Outline

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

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:

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

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

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

- Quick recap of ℓ_1-regression, and how to speed it up
- 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 $x_i \leftarrow x_i + \Delta_i$ for Δ_i in $\{-M, -M+1, \ldots, M-1, M\}$
  - $M \leq \mathrm{poly}(n)$
- 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 $\mathrm{A} \cdot \mathrm{x}$ in the stream
- Claim: from $\mathrm{A} \cdot \mathrm{x}$ 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 $\mathrm{A} \cdot \mathrm{x} = \mathrm{A} \cdot \mathrm{y}$
- 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. $A_{i,j} = j^{i-1}$

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

- Determinant of 2k x 2k submatrix of A with set of columns equal to $\{i_1, \ldots, i_{2k}\}$ 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)

Outline

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

Example Problem: Norms

- Suppose you want $|x|_p^p = \Sigma_{i=1}^n |x_i|^p$
- Want Z for which $(1-\varepsilon)\ |x|_p^p \leq Z \leq (1+\varepsilon)\ |x|_p^p$ with probability > 9/10
- p = 1 corresponds to total variation distance between distributions
- p = 2 useful for geometric and linear algebraic problems
- $p = \infty$ is the value of the maximum entry, useful for anomaly detection, etc.

Example Problem: Euclidean Norm

- Want Z for which $(1-\varepsilon)\ |x|_2^2 \leq Z \leq (1+\varepsilon)\ |x|_2^2$
- Sample a random CountSketch matrix S with $1/\epsilon^2$ rows
- Can store S efficiently using limited independence
- If $x_i \leftarrow x_i + \Delta_i$ in the stream, then $Sx \leftarrow Sx + \Delta_i S_{*,i}$
- At end of stream, output $|Sx|_2^2$
- With probability at least 9/10, $|Sx|_2^2 = (1 \pm \epsilon)|x|_2^2$
- Space complexity is $1/\epsilon^2$ words, each word is O(log n) bits

Example Problem: 1-Norm

- Want Z for which $(1-\varepsilon)\ |x|_1 \leq Z \leq (1+\varepsilon)\ |x|_1$
- Sample a random Cauchy matrix S?
- Can store S with $\frac{1}{\epsilon}$ words of space [Kane, Nelson, W]
- If $x_i \leftarrow x_i + \Delta_i$ in the stream, then $Sx \leftarrow Sx + \Delta_i S_{*,i}$
- Space complexity is $1/\epsilon^2$ words, each word is O(log n) bits
- At end of stream, output $|Sx|_1$?
- *Cauchy random variables have no concentration…*

1-Norm Estimator

- Probability density function f(x) of |C| for a Cauchy random variable C is $f(x) = \frac{2}{\pi(1+x^2)}$

- Cumulative distribution function F(z):

$$F(z) = \int_0^z f(x)dx = \frac{2}{\pi}\arctan(z)$$

- Since $\tan(\pi/4) = 1$, F(1) = ½, so median(|C|) = 1

- If you take $r = \frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}$ independent samples $X_1, \ldots, X_r$ from F, and $X = \mathrm{median}_i X_i$, then F(X) in [1/2-ϵ, 1/2+ ϵ] with probability 1-δ

- $F^{-1}(X) = \tan\left(\frac{X\pi}{2}\right) \in [1 - 4\epsilon, 1 + 4\epsilon]$

p-Norm Estimator

- Can achieve $1/\epsilon^2$ words of space for p-norm estimation for any 0 < p < 2
- Proof is similar to 1-norm estimation, and uses p-stable distributions, which exist only for 0 < p < 2
- No closed form expression for their probability density function but they are efficiently sampleable:
  - If $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $r \in [0,1]$ are uniformly random, then

$$\frac{\sin(p\,\theta)}{\cos^{\frac{1}{p}}\theta}\left(\frac{\cos(\theta(1-p))}{\ln\left(\frac{1}{r}\right)}\right)^{\frac{1-p}{p}}$$ is a sample from a p-stable distribution!

