Sublinear Time and Space Algorithms 2022B – Lecture 4 Amplifying Success Probability, ℓ_2 Point Queries, and Hash Functions*

Robert Krauthgamer

1 Amplifying Success Probability

To amplify the success probability of Algorithm CountMin (in general case), we use median of independent repetitions (instead of minimum), and analyze it using the standard concentration bounds, as follows.

Theorem 1 [Cormode-Muthukrishnan, 2005]: There is a streaming algorithm for ℓ_1 point queries that uses a linear sketch of dimension $O(\alpha^{-1}\log n)$ (which implies that its memory requirement is this number of words) to achieve accuracy $\alpha \in (0,1)$ with success probability $1-1/n^2$.

Algorithm CountMin++:

1. Run $k = O(\log n)$ independent copies of algorithm CountMin, keeping in memory the vectors $S^1,\ldots,S^k$ (and functions $h^1,\ldots,h^k$)

2. Output: To estimate x_i report the median of all basic estimates, i.e., $\hat{x}_i = \text{median}\{\tilde{x}_i^l : l \in [k]\}$

Lemma:

$$\Pr[\hat{x}_i \in x_i \pm \alpha\|x\|_1] \le 1/n^2.$$

Proof: as seen in class, we define an indicator Y_l for the event that copy $l \in [k]$ succeeds, then use one of the concentration bounds below.

Chernoff-Hoeffding concentration bounds: Let $X = \sum_{i\in[n]} X_i$ where $X_i \in [0,1]$ for $i \in [n]$ are independently distributed random variables. Then

$$\begin{aligned}
\forall t > 0, &\qquad \Pr[|X - \mathbb{E}[X]| \ge t] \le 2e^{-2t^2/n}.\\
\forall 0 < \varepsilon \le 1, &\qquad \Pr[X \le (1-\varepsilon)\,\mathbb{E}[X]] \le e^{-\varepsilon^2\,\mathbb{E}[X]/2}.\\
\forall 0 < \varepsilon \le 1, &\qquad \Pr[X \ge (1+\varepsilon)\,\mathbb{E}[X]] \le e^{-\varepsilon^2\,\mathbb{E}[X]/3}.\\
\forall t \ge 2e\,\mathbb{E}[X], &\qquad \Pr[X \ge t] \le 2^{-t}.
\end{aligned}$$

*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.

Exer: Use these concentration bounds to amplify the success probability of the algorithms we saw for Distinct Elements (say from constant to $1 - 1/n^2$).

Hint: use independent repetitions + median.

2 ℓ_2 Point Query via CountSketch

The idea is to hash coordinates to buckets (similar to algorithm CountMin), but furthermore use tug-of-war inside each bucket (as in algorithm AMS). The analysis will show it is a good estimate with error proportional to $\|x\|_2$ instead of $\|x\|_2$.

Theorem 2 [Charikar, Chen and Farach-Colton, 2003]: One can estimate ℓ_2 point queries within error α with constant high probability, using a linear sketch of dimension $O(\alpha^{-2})$. It implies, in particular, a streaming algorithm.

It achieves better accuracy than CountMin (ℓ_2 instead of ℓ_1), but requires more storage ($1/\alpha^2$ instead of $1/\alpha$).

Algorithm CountSketch:

1. Init: Set $w = 4/\alpha^2$ and choose a hash function $h : [n] \to [w]$
2. Choose random signs $r_1, \ldots, r_n \in \{-1, +1\}$
3. Update: Maintain vector $S = [S_1, \ldots, S_w]$ where $S_j = \sum_{i:h(i)=j} r_i x_i$.
4. Output: To estimate x_i return $\tilde{x}_i = r_i \cdot S_{h(i)}$.

Storage requirement: $O(w) = O(\alpha^{-2})$ words, not counting storage of the random bits.

Correctness: We saw in class that $\Pr[|\tilde{x}_i - x_i|^2 \geq \alpha^2 \|x\|_2^2] \leq 1/4$, i.e., with high (constant) probability, $\tilde{x}_i \in x_i \pm \alpha \|x\|_2$.

Exer: Explain how to amplify the success probability to $1 - 1/n^2$ using the median of $O(\log n)$ independent copies.

