Sketching Algorithms for Big Data Fall 2017

Lecture 01 — August 31, 2017

Prof. Jelani Nelson *Scribe: Vinh-Kha Le*

1 Overview

In this lecture, we overviewed the six main topics covered in the course, reviewed important inequalities from probability theory, and examined Robert Morris's method of counting large numbers in a small register [Mor78].

2 Topics Covered in the Course

This course is divided into six topics: sketching, streaming, dimensionality reduction, large-scale matrix computations, compressed sensing, and the sparse Fourier transform. Some of these topics overlap, and some of them require similar techniques to solve. The overarching theme of this course is to handle situations where a data set is so large or a computation is so complex that a naive approach would require an impractical amount of memory or time.

Topic 1: Sketching

Sketching is the compression $C(x)$ of some data set x that allows us to query $f(x)$. There are some things we might want when we are designing such a $C(x)$. Perhaps, we want f to take on two arguments instead of one, as is the case when we want to compute $f(x, y)$ from $C(x)$ and $C(y)$. Often, we want $C(x)$ to be composable. In other words, if $x = x_1x_2x_3 \dots x_n$, we want to be able to compute $C(xx_{n+1})$ using just $C(x)$ and x_{n+1}.

Topic 2: Streaming

If a data set is really large, it may not be possible to store or process all the data at the same time, as we do with the word RAM model. A stream is a sequence of data elements that comes in bit by bit, like items on a conveyor belt. Streaming is the act of processing these data elements on the fly as they arrive. The goal of streaming is to answers queries within the constraints of sublinear memory.

Topic 3: Dimensionality Reduction

There are many instances in the real world where we encounter data sets with high dimensionality. The example brought up in class was the problem of spam filtering. A simple approach to spam detection is the bag-of-words model, where each email can be represented as a high-dimensional vector whose indices come from a dictionary of words and the value at each index is the number of occurrences of the corresponding word. In situations like these where we have a high-dimensional computational geometry problem, we may want to reduce the number of dimensions in pre-processing while preserving the approximate geometric structure.

Topic 4: Large-Scale Matrix Computations

Many of the problems that deal with big data end up involving some large-scale matrix computation. Examples of large-scale matrix computations include regression and principal component analysis (PCA). For a least squares regression, we guess that $y = f(x)$ for some vector x of explanatory variables. We want to learn f, so we assume that $f(x) = \langle \beta, x \rangle$ for some β. Here, $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. We collect data points (x_i, y_i) and assume that $y_i = f(x_i) + \epsilon_i$ for some small noise or error term ϵ_i, and from these data points, it is possible to recover a vector of coefficients $\beta = \beta^{\mathrm{LS}}$ that minimizes

$$\sum_{i=1}^{n} (y_i - \langle \beta, x_i \rangle)^2 = \|X\beta - y\|_2^2$$

by computing $\beta^{\mathrm{LS}} = (X^\top X)^{-1} X^\top Y$. Here, $\|\cdot\|_2$ denotes the L^2 norm.

Topic 5: Compressed Sensing

As we have seen, many of the problems faced in the real world involve linear signals. Sometimes, when we change the basis we use to describe this linear signal, it becomes sparse. This allows us to store far fewer samples than otherwise necessary. Compressed sensing also involves the process of finding a basis in which a linear signal is sparse, taking a small number of linear measurements, and later approximately reconstructing the original signal from the measurements.

