Lecture 15

1 Streaming Algorithms: Frequent Items

Recall the streaming setting where we have a data stream $x_1, x_2, \ldots, x_n$ with $x_i \in [m]$, the available memory is $O(\log^2 n)$. Today we will see algorithms for finding frequent items in a stream. We first present a deterministic algorithm that approximates frequencies for the top $k$ items. We then introduce more efficient randomized algorithms that can handle insertions as well as deletions.

1.1 Deterministic algorithm

The following algorithm estimates item frequencies $f_j$ within an additive error of $n/k$ using $O(k \log n)$ memory.

1. Maintain set $S$ of $k$ counters, initialize to 0. For each element $x_i$ in stream:

   2. If $x_i \in S$ increment the counter for $x_i$.

   3. If $x_i \not\in S$ add $x_i$ to $S$ if space is available, else decrement all counters in $S$.

   An item in $S$ whose count falls to 0 can be removed, the space requirement for storing $k$ counters is $k \log n$ and the update time per item is $O(k)$.

   The algorithm estimates the count of an item as the value of its counter or zero if it has no counter.

   **Claim 1**
   
   The frequency estimate $n_j$ produced by the algorithm satisfies $f_j - n/k \leq n_j \leq f_j$.

   **Proof:** Clearly, $n_j$ is less than the true frequency $f_j$. Differences between $f_j$ and the value of the estimate are caused by one of the two scenarios: (i) The item $j \not\in S$, each counter in $S$ gets decremented, this is the case when $x_j$ occurs in the stream but the counter for $j$ is not incremented. (ii) The counter for $j$ gets decremented due to an element $j'$ that is not contained in $S$.

   Both scenarios result in $k$ counters getting decremented hence they can occur at most $n/k$ times, showing that $n_j \geq f_j - n/k$. □

1.2 Count min sketch

The turnstile model allows both additions and deletions of items in the stream. The stream consists of pairs $(i, c_i)$, where the $i \in [m]$ is an item and $c_i$ is the number of items to be added or deleted. The count of an item can not be negative at any stage, the frequency $f_j$ of item $j$ is $f_j = \sum c_j$.

The following algorithm estimates frequencies of all items up to an additive error of $\epsilon |f|_1$ with probability $1 - \delta$, the $\ell_1$ norm $|f|_1$ is the number of items present in the data set. The two parameters $k$ and $t$ in the algorithm are chosen to be $(\frac{2}{\epsilon}, \log(1/\delta))$. 
1. Maintain $t$ arrays $A[i]$ each having $k$ counters, hash function $h_i : U \rightarrow [k]$ drawn from a 2-wise independent family $\mathcal{H}$ is associated to array $A[i]$.

2. For element $(j, c_j)$ in the stream, update counters as follows:

$$A[i, h_i(j)] \leftarrow A[i, h_i(j)] + c_j \quad \forall i \in [t]$$

3. The frequency estimate for item $j$ is $\min_{i \in [t]} A[i, h(j)]$.

The output estimate is always more than the true value of $f_j$ as the count of all the items in the stream is non negative.

### 1.2.1 Analysis

To bound the error in the estimate for $f_j$ we need to analyze the excess $X$ where $A[i, h_1(j)] = f_j + X$. The excess $X$ can be expressed as a sum of random variables $X = \sum_i Y_i$ where the indicator random variable $Y_i = f_i$ if $h_1(j) = h_1(i)$ and 0 otherwise. As $h_1 \in \mathcal{H}$ is chosen uniformly at random from a 2-wise independent hash function family, $E[Y_i] = f_i/k$.

$$E[X] = \frac{|f|_1}{k} = \frac{\epsilon |f|_1}{2}$$

Applying Markov’s inequality, we have

$$Pr[X > \epsilon |f|_1] \leq \frac{1}{2}$$

The probability that all the excesses at $A[i, h_i(x_j)]$ are greater than $\epsilon |f|_1$ is at most $1/2^t \leq \delta$ as $t$ was chosen to be $\log(1/\delta)$. The algorithm estimates the frequency of item $x_j$ up to an additive error $\epsilon |f|_1$ with probability $1 - \delta$.

