CS 598CSC: Algorithms for Big Data	Lecture date: Sept 23, 2014
Instructor: Chandra Chekuri	Scribe: Chandra Chekuri

1 Priority Sampling and Sum Queries

Suppose we have a stream $a_1, a_2, \ldots, a_n$ (yes, we are changing notation here from m to n for the length of the stream) of objects and each a_i has a *non-negative* weight w_i. We want to store a representative sample $S \subset [n]$ of the items so that we can answer subset sum queries. That is, given a query $I \subseteq [n]$ we would like to answer $\sum_{i \in I} w_i$. One way to do this as follows. Suppose we pick S as follows: sample each $i \in [n]$ independently with probability p_i and if i is chosen we set a scaled weight $\hat{w}_i = w_i/p_i$. Now, given a query I we output the estimate for its weight as $\sum_{i \in I \cap S} \hat{w}_i$. Note that expectation of the estimate is equal to $w(I)$. The disadvantage of this scheme is mainly related to the fact that we cannot control the size of sample. This means that we cannot fully utilize the memory available. Related to the first, is that if we do not know the length of stream apriori, we cannot figure out the sampling rate even if we are willing to be flexible in the size of the memory. An elegant scheme of Duffield, Lund and Thorup, called priority sampling overcomes these limitations. They considered the setting where we are given a parameter k for the size of the sample S and the goal is maintain a k-sample S along with some weights $\hat{w}_i$ for $i \in S$ such that we can answer subset sum queries.

Their scheme is the following, described as if $a_1, a_2, \ldots, a_n$ are available offline.

1. For each $i \in [n]$ set priority $q_i = w_i/u_i$ where u_i is chosen uniformly (and independently from other items) at random from $[0, 1]$.
2. S is the set of items with the k highest priorities.
3. τ is the $(k+1)$'st highest priority. If $k \geq n$ we set $\tau = 0$.
4. If $i \in S$, set $\hat{w}_i = \max\{w_i, \tau\}$, else set $\hat{w}_i = 0$.

We observe that the above sampling can be implemented in the streaming setting by simply keeping the current sample S and current threshold τ. We leave it as an exercise to show that this informatino can be updated when a new item arrives.

We show some nice and non-obvious properties of priority sampling. We will assume for simplicity that $1 < k < n$. The first one is the basic one that we would want.

Lemma 1 $\mathbf{E}[\hat{w}_i] = w_i$.

Proof: Fix i. Let $A(\tau')$ be the event that the k'th highest priority among items $j \neq i$ is τ'. Note that $i \in S$ if $q_i = w_i/u_i \geq \tau'$ and if $i \in S$ then $\hat{w}_i = \max\{w_i, \tau'\}$, otherwise $\hat{w}_i = 0$. To evaluate $\Pr[i \in S \mid A(\tau')]$ we consider two cases.

Case 1: $w_i \geq \tau'$. Here we have $\Pr[i \in S \mid A(\tau')] = 1$ and $\hat{w}_i = w_i$.

Case 2: $w_i < \tau'$. Then $\Pr[i \in S \mid A(\tau')] = \frac{w_i}{\tau'}$ and $\hat{w}_i = \tau'$.

In both cases we see that $\mathbf{E}[\hat{w}_i] = w_i$. □

The previous claim shows that the estimator $\sum_{I \cap S} \hat{w}_i$ has expectation equal to $w(I)$. We can also estimate the variance of $\hat{w}_i$ via the threshold τ.

Lemma 2 $\mathbf{Var}[\hat{w}_i] = \mathbf{E}[\hat{v}_i]$ *where* $\hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$

Proof: Fix i. We define $A(\tau')$ to be the event that τ' is the k'th highest priority among elements $j \neq i$. The proof is based on showing that

$$\mathbf{E}[\hat{v}_i \mid A(\tau')] = \mathbf{E}[\hat{w}_i^2 \mid A(\tau')] - w_i^2.$$

