1 Dynamic Sampling (and applications)

Recall that in the previous lecture, we introduced the following subproblem as building block for an approach to solve the connectivity problem on dynamic graph streams:

1.1 Dynamic Sampling

Stream: general updates to a vector $x \in \{-1, 0, +1\}^n$

Goal: output $X \in [n]$ with $\Pr[X = i] = \frac{|x_i|}{\sum_{j=1}^n |x_j|}$.

Before describing and analyzing an algorithm achieving this goal, we describe (some) intuition behind it by considering the following two extreme cases, where we let $D = \{ i \in [n] : x_i \neq 0 \}$:

- If $|D| = 10$, then each $x_i \neq 0$ is a $\frac{1}{10}$-heavy hitter: we can use CountSketch to recover all of them using a total of $O(\log n)$ space.

- If $|D| = 10\sqrt{n}$, we can downsample. That is, we can first obtain a random subset $S \subseteq [n]$ by including independently each coordinate $i \in [n]$ with probability $1/\sqrt{n}$ (so that $|S| \approx \sqrt{n}$), and then focus on the substream involving indices $i \in S$. (Note that this allows to reduce to the previous case, since $\mathbb{E}[|D \cap S|] = 10$, and with high probability we should actually have $|D \cap S| \approx 10$.)

The actual algorithm will in some sense interpolate between these cases, by considering all possible “orders of magnitude” for $|D|$ (and focusing on the one that works.)

1.1.1 Basic Sketch

We start by choosing a hash function $g : [n] \rightarrow \{0, \ldots, n - 1\} \equiv \{0, 1\}^L$ (where $L = \log n$), and let $h : [n] \rightarrow [L]$ be the function defined by

$$h(i) = \max \left\{ k \in \{0, \ldots, L\} : \exists x \in \{0, 1\}^*, g(i) = x0^k \right\}, \quad i \in [n].$$

---

\footnote{This actually leverages some extra guarantees of CountSketch, not explicitly stated in the previous lectures. Namely, defining for a vector $x$ its tail $x^{\text{tail}}$ as $x$ restricted to the set of indices that do not belong to the top-$k$ coordinates (where in the previous lectures we had $k = 1/\phi$), then for all $i \in [n]$ CountSketch an estimate of $x_i$ accurate to within an additive $\pm ||x^{\text{tail}}||_2/k$.}
(That is, \(h(i)\) is the number of tail zeros in the binary expansion of \(g(i)\): if \(g(i) = 01000100\), then \(h(i) = 2\). This implies that, over the randomness of the hash function \(g\), for any \(i \in [n]\)

\[
\Pr[h(i) = j] = \begin{cases} 
\frac{1}{2^{j+1}} & \text{for } 0 \leq j \leq L - 1 \\
\frac{1}{2^L} & \text{for } j = L 
\end{cases}
\]

where the first expression comes from the fact that \(h(i) = j\) iff the last \(j\) bits are 0 and the bit just before \(j\) is 1 (which happens with probability \(2^{-j} \cdot \frac{1}{2}\)). In particular, \(\sum_{j=0}^{L} \Pr[h(i) = j] = 1\), as it should for a probability distribution.

The algorithm then partition the stream into \(L + 1\) substreams \(I_0, \ldots, I_L\), where the substream \(I_j\) only includes the indices \(i \in [n]\) such that \(h(i) = j\). The crucial observation here is that this implies that

\[
\mathbb{E}[|D \cap I_j|] = \frac{|D|}{2^{j+1}}, \quad 0 \leq j \leq L - 1
\]

i.e. “stream \(I_j\) corresponds to downsampling with probability \(1/2^{j+1}\).”

**Sketch.** For each \(0 \leq j \leq L\):

- Store \(CS_j\): **CountSketch** on \(I_j\), with parameter \(\phi \overset{\text{def}}{=} \frac{1}{100}\);

- Store \(DC_j\): **DistinctCountSketch** on \(I_j\), with parameter \(\varepsilon \overset{\text{def}}{=} \frac{1}{10}\) (for a 1.1-approximation);

both with success probability \(1 - \frac{1}{n}\) (for a union bound over all streams and iterations) (which costs an extra \(O(\log n)\) factor in the space complexity).\(^2\) Note that for **DistinctCountSketch** we can use the linear sketch Tug-of-War (which approximates the frequency moment \(F_2\)). Indeed, since each \(f_i \in \{-1, 0, 1\}\), we get \(\sum_i f_i^2 = \sum_i \mathbb{1}_{\{f_i \neq 0\}} = |D|\).

