CS7880: Special Topics in Theoretical Computer Science Spring 2019

Lecture 1 — January 17, 2019

Prof. Huy Nguyen *Scribe: David Stalfa*

In the previous class, we discussed the following problem: given a vector $x \in \mathbb{R}^n$ with the following operations:

- init: $x_i \leftarrow 0$
- update: $x_i \leftarrow x_i + \delta$

output an estimate of $\|x\|_2$.

1 Generalized Problem

In this class, we generalize the the problem to output $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$.

Note that, when we cannot store the entire vector x, even computing $\|x\|_1$ is non-trivial. There are three cases to consider:

1. $0 < p < 2$
2. $p = 2$
3. $p > 2$

Case (2) was solved in the previous lecture. In this lecture, we focus on case (1).

Generalizing previous techniques. Recall that in the previous lecture, we used the following property of the ℓ_2 norm.

Observation 1. *if X_1, X_2 are independent random variables, and $X_1, X_2 \sim N(0, 1)$ and $a_1X_1 + a_2X_2 = \sqrt{a_1^2 + a_2^2}X$, then $X \sim N(0, 1)$.*

That is, given two normal, independent random variables X_1, X_2, their sum is equal to a normal random variable scaled by the ℓ_2 norm of their coefficients. We would like to find distributions with a similar property for $0 < p < 2$, and use similar methods to compute $\|x\|_p$ as were used previously to compute $\|x\|_2$.

2 p-stable Distributions

Definition 2. *A distribution D is* p-stable *if, given independent random variables $X_1, X_2 \sim D$ and $a_1, a_2 \in \mathbb{R}$, $(a_1^p + a_2^p)^{1/p}X = a_1X_1 + a_2X_2$ implies $X \sim D$.*

$N(0,1)$ is, as we saw, a 2-stable distribution.

Lemma 3. *For any $p \in (0,2]$, there is some p-stable distribution D_p.*

For $p = 1$, the Cauchy distribution with pdf $f(x) = \frac{1}{\pi(x^2+1)}$ is p-stable, and we can use this as a stand-in when thinking about p-stable distributions in general. For arbitrary $0 < p < 2$, the pdf for D_p will look something like $f(x) = \frac{1}{c(x^{(1+p)}+1)}$ for some constant c.

Law of Large Numbers (LLN). One question that arises in this context is: does the law of large numbers contradict Lemma 3?

LLN states that, if we draw many samples from a distribution D, the samples will be distributed according to the normal distribution. This seems to contradict Lemma 3 in that the lemma states that if we draw many samples from D_p, they samples are distributed according to D_p. We can resolve this by noting that LLN assumes finite variance of the starting distribution D, but p-stable distributions like Cauchy have infinite variance.

Sampling from a p-stable Distribution. To sample from D_p, first sample θ uniformly from $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and r uniformly from $[0,1]$. The sample from D_p is

$$\frac{\sin(p\theta)}{(\cos\theta)^{1/p}}\left(\frac{\cos((1-p)\theta)}{\log(1/r)}\right)^{\frac{1-p}{p}}$$

Generally, however, we use polynomial approximations.

3 Algorithm to compute $\|x\|_p$

First attempt: The algorithm stores the linear sketch $\langle r, x\rangle$, where $r_1, \ldots, r_n$ are drawn from a p-stable distribution.

Observation: by Lemma 3 the dot product $\langle r, x\rangle$ will follow the p-stable distribution with scaling $\|x\|_p$.

When estimating the ℓ_2 norm, we compared the square of the result with the variance, and this yielded the estimate. However, in this case, the variance is infinite, so the same technique will not work. Instead, we use the *median* of the distribution.

Definition 4. *For any distribution D with pdf f, μ is the median of D if*

$$\int_{-\infty}^{\mu} f(x)dx = \frac{1}{2}$$

Note that the median is well defined only for some distributions, e.g. it is necessary for the distribution to be continuous to have a median. The median is well defined for all p-stable distributions, for $p \in (0,2)$.

Final Algorithm:

- let $k = \Theta(\frac{1}{\varepsilon^2} \log \frac{1}{d})$
- M is a $k \times n$ matrix whose entries are iid $\sim D_p$
- $y \leftarrow Mx$
- return $\frac{\text{median}(|y_1|,\ldots,|y_k|)}{\text{median}(|D_p|)}$

Note that taking the absolute value is essential here, because all D_p's are symmetric about the line $x = 0$, so the median will always be 0 if we do not take the absolute value.

3.1 Accuracy of Algorithm

- Let μ be the median of $|D_p|$ (or a close approximation).
- Let F be the pdf of $|D_p|$

  e.g. for Cauchy distribution with pdf f, $F(t) = 2(f(t))$
- Let $\alpha = \min_{t \in [\mu(1-\varepsilon), \mu(1+\varepsilon)]} F(t)$

  We require that $\alpha > 0$, i.e. the distribution in the neighborhood of the median bounded away from 0. This tells us that the median is well-defined.

Claim 5. $\Pr[\text{median}(|y_1|, \ldots, |y_k|) < (1 - \varepsilon)\mu] < d$.

Proof. For simplicity, we assume that $\|x\|_p = 1$ (since this is just the scale).

For any given y_i, we can bound the probability that it is far from the median μ.

$$\begin{aligned}\Pr[|y_i| < (1 + \varepsilon)\mu] &= \frac{1}{2} - \int_{\mu(1+\varepsilon)}^{\mu} F(t)dt \\ &\leq \frac{1}{2} - \varepsilon\mu\alpha\end{aligned}$$

where we know that $\alpha > 0$.

