Sketching Algorithms for Big Data Fall 2017

Lecture 3 — Sep 7, 2017

Prof. Piotr Indyk *Scribe: Zhixian Lei*

1 Overview

In the last lecture we talked about counting distinct elements in a stream. First we had an idealized algorithm which uses hash function $h : [n] \to [0, 1]$ and inspects the minimum one. Then we fixed the idealized algorithm to actual algorithm with pair-wise independent hash function. The space we need in the final version of the algorithm we got is $O(\frac{1}{\epsilon^2} \log^2 n \log \frac{1}{\delta})$ And we know that $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} + \log n)$ is achievable and this also matches the lower bound.

This is the first lecture in MIT so in this lecture we reintroduce the data stream model. Then we discussed about estimating L_2 norm of a data stream. We introduced two methods for estimating L_2 norm: a) Alon-Matias-Szegedy (AMS) algorithm [1] b) Johnson-Lindenstrauss (JL) lemma [2]. The AMS algorithm is often appeared in textbook which is a very elegant and simple algorithm while JL lemma is a very powerful tool which is widely used in designing big data algorithm.

2 Data Stream Model

We will not get into detail about introducing data stream model since it was done in first lecture but we will outline some important features of this model.

The data stream model consider the situation that we continuously receive a stream of data, do computation about it and answer queries about the stream of data we have seen. We can think of the data stream as a tape passing through us. The size of the data is huge and we can only see the data in this order so typically we can not access the data that is already passed away. The tricky point of this model is that we can not remember all the data we have seen so far since we are dealing with big data problem but still we want to do something meaningful and answer the queries within some accuracy with limited memory.

We denote the data stream by $i_1, i_2, \ldots i_n$ and each $x_i \in [m]$. We assume n and m are known and we can use that to set up the parameter of the algorithm. Our storage is sub-linear in n and m, typically of order $\log^{O(1)} n$ or $\log^{O(1)} m$. Usually our algorithm is randomized so we only guaranteed to succeed with probability $1 - \delta$ and the answer of the query is usually an approximation of the actual value so our answer is an up to $1 \pm \varepsilon$ multiplicative approximation of the real answer of the query

If the query we are interested in about care about the counts of each element in $[m]$, we can represent a data stream by a vector $x = (x_1, x_2, \ldots, x_m)$ where x_1 is the number of element i appeared in the stream we have seen so far. Then observing a new element of the stream is equivalent to plus one on corresponding coordinate of x. We can extend this representation by view a stream as a sequence of updates (i, a) which updates x only its ith coordinate

$$x_i \leftarrow x_i + a$$

And initially $x_i = 0$ for every i. There is no reason why a must be 1 or positive number so we can even allow a to be negative although sometimes $a < 0$ can be a problem when we design algorithm for some kind of query. That is a generalization of the model stream model and we can always think of the stream data model in this way throughout this lecture. In this formulation, estimating the count of distinct elements is the same as estimating $\|x\|_0$ and the algorithm we discussed in last lecture is also applicable in estimating $\|x\|_0$ in this generalized model. And in this lecture we talked about estimating $\|x\|_2$. As we mentioned above, we mainly introduced two algorithm for solving $\|x\|_2$: the AMS algorithm and JL lemma.

3 Motivation For Estimating L_2 Norm

Basically, $\|x\|_2$ is an important feature for vector x. If we think of x as a empirical distribution for data samples appeared in the stream, then $\|x\|_2$ corresponds to the second order moment of x which reveal the spicky-ness of x in some sense. Since $\|x\|_1 = n$ is fixed, $\|x\|$ takes its minimum value when every $x_i = n/m$ which corresponding to uniform distribution and $\|x\|_2$ gets bigger when x_i is further form uniform. Estimating L_2 norm of the data is originated in database application which I am not going to talk about.

4 AMS Algorithm

The basic idea for AMS algorithm is to get an unbiased estimator of $\|x\|_2^2$ by linear sketching and try to prove concentration by bounding the variance the estimator.

