The space complexity of approximating the frequency moments.

Seminar Report

Naser Dawod - 323953893 Ahmad Jorban - 211437223

1 Introduction

The paper was written by Noga Alon, Yossi Matias, and Mario Szegedy to addresses the problem of approximating the frequency moments of a data stream, which is a measure of the distribution of elements in the stream. The authors consider the space complexity of approximating the frequency moments, which refers to the amount of memory needed to compute the approximation.

The frequency moments, also known as the raw moments, are a set of statistical measures that provide information about the distribution of data and are commonly used in statistical analysis. The k-th moment, denoted as F_k, is the sum of the k-powers of the data values. Note that it is rather straightforward to maintain the frequency moments by maintaining a full histogram on the data (a counter m_i for each data value), which requires memory of size Ω(n) for a group of n values. However, the calculation of frequency moments can be computationally expensive, were the memory used is limited, especially for large datasets. This is a significant problem in today's era of big data, where the amount of data being collected and analyzed is increasing at a rapid pace. Thus, the problem of computing or estimating the frequency moments in one pass under memory constraints arises. The paper addresses this problem by exploring the use of approximation techniques to reduce the space complexity while maintaining the accuracy of the frequency moments.

The main research question of the paper is: How can the space complexity of approximating frequency moments be reduced while maintaining accuracy? The paper proposes an approach using approximation techniques to approximate frequency moments and compares the results with the exact calculation of frequency moments. By reducing the space complexity of approximating frequency moments, it will become possible to analyze large datasets more efficiently, providing valuable insights and improving the overall performance of statistical analysis.

The authors discuss the use of Randomized algorithms to approximate some of the frequency moments F_k using limited memory, such as Whang algorithm to approximate F_0 using only O(log n) memory bits and who showed how to approximate F_1 using O(log log n) memory bits. They also showed that F_k can be approximated randomly using at most $O(n^{1-\frac{1}{k}}log(n))$ memory bits. In addition they show a lower bound for every $k \geq 6$ any randomized algorithm that computes F_k requires at least $\Omega(n^{1-\frac{1}{k}})$ memory bits.

2 Definitions

2.1 Frequency Moments

The frequency moments are the sum of k-powers of the data values, for a given data stream.
Let A be a sequence of elements such that $A = (a_1, a_2, ..., a_m)$ where each $a_i \in N$ in which N $= \{1, 2, ..., n\}$ for $n \in \mathbb{N}$.
Let m_i be the number of occurrences of i in the sequence A, $m_i = |\,\{j|a_j = i\}\,|$.
For each $k \geq 0$ we define the K-th frequency moment of the data stream to be:

$$F_k = \sum_{i=0}^{n} m_i^k$$

Now we can understand the meaning of each frequency moment better.
F_0 is the number of distinct elements appearing in the sequence A.
F_1 is the length of the sequence A such as $F_1 =$ m.
F_2 is is the repeat rate or Gini's index of homogeneity needed in order to compute the surprise index of the sequence.
we Also define F^*_∞ to be the max appearances of an element:

$$F^*_\infty = \max_{0 \leq i \leq n} m_i$$

The numbers F_k are called the frequency moments of A and provide useful statistics on the sequence.

3 Results

The results of the paper are presented in several sections, each of which compares the exact calculation of frequency moments with the proposed approximation techniques. The first section introduces some of the Space efficient randomized approximation algorithms to compute the frequency moments, meanwhile the The second section presents the lower bounds of the space complexity of randomized algorithms that approximate the frequency moments.

In this sections we will list a summary and organize the results shown in different parts of the paper, then in the next section we well provide detailed explanations and proofs for some of the results listed in this sections.

3.1 Space efficient randomized algorithms

Firstly we'll describe several results of some space efficient randomized algorithms for approximating the frequency moments F_k found in the paper:

- **Estimating F_k:** For every $k \geq 1$, $\varepsilon \geq 0$ and $\lambda > 0$ we can compute Y such that

  $$Pr(|Y - F_k| > \lambda \cdot F_k) \leq \varepsilon$$

  using only $O(\frac{k \cdot log(\frac{1}{\varepsilon})}{\lambda^2} \cdot n^{1-\frac{1}{k}} \cdot (log(n) + log(m)))$ memory bits.

