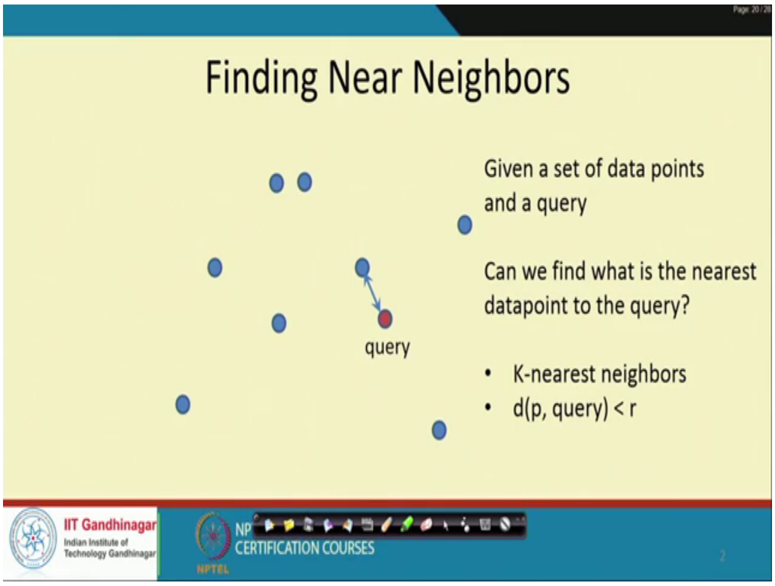
Scalable Data Science
Prof. Anirban Dasgupta
Department of Computer Science and Engineering
Indian Institute of Technology, Gandhinagar

Lecture -12 b
Building LSH Tables

Welcome to the lecture of Scalable Data Science. Today will talk a little more about locality sensitive hashing; in particular will start to see how to build hash tables out of the locality sensitive hash functions that we have been discussing right.

(Refer Slide Time: 00:33)



Finding Near Neighbors

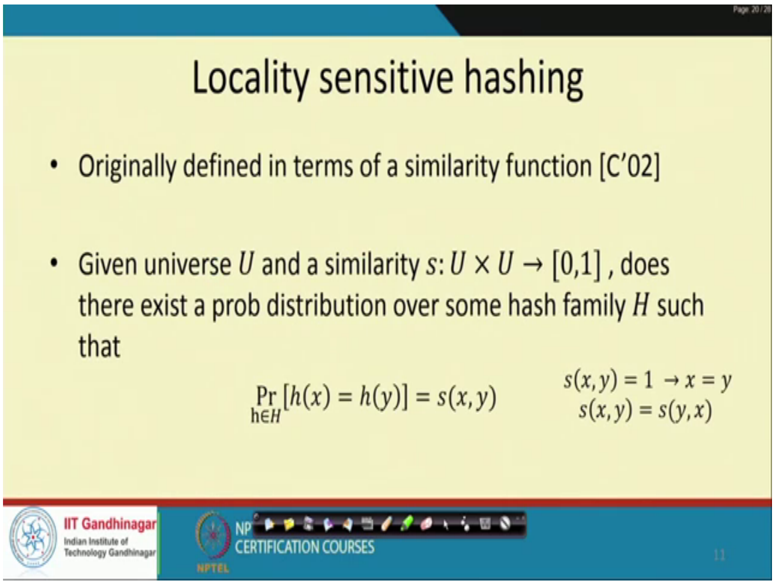
Given a set of data points and a query

Can we find what is the nearest datapoint to the query?

- K-nearest neighbors
- d(p, query) < r

But I start with just to review this was the problem of finding near neighbors, and we are trainable data structures that answer a queries like what are the K-nearest neighbors and what and give me the list of candidate nearest neighbors within the particular distance may be ok.

(Refer Slide Time: 00:49)



Locality sensitive hashing

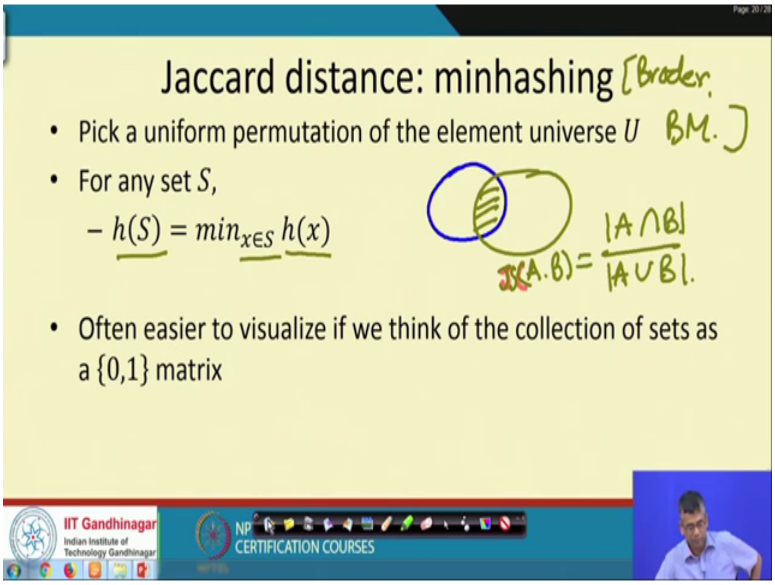
- Originally defined in terms of a similarity function [C'02]
- Given universe U and a similarity $s: U \times U \to [0,1]$, does there exist a prob distribution over some hash family H such that

$$\Pr_{h \in H}[h(x) = h(y)] = s(x,y) \qquad \begin{array}{c} s(x,y) = 1 \to x = y \\ s(x,y) = s(y,x) \end{array}$$

And our the tool that we focused on is are these kinds of hash functions right, but we have say this that suppose we have a similarity function right, and similarity function is something that takes any 2 objects right and intuitively gives us the similarity between these 2 right. And mathematically we can say that it satisfies a few simple properties like a s of x y equal to 1 if and only if x equal to y and s of x y equal to s of y x. So, given such a similarity measured, we never asking the question of does there exist some hash family, h such that we can choose let say uniformly for now let say uniformly from random from this hash family.

And so, that the probability over the choice of h, the probability that a h of x equal to h of y the probability that any 2 x and y collide right that that h puts in the same bucket is equal to the similarity of x and y exactly equal to the similarity. So, and the we were wondering whether such hash families even exist. So, we have seen at least for 2 such families; for instance the angle base similarity as well as the hamming similarity. At least for 2 such similarities we have seen that we have proven the existence of such hash families. In the lecture will investigate a little bit more about some other similarity measures, and then we see how to actually use this families such hash families.

(Refer Slide Time: 02:27)



Jaccard distance: minhashing [Broder, BM.]

- Pick a uniform permutation of the element universe U
- For any set S,
  - $h(S) = min_{x \in S}\, h(x)$
- Often easier to visualize if we think of the collection of sets as a $\{0,1\}$ matrix

$JS(A.B) = \frac{|A \cap B|}{|A \cup B|}.$

So, the first distance the first distance of similarity measure that we are considered is Jaccard distance right. If you remember if you do not remember a Jaccard distance was it was this is very simple idea, that if you have 2 sets A and B right. The Jaccard distance between them it is defined as right as this quantity right J of A B this is the Jaccard similarity, not the Jaccard distance Jaccard similarity between A and B is defined as a intersection B divided by A A union B the cardinality of these 2 right.

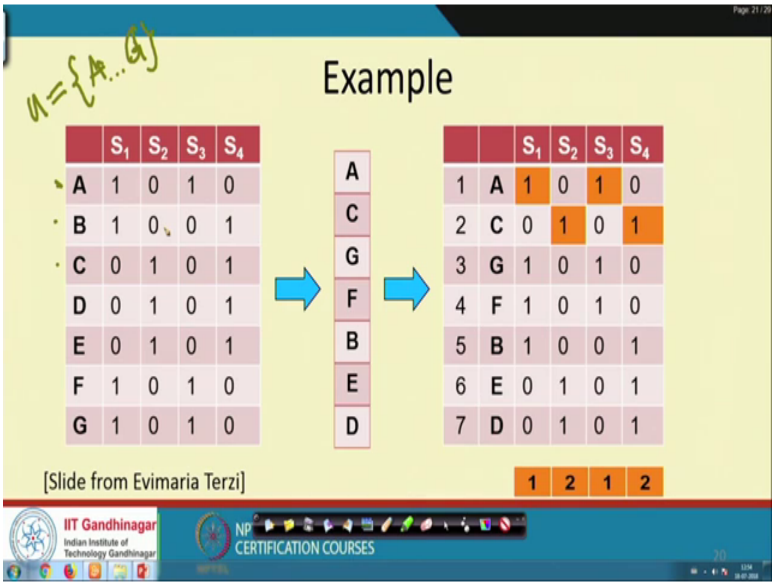
So, and we want and this is a similarity measure right so. So, now, we are asking that does there exist the hash family right that is locality sensitive for this similarity measure right and here is a solution. And the solution was initially suggested by Anre Broder in a beautiful paper and then subsequently worked upon by Broder and Micheal marker. So, the family consider all random permutations of the of the universe U right. So, there are if the universe size is n, then the n factorial random permutations pick anyone of them a random pick any one of them uniformly at random right.

So, my hash family is the set of all random permutations of the universe U and I get to pick any one of them uniformly at random and now I calculate the probability right and now how do I map a set right? I can say that I map each element of the set according to this random permutation right I am use an example right now, I map each element of the set according to this random permutation and take the minimum of these values right.

So, I take the minimum of the elements of the set, the values of h of x ok. So, you can imagine you can set of. So, without loss of generality, we can say that the universe consisted of the numbers from 1 to n right.

So, then taking the minimum is actually critical defined and then nice. So, we consider a random permutation of 1 to n and then take over all elements in the set you calculate the minimum of h of x. So, it is often easier to visualize this if we if you think of the collection of if you multiple sets, if you think of the collection of sets as a 0 1 matrix. So, not sort will do let us let us see an example right.

