**Sketching Algorithms for Big Data** Fall 2017

Lecture 6 — September 19, 2017

*Prof. Jelani Nelson* *Scribe: Demi Guo*

# 1 Overview

1. Wrap up JL Lower bound
2. New Topic: Heavy Hitters (in streams)

# 2 JL Lower bound Optimality

Let's first recap what we talked about last time.

## 2.1 Definition

**Theorem 1** (JL Lemma)**.** *Let $X \subset \mathbb{R}^d$ be any set of size n, and let $\varepsilon \in (0, 1/2)$ be arbitrary. Then there exists a map $f : X \to \mathbb{R}^m$ for some $m = O(\varepsilon^{-2} \lg n)$ such that*

$$\forall x, y \in X, (1-\varepsilon)\|x-y\|_2^2 \leq \|f(x)-f(y)\|_2^2 \leq \|x-y\|_2^2$$

In this lecture, we want to prove the optimality of that lower bound. In last lecture, we used lecture 3 to prove such lower bound $O(\varepsilon^{-2} \lg n)$ can be achieved. Now, we only need to prove that m must be greater or equal than $O(\varepsilon^{-2} \lg n)$.[2]. (note that our lecture is actualy based on the newer version of the paper which is not published)

## 2.2 Intuition

Here, we give a high level introduction of the main ideas in our proof. The proof goes via a counting argument. More specifically, we construct a large family $\mathcal{P} = \{P_1, P_2, \cdots\}$ of very different sets of n points in $\mathbb{R}^d$. We then assume all point sets in $\mathcal{P}$ can be embedded into $\mathbb{R}^m$ while preserving all pairwise distances to within $(1+\varepsilon)$. Letting $f_1(P_1), f_2(P_2), \cdots$, denote the embedded poitn sets, we then argue that our choice of $\mathcal{P}$ ensures that any two $f_i(P_i)$ and $f_j(P_j)$ must be very different. If m is too low, this is impossible as there are not enough sufficiently different point sets in $\mathbb{R}^m$.

## 2.3 Encoding Argument

Last time, we defined how point sets $\mathcal{P}$ are chosen as follows: Let $e_1, e_2, \cdots, e_d$ denote the standard unit vectors $\mathbb{R}^d$. For now, assume that $d = n/lg(1/\varepsilon))$ and $\varepsilon \in (0, 1)$. For any set $S \subset [d]$ of size

k, where $k = \varepsilon^{-2}/c_0^2$, define a vector $y_S = \sum_{j \in S} e_j/\sqrt{k}$. Here, $c_0$ is a sufficiently large constant. Let $Q = n - d - 1$. For every choice of Q sets $S_1, \cdots S_Q \subset [d]$ of k indices each, we add a point set $P$ to $\mathcal{P}$. The point set $P$ is simply $\{0, e_1, \cdots, e_d, y_{S_1}, \cdots, y_{S_Q}\}$. This gives us a family $\mathcal{P}$ with $|\mathcal{P}| = \binom{d}{k}^Q$

**Claim 2.** *Given that it is possible to embed every point set in $\mathcal{P}$ into $\mathbb{R}^m$ while perserving pairwise distances to within $(1+\varepsilon)$. We can then encode every point set into a big string of length $O(nm)$. Moreover, the encoding is injective: namely, the encoding guarantees that $P_i$ can be uniquely recovered from the encoding.*

Proving the claim is sufficient since then we will have $|\mathcal{P}| = \binom{d}{k}^Q \leq 2^{O(nm)}$, which can be simplified to $m = \Omega(\varepsilon^{-2} \lg n)$ when $\varepsilon > 1/n^{0.4999}$.
Thus, now we only need to find such an encoding algorithm, as we mentioned last time.

