COMS 4995-3: Advanced Algorithms — Jan 30, 2017

Lecture 4 – Sketching: Distinct Elements Count.

Instructor: *Alex Andoni* — Scribes: *Lampros Flokas, Jiao Li*

1 Last time

Last time we covered two algorithms related to hashing:

1. 2-level-hashing: Here each element is mapped to a first level hash function to a bucket i and then all n_i elements are stored in another set of buckets using a different hash function. This requires $O(m)$ space.
2. Power of 2-choices: Here each item is mapped to 2 buckets and it is assigned to the least loaded bucket. For m, the number of buckets, equal to n the maximum load is $\Theta(\frac{\log n}{\log\log n})$

2 Introduction

Today's lecture is about distinct element count: Given, a stream of size m, $x_1, x_2, \ldots, x_m$, containing numbers from [n], meaning $x_i \in [n] = \{1, 2, \ldots, n\}$, we have to determine the number of elements with non-zero frequency. In other words we want to find the number of different x_i in the stream. There are many potential applications for this kind of algorithm. A motivating example could be finding the unique number of IP addresses of packets that a router processes in order to determine the existence of a Denial of Service Attack.

We can have the following deterministic and exact solutions:

- Use a hast table of size $\sim m$. This requires total space $\approx O(m \log n)$ bits. This is equivalent to storing the whole stream.
- If $n \ll m$ then we can have a bit-vector q of length n where $q_i = 1$ if $\exists j : x_j = i$ and $q_i = 0$ otherwise.

This result is near optimal for deterministic and exact algorithms. Even if we relax exactness the algorithms above are again near optimal. Therefore in order to reduce the needed space complexity we must resort to probabilistic and approximate algorithms.

3 Approximate and Random Solution

Once again to reduce the space complexity we will use the concept of the hash function, but this time in a different fashion from the previous lectures. We shall use a hash function $h : [n] \to [0, 1]$ (we will relax this requirement at the end of the lecture).

Algorithm 1: The Flajolet-Martin Algorithm

```
Data: stream $x_1, x_2, \ldots, x_m$
Result: # distinct elements of the stream
$z \leftarrow 1$;
for $i \leftarrow 1$ to $m$ do
    $z \leftarrow \min(z, h(x_i))$;
end
return $\frac{1}{z} - 1$
```

3.1 Algorithm [Flajolet-Martin 1985]

3.1.1 Intuition

The Flajolet-Martin algorithm is shown in Algorithm 1. To gain some intuition why the algorithm works let us make some observations. Firstly, when z is updated it can only get smaller or stay the same as it the right hand side of the update rule is a min of z and another value.

Observation 1. *As the algorithm progresses z cannot increase.*

This makes sense as our algorithm sees more of the stream and possibly encounters unseen elements it should only be able to increase its estimate and therefore decrease z. We can also make another observation. Notice that just before the ith iteration $\forall j < i : z \leq h(x_j)$. So a repeating value in the stream does not affect the value of z.

Observation 2. *z can decrease only if we see a new value in the stream.*

Our last observation is that the algorithm is at least intuitively able to track the number of different elements. Even with observation 2 we cannot guarantee that every new value will result in a decrease but since the hash value of every element is uniformly distributed in $[0, 1]$ the more new elements we see the more chances we have to draw a smaller number form h, decrease z and therefore give a higher output.

Observation 3. *The more unique values we see, the more likely it is to give a higher value.*

3.1.2 Proofs

Now, we shall prove the following claim.

Claim 4. *Let d be number of distinct elements in the stream $x_1, x_2, \ldots, x_m$. Then*

$$\mathbb{E}_h[z] = \frac{1}{d+1}$$

.

Proof. We can see that z is $\min_{i=1,\ldots,n} h(x_i)$. Based on our observations in section 3.1.1 we can reduce the minimum to the d unique elements of the stream $x_{j_1}, \ldots, x_{j_d}$. So now z is just a minimum of d random numbers in $[0,1]$. There are a lot of ways to calculate the expected value of $1/z$ but we shall give the following: Pick another random number $a \in [0,1]$.

$$\Pr(a \leq z) = \Pr(a \leq \min_{j=1,\ldots,d} x_{j_d}) = \Pr(a = \min(a, \{x_{j_d}\}_{j=1}^{d})) = \frac{1}{d+1} \tag{1}$$

Since the $d+1$ numbers are identically distributed. We can also see that

$$\Pr(a \leq z) = \int_0^1 \left(\int_0^x f_a(y)\,\mathrm{d}y \right) f_z(x)\,\mathrm{d}x = \int_0^1 x f_z(x)\,\mathrm{d}x = \mathbb{E}[z] \tag{2}$$

Since $f_a(y) = 1$ for $y \in [0,1]$ from uniformity. By combining equations 1, 2 we immediately prove the required result. □

However we now that generally

$$\mathbb{E}[\frac{1}{z}] \neq \frac{1}{\mathbb{E}[z]}.$$

So our estimator is not unbiased. But we will prove a concentration bound on z in order to get a lower and upper bound on our estimate. To get a concentration bound form Chebysev's inequality we shall need a bound on the variance. This is provided by the following claim:

Claim 5. $\mathrm{var}_h[z] \leq \frac{2}{d^2}$ *and therefore* $\sigma[z] \leq \frac{\sqrt{2}}{d}$.

