U.C. San Diego—CSE203A_Advanced Algorithm Lecture 6 & 7_Streaming Algorithm
Professor Ramamohan Paturi Scribe by: H.Chen

Lecture 6 & 7 Streaming Algorithm

Lecture 1

We divide the contents of streaming algorithm into 2 parts, today we talk about the first part, which is more into general cases of streaming algorithm.

I. Why we study the streaming algorithm

Nowadays in computer engineering, the arrival of data stream is high-speed and continuous. High speed and continuous means that the potential amount of data is infinite. We are not able to estimate how much data is in one data stream, neither can we estimate how many data streams will come. The conventional method is to save the data on disk, which is not realistic in our case of processing large amount of data. (Even if the disk can store all the data, the main memory will not be able to read the data all at a time). The overhead of random access(in both access time and update time) is too large, thus we can only use one-pass scan. Also, since the data can not be stored, we will not be able to re-access those gone-away data. Usually we will have to make a ensemble decision based on only part of the data. Thus, the problem of computing or estimating the frequency moments in one pass under memory constraints arises naturally in the study of databases and data streams[1].

II. The concept of joint

We can think of 2 applications in which we need the streaming algorithm. Firstly, like IP addresses, many are repeated. Another example is doing joint on spaces.

Let's talk about the concept of joint:

.

Characteristics of elements

ID	NAME	CAR COMPANY			CAR COMPANY	MODELS
a	b	x	n	m	x	c
		x=d	d	d	x=d	

Shared Attribute

To do the joint, the two tables must share the same attribute. Here the car company is the shared attribute. The size of the resulting table: Typically the size has a product structure. Let's assume

x is a particular value. So the resulting table will have $m \times n$ many rows for this specific d. You only do the product when the values x are equal. So you want to know the size of joint before you perform the joint so that you can do the optimization.

So after introducing you the two applications, now let me introduce the general concept of frequency.

III. The general Algorithm

For streaming algorithm you only interested in the statistical properties of the stream.

$\forall 1 \le i \le n, m_i$ is the frequency of the i-th value in the stream.

Some value will appear multiple of times. This value of m_i could be zero, which means many values may not appear at all. Some maybe non-zero, some maybe very big, some maybe very small.

A. Frequency Moments F_k and the way we estimate it

Now let me define the frequency moment F_k. Our purpose is to raise the frequency to some power.

$\forall k \ge 0, F_k = \sum_{i=1}^{n} {m_i}^k$, when $k = 0$, define $0^0 = 0$, $F_0 =$ the number of distinct unique values, which means for ${m_i}^0 = 1$, frequency is 1. For $k = 0$ you need a separate algorithm.

Then we look at F_1, what is F_1? Simply the number of elements. In general, when a stream of data is coming through, we don't know what m (the length of the sequence) is. So in the algorithm let's first assume we know m, then later I will show how we can remove the assumption. So now you see why the sum of squares is related to the size of the joint. So the sum of the squares is going to relate to the size of the joint.

Fix k and make it at least 1($k \ge 1$), we have $F_1 = \sum_{i=1}^{n} {m_i}^1 = {m_1}^1 + {m_2}^1 + \cdots + {m_n}^1 = m$, so F_1 is the length of the sequence.

$$F_2 = \sum_{i=1}^{n} {m_i}^2 = {m_1}^2 + {m_2}^2$$

F_2 is also called the Gini's Coefficient. When $m_1 = m_2$, F_2 reaches its maximum value.

We are interested in the frequency of occurrence of i -th value, the frequency can be 0, which means that the element doesn't appear at all, some other elements maybe very big. So we want to find an algorithm for deterministics. Remember that for streaming algorithm, the data come in one pass, so your goal is to use a logarithmic small space to store whatever you learned about that particular value, so the space will be something like $(\log n + \log m)$, time is less important here so we focus on space. Later we can see some examples in which the space exceed this value. So the important thing is, you look at a certain value, remember something about it, once you have stored the things about it after one pass,that's all you have, you can't go back and look at it again. In the database applications you may be able to look at the data again, but in streaming algorithms, the data is one pass.