The memory required for the algorithm is the sum of the space for the array and the hash functions, $O(kt \log n + t \log m) = O(\frac{1}{\epsilon} \log(1/\delta) \log n)$. The update time per item in the stream is $O(\log \frac{1}{\delta})$.

### 1.3 Count Sketch

We present another sketch algorithm with error in terms of the $\ell_2$ norm $|f|_2 = \sqrt{\sum_j f_j^2}$. The relation between the $\ell_1$ and $\ell_2$ norms is $\frac{1}{\sqrt{n}} |f|_1 \leq |f|_2 \leq |f|_1$, the $\ell_2$ norm is less than the $\ell_1$ norm so the guarantee for this algorithm is better than that for the previous one.

1. Maintain $t$ arrays $A[i]$ each having $k$ counters, hash functions $g_i : U \rightarrow \{-1, 1\}$ and $h_i : U \rightarrow [k]$ drawn uniformly at random from a 2-wise independent families are associated to array $A[i]$.

2. For element $(j, c_j)$ in the stream, update counters as follows:

$$A[i, h_i(j)] \leftarrow A[i, h_i(j)] + g_i(j)c_j \quad \forall i \in [t]$$

3. The frequency estimate for item $j$ is the median over the $t$ arrays of $g_i(x_j)A[i, h(j)]$. 
1.3.1 Analysis

Again, the entry \( A[1, h_1(j)] = g_1(j) f_j + X \), we examine the contribution \( X \) from the other items by writing \( X = \sum_i Y_i \) where the indicator variable \( Y_i \) is \( \pm f_i \) if \( h_1(i) = h_1(j) \) and 0 otherwise. Note that \( E[Y_j] = 0 \), so the expected value of \( g_1(j) A[1, h(j)] \) is \( f_j \).

The random variables \( Y_i \) are pairwise independent as \( h_1 \) is a 2-wise independent hash function, so the variance of \( X \) can be expressed as,

\[
\text{Var}(X) = \sum_{i \in [m]} \text{Var}(Y_i) = \sum_{i \in [m]} \frac{f_i^2}{k} = \frac{|f_j|^2}{k}
\]

We will use Chebyshev’s inequality to bound the deviation of \( X \) from its expected value,

\[
Pr[|X - \mu| > \Delta] \leq \frac{\text{Var}(X)}{\Delta^2}
\]

The mean \( \mu = f_j \) and the variance is \( \frac{|f_j|^2}{k} \), choosing \( \delta = \epsilon |f_j|_2 \) and \( k = 4/\epsilon^2 \) we have,

\[
Pr[|X - \mu| > \epsilon |f_j|_2] \leq \frac{1}{\epsilon^2 k} \leq \frac{1}{4}
\]

For \( t = \theta(\log(1/\delta)) \), the probability that the median value deviates from \( \mu \) by more than \( \epsilon |f_j|_2 \) is less than \( \delta \) by a Chernoff bound. That is, the probability that there are fewer than \( t/2 \) success in a series of \( t \) tosses of a coin with success probability \( 3/4 \) is smaller than \( \delta \) for \( t = O(\log(1/\delta)) \).

Arguing as in the count min sketch the space required is \( O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n) \) and the update time per item is \( O(\log \frac{1}{\delta}) \).

1.4 Remarks

The count sketch approximates \( f_j \) within \( \epsilon |f_j|_2 \) but requires \( \tilde{O}(\frac{1}{\epsilon^2}) \) space, while the count min sketch approximates \( f_j \) within \( \epsilon |f_j|_1 \) and requires \( O(\frac{1}{\epsilon^2}) \) space. The approximation provided by the sketch algorithms is meaningful only for items that occur with high frequency.