MIT 6.854/18.415: Advanced Algorithms Spring 2016

Lecture 4 – February 16, 2016

Prof. Ankur Moitra *Scribe: Ben Eysenbach, Devin Neal*

1 Last Time

- Consistent Hashing - hash functions that "evolve" well
- Random Trees - routing schemes that deal with inconsistent views

Today: Distinct Elements and Count-Min Sketch. "What can we do if we can't store data, only stream it?"

2 Distinct Elements

Problem: Count the number of distinct elements in a sequence $X_1, X_2, \ldots, X_n$. For example, how many unique words did Shakespeare use?

Naively this problem takes $O(N)$ space, where N is the number of distinct elements in the sequence. For Shakespeare's total vocabulary, $N \approx 35,000$. However, it turns out that you can do *much better* than the naive method if you are willing to accept some level of approximation. There's a famous quote from a 2003 paper by Marianne Durand and Philippe Flajolet:

> Using only memory equivalent to 5 lines of printed text, you can estimate with a typical accuracy of 5% and in a single pass the total vocabulary of Shakespeare.

2.1 Using a Single Hash Function

Idea: Choose a random hash function $h : \mathcal{U} \to [0, 1]$ and pass once through the data, hashing each item and storing only the minimum of $h(X_1), h(X_2), \ldots, h(X_i), \ldots$.

Let $Y \triangleq \min_i\{h(X_i)\}$ be the minimum, and let N be the true number of distinct elements.

Lemma 1.

$$\mathbb{E}[Y] = \frac{1}{N+1}$$

Proof.

$$\begin{aligned}\mathbb{E}[Y] &= \int_0^1 \mathbb{P}[Y \geq z]dz \\ &= \int_0^1 (1-z)^N dz \\ &= \Big[-\frac{(1-z)^{N+1}}{N+1} \Big]_0^1 \\ &= \frac{1}{N+1}\end{aligned}$$

□

With some thought, you can confirm that $\mathbb{E}[Y]$ is the same as the probability of choosing $N+1$ numbers in the interval $[0,1]$ and having the last number be the minimum. By symmetry, this probability is $\frac{1}{N+1}$.

So, if we estimate N by $\frac{1}{Y} - 1$, we'll at least get the right answer in expectation.

To show that we also get the right answer with good probability, we're going to use the Chebyshev tail bound to bound the probability that our estimate for Y is close to its expectation. To do this, we must first compute the variance of Y.

Lemma 2.

$$Var[Y] \leq \left(\frac{1}{N+1}\right)^2$$

Proof.

$$\begin{aligned}Var[Y] &= \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \\ &= \int_0^1 z^2 N(1-z)^{N-1} dz - \left(\frac{1}{N+1}\right)^2 \\ &= \frac{2}{(N+1)(N+2)} - \frac{1}{(N+1)(N+1)} \\ &\leq \frac{2}{(N+1)(N+1)} - \frac{1}{(N+1)(N+1)} \\ &= \left(\frac{1}{N+1}\right)^2\end{aligned}$$

□

Unfortunately, we cannot apply Chebyshev directly because the variance of Y is too large. In particular, Chebyshev only gives error "resolution" down to size of Y's standard deviation (i.e. $\sqrt{Var[Y]}$). Zero is one standard deviation below $\mathbb{E}[Y]$, so Chebyshev would only tell us that $\mathbb{P}[Y=0]$ is some constant. This is bad because we cannot solve $N = \frac{1}{Y} - 1$ when $Y = 0$. We want $\mathbb{P}[Y=0]$ to be very small.

2.2 k Hash Functions

Fortunately, we can apply the standard technique of reducing the variance of our estimate via repetition.

Idea [Flajolet-Martin]: Use k hash functions $h_1, \ldots, h_k : \mathcal{U} \to [0, 1]$.

Now, evaluate each hash function on each item in the sequence, storing the minimum for each hash function separately. Let $\bar{Y} \triangleq \frac{1}{k}\sum_{i=1}^{k} Y_i$ be the average minimum. The variance of the sum of independent random variables is the sum of their variances. Thus, $Var[\bar{Y}] = \frac{1}{k}\sum_{i=1}^{k} Y_i \leq \frac{1}{k(N+1)^2}$, where we've used Lemma 2 for the inequality.

Applying Chebyshev:

$$\mathbb{P}\left[|\bar{Y} - \frac{1}{N+1}| \geq \frac{\epsilon}{N+1}\right] \leq \frac{Var[\bar{Y}]}{\left(\frac{\epsilon}{N+1}\right)^2} \leq \frac{1}{k\epsilon^2}$$

Thus, our estimate for the number of distinct elements, $\frac{1}{\bar{Y}}$, satisfies $\frac{N+1}{1+\epsilon} \leq \frac{1}{\bar{Y}} \leq \frac{N+1}{1-\epsilon}$ with probability at least $1 - \frac{1}{k\epsilon^2}$. For small ϵ, this guarantee is equivalent to:

$$(1 - O(\epsilon))N \leq \frac{1}{Y} - 1 \leq (1 + O(\epsilon))N$$

We need to set $k = O(1/\epsilon^2)$ to get an ϵ accurate estimate with probability 9/10, for example.

In practice, we don't need our hash functions to map to arbitrary real numbers. It is sufficient to use length $O(\log n)$ binary strings. For more details, see [1].

3 Heavy Hitters

We can compute many statistics about a stream of data beyond the number of distinct elements. One particularly popular and practically useful goal is to find elements that appear frequently in the stream. These items are simply called "frequent items" or sometimes "heavy hitters".