(Refer Slide Time: 05:24)



u={A...G}

Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

A
C
G
F
B
E
D

		S_1	S_2	S_3	S_4
1	A	1	0	1	0
2	C	0	1	0	1
3	G	1	0	1	0
4	F	1	0	1	0
5	B	1	0	0	1
6	E	0	1	0	1
7	D	0	1	0	1
		1	2	1	2

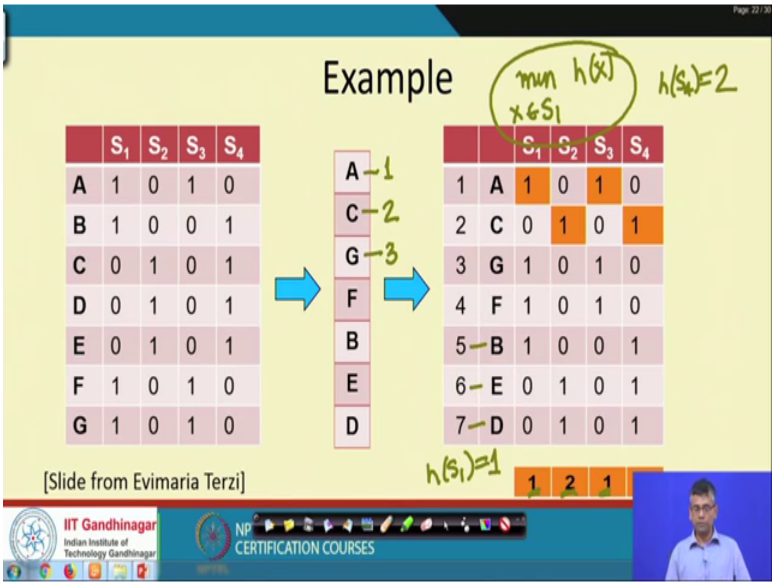
[Slide from Evimaria Terzi]

So, in this example what we have done is that, the rows here are the elements these are the elements. So, A B C D E F G this is the universe the universe consist of the set from A to G ok. Now there are 4 sets set 1 consists of the of the elements A B F and G, set 2 consists of the elements C D E set 3 consists of A F and G set 4 consists of B C D and E.

So, I have put in once in those places and zeros everywhere else now consider a random permutation of this set of this of this universe. So, what is the random permutation? We just randomly permitted the elements right which means that let us say that I mean A was still put in the first place, C is put in the second place G is put in the third place and so on right. So, now, once we randomly permute right let us randomly I mean order the rows of this matrix according to this random permutation right.

So, if you are worried about thinking about the matrix things about think about only one column right think about the column S 1. So, now, S 1 had the elements A B F and G right. So, previously A was in position A was row 1, row number 1 B was row number 2 and then came F and came G. Now according to this permutation, I select to write A as row number 1, then have to write G according to this order then I have to write F then I have to write B right.

(Refer Slide Time: 07:17)



Example

$\min_{x \in S_1} h(x)$ $h(S_4)=2$

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

A – 1
C – 2
G – 3
F
B
E
D

		S_1	S_2	S_3	S_4
1	A	1	0	1	0
2	C	0	1	0	1
3	G	1	0	1	0
4	F	1	0	1	0
5 –	B	1	0	0	1
6 –	E	0	1	0	1
7 –	D	0	1	0	1

$h(S_1)=1$ 1 2 1

[Slide from Evimaria Terzi]

So, then if I choose the if I choose what is the min over x belongs to S S 1 h of x right. So, think of h of x as giving me the I mean instead random permutation, you can also think of h of x is giving a number right. In the sense that it says that A goes to position 1 C goes to position 2 G goes to position 3 and so on G goes to position 3 F goes position 4, B goes position. This is really the permutation essentially for every slot, it tells you which is the which is item that I should go to the slot right.

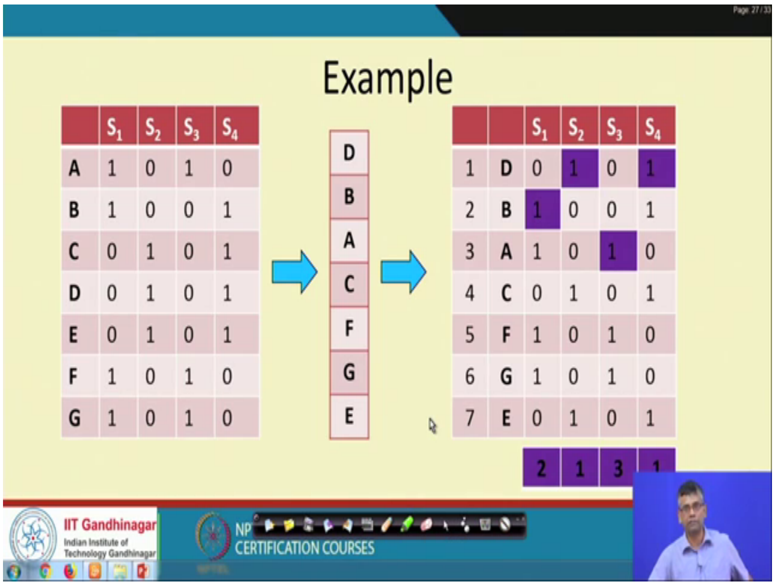
So, then if you calculate this min of this quantity, that what is the mean of our x belongs to S 1 h of x, effectively it sort of in terms of this matrix visualization, it turns out that you need to look at the first row for the column S 1, you need to look at the first row that contains a 1 right. Because that because the rows are ordered in terms of the in terms of the h x values the current h x values right and the first row that contains 1, and the row that contains 1 are the are the only rows that belong to the. The row the rows that contain a one for the column S 1 at the only row only elements that belong to this set right.

So, therefore, in this visualization, to in this operation could mean that you rewrite you reorder the rows according to the random permutation, and then for every column you choose the first row that contains 1 according to this reordering. So, here for S 1 it turns out to be A which is a number 1. So, S 1 gets map to 1. So, h of S 1 is 1 similarly what is h of S 2? So, 2 contains elements C D and E right; so, in this order.

So, if you look at the column of S 2 it still contains the elements C D any of course, because the sets not changed, but now C is placed first right. So, the so, h of S 2 is really now C right S 4. So, S 4 might be no interesting. So, in S 4 right now all that contains B right according to this ordering C comes earlier than B. So, so h of h of h of S 4 is now also is due to C which is also equal to 2 right.

So, then the sketches for this element; so, then the sketches for the corresponding sets is the sketch for S 1 is 1, sketch for S 2 is 2, sketch for S 3 is 1 sketch for S 4 is 1 2 right. So, according to this ordering the these are the these are the sketch values. Since, but we had, but will sort of see what it really is in few minutes. So, here is another permutation it is another permutation right.

(Refer Slide Time: 10:24)



Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

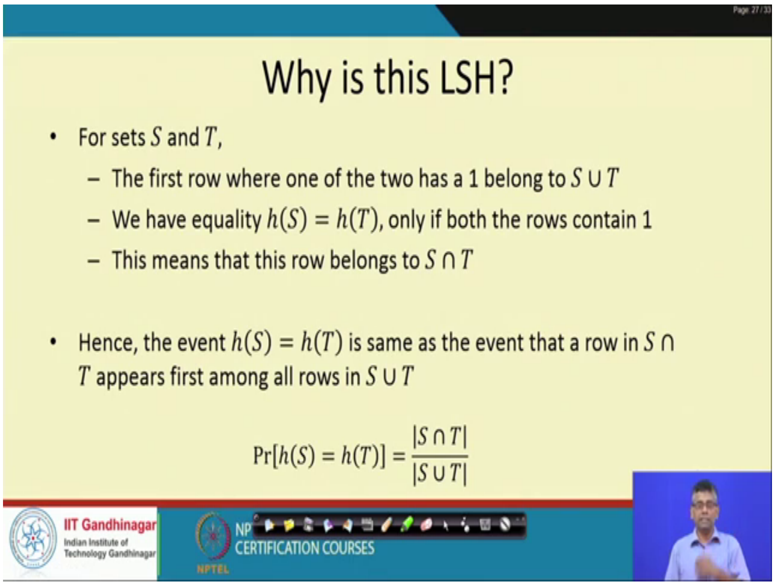
D
B
A
C
F
G
E

		S_1	S_2	S_3	S_4
1	D	0	1	0	1
2	B	1	0	0	1
3	A	1	0	1	0
4	C	0	1	0	1
5	F	1	0	1	0
6	G	1	0	1	0
7	E	0	1	0	1
		2	1	3	1

That suppose now, the permutation now again if the same sort of sets, but now the permutation is D first then B, then A then C F G and E. So, D is map to 1, B is map to 2, A is map to 3 and so on. So, what are the sketches of S 1 S 2 S 3 according to this permutation? Again you now reorder the rows according to this permutation and then go to the column of S 1. You go to the column of S 1 you see according to this order, which is the first row that contains a one in this column and that is the row number 2 corresponding to B.

So, the sketch of S 1 is 2 similarly the sketch of S 2 right is now corresponding to D it is 1. Sketch of S 3 corresponding to A it is 1 sketch of S 4 corresponding to D it is again 1. So, the sketch of the 4 of the 4 things is S 1 S 2 S 3 S 4 S 2 1 3 1 right sketch of S 1 is 2 (Refer Time: 11:28) I have written down the corresponding sketches in the in the in the columns. So, this example was taken from the lecture slides of numerators C.

(Refer Slide Time: 11:36)



Why is this LSH?

- For sets S and T,
  - The first row where one of the two has a 1 belong to $S \cup T$
  - We have equality $h(S) = h(T)$, only if both the rows contain 1
  - This means that this row belongs to $S \cap T$
- Hence, the event $h(S) = h(T)$ is same as the event that a row in $S \cap T$ appears first among all rows in $S \cup T$

$$\Pr[h(S) = h(T)] = \frac{|S \cap T|}{|S \cup T|}$$

So, the big question now right why is this locality sensitive hashing ok. So, assign what we are looking to argue is that, why is this statement true that, if you if I sample a hash function I random permutation right, and then I define the hash of a S to be according to this min thing, then why is the probability that h of S equals h of T exactly equal to the Jaccard similarity between S and T right and again it is a very simple, but very clever argument right.

