

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

Lecture – 14c
Fast LSH + Sparse Random Projection

Hello everybody and welcome to the course of Scalable Data Science I am Anirban and I am from Gandhinagar. So, today we will be talking about two different, but related algorithms; the first will be about an application of the fast Johnson Lindenstrauss transformation to locality sensitive hashing ok. So, here is the first slide for us.

(Refer Slide Time: 00:46)

Johnson Lindenstrauss Lemma [JL84]

$\epsilon > 0, k \geq \frac{C}{\epsilon^2} \log(n)$. There exists a **linear mapping** A such that whp, for all (i, j)

$$(1 - \epsilon)|x_i - x_j| \leq |Ax_i - Ax_j| \leq (1 + \epsilon)|x_i - x_j|$$

$x_i \in \mathbb{R}^d$

The slide contains a diagram illustrating the Johnson-Lindenstrauss Lemma. It shows a set of points in a high-dimensional space (left) being mapped to a lower-dimensional space (right) via a linear mapping A. The mapping preserves the relative distances between the points, as indicated by the inequality above. The diagram shows a set of points in a 2D space on the left, which are then mapped to a 1D space on the right. The mapping is represented by a blue arrow pointing from the left to the right. The points are represented by blue dots. The axes are represented by blue lines. The mapping A is represented by a red arrow pointing from the left to the right. The points in the lower-dimensional space are represented by red dots. The axes in the lower-dimensional space are represented by red lines. The text $x_i \in \mathbb{R}^d$ is written to the right of the diagram. The slide also features the IIT Gandhinagar logo and the NPTEL ONLINE CERTIFICATION COURSES logo at the bottom.

So, just to recall Johnson Lindenstrauss transformation was this particular linear mapping that preserves the distances between any two pairs of points. In the sense that if we are given a set of points x_1 to x_n right. And if I come up with a with a mapping A that let us say is either created out of independent Gaussian entries or according to the first Johnson Lindenstrauss transformation which will recap in a moment right. And it is A is of appropriate size for instance; A is of size at least some constant by epsilon square log n right for a suitable constant which happens to be pretty small in practice. And now, if I get image of each of the points x_i by multiplying it by A right and the x_i lie in d dimension right. So, A is of size k by d in order for us to be able to do the multiplication.

Then the Johnson Lindenstrauss lemma claims start with a very high probability, the distance between the l_2 norm between Ax_i and Ax_j will lie between within a $1 \pm \epsilon$ factor of the distance between x_i and x_j . So, the lemma as it is stated here is about existence, but as we have seen we can claim this that this will happen that for a particular once for a particular random sample of a this happens with a very high probability.

(Refer Slide Time: 02:15)

The slide is titled "Time taken for projection" and contains the following text:

- Projection matrix out of Gaussian or iid $\pm 1 = O(kd)$
 - $k = \Omega\left(\frac{1}{\epsilon^2}\right)$
- FJLT : projection matrix is $\frac{1}{\sqrt{d}}PHD$
 - H is the $d \times d$ Hadamard matrix
 - D is a random ± 1 diagonal matrix
 - P is a sparse Gaussian matrix
 - Time = $O(d \log d + k \log(nd))$

Handwritten notes in red ink include:

- A matrix $\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$
- The equation $H_d = \begin{pmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{pmatrix}$
- The expression $\frac{1}{\sqrt{d}}H$ circled in red.

A blue speech bubble points to the $\frac{1}{\sqrt{d}}$ term in the FJLT definition, containing the text: "Notice the normalization $\frac{1}{\sqrt{d}}$ ".

The slide footer includes the logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset showing a person at a computer.

So, just to recap again what is the time taken for projection? The time taken for if the random matrix are selected out of independent Gaussian entries or are from independent plus minus 1s and 0s the time taken for projection is order $k d$, which is basically the time taken to multiply k by d matrix with the vector of length t right. And remember that this k was at least $1/\epsilon^2$ right in fact, it had the $\log n$ factor or the $\log 1/\delta$ factor depending on what version, I mean whether you want it to hold for all pairs with high probability or whether you want to hold for a single for every single one with probability $1 - \delta$ ok.

So, basically the success probability has to come in to the log. The other construction that we saw was this fast Johnson Lindenstrauss transformation and here I have some correction to make for the previous video what we had mentioned is that the fast Johnson Lindenstrauss transformation has three components right. The first is the plus minus 1

diagonal matrix right, where every entry of d is generated from a random coin toss that gives you plus with probability half and minus one with probability half.

