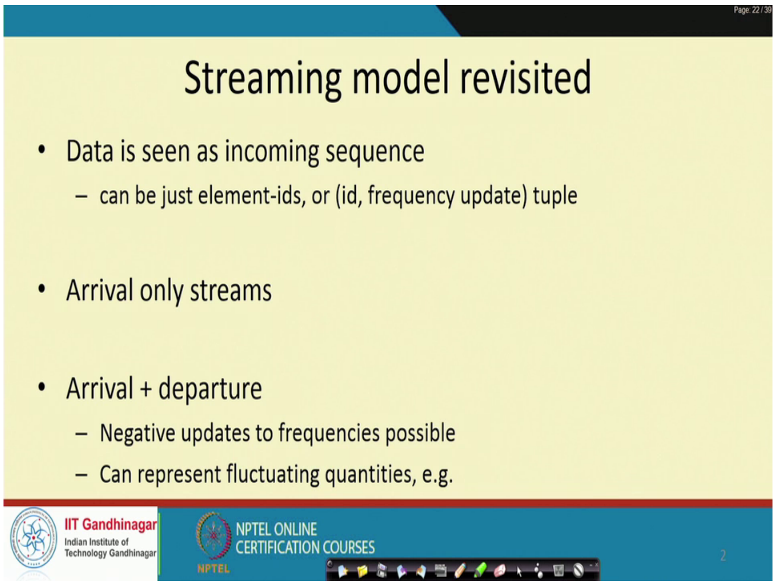
Scalable Data Science

Prof. Anirban Dasgupta
Department of Computer Science and Engineering
Indian Institute of Technology, Gandhinagar

Lecture - 10 b
Frequent Elements: Count Sketch

Welcome to today's lecture of Scalable Data Science. Today will talk about Frequent Elements again and will see at different data structure today one called Count Sketch. I am Anirban and I teach Computer Science and Engineering at IIT Gandhinagar.

(Refer Slide Time: 00:33)



Page: 22 / 39

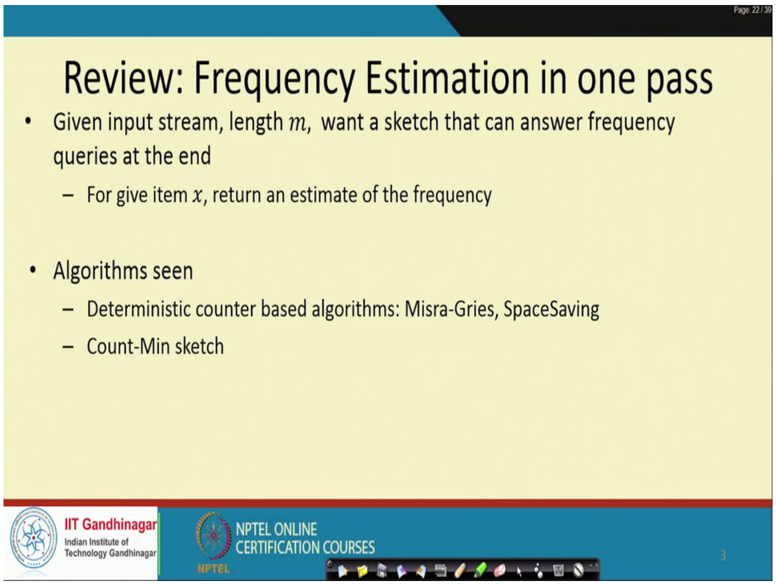
Streaming model revisited

- Data is seen as incoming sequence
  - can be just element-ids, or (id, frequency update) tuple
- Arrival only streams
- Arrival + departure
  - Negative updates to frequencies possible
  - Can represent fluctuating quantities, e.g.

IIT Gandhinagar Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES
2

So, we do not need much revisiting of distributing model, but again we are dealing with both arrival only, and arrival departure streams, and we are looking at the questions of frequency estimation in one pass.

(Refer Slide Time: 00:48)



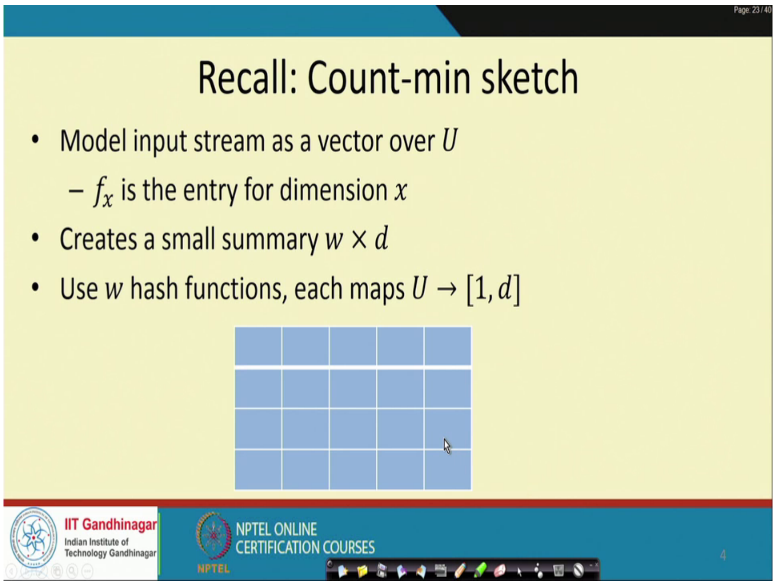
Review: Frequency Estimation in one pass

