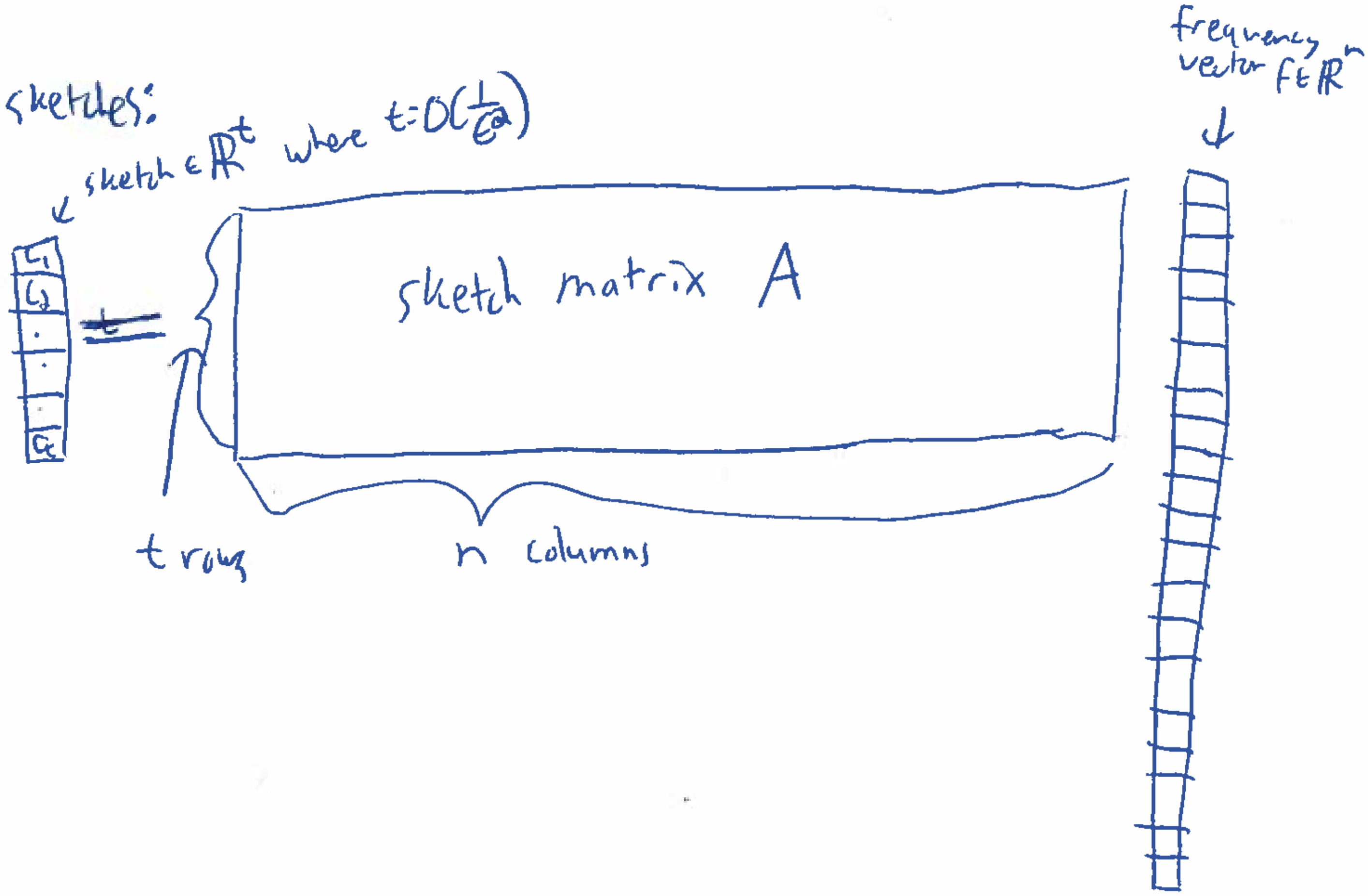
Linear Sketches, Johnson-Lindenstrauss, Random Projections for Dimensionality Reduction, and a Geometric Perspective on F_2 Sketching Algorithms

Linear sketches:



Recall the Tug-of-War Sketch from last lecture:

- $t \leftarrow c \cdot \frac{1}{\varepsilon^2}$ for some $c = O(1)$
- Choose $h_1, \ldots h_t : [n] \to \{\pm 1\}$ at random from a pairwise independent family of hash functions.
- Initialize counters $c_1, \ldots c_t$ to 0.
- While processing update (a_j, δ_j):
  - For $i = 1, \ldots, t$
    - $c_i \leftarrow c_i + \delta_j \cdot h_i(a_j)$
- Output $\frac{1}{t} \cdot \sum_{i=1}^{t} c_i^2$.