Next comes the Hadamard matrix right and we had defined the Hadamard matrix in the form that the H_2 is $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ right, H_4 is obtained by doing $H_2 H_2 H_2$ and minus H_2 right. But, we had forgotten out here was the normalization right because, if you had been paying attention that we always need I mean in the original Johnson Lindenstrauss we always need to normalize. So, that in expectation it becomes a matrix so, that it basically preserves the norm in expectation right so, it is unbiased the in terms of the l_2 square norm.

So, that is why we what we want to do is instead of take this particular H which essentially is the unnormalized Hadamard matrix and then do $1/\sqrt{d}$ times H right. This is what we call the Hadamard matrix, this is traditionally what is known as a Hadamard matrix that you have you create it using this recursive definition and then you normalize it. So, that the length of every column is exactly 1 ok the l_2 length of every column is exactly 1 ok.

So, what we will do is when I now onwards refer to the Hadamard matrix it will refer to this normalized Hadamard form and then we would not really keep this. We would not really maintain we would not really keep this $1/\sqrt{d}$ hanging around, but you have to understand that this is the normalized version of the Hadamard matrix ok. And finally, so, the final component was this was this P which was a sparse Gaussian matrix ok. And so, the time taken was given by order $d \log d$ which is the time taken to calculate the Hadamard transformation plus the time taken to calculate to multiply P with the dense vector that results from HDx and that time is $k \log n$ because P is sparse, P has small number of nonzero entries ok.

(Refer Slide Time: 05:40)

The slide is titled "Densification claim". It contains the following text and formulas:

- $x \in \mathbb{R}^d, \|x\|_2 = 1$
- Claim: $\max_i |(HDx)_i| \leq O\left(\frac{\log(nd)}{d}\right)^{1/2}$
- Application of Chernoff style tail inequality per coordinate and union bound

Handwritten notes in red ink include:

- A circle around $\frac{1}{\sqrt{d}}$
- $\sum x_i^2 = 1$
- $\frac{d}{\log(nd)}$

The slide footer includes the IIT Gandhinagar logo, the text "IIT Gandhinagar Indian Institute of Technology Gandhinagar", the NPTEL logo, and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset shows a man speaking.

So, and the densification claim that we had just to remember that again recall that here H refers to the normalized Hadamard matrix. The densification claim that we have is that for any fixed x right that satisfies $\|x\|_2 = 1$; if you look at the maximum coordinate of HDx that is that I mean will the average coordinate is $1/\sqrt{d}$ right. Because, there are d coordinates and they all sum up to $\sum x_i^2 = 1$ right. I mean summation $x_i^2 = 1$ right.

And then and then d of them say for the therefore, in some sense average coordinate $1/\sqrt{d}$ what this is saying is that the maximum coordinate is more is not much bigger than this, the maximum coordinate is $O\left(\frac{\log nd}{d}\right)^{1/2}$ right.


For this implies is that the vector HDx is basically dense right, because, if the maximum coordinate is $O\left(\frac{\log nd}{d}\right)^{1/2}$ which means that there has to be a significant number of nonzeros right. In some sense there has to be $O\left(\frac{d}{\log nd}\right)$ nonzero entries right because, they have to sum up to 1. So, and this you had obtained we had mentioned although we had not seen the proof as this can be obtained by some Chernoff style tail inequalities fine.

(Refer Slide Time: 07:01)

Projecting a dense vector

$$y = HDx, \max_i |y_i| \approx O(\sqrt{\log(nd)/d})$$
$$P = \begin{cases} 0, & \text{w.p. } 1 - q \\ N\left(0, \frac{1}{\sqrt{q}}\right), & \text{w.p. } q \end{cases}, P \in \mathfrak{R}^{k \times d}, k = O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$$

$z = \frac{1}{\sqrt{d}}PHDx$ is the final projected vector



IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

(Refer Slide Time: 07:08)

Fast JL Transform [AC09]

If $q = O(|x|_\infty^2) = O\left(\frac{\log(nd)}{d}\right)$, $\frac{1}{\sqrt{d}}PHD$ satisfies JL property

Calculating $y = PHDx$ takes time $O(d \log d + k \log(nd))$, potentially much faster than original Gaussian construction

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Let us so, what we had seen is that calculating PHDx takes time order d plus $k \log nd$. Now the question is that let us see an application of this and we will see an application in very interestingly in something that we have covered before in locality sensitive hashing ok.

