

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

Lecture – 13a
Multi Probe LSH

Welcome to the course on Scalable Data Science. I am Anirban from IIT, Gandhinagar. And today we will be talking about Multi Probe LSH.

(Refer Slide Time: 00:28)

Finding Near Neighbors

Given a set of data points and a query

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

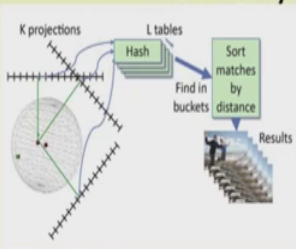
- K-nearest neighbors
- $d(p, \text{query}) < r$

The slide features a diagram with several blue dots representing data points and one red dot labeled 'query'. A blue arrow points from the query point to the nearest blue dot. The slide is part of an NPTEL online certification course from IIT Gandhinagar.

So, we talked about the problem of finding near neighbors. And just to recap this problem. We are given a set of data points. Let us say x_1 to x_n that lie in some d -dimensional space. We can pre process this data set as we want. And then at the query time we are given a particular query point q . So, our question is that we have to answer what is the nearest data point to this query right. And then there are various versions of this question. We could be asked the question of finding out the K -nearest neighbor to this given query or we could be asked a question that given the query point and the radius r return me any point that lies within radius within the distance r of the query point q .

(Refer Slide Time: 01:21)


Locality Sensitive Hashing



Given input data, radius r , approx factor c and confidence δ *confidence*.

Output: if there is any point at distance $\leq r$ then w.p. $1 - \delta$ return one at distance $\leq cr$ *c > 1*

Picture courtesy Slaney et al.



IIT Gandhinagar
Indian Institute of Technology Gandhinagar

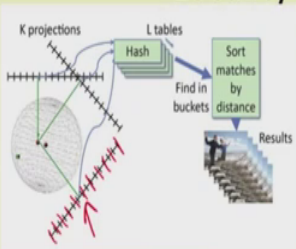
NPTEL ONLINE
CERTIFICATION COURSES

So, we also looked at the problem of we also looked at the algorithm of locality sensitive hashing as a potential solution to this problem right. And again just to recap this what we are given is we are given the version of the nearest neighbor problem, where the input radius is also specified right. So, now, we are solving an approximate version of this. So, we are given a radius r the input data as well as an approximation factor c and a confidence δ ok.

So, what we have to written is the following that if there is any query any point in your input data that lies within distance r of the query point right, then with probability $1 - \delta$ you have to return 1 that is a distance at most $c r$. So, c here could be is bigger than 1 right. So, we are returning an approximate near neighbor right. We do not need to return anything that is that is less than equal to r , but we certainly need to return any point that is less than equal to $c r$.

(Refer Slide Time: 02:37)

Locality Sensitive Hashing



The diagram illustrates the LSH process. It shows a 2D space with a point and a line representing a projection. The space is divided into buckets. The process involves hashing the point into buckets using k projections and L tables. The buckets are then sorted by distance to find matches. The results are shown as a list of buckets.

Given input data, radius r , approx factor c and confident δ

Output: if there is any point at distance $\leq r$ then w.p. $1 - \delta$ return one at distance $\leq cr$

Algo: Choose (k, L) .


do L times


- iid hash functions $\{h_{i1}, \dots, h_{ik}\}$
- Create hash table H_i by putting each x in bucket

$$H_i(x) = (h_{i1}(x), \dots, h_{ik}(x))$$

Store non-empty buckets in normal hash table

Picture courtesy Slaney et al.

 IIT Gandhinagar
Indian Institute of Technology Gandhinagar

 NPTEL ONLINE
CERTIFICATION COURSES

So, how did we do this, what we mentioned? And let us think about the when the distance metric is the Euclidean metric for now ok. What we said is that there are two main parameters of the locality sensitive hashing. First is k and the other is L ok. So, what do we have to do, so we will build L different hash tables right. As in this picture will build L different hash tables. And in order to create one hash table one of this hash tables we choose k hash functions right. Let us say we are building the i th hash table. Now, for that we need to choose h_{i1} to h_{ik} right. And each of them are IID and then being chosen from some hash family.

The hash family depends on the particular distance metric that you are interested in we talked about a number of different kinds of different kinds of metric and there corresponding hash families. If you do not, if you do not remember all of them just think about the Euclidean metric and the corresponding hash family. Now, for the Euclidean metric this is what we were doing. So, if we had the point, if we had if we had the input points, we were first drawing a line a random line in space right. Then we are projecting that line, then we are projecting the input point onto that line right. So, this is the projection of the point in this of this point on this line. And then we bucketised right.

So, we created these intervals of intervals of size w right and then and to return the id of the interval that this projection lies in ok. This is one of our hash functions h_{i1} . So, this gives me an interval number ok. And you have to create do this k times. And then you

have to concatenate them that creates that gives the index for the i th hash table right. So, the i th hash table is created by putting the point x in the bucket that is specified by this k table ok. So, of course there are potentially a lot many number of buckets, not all the buckets will have points in them. So, after we have created the hash i d for all the points, what we will do is that we will store these non empty buckets in a normal hash table that is we will use just a standard hash table to actually store the to actually instantiate the table h_i ok.

