CS 598CSC: Algorithms for Big Data	Lecture date: Sept 2, 2014
Instructor: Chandra Chekuri	Scribe: Chandra Chekuri

1 F_2 Estimation

We have seen a generic algorithm for estimating the F_k, the k'th frequency moment of a stream using $\tilde{O}(n^{1-1/k})$-space for $k \geq 1$. Now we will see an amazingly simple algorithm for F_2 estimation due to [2].

AMS-F_2-ESTIMATE:

$\mathcal{H}$ is a 4-universal hash family from $[n]$ to $\{-1, 1\}$
choose h at random from $\mathcal{H}$
$z \leftarrow 0$
While (stream is not empty) do
  a_j is current item
  $z \leftarrow z + h(a_j)$
endWhile
Output z^2

An conceptually equivalent way to describe the algorithm is the following.

AMS-F_2-ESTIMATE:

Let $Y_1, Y_2, \ldots, Y_n$ be $\{-1, +1\}$ random variable that are 4-wise independent
$z \leftarrow 0$
While (stream is not empty) do
  a_j is current item
  $z \leftarrow z + Y_{a_j}$
endWhile
Output z^2

The difference between the two is that the former one is a streaming friendly. Instead of keeping $Y_1, \ldots, Y_n$ explicity we sample h from a 4-wise independent hash family so that h can be stored compactly in $O(\log n)$-space and we can generate $Y_i = h(i)$ in $O(\log n)$ time on the fly. We will analyze the algorithm in the second description.

Let $Z = \sum_{i \in [n]} f_i Y_i$ be the random variable that represents the value of z at the end of the stream. Note that for all $i \in [n]$, $\mathbf{E}[Y_i] = 0$ and $\mathbf{E}[Y_i^2] = 1$. Moreover, since $Y_1, \ldots, Y_n$ are 4-wise-independent and hence also 2-wise independent, $\mathbf{E}[Y_i Y_{i'}] = 0$ for $i \neq i'$. The expected value of the output is

$$\mathbf{E}[Z^2] = \sum_{i,i' \in [n]} f_i f_{i'} \mathbf{E}[Y_i Y_{i'}] = \sum_{i \in [n]} f_i^2 \mathbf{E}[Y_i^2] + \sum_{i \neq i'} f_i f_{i'} \mathbf{E}[Y_i Y_{i'}] = \sum_{i \in [n]} f_i^2 = F_2.$$

We can also compute the variance of the output which is $\mathbf{E}[Z^4]$.

$$\mathbf{E}[Z^4] = \sum_{i \in [n]} \sum_{j \in [n]} \sum_{k \in [n]} \sum_{\ell \in [n]} f_i f_j f_k f_\ell \mathbf{E}[Y_i Y_j Y_k Y_\ell].$$

Via the 4-wise independence of the Y's we have that $\mathbf{E}[Y_iY_jY_kY_\ell] = 0$ if there is an index among i, j, k, ℓ that occurs exactly once in the multiset, otherwise it is 1. If it is 1 there are two cases: all indices are the same or there are two distinct indices that occur twice each. Therefore,

$$\mathbf{E}[Z^4] = \sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]}\sum_{\ell\in[n]} f_if_jf_kf_\ell\mathbf{E}[Y_iY_jY_kY_\ell] = \sum_{i\in[n]} f_i^4 + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2f_j^2.$$

Thus, we have

$$\begin{aligned}
\mathbf{Var}[Z^2] &= \mathbf{E}[Z^4] - (\mathbf{E}[Z^2])^2 \\
&= F_4 - F_2^2 + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2f_j^2 \\
&= F_4 - (F_4 + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2f_j^2) + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2f_j^2 \\
&= 4\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2f_j^2 \\
&\leq 2F_2^2.
\end{aligned}$$

Let $X = Z^2$ be the output estimate. We have $\mathbf{E}[X] = F_2$ and $\mathbf{Var}[X] \leq 2F_2^2 \leq 2\mathbf{E}[X]^2$. We now apply the standard idea of averaging $O(1/\epsilon^2)$ estimates to reduce variance, apply Chebyshev on the average estimator to see that it is a ϵ-approximation with $> 1/2$ probability. Then we apply the median trick with $\log(\frac{1}{\delta})$-independent averaged estimators to obtain an (ϵ, δ)-approximation. The overall space requirement is $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log n)$ and this is also the time to process each element.