(Refer Slide Time: 07:24)

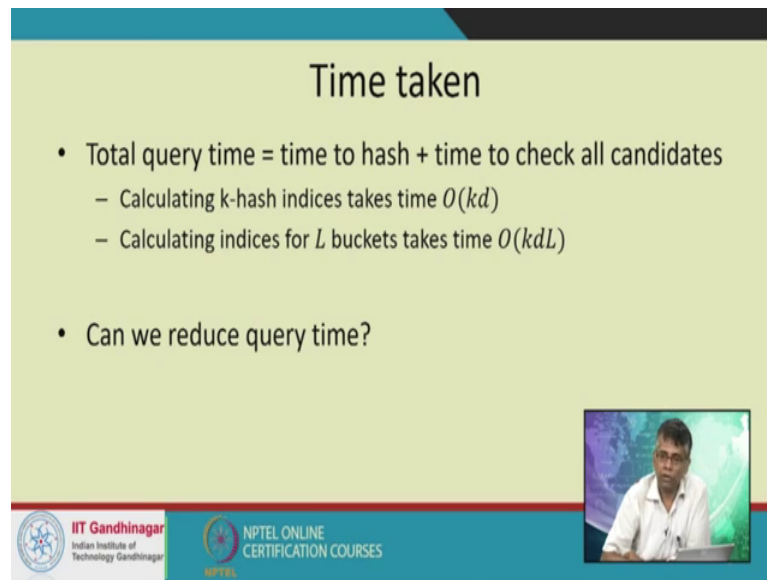
The slide is titled "Locality Sensitive Hashing" and includes a diagram on the left showing "K projections" and "L tables" leading to "Hash", "Find in buckets", "Sort matches by distance", and "Results". Handwritten red notes include "l2 H = (h1, 0)" and "w/c". The text on the right says: "Given input data, radius r, approx factor c and confident δ ". Below that, it says: "Output: if there is any point at distance $\leq r$ then w.p. $1 - \delta$ return one at distance $\leq cr$ ". A red circle highlights the formula
$$h_i(x) = \left\lfloor \frac{x \cdot v_i + b_i}{w} \right\rfloor$$
. At the bottom right is a small video inset of a man speaking. Logos for IIT Gandhinagar and NPTEL are at the bottom.

Just to recall what locality sensitive hashing and let us say that we right now we have been locality sensitive hashing for the l 2 distance ok. So, let us see that suggests to recall what we had done is that the ith index of a particular hash table, the I mean one hash table capital H, let us say we are talking about only creating only one hash table, one hash table is created by a k a tuple h_1 to h_k right.

And let us talk about creating let us say the ith index of this the ith value here of this tuple. So, h_i so, to create h_i of x we first choose a random vector v_i right, we take the dot product of the projection of x with v_i the dot product of x with v_i right. We add a shift b_i , where b_i is a uniform the chosen bit in the range from 0 to w for some hyper parameter w and then we divide it by w and then we return take the floor of this.

So, in some sense what we are returning is this and is the index of this, we are we have chopped off the, we have chopped off the picture the projection into these buckets of length w right. And then b_i you can think of as giving a random shift to this chopping off right, as in we do not really need to start at 0, we sort of add a random shift of b_i and then start the chopping off. And then we return the index of the bucket into which this particular projection of x falls into.

(Refer Slide Time: 08:50)



The slide is titled "Time taken" and contains the following content:

- Total query time = time to hash + time to check all candidates
 - Calculating k -hash indices takes time $O(kd)$
 - Calculating indices for L buckets takes time $O(kdL)$
- Can we reduce query time?

At the bottom of the slide, there are logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES. A small video inset shows a man speaking.

So, and we want and so, the time taken to calculate each of all the k hash indices is k times d because, the time taken with algorithm one of these indices is d right. So, and therefore, for L buckets right the time taken is $k d L$. So, can we reduce this right because if you remember I mean reducing this time so, what is the query time? The query time for LSH has 2 components right.