(Refer Slide Time: 05:09).

Locality Sensitive Hashing

Given input data, radius r , approx factor c and confident δ

Output: if there is any point at distance $\leq r$ then w.p. $1 - \delta$ return one at distance $\leq cr$

Query: Find out all points in buckets $H_1(q) \dots H_L(q)$ and return ones that are $\leq cr$

query time \propto # candidates.
space \propto L.

Picture courtesy Slaney et al.

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Now, what happens at query time, at query time again we will do the same. Given the query q we will find out which bucket q lies in the first hash table which bucket q lies in the second hash table, and similarly for 1 to l right. And then we will take the union of these candidates right. In the sense that we will take all the points that have fallen in the bucket $H_1(q)$, all the points that have fallen in the bucket $H_2(q)$ and all the points that have fallen in the bucket $H_l(q)$, we will take the union of them.

So, remember these buckets are not necessarily disjoint, because H_i each H_i contains all the data points right. All the data points have been mapped l different times. So, so H_i and then you take the union of $H_{i_1}(q)$, $H_{i_2}(q)$, $H_{i_3}(q)$, up to $H_l(q)$ right. So, these are your candidate near neighbors. Now, not all of them will be at distance less than equal to $c r$. So, what you will do is that you will go over this list of candidates. You will then compare each of these candidates to q itself. And throw away the ones which are further

apart which are whose distance from q is more than $c r$ right. So, then there are two issues right. So, the complexity the query time complexity depends on the number of candidates ok. And the space of the algorithm is proportional to l which is the number of hash tables ok. And our aim is to create it is to try to minimize these two ok.

(Refer Slide Time: 06:56)

Drawbacks

- Trading space with time, strongly super-linear space
 - Even in practice, typically 5-20 times more memory than dataset itself
- Space-time tradeoff mostly practical effective for medium-high dimensions, dense vectors
 - recent advances in ML about dense embeddings

$L = n^\rho$

query $\sim O(n^\rho)$

$\rho \propto f(c) \sim \frac{1}{c}$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, as we saw that there are a number of interesting space time tradeoffs that locality sensitive hashing does. And more importantly what it does allow us is to do the following that it gives the first provably sublinear query time algorithm for approximate nearest neighbors ok. So, so if you remember that what we could do is to set that L equal n to the ρ . And we could and we could also have a query time which is order n to the ρ right. And ρ is typically a function that that depends on c ok.

It could look like something like 1 to the power $1 + \frac{1}{c}$ sorry yeah it depend n the power $1 + \rho$, and ρ depends on $1/c$ order $1/c$ ok. So, the space is n to the power $1 + \rho$, and the query time is n to the ρ , and ρ is of the order of $1/c$. So, while theoretically this is a breakthrough in practice it has its problems. So, what are we doing here, we are trading off space with time right. Remember that the naive solution was to do a linear search. And a linear search space required would be n .

So, we are trading off space with time. And we have a strongly super linear space ok. And even in practice this shows the space required by a standard a typical locality sensitive hashing data structure is of the order of 5 to 20 times more than the more than

the memory taken for the data set itself. And this is sometimes a problem right, because the data sets are typically very large. And frankly while I mean theoretically it is a it is a it is an excellent algorithm and has found a lot of practice.

Most of the practical applications have lied in the regime of when the when the dimension is medium to high, when the dimension is of is of the order let us say from 100 to 1000 of that outer ok, and the vectors are fairly dense right; in that setting the space required the space time trade off that is achieved by the locality sensitive, by the locality sensitive hashing algorithm compares very favorably to the space required, and the time required for standard linear space right for standard linear search right for the knife linear search.

Beyond this if the if the dimension becomes very big or the data set, becomes very sparse knife search typically just beats localized sensitive hashing algorithms. There is an interesting sort of an interesting set of research here. In the sense that if you look at the recent lot of recent advances in machine learning, there really trying to get dense embeddings for sparse vectors right, and in that sense locality sensitive hashing still continues to be a very sort of appropriate technology for different machine learning algorithms.

(Refer Slide Time: 09:59)

Probing multiple times

- Idea: Can we reduce space while not affecting query time by too much?
 - need to hit buckets that have high probability of the containing the nearest neighbour

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, can we do better right? And since space is our main concern what we want to see is that can we reduce space. We have to do some tradeoffs. So, can we reduce space by

while not affecting query time by too much that I am willing to pay a little more, when we are doing the query right. Not as bad as a linear search, but maybe a little more than what I am sort of promising in the locality sensitive hashing algorithm, but I would not reduce the space drastically right.

So, so idea the basic intuition behind such data structures is to put multiple times right. So, imagine the sort experiment that suppose for a query right. We knew that which are the buckets that have the highest probability of containing the near neighbor of that query ok. So, natural algorithm that knows these probabilities for then go and probe these buckets right. So, the standard locality sensitive hashing algorithm is probing only 1 bucket per hash table. This ideal algorithm if it knew that some other buckets have none have a non trivial probability of containing the of containing the near neighbor, it will also probe that other bucket. So, it could probe multiple buckets per hash table can we implement this algorithm.