**Estimation.**

1. Find a substream \(I_j\) such that \(DC_j \in [1, 20]\): if none, output fail.

2. Recover all \(i \in I_j\) such that \(x_i \neq 0\) (using \(CS_j\)).

3. Output one of them uniformly at random.

**Analysis.** (We condition on all sketches \(DC_j, CS_j\) computed in the sketching stage to meet their guarantee, which overall by a union bound over all \(O(L)\) sketches happens with probability \(1 - o(1)\).)

- First, if \(0 < |D| < 10\), then there exists some \(j\) for which \(DJ_j \in [1, 11]\); thus, the algorithm does not output fail in the first step and the rest goes through.

- We can therefore assume \(|D| \geq 10\). Let \(k \geq 0\) be such that \(|D| \in [10 \cdot 2^k, 10 \cdot 2^{k+1})\); then,

\[
\mathbb{E}[|D \cap I_k|] = \frac{|D|}{2^{k+1}} \in [5, 10)
\]  \(^{(1)}\)

\(^2\)Note that it will become apparent later in the analysis that \(1 - \frac{1}{100\log n}\) or so would be sufficient, if one wanted to do precise bookkeeping.
and furthermore, setting \( p \overset{\text{def}}{=} \Pr[i \in I_k] = \frac{1}{2^{k+1}} \) for convenience, one can compute the variance as follows:

\[
\text{Var} |D \cap I_k| = \text{Var} \sum_{i \in D} 1_{\{i \in I_k\}} \\
= \sum_{i \in D} \text{Var} 1_{\{i \in I_k\}} \quad \text{((Pairwise) independence)} \\
= \sum_{i \in D} p(1 - p) \quad \text{(Variance of a Bernoulli)} \\
= |D| p(1 - p) \leq \frac{|D|}{2^{k+1}} \\
\leq 10. \quad \text{(by definition of } k) 
\]

Applying Chebyshev’s inequality, we obtain that

\[
\Pr[|D \cap I_k| \not\in [1, 20]] \leq \Pr[||D \cap I_k| - \mathbb{E}|D \cap I_k|| > 4] \leq \frac{\text{Var} |D \cap I_k|}{16} \leq \frac{10}{16} = 0.625. 
\]

Thus, we have \( DC_k \in [1, 20] \) with probability at least 0.375. Conditioning on this event, the algorithm does not output \text{fail} in Step 1: let then \( j \) be any index such that \( DC_j \in [1, 20] \) (there is at least one such index, namely \( k \)). This, along with the setting of the parameter \( \phi \), guarantees that \( CS_j \) will recover a heavy hitter: that is, \( i \in D \cap I_j \).

Finally, by symmetry, we can see that as long as the algorithm reaches Step 3 and outputs \( i \in D \cap I_j \) for some \( j \), then \( i \) is uniformly distributed over \( D \).

**Observation 1.** As the analysis only relied on independence for the computation of the variance (to apply Chebyshev’s inequality), a pairwise independent (family of) hash function(s) is sufficient for \( g \).

**Guarantees.** The algorithm we described and analyzed above, DYNSAMPLEBASIC, only offers the following guarantees:

- it fails with probability at most 0.625 (and we know when it does, as it explicitly outputs \text{fail});
- whenever it does not fail, it outputs \( i \) uniformly distributed in \( D \) (modulo a negligible probability that either one of the \( CS_j \)’s or \( DS_j \)’s does not succeed).

To reduce the failure probability, one can do the usual trick: that is, taking independent copies. Below is the overall algorithm, DYNSAMPLEFULL:

- Run \( \ell = O(\log n) \) independent copies of DYNSAMPLEBASIC: with probability at least \( 1 - (0.625)^\ell > 1 - \frac{1}{n} \), at least one of them will not output \text{fail}.
• The space needed overall is $O(\log^4 n)$ words, for $\ell = O(\log n)$ independent copies with each $L = O(\log n)$ different substreams, all involving $O(\log^2 n)$ space for $CS_j, DS_j$ called with error parameter $1/n$.\(^3\)

1.2 Back to Dynamic Graphs (and Connectivity)

Recall that in this setting, we are given the edges of an $n$-node graph $G = (V, E)$ as a stream of insertions or deletions. In particular, we can consider the following encoding of the graph, as node-edge incidence vectors: to each node-edge incidence or deletions. In particular, we can consider the following encoding of the graph, as node-edge incidence vectors: to each $v \in V = [n]$ corresponds a vector $x_v \in \mathbb{R}^p$ (for $p \overset{\text{def}}{=} \binom{n}{2}$), where

- for $j > v$, $x_v(v, j) = 1_{\{(v, j) \in E\}}$;
- for $j < v$, $x_v(v, j) = -1_{\{(v, j) \in E\}}$.