More generally we talk about F_k. I'm going to give a very general algorithm for computing F_k. It is not necessary the most efficient algorithm, but it can be applied to all $k \geq 1$ cases. One important thing is that you can not compute F_k precisely, because you have low space complexity, you can't remember everything, so you have to approximate F_k. In this algorithm, F_k is approximate by some factor $(1 \pm \lambda)F_k$. And in the next lecture I will look at 2 special cases F_1 and F_2 and give some more efficient algorithm for computing F_2, F_1 requires some different techniques.

For the case of a fixed F_k $(k \geq 1)$, let's first do without any space constrain. So how are we going to compute F_2 and F_k? Linear space means we have the space proportional to some variable. One relaxation is we can do approximation, so we don't have to compute it precisely. We simply don't have enough space. This conditions implies that we have to use a random variable. One can prove that without randomness, the algorithm can not get the sufficient result.So how can we get the information of F_2? For example you take two random variables, and check the probability that they are equal. Then you square the frequency and sum it. If the frequency of a certain element is high, then the probability of drawing it twice is also high. What we want is exactly the unbiased estimation of X -- expectation $E(X)$, this is the most natural thing we need to think of. You may be able to estimate without much effort. How do we select random variable whose expectation is exactly F_k? And how can we do the approximation without much time? So integration is one thing that we could think of. So we want the k -th power, basically the expectation is like integration. So let's assume that we also know m. Select j uniformly and randomly. So select one of the streaming indexes, let l be the value of the j -th element in the sequence. From the j -th index, start counting the number of occurrences of l in the stream.

This is essential because counting of j is what we can do, we can measure it several times, so you want to boost the signal by repeating the experiment (measure it several times independently) , then you have distribution, then you can estimate it. So let's see how to implement it. Since you know the value of m , you are going to select j uniformly and randomly. Keep going through the stream, once the j -th element comes, start the counter to count that value, and then whenever you see l in the stream from that onward, you can increment the counter. We only measure from some point onwards.

You don't have enough space, so you only maintain one counter, which is extreme case. You need log space to maintain one counter. Sometimes we have to maintain several counters, which requires Polylog space. Let's focus on one variable so this will only take logarithmic space. $k-1$ is the right kind of power we want to see(because it is the highest power of r we will see). Since we are going to integrate X , we are going to design X that $E(X)=F_k$. What's the formula of expectation? It can be expressed in the form as the value of random variable multiplies the probability, and the probability are related.

Inequality is important. Sometimes you can cancel things nicely and get a precise formula, but most of the time you can only get upper bound and lower bound.So for those cases, we need inequalities like Cauchy-schwarz inequality[2], Yansen's inequality[3], arithmetic and geometric inequality to calculate the upper bound and lower bound, so in total three or four inequalities capture all the calculation of streaming algorithms.

So let's focus on the i -th value, because I want to write a general formula then take all the values of probability and sum it. The i -th value is occurring in many locations of the stream.You might select the very first occurrence of the i -th value, or 2nd occurrence, or 3rd occurrence, or m_i occurrence. For concrete example, let's take i =15. $m_i=200$, which means 15 appears in the stream 200 times.What's the chance of selecting the first 15 in the stream? The stream length is m , and has many other elements, not just 15, for example, the stream has 1,5, 15, 200...etc. The index of the first 15 is 3, so what's the chance of selecting the index of 3? Yes, the probability of selecting the first 15(index 3) is $\frac{1}{m}$, each location of the stream is equally likely, see we are not selecting 15, we are selecting the index. What's the corresponding r of the random variable X ? What's the probability of selecting the second occurrence of 15? So you see if you are doing right, these things will cancel out (Telescoping series[4]). So what's the probability of selecting the penultimate(one before the last) 15? It is $\frac{1}{m}$. Because it is the i -th value. And the probability of selecting the last one is also $\frac{1}{m}$. Then you can sum these terms. The probability of selecting any index is $\frac{1}{m}$, when you start counting, will determine the value of r , so the value of the random variable keeps changing depends on which one is selected. We have the constant factor m which