Example. Images can be compressed using a Haar wavelet transform. Here, we describe an example of an implementation of the Haar wavelet transform. A grayscale image is an $n \times n$ matrix of integers ranging from 0 to 255, black to white. Divide the image into 2×2 blocks and label the pixels a, b, c, and d. We create four $n/2 \times n/2$ sub-images with the values $w = (a + b + c + d)/4$, $x = (a + d - b - c)/4$, $y = (a + b - c - d)/4$, and $z = (a + c - b - d)/4$. The values of w, x, y, and z are sufficient to represent each quadruple a, b, c, and d, and they are normalized so that each can be represented using 8 bits. Because a, b, c, and d are adjacent each other in the original image, we expect them to have similar values most of the time. When they do not have the same value, it will be on the edge of an object. (Edges are sparse compared to non-edges.) Thus, we expect x, y, and z to be close to zero most of the time. We can throw away the sub-images made of the pixels x, y, and z. This leaves us the $n/2 \times n/2$ sub-image made of the pixels w. If we want an even smaller image, we can apply recursion on the sub-image. JPEG uses a similar procedure that cuts a 256×256 image into 8×8 subimages.

Topic 6: Sparse Fourier Transform

For some fixed integer n, let $F_n = [F_{jk}]$ be the matrix furnished by the terms

$$F_{jk} = e^{-2\pi i jk/n}$$

and let $x = (x_1, x_2, \ldots, x_n)$ be a sequence of complex numbers. The discrete Fourier transform (DFT) sends x to $F_n x$. In 1942, Danielson and Lanczos published an algorithm that computed the DFT in $O(n \log n)$ floating-point operations (FLOPS) [DL42]. In 1965, Cooley and Tukey published a more general version of the fast Fourier transform (FFT) [CT65]. Cooley and Tukey are often credited with the discovery of the modern generic FFT algorithm, but in an unpublished manuscript from 1805, Gauss had already found a similar algorithm and used it to interpolate the orbits of Pallas and Juno. For more information, please see refer to the *Theoria Interpolationis Methodo Nova Tractata.* The sparse Fourier transform (SFT) is an algorithm that computes the Fourier transform in time $O(k \log n)$ if the output of the DFT is exactly k-sparse. If the output of the DFT is approximately k-sparse, SFT can approximate the Fourier transform in time that is approximately but a little worse than $O(k \log n)$.

3 Approximate Counting

We will now discuss our first detailed example of a sketching algorithm. In the following, we discuss a problem first studied in [Mor78].

Problem. Design an algorithm that monitors a sequence of events and upon request can output (an estimate of) the number of events thus far. More formally, create a data structure that maintains a single integer n and supports the following operations.

- `init();` sets $n \leftarrow 0$
- `update();` increments $n \leftarrow n + 1$
- `query();` outputs n or an estimate of n

Before any operations are performed, we assume that $n = 0$. A trivial algorithm stores n as a sequence of $\lceil \log n \rceil = O(\log n)$ bits (a counter). If we want `query();` to return the exact value of n, this is the best we can do. Suppose for some such algorithm, we use $f(n)$ bits to store the integer n. There are $2^{f(n)}$ configurations for these bits. In order for the algorithm to be able to store the exact value of all integers up to n, the number of configurations must be greater than or equal to the number n. Hence,

$$2^{f(n)} \geq n \Rightarrow f(n) \geq \log n.$$

The goal is to use much less space than $O(\log n)$, and so we must instead answer `query();` with some estimate $\tilde{n}$ of n. We would like this $\tilde{n}$ to satisfy

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \delta, \tag{1}$$

for some $0 < \varepsilon, \delta < 1$ that are given to the algorithm up front.

Morris's algorithm provides such an estimator for some ε, δ that we will analyze shortly. The algorithm works as follows:

- `init();` sets $X \leftarrow 0$
- `update();` increments X with probability 2^{-X}
- `query();` outputs $\tilde{n} = 2^X - 1$

Intuitively, the variable X is attempting to store a value that is approximately $\log_2 n$. Before giving a rigorous analysis in Section 5, we first give a probability review.

4 Probability Review

We are mainly discussing discrete random variables. For this section we consider random variables as taking values in some set $S \subset \mathbb{R}$. Recall the expectation of X is defined to be

$$\mathbb{E}X = \sum_{j \in S} j \cdot \mathbb{P}(X = j).$$

We now state a few basic lemmas and facts without proof.

