In the previous lecture, we discussed various approaches to sampling, namely reservoir sampling and priority sampling. The lecture ended with the introduction to $l_0$-sampling and an algorithm for $l_0$-sampling. In this lecture, we will begin by analyzing the $l_0$-sampling algorithm from last lecture.

Let us recall, in the general $l_p$-sampling setting, we are given a non-zero vector $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ and we want a random element $r \in [n]$ such that
\[ P[r = i] = \frac{|a_i|^p}{\sum_i |a_i|^p} \]

### 11.1 Analysis of $l_0$-sampling

In the algorithm, we used a hash function $h : [n] \rightarrow [n^3]$ which is $k$-wise independent and chosen uniformly at random from a $O(s)$-universal hash family where $s = O\left(\max\{\log \frac{1}{\epsilon}, \log \frac{1}{\delta}\}\right)$. We also defined $a[j]$ by the following,
\[ a[j]_i = \begin{cases} a_i, & \text{if } h(i) \leq \frac{n^3}{2} \\ 0, & \text{otherwise} \end{cases} \]

We must note the following,
- $a[0]$ is the same as $a$.
- $a[1]$ is a vector which has approximately half the coordinates of $a$ while the rest are zero.
- $a[2]$ is a vector which has approximately quarter the coordinates of $a$ while the rest are zero.

The algorithm gives the vector $a[j]$ as input to an $s$-sparse detection and recovery algorithm for $s = \log \left(\frac{1}{\delta}\right)$. We then choose the first vector for which the recovery succeeds and returns a random coordinate from the smallest $j$ such that $a[j]$ is $s$-sparse. We must also note that
\[ N = \|a\|_0 \\
N_j = \|a[j]\|_0 \]

We would like to have an estimate of the expected value in order to bound the probability using concentration inequalities later. Let $j$ be such that
\[ \frac{s}{4} \leq \mathbb{E}[N_j] \leq \frac{s}{2} \]

We wish to compute the probability of $1 \leq \|a_j\|_0 \leq s$. Since we have an estimate of the expected value, we can use the Chernoff bound,
\[ \mathbb{P}[|N_j - \mathbb{E}[N_j]| \geq r\mathbb{E}[N_j]] \leq e^{-\frac{2s^2|N_j|}{3r^2}} \leq 2^{-\Omega(s)} \leq \delta \text{ since } s = O\left(\log \frac{1}{\delta}\right) \]
Hence, with high probability, \( a[j] \) will be \( s \)-sparse. Suppose \( k = O \left( \frac{1}{\epsilon} \right) \) and \( h \) is chosen uniformly at random from a \( k \)-universal hash family, then for any non-zero coordinate of \( j \) of vector \( a \), we have

\[
P\left[ \arg \max_{i, a_i \neq 0} h(i) = j \right] = \frac{1 \pm \epsilon}{N}.
\]

### 11.2 \( l_2 \)-sampling

We will now study the algorithm for \( l_2 \)-sampling by A. Andoni, R. Krauthgamer and K. Onak in [AKO18]. Suppose we have a frequency vector \( f = (f_1, \ldots, f_n) \) where \( F_2 = \sum_i f_i^2 \) is the second moment. In the \( l_2 \) sampling problem, we wish to sample \( I \in [n] \) based on the following,

\[
P[I = i] = (1 \pm \epsilon) \frac{f_i^2}{F_2}
\]

For simplicity, we will assume \( F_2 = 1 \). The algorithm is as follows,

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textbf{\( l_2 \)-sampling} \tabularnewline
\hline
1. For each \( i \), let \( u_i \in (0, 1] \). \\
2. Let \( w_i = \frac{1}{\sqrt{u_i}} \). \\
3. Let \( g_i = w_i f_i = \frac{f_i}{\sqrt{u_i}} \). \\
4. Let \( g = (g_1, \ldots, g_n) \). \\
5. Suppose there is some large threshold value \( t \): \\
6. If there is a unique \( i \) such that \( g_i^2 \geq t \) (i.e., for all \( j \neq i \), \( g_j^2 < t \)), \\
7. Then return \((i, f_i)\). \\
8. Fail otherwise. \\
\hline
\end{tabular}
\end{table}

We will now calculate the probability of some \( i \in [n] \) being returned. This would happen when

\[
P \left[ g_i^2 \geq t \land \bigwedge_{j \neq i} g_j^2 < t \right] = P[g_i^2 \geq t] \prod_{j \neq i} P[g_j^2 < t]
\]

\[
= P \left[ u_i \leq \frac{f_i^2}{t} \right] \prod_{j \neq i} P \left[ u_j > \frac{f_j^2}{t} \right]
= \frac{f_i^2}{t}
\]

Hence, the probability that some \( i \) is returned would be approximately be the sum of probability of each \( i \)

\[
\sum_i \frac{f_i^2}{t} = \frac{\sum_i f_i^2}{t} = \frac{F_2}{t} = \frac{1}{t}
\]

We used the fact that \( F_2 = 1 \) for the above inequality. In order to boost the probability, we use the standard trick of boosting the probability by running it \( O \left( t \log \frac{1}{\delta} \right) \) to have a probability of \( \delta \) for some element to be
Next, we will look at another $l_2$-sampling algorithm, called $l_2$-precision sampling.