we assumed. The distribution of F_k can be very bad. So doing one measurement of X may not be sufficient. Because the distribution could be so bad that we will not be able to get the estimation after only one sampling. So we want to improve the distribution, by doing independent copies of X, and take their mean, for example. And that is more likely to be close to the truth.We are going to take that many of random variables, so for each variables between 1 and s_1, we have X. X_q has the same formula, and we repeat the measurement s_1 times, and we can take average for example. So each one has the same mean by taking the sample average. In this way we can improve the estimation. So let's define a variable Y. s_1 is the number of times we want to repeat this experiment. We are going to select s_1 independent random j. First write the expectation naively, naively means the probability of selecting j times the value of random variable X given when j is selected. For each index, there is certain probability of being selected. Let's re-organize the sum based on the value of index j. Which means I want to count all the j that have the particular value.Maybe there are no such values, so there are no such j, then the sum will be zero; but if there are 200 such values, then the index j will be selected 200 times, so it depends on which particular instance i that you select. So here the random element is j, so you select j first, that determines X. Then you multiply the probability of j with the value that j is indexing.

Example:

Tuition fee payment--People come and pay fees, this can be considered as time sequence, it is easy to sum this sequence up. We can take the payment of each student and then sum it up, or we can first consider the payment made by CS student, that's called a group, by grouping you also get the same total as without grouping. Sometimes regrouping may simplify life. So regrouping is important for these kind of problem. So the next step, we do s_1 measurements, we want s_1 to be small obviously, for each s_1, we have a corresponding index, so q maps to X_q.This is the sequence of dependencies.

Define X_q $(1 \le q \le s_1)$, s_1 is the number of times we want to repeat the experiment.

$$Y = \frac{1}{s_1}\sum_{q=1}^{s_1} X_q$$ Make the variance smaller by taking average

$$q \xrightarrow{\text{Determine}} j_q \text{ (random frequency)} \xrightarrow{\text{Determine}} X_q \text{ (random element)}$$

Select s_1, s, t, such that $VAR(Y) = \frac{1}{S_1^2}\sum_{q=1}^{s_1} VAR(X_q) \le \frac{1}{8}$

We want to improve the distribution, the central limit theorem[5] says that given certain conditions, the arithmetic mean of a sufficiently large number of iterates of independent random variables, each with a well-defined expected value and well-defined variance, will be approximately normally distributed(Gaussian Distribution), regardless of the underlying distribution. But basically having Gaussian distribution is not enough, we need to know more about the distribution. If the variance is too high, then the Gaussian distribution may not help us. The variance can be as high as the square of the mean. So central limit theorem is very basic, I don't think you even need dependence. So you want to make sure that variance become smaller by taking the average. Chernoff bound[6](Chernoff bound is tighter than Chebyshev bound.) is some what similar to central limit theory. That give you much sharper bounds. So our goal is to show that for this quantity, the variance is sufficiently small. We are going to show that the variance is bounded by some small constants.

So the general algorithm can be described as follow:

1. Select $1 \le j \le m$ uniformly and randomly.

2. Let $l = a_j$

3. $r = number$ of occurrences of l from a_j on wards (inclusive)

4. $X = r^k - (r-1)^k$

The expected value $E(X)$ is, by definition,

$$E(X) = \frac{m}{m}[(1^k + (2^k - 1^k) + \ldots + (m_1{}^k - (m-1)^k)) +$$
$$(1^k + (2^k - 1^k) + \ldots + (m_2{}^k - (m_2 - 1)^k)) + \ldots +$$
$$(1^k + (2^k - 1^k) + \ldots + (m_n{}^k - (m_n - 1)^k))]$$
$$= \sum_{i=1}^{n} m_i{}^k = F_k$$

$$VAR[a\sum X] = a^2 \sum VAR(X)$$

$$VAR(X) = \frac{m^2}{m}[(m_i{}^k - (m_i - 1)^k)^2 + ((m_i - 1)^k - (m_i - 2)^k)^2 + \ldots]$$