4.1 Description of the algorithm

Choose $r_1, r_2, \ldots, r_m$ i.i.d. random variable with $\mathbb{P}(r_i = 1) = \mathbb{P}(r_i = -1) = 0.5$ for every i. Then we maintain $Z = (r, x) = \sum r_i x_i$. Note that Z is linear in x, so when we update x by (i, a), we only need to increment Z by $r_i a$. Then our estimator for $\|x\|_2^2$ is Z^2. This is a very simple with very clever idea. Next we will see the analysis of the correctness of the algorithm.

4.2 Correctness of the algorithm

First we show that $\mathbb{E}(Z^2) = \|x\|_2^2$

$$\mathbb{E}(Z^2) = \mathbb{E}(x^T r r^T x) = x^T \mathbb{E}(rr^T) x = x^T x = \|x\|_2^2$$

Here we use the fact that for every i, $\mathbb{E}(r_i^2) = 1$ and for every $i \neq j$, $\mathbb{E}(r_i r_j) = \mathbb{E}(r_i)\,\mathbb{E}(r_j) = 0$

Next we show that $\mathrm{var}(Z^2)$ is $O(\|x\|_2^4)$

$$\mathrm{var}(Z^2) = \mathbb{E}(Z^4) - \mathbb{E}(Z^2)^2 = \mathbb{E}(Z^4) - \|x\|_2^4$$

We decompose $\mathbb{E}(Z^4)$ as

$$\mathbb{E}(Z^4) = \sum_{i,j,k,l} \mathbb{E}(x_i x_j x_k x_l r_i r_j r_k r_l) = \sum_{i,j,k,l} x_i x_j x_k x_l \,\mathbb{E}(r_i r_j r_k r_l)$$

Notice $\mathbb{E}(r_i r_j r_k r_l)$ is 0 if there is one index only appear once in i, j, k, l so we only need to consider the case where every distinct index appears at least twice. Then there are two cases such that $\mathbb{E}(r_i r_j r_k r_l) = 1$: a) there are two distinct pairs in (i, j, k, l) b) i, j, k, l are identical. So we have

$$\mathbb{E}(Z^4) = \frac{1}{2} C_4^2 \sum_{i \neq j} x_i^2 x_j^2 + \sum_i x_i^4 = 3 \sum_{i \neq j} x_i^2 x_j^2 + \sum_i x_i^4$$

Then we can bound $\mathbb{E}(Z^4)$ by

$$\mathbb{E}(Z^4) = 3 \sum_{i \neq j} x_i^2 x_j^2 + \sum_i x_i^4 \leq 3 \sum_{i \neq j} x_i^2 x_j^2 + 3 \sum_i x_i^4 \leq 3 \|x\|_2^4$$

So

$$\mathrm{var}(Z^4) \leq 2 \|x\|_2^4$$

Then we can apply Chebyshev inequality

$$\mathbb{P}(|\, \mathbb{E}(Z^2) - \|x\|_2^2| \leq \sqrt{2} c \|x\|_2^2) \leq 1/c^2$$

As we can observe, this bound is sometimes too loose to be informative for approximating $\|x\|_2^2$. For example, if we choose $c = 3$ ($\delta = 1/9$), then we have

$$\mathbb{P}(|\, \mathbb{E}(Z^2) - \|x\|_2^2| \leq 3\sqrt{2} \|x\|_2^2) \leq 1/9$$

However we know that $E(Z^2) \geq 0$, so the lower bound it gives is even worse than natural bound. To boost the error bound, we can just repeat this estimator for k times independently, then we use the average of those k estimations to get a more accurate estimator. Taking taking the mean of k independent estimators does not affect the mean of the estimator but it will reduce the variance by a factor of k.