- Given input stream, length m, want a sketch that can answer frequency queries at the end
  - For give item x, return an estimate of the frequency
- Algorithms seen
  - Deterministic counter based algorithms: Misra-Gries, SpaceSaving
  - Count-Min sketch

We are seen 2 algorithms until now; I mean 3 algorithms until now actually., We are seen to deterministic counter based algorithms, the Misra Gries, and the SpaceSaving right. That work only in the only in the arrival only setting we have seen the count-min sketch and that is have that is a proper sketch that works in the in the arrival only setting. And the slide variant of that let me call the count median sketch right where we say that we could take the median when we written in estimate, and this works in the arrival plus deletion.

Although we did not quite see a proof of this it is ready similar to that of the count-min sketch.

(Refer Slide Time: 01:28)



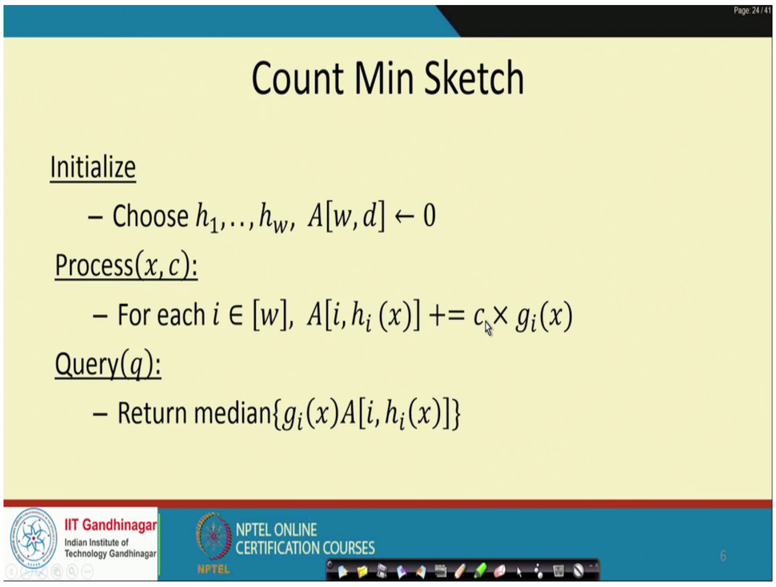
Recall: Count-min sketch

- Model input stream as a vector over U
  - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, each maps $U \rightarrow [1, d]$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, I mean before we go on to our new algorithm let us recall with the count-min sketch was, right. So, here we are using here we are using 2 parameters: one name w and the other name d right; w are the different w hash functions and d is the range of each of these hash functions. And then what the count-min sketch should do is that it given a particular update it would look at each the where each of the hash functions places; that particular places that element x go to the corresponding cell and update the corresponding cell that is all.

(Refer Slide Time: 02:30)



Count Min Sketch

Initialize

- Choose $h_1, .., h_w$, $A[w, d] \leftarrow 0$

Process(x, c):

- For each $i \in [w]$, $A[i, h_i(x)] += c \times g_i(x)$

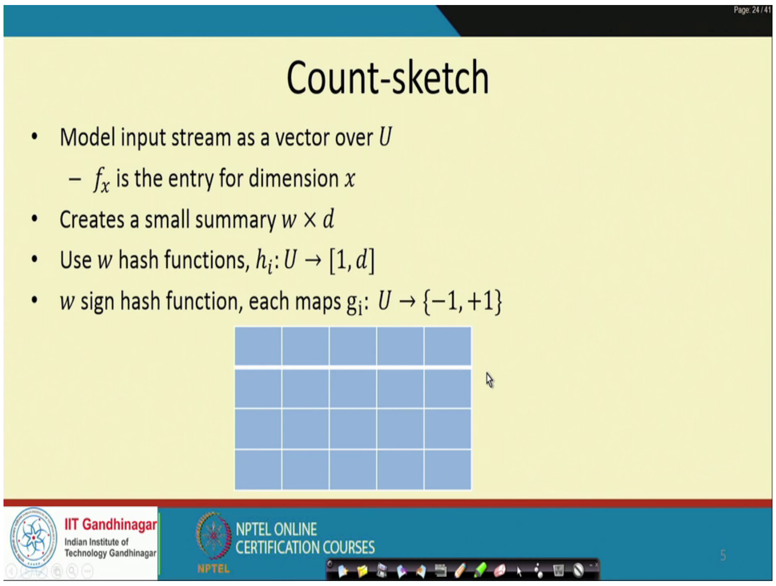
Query(q):

- Return median$\{g_i(x) A[i, h_i(x)]\}$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, the count sketch is actually something that is very similar right.

(Refer Slide Time: 02:40)



Count-sketch

- Model input stream as a vector over U
  - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, $h_i: U \to [1, d]$
- w sign hash function, each maps g_i: $U \to \{-1, +1\}$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, we have the same kind of data structure we have a small summary w by d right. There are w hash functions right h 1 to h w each of them map to the range each of them map to the range 1 to d, those also something additional right there are w sign hash functions right g 1 to g w.