$$\leq \sum \frac{m^2}{m} \cdot k \cdot m_i{}^{k-1} \cdot m_i{}^k$$

$$\leq k \cdot F_1 \cdot F_{2k-1} \leq k \cdot n^{1-1/k} \cdot F_K{}^2$$

$$P[|Y - F_k| \geq \lambda F_k] \leq \frac{1}{8} \leq \frac{VAR(Y)}{\lambda^2 F_k{}^2}$$ Chebyshev inequality

$$VAR(Y) = \frac{1}{s^2}\sum_{q=1}^{s_1} VAR(X_q)$$

So this is the kind of variance we want. Remember we want to approximate F_k . Y will be within $(1 \pm \lambda)F_k$. (Y deviates from the mean, F_k , by λF_k), we want this probability to be small. So Y deviates from the mean F_k by $\pm \lambda F_k$, so we want the probability of deviation to be as small as possible. So we select s_1 to make the variance of Y to be sufficiently small. Repeating the experiment--We do measurements independently several times, which means increasing the space. So each random variable X takes certain space, we need s_1 times space, so now the goal is simple, you want to make the variance to be small, so you repeat the experiment s_1 times, select a s_1 really large, so then Y will be within the range of $(1 \pm \lambda)F_k$ with probability at least $7/8$, to design the accuracy given by λ , now we begin the calculation of the variance.

Let μ denote the mean, and chebyshev inequality say, σ is the standard deviation.

$$P = [(Y - \mu) \geq a_\sigma] \leq \frac{1}{\sigma^2}$$

So let me rewrite the expression.

$$P = [|Y - \mu| \geq Z] \leq \frac{1}{(z/\sigma)^2} = \frac{\sigma^2}{Z^2} = \frac{VAR}{Z^2}$$ chebyshev inequality

This calculation is more tricky and arbitrary. We are going to use one important property of variance. Remember that Y is the sum of independent random variables. So you want to use this one to derive a bound. If variance can be expressed in terms of F_k , then things maybe canceled out to some constants, so we need to get here first. Use the telescoping rule, so things can canceled

out nicely.

$$VAR[a\sum X] = a^2 \sum VAR(X)$$

$$VAR(X) = \frac{m^2}{m}[(m_i{}^k - (m_i - 1)^k)^2 + ((m_i - 1)^k - (m_i - 2)^k)^2 + \ldots]$$
$$\leq \sum \frac{m^2}{m} \cdot k \cdot m_i{}^{k-1} \cdot m_i{}^k$$
$$\leq k \cdot F_1 \cdot F_{2k-1} \leq k \cdot n^{1-1/k} \cdot F_K{}^2$$

Constants will be squared, and the random variables are independent. I will give a distribution later, that is 2-wise independent, but not n -wise independent. We use the same kind of grouping.

The constant is m^2 . You select the first occurrence of the value i , $(m_i{}^k - (m_i - 1)^k)$

Probability times the square of the element.

$$VAR(Y) = \frac{1}{s^2}\sum_{q=1}^{s_1} VAR(X_q)$$

B. Approximating F_k

For any numbers $a > b > 0$:

$$a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \ldots + b^{k-1}) \leq (a-b)ka^{k-1}$$

We want to find a upper bound of this. And we want to find a good upper bound for variance. We want that upper bound to have a simpler quantity, somewhat manageable quantity. Suppose a and b are very close to each other.

$$VAR(X) = \frac{m^2}{m}[(m_i{}^k - (m_i - 1)^k)^2 + ((m_i - 1)^k - (m_i - 2)^k)^2 + \ldots]$$
$$\leq \sum \frac{m^2}{m} \cdot k \cdot m_i{}^{k-1} \cdot m_i{}^k$$
$$\leq k \cdot F_1 \cdot F_{2k-1} \leq k \cdot n^{1-1/k} \cdot F_K{}^2$$

Why is this more manageable? Expressing the variance in terms of frequency moment turns out to be in the structure of products. The ultimate goal is to show that the variance is no more than that.

Express upper bound variance in turns of F_k . So the deviation is measured in turns of the mean.

