Sublinear Time and Space Algorithms 2024A – Lecture 3
ℓ_1 and ℓ_2 Point Queries, Amplifying success probability*

Robert Krauthgamer

1 ℓ_1 Point Query via CountMin

Problem Definition: Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream, and let $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$ be its ℓ_p-norm. Let $\alpha \in (0,1)$ and $p \geq 1$ be parameters known in advance.

The goal is to estimate every coordinate with additive error, namely, given query $i \in [n]$, report $\tilde{x}_i$ such that WHP

$$\tilde{x}_i \in x_i \pm \alpha \|x\|_p.$$

Observe: $\|x\|_1 \geq \|x\|_2 \geq \ldots \geq \|x\|_\infty$, hence higher norms (larger p) give better accuracy. We will see an algorithm for ℓ_1, which is the easiest.

Exer: Show that the ℓ_1 and ℓ_2 norms differ by at most a factor of $\sqrt{n}$, and that this is tight. Do the same for ℓ_2 and ℓ_∞.

It is not difficult to see that ℓ_∞ point query is hard. For instance, with $\alpha < 1/2$ we could recover an arbitrary binary vector $x \in \{0,1\}^n$, which (at least intuitively) requires $\Omega(n)$ bits to store.

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

We will initially assume all $x_i \geq 0$.

Algorithm CountMin:

(Assume all $x_i \geq 0$.)

1. Init: set $w = 4/\alpha$ and choose a random hash function $h : [n] \to [w]$.
2. Update: maintain vector $S = [S_1, \ldots, S_w]$ where $S_j = \sum_{i:h(i)=j} x_i$.
3. Output: to estimate x_i report $\tilde{x}_i = S_{h(i)}$

*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.

Once again, the update step can be implemented in a streaming fashion because it is some linear map $L : \mathbb{R}^n \to \mathbb{R}^w$ (observe that $S_j = \sum_i \mathbb{1}_{\{h(i)=j\}} x_i$).

We call S a *sketch* to emphasize it is a succinct version of the input, and L a *sketching matrix*.

Analysis (correctness): We saw in class that $\tilde{x}_i \geq x_i$ and $\Pr[\tilde{x}_i \geq x_i + \alpha\|x\|_1] \leq 1/4$.

Algorithm CountMin+:

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

2. Output: the minimum of all estimates $\hat{x}_i = \min_{l \in [k]} \tilde{x}_i^l$

Analysis (correctness): As before, $\hat{x}_i \geq x_i$ and

$$\Pr[\hat{x}_i > x_i + \alpha\|x\|_1] \leq (1/4)^k = 1/n^2.$$

By a union bound, with probability at least $1-1/n$, for all $i \in [n]$ we will have $x_i \leq \hat{x}_i \leq x_i + \alpha\|x\|_1$.

Space requirement: $O(\alpha^{-1}\log n)$ words (for success probability $1 - 1/n^2$), without counting memory used to represent/store the hash functions.

Exer: Let $x \in \mathbb{R}^n$ be the frequency vector of a stream of m items (insertions only). Show how to use the CountMin+ sketch seen in class (for ℓ_1 point queries) to estimate the median of x, which means to report an index $j \in [n]$ that with high probability satisfies $\sum_{i=1}^{j} x_i \in (\frac{1}{2} \pm \varepsilon)m$.

General x (allowing negative entries):

Observe that Algorithm CountMin actually extends to general x that might be negative, and achieves the guarantee

$$\Pr[\tilde{x}_i \notin x_i \pm \alpha\|x\|_1] \leq 1/4.$$

Exer: complete the proof.

2 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.

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., $\hat{x}_i = \mathrm{median}_{l \in [k]} \tilde{x}_i^l$

Lemma:

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

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

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

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.

3 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 $\|x\|_1$.

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.

This algorithm 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.