CS 229r: Algorithms for Big Data Fall 2015

Lecture 4 — September 15, 2015

Prof. Jelani Nelson *Scribe: Hyunghoon Cho*

1 Recap

Recall the following definition of *p-stable distributions.*

Definition 1. *D_p is a p-stable distribution on $\mathbf{R}$ if, for any n independent samples $z_1, \ldots, z_n$ from D_p, the following holds: $\forall x \in \mathbf{R}^n, \sum_{i=1}^n x_i z_i \sim ||x||_p D_p$.*

In other words, any linear combination of a set of samples from a p-stable distribution must itself be a sample from the same distribution, scaled by the p-norm of the weight vector.

Theorem 1. *A p-stable distribution exists iff $0 < p \leq 2$.*

We know that the standard Gaussian distribution is 2-stable and the standard Cauchy distribution is 1-stable. Although in most cases a simple closed-form expression for p-stable distributions is not known, there is an efficient way to generate samples for any $p \in (0, 2]$. If we let $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $r \in [0, 1]$ be uniformly random samples, then

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

is a sample from a p-stable distribution [CMS76].

2 Indyk's Algorithm

2.1 Overview

Let $\Pi = \{\pi_{ij}\}$ be an $m \times n$ matrix where every element π_{ij} is sampled from a p-stable distribution, D_p. Given $x \in \mathbf{R}^n$, Indyk's algorithm [Indyk06] estimates the p-norm of x as

$$||x||_p \approx \text{median}_{i=1,\ldots,m} |y_i|,$$

where $y = \Pi x$. Recall from last lecture that in a turnstile streaming model, each element in the stream reflects an update to an entry in x. While a naive algorithm would maintain x in memory and calculate $||x||_p$ at the end, thus requiring $\Theta(n)$ space, Indyk's algorithm stores y and Π. Combined with a space-efficient way to produce Π (described later in the lecture) we achieve better space complexity.

2.2 Analysis

For simplicity of analysis, we will assume Π is generated with D_p such that if $Z \sim D_p$ then median$(|Z|) = 1$. In other words, we assume the probability mass of D_p assigned to interval $[-1, 1]$ is $1/2$. In addition, let $I_{[a,b]}(x)$ be an indicator function defined as

$$I_{[a,b]}(x) = \begin{cases} 1 & x \in [a, b], \\ 0 & \text{otherwise.} \end{cases}$$

Let Z_i be the i-th row of Π. We have

$$y_i = \sum_{j=1}^{n} Z_{ij} x_j \sim ||x||_p D_p, \tag{1}$$

which follows from the definition of p-stable distributions and noting that Z_{ij}'s are sampled from D_p. This implies

$$\mathbf{E}\left[I_{[-1,1]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2}, \tag{2}$$

since $y_i/||x||_p \sim D_p$.

Moreover, it can be shown that

$$\mathbf{E}\left[I_{[-1-\varepsilon,1+\varepsilon]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2} + \Theta(\varepsilon), \tag{3}$$

$$\mathbf{E}\left[I_{[-1+\varepsilon,1-\varepsilon]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2} - \Theta(\varepsilon). \tag{4}$$

Next, consider the following quantities:

$$C_1 = \frac{1}{m} \sum_{i}^{m} I_{[-1-\varepsilon,1+\varepsilon]}\left(\frac{y_i}{||x||_p} \right), \tag{5}$$

$$C_2 = \frac{1}{m} \sum_{i}^{m} I_{[-1+\varepsilon,1-\varepsilon]}\left(\frac{y_i}{||x||_p} \right). \tag{6}$$

C_1 represents the fraction of y_i's that satisfy $|y_i| \leq (1+\varepsilon)||x||_p$, and similarly, C_2 represents the fraction of y_i's that satisfy $|y_i| \leq (1-\varepsilon)||x||_p$. By linearity of expectation, we have $\mathbf{E}[C_1] = 1/2 + \Theta(\varepsilon)$ and $\mathbf{E}[C_2] = 1/2 - \Theta(\varepsilon)$. Therefore, in expectation, the median of $|y_i|$ lies in

$$[(1-\varepsilon)||x||_p, (1+\varepsilon)||x||_p]$$

as desired.

Now we analyze the variance of C_1 and C_2. We have

$$\mathbf{Var}\left(C_1\right) = \frac{1}{m^2} \times m \times (\text{variance of the indicator variable}). \tag{7}$$

Since variance of any indicator variable is at most 1, $\mathbf{Var}(C_1) \leq \frac{1}{m}$. Similarly, $\mathbf{Var}(C_2) \leq \frac{1}{m}$. With an appropriate choice of m now we can ensure that the median of $|y_i|$ is in the desired ε-range of $||x||_p$ with high probability.