$\boxed{\text{Sketch matrix } A \text{ for Tug-of-War sketch is defined as } A_{ij} := h_i(j).}$

Any algorithm of this form is called a linear sketch.
Any linear sketch works in the general turnstile update model.

Proof: Consider update (a_j, δ_j). What effect does this have on the frequency vector?

all 0's except a 1 in entry a_j

$$f_{a_j} \leftarrow f_{a_j} + \delta_j$$

in vector notation: $f \leftarrow f + \delta_j \cdot e_{a_j}$

What effect does this have on the sketch Af?

$$Af \leftarrow A(f + \delta_j e_{a_j})$$

by linearity, this is the same as $Af \leftarrow Af + \underbrace{A(\delta_j e_{a_j})}$

$= \delta_j \cdot A \cdot e_{a_j}$

$= \delta_j \cdot [a_j\text{th column of } A]$

So update to sketch only depends on a_j and δ_j and can be computed in a streaming manner.

- Note: [LNW14] prove a converse to this: Any turnstile streaming algorithm must be a linear sketch.

- Note: A has $t \cdot n$ entries, so a streaming algorithm cannot afford to store in explicitly. In Tug-of-War sketch, A has a succinct implicit representation since it is fully specified by the hash functions $h_1, \dots h_t$ (this is $O(\frac{\log n}{\varepsilon^2})$ bits total).

Dimensionality Reduction via Random Projections.

Note: We showed last time that with probability at least $\frac{2}{3}$, in Tug-of-War sketch
$\swarrow$ c_i is ith counter in Tug-of-War sketch

$$\frac{1}{t}\sum c_i^2 \in \left[(1-\epsilon)\|f\|_2^2,\ (1+\epsilon)\|f\|_2^2\right].$$

That is, the ~~linear~~ linear map $\frac{1}{\sqrt{t}}A : \mathbb{R}^n \to \mathbb{R}^t$ preserved the norm of f up to a factor of $(1\pm\epsilon)$ with probability $\geq \frac{2}{3}$.

Note: Any random projection matrix A that preserves the <u>norm</u> of any fixed vector with high probability also preserves <u>distances</u> between vectors with high probability. Formally:

Claim: Suppose for any fixed vector $x \in \mathbb{R}^n$, n big, $\|Ax\| \in [(1-\epsilon)\|x\|, (1+\epsilon)\|x\|]$ with probability $\geq 1-\delta$ over the random choice of A. Then for any set of m vectors $x_1, \ldots, x_m \in \mathbb{R}^n$, with probability $\geq 1-\binom{m}{2}\cdot\delta$ over the random choice of A, it holds that

$$\forall i \neq j,\quad \|Ax_i - Ax_j\| \in \left[(1-\epsilon)\|x_i - x_j\|,\ (1+\epsilon)\|x_i - x_j\|\right].$$

Proof: For each fixed pair $i \neq j$, apply the hypothesis to the vector $x := x_i - x_j$. Union bound over all $\binom{m}{2}$ pairs $i \neq j$.

Example application: Clustering. Say you want to cluster m points $x_1 \ldots x_m \in \mathbb{R}^n$ into k clusters so that all points in the same cluster are close to each other (i.e. "similar"). Rather than running a clustering algorithm on $x_1, \ldots, x_m$, which is expensive if n is big, first project $x_1 \ldots x_m$ into a much lower dimensional space and run the algorithm on the low-dimensional vectors. Much [illegible] and if the projection preserves pairwise distances, the output clustering [illegible] optimal.

A Different F_2 Algorithm

$O(\frac{1}{\epsilon^2})$

Rather than choosing the random $t \times n$ matrix A as per the Tug-of-War sketch, choose each entry of A to be an independent $N(0,1)$ variable. (Note: A does not have a small-space representation, but let us ignore this for now).

Claim: With probability at least $\frac{2}{3}$, $\frac{1}{t}\|Af\|_2^2 = (1 \pm \epsilon) \cdot \|f\|_2^2$.

This is the Johnson-Lindenstrauss Lemma.

Proof: Fact 1: $\mathbb{E}[\frac{1}{t} \cdot \|Af\|_2^2] = \|f\|_2^2$.

