Dealing with Massive Data February 7, 2011

Lecture 3: Counting on Streams

Lecturer: Sergei Vassilvitskii *Scribe: Dawei Shi & Yundi Zhang*

1 How many times an element appear in a stream?

Problem 1 *Given a stream $\mathcal{X}$ on n elements total and a specific element i, we ask how many times did x_i appear in the stream?*

Formally, let f_j be the # of times element j appears. Then we can define frequency moments:

Definition 1 $F_i = \sum_j f_j^i$.

For example, $F_0 =$ # distinct values; $F_1 =$ length of the stream; $F_2 =$ self join size, etc. Note that the value of F_2 depends on the skew of the distribution of data.

2 Count Min Sketch

Consider the following simple algorithm. Let $h : X \mapsto [1, k]$ be a hash function. We will store estimate counts in a short array:

Algorithm 1 Estimate count

```
for all $x_i \in X$ do
    index $\leftarrow h(x_i)$
    count[index] $\leftarrow count[index] + 1$
```

Observe that Algorithm 1 provides an upper bound on the true counts due to potential hash value conflicts. For every element $x_j, count[h(x_j)] \geq f_j$.

To analyze the algorithm, denote by $y_{ij} =$ contribution of element x_j to count for x_i.

$$y_{ij} = \begin{cases} f_j & \text{if } h(i) = h(j) \\ 0 & \text{if } h(i) \neq h(j) \end{cases}$$

Note that by definition of the hash function, $h(i) = h(j)$ with probability $1/k$. Therefore, $\mathbb{E}[y_{ij}] = f_j/k$. Therefore, the expected total overcounting is:

$$\mathbb{E}\left[\sum_{j \neq i} y_{ij}\right] = \sum_{j \neq i} \frac{f_j}{k} \leq F_1/k.$$

We need one more tool to complete the analysis.

Theorem 1 *(Markov's Inequality) Let Z be a non-negative random variable, then $Pr\left[Z \geq a \cdot E[Z]\right] \leq \frac{1}{a}$.*

Suppose we set $k = \frac{2}{\epsilon}$.Then, we have:

$$\mathbf{Pr}\left[\sum_{j \neq i} y_{ij} > \epsilon F_1\right] \leq \frac{1}{2}.$$

2.1 Boosting the success probability

The above analysis tells us that the probability that the estimate is wrong by more than an additive ϵF_1 factor is at most $1/2$. To increase the probability of success even higher, consider using multiple hash functions. Let $h_1 : X \mapsto [1,k]$, $h_2 : X \mapsto [1,k], \ldots, h_\ell : X \mapsto [1,k]$ be ℓ independent hash functions. We run ℓ independent copies of the counter above.

We know that in every count array, $Pr[\sum y_{ij} > \epsilon \cdot F_1] \leq \frac{1}{2}$. Therefore,

$$\mathbf{Pr}\left[\forall counts \sum y_{ij} \geq \epsilon \cdot F_1\right] = \left(\frac{1}{2}\right)^\ell.$$

Therefore, is we look at the minimum estimate to $f_j : \min_\ell count_\ell[h(x_j)]$, it will be correct with probability $2^{-\ell}$. Putting it together, if we use $\ell = O(\log \frac{1}{\delta}$ arrays each with k/ϵ elements, then with probability $1 - \delta$ each element is counted correctly, up to an additive ϵF_1 factor.

3 Count Sketch

The drawback of Count Min Sketch is that errors always accumulate. In other words, we always overestimate. Suppose instead we try to have the errors cancel each other out. Let $g : X \mapsto \{-1, +1\}$ be a hash function such that for any $x \in X$, $\mathbf{Pr}[g(x) = 1] = \mathbf{Pr}[g(x) = -1] = 1/2$. Consider the following algorithm:

Algorithm 2 Count Sketch

```
for all $x_i \in X$ do
    $Count[h(x_i)] = Count[h(x_i)] + g(x_i)$
    $Z = Z + g(x_i)$
if Look up answer for $i$ then
    return $Count[h(i)]g(i)$
```

We first show that the algorithm produces the correct answer in expectation. Note that this is a stark difference from the CountMin algorithm which always overestimates the answer.