- **Improved estimation for F_2:** For every $\lambda > 0$ and $\varepsilon > 0$ we can compute Y such that that

  $$Pr(|Y - F_2| > \lambda \cdot F_2) \leq \varepsilon$$

  using only $O(\frac{k \cdot log(\frac{1}{\varepsilon})}{\lambda^2} \cdot (log(n) + log(m)))$ memory bits.

- **Estimation of F_0:** For every $c > 2$ it is possible to compute Y such that the probability that the ratio between Y and F_0 is not between $\frac{1}{c}$ and c is at most $\frac{2}{c}$ using only $O(log(n))$ memory bits.

3.2 Lower Bounds

Now we'll present the results of the lower bounds found for the space complexity of randomized algorithms that approximate the frequency moments F_k

- **space complexity of approximating** F_∞^*: Any randomized algorithm that computes Y such that $Pr(|Y - F_\infty^*| > \frac{F_\infty^*}{3}) < \varepsilon$. Must use Ω(n) memory bits.

- **space complexity of approximating** F_k: Any randomized algorithm that computes Z_k for $k > 5$, $\gamma < \frac{1}{2}$ such that $Pr(|Z_k - F_k| > 0.1F_k) < \gamma$.
  Must use $\Omega(n^{1-\frac{5}{k}})$ memory bits.

4 Proofs

4.1 Estimating F_k

In order to prove the result we will prove the following Theorem:
For every $k \geq 1$, $\varepsilon \geq 0$, $\lambda > 0$ there exists a randomized algorithm that computes a number Y, given a sequence $A = (a_1, a_2, ..., a_m)$ of members of $N = \{1, 2, ..., n\}$, in one pass and $O(\frac{k \cdot log(\frac{1}{\varepsilon})}{\lambda^2} \cdot n^{1-\frac{1}{k}} \cdot (log(n) + log(m)))$ memory bits.
such that the probability that Y deviates from F_k by more than λF_k is at most ε

$$Pr(|Y - F_k| > \lambda \cdot F_k) \leq \varepsilon$$

let $A = (a_1, a_2, ..., a_m)$ be the data stream of length m where each $a_i \in \{1, 2, ..., n\}$. we define an algorithm that satisfies the theorem.
Note: for now we assume the length of the sequence m is known in advance.
The algorithm proceeds in these steps:

1. We first define $s_1 = \frac{8 \cdot k \cdot n^{1-\frac{1}{k}}}{\lambda^2}$ and $s_2 = 2 \cdot log(\frac{1}{\varepsilon})$.

2. then we define s_2 random variables $Y_1, ..., Y_{s_2}$ where each Y_i the average of s_1 random variables $X_{i,j} : 0 \geq j \geq s_1$, Each $X = X_{i,j}$ is computed the same way, Choose a random member a_p of the sequence A (index p is chosen randomly and uniformly). Let

$$r = |\{q | q \geq p \ \wedge \ a_q = a_p\}|$$

be the number of occurrences of a_p among the sequence A following p. define

$$X = m \cdot (r^k - (r-1)^k)$$

note that each $X_{i,j}$ requires $O(log(m) + log(n))$ memory bits.

3. Finally, we output Y the median of $Y_1, ..., Y_{s_2}$.

Now we prove the correctness of the algorithm, to do so we use Chebyshev's Inequality for each Y_i.
The expected value of X is, by definition

$$E[X] = \frac{1}{m} \sum_{i=1}^{n} (\sum_{j=1}^{m_i} m(j^k - (j-1)^k)) =$$
$$= \frac{m}{m} \cdot \sum_{i=1}^{n} [(1^k - (1-1)^k) + (2^k - (2-1)^k) + \cdots$$
$$+ ((\mathrm{m}_i - 1)^k - (m_i - 2)^k) + (m_i^k - (m_i - 1)^k)] =$$
$$= \sum_{i=1}^{n} m_i = F_k.$$

We found that $E[X] = F_k$.
Now we need to find the variance of X which is $Var[X] = E[X^2] - (E[X])^2$.