More specifically, we maintain $Z_1, Z_2, \ldots, Z_k$ where for every j, $Z_j = \sum_i r_{ji} x_i$. r_{ij} are i.i.d. random variables with the same distribution as above. Then our estimator for $\|x\|_2^2$ is $(\sum_j Z_j^2)/k$. To prove the correctness of this improved estimator, we can also identify the expectation and variance of the estimator. So $\mathbb{E}(Y) = (\sum_j \mathbb{E}(Z_j^2))/k = \|x\|_2^2$ and $\mathrm{var}(Y) = (\sum_j \mathrm{var}(Z_j^2))/k^2 \leq 2\|x\|_2^4/k$. The Chebyshev inequality gives

$$\mathbb{P}(|\, \mathbb{E}(Z^2) - \|x\|_2^2| \leq c\sqrt{2/k} \|x\|_2^2) \leq 1/c^2$$

If we set $c = \Theta(1)$ and $k = \Theta(1/\varepsilon^2)$, we get a $(1 \pm \epsilon)$ approximation with constant probability

The space we need for this algorithm is dominated by the space of storing Z_j for all j if we temporarily ignore the space to generate r_{ji}. For a fixed j, the maximum possible value for Z_j is mn so to store Z_j we need $\log(mn)$. And there are $O(1/\varepsilon^2)$ such Z_j so the total space we need is $O(\log(mn)/\epsilon^2)$ bits

Now we can consider how to actually generate these r_i. If we look at the proof of correctness in detail, we can realize that we only need 4-wise independence of r_i because throughout the analysis when we computing $\mathbb{E}(Z^2)$ and $\mathbb{E}(Z^4)$, the maximum degree of the polynomial in r_i is 4. If we pick r_i by 4-wise independent hash function, all the calculations of the expectations will remain the same so the analysis and error bounds still hold. We know that we can generate r_i from $O(\log m)$ random bits so this is not dominant space consumption in the algorithm.

We can formulate our algorithm in vector form. Let R be an k by m matrix where $R_{ij} = r_{ij}$, then basically we are just maintaining the vector $Z = Rx$ throughout the algorithm and answering the query by outputting $\|Rx\|_2^2/k$. Rx is exactly the linear sketch of x. We can understand the sketch as a compression of x which largely reduces the dimension of vector but it still has enough information to give a good estimation for the query we care about. Since it's a linear sketch, it's very convenient for data stream concatenation which often appears in real life application. For example, if there are two data stream x and y in two different places, and the headquarter want to estimate $\|x+y\|_2^2$, it's not necessary to send x and y to the headquarter and do the calculation. Instead, we can send Rx and Ry to headquarter and output $\|Rx+Ry\|_2^2/k$ which reduce the load of transferring the data.

Although AMS algorithm is very simple beautiful, the error bound is not tight enough for arbitrary error probability δ. If we let $c = O(1/\sqrt{\delta})$, we need $k = O(\frac{1}{\delta\varepsilon^2})$ which is linear in $O(1/\delta)$. In fact, the actual error bound should be much better than our analysis since we only consider variance and apply Chebyshev inequality. If we somehow consider higher moment of this estimator, we should get better error bound for this sketch. However, applying other sketching algorithm, we can prove exponential error bound and obtain $k = O(\log(1/\delta))$ easily. Next we will talk about JL sketching and prove the exponential error bound in this algorithm.

5 JL sketching

In JL sketching, we use normal distribution instead of two point distribution in AMS algorithm. The advantage of normal distribution is that it's fully analytic and it's close under linear operation so we can derive the exponential tail bound much easier. First we recap some basics about normal distribution. Normal distribution is distributed on the whole real number and the shape is like a bell. Normal distribution with mean μ and variance σ^2 is denoted by $\mathcal{N}(\mu, \sigma)$, the density function for $\mathcal{N}(\mu, \sigma)$ is

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(x-\mu)^2/2\sigma^2)$$

One important feature for Normal distribution is that it's close under linear transformation. For example, if X and Y are independent random variable with normal distribution. Then $X+Y$ has normal distribution. For more information about Normal distribution, please check corresponding wikipedia page.

5.1 Description of the sketch

The sketch matrix R is also an k by m matrix where each entry R_{ij} is an i.i.d. random variable from $\mathcal{N}(0,1)$ Then everything else is the same as the AMS sketch. That is, we also maintain the vector Rx and when we want to answer $\|x\|_2^2$, just output $\|Rx\|_2^2/k$. Next we will analyze this algorithm especially the exponential tail bound from this sketch