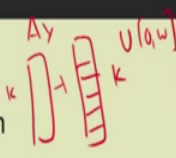
The query time for LSH has in component that says that you have to calculate all the hash functions, it also is a component that says that to calculate all the hash functions and find out the bucket that the query falls into you then get to go into the buckets and find out the candidates and compare the distance with each of the candidates ok.

So, then can we reduce this particular time can we at least reduce the time taken to calculate all the hash functions and why we talking about this because, now we know that we have some idea of because this kind of looks a lot like random projections right. So, and we know a faster ways of doing random projections so, can we utilize this right it is a natural question to ask.

(Refer Slide Time: 09:59)

Creating hash indices

- Looking at LSH as random projection + quantization
 - $A \in R^{k \times d}$ is a Gaussian JL matrix, $b \in R^k$
 - We first project and then bucketize
 - calculate $\left\lfloor \frac{Ax+b}{w} \right\rfloor$, k-index key calculated at once
- Time taken by matrix-vector multiplication = $O(kd)$ per hash table



In order to formalize the algorithm, let us first draw the connection between random projections and locality sensitive hashing in the 1 2 case a little bit more tightly right. And the connection is its pretty it is pretty intuitive right, we can basically look at LSH as a random projection happening onto k dimensions plus a quantization happening on each of the on each of the dimensions right. So, what is happening that you can think of I mean remember for each of the coordinates. So, multiplying the query we multiplying the data point x by a random vector v_i , think of creating a matrix A by just stacking these stacking these random vectors v_i in the rows right.

So, each row of a is one of the v_i is that we were using before right and then and then think of multiplying A with the with the vector x , we now get a k dimensional vector Ax right so, this is a k dimensional vector. Now we add a random I mean instead of choosing a single value b_i right, just choose I mean just choose one b_i for each of the coordinates just create a k dimensional vector with a with a with a random choice b_i for each of the coordinate that lies in 0 to w right. Then add these two so, a calculate $Ax + b$ this is again our k dimensional vector right. Now bucketize it each of the coordinates independently and you are done right, you in one in one sort of fell swoop of a of a matrix vector multiplication plus some shift and now you have the entire k sized index for a single hash table and this is I am not saying anything new with this, but this is what we have been doing. So, the time taken by this I mean so, the time taken remains the same because, the procedure is the same and so, it is $k d$ per hash table.

(Refer Slide Time: 11:49)

Collision Probability

- $p(u) = \Pr[h_i(p) = h_i(q)]$ when $|p - q|_2 = u$
- $p(u) = \int_0^w \frac{1}{u} f\left(\frac{t}{u}\right) \left(1 - \frac{t}{w}\right) dt$, $f(v) = \text{pdf of } |N(0,1)|$
- This is decreasing with increasing u

Handwritten notes:
 $\frac{u}{p} \quad q$
 $\phi \sim N(0, \sigma^2)$
 $\sigma^2 = \frac{q \cdot v_i}{p - q_i}$

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

Now, now you see the connection can we make this faster, so if I may want to make this faster I want to preserve some certain properties what is it that we that we want to preserve right. So, to do this let us step back a little into LSH and see that what was the important point quantity that we were talking about when we were analyzing the I mean how good of a locality sensitive hash function was this particular algorithm right, how do we analyze the quality of locality sensitive hash function? And if you remember we were analyzing it by looking at the collision probability.

So, what is the collision probability? The collision probability in a short sort of sentence is to say that that supposing two points are at distance u right, p and q on a distance u in terms of L_2 distance and I want to see what is the probability that they will collide. So, while we have not written this down exactly in the in the previous discussion for this particular for this particular construction you can actually write down a sort of nice form for the collision probability and I will just give an intuition for this right.

So, so, in the case that in the case that $h_i(x)$ is this particular quantity h_i I mean is the so h_i of x is this particular quantity right. We can write down what is the probability of collision if two points are a distance u right and the way we do this is as follows, I will just give you the an intuition for this. So, remember that x times v_i so, what is x times v_i or rather p times v_i versus q times v_i right.

So, if you look at the random variable $p \cdot v_i - q \cdot v_i$ right that is a Gaussian random variable right, because of the two stability property of Gaussians that we have seen. There is a Gaussian random variable and has variance that is distributed with the standard deviation that that scales with the with the L_2 distance between p and q ok. So, $p \cdot v_i - q \cdot v_i$ is a Gaussian variable with variance $p - q$ L_2 $p - q$ distance square ok.