(Refer Slide Time: 11:20)

The slide is titled "Entropy based LSH" and contains the following text:

- Assume that we know $R(p, q) =$ distance from query q to nearest neighbour p
 - Buckets are a random partition of the data
 - The success probability of a bucket (i.e. of containing p) depends only on $R(p, q)$
 - Ideally, we can sort the buckets by this probability

There are two diagrams on the slide. The first diagram shows a 2D space partitioned into several irregular buckets. A point p is marked in one bucket, and a point q is marked in another. Red dashed lines indicate the distance from q to the nearest neighbor p in each bucket. The second diagram shows a point q and a point p with a red line segment between them labeled $R(p, q)$.

The slide footer includes the logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES.

So, the basic problem is as follows right. So, how do we know this particular probability distribution ok? So, computing this probability distribution is as hard as really doing a linear search again. So, can we can we approximate this probability distribution in a smart way. And this is the basic elegant idea behind entropy based LSH. This was developed by Rina Panigrahy in 2006.

So, what it says is as follows that assume we know the distance $R_p q$. So, so here is the query point, and this is the nearest neighbor p and this distance is $R_p q$, and assume that we know this distance ok. What see, the intuition is that the buckets are a random partitioning of the data right. And this random partitioning happens as follows that suppose here is the data right. Now, based on the particular based on the hash functions that were chosen the data gets partition like this ok. So and because of the properties of a locality sensitive hashing algorithm the success probability of a bucket right that is the probability that this bucket contains the point p right, depends really on the distance $R_p q$ ok.

So, ideally if I can sort the buckets with this probability right, if I if for every bucket, I could calculate the probability. If I knew $R_p q$ right, I could, if I could write on the probability that this bucket contains the near neighbor p right, and then if I could sort the buckets with this probability, then I would look at only the top few buckets because I do not really want to look at all the buckets, because that is too expensive. So, then I would only probe the top few buckets, and be done with it right, but calculating this probability exactly is again hard for all the buckets right. It is very expensive.

(Refer Slide Time: 13:30)

The slide is titled "Entropy based LSH [Panigrahy' 06]". It contains the following text:

- Elegant way to sample from the success probability distribution
 - Perturb the query point repeatedly and probe
 - Buckets that have high probability should come up often
 - Theoretical guarantee

To the right of the text is a hand-drawn diagram of a sphere. A horizontal line divides the sphere into two hemispheres. The top hemisphere contains several 'v' characters, and the bottom hemisphere contains several 'x' characters. A point 'q' is marked on the bottom hemisphere, and a point 'p' is marked on the top hemisphere. A line segment connects 'q' and 'p', passing through the center of the sphere. The distance between 'q' and 'p' is labeled as $R_p q$.

At the bottom of the slide, there are logos for "IIT Gandhinagar Indian Institute of Technology Gandhinagar" and "NPTEL ONLINE CERTIFICATION COURSES". The number "19" is in the bottom right corner.

So, what Panigrahy does is that he gives a very elegant way to sample from this probability distribution without actually calculating this probability for every bucket right. And the way to sample is as follows what he notes is that suppose we take the

query point, and we build the radius $R(p, q)$ we build the ball of radius $R(p, q)$ around the query point q . So, and then we sample multiple points at this radius $R(p, q)$. And then we hash these points using the same set of hash functions h .

So, what he notes is that buckets that, because the probability that a bucket contains the point contains a near neighbor p is dependent only at the only at the depends only on the distance $R(p, q)$. Then all points on the surface of the sphere have the same probability of falling in the bucket or in that particular bucket b , and in that case if I if I take multiple points from the from the surface of this sphere of radius $R(p, q)$, and look at the buckets that these points fall in b , then the buckets that are more likely to contain p will come up more often b .

So, what it means is that the algorithm is as simple as follows that take the query point q b make a guess on the distance $R(p, q)$ sample from the multiple times from the ball from this from the surface of this ball which is which at the ball of radius $R(p, q)$ around the query point q , and then look at the buckets that these that these points fall in b . And, now consider, and this is all happening in a single hash table. And then we take all the candidates that these buckets have.

All the query points are all the data points of this buckets have. And these are our candidates. So, technically if you even if you have had only one hash table b , you get to probe multiple buckets, and consider their candidates b . And Rina has a very nice theoretical guarantee on this entropy based LSH. We would not go into the details of the theoretical guarantee, however.

Student: (Refer Time: 15:53)

(Refer Slide Time: 15:55)

The slide is titled "Multi-probe LSH". It contains two bullet points: "Look at neighbouring buckets!" and "Consider LSH for L2". Below the text is a diagram of a number line with buckets of width w . A query point q is shown with its projection $q \cdot v$ onto the line. A bucket boundary is at b . A random shift $s \sim U(0, w)$ is indicated by an arrow. The bucket containing q is labeled $h_{v,b}(q)$. A point p is shown nearby. A small video inset shows a man speaking. The slide footer includes the IIT Gandhinagar logo and the text "NPTEL ONLINE CERTIFICATION COURSES".

