Exercises on Bottom-k Sampling, Frequency, and Set Similarity

Mikkel Thorup

1 Hash functions for sampling

The simplest use of hashing for sampling is to pick all keys that hash below a certain threshold (c.f. [2, §3.1]). In the context of sampling, it is often convenient to consider hash functions $h : U \to [0, 1)$ mapping the key universe U to the unit interval $[0, 1)$, that is, for any $x \in U$, $0 \leq h(x) < 1$. In practice we may have a strongly universal hash function $h_m : U \to [m]$ for some large m, and then we define $h(x) = h_m(x)/m$.

Exercise 1 *(a) If $p \geq 100/m$, show that $p \leq \Pr[h_m(x)/m < p] < 1.01p$. (the interpretation of $p \geq 100/m$ is that when we for an application pick m and h_m for sampling, we should pick m so large that $100/m$ is smaller than any sampling probability p that we expect to use).*

(b) If $A \subseteq U$ and $m \geq 100|A|^2$, bound the probability that there are two keys $x, y \in A$ which get the same hash value $h_m(x)/m = h_m(y)/m$.

Below, for simplicity, we assume that we have a strongly universal hash function $h : U \to [0, 1)$. In particular we assume that h is collision free and that $\Pr[h(x) < p] = p$ for any given $p \in [0, 1)$. Define the sample of a key set $A \subseteq U$ as

$$S_{h,p}(A) = \{x \in A | h(x) < p\}.$$

For a given (non-random) p, we refer to this as *threshold sampling*. As described in [2, §3.1], we get

Lemma 1 *For a given $p \in [0, 1)$ and set $A \subseteq U$, $n = |A|$, let $X = |S_{h,p}(A)|$. Then $\mu = \mathsf{E}[X] = pn$ and $\sigma^2 = \mathsf{Var}[X] = (1 - p)\mu \leq \mu$. In particular, for any $r > 0$,*

$$\Pr[|X - \mu| \geq r\sqrt{\mu}] \leq 1/r^2. \tag{1}$$

2 Bottom-k sampling

One issue with the above threshold sampling is that the number of samples is a variable depending on the size of the set A that we sample from. In many applications, we want to specify a hard limit k on the number k of samples. A simple solution is to store a so-called *bottom-k sample*:

$$S_h^k(A) = \{\text{the } k \text{ keys } x \in A \text{ with the smallest hash values}\}$$

Here we assume that A has at least k keys and that there are no collisions between hash values from A.

If h was a truly random hash function, then $S_h^k(A)$ would be a uniformly random subset of A of size k. The reason is that the hash values would be distributed randomly between the keys in A, so any size-k subset would have exactly the same probability of getting the k smallest hash values. In particular each $x \in A$ would have exactly the same probability $p = k/n$ of belonging to $S_h^k(A)$.

2.1 Frequency estimation

Often the point of sampling is that we later want to estimate the frequency of a subset $C \subseteq A$ as the frequency of C among the k samples. That is, we estimate the frequency $f = |C|/|A|$ as $|C \cap S_h^k(A)|/k$.

Exercise 2 *Assuming that $S_h^k(A)$ is a uniformly random size-k subset of A, prove that $E[|C \cap S_h^k(A)|/k] = |C|/|A|$.*

Exercise 3 *A common context in which bottom-k samples is applied is that the keys from A arrive online as a stream $x_1, ..., x_n$.*

(a) What kind of data structure would you use to maintain the bottom-k sample as the keys arrive, that is, when you have received keys $x_1, ...x_i$, you should have their sample $S_h^k(\{x_1, ...x_i\})$?

(b) How long time would it take you to process the next key x_{i+1}.

2.2 Similarity estimation

As discussed in [2, Section 3.1], one of the important points in sampling is that we want to compare sets A and B via their samples. We will use bottom-k samples to estimate their so-called Jaccard similarity $|A \cap B|/|A \cup B|$. This is the frequency of the intersection $A \cap B$ inside the union $A \cup B$.

Exercise 4 *Assume that we have the bottom-k samples $S_h^k(A)$ and $S_h^k(B)$.*

(a) Prove that $S_h^k(A \cup B) = S_h^k(S_h^k(A) \cup S_h^k(B))$.

(b) Prove that $A \cap B \cap S_h^k(A \cup B) = S_h^k(A) \cap S_h^k(B) \cap S_h^k(A \cup B)$.

(c) Assuming that h is truly random and collision free, it now follows from Exercise 2 that

$$|S_h^k(A) \cap S_h^k(B) \cap S_h^k(S_h^k(A) \cup S_h^k(B))|/k$$

is an unbiased estimator of the Jaccard similarity $|A \cap B|/|A \cup B|$. How long time would it take you to compute the above estimate from $S_h^k(A)$ and $S_h^k(B)$. You may assume that $S_h^k(A)$ and $S_h^k(B)$ are sorted according to hash value.