So, then I can actually write down what the what the PDF of this random variable s right because, suppose f of v is the p is a PDF of the absolute value of n 01 right. Therefore, the PDF of this of let us say the if I multiply v by α the PDF of this quantity, the PDF of the random variable $f \alpha v$ is actually 1 by α $f v$ by α say. So, let me clean this up a little bit and so, that we can see better so, the PDF of αv is 1 by α f of v by α .

(Refer Slide Time: 14:34)

Collision Probability

• $p(u) = \Pr[h_i(p) = h_i(q)]$ when $|p - q| = u$

• $p(u) = \int_0^w \frac{1}{u} f\left(\frac{t}{u}\right) \left(1 - \frac{t}{w}\right) dt$, $f(v) = \text{pdf of } |N(0,1)|$
 $\text{pdf of } \alpha v = \frac{1}{\alpha} f\left(\frac{v}{\alpha}\right)$

• This is decreasing with increasing u

The slide includes a diagram of a line segment of length w with points t and w marked, and a handwritten note $(p-q) \cdot v_i$ with a downward arrow. There are also some scribbles in the top left corner.

And you can see this if you just think about it for a couple of minutes because, you are sort of in some sense multiplying the α is just reducing the scale it is just changing the scale of the random variable right. And then so, and so, the standard deviation changes and that is what is reflected here ok.

So, then what this is saying is that the probability of collision right that I mean if two points are a distance u right this part right kind of measures that what is the I mean what is the for a particular value of t right. What is the chance that the random variable p

minus q dot v i right, which is which is a random variable with this PDF what is the what is the total probability that p minus q dot v i lies in the range from 0 to t ok.

And this part kind of measures right that what is the chance that w right that the b or rather the b right so, what is the chance that b actually happens to drop a separator happens to remember what is b doing, b is just shifting these boundaries of these of these intervals right. I mean the boundaries could start at 0 or if you add I mean b to everything boundaries could start at b right.

So, you can think of b as just as just playing around as just randomly the separators between the intervals right. So, then what this part of it is the probability that b is chosen such that separator drops between 0 to t right or rather a separator does not drop between 0 to t right. Because the probability of that is that a separator drops only in this interval from t minus w to from t to w which is of size t minus w by w , which is exactly 1 minus t by w right.

So, this kind of captures the probability exactly that there is a collision between that $h_i p$ equals $h_i q$. let us know how to see that this is decreasing with decreasing with increasing u .

(Refer Slide Time: 16:41)

ACHash [DKS11] $H(x) = (h_1(x), h_2(x))$

- When calculating a k -tuple hash bucket index
 - $\lfloor \frac{Ax+b}{w} \rfloor$, use $A = PHD$
 - $q \approx O\left(\frac{\log(d)}{d}\right)$
 - Projection time = $O(d \log d + kL \log^2 d)$

$k \sim \frac{1}{2} \ll \log d$
 kd

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
 NPTEL ONLINE CERTIFICATION COURSES

So, now, and this is what we want to say that even if now why is this useful what we want to say is that if I do something other than the independent Gaussian. I want to make

sure that this collision probability does not suffer that the collision probability does not increase beyond what it was in the independent Gaussian case ok, this is how we are going to analyze this.

So, this comes from a paper of ours (Refer Time: 17:08) called the AC hash named after Ailon and Chazelle what we say is that that this is that our original sort of LSH was essentially calculating Ax plus b and then quantizing each of the each of the dimensions by I mean using these buckets using these intervals of size w , where A was a Gaussian matrix simply replace A by the Johnson by the first Johnson Lindenstrauss matrix. Which compose which is now composed of P then the normalized Hadamard and normalize Hadamard matrix and the diagonal matrix t ok, instead of instead of the Gaussian matrix just use P and choose q to be of more or less of the same type same order as we were choosing the original q in the f del t construction basically like $\log t$ by d I mean it is a little different slightly different, but not, but basically the same.

Now the projection time remember when A was Gaussian the projection time was $k d$ and now the projection time becomes $d \log d$ plus $k \log$ square d exactly there is ld construction ok. So, the projection time technically has improved right because, k is 1 by ϵ squared and so, it is potentially much less than $\log d$ for all practical values of d and ϵ ok.

