over the random bits drawn by the algorithm.

We want to *amplify* the probability of obtaining the correct solution.

We perform the following:

- We execute `A` k different times, where each execution draws independent random bits,
- and we return the majority of the answers.

Question : Show that if we select $k \geq c \log \frac{1}{\delta}$, for some constant c, then we can obtain the correct solution to **P** with probability $\geq 1 - \delta$, for any $\delta > 0$.

Solution :

So, let denote X_i the success of the algorithm **P** during i-th run . We know that X_i has the Bernoulli distribution with parameter $p = \dfrac{7}{8}$. We perform $k \geq c \log \frac{1}{\delta}$ trials and $X = \sum\limits_{i=1}^{k}$ shows the number of the correct answers that were made by **P** . Consequently, $\mu = E[X] = \dfrac{7}{8}k$.

Let's estimate the probability that the right answer will be given less than $\dfrac{k}{2}$ times with Chernoff bound (here $\gamma = \dfrac{3}{7}$):

$$
\begin{aligned}
Pr[X < \frac{k}{2}] &= Pr[X < (1-\gamma)\mu] \leq e^{-\frac{\gamma^2 \mu}{2}} \\
&= \exp\left(-\frac{\left(\frac{3}{7}\right)^2 \times \frac{7}{8}k}{2}\right) \\
&= \exp\left(-\frac{9}{112}k\right) \leq \exp\left(-\frac{9}{112}c \log \frac{1}{\delta}\right) \\
&= \delta^{\frac{9}{112}c} \\
&\leq \delta, \text{ when } c \geq \frac{112}{9}
\end{aligned}
$$

Therefore, when $c \geq \frac{112}{9}$ the probability that the right answer will be in majority is at least $1 - \delta$. In the table below you can find the values of k in accordance with the confidence level that you want to receive:

δ	$\lceil c \log \frac{1}{\delta} \rceil$
0.1	29
0.05	38
0.01	58
0.001	86

4 Problem 4

Design another algorithm for counting distinct elements in a data stream, *different* than the algorithm by Flajolet and Martin.

Consider a data stream $X = \{x_1, x_2, \ldots\}$, potentially infinite, where each $x_i \in U = \{1, \ldots, N\}$, N is very large

We can access a family of hash functions $\mathcal{F} : U \to [0, 1]$, s.t. any f drawn from $\mathcal{F}$ maps each item in U to a float in the interval $[0, 1]$, drawn from the *uniform distribution*.

Question:
If you can

- keep in memory *only two* floats
- use a hash function drawn from $\mathcal{F}$,

how to estimate the number of distinct elements that you have seen in the X.

Solution:

Let denote f the hash function that we use. We are going to keep in memory two floats: $Y = \min_i f(x_i)$ and $Z = \max_i f(x_i)$. If we have k distinct elements in the data stream, it can mean that we observe k independent and different (we consider that the probability of collision is zero) hashes which are distributed uniformly ($\sim U[0; 1]$). Let's try to estimate the expected value of Y and Z, but at first, we could try to find the probability density functions for them.

$$G(z) = Pr(Z \leq z) = \prod_{i=1}^{k} Pr(x_i \leq z) = z^k \text{ (cumulative function)}$$

$$\Rightarrow g(z) = G'(z) = kz^{k-1}, \text{ when } z \in [0;1] \text{ (density function)}$$

$$\Rightarrow E[z] = \int_0^1 zg(z)dz$$

$$= k \int_0^1 z^k dz$$

$$= \frac{k}{k+1}$$

The probability density function $h(y)$ for Y can be received in the similar way:

$$H(y) = Pr(Y \leq y) = 1 - Pr(Y \geq y)$$

$$= 1 - \prod_{i=1}^{k} Pr(x_i \geq y)$$

$$= 1 - (1-y)^k$$

$$\Rightarrow h(y) = H'(y) = k(1-y)^{k-1}, \text{ when } y \in [0;1]$$

$$\Rightarrow E[y] = \int_0^1 yg(y)dz$$

$$= k \int_0^1 z(1-z)^{k-1} dz$$

$$= \begin{vmatrix} u = z & du = dz \\ dv = (1-z)^{k-1} & v = -\frac{(1-z)^k}{k} \end{vmatrix}$$

$$= k \left(-\frac{z(1-z)^k}{k} \Big|_0^1 + \frac{1}{k} \int_0^1 (1-z)^k dz \right)$$

$$= \int_0^1 (1-z)^k dz$$

$$= -\frac{(1-z)^{k+1}}{k+1} \Big|_0^1 = \frac{1}{k+1}$$

Even if we could keep in the memory only the minimum Y, we could estimate the number of distinct elements as $k = \frac{1}{Y} - 1$. However, please note that

$E\left[\frac{1}{X}\right] \neq \frac{1}{E[x]}$ (you can verify it!). Therefore, $\frac{1}{Y} - 1$ is just a reasonable estimate of k, but the expected value of $E\left[\frac{1}{Y} - 1\right] \neq k$.

In our case the ratio of $\frac{Z}{Y}$ could be a reasonable estimate of k.