## 2.4 Encoding Method

(Disclaimer: In following proof, sometimes, the constant factor might not be exact, but it won't affect the correctness of the proof, since we can ignore constant factor in complexity analysis).
Before we go into encoding method, let's first take a look at the embedded norm and dot product. First, it's easy to see that $\|e_i\| = 1 \leq 1$ and $\|y_S\| \leq k(1/\sqrt{k})^2 = 1$. Now, for point set P, and it's corresponding mapping f that satisfies the JL-property. We have following two claims:

**Claim 3.** *f preserve norms of the vectors $x \in P$ to within $(1+\varepsilon)$.*

*Proof.*

$$(1-\varepsilon)\|x\|_2^2 = (1-\varepsilon)\|x-0\|_2^2 \leq \|f(x)-f(0)\|_2^2 = \|f(x)\|_2^2 = \|f(x)-f(0)\|_2^2 \leq (1+\varepsilon)\|x-0\|_2^2 = (1+\varepsilon)\|x\|_2^2$$

□

**Claim 4.** *f must preserve inner products $\langle e_j, y_S \rangle$ up to an additive of $O(\varepsilon)$.*

*Proof.*

$$\|f(e_j) - f(y_S)\|_2^2 = \|f(e_j)\|_2^2 + \|f(y_S)\|_2^2 - 2\langle f(e_j), f(y_S)\rangle \Rightarrow \quad (1)$$
$$2\langle f(e_j), f(y_S)\rangle \in (1 \pm \varepsilon)\|e_j\|_2^2 + (1 \pm \varepsilon)\|y_S\|_2^2 - (1 \pm \varepsilon)\|e_j - y_S\|_2^2 \Rightarrow \quad (2)$$
$$2\langle f(e_j), f(y_S)\rangle \in 2\langle e_j, y_S \ \pm \varepsilon(\|e_j\|_2^2 + \|y_S\|_2^2 + \|e_j - y_S\|_2^2) \Rightarrow \quad (3)$$
$$\langle f(e_j), f(y_S)\rangle \in \langle e_j, y_S\rangle \pm 4\varepsilon \quad (4)$$

□

This means that after applying f, there remains a gap of $(c_0 - 8)\varepsilon = \Omega(\varepsilon)$ between $\langle f(e_j), f(y_S)\rangle$ depending on whetehr or not $j \in S$. With this observation, we are ready to find the first encoding method.

### 2.4.1 Basic Method

We can encode $P_i = \langle 0, e_1, \cdots, y_{S_1}, \cdots \rangle$ with $\langle f_i(0), f_i(e_1), \cdots, f_i(y_{S_1}), \cdots \rangle$. This encoding method is injective since we can decode $P_i$ by decoding all the original $S_1, \cdots, S_Q$ using following property:

$$\langle e_j, y_S \rangle = 0 \text{ if } j \notin S \tag{5}$$

$$\text{and } \langle e_j, y_S \rangle = c_0 \varepsilon \text{ if } j \in S \tag{6}$$

Since $\langle f_i(e_j), f_i(y_S) \rangle$ is $(1+\varepsilon)$ approximation of $\langle e_j, y_S \rangle$, there will still be a gap (e.g. $2c'\varepsilon$ for some constant c') depending whether $j$ is in $S$. Thus, by calculating all pairs of $e_j$ and $S_z$ and $\langle f_i(e_j), f_i(y_{S_z}) \rangle$, we will know whether $j$ is in $S_z$, and recover $S_z$ $\forall z$.
However, this basic method doesn't work, because we can't encode into $O(nm)$ bits: each $f_i(x)$ $\forall x \in P_i$ is a real number, which will cost $\infty$ bits to encode.