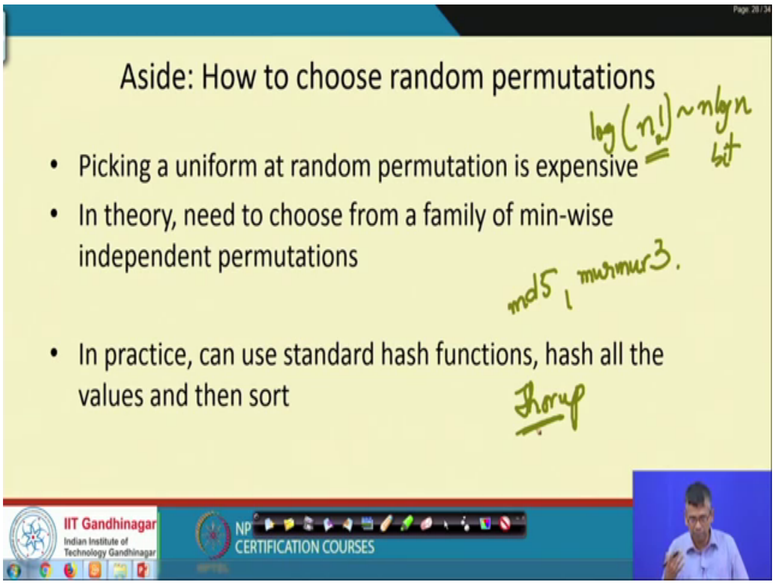
So, think of the 2 columns S and T, think of the 2 2 columns S and T. So, look at the different rows ok. So, rows that rows that are 0 in both S and T they do not really need to be considered at all right. Because they will never be the h of S will not be one of these rows and h of T will not be one of these rows. So, might is all through this rows out of the system. So, the first row right whether one of the two has a 1 that is whether either s the column S has a 1 or the column T has a 1 or may be both right. At least one of the 2 has a one right belongs to S union T right. The because that particular element by definition belongs to S union T right.

And we have a equality only if that row only if the first row right according to this permutation contains one in both positions S and T right. That means, for the first row actually belongs to S intersection T right. What that means, is that we can rewrite the statement that h of S is h of T only is the same as the event the event h of S is equal to h of T is a is the event that among all the rows in S union T right. The one that comes up first according to the permutation that have sampled is a row that actually belongs to S intersection T it cannot be anything else this is cleared.

So, as what we are saying is that, among all the rows that belong to S union T the row that that comes up first right is a row that belongs to S intersection section T. Only in this event will h of S equal h of T also the probability of this events are same right, but because the rows have been, but because this is a sample uniformly at random, but because this is uniform random permutation, the probability of any one of them I mean there are S intersection T rows right.

So, the probability of any one of them being the first I am S intersection T rows in the intersection. The probability of any one of them being the first among S union T rows is exactly S intersection T divided by S union T right. If you have 3 rows in 3 rows in S intersection 10 10 rows in S union T in a random permutation probability that 1 of 3 right is the first among one of this 10 is exactly 3 upon 10 ok.

(Refer Slide Time: 17:13)



Aside: How to choose random permutations

- Picking a uniform at random permutation is expensive
- In theory, need to choose from a family of min-wise independent permutations
- In practice, can use standard hash functions, hash all the values and then sort

$\log(n!) \sim n\log n$ bit

md5, murmur3.

Thorup

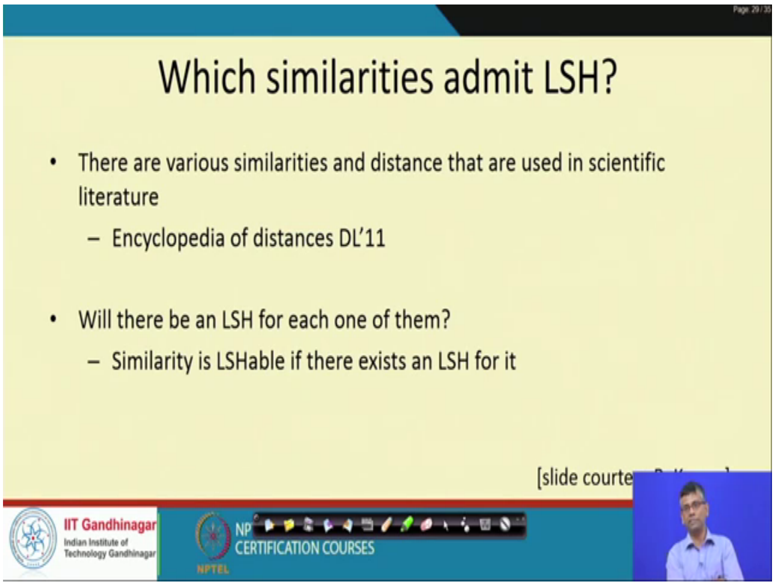
So, that proof that basically proofs that Jaccard similarity has a locality sensitive hash family. But again as a aside let us talk a little bit about how to choose random permutations right. So, as per the calculation we have been doing before, random permutation there are n factorial random permutations. So, we want to choose a random permutation, the algorithm needs at least log of n factorial bits which is like n log n bits right.

It needs that much storage in order to save the random keep the random permutation in storage this is quite a lot right. In theory what you need to define our a familiar the just like we have been defining small families of k y is independent or k k universal hash functions, here also you need to define families of min wise independent permutations right approximately min wise independent permutations.

So, in practice what we typically do is as follows; you take standard hash functions for instance you take you take m d 5 you take murmur 3, and any fast reasonable hash function right and then you hash all the values into let us say hexadecimal numbers or some strings and then you just sort them right that is all we do. So, there are some studies that show that there is some buyers that comes into this, but in some it is not very it is not it is not been sort of shown to effects real life applications very seriously, but this is an area of active research I must sort of admit that.

It is not a very satisfying solutions if we use hash families about which we do not really prove anything, I mean you should I mean there are other hash families. And, if you are interested in this, you should look at the work of Mikkel Thorup for in order to look at how to sort of create hash families that are fast, that are implementable in a fast manner as well as a provable properties.

(Refer Slide Time: 17:22)



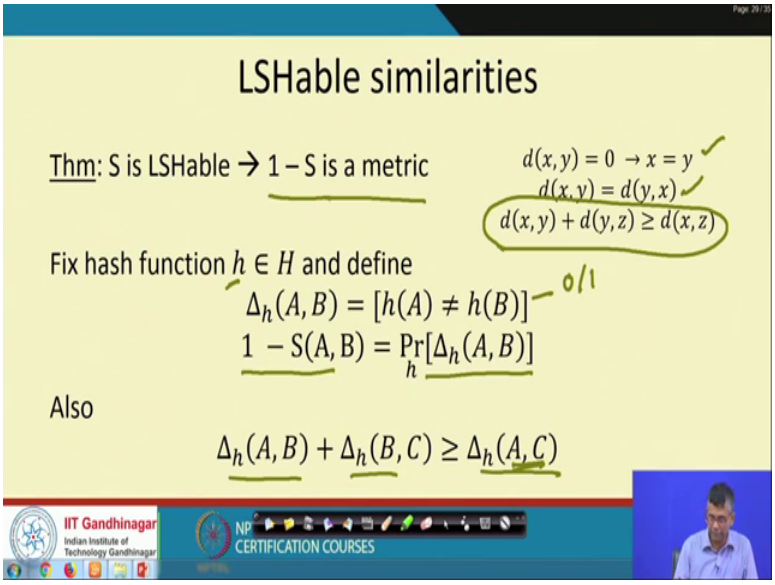
Which similarities admit LSH?

- There are various similarities and distance that are used in scientific literature
  - Encyclopedia of distances DL'11
- Will there be an LSH for each one of them?
  - Similarity is LSHable if there exists an LSH for it

[slide courte

So, then comes the question that they have been pretty lucky or maybe we have been pretty choosy in terms of talking about the distance functions talking about similarity functions right. So, there is a entire wide zoo of similarities and distances that are used in scientific literature. This actually a (Refer Time: 17:40) called encyclopedia of distances right. Will there be or encyclopedia similarities also right for instance. So, will there be an LSH for each one of them right does every similarity measure basically every similarity function that satisfies those simple properties, have corresponding LSH right.

(Refer Slide Time: 18:07)



LSHable similarities

Thm: S is LSHable → $\underline{1 - S \text{ is a metric}}$

$d(x,y) = 0 \to x = y$ ✓

$d(x,y) = d(y,x)$ ✓

$\boxed{d(x,y) + d(y,z) \geq d(x,z)}$

Fix hash function $h \in H$ and define

$\Delta_h(A,B) = [h(A) \neq h(B)]$ — 0/1

$\underline{1 - S(A,B)} = \Pr_h[\underline{\Delta_h(A,B)}]$

Also

$\underline{\Delta_h(A,B)} + \underline{\Delta_h(B,C)} \geq \underline{\Delta_h(A,C)}$

So, let us call a similarity measure to LSHable if there are exists an LSH for it just a short answer for that and here is a very simple theorem right. We say that a similarity remember we talking about similarity measures right. We say that similarity measure is LSHable, this implies that 1 minus S is actually a metric. What is the metric? Metric satisfies these 3 properties.

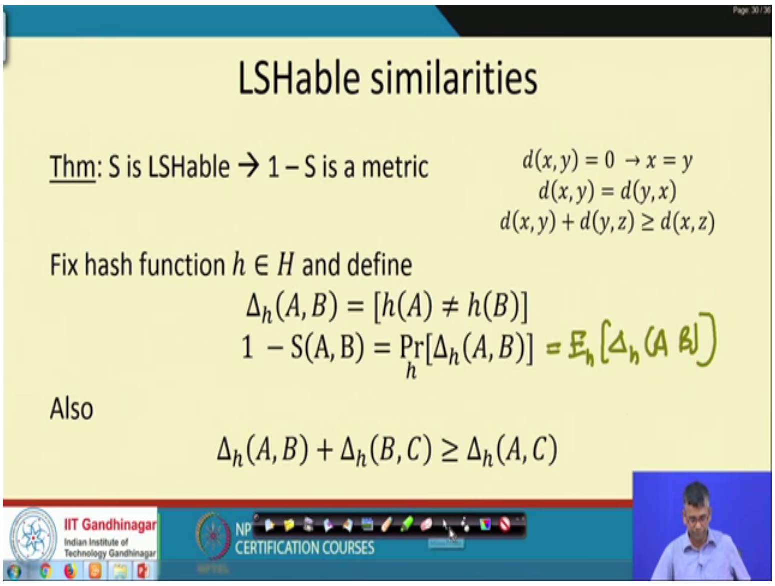
So, the d of x y is 0 implies that x equal to y, but this is already satisfied by any 1 minus s because s satisfies the corresponding property. Symmetry is also satisfied by any 1 minus s, the big question is this right. The does 1 minus s satisfy triangle inequality and what we say for this very simple theorem says is that, if s is LSHable then 1 minus s must satisfy triangle inequality. Or to take it the other way around if 1 minus s does not we can show that 1 minus s does not satisfy triangle inequality, then we can show that this similarity measure is not LSHable according to the definition of LSH that we have given. Let us start to prove this very small very cute theorem.