Telescoping Principle

So you express unknowns by other unknowns, so things can canceled out nicely. So the sum may only be a constant. Of course whether this is a constant or not depends on i , but you can always

replace these terms by upper bound. So therefore, they all appear same times. Select the largest possible a. Since a and b are very close to each other, so why you don't select the larger one of a and b to replace some of the terms to construct an upper bound?

C. Lower bounds

The paper obtains rather tight bounds for the minimum possible memory required to approximate the numbers F_k. They proved that for every $k > 0$, F_k can be approximated randomly using at most $O(n^{1-1/k} \log n)$ memory bits. The authors further show that for $k \geq 6$, any (randomized) approximation algorithm for F_k requires at least $\Omega(n^{1-5/k})$ memory bits and any randomized approximating algorithm for F_∞^* requires $\Omega(n)$ space. Surprisingly, F_2 can be approximated (randomly) using only $O(\log n)$ memory bits.

The space complexity of approximating F_k

Theorem 3.2 For any fixed $k > 5$ and $\gamma < \frac{1}{2}$, any randomized algorithm that outputs, given an input sequence A of at most n elements of $N = \{1,2,...,n\}$, a number Z_k such that $\text{Prob}(|Z_k - F_k|) > 0.1F_k) < \gamma$ uses at least $\Omega(n^{1-5/k})$ memory bits.

Tight lower bounds for the approximat¡on of F_0, F_1, F_2 *[1]*

Chernoff bound is typically tighter than Markov's inequality and Chebyshev bounds.

Logarithmic memory suffices to approximate randomly the frequency moments F_0, F_1 and F_2 of a sequence A of at most m terms up to a constant factor with some fixed small error probability. More precisely, $O(\log \log m)$ bits suffice for approximating F_1, $O(\log n)$ bits suffice for estimating F_0 and $O(\log n + \log \log m)$ bits suffice for approximating F_2.

Proposition 3.9

Let A be a sequence of at most n elements of $N = \{1,2,...,n\}$,

(i) Any randomized algorithm for approximating F_0 up to an additive error of $0.1F_0$ with probability at least 3/4 must use at least $\Omega(\log n)$ memory bits.

(ii) Any randomized algorithm for approximating F_1 up to an additive error of $0.1F_1$ with probability at least 3/4 must use at least $\Omega(\log\log m)$ memory bits.

(iii) Any randomized algorithm for approximating F_2 up to an additive error of $0.1F_2$ with probability at least 3/4 must use at least $\Omega(\log n + \log\log m)$ memory bits.

Summary of the first Lecture

You first get the frequency, then you get the value of m . The frequencies are computed in parallel. Then you want to express the variance in terms of the mean square. We also want to get an upper bound. Then we use approximations, for any numbers $a > b > 0$:

$a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \ldots + b^{k-1}) \le (a-b)ka^{k-1}$

since a and b are very close to each other, so you can kind of approximate them as the same quantity. So this can be approximated by ka^{k-1} , assuming that a is the largest term. So this is the first important element. You split the square into two factors. Pick one factor, and take the other factor and applying the same value. We linearize this, then we can use telescoping, we apply telescoping on top of them. km^{k-1} , so remember that this is the sum of the frequencies. And m is the first moment ($m = F_1$).

The paper[1] in section 2 they described space-efficient randomized algorithms for approximating the frequency moments. The tools applied include the known explicit constructions of small sample spaces which support a sequence of four-wise independent uniform binary random variables, and the analysis is based on Chebyshev's inequality and a simple application of the Chernoff bound. In section 3 the paper present lower bounds which are mostly based on techniques from communication complexity. The final section 4 contains some concluding remarks and open problems.