(Refer Slide Time: 18:26)

$p(\tilde{H}(p) = \tilde{H}(q)) = p_{AC}(u)$ ACHash $p(\tilde{h}_i(p) = \tilde{h}_i(q)) = p(u)$

$p_{AC}(u) =$ probability that a k -tuple hash bucket has same value for two points at distance u $p(H(p) = H(q)) = p(u)$

We can show that

$$-(k+1)\delta + p^k((1+\epsilon)u) \leq p_{AC}(u) \leq p^k((1-\epsilon)u) + (k+1)\delta.$$

i.e. collision prob does not change much

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

So, but that is not enough now, we have to argue that this particular construction that we have that we have sort of done using LSH has the same properties as the original LSH construction has had in terms of the collision probability right. And this is not an easy argument right because now, because now what has happened is that the I mean just intuitively the different sort of indices of a particular hash function let us say if a particular hash table H_x is now h_1 of x times h_2 of x h_1 of x and h_2 of x are no longer independent of each other because of the particular projection $f \circ g \circ l \circ g$ construction that we have done. So, that complicates the analysis quite a bit, but what we are now still able to say is that if you take see what was it previously.

So, previously what we are saying that for each h_i of x right probability that h_i of p equals h_i of q was p of u if u is a distance between p and q . So, therefore, probability that capital H of x p equals capital H of q equals p to the k of u right because, remember because a hash table is constructed out of k such hash functions that are iid right. So, therefore, the probability of collision becomes p to the p u to the power k ok.

So, then this you should think about as the I mean this basically is like the original LSH collision probability. Now what we are denoting in the probability of collision of the probability of collision will so, let us say H_{tilde} is my ν hash index which is a k bit index again. Now we define the probability that H_{tilde} of p equals H_{tilde} of q to be P_{AC} of u . So, now, this is the this is the combined probability that the particular k topple of that have created for a single hash table collides with the particular topple for p and collides with k topple for q ok.

And we have to we cannot segregate we cannot this is not independent all as I mentioned. So, now, we have to give a different set of bound that basically says that the collision problem remains more or less the same as before it does not increase or decrease by too much ok. So, and we do not need to go into the details of this bound more than that.

(Refer Slide Time: 20:53)

DHash

We can make the projection faster

D = random diagonal matrix of ± 1
 G = random diagonal matrix, $G_{ii} \sim N(0, 1)$
 M = random permutation matrix

Hash value calculated as $\left\lfloor \frac{HGMHDx + b}{w} \right\rfloor$

Logos: IIT Gandhinagar, NPTEL ONLINE CERTIFICATION COURSES

But the algorithm itself is fairly simple right because, we just replaced the Gaussian random projection by the FJLT random projection. So, you can actually do better than this and this is what we call the; and this comes from the same paper and this is what we call the DH hash right.

So, the components of this are a little different so, we will have let me define a few of a few such notations. So, we have the D right, which is part of our randomized Hadamard transformation which we have been using, we have a new matrix here called the G that we denote by G what is G ? G is again a diagonal matrix except that it looks very similar to D right, except that every entry is now sampled from a independent Gaussian. D was being sampled from independent plus minus 1 G is being sampled from independent Gaussian I mean random variables each entry each d_{ii} is a independent Gaussian random variable. We are we also have M which looks like a in random permutation matrix.

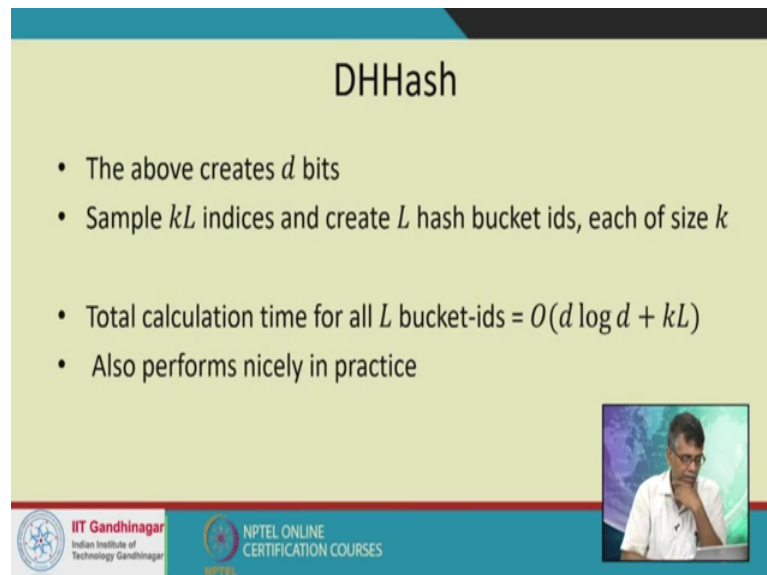
So, for a 3 by 3 for instance it says that so, if you have a 3 by 3 it says that maybe this goes here maybe this goes here maybe this goes here right. So, basically every row and every column of M has only 1 1 1 and everything else is 0. So, is it is just I mean M applied to any vector x just permutes the I mean the values of the x_i the values x_i . And now instead of calculating this $Ax + b$ right with a being PHD this is what we do.