$$
\begin{aligned}
E[X^2] &= \tfrac{1}{m}\textstyle\sum_{i=1}^{n}(\sum_{j=1}^{m_i} m(j^k-(j-1)^k))^2 = \\
&= \tfrac{m^2}{m}\textstyle\sum_{i=1}^{n}(\sum_{j=1}^{m_i}(j^k-(j-1)^k)\cdot(j^k-(j-1)^k)) \leq \\
&\leq m\textstyle\sum_{i=1}^{n}(\sum_{j=1}^{m_i} k\cdot j^{k-1}\cdot(j^k-(j-1)^k)) \qquad (1)\\
&= m\textstyle\sum_{i=1}^{n}[k\cdot 1^{k-1}\cdot(1^k-(1-1)^k)+k\cdot 2^{k-1}\cdot(2^k-(2-1)^k)+\cdots+k\cdot m_i^{k-1}\cdot(m_i^k-(m_i-1)^k)] \leq \\
&\leq m\textstyle\sum_{i=1}^{n}[k\cdot m_i^{k-1}\cdot(1^k-(1-1)^k)+k\cdot m_i^{k-1}\cdot(2^k-(2-1)^k)+\cdots+k\cdot m_i^{k-1}\cdot(m_i^k-(m_i-1)^k)] = \\
&= m\textstyle\sum_{i=1}^{n} k\cdot m_i^{2k-1} = m\cdot k\sum_{i=1}^{n}\cdot m_i^{2k-1} = m\cdot k\cdot F_{2k-1}.
\end{aligned}
$$

$$E[X^2] = k\cdot F_1 F_{2k-1}.$$

Where in (1) we used the inequality: $a^k - b^k \leq (a-b)k\cdot a^{k-1}$.

By the definition that each Y_i is the average of s_1 variables X we know that

$$E[Y_i] = E[\frac{\sum_{j=1}^{s_1} X_{i,j}}{s_1}] = \frac{\sum_{j=1}^{s_1} E[X_{i,j}]}{s_1} = \frac{\sum_{j=1}^{s_1} E[X]}{s_1} = \frac{s1\cdot F_k}{s_1} = F_k$$

whereas the variance of Y_i is:

$$Var[Y_i] = Var[\frac{\sum_{j=1}^{s_1} X}{s_1}] = \frac{\sum_{j=1}^{s_1} Var[X]}{s_1^2} = \frac{Var[X]}{s_1} = \frac{E[X^2]-E[X]^2}{s_1} \leq \frac{E[X^2]}{s_1} = \frac{kF_1F_{2k-1}}{s_1}$$

Fact: For every n positive reals $m_1, m_2 ..., m_n$

$$(\textstyle\sum_{n}^{i=1} m_i)(\sum_{n}^{i=1} m_i^{2k-1}) \leq n^{1-\frac{1}{k}}(\sum_{n}^{i=1} m_i^k)^2$$

using the fact mentioned above we can get that

$$Var[Y_i] \leq \frac{kF_1F_{2k-1}}{s_1} = \frac{k(\sum_{n}^{i=1} m_i)(\sum_{n}^{i=1} m_i^{2k-1})}{s_1} \leq \frac{kn^{1-\frac{1}{k}}(\sum_{n}^{i=1} m_i^k)^2}{s_1} = \frac{kn^{1-\frac{1}{k}}F_k^2}{s_1}$$

Using Chebyshev's Inequality and by the definition of s_1:

$$Pr(|Y_i - F_k| > \lambda F_k) \leq \frac{Var[Y_i]}{\lambda^2 F_k^2} = \frac{kn^{1-\frac{1}{k}}F_k^2}{s_1\lambda^2 F_k^2} = \frac{kn^{1-\frac{1}{k}}}{\frac{8\cdot k\cdot n^{1-\frac{1}{k}}}{\lambda^2}\lambda^2} = \frac{1}{8}$$

Therefor we got that for each Y_i: $Pr(|Y_i - F_k| > \lambda F_k) \leq \frac{1}{8}$.