Lecture 2

I. General Algorithm

We have to use low space, and we want the space to be logarithmic small. From last lecture, we have:

$\forall i \in N$, frequency for any i is m_i, $F_k = \sum m_i^{\,k}$,

$E(x) = F_k$ -- $poly(\log m + \log n)$ space requirement

$s_1 = 8kn^{1-\frac{1}{k}}$ (this is not logarithmic space)

$s_2 = 2\lg(1/\varepsilon)$, ($s_2$ is extremely small)

Theorem 2.1[1] For every $k \geq 1$, every $\lambda > 0$ and every $\varepsilon > 0$ there exists a randomized algorithm that computes, given a sequence $A = (a_{1,\ldots,} a_m)$ of members of $N = \{1,2,\ldots,N\}$, in one pass and using $O(\frac{k\log(1/\varepsilon)}{\lambda^2} n^{1-1/k}(\log n + \log m))$ memory bits, a number Y so that the probability that Y deviates from F_k by more than λF_k is at most ε.

Proof of Theorem 2.1. Define $s_1 = \frac{8kn^{1-1/k}}{\lambda^2}$ and $s_2 = 2\log(1/\varepsilon)$. (To simplify the presentation we omit, from now on, all floor and ceiling signs whenever these are not essential). We first assume the length of the sequence m is known in advance, and then comment on the required modifications if this is not the case.

The algorithm computes s_2 random variables $Y_1, Y_2, \ldots, Y_{s_2}$ and outputs their median Y. Each Y_i is the average of s_1 random variables X_{ij} : $1 \leq j \leq s_1$, where the X_{ij} are independent, identically distributed random variables. Each of the variables $X = X_{ij}$ is computed from the sequence in the same way, using $O(\log n + \log m)$ memory bits, as follows. Choose a random number a_p of the sequence A, where the index p is chosen randomly and uniformly among the numbers 1,2,...,m. Suppose that $a_p = l (\in N = \{1,2,\ldots,n\}.)$ Let $r = \left|\{q : q \geq p, a_q = l\}\right| (\geq 1)$ be the number of occurrences of l among the members of the sequence A following a_p (inclusive),

and define $X = m(r^k - (r-1)^k)$.

Note that in order to compute X we only need to maintain the $\log n$ bits representing

$a_p = l$ and the $\log m$ bits representing the number of occurrences of l.

The expected value $E(X)$ of X, by definition,

$$E(X) = \frac{m}{m}[(1^k + (2^k - 1^k) + \ldots + (m_1{}^k - (m-1)^k)) +$$
$$(1^k + (2^k - 1^k) + \ldots + (m_2{}^k - (m_2 - 1)^k)) + \ldots +$$
$$(1^k + (2^k - 1^k) + \ldots + (m_n{}^k - (m_n - 1)^k))]$$
$$= \sum_{i=1}^{n} m_i{}^k = F_k$$

To estimate the variance $Var(X) = E(X^2) - (E(X))^2$ of X we bound $E(X^2)$;

$$E(X^2) = \frac{m^2}{m}[(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_1{}^k - (m_1 - 1)^k)^2) +$$
$$(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_2{}^k - (m_2 - 1)^k)^2) + \ldots +$$
$$(1^{2k} + (2^k - 1^k)^2 + \ldots + (m_n{}^k - (m_n - 1)^k)^2)]$$
$$\leq m[(k1^{2k-1} + k2^{k-1}(2^k - 1^k) + \ldots + km_1^{k-1}(m_1{}^k - (m_1 - 1)^k)) +$$
$$(k1^{2k-1} + k2^{k-1}(2^k - 1^k) + \ldots + km_2^{k-1}(m_2{}^k - (m_2 - 1)^k)) + \ldots +$$
$$(k1^{2k-1} + k2^{k-1}(2^k - 1^k) + \ldots + km_n^{k-1}(m_n{}^k - (m_n - 1)^k))] \quad (1)$$
$$\leq m[km_1^{2k-1} + km_2^{2k-1} + \ldots + km_n^{2k-1} = kmF_{2k-1} = kF_1F_{2k-1}$$

Where (1) is obtained from the following inequality which hold for any numbers $a > b > 0$:

$$a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \ldots + b^{k-1}) \leq (a-b)ka^{k-1}$$

So the general algorithm is simple, given a sequence

$$a_1, \ldots a_m, a_i \in \{1,2,\ldots,n\} = N$$

$$a_i \in \{1,2,\ldots,n\} = N$$

Step 1. Select an index $i \underset{u,r}{\in} \{1,2,\ldots,m\}$ uniformly and randomly without even knowing the value of the element. Staring from the i-th value and count the occurrence r of that value.