Multi-probe LSH

- Look at neighbouring buckets!
- Consider LSH for L2

$h_{v,b}(q) = \left\lfloor \frac{q \cdot v + b}{w} \right\rfloor$

$h_{v,b}(q) - 1$

$h_{v,b}(q) + 1$

shift $\sim U(0, w)$

$q \cdot v$

b

w

p

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

What we will look into is a modification of the scheme ok. So, the modification of this scheme is something called multi probe LSH. And this is really the same idea in a slightly more efficient manner. What it says is as follows that look sampling from the sampling first from the surface of the ball, and then and then looking at the at the at the at the hash buckets is pretty expensive. What we will do instead is that we will look at the bucket that this query point has fallen into. And then we will perturb the idea of this bucket a little bit right.

So, what does this mean let us look at this in the in a very specific case, when we are when we have a LSH, when we are building a LSH table for l 2 right. Remember what we were doing there we were taking a random line right. This is the line v right; this is the query point; first we project q onto v right. So, this is $q \cdot v$ and then we bucketize them in buckets of size w , and apply a random shift. So this is a shift right that is drawn uniformly from 0 to w ok.

So, the so the hash value of q is this i d the i d of this bucket ok. So, then imagine that there is some other point p that is actually predict near to q ok. So, then if p has not been assigned the same idea as that of q right, then it is likely that it falls maybe either here or here that is if is not in the same bucket as q then, it is likely that it is in a neighboring buckets right, which means that the that the 2 buckets that if I do not find if p does not happen to be in thus in this bucket $h_{v,b}(q)$ right.

Then it is probably being assigned $h v b q$ plus 1 or $h v b q$ minus 1. These are two immediate sort of possibilities the most likely possibilities of the hash value of p under this hash function right. So, this is what we will try to face that instead of actually doing the perturbation on q and then figuring out for the hash buckets should be we will do the perturbation directly on the hash indices itself ok.

(Refer Slide Time: 18:30)

The slide is titled "Multi-probe LSH". It contains the following text and handwritten notes:

- Suppose $k = 3$
- $H_1(q) = (5, 8, 3)$
- We consider buckets that differ in one position, two positions, ...

Handwritten notes include:

- $k=3$ (underlined)
- A list of buckets: $(5+1, 8, 3)$, $(5-1, 8, 3)$, $(5, 8+1, 3)$, $(5, 8-1, 3)$, and a larger set $(5+1, 8+1, 3)$, $(5-1, 8-1, 3)$.
- A downward arrow pointing from the first two buckets to the last two.

The slide footer includes the IIT Gandhinagar logo and the text "NPTEL ONLINE CERTIFICATION COURSES".

So, so just to be very specific right supposing k equal to 3 ok. So, the first hash table has buckets that are tuples of 3 has id has bucket ids that are tuples of 3 ok. Supposing H_1 is the first hash table. So, and suppose q falls in H_1 of q which is 5, 8, 3. So, then the immediate next ones would be have to look at 5 plus 1, 8 3, 5 minus 1, 8, 3, 5, 8 plus 1, 3, 5, 8 minus 1, 3 you get the idea ok. And these are the and these are buckets are differ in only one of the positions from the query bucket.

Once we have exhausted them, and looked through them. Then we could also start to look at buckets at different two positions right. Maybe 5 plus 1, 8 plus 1, 3, 5 plus 1, 8 minus 1, 3 and so on and so forth ok. And then buckets that differ in three positions and so on. And this would eventually give me all the buckets right that differ in at most let us say, s positions ok. So, then, so, how do we do this in a sort of principled way ok? How many buckets do we get two should we really be searching for and is there a particular order in which we should between these perturbations and so on. So, in order to analyze this, let us formalize this a little bit right.

(Refer Slide Time: 20:05)

Formalizing

- $\Delta \in \{-1, 0, +1\}^k$ be a "perturbation" vector
 - E.g. $\Delta = (-1, 0, +1, +1, 0, \dots, -1)$
 - We get a new hash bucket by doing $H(q) + \Delta$
 - Say Δ has at most S nonzeros
 - Number of possible Δ is: $\binom{k}{S} 2^S$
- Is there a natural way to order these buckets for searching?

Handwritten notes: $k=3$
 $\Delta \in \begin{pmatrix} -1, 0, +1 \\ -1, -1, 0 \\ +1, +1, 0 \end{pmatrix}$

IIT Gandhinagar Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES
13

So, let me consider this delta called delta to be a perturbation vector. So, it is something very simple right. So, suppose in the case when k equal to 3 right, the deltas look like let us say minus 1, 0, 1 plus 1, minus 1, minus 1, 0, plus 1, minus 1, 0 and so on right which means that it is basically all possibilities having where each bucket is being perturbed either by a plus or minus ok. And there are k and some bucket might not be perturbed right in that case some bucket id might some specific id might not be perturbed in which case you have a 0 in delta.