Proof. It can be done in a similar fashion described in the previous theorem. Firstly like in equation 2 we have that $\mathbb{E}[z^2] = \Pr(a \leq z^2)$. To calculate $\Pr(a \leq z^2)$ we first calculate for a fixed $y \in [0,1]$

$$\Pr(y \leq z^2) = \Pr(\sqrt{y} \leq z) = \prod_{i=1}^{d} \Pr(\sqrt{y} \leq h(x_{j_i})) = (1 - \sqrt{y})^d \tag{3}$$

as h is fully random and so its values are independent. Moreover we get

$$\mathbb{E}[z^2] = \Pr(a \leq z^2) = \int_0^1 \Pr(y \leq z^2) f_a(y)\,\mathrm{d}y = \int_0^1 (1-\sqrt{y})^d\,\mathrm{d}y = \frac{2}{(d+1)^2+1} \leq \frac{2}{d^2}. \tag{4}$$

To conclude we know that $\mathrm{var}_h[z] = \mathbb{E}[z^2] - \mathbb{E}[z]^2 \leq \mathbb{E}[z^2] \leq \frac{2}{d^2}$ □

By Chebyshev we have with probability at least 1/2:

$$z \in \left[\frac{1}{d+1} - \frac{2\sqrt{2}}{d}, \frac{1}{d+1} + \frac{2\sqrt{2}}{d} \right] \tag{5}$$

We can get an equivalent interval for $1/z - 1$ but it will not be centred around d.

3.2 FM+ Algorithm

Just like with Morris Algorithm in the previous lecture we can linearly increase the space by a factor k and reduce the variance by factor $1/k$ which translates to a $(1+\epsilon)$ approximation of $1/(d+1)$ when $k = O(1/\epsilon^2)$. This leads to the FM+ Algorithm which is explained in algorithm 2.

Algorithm 2: The FM+ Algorithm

```
Data: stream $x_1, x_2, \ldots, x_m$, number of repeats $k$
Result: # distinct elements of the stream
for $i \leftarrow 1$ to $k$ do
    $z_i \leftarrow \mathrm{FM}(\{x_i\}_{j=1}^{m})$;
end
return $\frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{z_i} - 1\right)$
```

3.3 Alternative to FM+: Bottom-k

Instead of the FM+ Algorithm, which employs k hash functions we can use Bottom-k which updates the k minimal values seen up to the current point of just one run of the simple FM algorithm. Algorithm 3 shows the algorithm where the Kmin procedure returns the k smallest values from its input.

Algorithm 3: The Bottom-k Algorithm

```
Data: stream $x_1, x_2, \ldots, x_m$
Result: # distinct elements of the stream, number of minimal hashes $k$
$z \leftarrow (1, \ldots, 1)$;
for $i \leftarrow 1$ to $m$ do
    $z \leftarrow \mathrm{Kmin}(\{z_j\}_{j=1}^{k}, h(x_i))$;
end
return $\hat{d} = \frac{k}{z_{max}}$ where $z_{max} = \max\{z_1, \ldots, z_k\}$
```

Now we can prove that with high probability we have a $(1+\epsilon)$ approximation when $k = O(1/\epsilon^2)$.

Claim 6. *The following are true:*

$$\Pr[\hat{d} > (1+\epsilon)d] \leq 0.05$$
$$\Pr[\hat{d} < (1-\epsilon)d] \leq 0.05$$

Proof. For the first inequality, we define

$$X_i = \begin{cases} 1, & \text{if } h(x_{j_i}) < \frac{k}{(1+\epsilon)d} \\ 0, & \text{otherwise} \end{cases}$$

We can use this definition to simplify the requested probability.