Now, each of g 1 to g w gives you a random sign for each element of the universe. So, think of it as flipping a coin for each element of the universe if the coin comes out heads you give it your sign it as sign plus 1. If the coin comes out tails as a inter sign minus 1 and of course, the hash function h 1 h i implicitly sort of assign I mean given any particular element with a sign set to one of the positions uniformly at random right with probability 1 over d.

And again what we what will point out today we should did not I guess to last time is a fact that, while we do the analysis for in random completely random hash functions we could as well have used k wise independent hash functions, for actually k to be equal to log 1 over delta that is all.

In fact in fact here will need a smaller value of k, so let me erase this here we actually need k to be 4 wise independent that will actually be enough. So, let see this is actually the count sketch. So we do the initialization as expected right.

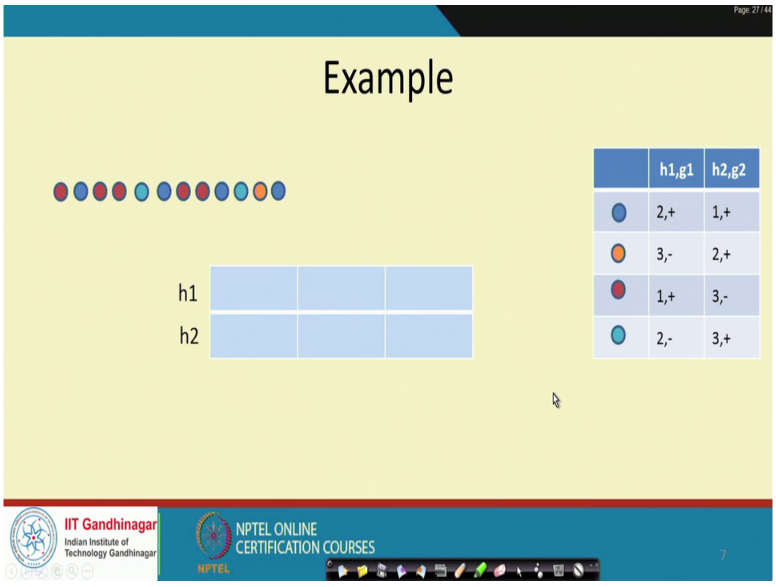
We initialize, we choose we choose h 1 to h w we also choose g 1 to g w and then we initialize the array a to be all 0. For any item and it is update x and I the update c which is one element of the stream, so here is what we do now right.

You look at for each of the hash functions you look at where that hash function is spacing that item you go to that cell. Now instead of updating it by c, you multiply c by the sign g i of x right. So, this becomes either plus c or it becomes minus c right and you do the update of that bucket with this sign right.

So, remember that each element gets only one sign right, the next time element comes again give the same sign because this is the hash function. This is not a any random variable that we are sort of conjuring up every time a new copy of the item comes. So, this is a hash function that gives assign the sign to the item and the always do a plus c or we always do a minus c if the if the update is c for that particular item. So, now what do we do when we are trying answer the query right. So, we look at now again we have w estimates right 1 to w i equal 1 to w.

So, an each of these estimates is now g i of x multiplied with a h i A i comma h i of x. So, why are you multiplying pre multiplying with g i of x that is really obvious right, because if g i of x was minus 1 you want to get away from the get rid of the minus, so you again multiply it with g i of x so that is fine right. But now, we are taking the median of these, estimate instead of taking that average right and will see that that is useful.

(Refer Slide Time: 06:20)


Page: 27 / 44
h1
h2
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
7

Example

	h1,g1	h2,g2
🔵	2,+	1,+
🟠	3,-	2,+
🔴	1,+	3,-
●	2,-	3,+

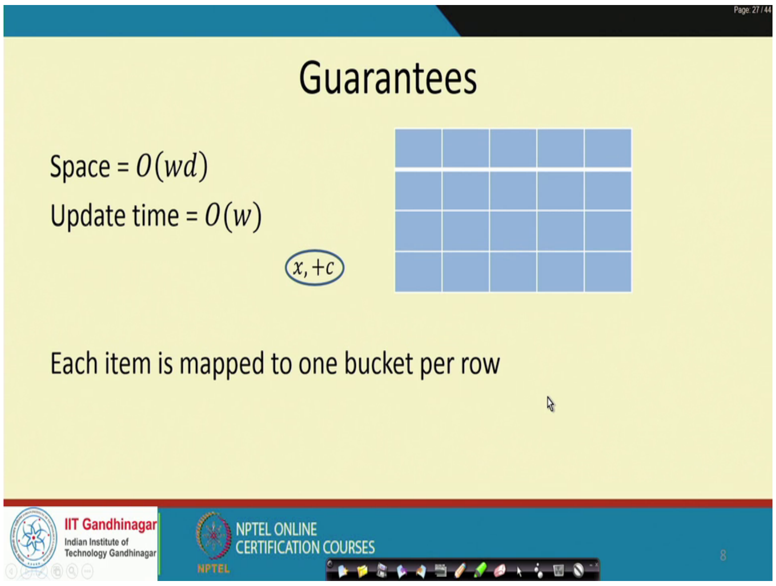
So, as is our tradition let us run then a small example. So, we have this stream and we have encoded the hash table the hash functions in this format. So, what it means is that h 1 maps blue to the position 2 and g 1 maps blue to the sign plus, h 1 maps yellow to position 3 and g 1 maps yellow to sign minus and so on right. So, the tuple denotes the first element of the tuple is the position to which h 1 is mapping the particular element. The next sign the next element of the tuple is a sign that says that what is the sign that g that the corresponding g function is assigning to this element.