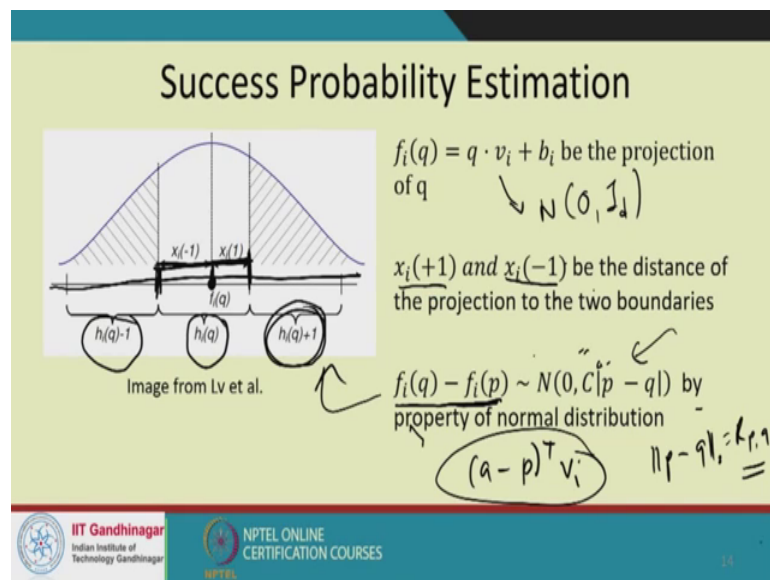
So, given and given a bucket $H(q)$ we get a new hash bucket by doing $H(q) + \Delta$ right and suppose delta has at most S non-zeros. So, this S will be a hyper parameter that is kind of chosen based on the amount of time you want to spend in query and so on right. So, for instance think of S equal to 1 right that is I only want to look at buckets that differ from the original bucket in only one position right. Then what is the and if and for a general S , then what is the possible number of this delta right. So, the possible number of this delta is that you have to choose S positions from K positions that happens in K choose S right. And each of these positions gets a plus 1 or minus 1, so that is 2 to the S right.

So, once you fix S , the number of possible perturbations you can get is K choose S times 2 to the S . And this is for a single hash table right we could be if there are multiple hash tables then we should really be doing this independently for each of the hash tables ok.

So, this is fairly large right. Once, once S gets to be a little big this increases quite fast. So, it is that a natural way to order this buckets for searching right I mean you could you could potentially sort of already have an idea about this. If you say that once you start thinking about that that if a if a point if a neighboring point is really close to q, then it is more likely that it will fall in a bucket of 0 or 1 perturbations than two perturbations.

So, we should really search the buckets with 0 that has at most one perturbation before we search the buckets that have at most two perturbations maybe. And we should search one perturbation and do perturbation buckets before we go to searching three perturbation buckets right. But it is there a better way to structure this search?

(Refer Slide Time: 22:44)



So, here is a very interesting way right that was proposed by the by the original paper of multi-probe LSH. And what this says is that let us look at let us look at the actual projections of the points right that is suppose this was a line v that we were drawing before right. And this is the projection of the of the point q , this is a projection of the query point q right. So, if i of q so this is the i th, so this is the i th bit let us say i th hash index i d and this is obtained by $q \cdot v_i + b_i$. So, b_i is the random shift ok.

So, now, v_i is chosen according to uniform distribution on this sphere if you remember. Equivalent way of looking at it v_i is chosen according to n zero 1_d which means that every index of v_i , v_i is a vector of d dimensions. So, same dimension as that of q . And every index is a Gaussian distribution from $N(0, 1)$ over square root t ok.

Now, if I know this then I also know the distribution, if some I mean if some point p if the projection of p is a $f_i p$ right, then I can actually write down what is the distribution of $f_i p$ minus $f_i q$ ok. So, the so remember that $f_i p$ minus $f_i q$ is really a random variable because of the because of the random choice of v_i right and b_i right, v_i and b_i are the random variables here. So, therefore, the distance $f_i p$ minus $f_i q$ right is really $v_i q$ minus p transpose v_i right this is this random variable $f_i q$ minus $f_i p$.

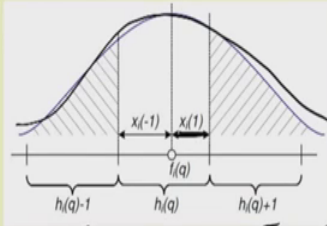
And it is if you know if you know about the properties of run of normal distribution. You can easily say that this particular random variable right this particular random variable has a distribution of this form that is it has mean zero right and the variance is dependent on the l_2 norm of p minus q that is on the distance $R_{p, q}$ the variance depends on that right. So, so which so what; that means is as follows that let me draw the let me draw in this in here that here is the here is $f_i q$ right and I am drawing the distribution of the random variable $f_i q$ minus $f_i p$ and. So, it is a Gaussian that is centered at $f_i q$ right and the variance depends on this on the on the norm p minus q ok.