In particular, non-zero coordinates of $x_v$ correspond to edges incident to $v$ (and the sign of $x_v(v, j)$ is an “artificial orientation” we imposed to it).

**Idea.** We can use dynamic sampling to sample uniformly an edge incident to each $v$. Then, we use these edges to collapse the graph: replacing two nodes $u, v$ connected by an edge we sampled by a “meta-node,” combining the incident edges to both $u$ and $v$.

Ideally, we would like to iterate until either we are left with a single meta-node (in which case the graph was connected) or strictly more than one (in which case the graph was not). The issue, however, lies in this iteration: namely, how to sample edges incident to these “meta-nodes,” while we only have (streaming) access to the edges of the actual graph $G$?

The answer lies in the following crucial observation, which also explains the particular type of $\pm 1$ encoding that was chosen for the vectors $x_v$:

**Claim 2.** For a set $Q \subseteq V$, define the node-edge incidence vector of the “meta-node” $Q$ as $x_Q \overset{\text{def}}{=} \sum_{v \in Q} x_v \in \mathbb{R}^p$. Then, $x_Q \in \{-1, 0, 1\}^V$, and moreover it has a non-zero component at coordinate $(i, j)$ if and only if edge $(i, j)$ exists and crosses from $Q$ to $V \setminus Q$. (That is, $(i, j) \in E$ and $|\{i, j\} \cap Q| = 1$.)

**Proof.** If $(i, j) \notin E$, then $x_Q(i, j) = x_i(i, j) + x_j(i, j) = 0 + 0 = 0$. Otherwise, assume $(i, j) \in E$: if $|\{i, j\} \cap Q| = 2$, then $x_Q(i, j) = x_i(i, j) + x_j(i, j) = 1 - 1 = 0$. If $|\{i, j\} \cap Q| = 0$, then $x_Q(i, j) = 0$ immediately (no term in the sum with something in that coordinate). Only the case $|\{i, j\} \cap Q| = 1$ results in $x_Q(i, j) = 1$ or $x_Q(i, j) = -1$, depending on which of $i, j$ belongs to $Q$. \qed

Another very useful property of these $x_Q$: to compute them, we only need the sketches of the $x_v$’s, since they are linear sketches. Therefore, we can actually sample a random edge from $x_Q$ (for any fixed $Q \subseteq V$), using only $|V| \cdot O(\text{poly log } |V|) = O(n \text{ poly log } n)$ space!

$$\text{DYN SAMPLE FULL}(\sum_{v \in Q} x_v) = \sum_{v \in Q} \text{DYN SAMPLE FULL}(x_v), \quad \forall Q \subseteq V.$$ 

An (almost correct) idea would therefore to do the following:

1. Initiate a sketch (of DYN SAMPLE FULL) for each of the $n$ vectors $x_v$\(^3\)

3 Note that as hinted before, this can be reduced to $O(\log^3 n \cdot \log \log n)$ words, setting the error probability of each $CS_j, DS_j$ to be only $1/\log n$.  

4
2. Check connectivity with the following recursive way, outputting \textbf{no} whenever one of the steps outputs \textit{fail}, and \textbf{yes} if we reach a graph with only one (meta)-node:

(a) sample an edge for each of the meta-nodes $v$ of the current graph;
(b) contract all sampled edges, obtaining a smaller graph with meta-nodes corresponding to connected components of the previous one;
(c) recurse on the new graph.

Since at every recursive step, the number of meta-nodes is easily seen to be reduced by a factor at least 2, only $O(\log n)$ such steps are needed overall.

However, the above has a big flaw: namely, the iterations (calls to \textsc{DynSampleFull} on the “new meta-nodes”) are \textit{not} independent, compromising correctness... (Indeed, the guarantees of \textsc{DynSampleFull} do not apply if the queries are made on \textit{adaptively} chosen combinations of previous queries: an adversary could basically learn enough about the sketches to eventually query some $x_Q$ on which \textsc{DynSampleFull} is ensured to fail.)

A simple solution: we can use new independent copies of \textsc{DynSampleFull} at every iteration... only costing an $O(\log n)$ blowup in the space required (since this is the number of iterations).

2 Dimension Reduction

Barely scratched... next lecture.