Lemma 1 (Linearity of expectation)**.** *For all reals a and b and random variables X and Y,*

$$\mathbb{E}(aX + bY) = a \cdot \mathbb{E}X + b \cdot \mathbb{E}Y. \tag{2}$$

Lemma 2 (Markov)**.** *If X is a nonnegative random variable, then for all $\lambda > 0$,*

$$\mathbb{P}(X > \lambda) < \frac{\mathbb{E}X}{\lambda}$$

Lemma 3 (Chebyshev)**.** *Under the same conditions as Markov,*

$$\mathbb{P}(|X - \mathbb{E}X| > \lambda) < \frac{\mathbb{E}(X - \mathbb{E}X)^2}{\lambda^2} = \frac{\mathrm{Var}[X]}{\lambda^2} \tag{3}$$

Proof. Observe that
$$\mathbb{P}(|X - \mathbb{E}X| > \lambda) = \mathbb{P}((X - \mathbb{E}X)^2 > \lambda^2).$$
The claim follows by Markov's inequality. □

Note that
$$\mathbb{P}(|X - \mathbb{E}X| > \lambda) = \mathbb{P}(|X - \mathbb{E}X|^p > \lambda^p)$$
for all $p \geq 1$. If we apply Markov's inequality to this statement, we get a more general version of Chebyshev's inequality:

$$\mathbb{P}(|X - \mathbb{E}X| > \lambda) < \frac{\mathbb{E}|X - \mathbb{E}X|^p}{\lambda^p}. \tag{4}$$

This is very similar to the statement of Chebyshev's inequality encountered in measure theory. By a calculation and picking p optimally one can also obtain the following "Chernoff bound." We show a different proof below, of a weaker statement.

Lemma 4 (Chernoff)**.** *Suppose $X_1, X_2, \ldots, X_n$ are independent random variables with $X_i \in [0, 1]$. Let $X = \sum_{i=1}^n X_i$ with $\mu = \mathbb{E}X$. If $0 < \epsilon < 1$, then*

$$\mathbb{P}(|X - \mathbb{E}X| > \varepsilon\mu) \leq 2 \cdot e^{-\varepsilon^2 \mu / 3}. \tag{5}$$

Proof. We will prove a weaker statement: we assume that the X_i are independent Bernoulli. Let $X_i = 1$ with probability p_i and $X_i = 0$ otherwise. Note then $\mu = \sum_{i=1}^n p_i$. Furthermore, we also do not attempt to achieve the constant 1/3 in the exponent, but are happy to settle for any fixed constant. For the upper tail, we have for any $t > 0$ that

$$\begin{aligned}
\mathbb{P}(X > (1+\varepsilon)\mu) &= \mathbb{P}\left(e^{tX} > e^{t(1+\varepsilon)\mu}\right) \\
&\le e^{-t(1+\varepsilon)\mu} \cdot \mathbb{E}e^{tX} \text{ (Markov)} \\
&= e^{-t(1+\varepsilon)\mu} \cdot \mathbb{E}e^{\sum_i tX_i} \\
&= e^{-t(1+\varepsilon)\mu} \cdot \mathbb{E}\prod_i e^{tX_i} \\
&= e^{-t(1+\varepsilon)\mu} \cdot \prod_i \mathbb{E}e^{tX_i} \text{ (independence)} \\
&= e^{-t(1+\varepsilon)\mu} \cdot \prod_i (1 - p_i + p_i e^t) \\
&= e^{-t(1+\varepsilon)\mu} \cdot \prod_i (1 + p_i(e^t - 1)) \\
&\le e^{-t(1+\varepsilon)\mu} \cdot \prod_i e^{p_i(e^t-1)} \text{ (since } 1 + x \le e^x) \\
&= e^{-t(1+\varepsilon)\mu + (e^t-1)\mu} \\
&= \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mu} \text{ (set } t = \ln(1+\varepsilon)) \qquad (6)
\end{aligned}$$