So, so notice so and let me define two more quantities let us say that these are the boundaries remember with bucketize, the bucketize these projections right into buckets of length w right. So, so let us say that these are these are the boundaries of this bucket. So, this is the bucket $h_i q$ by definition this is the bucket $h_i q$ minus 1 and this is the bucket $h_i q$ plus 1 right and so and so these are the boundaries. So, let us say so let us say the distance from the distance of $f_i q$ to this left boundary is denoted by x_i minus 1; and the distance of $f_i q$ to the right hand side boundary is denoted by x_i 1 right.

So, so what you can now intuitively see is that the probability that the probability that this particular $f_i p$ lies to the right hand side lies to lies in the bucket $h_i q$ plus 1 will depend on this distance x_i 1 x_i plus 1. And the probability that $f_i p$ right lies to the left hand side lies in the bucket $h_i q$ minus 1 will depend on x_i minus 1 right. So, let us do this formally right. And by doing so we will be able to assign some kind of likelihood some I mean we will be able to estimate the success probabilities of each of the buckets $h_i q$ minus 1 and $h_i q$ plus 1 ok.

(Refer Slide Time: 26:59)

Success Probability Estimation



$x_i(+1)$ and $x_i(-1)$ be the distance of the projection to the two boundaries

$f_i(q) - f_i(p) \sim N(0, C|p - q|)$ by property of normal distribution

$\Pr[h_i(p) = h_i(q) + 1] \approx \exp(-Cx_i(+1)^2)$

Image from Lv et al.

15

So, if I have to write down this probability right, you could say that the probability that $h_i(p)$ is equal to $h_i(q) + 1$ will then depend on exactly the PDF of the normal distribution, which is \exp of minus some C times x_i plus 1 square because this distance and the C will depend on the radius also ok. So, so that means, so what; that means, is that the C will depend on the distance p minus q right. So, what that means is that now we can assign success probabilities to the perturbations plus 1 to the perturbations plus 1 and minus 1 for this particular index $h_i(q)$ ok.

(Refer Slide Time: 27:54)

Ordering buckets

- If $\Delta = (\delta_1 \dots \delta_k)$ then $\delta_i \in \{-1, 0, +1\}$

$$\Pr[H(p) = H(q) + \Delta] = \Pr[\prod [h_i(p) = h_i(q) + \delta_i]]$$

$$\approx \prod \exp(-Cx_i(\delta_i)^2) = \exp\left(-C \sum x_i(\delta_i)^2\right)$$

Ex: $\Delta = (+1, 0, -1)$,

~~$x_i(\delta_i)$~~
 $x_1(+1)^2 + x_2(0)^2 + x_3(-1)^2$

15



Let us write down this in a slightly more formal. Again that supposing the perturbation vector capital delta is delta 1 to delta k. Remember that each of the delta is either minus 1, 0 or plus 1 ok. Then what I could write down is that the probability that h i p equals h i q plus delta right is nothing but the product of the probabilities because each the for each i the hash function h i is independent of each other. So, if the product of the probabilities that h i q equals h i h i h i p this should be p h i p equals h i q plus delta i and again delta is either minus 1 or plus 1.

And by the previous calculation that we did that this depends on really the distance of f i q of f i q from the corresponding boundary right, which is given as x i delta i right. So, it is the product I mean this can be this can be I have kind of hand waved this a little bit, but this can be written as a product of the exponentials of minus C x i delta i square which the product translates to the to the exponent, and we give and we get a exp of minus C summation x i delta i square right. For instance if delta is plus 1, 0, minus 1, you would get x 1 plus 1 square plus x 2 0 square plus x 3 minus 1 square ok, this would be the exponent. So, now this allows us to define a score right.

(Refer Slide Time: 29:29)

Ordering buckets

- Define $score(\Delta) = \sum x_i(\delta_i)^2$
- Lower the score, higher the probability of p being in the bucket
- Order the buckets by the score and search them in this order

 IIT Gandhinagar
 Indian Institute of Technology Gandhinagar
  NPTEL ONLINE
 CERTIFICATION COURSES


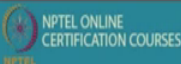
A score for each perturbation vector delta right; and that score just depends on this on the quantity in the in the exponent right. Because we want to maximize the success probability you gone to minimize the summation the summation x i delta is square right, because if this is minimized right, then the success probability will be maximized, the

lower the score the higher the probability of p being in this bucket. So, then what we do is to order buckets by this score and search them in this in this order.

(Refer Slide Time: 30:04)

Query directed ordering

- When a query q arrives
 - Calculate $H(q)$
 - Calculate $\{x_i(+1)^2, x_i(-1)^2, i = 1 \dots k\}$
 - Sort

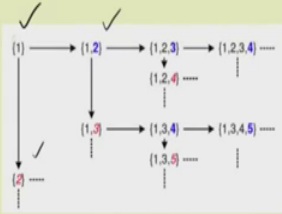


19



So, in a particular query q arrives right, we then create the order in which the buckets and I mean we then want to sort the buckets according to the score delta right. And remember that the score delta depends on the query q . So, how do we do this efficiently? First we calculate the $x_i + 1$ and $x_i - 1$ square for i equal to 1 to k and then we sort them.