So, let us number the positions 1 2 3, so first blue comes. So, first red comes red goes to 1 with plus and red goes to 3 with minus, blue comes blue goes to 2 with plus and blue goes to 1 with plus. Then 2 reds come 2 reds go to 1 with plus and here with minus, turquoise comes turquoise go to 2 with minus turquoise goes to 3 with a plus. Blue goes to 2 with a plus 1 with a plus, 2 reds come again go to 1 with plus and 3 with minus, 1 blue comes goes to 2 with plus goes to 1 with plus. Turquoise goes to 2 with minus goes to 3 with plus yellow goes to 3 with minus goes to 2 with plus, blue goes to 2 with plus goes to 1 with sorry blue goes to 2 with 2 with plus, blue goes to 1 with plus.

So, therefore in the in the cell 1 right here the total count is 5 here the total count is here the total count is 1 2 3 4 minus 2 2, here the total count is 1 2 3 5 minus 2 3; here the total count is minus 1 here the total count is plus 1 here the total count is 3 minus 3 3 minus 1 2. So, that is why the data structures is going to look like and the and how you answer the query is really obvious that suppose you are queried with w then in that case you go to you go to position 3 you multiplied with minus.

So, that becomes minus 1 times minus 1 times 1 and then you go to position and then you go to position 2 of the of the hash bucket so that is 1 and then you multiply that with the with the plus, because g is plus s so plus 1. So, median of 1 and 1 and so we get a 1 here. So, at least w, I mean for yellow it is answered correctly which is of course, on the case for all the other elements.

(Refer Slide Time: 10:03)



Guarantees

Space = $O(wd)$

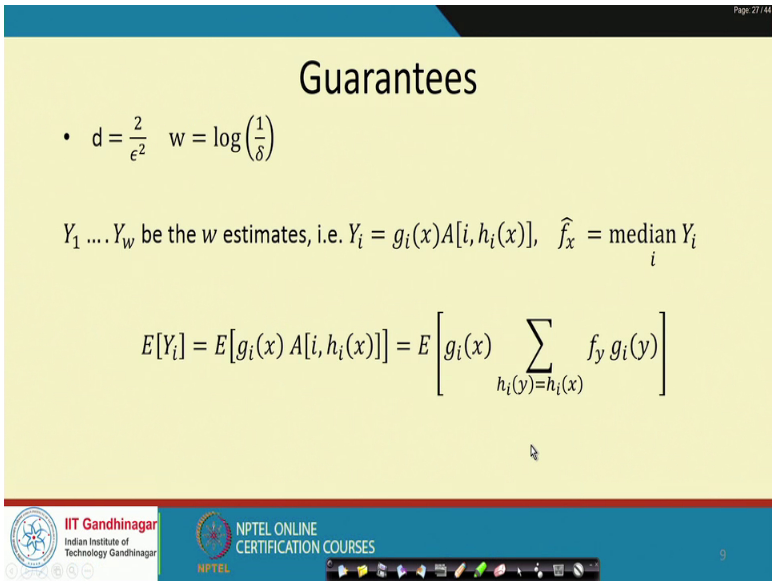
Update time = $O(w)$

$x, +c$

Each item is mapped to one bucket per row

So, again the same kind of guarantees for the space as well as the update time right, because each item is mapped to one bucket per rows it is not a surprise that; that you get the same kind of guarantees the.

(Refer Slide Time: 10:16)



Guarantees

- $d = \frac{2}{\epsilon^2}$ $w = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = g_i(x)A[i, h_i(x)]$, $\hat{f}_x = \underset{i}{\text{median}}\, Y_i$

$$E[Y_i] = E\big[g_i(x)\, A[i, h_i(x)]\big] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

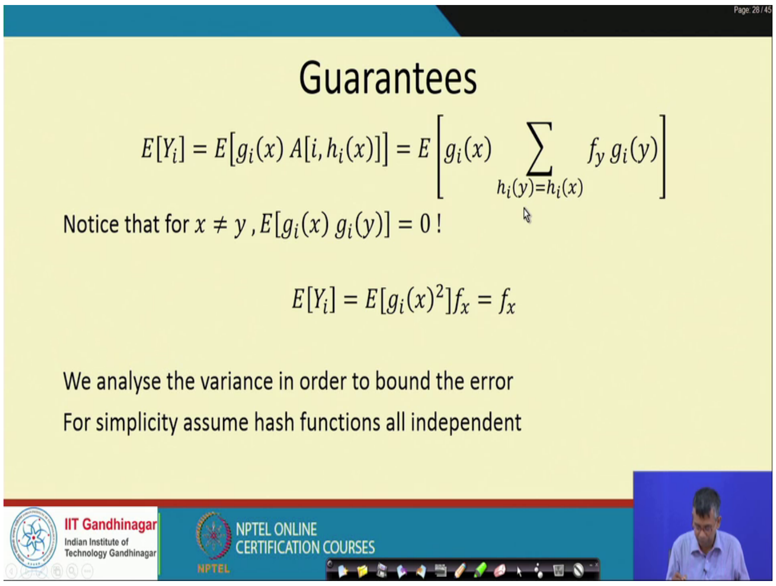
So, the question is that what is what is d and what is w right and again following the principle that we had that in order to set the values of d and w first you have to do error analysis right. And based on a based on the on the error analysis you want to say that what is the dependence of w and d and the 2 typical error parameters the approximation factor epsilon right, as well as the error probability delta.

