Randomized Algorithms 2015A
Lecture 7 – Data streams and the AMS algorithm for ℓ_2-norm*

Robert Krauthgamer

1 Data streams and the AMS algorithm for ℓ_2-norm

Data stream model:

Motivation: We receive a stream of m items, each in the range $[n]$, and we let x_i be the frequency of item i. Then F_2-frequency moment is just $\|x\|_2^2$. Upon seeing an item $i \in [n]$, we update $x_i \leftarrow x_i + 1$. In the simplest model, we allow any increment $a > 0$. A more general one allows any $a \in \mathbb{R}$, but assumes $x_i \geq 0$. The most general one allows any $x_i \in \mathbb{R}$.

ℓ_p-norm problem:

Input: a vector $x \in \mathbb{R}^n$, given as a stream (sequence) of m updates of the form (i, a), meaning $x_i \leftarrow x_i + a$.

Assumption: updates a are integral and $|x_i| \leq \text{poly}(n)$.

Goal: estimate its ℓ_p-norm $\|x\|_p$. We focus on $p = 2$.

Note: could have $a < 0$ (deletions) and maybe even $x_i < 0$.

Linear sketch (summarization): We shall use a randomized function $L : \mathbb{R}^n \to \mathbb{R}^s$ for small s. The algorithm will only maintain Lx, which is easy to update since:

$$L(x + ae_i) = Lx + a(Le_i).$$

Of course, one has to "construct" L that somehow "stores" $\|x\|_2$.

The memory requirement depends on: dimension s, accuracy needed for each coordinate, and resources (randomness) to compute Le_i.

Note: L is essentially an $s \times n$ (real) matrix.

Theorem 1 [Alon-Matthias-Szegedy'96]: One can estimate the ℓ_2 norm within factor $1 + \varepsilon$ using a linear sketch of $s = O(\varepsilon^{-2})$ memory words. [with high constant probability]

*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.

Remark: We will later discuss how to limit the randomness (because bits that were generated need to be stored).

Algorithm A:

1. Choose initially $r_1, \ldots, r_n$ independently and uniformly at random from $\{-1, +1\}$.
2. Maintain $Z = \sum_i r_i x_i$ (a linear sketch, hence can be updated as above).
3. Output: Z^2.

Analysis of expectation: As seen in class, $\mathbb{E}[Z^2] = \|x\|_2^2$.

We aren't done yet since we want to get $1 + \varepsilon$ accuracy...

Analysis of second moment:

As seen in class, $\mathrm{Var}(Z^2) \leq 2\mathbb{E}[Z^2]$. This is not small enough, but we can repeat several times and take their average.

Algorithm B: Execute $t = O(1/\varepsilon^2)$ independent copies of Algorithm A, denoting their estimates by $Y_1, \ldots, Y_t$, and output their mean $\tilde{Y} = \sum_j Y_j / t$.

Observe that the sketch $(Y_1, \ldots, Y_t)$ is still linear.

Analysis: Clearly, $\mathbb{E}[\tilde{Y}] = \mathbb{E}[Y_1] = \mathbb{E}[Z^2]$.

By independence of the t executions,

$$\mathrm{Var}(\tilde{Y}) \leq \frac{1}{t} \cdot 2(\mathbb{E}[Z^2])^2,$$

and by Chebychev's inequality,

$$\Pr[|\tilde{Y} - \mathbb{E}\tilde{Y}| \geq \varepsilon \mathbb{E}\tilde{Y}] \leq \tfrac{3}{t\varepsilon^2}.$$

Choosing appropriate $t = O(1/\varepsilon^2)$ makes the probability of error an arbitrarily small constant.

Space requirement: $t = O(1/\varepsilon^2)$ words (for constant success probability), without counting memory used to represent/store L.

Concern: How do we store the n values $r_1, \ldots, r_n$?

Exer: For what value of k would the basic analysis work assuming that $r_1, \ldots, r_n$ are k-wise independent?

Exer: What would happen (to accuracy analysis) if the r_i's were chosen as standard gaussians $N(0, 1)$?