So, we multiply HDx , then we multiply again it by M , then we multiply it again by G , then we multiply it again by H and then and this is my random projection first. And then

we and then we add a shift to it and then we quantize so, you might ask that what is happening instead of multiplying it by single A we are multiplying it by more matrices now right.

So, how is this any faster well this is faster right because, although we are multiplying it by 1 2 3 4 5 matrices right 3 of them are diagonal. So, multiplying by a diagonal matrix is very easy and 2 of them are Hadamard so, multiplying by a Hadamard matrix is also very easy right. So, although instead of a single matrix we are now I mean have multiple such operations to do the overall effect is that it is actually much faster.

(Refer Slide Time: 23:15)



The slide is titled "DHash" and contains the following bullet points:

- The above creates d bits
- Sample kL indices and create L hash bucket ids, each of size k
- Total calculation time for all L bucket-ids = $O(d \log d + kL)$
- Also performs nicely in practice

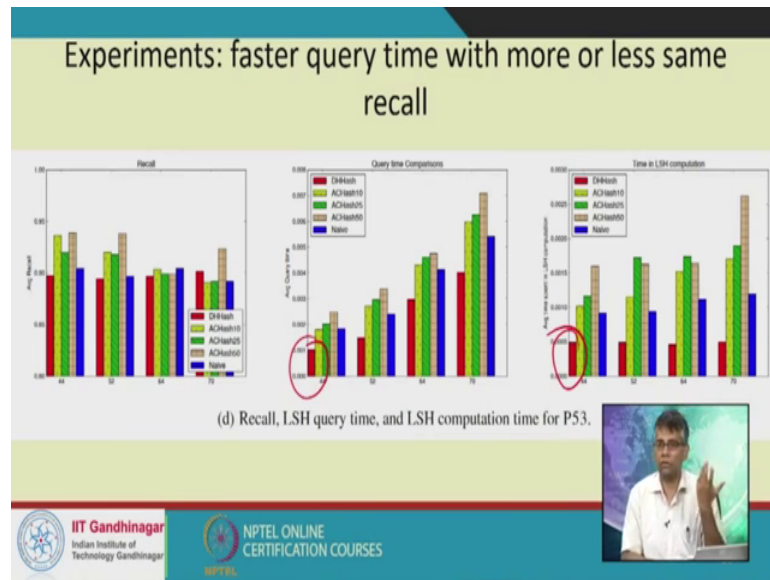
The slide also features logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

And we can also I mean so, one thing now so, now, so, one other thing to notice that see all these matrices are now d by d ok. So, now, we have this vector which is actually a sort of length d vector right, if I do $HGMx$ the this quantity.

So, what we could do is that we could sample we could sample k sort of indices from it. So, we take one sample of k indices from it that forms the index for my first hash bucket, we again sample k indices from it that forms the index of my second hash bucket and so on and so forth right. So basically, we are not redoing the calculation L times right, we are just we are just doing I mean taking multiple samples from here and then sort of and then calculating the index of each bucket ok.