### 11.3 $l_2$-precision sampling

We can describe the algorithm as follows,

**$l_2$-precision sampling**

1. Let $u_1, \ldots, u_n \in [\frac{1}{n^2}, 1]$
2. Let $D$ be a CountSketch for $g = (g_1, \ldots, g_n)$.
3. Given $(j, c)$, we feed $(j, \sqrt{u_j})$ to our CountSketch $D$.
4. Let $\hat{g}_j$ be an estimate for $g_j$ from $D$.
5. Let $\hat{f}_j = \hat{g}_j \sqrt{u_j}$.
6. 
   $$X_j = \begin{cases} 
   1, & \text{if } \hat{g}_j^2 \geq \frac{1}{2} \\
   0, & \text{otherwise}
   \end{cases}$$
7. If there exists a unique $j$ with $X_j = 1$, then return $(j, f_j^2)$.

**Lemma 1** Let $F_2 = \sum_j f_j^2$ and $F_2(f) = \sum_i g_i^2$. Suppose $F_2 = 1$, then $F_2(g) \leq O(\log n)$.

**Proof.**

$$F_2(g) = \sum_i g_i^2$$

$$= \sum_i \frac{f_i^2}{u_i}$$

$$\leq F_2 \int_{\frac{1}{n^2}}^{1} \frac{1}{u} \cdot du$$

$$= F_2 \frac{\ln n^2}{1 - \frac{1}{n^2}}$$

$$\leq 5 \log n = O(\log n)$$

Suppose we use CountSketch with parameters $(k, d)$ for $g_i$ where $k = O\left(\frac{1}{\epsilon}\right)$ and $d = O(\log n)$. We can write $\hat{g}_j^2$ as the following,

$$\hat{g}_j^2 = g_j^2 + Z_j^2$$
where $Z_j$ was the sum of contribution of $i \neq j$ where their hash functions collide. We showed $\mathbb{E}[Z_j] \leq \frac{F_2(g)}{k}$ and using Markov’s inequality, we get
\[
P\left[Z_j^2 > \frac{3F_2(g)}{k}\right] < \frac{1}{3}
\]
In this case, with probability greater than $\frac{2}{3}$, we have $Z_j^2 < 3\epsilon F_2(g)$. We will use the fact $1 \pm \epsilon \approx e^{\pm \epsilon}$ many times in our analysis.

- Case 1: When $|g_j| \geq \frac{2}{\epsilon}$

  This would imply
  \[
  \hat{g}_j^2 = (g_j + Z_j)^2 = g_j^2 + 2Z_jg_j + Z_j^2
  \]

  The latter term is small because $\mathbb{E}[Z_j] \leq \frac{F_2(g)}{k}$ and $k = O\left(\frac{\log n}{\epsilon}\right)$.

- Case 2: When $|g_j| > \frac{2}{3}$

  This would imply
  \[
  |\hat{g}_j^2 - g_j^2| = (g_j + Z_j)^2 - g_j^2
  = Z_j^2 + 2g_jZ_j
  \leq Z_j^2 \left(1 + \frac{4}{\epsilon}\right)
  = 6\epsilon Z_j^2
  \]

  Hence, with probability $\frac{2}{3}$, we have that $|\hat{g}_j^2 - g_j^2| < \frac{18F_2(g)}{\epsilon k}$. Suppose we choose $k = O\left(\frac{\log n}{\epsilon}\right)$, then with probability greater than $1 - \left(\frac{1}{10} + \frac{1}{3}\right)$, we have $F_2(g) \leq 50\log n$ and $Z_j \leq \frac{3F_2(g)}{k}$.

  Suppose we have
  \[
  \hat{g}_j^2 = (1 \pm \epsilon)g_j^2 \pm 1 \implies f_j^2 = (1 \pm \epsilon)f_j^2 \pm u_j
  \]

  Suppose $X_j = 1$, that means $u_j \ll \frac{\epsilon f_j^2}{\epsilon}$ which implies
  \[
  \hat{f}_j^2 = (1 \pm \epsilon)f_j^2 \pm \frac{\epsilon f_j^2}{4}
  \implies \hat{f}_j^2 = (1 \pm O(\epsilon))f_j^2
  \]

References