- Can discretize them and store a sketching matrix of samples from the p-stable distribution using limited independence

p-Norm Estimator for p > 2

- For p > 2, p-stable distributions do not exist!
- We will see later that $\Omega(n^{1-\frac{2}{p}})$ bits of space needed to approximate p-norms, p > 2, up to a constant factor with constant probability
- To achieve an $\widetilde{O}(n^{1-2/p})$ bits of space algorithm, we will use exponential random variables. We will focus on constant approximation parameter ϵ
- Our sketch will be $P \cdot D$:

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

Stability of Exponential Random Variables

- Exponential random variable E with parameter λ
  - (PDF) probability density function: f(x) = $\lambda e^{-\lambda x}$ if x ≥ 0, and 0 otherwise
  - (CDF) cumulative density function: F(x) = 1-$e^{-\lambda x}$ for x ≥ 0
  - $t \cdot$E for scalar $t \geq 0$ has CDF $F(x) = 1 - e^{-\frac{\lambda}{t}x}$

- Stability: consider independent exponential random variables $E_1, \ldots, E_n$ and scalars $|y_1|, \ldots, |y_n|$, let q = min($\frac{E_1}{|y_1|^p}, \ldots, \frac{E_n}{|y_n|^p}$)

- $\Pr[q > x] = \Pr\left[\forall i,\ \frac{E_i}{|y_i|^p} \geq x\right] = \prod_i e^{-x|y_i|^p} \quad = e^{-x|y|_p^p}$

- So q is an exponential random variable with $\lambda = |y|_p^p$, that is,

  $q \equiv \left(\frac{1}{|y|_p^p}\right) E$ for a standard exponential random variable E

Stability of Exponential Random Variables

- Recall our sketch P*D =

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/E_1^{1/p} & & & \\ & 1/E_2^{1/p} & & \\ & & \cdots & \\ & & & 1/E_n^{1/p} \end{bmatrix}$$

- What does $|Dy|_\infty$ look like for an arbitrary y?

- $|Dy|_\infty^p = \max_i \left(\frac{|y_i|^p}{E_i}\right) = \frac{1}{\min_i \frac{E_i}{|y_i|^p}} \equiv \frac{1}{E \cdot \frac{1}{|y|_p^p}} = \frac{|y|_p^p}{E}$

- $\Pr\left[E \in \left[\frac{1}{10}, 10\right]\right] = (1 - e^{-10}) - \left(1 - e^{-\frac{1}{10}}\right) = e^{-\frac{1}{10}} - e^{-10} > \frac{4}{5}$

Stability of Exponential Random Variables

- We know $|Dy|_\infty \in [\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\ ]$ with probability at least $\frac{4}{5}$

- So $|Dy|_\infty$ is a good estimate of $|y|_p$, but Dy is an n-dimensional vector!

- Recall our sketch P*D =

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

- What can we say about $|PDy|_\infty$ if P has s rows?

- Intuitively P is hashing coordinates of Dy into buckets and taking a signed sum of the entries. Expect everything to cancel out and $|PDy|_\infty \approx |Dy|_\infty$ ☺

Understanding $|PDy|_\infty$

- Let s be the number of rows of P, which we can think of as hash buckets

- P is a CountSketch matrix with hash functions h and σ
  - $h: [n] \rightarrow [s]$
  - $\sigma: [n] \rightarrow \{-1,1\}$
  - Let's assume h and σ are truly random (can be derandomized)

- We know $|Dy|_\infty \in [\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p]$ with probability at least 4/5

- To achieve $|PDy|_\infty \approx |Dy|_\infty$ with good probability, we want

  - (1) in each bucket i not containing the coordinate j for which $|(Dy)_j| = |Dy|_\infty$, we have $(PDy)_i \leq \frac{|y|_p}{100}$

  - (2) in the bucket i containing the coordinate j for which $|(Dy)_j| = |Dy|_\infty$, we have $||(PDy)_i| - |Dy|_\infty| \leq |y|_p/100$

Analyzing $|\mathrm{PDy}|_\infty$

- Let $\delta(\mathrm{E}) = 1$ if event E holds, and $\delta(\mathrm{E}) = 0$ otherwise
- What does the i-th bucket value $(\mathrm{PDy})_\mathrm{i}$ look like?
- $(\mathrm{PDy})_\mathrm{i} = \sum_\mathrm{j} \delta(\mathrm{h(j) = i})\, \sigma_\mathrm{j} (\mathrm{Dy})_\mathrm{j}$
- $\mathrm{E}[(\mathrm{PDy})_\mathrm{i}] = 0$
- What about the variance of $(\mathrm{PDy})_\mathrm{i}$?