2 (Linear) Sketching and Streaming with Updates

The F_2 estimation algorithm is amazingly simple and has the following interesting properties. Suppose σ_1 and σ_2 are two streams and the algorithm computes z_1 and z_2 on σ_1 and σ_2 It is easy to see that the algorithm on $\sigma = \sigma_1 \cdot \sigma_2$ (the concatenation of the two streams) computes $z_1 + z_2$. Thus the algorithm retains z as a *sketch* of the stream σ. Note that the output of the algorithm is not z but some function of z (in the case of F_2 estimation it is z^2). Moreover, in this special case the sketch is a *linear* sketch which we will define more formally later.

Formally a sketch of a stream σ is a data structure $z(\sigma)$ that has the property that if $\sigma = \sigma \cdot \sigma_2$, $z(\sigma)$ can be computed by combining the sketches $z(\sigma_1)$ and $z(\sigma_2)$. Ideally the combining algorithm should take small space as well. Note that the algorithm can post-process the sketch to output the estimator.

The power of sketching algorithms is illustrated by thinking of more general streaming models than what we have seen so far. We have considered streams of the form $a_1, a_2, \ldots, a_m$ where each a_i is a token, in particular an integer from $[n]$. Now we will consider the following model. We start with a n-dimensional vector/signal $\mathbf{x} = (0, 0, \ldots, 0)$ and the stream tokens consists of *updates* to coordinates of $\mathbf{x}$. Thus each token $a_t = (i_t, \Delta_t)$ where $i_t \in [n]$ and Δ_t is a number (could be a real number and be negative). The token a_t udpates the i'th coordinate of $\mathbf{x}$:

$$x_{i_t} \leftarrow x_{i_t} + \Delta_t.$$

We will let $\mathbf{x}_t$ be the value of $\mathbf{x}$ after the updates corresponding to $a_1, a_2, \ldots, a_t$.

If the Δ_t's are allowed to be negative the model is called *turnstile streams*; note that Δ_t being negative allows items to be deleted. If $\mathbf{x}_t$ is required to be always non-negative, we have the *strict* turnstile stream model. A further special case is when Δ_t is required to be positive and is called the *cash register* model.

Linear sketches are particularly simple and yet very powerful. A linear sketch corresponds to a $k \times n$ projection matrix M and the sketch for vector $\mathbf{x}$ is simply $M\mathbf{x}$. Composing linear sketches corresponds to simply addding the sketches since $M\mathbf{x} + M\mathbf{x}' = M(\mathbf{x}+\mathbf{x}')$. In the streaming setting when we see a token $a_t = (i_t, \Delta_t)$, updating the sketch corresponds to adding $\Delta_t M e_{i_t}$ to the sketch where e_{i_t} is the vector with 1 in row i_t and 0 every where else. To implement the algorithm in small space it would suffice to be able to generate the i'th column of M efficienty on the fly rather than storing the entire matrix M.

F_2 estimation as linear sketching: It is not hard to see that the F_2 estimation algorithm we have seen is essentially a linear sketch algorithm. Consider the matrix M with $k = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ rows where each entry is in $\{-1, 1\}$. The sketch is simply $\mathbf{z} = M\mathbf{x}$. The algorithm post-processes the sketch to output its estimator.

Note that because the sketch is linear it does not matter whether $\mathbf{x}$ is negative. In fact it is easy to see this from the analysis as well. In particular this implies that we can estimate $||f_\sigma - f_{\sigma'}||_2$ where f_σ and $f_{\sigma'}$ are the frequency vectors of σ and σ' respectively. Similarly, if $\mathbf{x}$, $\mathbf{x}'$ are two n-dimensional signals representing a time-series then the ℓ_2 norm of their difference can be estimated by making one pass of the signals even when the signals are given via a sequence of updates which can even be interspersed (of course we need to know the identity of the signals from which the updates are coming from).

3 Johnson-Lindenstrauss Lemma and Dimension Reduction in ℓ_2

The AMS linear sketch for F_2 estimation appears magical. One way to understand this is via the dimensionality reduction for ℓ_2 spaces given by the well-known Johnson-Lindenstrauss lemma which has many applications. The JL Lemma can be stated as follows.

Theorem 1 (JL Lemma) *Let Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be any n points/vectors in $\mathbb{R}^d$. For any $\epsilon \in (0, 1/2)$, there is linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ where $k \leq 8 \ln n/\epsilon^2$ such that for all $1 \leq i < j \leq n$,*

$$(1-\epsilon)||v_i - v_j||_2 \leq ||f(v_i) - f(v_j)||_2 \leq ||v_i - v_j||_2.$$

Moreover f can be obtained in randomized polynomial-time.

The implication of the JL Lemma is that any n points in d-dimensional Euclidean space can be projected to $O(\ln n/\epsilon^2)$-dimensions while preserving all their pairwise Euclidean distances.