So, here is what will do, suppose there is a hash family h then any hash family and. So, so pick a hash function h belongs to H and define the event. So, we defining 0 1 0 1 indicated random variable. So, that the random variable takes in 3 things, delta h A and B 2 sets A and B. And the random variable is 1 right if h of A is not equal to h of B and it is 0 else. So, it is a 0 1 random variable right. So, now, it is also easy to see that 1 minus S A B is a probability of h delta h A B right by definition. I mean because I mean that suppose let us the contradiction, that suppose we say suppose we say that that I mean that s is LSHable. Well not to leave a contradiction where assuming that S is LSHable and that is what we go with. So, s is LSHable and therefore, this is true ok.

So, let us let us proceed, but now what you can say is that this particular statement is true, that the 3 the random variables are the variables in the 0 1 random variables delta h A B, delta h B C and delta h A C they must satisfy this property then it is easy to see why right. Because the left hand side right says that A is. So, h of A not equal to h of B or h of B not equal to h of C. Any one of these events happen then it must happen right.

So, other let us say the let us say let us argue the other way around; that supposing h of A is not equal to h of C right then it must be the case then either h of A not equal to h of B or h of A or h of B not equal to h of C right? Which means that if this if this right hand side if this delta h AC is equal to 1, it must be that one of the left hand side random variables also becomes 1; which means that this triangle inequality is satisfied by the by these random variables. But if that is the case right, but if that is the case then just plugging in the I mean I mean then just plugging in the expectations.

(Refer Slide Time: 21:55)



Page 30/36

LSHable similarities

Thm: S is LSHable → 1 – S is a metric

$d(x,y) = 0 \rightarrow x = y$

$d(x,y) = d(y,x)$

$d(x,y) + d(y,z) \geq d(x,z)$

Fix hash function $h \in H$ and define

$$\Delta_h(A,B) = [h(A) \neq h(B)]$$

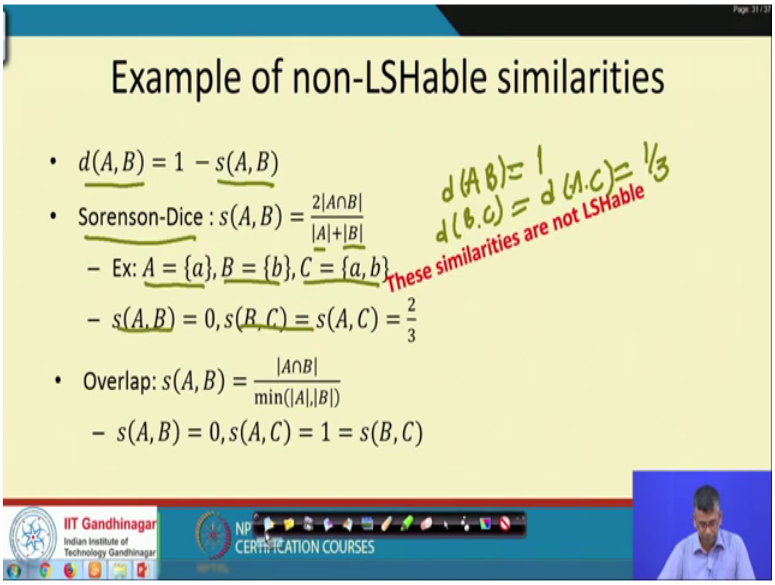
$$1 - S(A,B) = \Pr_h[\Delta_h(A,B)] = E_h[\Delta_h(A\ B)]$$

Also

$$\Delta_h(A,B) + \Delta_h(B,C) \geq \Delta_h(A,C)$$

The plugging in the fact that this is also equal to the expectation of h delta h AB right we get the expectation of the of the left hand side and the right hand side, and then we immediately get that 1 minus S has to be a metric.

(Refer Slide Time: 22:15)



Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a,b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

$d(A,B) = 1$

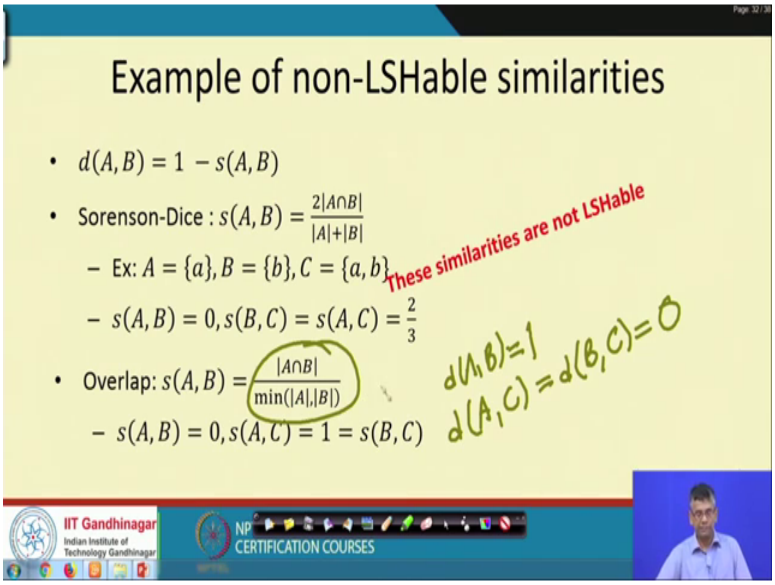
$d(B,C) = d(A.C) = 1/3$

These similarities are not LSHable

So, let us look at some examples of non LSHable similarities right. So, so here is I mean will define a similarity and then we look at the corresponding distance function to be 1 minus d AB to be d A B to be 1 minus s A B. So, the popular one is Sorenson dice. It is s A B is 2 A intersection B divided by cardinality of A plus cardinality of B it is. So, if you take A equal to just the element a B equal to just the element b c equal to just the element a b, you can easily see that s A B is 0 and s B C and s A C are both 2 thirds right or that means, is that I mean if you take the corresponding distance functions, the distance of distance of A B would be 1 distance of B C equal to distance of A C would be equal to one third right.

So, this does not be a triangle inequality similarly another popular one is the overlap metric, which says that you take a intersection b divided by minimum on the cardinality of A and the cardinality of B right. Even here for the same example you see that s A B equal to 0 s A C equal to one equal to s B C and therefore, you can you can you can calculate the corresponding distance values distance of A B is 1.

(Refer Slide Time: 23:51)



Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a,b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

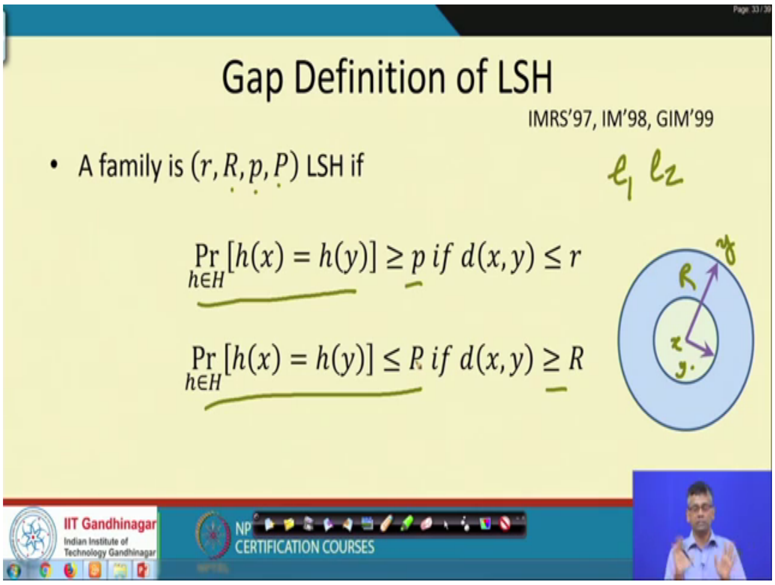
These similarities are not LSHable

$d(A,B)=1$

$d(A,C)=d(B,C)=0$

Distance of A C equal to distance of B C equal to 0 right. So, therefore, it does not be a triangle in equality right. And so, that that simple thing gives you a proof that that there is no that is that is similarity measures are not LSHable right. So, now we have seen both example, we have seen examples of similarity measures, we have seen a bunch of important similarity measures and we have seen LSH constructions for this and we have seen LSH we have seen similarity measures certain are not LSHable.