Understanding $|PDy|_\infty$

- $E_P\left[(PDy)_i^2\right] = \sum_{j,j'} E[\delta(h(j) = i)\delta(h(j') = i)\sigma_j\sigma_{j'}](Dy)_j(Dy)_{j'} = \left(\frac{1}{s}\right)|Dy|_2^2$

- $E_D[|Dy|_2^2] = \sum_i y_i^2 \cdot E[D_{i,i}^2]$

- $E\left[D_{i,i}^2\right] = \int_{t\geq 0} t^{-2/p}e^{-t}\,dt$

$$= \int_{t\in[0,1]} t^{-2/p}e^{-t}dt + \int_{t>1} t^{-2/p}e^{-t}dt$$
$$\leq \int_{t\in[0,1]} t^{-2/p}dt + \int_{t>1} e^{-t}dt$$
$$= \left(\frac{1}{1-\frac{2}{p}}\right) \cdot t^{1-2/p}|_0^1 - e^{-t}|_1^\infty$$
$$= O(1)$$

- So, $E\left[(PDy)_i{}^2\right] = O\left(\frac{1}{s}\right)|y|_2^2 = O\left(\frac{1}{s}\right)(n^{1-\frac{2}{p}}|y|_p^2)$. Why?

Understanding $|\mathrm{PDy}|_\infty$

- $\mathrm{E}[(\mathrm{PDy})_i] = 0$ for each hash bucket i, and $\mathrm{E}\left[(\mathrm{PDy})_i{}^2\right] = O\left(\frac{1}{s}\right)(n^{1-\frac{2}{p}}|y|_p^2)$

- Bernstein's bound: Suppose $R_1, \ldots, R_n$ are independent, and for all j, $|R_j| \le K$, and $\mathrm{Var}\left[\sum_j R_j\right] = \sigma^2$. There are constants C, c, so that for all t > 0,

$$\Pr\left[\left|\sum_j R_j - E\left[\sum_j R_j\right]\right| > t\right] \le C\left(e^{-\frac{ct^2}{\sigma^2}} + e^{-\frac{ct}{K}}\right)$$

- Recall $(\mathrm{PDy})_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (\mathrm{Dy})_j$, and set $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (\mathrm{Dy})_j$

- Want $|\mathrm{PDy}|_\infty \approx |\mathrm{Dy}|_\infty$, where $|\mathrm{Dy}|_\infty \in [\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p]$ with probability > 4/5

- Set t = $\frac{|y|_p}{100}$ and s = $\Theta(n^{1-\frac{2}{p}} \log n)$, to get $\frac{1}{n^2}$ error probability in Bernstein's bound
- But what is K = $\max_j |R_j|$?

Understanding the Large Elements

- Recall $(PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$, and set $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$

- We will separately handle those R_j for which $|R_j| > \frac{\alpha |y|_p}{\log n}$, for a sufficiently small constant $\alpha > 0$. If $|R_j| > \frac{\alpha |y|_p}{\log n}$, then necessarily $|(Dy)_j| \geq \frac{\alpha |y|_p}{\log n}$
  - We call such a j large if $|(Dy)_j| \geq \frac{\alpha |y|_p}{\log n}$, otherwise j is small. How many indices j are large?