(Refer Slide Time: 30:29)

Query directed ordering

- When a query q arrives
 - Calculate $H(q)$
 - Calculate $\{x_i(+1)^2, x_i(-1)^2, i = 1 \dots k\}$
 - Sort (call these as $z_1 \leq z_2 \dots \leq z_{2k}$)
- Start with $A = \{1\}$
- Repeatedly do either *shift* or *expand*
 - *shift* replace $\max(A)$ by $1 + \max(A)$
 - *expand* adds $1 + \max(A)$ to A



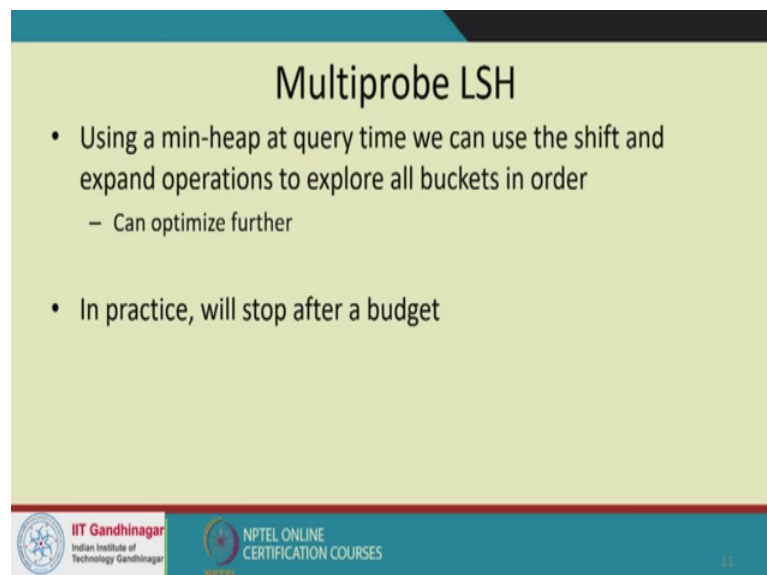


20

So, let us say that the sorted values as z_1 to z_{2^k} . So, there are 2^k values because each index gets either a plus 1 or minus 1. And beyond that it is really a sort of mean heap based algorithm right. Basically what we will do is that let us say that i equal to one right has the minimum $x_i \Delta_i$ right the z_1 , the z_1 one contains the minimum $x_i \Delta_i$ of these of these values. So, then we start with that right.

And then we either at every step we either keep on expanding that right either add the very next one right I mean for instance if 2 is the next highest, if 2 is the next smallest value, either we go from the set one which is the bucket id 1 to the bucket id 1 2. So, it is a bucket id 2 would be the next one to explore or we go from bucket id 1 and we replace that by bucket id 2 right.

So, at every step right so we could define a search procedure in which at every step we either go to replacing the max of the max the max value with the next max value or we augment it with the next max value right. So, this gives us a way to search through all the possible buckets right. So, we first search bucket 1 and then we either first search bucket 1 or bucket or bucket, bucket 1 2 together and so on and so forth ok. So, in practice, we kind of have a query time budget and we stop after that ok.

(Refer Slide Time: 31:52)



Multiprobe LSH

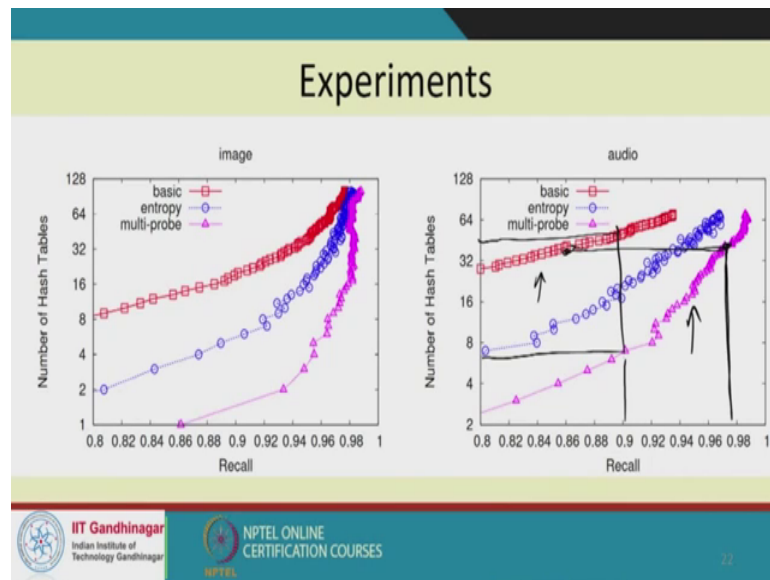
- Using a min-heap at query time we can use the shift and expand operations to explore all buckets in order
 - Can optimize further
- In practice, will stop after a budget