Let Y be the median of $Y_1, ..., Y_{s_2}$ we prove that $Pr(|Y - F_k| > \lambda F_k) \leq \varepsilon$:

using the union bound to find $Pr(|Y_i - F_k| > \lambda F_k,\ for\ at\ least\ one\ i)$ we get

$$
\begin{aligned}
&Pr(|Y_i - F_k| > \lambda F_k,\ for\ at\ least\ one\ i) = \\
&= Pr(|Y_1 - F_k| > \lambda F_k) + \cdots + Pr(|Y_{s_2} - F_k| > \lambda F_k) = s_2\cdot\tfrac{1}{8}
\end{aligned}
$$

Next, note that the median of a set of random variables is the value that separates the lower half from the upper half of the variables. So, if more than half of the Y_i's deviate from F_k by more than λF_k, then Y will also deviate from F_k by more than λF_k. Therefore, we have:

$$Pr(|Y - F_k| > \lambda F_k) \leq Pr(|Y_i - F_k| > \lambda F_k\ for\ at\ least\ one\ i)$$

$$Pr(|Y - F_k| > \lambda F_k) \leq s_2\cdot\tfrac{1}{8} = 2\cdot log(\tfrac{1}{\varepsilon})\cdot\tfrac{1}{8}$$

As we know $2log(\frac{1}{\varepsilon}) \cdot \frac{1}{8} < \varepsilon$ for any $0 < \varepsilon < 1$ thus:

$$Pr(|Y - F_k| > \lambda F_k) \leq \varepsilon$$

The algorithm need to needs to store each of the $Y_1, ..., Y_{s_2}$ random variables and each Y_i is the average of s_1 random variables $X_{i,j}$ where each $X_{i,j}$ requires $O(log(m)+log(n))$ memory bits.
overall the algorithm requires:

$$O(s_1 \cdot s_2 \cdot (log(m) + log(n))) = O(\frac{k \cdot log(\frac{1}{\varepsilon})}{\lambda^2} \cdot n^{1-\frac{1}{k}} \cdot (log(m) + log(n)))$$

Memory bits. □

4.2 Improved estimation for F_2

In order to prove the result we will prove the following Theorem:
For every $\lambda > 0$ and $\varepsilon > 0$ there exists a randomized algorithm that computes a number Y, given a sequence $A = (a_1, ..., a_m)$ of members of N, in one pass and using

$$O(\frac{log(\frac{1}{\varepsilon})}{\lambda^2}(log(m) + log(n)))$$

memory bits. Such that the probability that Y deviates from F_2 by more than λF_2 is at most ε.

$$Pr(|Y - F_2| > \lambda \cdot F_k) \leq \varepsilon$$

Proof: As we did in the previous algorithm, the output Y of the present algorithm is the median of some random variables $Y_1, ...Y_{s_2}$ where each Y_i is the average of s_1 random variables $X_{i,j}$ for $1 \leq j \leq s_1$, But in this algorithm we will use different approach to compute the $X = X_{i,j}$ values.
Let $s_1 = \frac{16}{\lambda^2}$ and $s_2 = 2 \cdot log(\frac{1}{\varepsilon})$.
Fix an explicit set vectors $V = \{v_1, ..., v_h\}$ where $h \in O(n^2)$ and $v_i \in \{1, -1\}^n$, which are four-wise independent. Group of vectors is called four-wise independent if for every 4 coordinates $1 \leq i_1 \leq ... \leq i_4 \leq n$ and values $\epsilon_1, ..., \epsilon_4 \in \{1, -1\}$: exactly $\frac{1}{16}$ of the vectors have the value ϵ_j in their i_j coordinate.
such set can be constructed using the parity check matrices of BCH codes, To implement this construction we need an irreducible polynomial of degree d where d is the smallest power of 2 that applies $2^d \geq n$ over the field GF(2), Its possible to find such a polynomial, and once it found it requires only log(n) memory bits to store it, since its of degree d. given such polynomial its possible to compute each coordinate of each v_i in O(log n) space using a constant number of multiplications in the finite field $GF(2^d)$ and binary inner products of vectors of length d.
going back to X, we choose a random vector $v_p = (\epsilon_1, ..., \epsilon_n)$ (p is chosen uniformly).