So, let us start by setting d equal to 2 by epsilon square and w equal to log 1 over delta right. So, then the w is let us denote y 1 by to y w as a w estimates. So, the i-th estimate is this particular quantity right, it is equal to you look at the i-th hash function, I am not gives you the cell A i h i of x. So, you multiply this cell by the by the sign hash g i of x that gives you the i-th estimate Y i and what are we doing at the end? We returning the median of the i-th estimates. So, let us at least calculate the expectation of each of thees estimates.

So, the expectation Y i is nothing is in exactly this expectation because, that is the formula and now I can expand out the A i h i of x in terms of the sum, what is this sum? Is nothing but you look at all the y that have appeared in the stream right and that and those y's that the hash function puts in this bucket right.

That the i-th hash function puts in this bucket look at exactly those ys. You look at this sign g i of y and then you get a weighted sum with the frequencies right, the summation of f y g i of y where y is exactly that set of ys such that h i of y is equal to this bucket this bucket i d. So, that is the so that is expectation of Y i, so it might look a little complicated now but it is going to simplify itself very soon.

(Refer Slide Time: 12:29)



Guarantees

$$E[Y_i] = E[g_i(x)\, A[i, h_i(x)]] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

Notice that for $x \neq y$, $E[g_i(x)\, g_i(y)] = 0$!

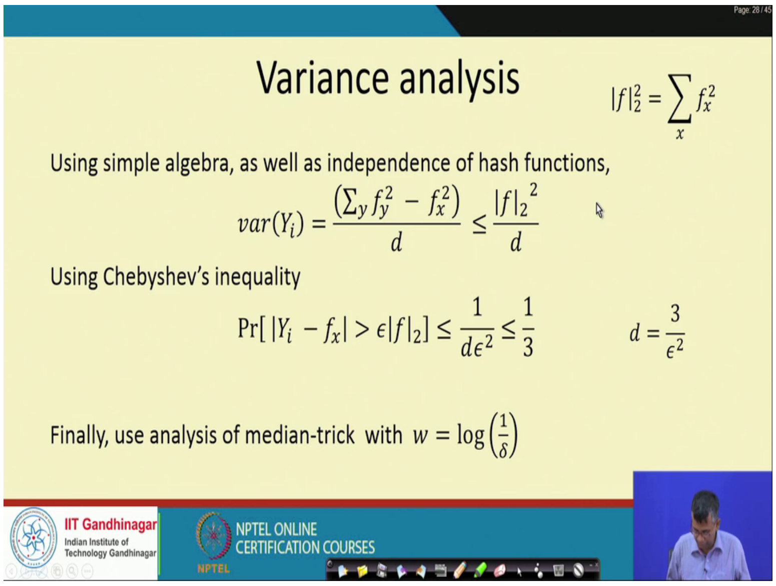
$$E[Y_i] = E[g_i(x)^2] f_x = f_x$$

We analyse the variance in order to bound the error

For simplicity assume hash functions all independent

So, let separate out 2 doubts first of all we know that this sum has the element the sum has the sum has the element x.

(Refer Slide Time: 12:52)



Variance analysis

$$|f|_2^2 = \sum_x f_x^2$$

Using simple algebra, as well as independence of hash functions,

$$var(Y_i) = \frac{\left(\sum_y f_y^2 - f_x^2\right)}{d} \le \frac{|f|_2{}^2}{d}$$

Using Chebyshev's inequality

$$\Pr[\,|Y_i - f_x| > \epsilon|f|_2] \le \frac{1}{d\epsilon^2} \le \frac{1}{3} \qquad d = \frac{3}{\epsilon^2}$$

Finally, use analysis of median-trick with $w = \log\left(\frac{1}{\delta}\right)$

This sum has the element x so let us take out the element x right and if you take out the element x. You will get a term of the sum that looks as follows g i of x square f x plus and now the sum is over x not equal to y and so on.

So, by linearity of expectation let us calculate the expectation of this let us calculate the expectation of the second part let us take this one first. So, what is g i of x? So see f x is a constant there is nothing to worry about that and what is g i of x square? G i of x is only plus 1 and minus 1. So, therefore g i of x square is always 1 right, so therefore the expectation of this part is exactly f x right which is what you have written here so that is nice.

