Rough Algorithm Overview

$\min_{x \text{ in } R^d} |Ax-b|_1 = \min_{x \text{ in } R^d} |Ux - b'|_1$

Sample poly(d/ε) rows of U◦b' proportional to their l_1-norm.

Compute poly(d)-approximation

Compute well-conditioned basis

Find x' such that $|Ax'-b|_1 \leq \text{poly}(d) \min_{x \text{ in } R^d}$
Let b' = b-Ax' be the residual

Find a basis A=UW so that for all x in R^d,
$|x|_1/\text{poly}(d) \leq |Ux|_1 \leq \text{poly}(d)\, |x|_1$

Takes nnz(A)

Now generic linear programming is efficient

Solve l_1-regression on the sample, obtaining vector x, and output x

Will focus on showing how to quickly compute

1. A poly(d)-approximation

2. A well-conditioned basis

Sketching Theorem

Theorem

- There is a probability space over $(d \log d) \times n$ matrices R such that for any $n \times d$ matrix A, with probability at least 99/100 we have for all x:

$$|Ax|_1 \leq |RAx|_1 \leq d \log d \cdot |Ax|_1$$

Embedding

- is linear
- is independent of A
- preserves lengths of an infinite number of vectors

Application of Sketching Theorem

Computing a d(log d)-approximation

- Compute RA and Rb
- Solve $x' = \mathrm{argmin}_x |RAx-Rb|_1$
- Main theorem applied to $A \circ b$ implies x' is a d log d – approximation
- RA, Rb have d log d rows, so can solve l_1-regression efficiently

Application of Sketching Theorem

Computing a well-conditioned basis

1. Compute RA
2. Compute W so that RAW is orthonormal (in the l_2-sense)
3. Output U = AW

U = AW is well-conditioned because

$$|AWx|_1 \leq |RAWx|_1 \leq (d \log d)^{1/2} |RAWx|_2 = (d \log d)^{1/2} |x|_2 \leq (d \log d)^{1/2} |x|_1$$

and

$$|AWx|_1 \geq |RAWx|_1/(d \log d) \geq |RAWx|_2/(d \log d) = |x|_2/(d \log d) \geq |x|_1 /(d^{3/2} \log d)$$

Sketching Theorem

Theorem:

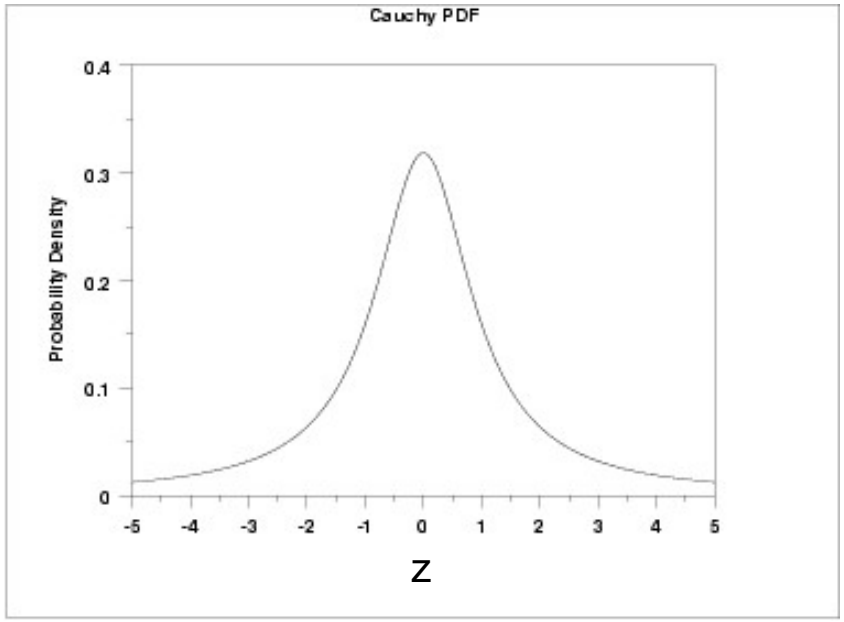
- There is a probability space over $(d \log d) \times n$ matrices R such that for any $n \times d$ matrix A, with probability at least 99/100 we have for all x:

$$|Ax|_1 \leq |RAx|_1 \leq d \log d \cdot |Ax|_1$$

A dense R that works:

The entries of R are i.i.d. Cauchy random variables, scaled by $1/(d \log d)$

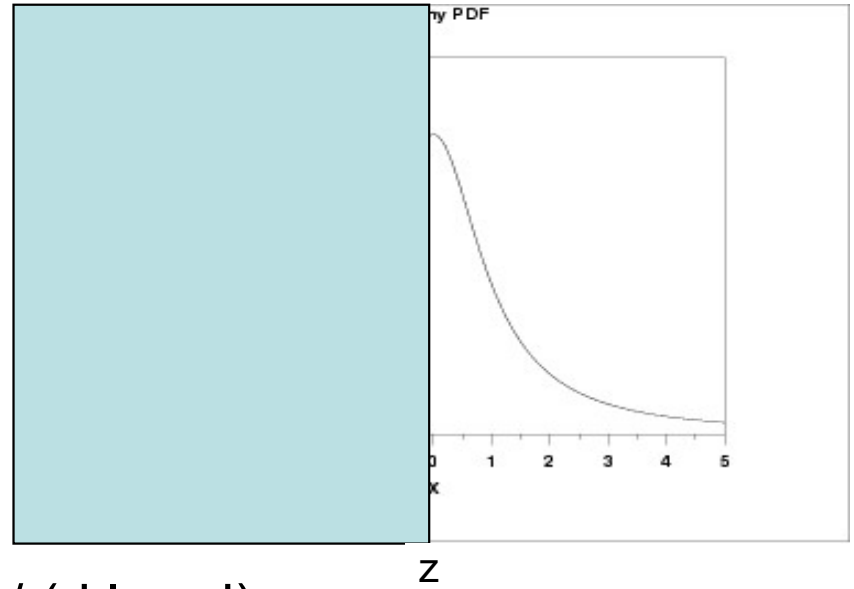
Cauchy Random Variables



- pdf(z) = $1/(\pi(1+z^2))$ for z in $(-\infty, \infty)$
- Undefined expectation and infinite variance
- 1-stable:
  - If $z_1, z_2, \ldots, z_n$ are i.i.d. Cauchy, then for $a \in R^n$,

    $$a_1 \cdot z_1 + a_2 \cdot z_2 + \ldots + a_n \cdot z_n \sim |a|_1 \cdot z, \text{ where z is Cauchy}$$
- Can generate as the ratio of two standard normal random variables

Proof of Sketching Theorem



- By 1-stability,
  - For all rows r of R,
    - $\langle r, Ax\rangle = |Ax|_1 \cdot Z / (d \log d)$, where Z is a Cauchy
- $RAx = (|Ax|_1 \cdot Z_1, \ldots, |Ax|_1 \cdot Z_{d \log d}) / (d \log d)$, where $Z_1, \ldots, Z_{d \log d}$ are i.i.d. Cauchy
- $|RAx|_1 = |Ax|_1 \sum_j |Z_j| / (d \log d)$
  - The $|Z_j|$ are half-Cauchy
- $\sum_j |Z_j| = \Omega(d \log d)$ with probability $1-\exp(-d \log d)$ by Chernoff
- But the $|Z_j|$ are heavy-tailed…

Proof of Sketching Theorem

- $\sum_j |Z_j|$ is heavy-tailed, so $|RAx|_1 = |Ax|_1 \sum_j |Z_j| / (d \log d)$ may be large
- Each $|Z_j|$ has c.d.f. asymptotic to $1-\Theta(1/z)$ for z in $[0, \infty)$
- There *exists* a well-conditioned basis of A
  - Suppose w.l.o.g. the basis vectors are $A_{*1}, \ldots, A_{*d}$
- $|RA_{*i}|_1 = |A_{*i}|_1 \cdot \sum_j |Z_{i,j}| / (d \log d)$
- Let $E_{i,j}$ be the event that $|Z_{i,j}| \leq d^3$
  - Define $Z'_{i,j} = |Z_{i,j}|$ if $|Z_{i,j}| \leq d^3$, and $Z'_{i,j} = d^3$ otherwise
  - $E\left[|Z_{i,j}| \mid E_{i,j}\right] = E\left[Z'_{i,j} \,\middle|\, E_{i,j}\right] = O(\log d)$
- Let E be the event that for all i,j, $E_{i,j}$ occurs
  - $\Pr[E] \geq 1 - \frac{\log d}{d}$
- What is $E\left[Z'_{i,j} \mid E\right]$?

Proof of Sketching Theorem

- What is $E\left[Z'_{i,j} \mid E\right]$?