There are two regimes of interest for Eqn. (6). When $\varepsilon < 2$, by Taylor's theorem we have $\ln(1+\varepsilon) = \varepsilon - \varepsilon^2/2 + O(\varepsilon^3)$. Then replacing $(1+\varepsilon)^{1+\varepsilon}$ by $e^{(1+\varepsilon)\ln(1+\varepsilon)}$ and applying Taylor's theorem, this leads to (6) $\simeq e^{-\Theta(\varepsilon^2\mu)}$ in that regime. When $\varepsilon > 2$ we have $\ln(1+\varepsilon) = \Theta(\ln(\varepsilon))$, in which case the tail bounds becomes $\varepsilon^{-\Theta(\varepsilon\mu)}$. The lower tail analysis for $\mathbb{P}(X < (1-\varepsilon)\mu)$ is similar, noting that $X < (1-\varepsilon)\mu$ iff $e^{-tX} > e^{-t(1-\varepsilon)\mu}$. We then apply Markov then analyze $\mathbb{E}e^{-tX}$ and eventually set $t = -\ln(1-\varepsilon)$. The right hand side then becomes $(e^{-\varepsilon}/(1-\varepsilon)^{1-\varepsilon})^{\mu}$. □

5 Analysis of Morris's Algorithm

Let X_n denote X in Morris's algorithm after n updates.

Claim 5. *For Morris's algorithm, $\mathbb{E}2^{X_n} = n + 1$.*

Proof. We will prove by induction. Consider the base case where $n = 0$. We have initialized $X \leftarrow 0$ and have yet to increment it. Thus, $X_n = 0$, and $\mathbb{E}2^{X_n} = n + 1$. Now suppose that $\mathbb{E}2^{X_n} = n + 1$ for some fixed n.

We have

$$\begin{aligned}
\mathbb{E}2^{X_{n+1}} &= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) \cdot \mathbb{E}(2^{X_{n+1}} \mid X_n = j) \\
&= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) \cdot \left(2^j \left(1 - \frac{1}{2^j}\right) + \frac{1}{2^j} \cdot 2^{j+1}\right) \\
&= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) 2^j + \sum_j \mathbb{P}(X_n = j) \\
&= \mathbb{E}2^{X_n} + 1 \\
&= (n+1) + 1.
\end{aligned} \tag{7}$$

This completes the inductive step. □

It is now clear why we output our estimate of n as $\tilde{n} = 2^X - 1$: it is an unbiased estimator of n. In order to show (1) however, we will also control on the variance of our estimator. This is because, by Chebyshev's inequality,

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \mathbb{E}(\tilde{n} - n)^2 = \frac{1}{\varepsilon^2 n^2} \cdot \mathbb{E}(2^X - 1 - n)^2. \tag{8}$$

When we expand the above square, we find that we need to control $\mathbb{E}2^{2X_n}$. The proof of the following claim is by induction, similar to that of Claim 5.

Claim 6. *For Morris's algorithm, we have*

$$\mathbb{E}2^{2X_n} = \frac{3}{2}n^2 + \frac{3}{2}n + 1. \tag{9}$$

Proof. We again prove this by induction. It is clearly true for $n = 0$. Then

$$\begin{aligned}
\mathbb{E}2^{2X_{n+1}} &= \sum_j \mathbb{P}(2^{X_n} = j) \cdot \mathbb{E}(2^{2X_{n+1}} \mid 2^{X_n} = j) \\
&= \sum_j \mathbb{P}(2^{X_n} = j) \cdot \left(\frac{1}{j} \cdot 4j^2 + \left(1 - \frac{1}{j}\right) \cdot j^2\right) \\
&= \sum_j \mathbb{P}(2^{X_n} = j) \cdot (j^2 + 3j) \\
&= \mathbb{E}2^{2X_n} + 3 \cdot \mathbb{E}2^{X_n} \\
&= \left(\frac{3}{2}n^2 + \frac{3}{2}n + 1\right) + (3n + 3) \\
&= \frac{3}{2}(n+1)^2 + \frac{3}{2}(n+1) + 1
\end{aligned}$$