Now, what is the expectation of the second part, of the second part which is the sum over all x not equal to y which is the sum over all x not equal to y right. This quantity sum over x not equal to where as you take the sum over all x not equal to y right, so might's in complicated but again. So just look at this very simple observation, that now that because x not equal to y you have g i of x and you have g i of y right. So, the g is an independent of the h's, the value the g hash functions are independent of the h hash functions. So, therefore right whatever else you have you will have this expression right for each term you will have a expectation of g i of x times g i of y right. But now again g i of x is independent of g i of y and each of them have 0 expectation right.

So, the so the expectation of this is 0 the expectation of this is 0. In fact, the expectation any one of them is 0 and that is enough right, which means that this entire sum has expectation 0 for every y not equal to x right. And so the entire sum has expectation 0 each because, each term has expectation 0 for each y not equal to x. So, this seems somehow it all sort of boiled down to giving me the right expectation right, which was the entire of this of these of these sign hash functions.

But now what we need to do in a is we need to analyze the variance in order to bound the error and this is where the independence of the hash functions will come in. Well, they have also come in they we have also kind of use the independence in the in sort of saying that expectation of g i of x is g i of y, but that is the independence of the different hash functions right.

Now what we are going to says that the fact that whether h; I mean. So, you need to know that the 2 kinds of independence here whether h 1 and h 2 are independent and whether h itself is chosen uniformly at random, I mean completely random hash function right. Now in the in analysis of variance right you need to I mean here will assume that h i is we chosen at random right and there and I am will see that we can relax that to make it only something like 2 wise independent or 2 wise universal.

So, I will not do the entire variance calculation because a lot of it is algebra, but what you can say is that the variance of Y i is going to looked has something like this, you going to have summation y f y square summation of or a minus f x square and right. And we would not would not use this exact form what you can use is a simplifying form, that tells me I mean that goes as follows, that you sort of upper bound this quantity by the 2 norm of the frequency vector f.

So, this is the 2 norm of the frequency vector f and you can upper bound this quantity and you can upper bound the numerator by this quantity and the denominator you get a 1 by d right. You get a 1 by d because I mean in the denominator you get a d, because that is the range of the hash function right.

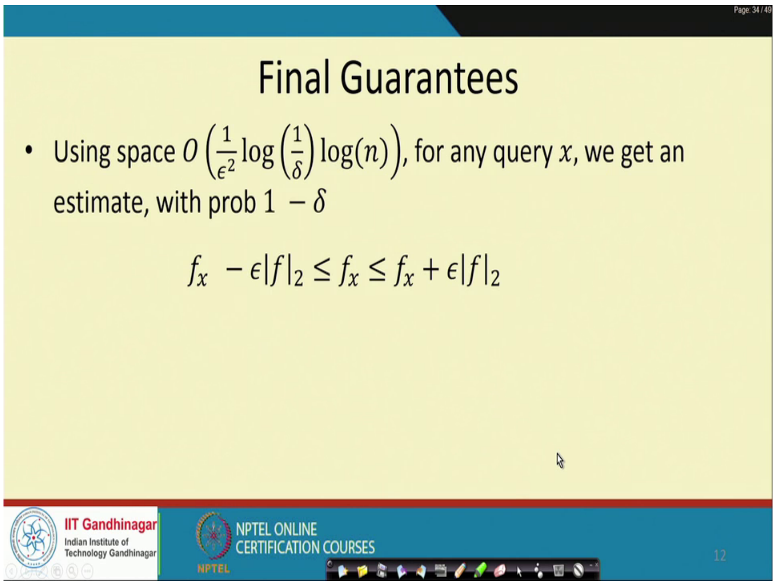
So, implicitly the probability of any collision is 1 by d and once you do I mean this analysis of the variance is really simple I know you just need to boiled down. I mean just need to write down expectation of the expectation of Y i square right and distribute that distribute the square over the sum and so on and will get this hence very simple.

But once you have this bound on the variance now we can apply Chebyshev’s inequality. Because, we know the expectation of the random variable is exactly f x which is what we want, we know the you know a bound on the variance and let us apply Chebyshev’s inequality which is what will say that the property that Y i minus f x. Absolute value of that is bigger than epsilon times the 2 norm of the random of the of epsilon times the 2 norm of the of the frequency vector is less than equal to the variance divided by epsilon square 2 norm square right and we have bound the variance as f 2 norm square by d. So, therefore you get a f 2 norm square by d divided by epsilon square f 2 norm square. So, therefore the f 2 norm squares cancels out and then you get a 1 by d epsilon square in the in the in the probability bound.

So, now we can set I mean remember our choice of d right. So, in reality you would choose d only after seeing this expression. When you see this expression and then you say that it makes I want to make this probability less than one third. So, let me go ahead and choose d equal to 3 by epsilon square, that may that says that each Y i lies within f x plus minus epsilon times the 2 norm of the of the frequency vector with probability one third right.

But what are we doing? We taking the median of these of these of these Y ’s right and now what you can use is the is the same median trick that is say median trick that we are being doing before, and that will tell you that the. And that will tell you that the probability that that the probability that that the median lies within the probability of the median, does not lie within this f x plus minus epsilon.