Step 2. $r = \left|\left\{j \middle| j \geq i, a_j = a_i\right\}\right|$, so this is a basic random variable, then we have to amplify this

one,

Step 3. $X = (r^k - (r-1)^k) \cdot m$ works for $k \geq 1$ then we do s_1 random variables like this.

Step 4. Let's define $Y_l = \sum_{j=1}^{s_1} X_j / s_1$, (take the average of s_1 independent copies of X_j, this will boost the variance and give us the right kind of standard deviation, also, Y will be more close to expectations)

Step 5. Compute $\{Y_l | 1 \leq j \leq s_2\}$

Step 6. Output median of $\{Y_j\}$

$X = \sum_{i=1}^{n} \varepsilon_i m_i$, $\{\varepsilon_i\}_{1 \leq i \leq n}$ $\varepsilon = n^{-2}$

$E(X^2) = \sum_i \frac{m}{m^2}[((m_i^k - (m_i - 1)^k)^2 +$ --quadratic

$((m_i - 1)^k - (m_i - 2)^k)^2 + \ldots$

$= \sum_i m[km_i^{k-1}(m_i^k - (m_i - 1)^k +$

$k(m_i - 1)^k (m_i - 1)^k ((m_i - 1)^k - (m_i - 2)^k) + \ldots$ --linearization

$\leq \sum_i mkm_i^{k-1} \cdot m_i^k = km \sum_i m_i^{2k-1} \leq km \cdot F_{2k-1} \leq kn^{1-\frac{1}{k}} F_k^2$

$E(X^2) = F_2 = E\left[\left(\sum_i \varepsilon_i m_i\right)^2\right] = E\left[\sum {m_i}^2 + 2\sum_{i<j} \varepsilon_i \varepsilon_j m_i m_j\right]$

$= \sum {m_i}^2 + \sum_{i<j} 2m_i m_j E(\varepsilon_i \varepsilon_j)$ --sign of i-th value is -1

$= \sum {m_i}^2 = F_2$

$VAR(X) \leq E(X^4) - (E(X^2))^2 = E((\sum_I \varepsilon_i m_i)^4) - {F_2}^2$

$= \sum_i {m_i}^4 + 6\sum_{i<j} {m_i}^2 {m_j}^2 + \sum_{i<j} \varepsilon_i \varepsilon_j {m_i}^3 m_j + \sum_{i<j<l<t} m_i m_j m_l m_t \varepsilon_i \varepsilon_j \varepsilon_l \varepsilon_t$ the variance is not too large

$VAR(X) \leq kn^{1-1/k} {F_k}^2$

$\text{Prob}[|Y - \mu| \geq t\sigma] \leq \frac{1}{t^2}$, $e^{-\frac{t^2}{2}}$ t is a number of standard deviation.

$$\text{Prob}\left[\left|Y - F_k\right| \geq \lambda F_k\right] \leq \frac{VAR(Y)}{\lambda^2 {F_k}^2} = \frac{\frac{s_1}{{s_1}^2} VAR(X)}{\lambda^2 {F_k}^2} = \frac{VAR(X)}{\frac{8kn^{1-1/k}}{\lambda^2} \cdot \lambda^2 {F_k}^2}$$

$$= \frac{VAR(X)}{8kn^{1-1/k} \cdot F_k}$$

$$VAR(Y) \leq E(Y^2) = E(Y^2) - (E(Y))^2$$

$$(E(Y))^2 = {F_k}^2$$

By Chebyshev inequality (F_k --mean)

$$\text{Prob}\left[\left|Y - F_k\right| \geq \lambda F_k\right] \leq \frac{VAR(Y)}{\lambda^2 {F_k}^2} = \frac{\frac{s_1}{{s_1}^2} VAR(X)}{\lambda^2 {F_k}^2} = \frac{VAR(X)}{\frac{8kn^{1-1/k}}{\lambda^2} \cdot \lambda^2 {F_k}^2}$$

$$= \frac{VAR(X)}{8kn^{1-1/k} \cdot F_k}$$

For every fixed i,

$$Var(Y_i) = Var(X)/s_1 \leq E(X^2)/s_1 \leq kF_1F_{2k-1}/s_1 \leq kn^{1-1/k}{F_k}^2/s_1,$$