From the proof outline of the preceding lemma, we estimate the lhs of as

$$\begin{aligned} \mathbf{E}[\hat{v}_i \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{v}_i \mid i \in S \wedge A(\tau')] \\ &= \min\{1, w_i/\tau'\} \times \tau' \max\{0, \tau' - w_i\} \\ &= \max\{0, w_i\tau' - w_i^2\}. \end{aligned}$$

Now we analyze the rhs,

$$\begin{aligned} \mathbf{E}[\hat{w}_i^2 \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{w}_i^2 \mid i \in S \wedge A(\tau')] \\ &= \min\{1, w_i/\tau'\} \times (\max\{w_i, \tau'\})^2 \\ &= \max\{w_i^2, w_i\tau'\}. \end{aligned}$$

□

Surprisingly, if $k \geq 2$ then the covariance between $\hat{w}_i$ and $\hat{w}_j$ for any $i \neq j$ is equal to 0.

Lemma 3 $\mathbf{E}[\hat{w}_i \hat{w}_j] = 0$.

In fact the previous lemma is a special case of a more general lemma below.

Lemma 4 $\mathbf{E}[\prod_{i \in I} \hat{w}_i] = \prod_{i=1}^{k} w_i$ *if* $|I| \leq k$ *and is* 0 *if* $|I| > k$.

Proof: It is easy to see that if $|I| > k$ the product is 0 since at least one of them is not in the sample. We now assume $|I| \leq k$ and prove the desired claim by inducion on $|I|$. In fact we need a stronger hypothesis. Let τ'' be the $(k - |I| + 1)$'th highest priority among the items $j \neq i$. We will condition on τ'' and prove that $\mathbf{E}[\prod_{i \in I} \hat{w}_i \mid A(\tau'')] = \prod_{i \in I} w_i$. For the base case we have seen the proof for $|I| = 1$.

Case 1: There is $h \in I$ such that $w_h > \tau''$. Then clearly $h \in S$ and $\hat{w}_h = w_h$. In this case

$$\mathbf{E}[\prod_{i \in I} \hat{w}_i \mid A(\tau'')] = w_h \cdot \mathbf{E}[\prod_{i \in I \setminus \{h\}} \hat{w}_i \mid A(\tau'')],$$

and we apply induction to $I \setminus \{h\}$. Technically the term $\mathbf{E}[\prod_{i \in I \setminus \{h\}} \hat{w}_i \mid A(\tau'')]$ is referring to the fact that τ'' is the $k - |I'| + 1$'st highest priority where $I' = I \setminus \{h\}$.

Case 2: For all $h \in I$, $w_h < \tau''$. Let q be the minimum priority among items in I. If $q < \tau''$ then $\hat{w}_j = 0$ for some $j \in I$ and the entire product is 0. Thus, in this case, there is no contribution to the expectation. Thus we will consider the case when $q \geq \tau''$. The probability for this event is $\prod_{i \in I} \frac{w_i}{\tau''}$. But in this case all $i \in I$ will be in S and moreover $\hat{w}_i = \tau''$ for each i. Thus the expected value of $\prod_{i \in I} \hat{w}_i = \prod_{i \in I} w_i$ as desired. □

Combining Lemma 2 and 3 the variance of the estimator $\sum_{i \in I \cap S} \hat{w}_i$ is

$$\mathbf{Var}[\sum_{i \in I \cap S} \hat{w}_i = \sum_{i \in I \cap S} \mathbf{Var}[\hat{w}_i] = \sum_{i \in I \cap S} \mathbf{E}[\hat{v}_i].$$

The advantage of this is that the variance of the estimator can be computed by examining τ and the weights of the elements in the $S \cap I$.

2 ℓ_0 Sampling

We have seen ℓ_2 sampling in the streaming setting. The ideas generalize to ℓ_p sampling to ℓ_p sampling for $p \in (0,2)$ — see [] for instance. However, ℓ_0 sampling requires slightly different ideas. ℓ_0 sampling means that we are sampling near-uniformly from the distinct elements in the stream. Surprisingly we can do this even in the turnstile setting.