(Refer Slide Time: 24:27)



Gap Definition of LSH

IMRS'97, IM'98, GIM'99

- A family is (r, R, p, P) LSH if

$$\Pr_{h \in H}[h(x) = h(y)] \geq p \; if \; d(x,y) \leq r$$

$$\Pr_{h \in H}[h(x) = h(y)] \leq P \; if \; d(x,y) \geq R$$

ℓ_1 ℓ_2

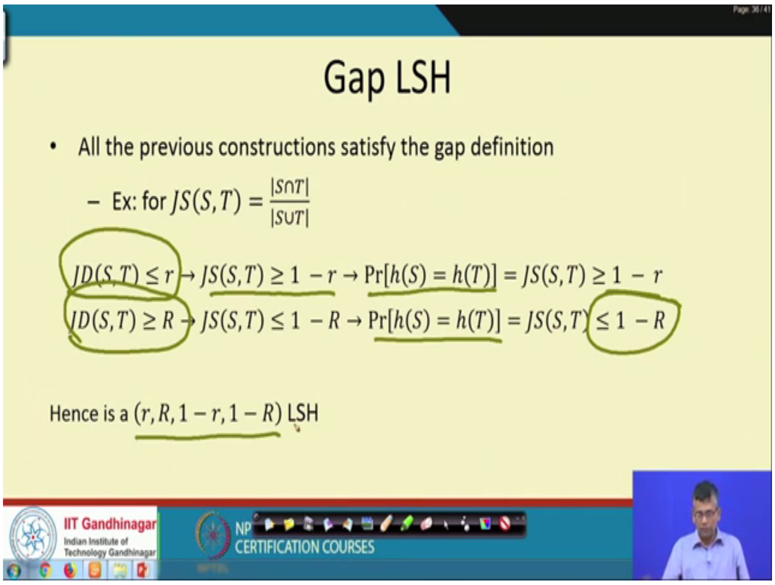
However we need to go a little further right we need to relax as well soon see that for at least one more important distance function specifically the set of the set of l 1 l 2 matrix, we need to relax the definition of LSH that we have right and here is what will do? And now, we are shifting from defining LSH in terms of similarity measures to defining it in terms of distance measures. Because, we saw that being a distance being a distance metric is actually more important right because be I mean all similarity measures do not have LSH, but I mean there is it does is the probability of being a distance function at least, although that is not a sufficient condition, that the necessary condition.

So, we say that a family. So, we say that a family h is small r big R capital P small p capital P LSH if right if let us say if 2 let us say this is x if 2 if 2 points x and y are distance less than equal to r small r, then the probability of collision according to h is at least p or some p. If 2 points x and y are at least capital R distance apart, then the probability of collision is less than capital P.

So, what are we saying now? Restricted relaxing the definition that we had because previously we said that no it monotonically decay it monotonically decays with the probability of collision, monotonically decays with the, with 1 minus distance or monotonically decays with similarity right. So, now, we are saying that no we do not really bother so much, for points we have 2 radius right I mean these radius will be fixed based on I mean flip this later based on applications, we have a smaller radius we have a bigger radius bigger radius is probably constant fraction on the smaller radius.

If the points are closer than this bigger than this smaller radius, then we say then we want our high probability of collision if the points are further apart from that bigger radius, we would not have a low probability of collision just these 2 that is it. So, what can we do with this?

(Refer Slide Time: 22:57)



Gap LSH

- All the previous constructions satisfy the gap definition
  - Ex: for $JS(S,T) = \frac{|S\cap T|}{|S\cup T|}$

$JD(S,T) \le r \rightarrow JS(S,T) \ge 1 - r \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \ge 1 - r$

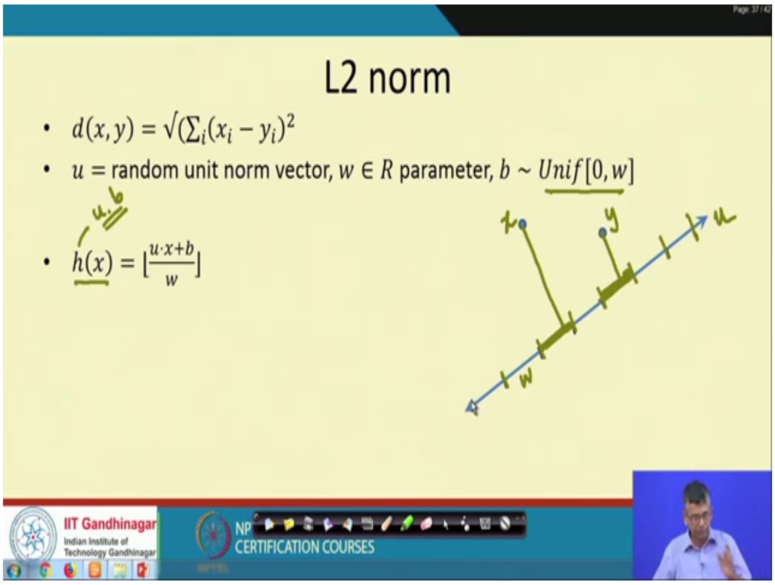
$JD(S,T) \ge R \rightarrow JS(S,T) \le 1 - R \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \le 1 - R$

Hence is a $(r, R, 1 - r, 1 - R)$ LSH

So, first of all why did we need this? So it is easy to see that this is the generalization of the previous definition right. For instance if you take the Jaccard similarity right and if you take the it is and the corresponding distance, if you say that the Jaccard distance is less than or equal to r, I mean it is easy to do this line of arguments that says the Jaccard in that case the Jaccard similarities at least 1 minus r and the previous hash function that is min wise hash function that we were doing is the probability of collision according to that is at least 1 minus r.

Is the Jaccard distance is bigger than capital R right, then the probability of collision is at most 1 minus capital R. So, then our previous LSH min wise independent permutations, they gave us a small r capital R 1 minus small r 1 minus capital R LSH that as well as easy to see.

(Refer Slide Time: 27:47)



L2 norm

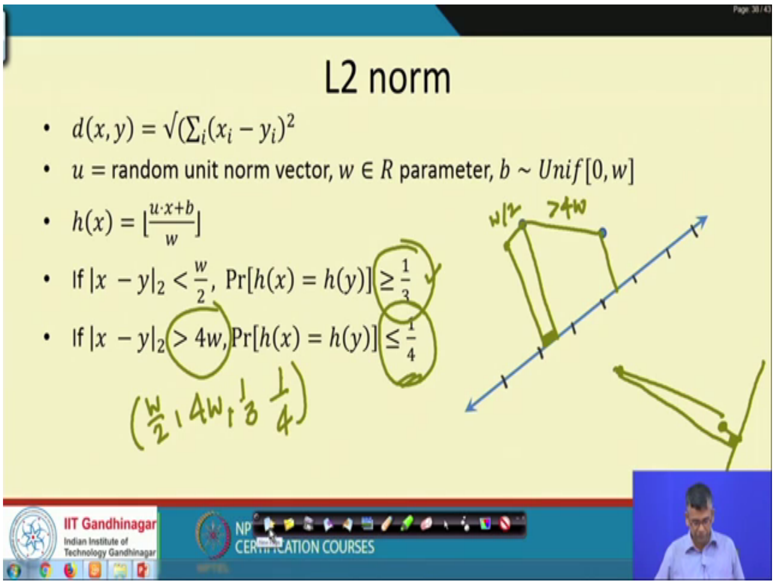
- $d(x,y) = \sqrt{(\sum_i(x_i - y_i)^2}$
- u = random unit norm vector, $w \in R$ parameter, $b \sim \underline{Unif}[0,w]$
- $\underline{\overset{u,b}{h}}(x) = \lfloor\frac{u\cdot x+b}{w}\rfloor$

So, here is here is one for which we did not I we do not know of any LSH according to the previous stronger definition, the chunk similarity definition there is an LSH according to this slightly weaker definition. So, and this is the standard Euclidean norm right. So, suppose we have suppose we have the x and y, we measure the Euclidean norm as the some square of the distance square root of that.

So, how they construct an LSH for that? So, we have 3 components first we take a unit vector u right then we chunk we sort of have a w, and then we sort of build part it chunks on these intervals in the unit vector of size w right. We also have a b that is decided uniformly within 0 and w. So, think of this as jittered in the that sort of decides where the 0 of this chunk of each chunk is, it sort of does has a random shift right. So, then the h of q of a particular the h is now a function of u and b these are the 2 random choices right. And now h of x is defined as you take the dot product of u x, you add the shift and then you sort of bucket it by w.

So, in some sense what you are trying to do is that you are trying see that if I do a random projection right of this x on to this line u, if I do a random projection on y on to this line u, what is the idea of the bucket that it falls into right. So, the bucketing is of size w and it is random shifted randomly shifted bucket. So, you want to see: what is the idea of this bucket that it falls into ok. So, this is what will try to do, and this is the hash, this is the hash value for any x. So, why is this an LSH?

(Refer Slide Time: 29:46)



L2 norm

- $d(x,y) = \sqrt(\sum_i (x_i - y_i)^2$
- u = random unit norm vector, $w \in R$ parameter, $b \sim Unif[0,w]$
- $h(x) = \lfloor \frac{u \cdot x + b}{w} \rfloor$
- If $|x - y|_2 < \frac{w}{2}$, $\Pr[h(x) = h(y)] \geq \frac{1}{3}$
- If $|x - y|_2 > 4w$, $\Pr[h(x) = h(y)] \leq \frac{1}{4}$

$\left(\frac{w}{2}, 4w, \frac{1}{3}, \frac{1}{4}\right)$

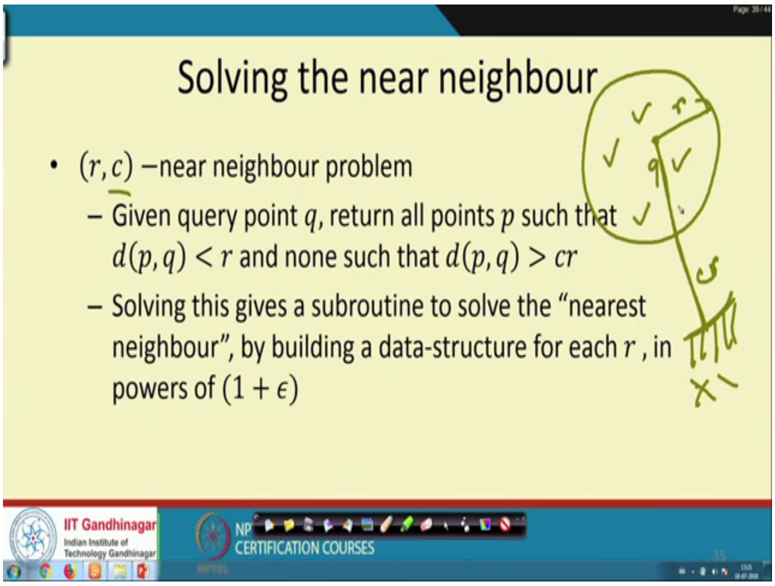
w/2
>4w
Page 30/43
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES

So, I would not go through the exact proof of this, and these numbers are I have written down on is moral approximate, but basically here is the idea. That suppose you know that x and y are very similar, suppose you know that x and y are less than w by 2 apart right. Then this is a very high chance, because then the distance there distance in their I mean in after projection is also less than w by 2. So, the only way they could sort of fall in different partitions is because of the random shift, and the probability of that is really small right you can bound the probability that there is the random shift is less than let us say if d is less than one third right then they would not really be shift then they will fall in the same bucket in some sense.