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

21

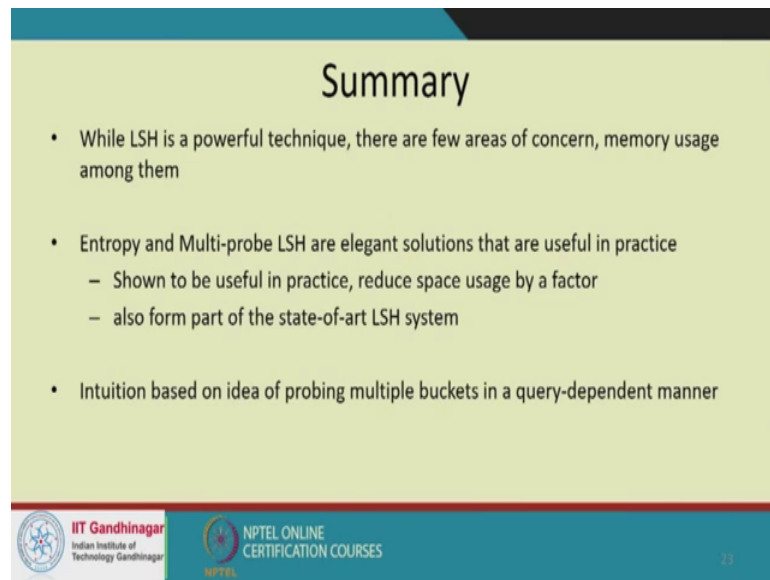
(Refer Slide Time: 32:01)



So, in terms of experiments what this shows is that this is something that has been that is really that effective in reducing the space. For instance, I will just point you to one of the plots in one of the plot in the in the y-axis is the recall which means the fraction of near neighbors that we have been able to you know let us say the fraction of the nearest neighbors that we have been able to return using LSH. And on the y-axis is the number of hash tables.

If you see that the multi probe plot right uses much less hash tables than let us say the basic LSH plot right, and for the and it is able to let us say this point right. So, so using the same number of hash tables this gives a much higher recall or rather if you want to look at it this way that if I want to achieve point nine recall number of hash tables we need here is let us say only 6 or 7 number of hash tables we need here is of the order of 32, 33 and the basic LSH ok.

(Refer Slide Time: 32:56)



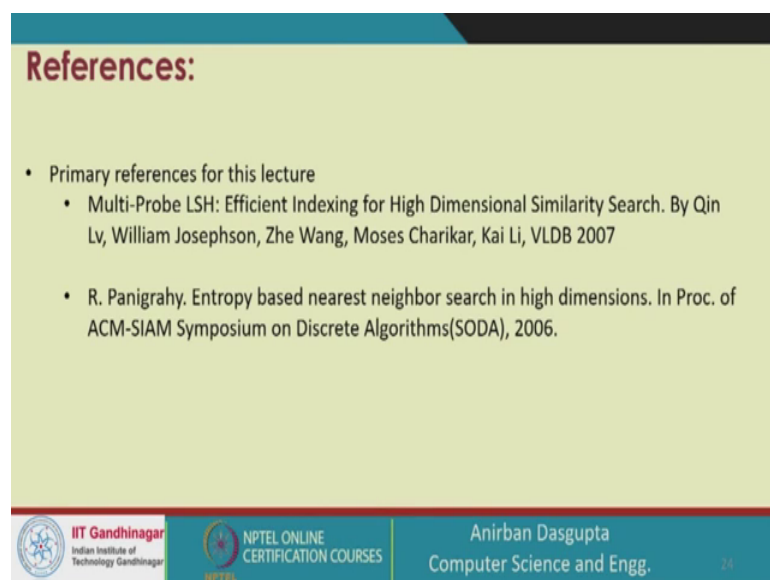
Summary

- While LSH is a powerful technique, there are few areas of concern, memory usage among them
- Entropy and Multi-probe LSH are elegant solutions that are useful in practice
 - Shown to be useful in practice, reduce space usage by a factor
 - also form part of the state-of-art LSH system
- Intuition based on idea of probing multiple buckets in a query-dependent manner

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL NPTEL ONLINE CERTIFICATION COURSES 23

So, just to summarize the LSH is a powerful technique, but the few areas of concern, memory usage among them. The entropy and the multiple of LSH elegant solutions that are very useful in practice, and they and they also form part of the state-of-art locality sensitive hashing system. And the basic intuition that you should remember is that we want to probe multiple buckets in a query-dependent manner.

(Refer Slide Time: 33:18)



References:

- Primary references for this lecture
 - Multi-Probe LSH: Efficient Indexing for High Dimensional Similarity Search. By Qin Lv, William Josephson, Zhe Wang, Moses Charikar, Kai Li, VLDB 2007
 - R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In Proc. of ACM-SIAM Symposium on Discrete Algorithms(SODA), 2006.

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL NPTEL ONLINE CERTIFICATION COURSES Anirban Dasgupta Computer Science and Engg. 24

Primary reference for this lecture was this paper by on Multi-Probe LSH by Qin Lv, Josephson, Wang, Charikar and Li which appeared in VLDB. And also the paper by Rina Panigrahy on the entropy based LSH and.

Thank you.