Whereas

$$E(Y_i) = E(X) = F_k.$$

Therefore, by Chebyshev's Inequality and by the definition of s_1, for every fixed i,

$$\text{Prob}[|\ Y_i - F_k\ | > \lambda F_k] \leq \frac{Var(Y_i)}{\lambda^2 {F_k}^2} \leq \frac{1}{8}$$

$$(1-\lambda)F_k \leq output \leq (1+\lambda)F_k.$$

II. Bernoulli trials:

p is the probability of success. Let's do this experiment s times. Basically all the times are independent. So let's define a random variable u .So what is the expected number of success? Success means that the value of Y is in this range. What we are interested in this experiment is what's the probability that the number of success is less than half. Later we can show that the

probability of this is at least $\frac{7}{8}$, so you do the experiment many times, remember that more than half are good events, by taking the median, you know that the value is within this range. So you get a bunch of values, some are in this range, some are outside this range, but more than half are inside of this range. You take the median, get a value, that is in the range. So the bad event is that p is less than half.

p= probability success $p > \frac{1}{2}$, $p = \frac{1}{2} + \delta$,

$$u\begin{cases}1: success\\0: otherwise\end{cases}$$

$$\text{Prob}[T \le \frac{s}{2}] \le p[\left|(T - \frac{s}{2}\right| \ge \delta s] \le 2l^{-\delta^2 s/2}$$

$$T = \sum_{i=1}^{S} u_i$$

Since these are independent random trials, the expectation is:

$$E(T) = sp \ge \frac{s}{2} + \delta s$$

Let $Z = c\sum_{i=1}^{s} T_i$

$$VAR(Z) = c^2 \sum VAR(T_i) \le \frac{1}{8}$$

For example:

So you can throw n face dice , which means you have random number with index from 1 to n, When you see a new element, which you didn't count before, you must change the probability of the new element and also the probability of the previous element.

You don't know how many numbers will come, so you see the first element, then you have to make a decision--select the element or not. Let's say you selected it. Then you have to see the next element, you may want to discard that and then select the next new element, you have to change the probability of the 2nd element, the probability of the previous element must change too. You don't know the value of the random numbers, when you see a new element which you didn't count before(you discard before), with probability half you throw it away. Let's take 3 elements for example and then expand the experiment to n elements.

Suppose a_n are n ($n \ge 1$, $n \in$ integers) face dices, so a_1 has 1 face, a_2 has 2 faces, ... a_n has n faces, etc. We have the probability list:

a_1	a_2	a_3	...	a_n
1				
$1/2$	$1/2$			
$1/3$	$1/3$	$1/3$		
$1/n$	$1/n$	$1/n$	...	$1/n$

We select the new element a_3 with probability $\frac{1}{3}$, and discard the previous probability $\frac{2}{3}$, so as the number of elements increases, the chance of a certain element to be selected goes down. So you see the 4th element you can select the element with probability $\frac{1}{4}$. Applying the same principle, we can expand the table to a_n, with element probability $\frac{1}{n}$.

References

[1]The space complexity of approximating the frequency moments, Noga Alon, Yossi Matias & Mario Szegody, STOC'96, Philadelphia PA, USA

[2]Cauchy-Schwarz Inequality, http://en.wikipedia.org/wiki/Cauchy%E2%80%93Schwarz_inequality

[3]Jensen's inequality, http://en.wikipedia.org/wiki/Jensen's_inequality

[4]Telescoping series, http://en.wikipedia.org/wiki/Telescoping_series

[5]Central Limit Theorem, http://en.wikipedia.org/wiki/Central_limit_theorem

[6]Chernoff Bounds, Chapter 4, Michael Mitzenmacher and Eli Upfal, Probability and Computing.