So, this is fine. So, now, suppose x and y are very far apart they bigger than 4 w right. So, now, if they are falling in the same bucket, which means that if they did fall in the same bucket which means that the picture something like this. They very far apart, but maybe the random plane is somewhere here, which means that that they are their projection in their in this projected line the distance is not very big. But that means, that the direction; that means, something about the direction of the projection line; that means, of the direction of the projection line right has a particular angle with the direction of the line x y right. In fact, it sort of lies in a particular cone around the around the line x y right and you can bound the probability of that right.

And so, we can make statements that say that that if that if the if the distance is less than w by 2, then the probability of collision is at least one third, the distance is less than let us say 4 w then the probability of collision is at most one third; which means that we have a hash function that is w by 2, 4 w one third, one fourth LSH according to our previous definition right ok. So, now, what we do with this functions ok.

(Refer Slide Time: 32:00)



Solving the near neighbour

- (r, c) –near neighbour problem
  - Given query point q, return all points p such that $d(p,q) < r$ and none such that $d(p,q) > cr$
  - Solving this gives a subroutine to solve the "nearest neighbour", by building a data-structure for each r, in powers of $(1+\epsilon)$

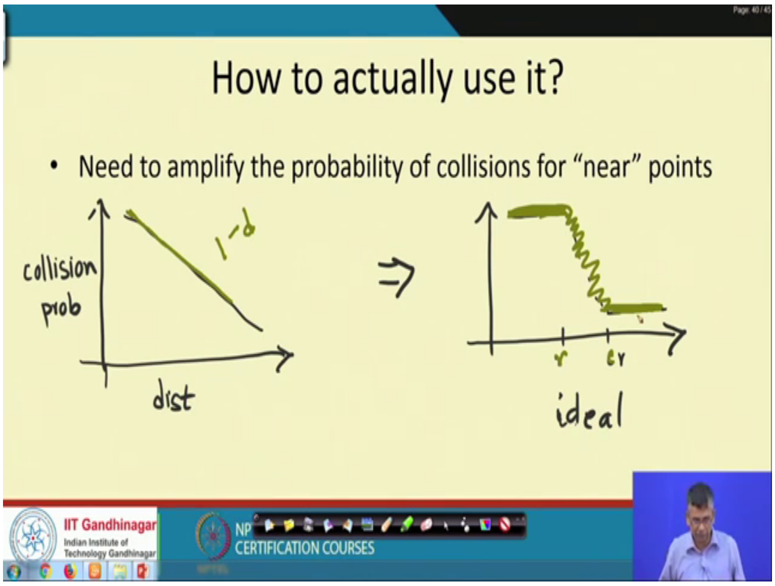
r
q
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES

So, now here is the problem that we want to solve. Suppose we want to solve an r c near neighbor problem what is that mean what; that means, is that given a query point q and the target distance r, I want to return all points that are within this target distance right. And I do not want you to return any point that is c r away from the target distance right. I do not want you to return points outside of the c r, this is the strict no no, but I want you to return everything inside this smaller wall and for things in between I do not care you might return them.

So, why is this important subroutine? This is an important subroutine because suppose you do this, that suppose you have a subroutine to do this then at least theoretically what you could do as follows. You could sort of take powers of r r to be 1 plus epsilon times the minimum distance r to be I mean start with r to be the radius of the entire data set and decrease it in powers of 1 plus epsilon right. So, either logarithmic number of possible values of r, build the data structure for each of them right.

Now, given a particular query search in each of this data structures right, and you can you can say that I searching each of the data structures you can actually find out then nearest neighbor to the query q right. Because the nearest neighbor will have some distance will have some distance r right on that right and that right data structure, you will you will stop right and then and then you will be able to answer this nearest neighbor question, if you have the subroutine.

(Refer Slide Time: 33:47)



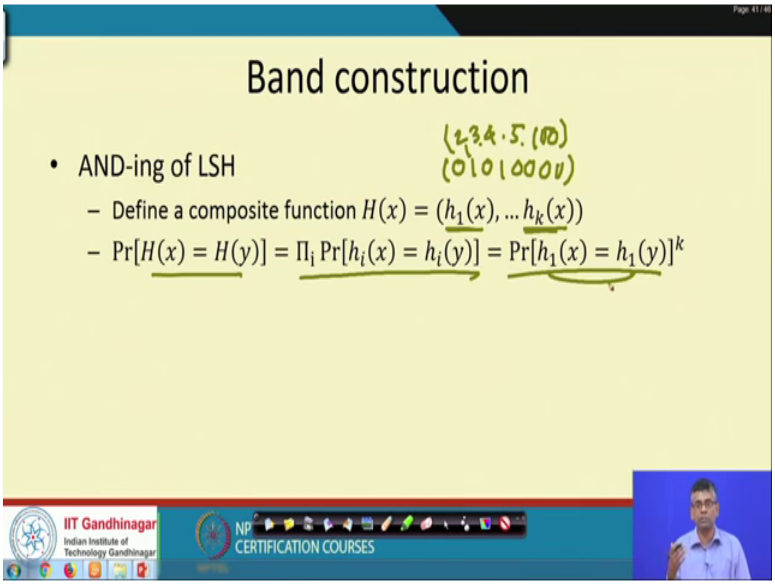
How to actually use it?

- Need to amplify the probability of collisions for "near" points

But then how all we actually using this hash functions for this right. Because here is the problem the problem is that that you look if you plot the distance versus the collision probability of the hash functions, let us take the Jaccard distance right. Instead of goes down very smoothly right it is like 1 minus 1 minus d right.

So, we do not want that. What we want picture something like this that we are given r, we are given c r what I want is that collision probability should be very high if distance is less than r, collision probability should be very low if the distance is bigger than c r and in here I do not care and it can be anything. Just that I want this to be high and I want this to be low right. So, how do I achieve that? Are this is the next big idea in LSH, which is by helping you do this amplification in probabilities.

(Refer Slide Time: 34:42)



Band construction

- AND-ing of LSH
  - Define a composite function $H(x) = (\underline{h_1}(x), \ldots \underline{h_k}(x))$
  - $\Pr[\underline{H(x) = H(y)}] = \underline{\Pi_i \Pr[h_i(x) = h_i(y)]} = \underline{\Pr[h_1(x) = h_1(y)]}^k$

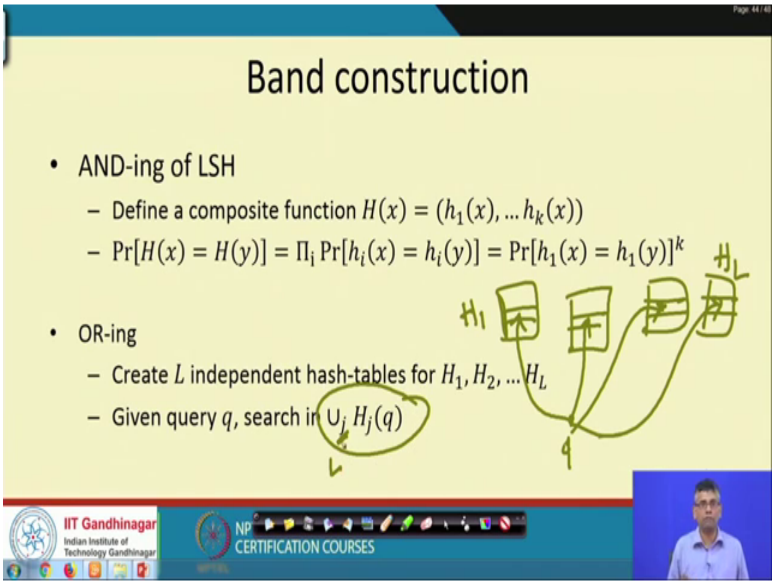
(2,3.4·5.(10)
(0101000U)

So, here is the 2 here the 2 operations it will do. That first will do what is known as a AND-ing what is AND-ing. Instead of a single hash bit see this is this is one of these one of the original hash functions that we were constructing, instead of taking one of them tik k of them and then build tuple out of them. Let see these are these are 0 one then build a tuple out of them k tuple right. If these are numbers build a tuple out of them right and now my new hash function is this tuple, new hash value of this tuple. So, now, what is the probability that that H of x equals to H of y? Of course, it is the because you also choose them independently this key functions independently so, then it is a product of this probabilities.

So, then if we if are follow them and identically distributed; so, it is a probability that one of them collides the power k; because from I mean because this probability is a function of the distance between x and y nothing else. But see what might happen is that while see everything because we are doing everything to the power k all the probabilities fall down right. But then if 2 points are closed even that probabilities falling down, even that probabilities falling down; so, now I want to raise that right.

Because, I want to say I mean we might have achieve this, we might have achieve this part because we have pulled a all the probabilities down. But now I also want to achieve this part. The fact that the, fact that if the distance is small right the probability of collision is very high.