Further work studied other ℓ_p-norms and lower bounds.

High probability bound:

Lemma: Let B be a randomized algorithm to approximate some function $f(x)$, i.e.,

$$\forall x, \quad \Pr[B(x) \in (1 \pm \varepsilon) f(x)] \geq 2/3.$$

Then algorithm C which outputs the median of $O(\log \frac{1}{\delta})$ times independent executions of B satisifies

$$\forall x, \quad \Pr[C(x) \in (1 \pm \varepsilon) f(x)] \geq 1 - \delta.$$

Exer: prove this lemma. (Hint: Use the Chernoff-Hoeffding bound.)

Remark: Notice that we obtained a $1+\varepsilon$ estimate for $\|x\|_2^2$, but this immediately gives also a $1+\varepsilon$ estimate for $\|x\|_2$.

2 Count-min sketch for ℓ_1 point queries

ℓ_p point query problem:

Goal: at the end of the stream, given query i, report, for a parameter $\alpha \in (0,1)$,

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

Observe: $\|x\|_1 \geq \|x\|_2 \geq \ldots \geq \|x\|_\infty$, hence higher norms (larger p) gives 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 ℓ_∞ is hard. For instance, with $\alpha = 1/2$ we could recover a binary vector $x \in \{0,1\}^n$, which (at least intuitively) requires $\Omega(n)$ bits to store.

Theorem 2 [Cormode-Muthukrishnan'05]: One can answer ℓ_1 point queries within error α with probability $1 - 1/n^2$ using a linear sketch of $O(\alpha^{-1} \log n)$ memory words.

Algorithm D: (We assume for now $x_i \geq 0$ for all i.)

1. Set $w = 2/\alpha$ and choose a random hash function $h : [n] \to [w]$.
2. Maintain a table $Z = [Z_1, \ldots, Z_w]$ such that $Z_j = \sum_{i:h(i)=j} x_i$.
3. When asked to estimate x_i, return $\tilde{x}_i = Z_{h(i)}$.

Analysis (correctness): As seen in class, $\tilde{x}_i \geq x_i$ holds always, and using Markov's inequality, $\Pr[\tilde{x}_i - x_i \geq \alpha \|x\|_1] \leq 1/2$.

Algorithm E: Execute $t = O(\log n)$ independent copies of algorithm D, i.e., maintain vectors $Z^1, \ldots, Z^t$ and functions $h^1, \ldots, h^t$. Output the estimator $\hat{x}_i = \min_l Z^l_{h^l(i)}$.

Analysis (correctness): Setting $t = O(\log n)$ we have

$$\Pr[|\hat{x}_i - x_i| \geq \alpha \|x\|_1] \leq (1/2)^t = 1/n^2.$$

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: Extend the algorithm to general x. (Hint: replace the min operator by median.)

3 Heavy hitters via point queries

Heavy hitters set: For parameter $\phi \in [0,1]$, define $HH_\phi^p(x) = \{i : \ |x_i|^p \geq \phi \|x\|_p^p\}$.

Observe that the number of HH is bounded by $1/\phi$.

ℓ_p **heavy hitters problem:**

Parameters: $\phi \geq \varepsilon \geq 0$.

Goal: return a set $S \subseteq [n]$ such that

$$HH_\phi^p \subseteq S \subseteq HH_{\phi-\varepsilon}^p.$$

Reduction from HH to point query (for $p = 1$):

Assume we have an algorithm for ℓ_1 point queries with parameter $\alpha = \varepsilon/2$. Amplify the error probability to $1/3n$ (if needed).

Then we compute for every $i \in [n]$ an estimate $\tilde{x}_i$ (this step takes time $O(n \log n)$ or even more) and report the set $S = \{i : \ \tilde{x}_i \geq \phi - \varepsilon/2\}$.

Analysis: With probability $\geq 2/3$, all the n estimates are correct within additive $\varepsilon/2$. In this case, S contains all the ϕ-HH, and is contained in the $(\phi - \varepsilon)$-HH.