2.3 Derandomizing Π

We have shown that Indyk's algorithm work, but independently generating and storing all mn elements of Π is expensive. Can we get by with a smaller degree of randomness? To invoke the definition of p-stable distributions for Equation 1, we need the entries in each row to be independent from one another. And the rows need to be pairwise independent for our calculation of variance to hold. As a side note, generating pairwise independent Gaussian or Cauchy random variables can be achieved by discretizing the space and treating them as discrete random variables. One can make a claim that, with fine enough discretization, the algorithm still succeeds with high probability.

Now suppose $w_i = \sum_{j=1}^{n} Q_{ij}x_j$ where Q_{ij}'s are k-wise independent p-stable distribution samples. What we want is an argument of the form

$$\mathbf{E}\left[I_{[a,b]}\left(\frac{w_i}{||x||_p}\right)\right] \approx_\varepsilon \mathbf{E}\left[I_{[a,b]}\left(\frac{y_i}{||x||_p}\right)\right]. \tag{8}$$

If we can make such claim, then we can use k-wise independent samples in each row in lieu of fully independent samples to invoke the same arguments in the analysis above. This has been shown for $k = \Omega(1/\varepsilon^p)$ [KNW10], but we are not going to cover this in class. With this technique, we can specify Π using only $O(k \lg n)$ bits; across rows, we only need to use 2-wise independent hash function that maps a row index to a $O(k \lg n)$ bit seed for the k-wise independent hash function. Indyk's approach to derandomizing Π, while the results are not as strong, employs a useful technique, so we will cover that here instead.

2.3.1 Branching Programs

A *branching program* can be described with a grid of nodes representing different states of the program—we can think of it as a DFA. The program starts at an initial node which is not part of the grid. At each step, the program reads S bits of input, reflecting the fact that space is bounded by S, and makes a decision about which node in the subsequent column of the grid to jump to. After R steps (number of columns in the grid), the final node visited by the program represents the outcome. The entire input, which can be represented as a length-RS bit string, induces a distribution over the final states. Our goal is to generate the input string using fewer ($\ll RS$) random bits such that the original distribution over final states is well preserved. The following theorem addresses this goal.

Theorem 2. *([Nisan92]) There exists $h : \{0,1\}^t \to \{0,1\}^{RS}$ for $t = O(S \lg R)$ such that*

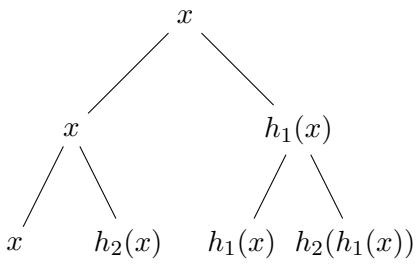
$$\left|P_{x\sim U(\{0,1\}^{RS})}\{f(B(x)) = 1\} - P_{y\sim U(\{0,1\}^t)}\{f(B(h(y))) = 1\}\right| \leq \frac{1}{2^S}. \tag{9}$$

for any branching program B and any function $f : \{0,1\}^S \to \{0,1\}$.

In other words, such function h can simulate the input to the branching program with only t random bits such that it is almost impossible to discriminate the outcome of the simulated program from that of the original program.

2.3.2 Nisan's Pseudorandom Generator (PRG)

What does such a function look like? We start by taking a random sample x from $\{0,1\}^S$. Then we place x at the root and repeat the following procedure to create a complete binary tree. At each node, create two children and copy the string over to the left child. For the right child, use a random 2-wise independent hash function $h_j : [2^S] \to [2^S]$ chosen for the corresponding level of the tree and record the result of the hash. Once we reach R levels, output the concatenation of all leaves, which is a length-RS bit string. To illustrate, the first few levels of our tree would look like:



Since each hash function requires S random bits and there are $\lg R$ levels in the tree, this function uses $O(S \lg R)$ bits total. We will not discuss why this function satisfies the condition in Theorem 2.

2.3.3 Back to Indyk's Algorithm

Now we will use Nisan's PRG to derandomize Π in Indyk's algorithm. Consider the following program:

1. Initialize $c_1 \leftarrow 0$, $c_2 \leftarrow 0$
2. For $i = 1, \ldots, m$:
   (a) Initialize $y \leftarrow 0$
   (b) For $j = 1, \ldots, n$:
      i. Update $y \leftarrow y + \pi_{ij} x_j$
   (c) If $y \leq (1+\varepsilon)||x||_p$, then increment c_1
   (d) If $y \leq (1-\varepsilon)||x||_p$, then increment c_2

Note that this program only uses $O(\lg n)$ bits and is a branching program that mimics the proof of correctness for Indyk's algorithm. More specifically, Indyk's algorithm succeeded iff at the end of this program $c_1 > \frac{m}{2}$ and $c_2 < \frac{m}{2}$. The only source of randomness in this program are the π_{ij}'s. We will use Nisan's PRG to produce these random numbers. We invoke Theorem 2 with the above program as B and an indicator function checking whether the algorithm succeeded or not as f. Note that the space bound is $S = O(\lg n)$ and the number of steps taken by the program is $R = O(mn)$, or $O(n^2)$ since $m \leq n$. This means we can "fool" the proof of correctness of Indyk's algorithm only using $O(\lg^2 n)$ random bits to generate Π.