3 Hash Functions

Idea: The idea is to replace a truly random function $h : [n] \to [n]$ with something that is easier to store. As a running example, consider $h_{p,q}(i) = pi + q \pmod{n}$, where p, q are chosen at random. This can also be viewed as choosing h from a family $H = \{h_{p,q} : p, q\}$.

To analyze these families formally, we need some definitions.

Independent random variables: Recall that two (discrete) random variables X, Y are independent if

$$\forall x, y \qquad \Pr[X = x, Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

This is equivalent to saying that the conditioned random variable $X|Y$ has exactly the same distribution as X. It implies that in particular $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

The above naturally extends to $k > 2$ variables, and then we say the random variables are mutually (or fully) independent.

Pairwise independence: A collection of random variables $X_1, \ldots, X_n$ is called *pairwise independent* if for all $i \neq j \in [n]$, the variables X_i and X_j are independent.

Example: Let $X, Y \in \{0,1\}$ be random and independent bits, and let $Z = X \oplus Y$. Then X, Y, Z are clearly not mutually (fully) independent, but they are pairwise independent.

Observation: When $X_1, \ldots, X_n$ are pairwise independent, the variance $\mathrm{Var}(\sum_i X_i)$ is exactly the same as if they were fully independent, because

$$\mathrm{Var}(\sum_i X_i) = \mathbb{E}[(\sum_i X_i)^2] - (\mathbb{E}[\sum_i X_i])^2 = \sum_{i,j} \mathbb{E}[X_i X_j] - (\sum_i \mathbb{E}[X_i])^2.$$

Consequently (and this is well-known): If $X_1, \ldots, X_n$ are pairwise independent (and have finite variance), then $\mathrm{Var}(\sum_i X_i) = \sum_i \mathrm{Var}(X_i)$.

The above definition extends to k-wise independence, where every subset of k random variables should be independent.

Pairwise independent hash family: A family H of hash functions $h : [n] \to [M]$ is called *pairwise independent* if for all $i \neq j \in [n]$,

$$\forall x, y \in [M] \qquad \Pr_{h \in H}[h(i) = x, h(j) = y] = \Pr[h(i) = x] \cdot \Pr[h(j) = y].$$

This is the same as saying that $h(1), \ldots, h(n)$ are pairwise independent (when choosing random $h \in H$).

A common scenario is that each $h(i)$ is uniformly distributed over $[M]$, although this is not required in the above definition.

Universal hashing: A family H of hash functions $h : [n] \to [M]$ is called *2-universal* if for all $i \neq j \in [n]$,

$$\Pr_{h \in H}[h(i) = h(j)] \leq 1/M.$$

Observe that 2-universality is weaker than (follows from) pairwise independence when each $h(i)$ is distributed uniformly over $[M]$, but it suffices for many algorithms.

Construction of pairwise independent hashing:

Assume $M \geq n$ and that M is a prime number (if not, we can pick a larger M that is a prime). Pick random $p, q \in \{0, 1, 2, \ldots, M-1\} = [M]$ and set accordingly $h_{p,q}(i) = pi + q \pmod{M}$.

The family $H = \{h_{p,q} : p, q\}$ is pairwise independent because for all $i \neq j$ and all x, y,

$$\Pr_{h \in H}[h(i) \equiv x, h(j) \equiv y] = \Pr_{p,q}\left[\begin{pmatrix} i & 1 \\ j & 1 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} \equiv \begin{pmatrix} x \\ y \end{pmatrix}\right] = \Pr_{p,q}\left[\begin{pmatrix} p \\ q \end{pmatrix} \equiv \begin{pmatrix} i & 1 \\ j & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \end{pmatrix}\right] = \frac{1}{M^2},$$

where we relied on the above matrix being invertible.

Storing a function $h_{p,q}$ from this family can be done by storing p, q, which requires $\log |H| = O(\log M)$ bits. In general, $\log |H|$ bits suffice to store an index of $h \in H$.

One can reduce the size of the range $[M]$ (from large $M \geq n$ to $M = 2$ or say $4/\alpha$), with a small overhead/loss.