1 Overview

This lecture introduces Priority and $l_0$-Sampling: Priority sampling is the problem that given a data stream of items with weights $w_1, \ldots, w_n$, we want to store a representative sample of the items so that we can answer subset sum queries. That is, given a set $I \subset [n]$, we would like to answer queries about the value $\sum_{i \in I} w_i$. Then we introduce $l_0$-Sampling using ideas from linear sketching and sparse recovery.

2 Priority Sampling

**Problem:** given a data stream of items with weights $w_1, \ldots, w_n$, we want to store a representative sample $S$ of the items so that we can answer subset sum queries. That is, given a query $I \subset [n]$, we would like to approximate $W_I = \sum_{i \in I} w_i$.

One scheme is the following:

1. For data $w_1, \ldots, w_n$, we sample $u_i \in [0, 1]$ uniformly at random and independently.
2. We compute the priorities $q_i = w_i/u_i$ for each $i \in [n]$.
3. We always keep the set of items $S_k$ containing the $k$-largest priorities seen so far, as well as $\tau$ the value of the $(k+1)$-largest priority.
4. Given a query $I \subset [n]$, we output $\hat{W}_I = \sum_{j \in I \cap S_k} \max\{\tau, w_j\}$.

This scheme has strong optimality guarantees.

2.1 Analysis

For convenience in the analysis we define a different set of weights through $\hat{w}_i = \begin{cases} \max\{\tau, w_i\} & \text{if } i \in S_k \\ 0 & \text{otherwise} \end{cases}$

Then, the output of the algorithm is equivalent to $\hat{W}_I = \sum_{j \in I} \hat{w}_j$.

We want to prove the following two results.

**Lemma 1.** $E[\hat{w}_i] = w_i$. 
Proof. Let $A(\tau')$ be the event that the $(k+1)$-th highest priority is $\tau'$, then for all $i \in S$, we must have that $q_i = w_i/u_i > \tau'$, and the corresponding weight is $\tilde{w}_i = \max(\tau', w_i)$, otherwise for $i \notin S$ $q_i \leq \tau'$ and $\tilde{w}_i = 0$. Let’s look at $\mathbb{P}(i \in S | A(\tau'))$. We distinguish two cases:

1. $w_i > \tau'$: then $\mathbb{P}(i \in S_k | A(\tau')) = 1$ and $\tilde{w}_i = w_i$.

2. $w_i \leq \tau'$: then $\mathbb{P}(i \in S_k | A(\tau')) = \mathbb{P}(u_i \leq \frac{w_i}{\tau'}) = \frac{w_i}{\tau'}$ and $\tilde{w}_i = \tau'$.

In both cases $\mathbb{E}[\tilde{w}_i] = w_i$.

Lemma 2. $\mathbb{E}\left[\prod_S \tilde{w}_i\right] = \prod_S w_i$, $|S| \leq k$, $\text{Var}(\sum_I \tilde{w}_i) = \sum_I \text{Var}(\tilde{w}_i)$.

Proof. Proof left to the readers.

3 $\ell_0$-Sampling

Problem: given a vector $(a_1, \ldots, a_n)$, we want to sample a random element $I \in [n]$ of the vector such that $\mathbb{P}(I = i) = \frac{|a_i|^p}{\sum |a_j|^p}$. This is called $\ell_p$-sampling.

We have seen examples of $\ell_p$-sampling in the streaming setting. For example, reservoir sampling is $\ell_1$-sampling with only positive update. In general though, we relax our requirements and we are content if we can sample $i$ with probability $(1 + \epsilon)\frac{|a_i|^p}{\sum |a_j|^p} \pm \delta$ for some $\epsilon, \delta > 0$.

$\ell_0$-sampling means that we are sampling near-uniformly from the distinct elements in the stream.

3.1 Algorithm

We use ideas from linear sketch and sparse recovery. We assume that given an $x \in \mathbb{R}^n$, we can use a linear sketch $y = Ax$ to compute $z$ such that $||x - z||_p \leq C||x - x^*||_p$. Observe, if $x$ had few non-zero coordinates, $z$ recovers $x$ exactly. We are going to exploit this fact to perform $\ell_0$-sampling.

We use the following scheme: pick a nest of random subsets $I_h$ of the index of $x$ with size $n, n/2, \ldots, n/2^r$, for some $r \leq \log(n)$. To perform the sampling efficiently, we use a $k$-wise independent hash function: $h: [n] \mapsto [n^3]$.

The procure is the following:

1. Sampling:
   if $h(i) \leq n^3/2^j$, then $a(j)_i = a_i$

2. Recovery:
   run sparse recovery algorithm on $x$ restricted to subset $I_h$. If any of the sparse-recoveries succeeds then output a $s$–sparse vector for $s = O(\log(1/\delta))$. Algorithm fails if none of the sparse-recoveries output a valid vector
3. Selection:

pick a level $j$ that has successful recovery, and output the index $i$ of the smallest hash value $h(i)$. We can understand this as output a random coordinate from the first sparse recovery that succeeds.

3.2 Analysis of the scheme

We define $N_j = |a(j)|_0$ as the number of non-zero coordinates. Then $s/4 \leq E[N_j] \leq s/2$. We have the inequality

$$P(|N_j - E[N_j]| \leq E[N_j]) \leq P(1 \leq N_j \leq 2E[N_j]) \leq P(1 \leq N_j \leq s),$$

and since $k \geq rE[N_j]$, we have a Chernoff bound-like result for sum of $k$-wise independent $[0,1]$ variables,

$$P(|N_j - E[N_j]| \leq rE[N_j]) \leq e^{-rE[N_j]/3}$$

Since $E[N_j] \geq s/4$, if we set $s = 12 \log(1/\delta)$, the failure probability $P[N_j > s]$ is less than $\delta = e^{-s/12}$.

References