### 2.4.2 Fix 1

The idea is that we approximate $f_i(x)$ to some finite points, and claim that we can still use approximate $f_i(X)$ to recover $P_i$ use a similar strategy.
First, we know that $\forall j$, $\|f_i(e_j)\|, \|f_i(y_{S_j})\| \leq 1+\varepsilon$. If we denote $B_\infty^m$ as a unit ball in dimension space under $l_\infty$ norm. Then, it's obvious that, $\forall j$. $f_i(e_j)$ and $f_i(y_{S_j}) \in (1+\varepsilon)B_\infty^m$.
Let's approximate all $f_i(e_j)$ and $f_i(y_{S_j})$ to a multiple of $\gamma$ (each dimension's coordinate is a multiple of $\gamma$). Denote those approximations as $\hat{f}_i$.
We claim we can still recover $P_i$ using the approximate $f_i$. This is because $\langle \hat{a}, \hat{b} \rangle$ is within $(1+\varepsilon)$ of $\langle a, b \rangle$ for $\gamma = \frac{\varepsilon}{\sqrt{m}}$:

$$\langle \hat{a}, \hat{b} \rangle = \sum_i \hat{a_i} \hat{b_i} \tag{7}$$

$$= \sum_{i=1}^{m} (a \pm \gamma)(b \pm \gamma) \tag{8}$$

$$= \langle a, b \rangle \pm \sum_i \gamma(|a_i| + |b_i|) + \gamma^2 m \tag{9}$$

$$= \langle a, b \rangle \pm \gamma(\|a\|_1 + \|b\|_1) + \gamma^2 m \tag{10}$$

$$= \langle a, b \rangle \pm \gamma\sqrt{m}(\|a\|_2 + \|b\|_2) \pm \gamma^2 m \tag{11}$$

$$= \langle a, b \rangle \pm \gamma\sqrt{m}c'(1+\varepsilon) \pm \gamma^2 m \tag{12}$$

$$= \langle a, b \rangle \pm c'\varepsilon(1+\varepsilon) \pm \varepsilon^2 \tag{13}$$

Thus, $\langle \hat{f}_i(e_j), \hat{f}_i(y_{S_z}) \rangle$ is $(1+\varepsilon)$ approximation of $\langle e_j, y_{S_z} \rangle$. Thus, there's still a gap between the case $j$ is not in $S_z$ and $j$ is in $S_z$.

Now, since there are $N_1 = (\frac{1+\varepsilon}{\gamma})^m = (\frac{(1+\varepsilon)\sqrt{m})}{\varepsilon})^m$ such points (multiple of $\gamma$) in $(1+\varepsilon)B_\infty^m$, solving $\binom{d}{k}^Q \leq N_1$, we get a bound $m = \Omega(\frac{1}{\varepsilon^2} \frac{\lg n}{lg \frac{m}{\varepsilon}})$, which is still not good enough. (In other words, we get an encoding algorithm with $O(\lg N_1)$ bits instead of $O(nm)$ bits)

### 2.4.3 Fix 2

The problem with Fix 1 is that there are still a lot of multiples of $\gamma$. We want approximate $f_i$ using a smaller set of points.

We first claim that if $\hat{f}_i$ is still a $(1+\varepsilon)$ approximation of $f_i$ in $l_2$ norm, then we can use a similar strategy to recover $P_i$:
We have $\|\hat{f}_i(e_j)-f_i(e_j)\|_2^2 \leq \varepsilon$ and $\|\hat{f}_i(y_{S_l})-f_i(y_{S_l})\|_2^2 \leq \varepsilon$ , then by triangle inequality, the distance $\|\hat{f}_i(e_j)-\hat{f}_i(y_{S_l})\|_2^2$ is also a $(1+O(\varepsilon))$ approximation to $\|e_j - y_{S_z}\|$. This follows that the inner products will be preserved. Thus, like Fix 1 and basic method, we can still use the inner product $\langle \hat{f}_i(e_j), \hat{f}_i(y_{S_z})\rangle$ to recover whether j is in $S_z$.