This completes the inductive step. □

Now note $\mathrm{Var}[Z]$ in general is equal to $\mathbb{E}Z^2 - (\mathbb{E}Z)^2$. This implies that

$$\mathbb{E}(\tilde{n} - n)^2 = \mathrm{Var}[2^{X_n} - 1] = (1/2)n^2 - (1/2)n - 1 < (1/2)n^2$$

and thus

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\varepsilon^2}, \tag{10}$$

which is not particularly meaningful since the right hand side is only better than 1/2 failure probability when $\varepsilon \geq 1$ (which means the estimator may very well always be 0).

5.1 Morris+

To decrease the failure probability of Morris's basic algorithm, we instantiate s independent copies of Morris's algorithm and average their outputs. That is, we obtain independent estimators $\tilde{n}_1, \dots, \tilde{n}_s$ from independent instantiations of Morris's algorithm, and our output to a query is

$$\tilde{n} = \frac{1}{s} \sum_{i=1}^{s} \tilde{n}_i$$

Since each $\tilde{n}_i$ is an unbiased estimator of n, so is their average. Furthermore, since variances of independent random variables add, and multiplying a random variable by some constant $c = 1/s$ causes the variance to be multiplied by c^2, the right hand side of (10) becomes

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{2s\varepsilon^2} < \delta$$

for $s > 1/(2\varepsilon^2\delta) = \Theta(1/(\varepsilon^2\delta))$.

5.2 Morris++

It turns out there is a simple technique (which we will see often) to reduce the dependence on the failure probability δ from $1/\delta$ to $\log(1/\delta)$. The technique is as follows.

We run t instantiations of Morris+, each with failure probability $\frac{1}{3}$. Thus, for each one, $s = \Theta(1/\varepsilon^2)$. We then output the median estimate from all the s Morris+ instantiations. Note that the expected number of Morris+ instantiations that succeed is at least $2t/3$. For the median to be a bad estimate, at most half the Morris+ instantiations can succeed, implying the number of succeeding instantiations deviated from its expectation by at least $t/6$. Define

$$Y_i = \begin{cases} 1, & \text{if the } i\text{-th Morris+ instantiation succeeds.} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Then by the Chernoff bound,

$$\mathbb{P}\left(\sum_{i=1}^{t} Y_i \leq \frac{t}{2}\right) \leq \mathbb{P}\left(\left|\sum_{i=1}^{t} Y_i - \mathbb{E}\sum_{i=1}^{t} Y_i\right| \geq \frac{t}{6}\right) \leq 2e^{-t/3} < \delta \tag{12}$$

for $t \in \Theta(\lg(1/\delta))$.

Overall space complexity. Note the space is a ranadom variable. We will not show it here, but one can show that the total space complexity is, with probability $1 - \delta$, at most

$$O(\varepsilon^{-2} \lg(1/\delta)(\lg \lg(n/(\varepsilon\delta))))$$

bits. In particular, for constant ε, δ (say each $1/100$), the total space complexity is $O(\lg \lg n)$ with constant probability. This is exponentially better than the $\log n$ space achieved by storing a counter.

An improvement One issue with the above is that the space is $\Omega(\varepsilon^{-2} \lg \lg n)$ for $(1 + \varepsilon)$-approximation, but the obvious lower bound is only $O(\lg(\lg_{1+\varepsilon} n)) = O(\lg(1/\varepsilon) + \lg \lg n)$. This can actually be achieved. Instead of incrementing the counter with probability $1/2^X$, we do it with probability $1/(1 + a)^X$ and choose $a > 0$ appropriately. We leave it to the reader as an exercise to find the appropriate value of a and to figure out how to answer queries.

References

[CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comp.*, 19(90):297–301, 4 1965.

[DL42] Gordon C. Danielson and Cornelius Lanczos. Some improvements in practical fourier analysis and their application to x-ray scattering from liquids. *J. Franklin Inst.*, 233(4):365–380, 4 1942.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 10 1978.