Recall that one of the applications we saw for the Count-Sketch is ℓ_2-sparse recovery. In particular we can obtain a $(1+\epsilon)$-approximation for $\mathrm{err}_2^k(\mathbf{x})$ with high-probability using $O(k \log n/\epsilon)$ words. Suppose $\mathbf{x}$ is k-sparse then $\mathrm{err}_2^k(\mathbf{x}) = 0$! It means that we can detect if $\mathbf{x}$ is k-sparse, and in fact identify the non-zero coordinates of $\mathbf{x}$, with high-probability. In fact one can prove the following stronger version.

Lemma 5 *For $1 \leq k \leq n$ there and $k' = O(k)$ there is a sketch $L : \mathbb{R}^n \to \mathbb{R}^{k'}$ (generated from $O(k \log n)$ random bits) and a recovery procedure that on input $L(\mathbf{x})$ has the following feature: (i) if $\mathbf{x}$ is k-sparse then it outputs $\mathbf{x}' = \mathbf{x}$ with probability 1 and (ii) if $\mathbf{x}$ is not k-sparse the algorithm detects this with high-probability.*

We will use the above for ℓ_0 sampling as follows. We will first describe a high-level algorithm that is not streaming friendly and will indicate later how it can be made stream implementable.

1. For $h = 1, \ldots, \lfloor \log n \rfloor$ let I_h be a random subsets of $[n]$ where I_j has cardinality 2^j. Let $I_0 = [n]$.
2. Let $k = \lceil 4 \log(1/\delta) \rceil$. For $h = 0, \ldots, \lfloor \log n \rfloor$, run k-sparse-recovery on $\mathbf{x}$ restricted to coordinates of I_h.
3. If any of the sparse-recoveries succeeds then output a random coordinate from the first sparse-recovery that succeeds.
4. Algorithm fails if none of the sparse-recoveries output a valid vector.

Let J be the index set of non-zero coordinates of $\mathbf{x}$. We now show that the algorithm with probability $(1-\delta)$ succeeds in outputting a uniform sample from J. Suppose $|J| \leq k$. Then $\mathbf{x}$ is recovered exactly for $h = 0$ and the algorithm outputs a uniform sample from J. Suppose $|J| > k$. We observe that $\mathbf{E}[|I_h \cap J|] = 2^h |J|/n$ and hence there is a h^* such that $\mathbf{E}[|I_{h^*} \cap J|] = 2^{h^*}|J|/n$ is between $k/3$ and $2k/3$. By Chernoff-bounds one can show that with probability at least $(1-\delta)$, $1 \leq |I_{h^*} \cap J| \leq k$. For this h^* the sparse recovery will succeed and output a random coordinate of J. The formal claims are the following:

- With probability at least $(1-\delta)$ the algorithm outputs a coordinate $i \in [n]$.
- If the algorithm outputs a coordinate i then the probability that it is not a uniform random sample is because the sparse recovery algorithm failed for some h; we can make this probability be less than $1/n^c$ for any desired constant c.

Thus, in fact we get zero-error ℓ_0 sample.

The algorithm, as described, requires one to sample and store I_h for $h = 0, \ldots, \lfloor \log n \rfloor$. In order to avoid this we can use Nisan's pseudo-random generator for small-space computation. We skip the details of this; see [2]. The overall space requirements for the above procedure can be shown

to be $O(\log^2 n \log(1/\delta)$ with an error probability bounded by $\delta + O(1/n^c)$. This is near-optimal for constant δ as shown in [2].

Bibliographic Notes: The material on priority sampling is directly from [1] which describes applications, relationship to prior sampling techniques and also has an experimental evaluation. Priority sampling has shown to be "optimal" in a strong sense; see [3].

The ℓ_0 sampling algorithm we described is from the paper by Jowhari, Saglam and Tardos [2]. See a simpler algorithm in the chapter on signals by McGregor-Muthu draft book.

References

[1] Nick Duffield, Carsten Lund, and Mikkel Thorup. Priority sampling for estimation of arbitrary subset sums. *Journal of the ACM (JACM)*, 54(6):32, 2007.

[2] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. *CoRR*, abs/1012.4889, 2010.

[3] Mario Szegedy. The dlt priority sampling is essentially optimal. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 150–158. ACM, 2006.