Proof: For each $i \in [t]$, $\mathbb{E}[(Af)_i^2] = \mathbb{E}[(\sum_{j=1}^{n} A_{ij} f_j)^2]$

$$= \mathbb{E}\Big[\sum_{j,j'} A_{ij} \cdot A_{ij'} \cdot f_j \cdot f_{j'}\Big]$$

linearity of expectation

$$= \sum_{j,j'} f_j \cdot f_{j'} \, \mathbb{E}[A_{ij} \cdot A_{ij'}] = \sum_j f_j^2 \cdot \mathbb{E}[A_{ij}^2]$$

if $j \neq j'$, this $= \mathbb{E}[A_{ij}] \cdot \mathbb{E}[A_{ij'}] = 0$,

A_{ij}^2 is distributed according to the χ^2-distribution with 1 degree of freedom. Its expected value is 1, so the final sum is just $\sum_j f_j^2$ ∎

Can bound the probability that $\frac{1}{t}\|Af\|_2^2$ deviates significantly from its expectation can be bounded via standard arguments about χ^2 distributions. I will skip the details.

An alternative proof of Fact 1:

Recall $(Af)_i = \sum_{j=1}^{n} A_{ij} \cdot f_j$, where each A_{ij} is independent and distributed $\sim N(0,1)$. A standard fact about the normal distribution is that $(Af)_i$ has the same distribution as $\|f\|_2 \cdot Y$ where $Y \sim N(0,1)$.

Thus, $\mathbb{E}[(Af)_i^2] = \mathbb{E}[(\|f\|_2 \cdot Y)^2] = \|f\|_2^2 \cdot \mathbb{E}[Y^2] = \|f\|_2^2$

Stable Distributions and Approximating $\|f\|_p$ for $0<k\le 2$. [Indyk 2006

Recall $\|f\|_p = \left(\sum_i |f_i|^p\right)^{1/p}$.

Definition: Let $p>0$ be a real number. A probability distribution D_p over the reals is <u>p-stable</u> if for all integers $n \ge 1$ and all $\vec{c} = (c_1, \ldots, c_n) \in \mathbb{R}_+^n$, then the following holds. If $X_1, \ldots X_n$ are independent and each $X_i \sim D_p$, then $c_1 X_1 + \ldots + c_n X_n$ has the same distribution as $\bar{c} X$ where $X \sim D_p$ and

$$\bar{c} = \left(\sum_i c_i^p\right)^{1/p} = \|\vec{c}\|_p.$$

Fact: Stable distributions exist for all $p \in (0,2]$. The normal distribution is 2-stable. The <u>Cauchy distribution</u> is 1-stable.

Key point: For any $p \le 2$, it is easy to generate a sample from a p-stable distribution [Chambers-Mallows-Stuck 1976].

• Note: the expected value of the Cauchy distribution is infinite.

A Basic Estimator for $\|f\|_p$ for any $0 < p \leq 2$.

(Recall for $p > 2$, estimating $\|f\|_p$ requires $\tilde{\Theta}(n^{1-1/p})$ space)

- For each $j \in [n]$, let $c_j \sim D_p$ ← requires storing n numbers, but let's ignore this for now.
- $x \leftarrow 0$
- When processing update (a_j, δ_j):
  $x \leftarrow x + \delta_j \cdot c_{a_j}$.
- Output $x / \text{median}(D_p)$ ← can't use $\text{mean}(D_p)$ since this might be infinite (e.g. for $p=1$, D_p is Cauchy)

Claim: The median of the distribution of the basic estimator is precisely $\|f\|_p$.

Proof: By p-stability of D_p, $x \sim \|f\|_p \cdot Y$ where $Y \sim D_p$.
So the estimate is distributed as $\left(\frac{\|f\|_p}{\text{median}(D_p)}\right) \cdot Y$, which has median $\frac{\|f\|_p}{\text{median}(D_p)} \cdot \text{median}(Y) = \|f\|_p$ ∎

Final Algorithm: Output the median of $t = O\left(\frac{\log(\frac{1}{\delta})}{\epsilon^2}\right)$

Copies of the Basic Estimator.

Two additional Issues:

1) The Basic Estimator required storing n numbers drawn from D_p

2) In addition to there being $\underset{\text{bits}}{n}$ numbers, these are real numbers so may take infinitely many bits to represent.

Deal with 2) by rounding to $O(\log(n \cdot \epsilon^{-1} \cdot \delta^{-1}))$ bits of precision and argue that this doesn't introduce too much error.

Deal with 1) by using a pseudorandom number generator which requires storing only a small random seed and generates bits that "look random" to the Basic Estimator.