3 Bottom-k sampling with strong universality

We are now going to bound the error probability of bottom-k estimates when based on a strongly universal hash function $h : U \to [0, 1)$. We are given the bottom-k sample $S = S_h^k(A)$ of a set A. We wish to use the sample S to estimate the frequency $f = |C|/|A|$ of a given subset $C \subseteq A$ as $|C \cap S|/k$. Using that $h : U \to [0, 1)$ is strongly universal, we will prove for any $r \leq \bar{r} = \sqrt{k}/3$ that

$$\Pr\left[|C \cap S|/k > f + 3r\sqrt{f/k}\right] \leq 2/r^2. \tag{2}$$

Note how increasing the number k of samples decreases the error $3r\sqrt{f/k}$.

There is a symmetric bound for under estimates

$$\Pr\left[|C \cap S|/k < f - 3r\sqrt{f/k}\right] \leq 2/r^2, \tag{3}$$

but it will not be proved during this assignment.

3.1 A union bound

For positive parameters $a < 1$ and b to be chosen later, we will bound the probability of the overestimate

$$|C \cap S| > \frac{1+b}{1-a} fk. \tag{4}$$

Define the threshold probability

$$p = \frac{k}{n(1-a)}.$$

Note that p is defined deterministically from the input, independent of any samples. You will prove that the overestimate (4) implies at least one of the following two threshold sampling events:

(I) The number of elements from A that hash below p is less than k.

(II) The number of elements from C that hash below p is more than $(1+b)p|C|$.

Exercise 5 *Prove that if (I) and (II) are both false, then so is (4). As a first step, note that when (I) is false, all elements from the bottom-k sample S must hash below p.*

By Exercise 5, if (4) is true, then so is (I) or (II). By union, the probability that (I) and (II) is true is bounded by the sum of their probabilities, so we have

Proposition 2 *The probability $P_{(4)}$ of the overestimate (4) is bounded by $P_{(I)} + P_{(II)}$ where $P_{(I)}$ and $P_{(II)}$ are the probabilities of the events (I) and (II), respectively.*

3.2 Upper bound with 2-independence

For any given $r \leq \sqrt{k}/3$, we will fix a and b to give a combined error probability of $2/r^2$. More precisely, we will fix $a = r/\sqrt{k}$ and $b = r/\sqrt{fk}$. This also fixes $p = k/(n(1-a))$. Note that f is not known to the algorithm. However, $f = |C|/|A|$ is a number determined by the input, and for our mathemtical analysis, we are free to define a and b in terms of the f that we are trying to estimate. We note for later that $a \leq 1/3$ and $a \leq b$. This implies $(1+b)/(1-a) \leq (1+3b)$, so

$$(1+b)/(1-a) \leq (1+3b) = 1 + 3r/\sqrt{fk}. \tag{5}$$

In connection with (I) we study the number X_A of elements from A hashing below p. The mean is $\mu_A = \mathsf{E}[X_A] = pn = k/(1-a)$. Now (I) is true if and only if $X_A < k = \mu_A(1-a) = \mu_A(1 - r/\sqrt{k})$.

Exercise 6 *Use Lemma 1 to prove that*

$$P_{(I)} = \Pr[X_A < k] \leq 1/r^2.$$

In connection with (II) we study the number X_C of elements from C hashing below p. The mean is $\mu_C = p|C| = pfn = fk/(1-a) > fk$. Now (II) is true if and only if $X_C > \mu_C(1+b)$ where $b = r/\sqrt{fk}$.

Exercise 7 *Use Lemma 1 to prove that*

$$P_{(II)} = \Pr[X_C > (1+b)\mu_C] \leq 1/r^2.$$

By Proposition 2 we conclude that the probability $P_{(4)}$ of (4) is bounded by $P_{(I)} + P_{(II)} \leq 2/r^2$.

Using (5) for the inequality below, we now complete the proof of (2):

$$\begin{aligned}\Pr\left[|C \cap S|/k > f + 3\sqrt{f/k}\right] &= \Pr\left[|C \cap S| > (1 + 3r/\sqrt{fk})fk\right] \\ &\leq \Pr\left[|C \cap S| > \frac{1+b}{1-a}\, fk\right] \\ &= P_{(4)} \leq P_{(I)} + P_{(II)} \leq 2/r^2.\end{aligned}$$

Note This assignment is taken from [1] that contains a lot more material, including the proof of (3). The assignment is mostly asking questions about details not explained in [1], so you should only read [1] if you want to know more about this kind of work.

References

[1] M. Thorup. Bottom-k and priority sampling, set similarity and subset sums with minimal independence. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 371–380, 2013.

[2] M. Thorup. High speed hashing for integers and strings, 2014.