Lemma 7. $\hat{d} > (1+\epsilon)d \iff \sum_{i=1}^{d} X_i > k$

Proof.

$$\begin{aligned}\sum_i X_i > k &\iff \exists \text{ at least k numbers for which } h(x_{j_i}) < \frac{k}{(1+\epsilon)d}\\ &\iff z_k < \frac{k}{(1+\epsilon)d}\\ &\iff \frac{k}{z_k} > (1+\epsilon)d\\ &\iff \hat{d} > (1+\epsilon)d\end{aligned}$$

□

It remains to analyze $\Pr[\sum_{i=1}^d X_i > k]$. We have,

$$\mathbb{E}[\sum_{i=1}^d X_i] = \sum_{i=1}^d \mathbb{E}[X_i] = \sum_{i=1}^d \frac{k}{(1+\epsilon)d} = \frac{k}{(1+\epsilon)},$$

and,

$$\text{var}[\sum_{i=1}^d X_i] = \sum_{i=1}^d \text{var}[X_i] = \sum_{i=1}^d (\mathbb{E}[X_i^2] - \mathbb{E}^2[X_i]) \le \frac{k}{1+\epsilon} \le k,$$

and the fact $\mathbb{E}[X_i^2] = \mathbb{E}[X_i]$, since $X_i \in \{0,1\}$ is used in the last step. By applying Chebushev's Inequality, we have

$$\Pr\left[\left|\sum_{i=1}^d X_i - \frac{k}{1+\epsilon}\right| > \sqrt{20k}\right] \le 0.05 \implies \Pr\left[\sum_{i=1}^d X_i > \frac{k}{1+\epsilon} + \sqrt{20k}\right] \le 0.05.$$

Setting $\epsilon < \frac{1}{2}$ and $k = \frac{c}{\epsilon^2}$, the rest probability

$$\begin{aligned}\sum_{i=1}^d X_i &\le \frac{k}{1+\epsilon} + \sqrt{20k}\\ &\le k(1-\epsilon+\epsilon^2) + \sqrt{20k} \quad \text{(by Taylor Series Expansion: } 1-\epsilon \le \frac{1}{1+\epsilon} \le 1-\epsilon+\epsilon^2)\\ &\le k \underbrace{-\frac{c}{\epsilon} + c + \sqrt{20}\frac{\sqrt{c}}{\epsilon}}_{\text{for big c, this term} < 0}\\ &\le k \quad (\text{set } c > 100)\end{aligned}$$

Therefore:

$$\Pr[\sum_{i=1}^d X_i > k] \le \Pr[\sum_{i=1}^d X_i > \frac{k}{1+\epsilon} + \sqrt{20k}] \le 0.05.$$

The proof of $\Pr[\hat{d} < (1-\epsilon)d] \le 0.05$ is similar. □

3.4 Picking the Hash function

Saving a hash function that maps to real numbers may be impractical and in fact it is unnecessary. In the proof of the Bottom-k Algorithm we did not invoke the uniformity property in $[0,1]$. So having a discrete hash function should give the same results. Our first solution is to construct a hash function $h_d : [n] \to \{0, \ldots, M\}$ and then let h be h_d/M so that it has a range in $[0,1]$. We shall also need $M \gg n^3$ if we want the probability that $d \leq n$ elements collide is less than $1/n$.

Additionally, in order to get a bound form Chebysev's inequality we only need to calculate the variance of $\sum_{i=1}^{d} X_i$. In the variance calculation only pairs of terms like X_1X_2 come up so it is of no interest for us what happens with higher order terms like $X_1X_2X_3$. As a result having a fully random hash function is superfluous. We only need a hash function where all samples are pairwise independent. We will call these functions 2-universal and we will define them as

Definition 8. *A hash function $h : [n] \to [M]$ is called 2-universal if*

$$\forall i, j \in [n], i \neq j, \forall a, b : \Pr[h(i) = a \wedge h(i) = b] = 1/M^2 \tag{6}$$

2-universality implies universality:

Claim 9. *If a hash function is 2-universal then it is also universal.*

Proof. For a 2-universal hash function h:

$$\begin{aligned}\forall i, j, i \neq j : \Pr[h(i) = h(j)] &= \sum_{a \in [M]} \Pr[h(i) = h(j) = a] \\ &= \sum_{a \in [M]} \Pr[h(i) = a \wedge h(j) = a] \\ &= \sum_{a \in [M]} 1/M^2 = M/M^2 = 1/M \end{aligned} \tag{7}$$

So h is also universal. □

4 Questions from Piazza

Clarifications about random hash functions: In order to talk about random hash functions we need a family of hash functions $h : [n] \to [M]$. Let us call this family H. Picking a hash function means taking an $h \in H$.

Example 10. *If we have a fully random hash function then the associated family is the family of all hash functions, namely $H = \{\text{all } h : [n] \to [M]\}$. A sample hash function from this family is indeed fully random as it can end up being any hash function from the set of all possible functions.*

When it comes to describing a random hash function, any hash function form H can be fully identified by its order in an enumeration of H and therefore requires $\log_2 |H|$ bits.