5.2 Analysis of the algorithm

As before we still have $\mathbb{E}(\|Rx\|_2^2/k) = \|x\|_2^2$ since

$$\mathbb{E}(\|Rx\|_2^2/k) = \frac{1}{k}\,\mathbb{E}(x^T R^T Rx) = \frac{1}{k}x^T\,\mathbb{E}(R^T R)x = \|x\|_2^2$$

where $\mathbb{E}(R^T R)$ is a diagonal matrix with all diagonal entries being k. For every $i \in [k]$, the ith diagonal entry is $\mathbb{E}(\sum_k R_{ki}^2) = \sum_k \mathbb{E}(R_{ki}^2) = k$. For every (i,j) where $i \neq j$, the (i,j) entry is $\mathbb{E}(\sum_k R_{ki}R_{kj}) = \sum_k \mathbb{E}(R_{ki}R_{kj}) = 0$, Next we will show that

$$\mathbb{P}(|\|Rx\|_2^2 - k\|x\|_2^2| \geq \varepsilon k\|x\|_2^2) \leq \exp(-C\varepsilon^2 k)$$

The proof follows the notes by Ben Rossman and Michel Goemans. Let's suppose $\|x\|_2^2 = 1$ and let $Z = Rx$, Then we are going to prove

$$\mathbb{P}(|\|Z\|_2^2 - k| \geq \varepsilon k) \leq \exp(-C\varepsilon^2 k)$$

We provide the prove for one side

$$\mathbb{P}(\|Z\|_2^2 \geq (1+\varepsilon)k) \leq \exp(-\varepsilon^2 k + O(k\varepsilon^3))$$

And the other is very similar. Let $Y = \|Z\|_2^2$ and let $\alpha = k(1+\epsilon)^2$, we have for $s > 0$

$$\mathbb{P}(Y > \alpha) = \mathbb{P}(\exp(sY) > \exp(s\alpha)) \leq \exp(-s\alpha)\,\mathbb{E}(\exp(sY))$$

Here we use Markov inequality. We can decompose $\mathbb{E}(\exp(sY))$ by independence:

$$\mathbb{E}(\exp(sY)) = \prod_i E(\exp(sZ_i^2))$$

By closure property, Z_i also has normal distribution. Since for every i

$$\mathbb{E}(Z_i) = \sum_j \mathbb{E}(r_{ij}x_j) = 0$$

$$\mathrm{var}(Z_i) = \mathbb{E}(Z_i^2) = \sum_j \mathbb{E}(r_{ij}^2 x_j^2) = \|x\|_2^2 = 1$$

we obtain that $Z_i \sim \mathcal{N}(0,1)$, so we can calculate $E(\exp(sZ_i^2))$ analytically

$$\mathbb{E}(\exp(sZ_i^2)) = \frac{1}{\sqrt{2\pi}}\int \exp(st^2)\exp(-t^2/2)dt = \frac{1}{\sqrt{1-2s}}$$

So we get

$$\mathbb{P}(Y \geq \alpha) = \exp(-s\alpha)(1-2s)^{-k/2}$$

choose $\alpha = k(1+\varepsilon)^2$ and plug it into above equation we have

$$\mathbb{P}(Y \geq \alpha) = \exp(-\varepsilon k - \varepsilon^2 k/2 + k\ln(1+\varepsilon)) = \exp(-k\varepsilon^2 + kO(\varepsilon^3))$$

Here we use the taylor expansion $ln(1+x) = x - x^2/2 + O(x^3)$. Upon having this tail bound, we prove the correctness of the estimation and we obtain better parameter for k. Let $\exp(-C\varepsilon^2 k) = \delta$ we have $k = O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ to get $1 \pm \varepsilon$ approximation with probability $1-\delta$

We can also reduce the randomness needed in JL sketching by using k-wise independence to generate R, however we can not achieve exponential tail bound if we only allow const-wise independent randomness since in the proof we decompose $\mathbb{E}(\exp(sY))$ into $\prod_i \mathbb{E}(\exp(sZ_i^2))$ where we need k-wise independence to make this go through.

As for time complexity of computing these sketch, basically we need $O(k)$ to update the sketch. This could be a problem is k is large but there are some improvements that we can apply to reduce updating time. We will introduce fast JL and sparse JL a few lectures later

References

[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[2] Johnson, William B., and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics.*, 26(1):189–206, 1984.