- $$E\left[Z'_{i,j} \middle| E_{i,j}\right] = E\left[Z'_{i,j} \middle| E_{i,j}, E\right] \Pr[E \mid E_{i,j}] + E\left[Z'_{i,j} \middle| E_{i,j}, \neg E\right] \Pr[\neg E \mid E_{i,j}]$$
$$\geq E\left[Z'_{i,j} \middle| E_{i,j}, E\right] \Pr[E \mid E_{i,j}]$$
$$= E\left[Z'_{i,j} \middle| E\right] \cdot \left(\frac{\Pr\left[E_{i,j} \middle| E\right] \Pr[E]}{\Pr\left[E_{i,j}\right]}\right)$$
$$\geq E\left[Z'_{i,j} \middle| E\right] \cdot \left(1 - \frac{\log d}{d}\right)$$

- So, $E\left[Z'_{i,j} \middle| E\right] = O(\log d)$
- $|RA_{*i}|_1 = |A_{*i}|_1 \cdot \sum_j |Z_{i,j}| / (d \log d)$
- With constant probability, $\sum_i |RA_{*i}|_1 = O(\log d) \sum_i |A_{*i}|_1$

Proof of Sketching Theorem

- With constant probability, $\sum_i |RA_{*i}|_1 = O(\log d) \sum_i |A_{*i}|_1$
- Recall $A_{*1}, \ldots, A_{*d}$ is a well-conditioned basis, and we showed the existence of such a basis earlier
- We will use the Auerbach basis which always exists:
  - For all x, $|x|_\infty \leq |Ax|_1$
  - $\sum_i |A_{*i}|_1 = d$
- $\sum_i |RA_{*i}|_1 = O(d \log d)$
- For all x, $|RAx|_1 \leq \sum_i |RA_{*i}\, x_i| \leq |x|_\infty \sum_i |RA_{*i}|_1$

$$= |x|_\infty O(d \log d)$$

$$= O(d \log d)\, |Ax|_1$$

Where are we?

- Suffices to show for all x with $|x|_1 = 1$, that $|Ax|_1 \leq |RAx|_1 \leq d \log d \cdot |Ax|_1$
- We know
  - (1) there is a γ-net M, with $|M| \leq \left(\frac{d}{\gamma}\right)^{O(d)}$, of the set {Ax such that $|x|_1 = 1$}
  - (2) for any fixed x, $|RAx|_1 \geq |Ax|_1$ with probability $1 - \exp(-d \log d)$
  - (3) for all x, $|RAx|_1 = O(d \log d)|Ax|_1$

- Set $\gamma = 1/(d^3 \log d)$ so $|M| \leq d^{O(d)}$
  - By a union bound, for all y in M, $|Ry|_1 \geq |y|_1$

- Let x with $|x|_1 = 1$ be arbitrary. Let y in M satisfy $|Ax - y|_1 \leq \gamma = 1/(d^3 \log d)$

- $$\begin{aligned} |RAx|_1 &\geq |Ry|_1 - |R(Ax - y)|_1 \\ &\geq |y|_1 - O(d \log d)|Ax - y|_1 \\ &\geq |y|_1 - O(d \log d)\gamma \\ &\geq |y|_1 - O\left(\frac{1}{d^2}\right) \\ &\geq |y|_1/2 \quad \text{(why?)} \end{aligned}$$

Outline

- Quick recap of ℓ_1-regression, and how to speed it up
- Introduction to the Streaming Model and Estimating Norms

L_1 Regression Algorithm Recap

Compute poly(d)-approximation

Compute well-conditioned basis

Sample rows from the well-conditioned basis and the residual of the poly(d)-approximation

Solve l_1-regression on the sample, obtaining vector x, and output x

We saw how to solve the above problems by sketching by a matrix of i.i.d. Cauchy random variables

Sketching to solve l_1-regression [CW, MM]

- Most expensive operation is computing R*A where R is the matrix of i.i.d. Cauchy random variables
- All other operations are in the "smaller space"
- Can speed this up by choosing R as follows:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_1 & & & & \\ & C_2 & & & \\ & & C_3 & & \\ & & & \ldots & \\ & & & & C_n \end{bmatrix}$$

- For all x, $\left(\frac{1}{d^2 \log^2 d}\right) |Ax|_1 \leq |RAx|_1 \leq O(d \log d)\, |Ax|_1$
- Overall time for ℓ_1-regression is nnz(A) + poly(d/ϵ)

Further sketching improvements [WZ]

- Can show you need a fewer number of sampled rows in later steps if instead choose R as follows

- Instead of diagonal of Cauchy random variables, choose diagonal of reciprocals of exponential random variables

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/E_1 & & & & \\ & 1/E_2 & & & \\ & & 1/E_3 & & \\ & & & \ldots & \\ & & & & 1/E_n \end{bmatrix}$$

- For all x, $\left(\frac{1}{d^{.5}\mathrm{poly}(\log(nd))}\right)|Ax|_1 \leq |RAx|_1 \leq O(d \log d)\,|Ax|_1$

Fun Fact about Cauchy Random Variables

- Suppose you have i.i.d. copies $R_1, \ldots, R_n$ of a random variable with mean 0 and variance σ^2
- What is the distribution of $\frac{\sum_i R_i}{n}$?
- By Central Limit Theorem, this approaches a normal random variable N(0, σ^2/n)
- Intuitively, the variance is decreasing and the average is approaching its expectation
- Now suppose you have i.i.d. copies $R_1, \ldots, R_n$ of a standard Cauchy random variable
- What is the distribution of $\frac{\sum_i R_i}{n}$?
- It's still a standard Cauchy random variable!