Similar to Fix 1, since $\forall j,\ \|f_i(e_j)\|, \|f_i(y_{S_j})\| \leq 1+\varepsilon$, we know $\forall j,\ f_i(e_j), f_i(y_{S_j}) \in (1+\varepsilon)B_2^m$ (where $B_2^m$ is a unit ball in m dimensional space under $l_2$ norm).
Consider a minimum cover C of $(1+\varepsilon)B_2^m$ using $\varepsilon B_2^m$ balls (minimum cover here means: a cover with minimized $|C|$). Then for each $f_i(x) \in (1+\varepsilon)B_2^m$ we can find the $\varepsilon B_2^m$ ball $B_0$ in the cover, and approximate $f_i(x)$ using the center of the ball. Since the ball has size $\varepsilon$, we know it's still a $(1+\varepsilon)$ approximation of $f_i(x)$. In this way, instead of encoding $f_i(x)$ we can encode the index of the $\varepsilon B_2^m$ ball containing $f_i(x)$, and approximate it using the center of that ball. We thus get an encoding algorithm with $\lg(|C|)$ bits.

Now, the question is, what is $|C|$? Let's first take a look at the following 2 lemmas.

**Lemma 5.** *Let $\mathcal{K}$ be a convex body, and $P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K})$ be any maximal packing of $\mathcal{K}$ using $\frac{\alpha}{2}\mathcal{K}$. (A maximal packing of A using B = a bunch of disjoint bodies B with centers inside A, s.t. you can't add more body B anymore). Then, $P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K}) \leq (1+\frac{2}{\alpha})^{dim(\mathcal{K})}$.*

*Proof.* (If it's confusing that we are using "convex" body, you can think of $\mathcal{K}$ as a unit ball, and $\frac{\alpha}{2}\mathcal{K}$ as a ball with radius $\frac{\alpha}{2}$)
Since the centers of all $\frac{\alpha}{2}\mathcal{K}$ bodies are contained in $\mathcal{K}$ body. Thus, all $\frac{\alpha}{2}\mathcal{K}$bodies will be fully contained in a $(1+\frac{\alpha}{2})\mathcal{K}$ body. Since all $\frac{\alpha}{2}\mathcal{K}$ bodies are disjoint. We have $VOL(\frac{\alpha}{2}\mathcal{K})P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K}) \leq VOL((1+\frac{\alpha}{2})\mathcal{K}$, which gives us the desired result: $P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K}) \leq (1+\frac{2}{\alpha})^{dim(\mathcal{K})}$. □

**Lemma 6.** *We want to cover $\mathcal{K}$ using some copies of $\alpha\mathcal{K}$ (a smaller scaled version of $\mathcal{K}$). Define $N(\mathcal{K}, \alpha\mathcal{K})$ as the minimum number of copies of $\alpha\mathcal{K}$ we need to cover body $\mathcal{K}$. Then, $N(\mathcal{K}, \alpha\mathcal{K}) \leq P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K})$.*

*Proof.* For any maximal packing of $\mathcal{K}$ using $\frac{\alpha}{2}\mathcal{K}$ bodies, we can double the size of the bodies to $\alpha\mathcal{K}$. Then, those bodies will form a cover of $\mathcal{K}$:
If there is a point $p$ in $\mathcal{K}$ not covered. Then, $p$ is at least $\alpha = \frac{\alpha}{2} + \frac{\alpha}{2}$ away from all centers of the bodies. Then, if we draw a $\frac{\alpha}{2}\mathcal{K}$ body centered at $p$, it won't overlap with any other bodies. Thus, it implies that P is not a maximal packing, which contradicts with the statement. Thus, all points are covered.
Thus, it follows that $N(\mathcal{K}, \alpha\mathcal{K}) \leq P(\mathcal{K}, \frac{\alpha}{2}\mathcal{K})$. □

Combining Lemma 5 and Lemma 6, we have:

**Corollary 7.** $N(\mathcal{K}, \alpha\mathcal{K}) \leq (1 + \frac{2}{\alpha})^{dim(\mathcal{K})}$

Back to our original problem (in Fix 2), we have $\mathcal{K} = (1+\varepsilon)B_2^m$, $dim(\mathcal{K}) = m$, and $\alpha = \frac{\varepsilon}{1+\varepsilon}$. Thus, we can get a bound for $|C| = 2^{\Omega(m \lg(1/\varepsilon))}$. Tge $\lg(1/\varepsilon)$ factor loss leaves us with a lower bound on $m$ of no more than $m = \Omega(\varepsilon^{-2} \lg(\varepsilon^2 n/\lg(1/\varepsilon))/\lg(1/\varepsilon))$, roughly recovering the lower bound of Alon [1] by a different argument.

### 2.4.4 Fix 3

Now, let's consider our final fix. To encode $\hat{f}_i(e_j)$, we can use the same method in Fix2:
Observe that we chose $d = n/\lg(1/\varepsilon)$. Thus we can spend up to $O(m \lg(1/\varepsilon))$ bits encoding each $f_i(e_j)$'s. Thus, we simply encode approximations $\hat{f}_i(e_j)$ by specifying indices into a covering $C_2$ of $(1+\varepsilon)B_2^m$ by $\varepsilon B_2^m$ as outlined above.

For the $f_i(y_{S_z})$'s, we have to be more careful. First, we define the $d \times m$ matrix A having the $\hat{f}_i(e_j)$ as rows. **Note** that this matrix can be reconstructed from the part of encoding specifying the $\hat{f}_i(e_j)$'s. Now observe that the $j'th$ coordinate of $Af_i(y_{S_z}$ is within $O(\varepsilon)$ of $\langle e_j, y_{S_z}\rangle$. The coordinates of $Af_i(y_{S_z})$ thus determine $S_z$. We therefore seek to encode $Af_i(y_{S_z})$ efficiently.

To encode $Af_i(y_{S_z})$, first note that $\|Af_i(y_{S_z})\|_\infty = O(\varepsilon)$, because it's within $O(\varepsilon)$ of $\langle e_j, y_{S_z}\rangle = O(\varepsilon)$. If W denotes the $\leq m$ dimensional subspace spanned by the column of A, we also have that $Af_i(y_{S_z}) \in W$. Now define the convex body $T = B_\infty^d \cap W$, where $B_\infty^d$ denote the $l_\infty$ unit cube in $\mathbb{R}^d$. Then, $Af_i(y_{S_z}) \in O(\varepsilon)T$.

Now, recall that there is a gap of $\Omega(\varepsilon)$ between inner products $\langle \hat{f}_i(e_j), \hat{f}_i(y_{S_z})\rangle$ depending on whether $j \in S_z$ or not. Letting $c_1$ be a constant such that the gap is more than $2c_1\varepsilon$, this implies that if we approximate $Af_i(y_{S_z})$ by a point $\hat{f}_i(y_{S_z})$ such that $(\hat{f}_i(y_{S_z}) - Af_i(y_{S_z})) \in c_1\varepsilon \cdot B_\infty^d$, then the coordinates of $\hat{f}_i(y_{S_z})$ still uniquely determine the indices $j \in S_z$. Exploiting that $Af_i(y_{S_z}) \in O(\varepsilon)T$m we therefore create a covering $C_\infty$ of $O(\varepsilon)T$ by copies of $c_1\varepsilon T$ and approximate $Af_i(y_{S_z})$ by the center of the convex body in $C_\infty$ containing it (just like Fix 2). Applying Corollary 7, we have $\alpha = c_1$ , $\mathcal{K} = \varepsilon T$, and $dim(\varepsilon T) \leq m$ (since it is contained in W). Thus, $|C_\infty|$ is bounded by $|C_\infty| = 2^{O(m)}$. Specifying indices into $C_\infty$ thus costs only $O(m)$ bits to encode $Af_i(y_{S_z})$. Encoding for all $z$, we only need to use in total $O(nm)$ bits.