(Refer Slide Time: 20:01)



Final Guarantees

- Using space $O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\log(n)\right)$, for any query x, we get an estimate, with prob $1 - \delta$

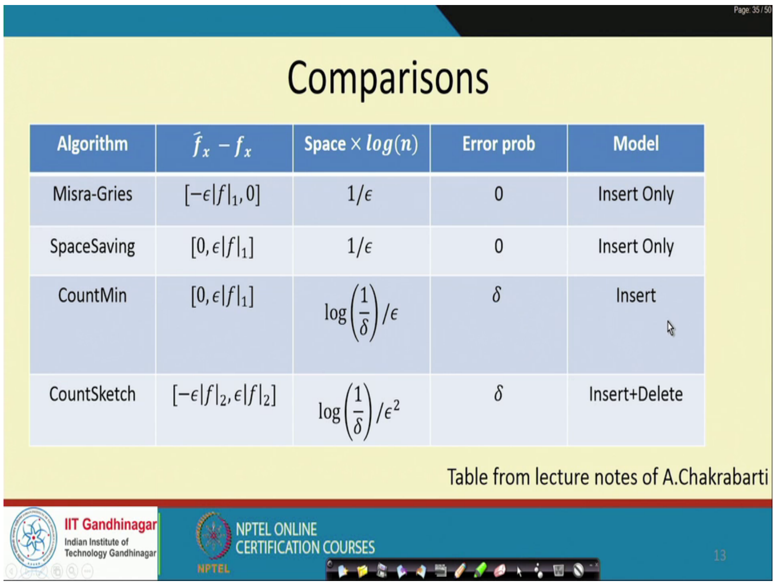
$$f_x - \epsilon|f|_2 \leq f_x \leq f_x + \epsilon|f|_2$$

I mean so f x this is epsilon times f 2 norm, this is the probability that the median does not lie does not lie in this interval is less than delta right. Which means that the probability 1 minus delta right for any query for any fixed query x we get an estimate with this probability, we get an estimate that lies within f x minus epsilon times the 2 norm of f and f x plus epsilon times the 2 norm of f.

So, couple of deleted things I would like to point out is that, this probability holds for a fixed query and this is always the case when we are dealing with these when we are making this statement right. If you have to answer for all queries you have to take a union bound right, which is not for they we doing here we are answering for a fixed query x right.

If you have to answer 10 queries you have to take a union bound over 10 queries and what is the space that we need? See the size of a data structure is 1 over epsilon square log 1 over delta and each in each of the position. So, storing a counter it is 1 over epsilon square log 1 over delta times log n, so that is the data structure that we have.

(Refer Slide Time: 21:16)



Comparisons

Algorithm	$\hat{f}_x - f_x$	Space $\times$ $log(n)$	Error prob	Model
Misra-Gries	$[-\epsilon\lvert f\rvert_1, 0]$	$1/\epsilon$	0	Insert Only
SpaceSaving	$[0, \epsilon\lvert f\rvert_1]$	$1/\epsilon$	0	Insert Only
CountMin	$[0, \epsilon\lvert f\rvert_1]$	$\log\left(\frac{1}{\delta}\right)/\epsilon$	δ	Insert
CountSketch	$[-\epsilon\lvert f\rvert_2, \epsilon\lvert f\rvert_2]$	$\log\left(\frac{1}{\delta}\right)/\epsilon^2$	δ	Insert+Delete

Table from lecture notes of A.Chakrabarti

So, by now we have seen at least at least 4 I mean 4 of this algorithms, in fact we have seen a variant of count means I would not talk about that we have I mean.

But we have seen 4 algorithms that that each sort of answer the that each answer the all the question of frequency estimation right and these 4 algorithms are slightly different for each other and therefore, it is its very useful to 2 kind of get a over view comparison of them. So, this epsilon table is from the lecture notes of Amit Chakrabarti, the 4 algorithms that we have seen a Misra Gries space saving count-min and count sketch right.

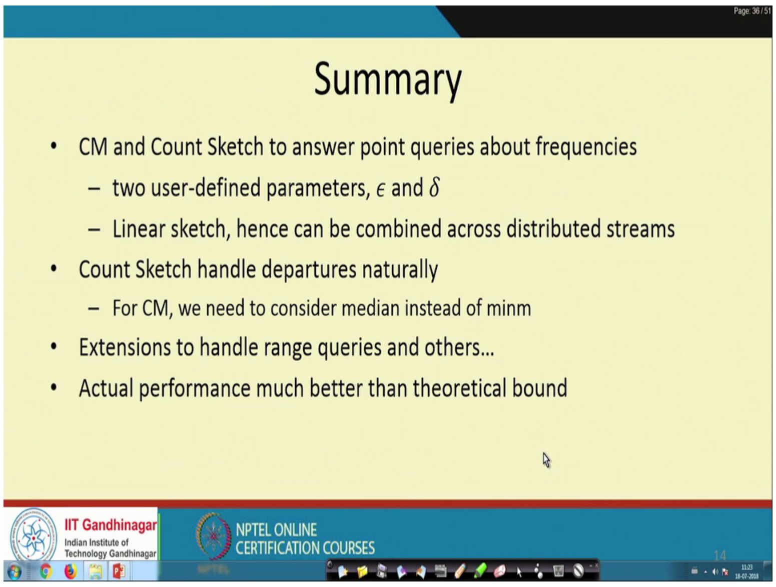
So, the first column contains the bound f hat x minus f x that we have proven right. So, the 2 notations that we have used here is that when we say, when we say f the one norm of f that is nothing, but summation f x right which is basically equal to m the length of the stream right. When we say the 2 norm of f that is equal to the summation f x square for all x right, that is what we just saw the square root of that. So, f 2 norm and see now you start noticing the trade of that we are having. Both Misra Gries and space saving both of them use only 1 by epsilon counters. Let us say I mean actually 2 by epsilon counters both of them are deterministic.