Outline

- Introduction to the Streaming Model
- Estimating Norms in the Streaming Model

Turnstile Streaming Model

- Underlying n-dimensional vector x initialized to 0^n
- Long stream of updates $x_i \leftarrow x_i + \Delta_j$ for Δ_j in {-M, -M+1, …, M-1, M}
  - $M \leq \mathrm{poly}(n)$
- Throughout the stream, x is promised to be in $\{-M, -M+1, \ldots, M-1, M\}^n$
- Output an approximation to f(x) with high probability over our coin tosses
- Goal: use as little space (in bits) as possible
  - Massive data: stock transactions, weather data, genomes

Testing if x = 0^n

- How can we test, with probability at least 9/10, over our random coin tosses, if the underlying vector $\mathrm{x} = 0^{\mathrm{n}}$?

- Can we use O(log n) bits of space?

- We saw that for any fixed vector x, if S is a CountSketch matrix with $\mathrm{O}(\frac{1}{\epsilon^2})$ rows, then $|\mathrm{Sx}|_2^2 = (1 \pm \epsilon)|\mathrm{x}|_2^2$ with probability at least 9/10

- If we set $\epsilon = \frac{1}{2}$, we use O(log n) bits of space to store the O(1) entries of Sx

- We can store the hash function and sign function defining S using O(log n) bits

Testing if x = 0^n

- Is there a deterministic, i.e., zero-error, streaming algorithm to test if the underlying vector $\mathrm{x} = 0^{\mathrm{n}}$ with o(n log n) bits of space?

- Theorem: any deterministic algorithm requires $\Omega(\mathrm{n} \log \mathrm{n})$ bits of space

- Suppose the first half of the stream corresponds to updates to a vector a in $\{0, 1, 2, \ldots, \mathrm{poly}(\mathrm{n})\}^{\mathrm{n}}$

- Let S(a) be the state of the algorithm after reading the first half of the stream
  - If |S(a)| = o(n log n), there exist a$\neq$ a' for which S(a) = S(a')

- Suppose the second half of the stream corresponds to updates to a vector b in $\{0, -1, -2, \ldots, -\mathrm{poly}(\mathrm{n})\}^{\mathrm{n}}$

- The algorithm must output the same answer on a+b and a'+b, so it errs in one case

Example: Recovering a k-Sparse Vector

- Suppose we are promised that x has at most k non-zero entries at the end of the stream
- k is often small – maybe we see all coordinates of a vector a followed by all coordinates of a *similar* vector b, and a-b only has k non-zero entries
- Can we recover the indices and values of the k non-zero entries with high probability?
- Can we use k poly(log n) bits of space?
- Can we do it deterministically?

Example: Recovering a k-Sparse Vector

- Suppose A is an s x n matrix such that any 2k columns are linearly independent
- Maintain $A \cdot x$ in the stream
- Claim: from $A \cdot x$ you can recover the subset S of k non-zero entries and their values
- Proof: suppose there were vectors x and y each with at most k non-zero entries and $A \cdot x = A \cdot y$
- Then A(x-y) = 0. But x-y has at most 2k non-zero entries, and any 2k columns of A are linearly independent. So x-y = 0, i.e., x = y.
- Algorithm is deterministic given A. But do such matrices A exist with a small number s of rows?

Example: Recovering a k-Sparse Vector

- Vandermonde matrix A with s = 2k rows and n columns. $A_{i,j} = j^{i-1}$

$$\begin{bmatrix} 1 & 1 & 1 & \dots \\ 1 & 2 & 3 & \dots \\ 1 & 4 & 9 & \dots \\ 1 & 8 & 27 & \dots \end{bmatrix}$$

- Determinant of 2k x 2k submatrix of A with set of columns equal to $\{i_1, \dots, i_{2k}\}$ is: $\prod_j i_j \prod_{j<j'}(i_j - i_{j'}) \neq 0$, so any 2k columns of A are linearly independent

- But entries of A are exponentially increasing – how to store A and $A \cdot x$?

- Just store $A \cdot x$ mod p for a large enough prime p = poly(n)