To summarize, the decoding process looks like following:
We have a fixed covering $C_2$ of the fixed ball $(1+\varepsilon)B_2^m$ (centered at origin). Note that we don't need any encoding to "decode" $C_2$, because all conditions is fixed if we use some constant bits to encode $m, \varepsilon$ etc.
Let $a_j$ be the index of the ball in the covering that contains $f_i(e_j)$. We can find that ball in the fixed covering $C_2$, and approximate $f_i(e_j)$ using the center of that ball.
Using those $\hat{f}_i(e_j)$, we can recover matrix A, and find the covering $C_\infty$ of A. Note that we only need A to decode this covering. Now, for each $Af_i(y_{S_z})$, we have the encoded index of the ball in $C_\infty$. Using the decoded covering $C_\infty$, we can find the center of that ball, and use it to approximate $Af_i(y_{S_z})$. Now, by considering each bit $j$ in $Af_i(y_{S_z})$, and compare them with certain threshold (e.g. $c_1\varepsilon$), we know if j is in $S_z$. In this way, we can decode all $S_z$ and thus $P_i$.

# 3 Heavy Hitters

We have following definitions:

1. $\ell_1$ point query: $query(i) = x_i \pm \varepsilon\|x\|_1$
2. $\ell_1$ heavy hitters: $query()$ return $L \in [n]$ s.t. :
   (a) $|x_i| > \varepsilon\|x\|_1 \implies i \in L$
   (b) $|L| = O(1/\varepsilon)$

Note that the number of $\varepsilon$-heavy hitters, i.e. those $i$ satisfying $|x_i| > \varepsilon\|x\|_1 \to i \in L$, is less than $1/\varepsilon$, so the second requirement is just saying that $L$ should not be more than a constant factor larger than this maximum possible size.

The heavy hitters problem shows up, for example, when we are trying to find frequent items in a data stream. In the turnstile model with deletions, if we interpret updates in one time period $T$ as decrementing frequencies and in some other disjoint time interval $T'$ as increasing frequencies, then note that $x$ during a query will be the difference of two frequencies. Then a heavy hitter corresponds to an index $i$ that *changed* significantly in frequency, and thus turnstile heavy hitters algorithms can also be used to detect large frequency changes.

We will also study the notion of point query with a tail guarantee, and "$\varepsilon$-tail heavy hitters".

1. $\ell_1$ point query with a tail guarantee: $query(i) = x_i \pm \varepsilon\|x_{\overline{[1/\varepsilon]}}\|_1$
2. $\ell_1$ tail heavy hitters: $query()$ return $L \in [n]$ s.t. :
   (a) $|x_i| > \varepsilon\|x_{\overline{[1/\varepsilon]}}\|_1 \implies i \in L$
   (b) $|L| = O(1/\varepsilon)$

Here we use $x_{\overline{[k]}}$ to denote the vector $x$ after zeroing out its largest $k$ entries in magnitude. Note that the number of $i$ such that $|x_i| > \varepsilon\|x_{\overline{[]k]}}\|_1$ is at most $k + 1/\varepsilon$, since other than the set $S \subset [n]$ of top $k$ entries in $x$, the number of other $i$ satisfying $x_i > \varepsilon \sum_{j \notin S} |x_j|$ must be less than $1/\varepsilon$.

We will see next lecture that $\ell_1$ point query with $\varepsilon$ error (and in fact even with the tail guarantee) can be achieved using space $O(\varepsilon^{-1}\log(1/\delta))$ words to have failure probability $\delta$ per query. We will also show how to achieve space $O(\varepsilon^{-1}\log n)$ words to solve $\varepsilon$-tail heavy hitters in $\ell_1$.

# References


[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 20–29, 1996.

[2] Kasper Larsen, Jelani Nelson. Optimality of the Johnson-Lindenstrauss Lemma. *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* 2017.