3 The $p > 2$ Case

Indyk's algorithm uses p-stable distributions which only exist for $p \in (0, 2]$. What do we do when $p > 2$?

Theorem 3. *$n^{1-2/p} poly(\frac{\lg n}{\varepsilon})$ space is necessary and sufficient.*

A nearly optimal lower bound was given by [BarYossef04] and first achieved by [IW05]. In this lecture we will discuss the algorithm of [Andoni12], which is based on [AKO11] and [JST11]. We will focus on $\varepsilon = \Theta(1)$. Refining this result may be included in the next homework.

In this algorithm, we let $\Pi = PD$. P is a $m \times n$ matrix, where each column has a single non-zero element that is either 1 or -1. D is a $n \times n$ diagonal matrix with $d_{ii} = u_i^{-1/p}$, where $u_i \sim \text{Exp}(1)$. In other words,

$$P\{u_i > t\} = \begin{cases} 1 & t \leq 0, \\ e^{-t} & t > 0. \end{cases}$$

Similar to the $0 < p \leq 2$ case, we will keep $y = \Pi x$, but here we estimate $||x||_p$ with

$$||x||_p \approx ||y||_\infty = \max_i |y_i|. \tag{10}$$

Theorem 4. *$P\left\{\frac{1}{4}||x||_p \leq ||y||_\infty \leq 4||x||_p\right\} \geq \frac{11}{20}$ for $m = \Theta(n^{1-2/p} \lg n)$.*

Let $z = Dx$, which means $y = Pz$. To prove Theorem 4, we will first show that $||z||_\infty$ provides a good estimate and then show that applying P to z preserves this. The latter step is deferred to next class due to time constraints.

Claim 1. *$P\left\{\frac{1}{2}||x||_p \leq ||z||_\infty \leq 2||x||_p\right\} \geq \frac{3}{4}$*

Proof. Let $q = \min\left\{\frac{u_1}{|x_1|^p}, \ldots, \frac{u_n}{|x_n|^p}\right\}$. We have

$$\begin{aligned} P\{q > t\} &= P\left\{\forall i, u_i > t|x_i|^p\right\} && (11) \\ &= \prod_{i=1}^{n} e^{-t|x_i|^p} && (12) \\ &= e^{-t||x||_p^p}, && (13) \end{aligned}$$

which implies $q \sim \frac{Exp(1)}{||x||_p^p}$. Thus,

$$\begin{aligned} P\left\{\frac{1}{2}||x||_p \leq ||z||_\infty \leq 2||x||_p\right\} &= P\left\{\frac{1}{2^p}||x||_p^{-p} \leq q \leq 2^p||x||_p^{-p}\right\} && (14) \\ &= e^{-\frac{1}{2^p}} - e^{-2^p} && (15) \\ &\geq \frac{3}{4}, && (16) \end{aligned}$$

for $p > 2$.

□

References

[CMS76] J. M. Chambers, C. L. Mallows, and B. W. Stuck. A method for simulating stable random variables. *Journal of the american statistical association*, 71.354 (1976): 340-344.

[KNW10] Daniel Kane, Jelani Nelson, David Woodruff. On the exact space complexity of sketching and streaming small norms. *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, 2010.

[BarYossef04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68 (2004): 702-732.

[Andoni12] Alexandr Andoni. High frequency moments via max-stability. Manuscript, 2012. `http://web.mit.edu/andoni/www/papers/fkStable.pdf`

[AKO11] Alexandr Andoni, Robert Krauthgamer, Krzysztof Onak. Streaming Algorithms via Precision Sampling. *FOCS*, pgs. 363–372, 2011.

[JST11] Hossein Jowhari, Mert Saglam, Gábor Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. *PODS*, pgs. 49–58, 2011.

[Indyk06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53(3): 307–323, 2006.

[IW05] Piotr Indyk, David P. Woodruff. Optimal approximations of the frequency moments of data streams. *STOC*, pgs. 202–208, 2005.

[Nisan92] Noam Nisan. Pseudorandom Generators for Space-Bounded Computation. *Combinatorica*, 12(4):449-461, 1992.