- Recall: $|(Dy)_j| \equiv |y_j|/E_j^{1/p}$

- $\Pr_D\left[\left|(Dy)_j\right| \geq \frac{\alpha |y|_p}{\log n}\right] = \Pr\left[\frac{|y_j|}{E_j^{\frac{1}{p}}} \geq \frac{\alpha |y|_p}{\log n}\right] = \Pr[\frac{\alpha^p |y_j|^p}{|y|^p{}_p}\left(\log^p n\right) \geq E_j]$

$$= 1 - e^{-\frac{\alpha^P |y_j|^p (\log^p n)}{|y|_p^p}} \leq \frac{\alpha^p |y_j|^p (\log^p n)}{|y|_p^p}$$, so the expected number of large j is $O(\log^p n)$

Understanding the Large Elements

- Recall $(\mathrm{PDy})_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (\mathrm{Dy})_j$, and set $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (\mathrm{Dy})_j$

- We have shown the expected number of large j is $O(\log^p n)$, so by a Markov bound we have $O(\log^p n)$ large j with constant probability and we condition on D satisfying this

- We also condition on $|\mathrm{Dy}|_\infty \in \left[\frac{|y|_p}{10^{\frac{1}{p}}}, 10^{\frac{1}{p}}|y|_p\right]$, which held with probability > 4/5

- All the large j get perfectly hashed into separate hash buckets by P
  - We are throwing $O(\log^p n)$ balls into $s \geq n^{1-2/p}$ bins

- We can apply Bernstein on the small indices j inside a hash bucket!

Understanding the Large Elements

- $E[(PDy)_i] = 0$ for each hash bucket i, and $E\left[(PDy)_i{}^2\right] = O\left(\frac{1}{s}\right)\left(n^{1-\frac{2}{p}}|y|_p^2\right)$
- Bernstein's bound: Suppose $R_1, \ldots, R_n$ are independent, and for all j, $|R_j| \le K$, and $\mathrm{Var}\left[\sum_j R_j\right] = \sigma^2$. There are constants C, c, so that for all t > 0,
  - $\Pr[|\sum_j R_j - E[\sum_j R_j]| > t] \le C\left(e^{-\frac{ct^2}{\sigma^2}} + e^{-\frac{ct}{K}}\right)$
- $(PDy)_i = \sum_j \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$, and $R_j = \delta(h(j) = i) \cdot \sigma_j \cdot (Dy)_j$
- Can assume K = $\max_j |R_j| \le \frac{\alpha |y|_p}{\log n}$, since there is at most one large j in any hash bucket $(PDy)_i$
- Set t = $\frac{|y|_p}{100}$, and s = $\Theta(n^{1-\frac{2}{p}} \log n)$ in Bernstein's bound, to get for a bucket $(PDy)_i$:

$$\Pr\left[\left|\sum_{\text{small } j} \delta(h(j) = i)\, \sigma_j (Dy)_j\right| > \frac{|y|_p}{100}\right] \le C\left(e^{-\Theta(\log n)} + e^{-c\frac{(\log n)}{100\alpha}}\right) \le \frac{1}{n^2}$$

- By a union bound over all the s buckets, the "signed sum" of small j in every bucket will be at most $\frac{|y|_p}{100}$

Wrapping Up

- For all i,
  - $|(PDy)_i| \leq \frac{|y|_p}{100}$ if no large indices in i-th bucket
  - $|(PDy)_i| = |\sigma_j (Dy)_j| \pm \frac{|y|_p}{100}$ if exactly one large index j in i-th bucket
  - No bucket contains more than 1 large index j

- We conditioned on $|Dy|_\infty \in \left[\frac{|y|_p}{10^{\frac{1}{p}}}, 10^{\frac{1}{p}}|y|_p\right]$

- What is $|PDy|_\infty$?
  - $|PDy|_\infty \leq 10^{\frac{1}{p}}|y|_p + \frac{|y|_p}{100}$ and $|PDy|_\infty \geq \frac{|y|_p}{10^{\frac{1}{p}}} - \frac{|y|_p}{100}$

- So just output $|PDy|_\infty$ as your estimate to $|y|_p$

- Total space is $s = O(n^{1-\frac{2}{p}} \log n)$ words, which is $O(n^{1-\frac{2}{p}} \log^2 n)$ bits