So the excellent, but the only work for insert only streams not for inserts and deletes count-min has sort of has. This thing has a bound that is that is similar to that of space saving right. However, it uses a little more a little more space right it has it is randomized. So, it has some error probability these 2 as 0 error probability 0 because they deterministic right.

However, count-min can also be used to answer some, some more general queries we did not really talk about that and it also works for inserts only. However if I, with to change the count-min to count median right, we will get a similar bound here we would get how it would have been minus epsilon times one norm of f and epsilon times one norm of f and it would work it would work both for inserts and deletes. Count sketch gives you a error bound in terms of the 2 norm of the frequency vector right. Remember that the 2 norm of the frequency vector is actually little bit smaller than the one norm is potentially smaller than the one norm of the frequency vector and but it takes a little more space also.

So, now the space bound is 1 by epsilon square depends on 1 by epsilon square rather than 1 by epsilon for the other algorithms it works it works nicely for both inserts and deletes. So, this is an interesting trade off and it sort of really depends on the on your use case as to which of these algorithms really performs the better test.

(Refer Slide Time: 24:32)



Summary

- CM and Count Sketch to answer point queries about frequencies
  - two user-defined parameters, ϵ and δ
  - Linear sketch, hence can be combined across distributed streams
- Count Sketch handle departures naturally
  - For CM, we need to consider median instead of minm
- Extensions to handle range queries and others…
- Actual performance much better than theoretical bound

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, just to summaries in this in this lecture and in the previous one we talked about count-min, and count sketch to answer point queries about frequencies, they really 2 define use a different parameters both of them are dependent on these epsilon the error parameter and delta the probability bound right.

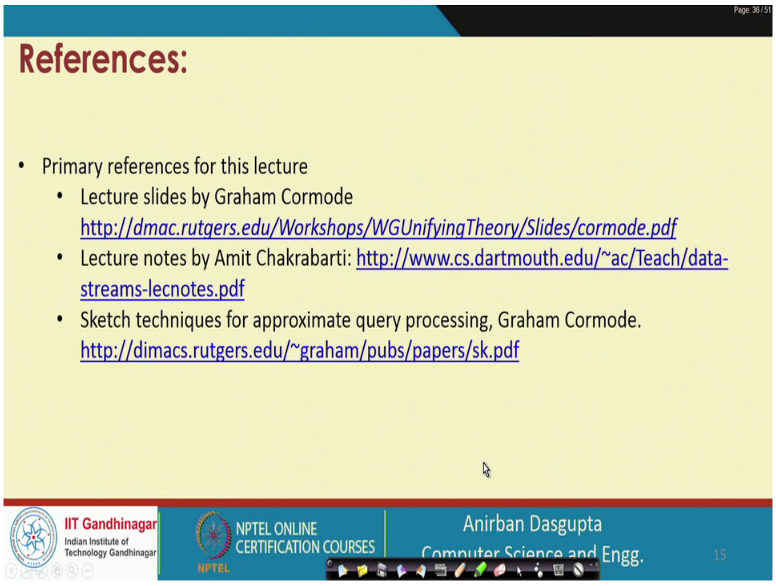
Both of these are linear sketches right and this is very effective because they are now they can be combined across distributed streams. Remember the setting that we are talking about that suppose we have 2 data streams 1 and machine 1. And the other machine 2 and you do not somehow you want to combine the statistics across these 2 streams right. Because, you have these sketches to be linear you can easily combine this statistics and that is not so true, if you have this counter base sketches.

This is the extension this is one of the advantages that count-min gives you over the over let us say space saving and or Misra Gries. Count sketch can handle departures naturally, but for count-min we need to consider median instead of instead minimum.

There are other extensions to handle range queries and others and actual performance of any of their algorithms is much better than the theoretical bound. They have been some efforts to analyze these in terms of saying that if you have power law distribution if the distribution is very skewed.

It is known that and it can be proven that a lot of this algorithms perform a lot better, the hardest instance of this algorithm is when the frequency distribution is more or less flat is it is more or less flat, but there is only a small number of when the heavy it as only slightly bigger than the than the average right. If the heavy it as is really begin much bigger than the average then all of them perform really nicely ok.

(Refer Slide Time: 26:16)



References:

- Primary references for this lecture
  - Lecture slides by Graham Cormode http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf
  - Lecture notes by Amit Chakrabarti: http://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf
  - Sketch techniques for approximate query processing, Graham Cormode. http://dimacs.rutgers.edu/~graham/pubs/papers/sk.pdf

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES
Anirban Dasgupta
Computer Science and Engg.

So, again just the same sort of references for our for this lecture and which are the lecture slides by Graham as well as the lecture notes by Amit and Graham so that is it.

Thank you.