Sublinear Time and Space Algorithms 2020B – Lecture 8
ℓ_0-sampling and streaming of graphs*

Robert Krauthgamer

1 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 $\text{supp}(x) = \{i \in [n] : x_i \neq 0\}$.

Algorithms may have some errors either in the probabilities being approximately correct (e.g., $\pm\delta$) and/or that with some probability the algorithm gives a wrong answer (returns 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 add 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 $y^{(l)} \in \mathbb{R}^n$ which is obtained from x by zeroing out 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 $\Pr[|\text{supp}(y)| = 1] = \Omega(1)$.

*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.

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 how to achieve a similar guarantee using a hash function h that is only pairwise independent. (However, now the "surviving" coordinate might be non-uniform.)

The success probability 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 $y^{(l)}$ 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.

Notice that A, B form a linear sketch whose size (dimension) is 2 words. Moreover, they can be maintained over the original stream x (no need to maintain the virtual stream y explicitly).

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

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

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 the virtual streams (levels and their repetitions) in a fixed order, and reports the first coordinate that is recovered successfully and passes the detection test (otherwise FAIL).

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

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, 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 $\text{supp}(x)$.

2 Streaming of Graphs

Basic model: Consider an input stream that represents a graph $G = (V, E)$ as a sequence of edges on the vertex set $V = [n]$. Denote $m = |E|$.

It can be viewed as a sequence of edge insertions to a graph.

Remark: We will consider later a more general model that allows edges deletions (called dynamic graphs).

Semi-streaming: The usual aim is space requirement $\tilde{O}(n)$, which can generally be much smaller than the trivial bound $O(m)$ of storing the current graph explicitly (but without extra workspace an algorithm may need).

For many problems, $\Omega(n)$ storage is required (even to get approximate answers).

Connectivity: Determine whether the graph G is connected (or even which pairs $u, v \in V$ are connected).

Can be solved (in the insertions-only model) with storage requirement $O(n)$ words, by maintaining a spanning forest...

Distances: Maintain all the distances in the graph, i.e., given a query $u, v \in V$ report their distance.

Theorem: Can be solved within approximation $2k - 1$ (for integer $k \geq 1$) in the insertions-only model with storage requirement $O(n^{1+1/k})$ words.

The idea is to use a greedy spanner construction by [Althofer, Das, Dobkin, Joseph and Soares, 1993].

Proof: Create and store a subgraph G' as follows. When an edge (u, v) arrives, check if the distance between its endpoints in G' is $d_{G'}(u, v) \leq 2k - 1$. If it is not, then add the edge to G' (otherwise, do nothing).

It is not difficult to verify that

$$\forall u, v \in V, \qquad d_G(u, v) \leq d_{G'}(u, v) \leq (2k - 1) d_G(u, v).$$

The bound on the number of edges in G' follows by a theorem from extremal graph theory, because its girth (length of shortest cycle) is $g \geq 2k + 1$.

Exer: Show how to 2-approximate maximum matching and vertex-cover using space of $O(n)$ words.