3.1 Misra-Gries, 1982 [3]

We begin with a straight forward version of the heavy hitters problem:

Given a sequence of elements $X_1, X_2, ..., X_n$, output a list with at most k values, ensuring that every element which occurs at least $\frac{n}{k+1} + 1$ times in the sequence is on the list. Note that there can only be k such items, although there could be fewer. We allow false positives in the list.

Here's the algorithm:

```
initialize empty list
for each item
    if item on list
      increment its counter
    else if length(list) < k
```

```
    add item to list
    set item's counter to 1
  else
    throw away item
    decrement counter of every item in list
    delete items in list with counter = 0
```

Fact: Since the Misra-Gries algorithm stores k counters with value at most n, it uses $O(k \log n)$ space.

Lemma 3. *Let f_x denote the frequency of item x. When Misra Gries terminates, the counter for x is at least $f_x - \frac{n}{k+1}$.*

Note that x could have a counter equal to 0, in which case it will not be on the list.

Proof. The final value of x's counter is equal to the number of times it appears in our sequence, f_x, minus the number of times x was thrown out because the list was full at the time. We argue that this number can't be higher than $\frac{n}{k+1}$.

When an item is thrown out, each element contained in the list is decremented. Additionally, x's "virtual counter" is effectively decremented from 1 to 0. Since the entire list must be filled, this corresponds to $k+1$ tokens being destroyed every time x is thrown out. There are a total of n tokens received, so this event can occur at most $\frac{n}{k+1}$ times.

We conclude that the counter for x is at least $f_x - \frac{n}{k+1}$. So, if $f_x > \frac{n}{k+1}$, it will appear on the list at the end of the algorithm, as desired. □

3.2 Count-Min Sketch

We can also solve a more ambitious version of the heavy hitters problem:

Given a sequence $X_1, X_2, ..., X_n$, compute f_x to within additive error ϵn.

Note that an additive approximation is more meaningful for heavier items in the list and becomes meaningless when $f_x \leq \epsilon n$.

We will use the Count-Min Sketch (Cormode, Muthukrishnan 2005 [2]). Choose ℓ random hash functions $h_1, ..., h_\ell : \mathcal{U} \to \{1, 2, ..., b\}$. Initialize an $\ell \times b$ array $CMS[\ell][b]$ with zeros. Then, as elements X_i in the sequence are streamed in, for each hash function $h_j(\cdot)$, increment the $(j, h_j(X_i))$ entry of the table.

To estimate the frequency of item x, compute

$$\texttt{Count(x)} = \min_j \{ \quad CMS[j][h_j(x)] \quad \}$$

Claim 4. *For any fixed item x and index j, $CMS[j][h_j(x)] \geq f_x$.*

Proof. We increment $CMS[j][h_j(x)]$ each time we see x. However, this entry will be incremented if $h_j(y) = h_j(x)$ for some other item $y \neq x$ that also appears in the stream. Hence the inequality. □

3.2.1 Analysis

Let $z_j \triangleq CMS[j][h_j(x)]$, and note that $z_j = f_x + \sum_{y \neq x, h_j(y)=h_j(x)} f_y$. We want to examine the expected value of z_j:

$$\begin{aligned}
\mathbb{E}[z_j] &= f_x + \sum_{y \neq x} f_y \, \mathbb{P}[h_j(x) = h_j(y)] \\
&= f_x + \frac{1}{b} \sum_{y \neq x} f_y \\
&\leq f_x + \frac{1}{b} \sum_{y} f_y \\
&= f_x + \frac{n}{b}
\end{aligned}$$

Note that z_j is a *biased estimator*: its expected value is greater than f_x, the quantity we hope to estimate.

Now, we want to show that z_j is close to f_x. Using the Markov Bound when $b = \frac{2}{\epsilon}$,

$$\mathbb{P}[z_j \geq \epsilon n] \leq \frac{1}{2}$$

When we have ℓ hash functions, our estimate is the minimum z_j. Our estimate is bad iff every z_j is much larger than f_x:

$$\begin{aligned}
\mathbb{P}[(\min_j z_j) \geq f_x + \epsilon n] &= \mathbb{P}[\forall j, z_j \geq f_x + \epsilon n] \\
&\leq \frac{1}{2^\ell}
\end{aligned}$$

Setting $\ell = O(\log(n))$, we get $\mathbb{P}[(\min_j z_j) \geq f_x + \epsilon n] \leq \frac{1}{n}$. Accordingly, our estimate is accurate up to ϵn error with high probability.

3.2.2 Comparison with Misra Gries

By setting $\epsilon = \frac{1}{2k}$, the Count-Min sketch solves almost the same problem as Misra-Gries. The key difference is that the Count-Min sketch guarantees that for every x, if `Count(x)` is large then x is a heavy hitter. In Misra-Gries, an item on the returned list might not be a heavy hitter.

We pay for this guarantee in space. Misra-Gries takes $O(k \log(n))$ space, while the Count-Min sketch requires $O(k \log^2(n))$ space (setting $\epsilon = \frac{1}{2k}$ and noting that each counter stores a value of at most n). Also note that Count-Min only obtains its solution with high probability. Misra-Gries is deterministic and so it always succeeds.

References

[1] Flajolet, Philippe and Martin, Nigel. 1985. Probabilistic counting algorithms for data base applications. In *Journal of Computer and System Sciences*, Volume 31, Number 2. pp. 182–209.

[2] Cormode, Graham, and Muthukrishnan, S. 2005. An improved data stream summary: the count-min sketch and its applications. In *Journal of Algorithms*. pp. 58–75.

[3] Misra, Jayadev and Gries, David. 1982. Finding repeated elements. In *Science of Computer Programming*. pp. 143–152.