(Refer Slide Time: 36:39)



Band construction

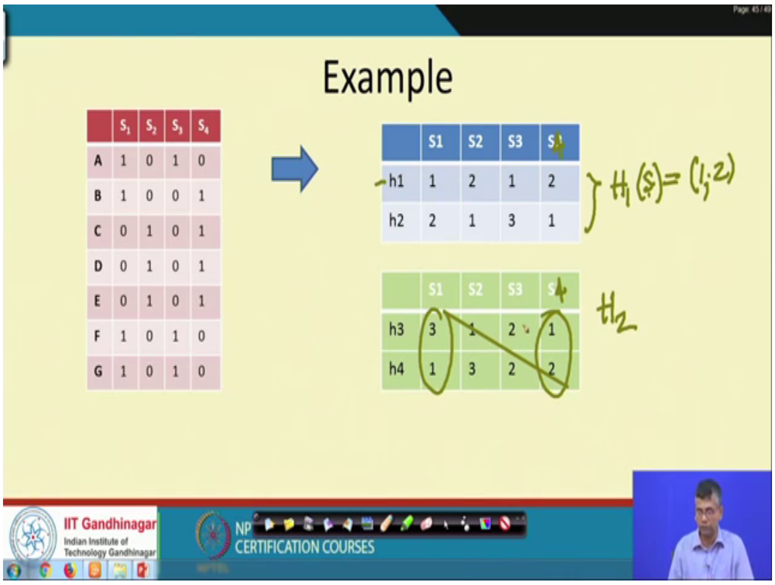
- AND-ing of LSH
  - Define a composite function $H(x) = (h_1(x), \ldots h_k(x))$
  - $\Pr[H(x) = H(y)] = \Pi_i \Pr[h_i(x) = h_i(y)] = \Pr[h_1(x) = h_1(y)]^k$
- OR-ing
  - Create L independent hash-tables for $H_1, H_2, \ldots H_L$
  - Given query q, search in $\cup_j H_j(q)$

Then the next thing that we do is oring and this is actually very simple. This is says that you have one search each, now build multiple search hash tables right one for each of the H s. So, create some capital L number of H s each of them. So, each such each such capital H 1 is again build out of a tuple right, but now build capital L such capital L such L such big hash tables now right.

Now, when we search for a query right when we search for a query, you go to each of the hash tables and then you search for each of them right? You basically then take the union of these of these of these hash buckets right and then you search for a near neighbor in the union right.

So, then this union is a union over the over the L tables. You looking at the candidates in all of them putting them together and then and then trying to find out the near neighbor. So, there 2 steps one is a is a AND-ing operation in some sense, in which you building a composite hash function using a tuple using a tuple of smaller hash functions. And the other is a OR-ing operation, in which you are you are sort of creating multiple tables and you are putting them together. And you use all the tables, when you are trying to look for query.

(Refer Slide Time: 38:06)



Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

	S1	S2	S3	S4
h1	1	2	1	2
h2	2	1	3	1

$H_1(S) = (1,2)$

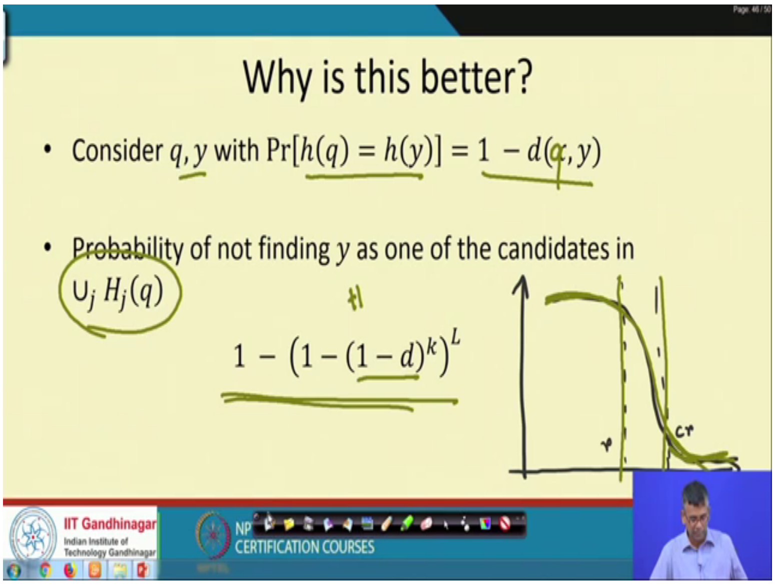
	S1	S2	S3	S4
h3	3	1	2	1
h4	1	3	2	2

H_2

So, it is the simple example for instance right because previously remember we saw 2 we had these sets S 1 S 2 S 3 S 4, we had one random permutation h one and that had this sketch for S 1 it had one for S 2 it had 2 S 3 and this should be S 4 really S 4 it had this, and similarly we had h 2 which are the sketch. So, now, this would be capital H 1. So, capital H 1 of S 1 would be equal to the tuple 1 comma 2.

Then capital H 1 of S 2 would be the capital 2 comma one right and then we would build a table out of this. Then next table would be capital H 2 which is build out of yet another 2 independent hash functions h 3 and h 4. So, S 1 is map to this tuple S 4 is map to this tuple and so, on. So, that is. So, that is what will do.

(Refer Slide Time: 39:05)



Why is this better?

- Consider q, y with $\Pr[h(q) = h(y)] = 1 - d(q, y)$
- Probability of not finding y as one of the candidates in $\cup_j H_j(q)$

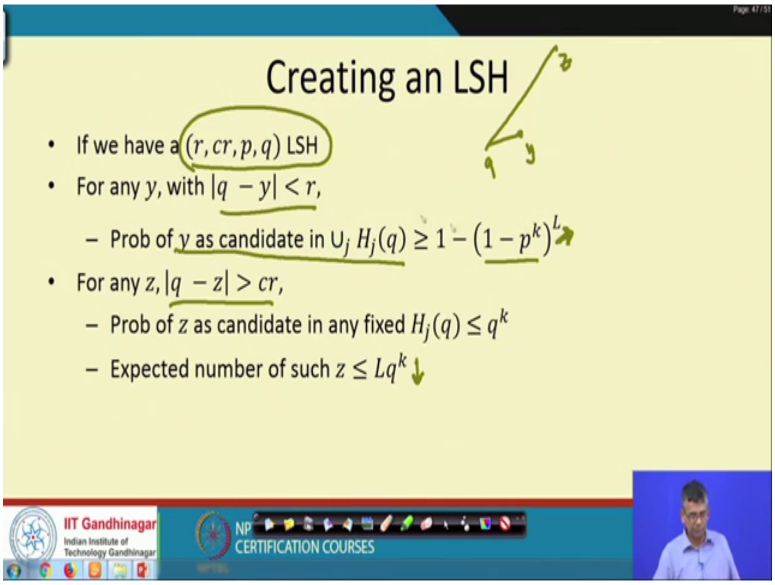
$$1 - \left(1 - (1-d)^k\right)^L$$

So, why is this good and this is a intuition. That suppose you are trying to write down the probability for given q and any y you trying to write down the probability that h q equal to h y right and we know that for the basic hash function, this probability depends as a distance between q and y 1 minus distance between q and y right. Then the probability of finding y as one of the candidates in this, is this expression and let us analyze this.

So, this expression is a probability that in one of that it is in one of the capital H of q equal to capital H of y right this is the probability that one it is 1 minus d to the k. So, then 1 minus that is a probability that capital H of q is not equal to capital H of y, that raised to the power L is a probability than capital H of q that q and y fall in different buckets in all the L hash tables right.

So, 1 minus that is a probability that at least in one of the capital H, one of the capital L hash tables q and y have been placed in the same bucket ok. So, the 3 there are 3 steps here and you have to carefully go for that. And it turns out very interestingly that if you plot this function it looks something like the sigma it curve more or less right; which is that you can choose the values of k k k and l depending on the depending on these radius on the radius r and c r, such that it is high in this spot and it is low in this spot and it is can go some smoothly here right and this and that is what will do.

(Refer Slide Time: 40:49)



Creating an LSH

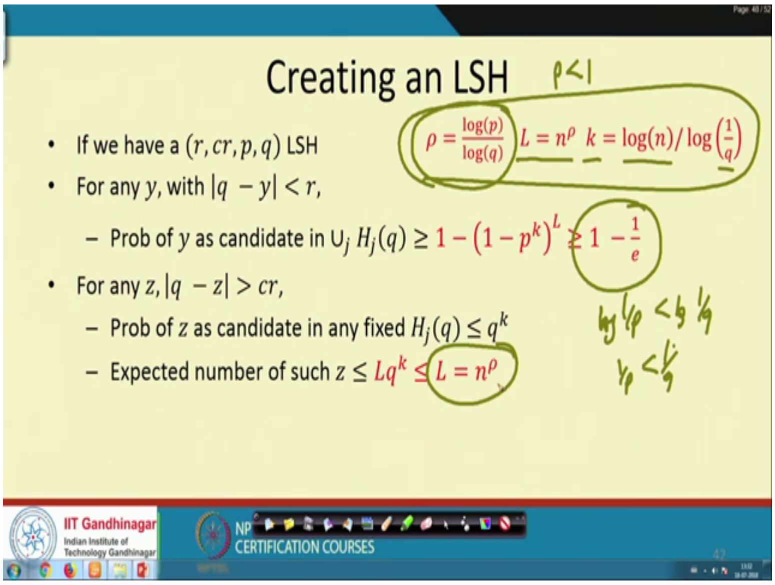
- If we have a (r, cr, p, q) LSH
- For any y, with $|q - y| < r$,
  - Prob of y as candidate in $\cup_j H_j(q) \geq 1 - (1 - p^k)^L$
- For any z, $|q - z| > cr$,
  - Prob of z as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k$

z
q
y
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES

So, just to being formally write down formally that if we have a base hash family that is r c r let say p q LSH right then for any y such that q minus y is less than r, the probability of y as a candidate in this in this after looking at the L capital L hash tables is 1 minus 1 minus p to the k to the this raise to the power L. Because, this is the probability that it is a candidate in one of the hash tables is not a candidate in one of the hash tables that raise to the power L is that none of the hash tables return it as a candidates 1 minus that is the probability of getting it as a candidate in at least one of the hash tables right and similarly if q minus z is bigger than c r.

So, now are looking at q we are looking at 1 y that is close and o1 z that is far right. Similarly anyone can say that the probability of that z appearing as a candidate in any fixed hash table is less than q to the k by this definition. Therefore, the expected number of z is L times q to the k. So, now, what do I want I want to choose the values of k and capital L, I want to choose it so, that this probability is high and this expectation is small right. Because this expectation is what is driving my query time making these expectations on will reduce my query time, making this probability high will increase the probability that I actually find the near neighbor y right.