Let $Z = \sum_{i=1}^{n} \epsilon_i m_i$, Z is a linear function of the numbers mi, thus it can be computed in one pass from sequence A, during the process we only have to maintain the current value of the sum along with p, Therefore, Z can be computed using only O(log n + log m) bits.
Define $X = Z^2$.
Now we prove correctness of the algorithm.
Note: the random variables ϵ_i are pairwise independent and E[ϵ_i] = 0 for all i.
First we compute the expectation and variance of X:

$$E[X] = E[Z^2] = E[(\sum_{i=1}^{n} \epsilon_i m_i)^2] = \sum_{i=1}^{n} E[\epsilon_i^2] m_i^2 + 2\sum_{1\le i<j\le n}^{n} E[\epsilon_i]E[\epsilon_j]m_i m_j =$$
$$= \sum_{i=1}^{n} E[\epsilon_i^2] m_i^2 = \sum_{i=1}^{n} m_i^2 = F_2$$
$$E[X] = F_2.$$

Similarly, the fact that the variables ϵ_i are four-wise independent implies that

$$E[X^2] = E[Z^4] = E[(\sum_{i=1}^{n} \epsilon_i m_i)^4] =$$
$$= \sum_{i=1}^{n} E[\epsilon_i^4] m_i^4 + 6\sum_{1\le i<j\le n}^{n} E[\epsilon_i^2]E[\epsilon_j^2]m_i^2 m_j^2 =$$
$$= \sum_{i=1}^{n} m_i^4 + 6\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2$$
$$E[X^2] = \sum_{i=1}^{n} m_i^4 + 6\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2.$$

Thus the variance of X is

$$Var[X] = E[X^2] - E[X]^2 = \sum_{i=1}^{n} m_i^4 + 6\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2 - \sum_{i=1}^{n} m_i^2)^2 =$$
$$= \sum_{i=1}^{n} m_i^4 + 6\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2 - \sum_{i=1}^{n} m_i^4 + 2\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2 =$$
$$= 4\sum_{1\le i<j\le n}^{n} m_i^2 m_j^2 \le 2F_2^2$$

The variance of Y_i would be:

$$Var[Y_i] = Var[\frac{\sum_{j=1}^{s_1} X_{i,j}}{s_1}] = \frac{\sum_{j=1}^{s_1} Var[X]}{s_1^2} = \frac{s_1 Var[X]}{s_1^2} = \frac{Var[X]}{s_1}$$

Therefore, by Chebyshev's Inequality, for each fixed i:

$$Pr(|Y_i - F_2| > \lambda F_2) \le \frac{Var[Y_i]}{\lambda^2 F_2^2} = \frac{Var[X]}{s_1 \lambda^2 F_2^2} \le \frac{2F_2^2}{s_1 \lambda^2 F_2^2} = \frac{2F_2^2}{\frac{16}{\lambda^2}\lambda^2 F_2^2} = \frac{1}{8}$$

As in the previous proof, the probability that the median Y of the numbers $Y_1, ..., Y_{s_2}$ deviates from F_2 by more than λF_2 is at most ε

$$Pr(|Y_i - F_2| > \lambda F_2) \le \varepsilon$$

The algorithm need to needs to store each of the $Y_1, ..., Y_{s_2}$ random variables and each Y_i is the average of s_1 random variables $X_{i,j}$ where each $X_{i,j}$ requires $O(log(m)+log(n))$ memory bits.
overall the algorithm requires:

$$O(s_1 \cdot s_2 \cdot (log(m) + log(n))) = O(\frac{log(\frac{1}{\varepsilon})}{\lambda^2}(log(m) + log(n)))$$

Memory bits. □

4.3 Estimation of F_0:

Theorem: For every $c > 2$ exists an algorithm that, given a sequence $A = (a_1, ..., a_m)$ of members of $N = \{1, ..., n\}$, computes a number Y using O(log(n)) memory bits, where the probability that the ratio between Y and F_0 is not between $\frac{1}{c}$ and c is at most $\frac{2}{c}$.

$$Pr(\tfrac{Y}{F_0} < \tfrac{1}{c} \vee \tfrac{Y}{F_0} > c) \leq \tfrac{2}{c}.$$

Proof: we define an algorithm that satisfies the Theorem.
The algorithm proceeds in these steps:

1. Define d to be the smallest integer such that $2^d > n$.
2. Consider the members of N as elements of the finite field $F = GF(2^d)$, where each $v \in F$ is represented by binary vector of length d.
3. Choose $a, b \in F$, uniformly and independently.
4. For each $a_i \in A$ compute $z_i = a \cdot a_i + b$, z_i is represented by a binary vector of length d, Let $r(z_i)$ denote the largest r so that the r rightmost bits of z_i are all 0 and set $r_i = r(z_i)$.
5. Let R be maximum value of r_i, where the maximum is over all elements of A
6. Output $Y = 2^R$

Space Complexity: to implement the algorithm we only have to maintain d = O(log(n)) bits representing a and b and the O(log(log(n))) bits representing the current maximum r_i value, Overall the space complexity is O(log(n)) .

Now lets estimate the probability that Y deviates considerably from F_0, Suppose r is the current maximum value of r_i, since the random mapping $z_i = a \cdot a_i + b$ is uniformly distributed over F we get $Pr(r(z_i) > r) < \frac{1}{2^r}$ and that this mapping is pairwise independent thus for every $a_i,\ a_j\ we\ get\ Pr(r(z_i) > r \wedge r(z_j) > r) = \frac{1}{2^{2r}}$.

Fix a value r for each $x \in N$ that appears in the sequence, let W_x be the indicator random variable where:

$$W_x = \begin{cases} 1 & \text{r(ax + b)} \geq \text{r} \\ 0 & \text{otherwise} \end{cases}$$

since W_x is indicator random variable we get $E[W_x] = \frac{1}{2^r}$.
by the pairwise independence: $Var[W_x] = E[W_x^2] - E[W_x]^2 = \frac{1}{2^r} - \frac{1}{2^{2r}} = \frac{1}{2^r}(1 - \frac{1}{2^r})$

Let $Z_r = \sum_{x \in N} W_x$, the expectation of Z_r is: $E[Z_r] = E[\sum_x W_x] = \sum_x E[W_x] = \frac{F_0}{2^r}$
And by pairwise independence:

$$Var[Z_r] = Var[\sum_x W_x] = \sum_x Var[W_x] = F_0 \cdot \frac{1}{2^r}(1 - \frac{1}{2^r}) \leq \frac{F_0}{2^r}.$$

Note that Z_r cant be negative.
By Markov's Inequality we get that if $2^r > cF_0$:

$$Pr(Z_r > 0) = Pr(Z_r \geq 1) < \frac{E[Z_r]}{1} = \frac{F_0}{2^r} \leq \frac{F_0}{cF_0} \leq \frac{1}{c}$$

Similarly, by Chebyshev's Inequality if $c2^r < F_0$:

$$Pr(Z_r = 0) = Pr(Z_r \leq 0) = Pr(Z_r - E[Z_r] \leq -E[Z_r]) \leq$$

$$\leq Pr(Z_r - E[Z_r] \leq -E[Z_r]) + Pr(Z_r - E[Z_r] > E[Z_r]) =$$

$$= Pr(|Z_r - E[Z_r]| \geq E[Z_r]) = \frac{Var[Z_r]}{E[Z_r]^2} \leq \frac{\frac{F_0}{2^r}}{(\frac{F_0}{2^r})^2} = \frac{2^r}{F_0} \leq \frac{1}{c}$$

Since our algorithm outputs $Y = 2^R$ where R is the maximum r for which $Z_r > 0$, also we know that .

$$Pr(\frac{Y}{F_0} < \frac{1}{c} \vee \frac{Y}{F_0} > c) = Pr(\frac{Y}{F_0} < \frac{1}{c}) + Pr(\frac{Y}{F_0} > c)$$

Using the two inequalities above we get:
the probability that $\frac{Y}{F_0} < \frac{1}{c} \Leftrightarrow Y > cF_0$ is at most $\frac{1}{c}$, and the probability of $\frac{Y}{F_0} > c$ is also at $\frac{1}{c}$.
Therefor we get

$$Pr(\frac{Y}{F_0} < \frac{1}{c} \vee \frac{Y}{F_0} > c) \leq \frac{1}{c} + \frac{1}{c} \leq \frac{2}{c}$$

Thus completing the proof. $\square$