We know that $\text{median}(|y_1|, \ldots, |y_k|) < (1+\varepsilon)\mu$ only if at least half of the $|y_i|$'s are less than $(1+\varepsilon)\mu$. So, by the Chernoff bound,

$$\begin{aligned}\Pr[\text{median}(|y_1|, \ldots, |y_k|) < (1 + \varepsilon)\mu] &\leq \exp(-\Omega(\varepsilon^2\alpha^2\mu^2 k)) \\ &\leq d \qquad \text{if } k = \Theta\left(\frac{1}{\varepsilon^2} \log \frac{1}{d}\right)\end{aligned}$$

Where the last line follows from the fact that, for any D_p, μ and α are constant. □

Showing $\Pr[\text{median}(|y_1|, \ldots, |y_k|) > (1 - \varepsilon)\mu] < d$ is analogous.

3.2 Space Complexity of Algorithm

Here we have the same problem as in the previous lecture: we would like to use n random numbers without storing them. We could, as before, use bounded independence, but this analysis is non-trivial. Instead, we will use a different technique: pseudo-random generators. While this technique does not yield optimal space bounds ($\log^2 n$ instead of $\log n$) it is more use friendly and easily applicable. Understanding pseudo-random generators will first require an understanding of read once branching programs.

3.2.1 Read Once Branching Programs (ROBP)

A computer program with bounded memory: s bits. At every step, the program recieves s bits of input, and makes a transition (where the input includes both problem input and any randomly generated bits). After R steps, the final state of the memory is the output. The program operates under the restriction that each input can be read exactly once; specifically, the random bits in earlier input cannot be recovered later in the program (though new random bits can always be generated).

3.2.2 Pseudo-Random Generators (PRG's)

A PRG h satisfies the following conditions.

- Let U_n be a uniformly random string in $\{0,1\}^n$
- $\exists h : \{0,1\}^{s \log R} \to \{0,1\}^{sR}$

  $\{0,1\}^{s \log R}$ is our *seed*, and $\{0,1\}^{sR}$ is the *output* of the PRG.
- $\Pr[f(U_n) = 1] - \Pr[f(h(U_{s \log R})) = 1] \leq 2^{-O(s)}$

  where f is any ROBP.

  I.e. we require that no ROBP can distinguish the output from an actual random string.

3.2.3 Nisan's PRG (from Nisan '90 [1])

If we want to generate n random numbers using a naive approach, we would need to store $n \log n$ random bits. Nisan's PRG will allow us to generate n random numbers while storing only $\log n$ bits for s and $\log n$ bits for R. So the total space for the random generator is $\log^2 n$. (This can be reduced to $\log n$ using bounded independence, but this is difficult.) This allows us to reduce the total space for the algorithm to

$$(\log^2 n)\frac{1}{\varepsilon^2}\log\frac{1}{\delta}$$

The generator uses $\log n$ pairwise independent functions

$$h_1, h_2, \ldots, h_{\log n} : [2^s] \to [2^s]$$

and generates a random number using a tree structure as shown in Figure 1. The bottom level of this tree will have $s \log n$ bits, and this is our random number.

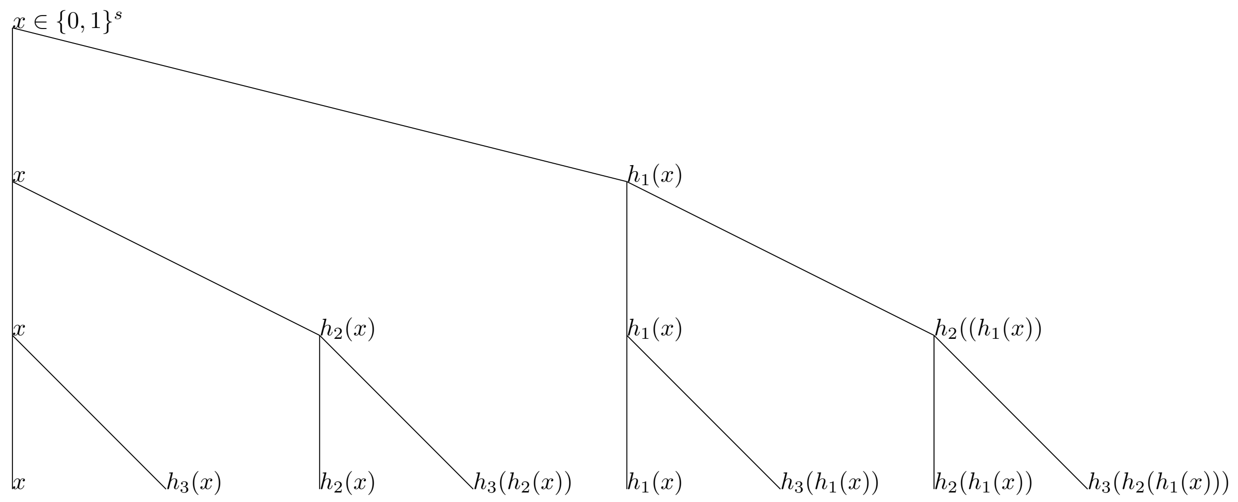

Figure 1

References

[1] Noam Nisan. Psuedorandom generators for space-bounded computation. In *STOC*, 1990.