(Refer Slide Time: 42:25)



Creating an LSH $\rho < 1$

- If we have a (r, cr, p, q) LSH
- For any y, with $|q - y| < r$,
  - Prob of y as candidate in $\cup_j H_j(q) \geq 1 - (1 - p^k)^L \geq 1 - \frac{1}{e}$
- For any z, $|q - z| > cr$,
  - Prob of z as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k \leq L = n^\rho$

$\rho = \frac{\log(p)}{\log(q)}$ $L = n^\rho$ $k = \log(n)/\log\left(\frac{1}{q}\right)$

$\log 1/p < \log 1/q$

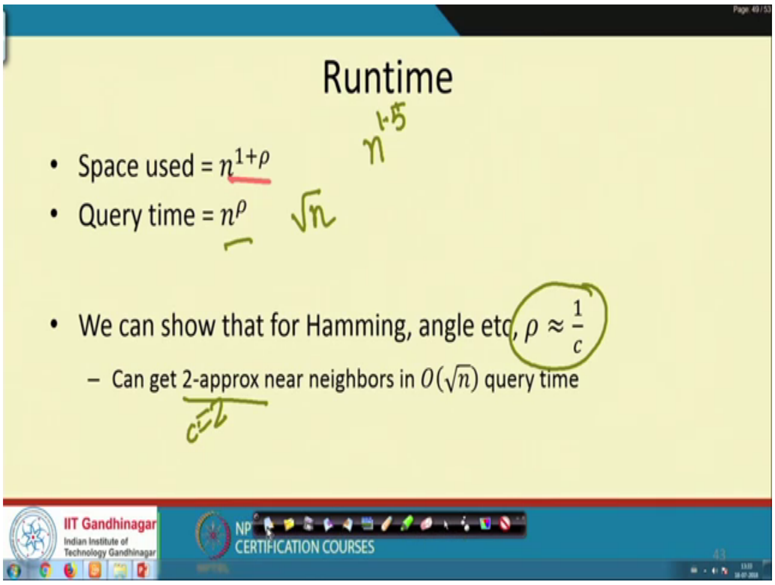
$1/p < 1/q$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES
42

So, let us see if you can choose this values right. It turns out and this you should really convince yourself by plugging in these values, where it is possible I mean the important quantity is this is this ratio rho between the log p and the log q right. And if you set l by n to the rho and it is the important to note that rho is less than 1. Rho is less than 1 because p is because p is smaller than p is larger than q p is log 1 by p is less than log 1 by q. So, 1 by p is less than q we need that which means that p is larger than q also.

So, and I need to choose L to be n to the rho and we need to choose k to be log n by log 1 by q. And you should plug in these values you can make this probability to be at least 1 by e 1 minus 1 by e and you can make this to be less than equal to n to the power rho right. This might not seem very exciting to you because it is still n to the power something, but see it is its bit big true right. Because now this is the query time is now n to the rho and we can make which is which is sub linear, and it was not at all clear if we could make it sub linear ever right.

(Refer Slide Time: 43:50)



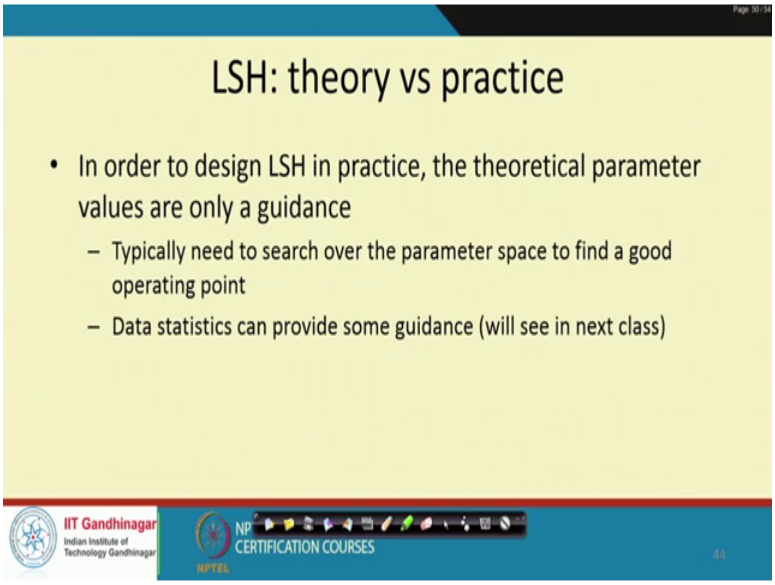
Runtime

- Space used = $n^{1+\rho}$
- Query time = n^{ρ}
- We can show that for Hamming, angle etc, $\rho \approx \frac{1}{c}$
  - Can get 2-approx near neighbors in $O(\sqrt{n})$ query time

And so the number of and so, the space required the space required is n to the power 1 plus rho right. Because you have n to the rho hash tables each of them needs n space and the query time is n to the rho right. So, I need I need also turns out that if you analyze the actual basic hash functions of hamming angle etcetera, you can for if you want a c approximate near neighbor, you can set rho to be 1 by c I mean it is an offer rho to be 1 by c right, which mean what does that mean?

We can get a 2 approximate near neighbor, is c equal to 2 right we can get a 2 approximate near neighbor, in time that is query time that is square root of n and space that is n to the power 1 plus half one 1.5 right. So, so by sort of not blowing up the space too much, we were still able to substantially reduce the query time from n square root n.

(Refer Slide Time: 44:51)



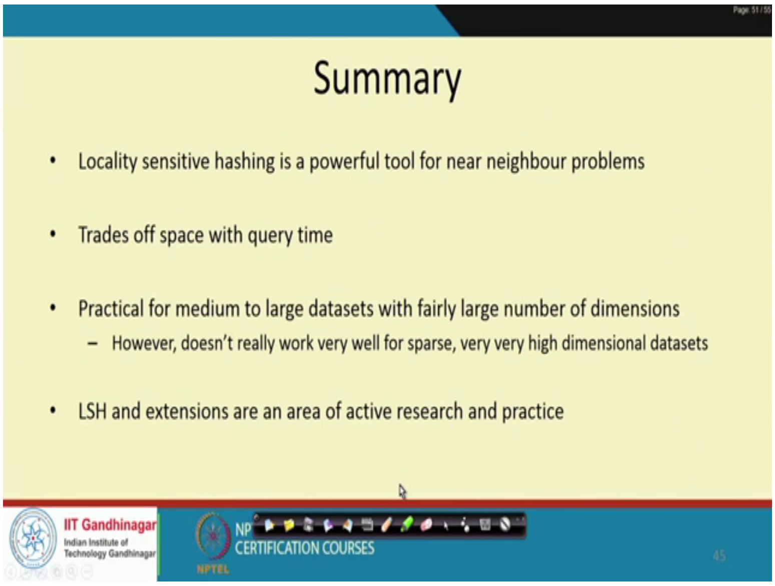
LSH: theory vs practice

- In order to design LSH in practice, the theoretical parameter values are only a guidance
  - Typically need to search over the parameter space to find a good operating point
  - Data statistics can provide some guidance (will see in next class)

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES
44

So, just to summarize, in order to design I mean this is all nice in theory, but in practice the theoretical parameter current is our only a guidance. What you really need to search for, I mean in practice is you need to really search over the parameter space to find a good operating point the values of k and l. There are some ways of guiding the search to by looking at some data statistics and will see that in next class and.

(Refer Slide Time: 45:18)



Summary

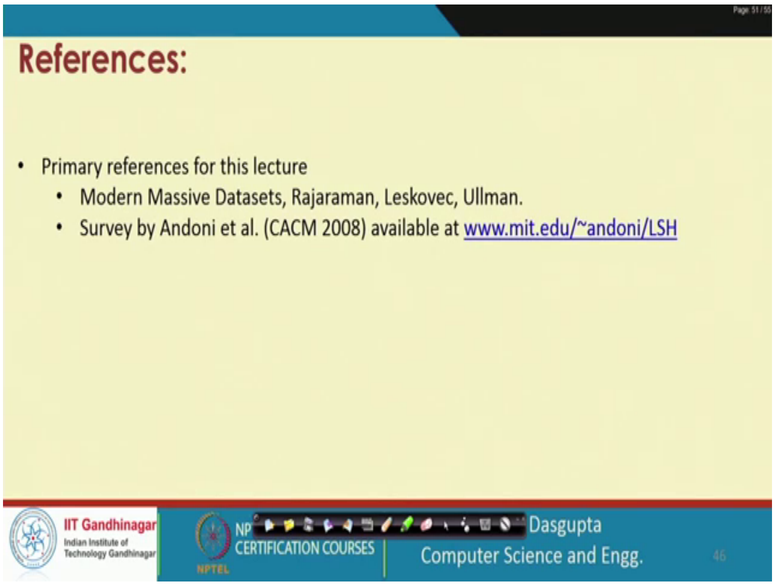
- Locality sensitive hashing is a powerful tool for near neighbour problems
- Trades off space with query time
- Practical for medium to large datasets with fairly large number of dimensions
  - However, doesn't really work very well for sparse, very very high dimensional datasets
- LSH and extensions are an area of active research and practice

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL CERTIFICATION COURSES
45

So, just summarizing over all the LSH is actually a very powerful tool for nearest neighbor search problems, it has a it is a very active area of research as well as practice it is use a lot in practice. I at the basic theme is to trade of space with query time right and it is one big kind of open question is that, it is really practical for medium to large data sets because you are blowing up space right.

So, you cannot really deal with billions of data sets. And sort of doing it in distributed manage also kind of an open question. it is practical from medium to large data sets with fairly large number of dimensions, but for clear how to make it work for very very sparse very very high dimensional that it is right you should you should look at some of these references.

(Refer Slide Time: 46:09)



References:

- Primary references for this lecture
  - Modern Massive Datasets, Rajaraman, Leskovec, Ullman.
  - Survey by Andoni et al. (CACM 2008) available at www.mit.edu/~andoni/LSH

Page 51 / 55
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL
CERTIFICATION COURSES
Dasgupta
Computer Science and Engg.
46

The one of them is the is the book that we are that we have been that we have pointed to in the syllabus, modern massive data sets by Rajaraman Leskovec and Ullman, and other is this excellent survey as well as a number of other very recent papers by Alexon Andoni in the page that you maintains on LSH at MIT and the that is it for today.

Thank you.