The simple randomized algorithm that proves the JL Lemma is the following. Let M be a $k \times d$ matrix where each entry M_{ij} is picked independently from the standard $\mathcal{N}(0, 1)$ normal distribution. Then the map f is givens as $f(\mathbf{v}) = \frac{1}{\sqrt{k}} M\mathbf{v}$. We now sketch why this works.

Lemma 2 *Let $Z_1, Z_2, \ldots, Z_k$ be independent $\mathcal{N}(0, 1)$ random variables and let $Y = \sum_i Z_i^2$. Then, for $\epsilon \in (0, 1/2)$, there is a constant c such that,*

$$\Pr[(1-\epsilon)^2 k \leq Y \leq (1+\epsilon)^2 k] \leq 2e^{c\epsilon^2 k}.$$

In other words the sum of squares of k standard normal variables is sharply concentrated around its mean which is k. In fact the distribution of Y has a name, the χ^2 distribution with parameter k. We will not prove the preceding lemma. A proof via the standard Chernoff-type argument can be found in various places.

Assuming the lemma we can prove the following.

Lemma 3 *Let $\epsilon in (0, 1/2)$ and $\mathbf{v}$ be a unit-vector in $\mathbb{R}^d$, then $(1-\epsilon) \leq ||M\mathbf{v}||_2 \leq (1+\epsilon)||$ with probability at least $(1 - 2e^{c\epsilon^2 k})$.*

Proof: First we observe a well-known fact about normal distributions. Let X and Y be independent $\mathcal{N}(0,1)$ random variables. Then $aX + bY$ is $\mathcal{N}(0, \sqrt{a^2+b^2})$ random variable.

Let $\mathbf{u} = \sqrt{k}M\mathbf{v}$. Note that $\mathbf{u}$ is a random vector. Note that $u_i = \sum_{j=1}^{n} v_j \sqrt{k} M_{ij}$. Since each $\sqrt{k}M_{ij}$ is $\mathcal{N}(0,1)$ random variable and all entries are independent we have that $u_i \simeq \mathcal{N}(0,1)$ because the variance of u_i is $\sum_j v_i^2 = 1$ (note that $\mathbf{v}$ is a unit vector). Thus $u_1, u_2, \ldots, u_k$ are independent $\mathcal{N}(0,1)$ random variables. Therefore $||u||_2^2 = \sum_i =^k u_i^2$. Applying Lemma 2, we have

$$\Pr[(1-\epsilon)^2 k \leq ||u||_2^2 \leq (1+\epsilon)^2 k] \geq 1 - 2e^{c\epsilon^2 k}.$$

□

Unit-vectors are convenient for the proof but by scaling one obtains the following easy corollary.

Corollary 4 *Let $\epsilon in (0, 1/2)$ and $\mathbf{v}$ be any vector in $\mathbb{R}^d$, then $(1-\epsilon)||\mathbf{v}||_2 \leq ||M\mathbf{v}||_2 \leq (1+\epsilon)||||\mathbf{v}||_2$ with probability at least $(1 - 2e^{c\epsilon^2 k})$.*

Now the JL Lemma follows easisly via a union bound. Let $k = c' \ln n/\epsilon^2$ where c' is chosen based on c. Consider any pair $\mathbf{v}_i, \mathbf{v}_j$.

$$\Pr[(1-\epsilon)||\mathbf{v}_i - \mathbf{v}_j||_2 \leq ||M(\mathbf{v}_i - \mathbf{v}_j||_2 \leq (1+\epsilon)||\mathbf{v}_i - \mathbf{v}_j||_2] \geq (1 - 2e^{c\epsilon^2 k}) \geq 1 - 2e^{c\epsilon^2 \cdot c' \ln n/\epsilon^2} \geq 1 - \frac{2}{n^{cc'}}.$$

If $cc' \geq 3$ then the probability of the distance between $\mathbf{v}_i$ and $\mathbf{v}_j$ being preserved to within a relative ϵ-approximation is at least $1 - 1/n^3$. Since there are only $n(n-1)/2$ pairs of distances, the probability that all of them will be preserved to this error tolerance is, via the union bound, at least $(1 - 1/n)$.

Bibliographic Notes: See Chapter 6 of Amit Chakrabarti's lecture notes. A lot of work has been done in the algorithmic community to make the dimensionality reduction faster to evaluate. An interesting result is due to Achlioptas [1] who showed that the matrix M whose entries we chose from $\mathcal{N}(0,1)$ can in fact be chosen from $\{-1, 0, 1\}$; the discrete entries create a sparser matrix and the resulting matrix multiplication is computationally easier.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. Preliminary version in *Proc. of ACM STOC* 1996.