Lemma 1 *For any element, i, $\mathbb{E}\left[Count[h(i)] \cdot g(i)\right] = f_i$.*

Proof. Let S be the set of elements that map to the same bucket as i, that is for any $s \in S$, $h(s) = h(i)$. Then:

$$\begin{aligned}
\mathbb{E}\left[Count[h(i)] \cdot g(i)\right] &= \mathbb{E}\left[\sum_{s \in S} f_s g(s) g(i)\right] \\
&= \mathbb{E}\left[\sum_{s \in S, s \neq i} f_s g(s) g(i) + f(i) g(i) g(i)\right] \\
&= \sum_{s \in S, s \neq i} f_s \mathbb{E}[g(s)] \mathbb{E}[g(i)] + f_i \mathbb{E}[g(i)^2] \\
&= \sum_{s \in S, s \neq i} f_s \cdot 0 \cdot \mathbb{E}[g(i)] + f_i \mathbb{E}[1] \\
&= f_i
\end{aligned}$$

□

To see an alternative proof, as before let y_{ij} be the contribution of element j to the count of i. We have:

$$y_{ij} = \begin{cases} f_j \text{ if } h(i) = h(j) \text{ and } g(i) = g(j) \\ -f_j \text{ if } h(i) = h(j) \text{ and } g(i) \neq g(j) \\ 0 \text{ if } h(i) \neq h(j) \end{cases}$$

Since $g(i) = g(j)$ with probability $1/2$ we have that $\mathbb{E}[y_{ij}] = 0$.

We have shown that the estimate is unbiased, it remains to show that it produces a good estimate with high probability. We turn to another way to bound the deviation of a random variable from its mean.

Theorem 2 *(Chebyshev's Inequality) Assume that Z is a random variable with variance σ^2, then* $\Pr[|Z - E[Z]| \geq k\sigma] \leq \frac{1}{k^2}$

In order to use the inequality we need to find the variance of the over-estimate. Recall that the overestimate for the count of i was $\sum_{j \neq i} y_{ij}$. Since these are independent,

$$\mathbf{var}\left[\sum_{j \neq i} y_{ij}\right] = \sum_{j \neq i} \mathbf{var}[y_{ij}].$$

It is easy to see that $\mathbf{var}[y_{ij}] = \mathbb{E}[y_{ij}^2] - \mathbb{E}^2[y_{ij}] = \frac{f_j^2}{k}$. Therefore,

$$\mathbf{var}\left[\sum_{j \neq i} y_{ij}\right] = \frac{1}{k} \sum_{j \neq i} f_j^2 \leq F_2/k.$$

We can now apply Chebyshev's Inequality with $\sigma = \sqrt{\frac{F_2}{k}}$. Then:

$$\Pr[\sum y_{ij} \geq \epsilon\sqrt{F_2}] = \Pr[\sum y_{ij} \geq \epsilon\sqrt{k}\sigma] \leq \frac{1}{k\epsilon^2}$$

if fix $k = \frac{3}{\epsilon^2}$, then

$$\Pr[\sum y_{ij} \geq \epsilon\sqrt{F_2}] \leq \frac{1}{3}.$$

At this point we have shown that the estimate is wrong by an additive factor of $\epsilon\sqrt{F_2}$ with probability at most $1/3$. In order to further reduce the failure probability, we can run t such estimate. Setting $t = O((\log 1/\delta)$, we can use Chernoff bounds to show that the total probability of error drops to δ.

Thus we have given an algorithm that returns an estimate $\hat{f}_j$ such that:

$$\Pr[|\hat{f}_j - f_j| \geq \epsilon\sqrt{F_2}] \leq \delta.$$

The total space used by the algorithm is O$(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ counters.

As a point of contrast, the previous method has a total space complexity: O$(\frac{1}{\epsilon} \log \frac{1}{\delta})$, and achieves an additive error of at most ϵF_1

Comparing the methods If the distribution of elements in the stream is approximately uniform, then the two methods give similar approximation guarantees. For example, if there are n elements, and each element appears once, then: $F_1 = n, F_2 = \Sigma 1^2$ =n, $\sqrt{F_2} = \sqrt{n}$

On the other hand, if the distribution is skewed, then $\sqrt{F_2} < F_1$. For example, suppose element x_1 appears n times, $x_2, \ldots, x_n$ appears once, $F_1 = n + n - 1 = 2n - 1$,$F_2 = n^2 + (n-1) \rightarrow \sqrt{F_2} = O(\sqrt{n})$