Sublinear Time and Space Algorithms 2024A – Lecture 6
Reservoir Sampling and ℓ_0-sampling*

Robert Krauthgamer

1 Reservoir Sampling

Problem definition: Pick a uniformly random item from a stream of length at most m.

Reservoir Sampling [Vitter, 1985]:

1. Init: $s = \text{null}$
2. Update (next item a): increment j, and with probability $1/j$ let $s = a_j$
3. Output: s

Lemma: Assuming stream items come from $[n]$, this algorithm uses storage $O(\log(n+m))$. Its output is a uniform item from the stream, i.e., each position j is picked (and outputted) with the same probability $1/m$.

Note that items appearing many times are output with high probability.

Exer: Prove this lemma.

Exer: Design a streaming algorithm that at every time m (not known in advance) receives a query $S \subset [n]$ and outputs an estimate what fraction of items in the stream belong to S within additive error ϵ. Note that S is given only at query time (not in advance).

Hint: Maintain $O(1/\epsilon^2)$ random samples and use them to estimate the fraction in S.

Exer: Design an algorithm that samples k items *without replacement* from an input stream $\sigma = (\sigma_1, \ldots, \sigma_m)$. The algorithm's memory requirement should be $O(k)$ words. and the parameter k is known in advance. Prove that the algorithm's output has the correct distribution.

Hint: The goal is essentially to sample k distinct indices $(i_1 < \cdots < i_s)$ uniformly at random. In contrast, executing the Reservoir Sampling algorithm k times in parallel gives k samples *with* replacement, i.e., the same $i \in [m]$ could be reported more than once.

We next consider: more interesting distributions, for example taking duplicates into account.

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

2 ℓ_0-sampling

Problem Definition (ℓ_p-sampling): Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream. The goal is to draw a random index from $[n]$ where each i has probability $\frac{|x_i|^p}{\|x\|_p^p}$.

We will see today the case $p = 0$, where the goal is to draw a uniformly random i from the set $\text{supp}(x) = \{i \in [n] : x_i \neq 0\}$, i.e., it samples from the distinct elements in the stream.

Sampling algorithms may have errors either in the probabilities being approximately correct (e.g., $\pm\delta$) and/or that with some probability they return a wrong answer (FAIL or a sample not according to the desired distribution).

Framework for ℓ_0-sampling [following Cormode and Firmani, 2014]:

(A) subsample the coordinates of x with geometrically decreasing rates

(B) detect if the resulting vector y is 1-sparse

(C) if y is 1-sparse, recover its nonzero coordinate.

(A) Subsampling:

The algorithm chooses a random hash function $h : [n] \to [\log n]$, such that for each $i \in [n]$,

$$\Pr[h(i) = l] = 2^{-l}, \qquad \forall l \in [\log n].$$

(The probabilities do not sum up to 1, and in the remaining probability we can set $h(i)$ to nil, i.e., no level.)

For each "level" $l \in [\log n]$, create a virtual stream for the coordinates in $h^{-1}(l)$, formally defined as $y^{(l)} \in \mathbb{R}^n$ obtained from x by zeroing coordinates outside $h^{-1}(l)$.

Observe that y is obtained from x by a linear map.

Lemma: If $x \neq 0$, then there exists $l \in [\log n]$ for which $\Pr[|\text{supp}(y)| = 1] = \Omega(1)$.

Proof: Was seen in class.

Exer: Show that whenever $\text{supp}(y)$ contains only one coordinate, that coordinate is indeed drawn uniformly from $\text{supp}(x)$.

Exer: Show that the lemma holds even if the hash function h is only pairwise independent. (However, now the "surviving" coordinate might be non-uniform.)

The success probability (getting $|\text{supp}(y)| = 1$) can be increased to $1 - \delta$ by $O(\log \frac{1}{\delta})$ repetitions. The overall result is a $O(\log n \log \frac{1}{\delta})$ virtual streams y.

(C) Sparse recovery (of a 1-sparse vector): Suppose $y \in \mathbb{R}^n$ (which is some $y^{(l)}$ from above) is 1-sparse. How can we find which coordinate i is nonzero?

Compute $A = \sum_i y_i$ and $B = \sum_i i \cdot y_i$ and report their ratio B/A.

For 1-sparse vector the output is always correct, as this step is deterministic.

Observe that A, B form a linear sketch whose size (dimension) is 2 words. Thus, they can be easily maintained over the virtual stream y (and also over the original stream x), even in the presence of deletions.

(B) Detection (if a vector is 1-sparse):

Lemma: There is a linear sketch to detect whether $y \in \mathbb{R}^n$ is 1-sparse, that has one-sided error probability $1/n^3$ (i.e., if $|\text{supp}(y)| = 1$ it always accepts, otherwise it accepts with probability at most $1/n^3$) and uses $O(\log n)$ words.

Proof: Was seen in class, using the AMS sketch to test if ℓ_2 norm is zero.

Exer: Show how to improve the storage to $O(1)$ words by a more direct approach.

Hint: Use a linear map (of y) with random coefficients from $[-n^3, n^3]$.

Overall Algorithm:

The algorithm goes over all virtual streams in a fixed order (all $O(\log n)$ levels and all $O(\log \frac{1}{\delta})$ repetitions), and reports the first coordinate that is recovered successfully and passes the detection test. If none of them succeeded, it reports FAIL.

Storage: The total storage is $O(\log^2 n \log \frac{1}{\delta})$ words, not including randomness.

Error: As seen in class, there are two possible bad events, and overall each $i \in \text{supp}(x)$ is reported with probability at least $\frac{1}{|\text{supp}(x)|} - \delta - 1/n^2$.

However, using limited randomness in the subsampling (necessary to reduce randomness) might introduce some bias to the uniform probabilities.

Variations of this approach: Detection and recovery of vectors with sparsity $s = 1/\varepsilon$ instead of $s = 1$, using k-wise independent hashing in the subsampling, or using Nisan's pseudorandom generator to reduce storage.

Theorem [Jowhari, Saglam and Tardos, 2011]: There is a streaming algorithm with storage $O(\log^2 n \log \frac{1}{\delta})$ bits, that with probability at most δ reports FAIL, with probability at most $1/n^2$ reports an arbitrary answer, and with the remaining probability produces a uniform sample from $\text{supp}(x)$.