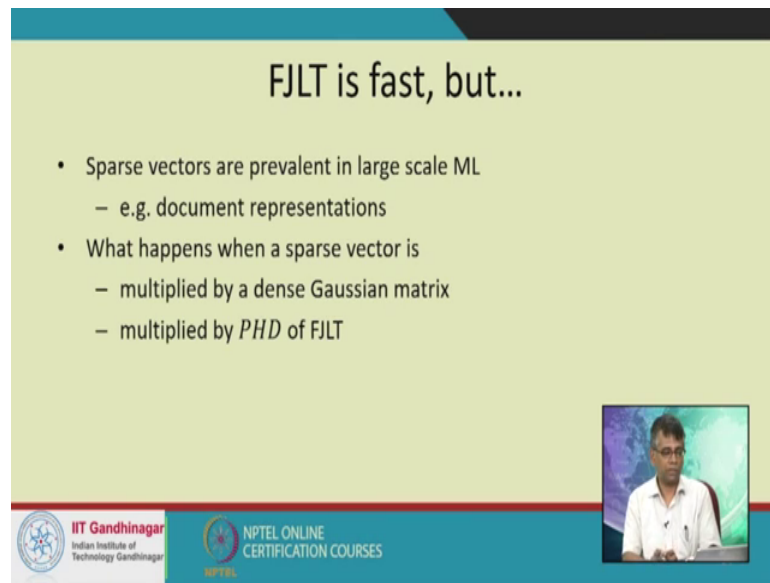
And this transfer to work fine right and we can prove a similar theoretical guarantee for it and the total calculation time for all the L bucket IDs that answer to be $d \log d$ plus $k L$ only right which is a I mean again faster than the AC hash that we are doing and certainly a lot faster than the original thing ok002E

(Refer Slide Time: 24:35)



So, I mean just to summarize just to give some experimental results you should look at the paper if you really want to read about this; what it says is that I mean in terms of the recall we are similar to the original LSH right. In, but our query time and our average query time and our average time in actually calculating hash buckets are both are both much faster than the original LSH which is a which is a win ok.

(Refer Slide Time: 25:02)



The slide features a light green background with a dark blue header and footer. The title 'FJLT is fast, but...' is centered at the top. Below it, a bulleted list discusses sparse vectors in machine learning and compares multiplication by a dense Gaussian matrix with multiplication by the PHD of FJLT. A small video inset shows a man speaking. Logos for IIT Gandhinagar and NPTEL are at the bottom.

FJLT is fast, but...

- Sparse vectors are prevalent in large scale ML
 - e.g. document representations
- What happens when a sparse vector is
 - multiplied by a dense Gaussian matrix
 - multiplied by *PHD* of FJLT

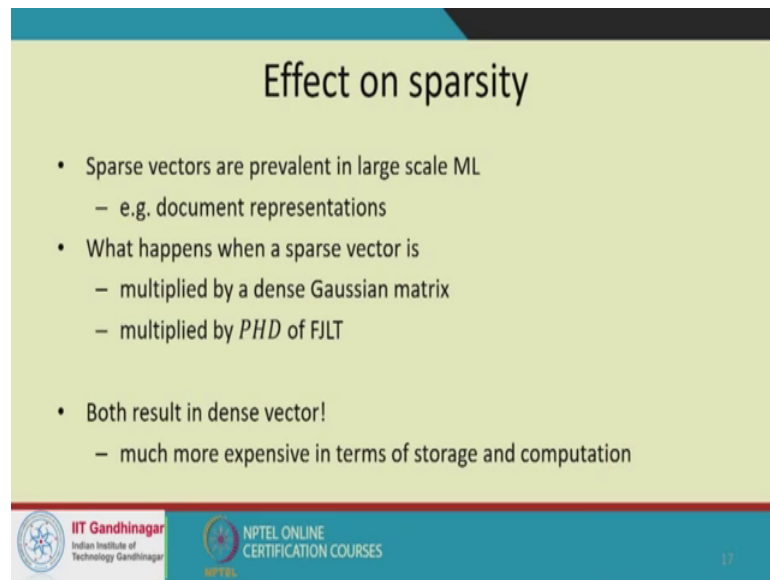
IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, now next to just to sort of go forward we have seen already one variant of the of the random projection, but there is more to come right. So, FJLT is fast right so, definitely a big improvement it is a amazing algorithm right, but here is A is problem right that I mean it does not it does not do well with the particular type of vectors that we are very interested sort of in machine learning relative applications and these are sparse vectors. For instance, if you are representing a document and it is always it is almost always a sparse vector right because, the size of the vocabulary is way larger than the actual number of words in the document right.

Now, let us see let us kind of think a little bit about what happens? When I have the input x to be such a sparse vector and I apply random projections to it right, either I apply sort of multiplying it by the by a Gaussian matrix as we have been doing or we multiplying it by PHD what happens then?

(Refer Slide Time: 26:03)



The slide is titled "Effect on sparsity" and contains the following text:

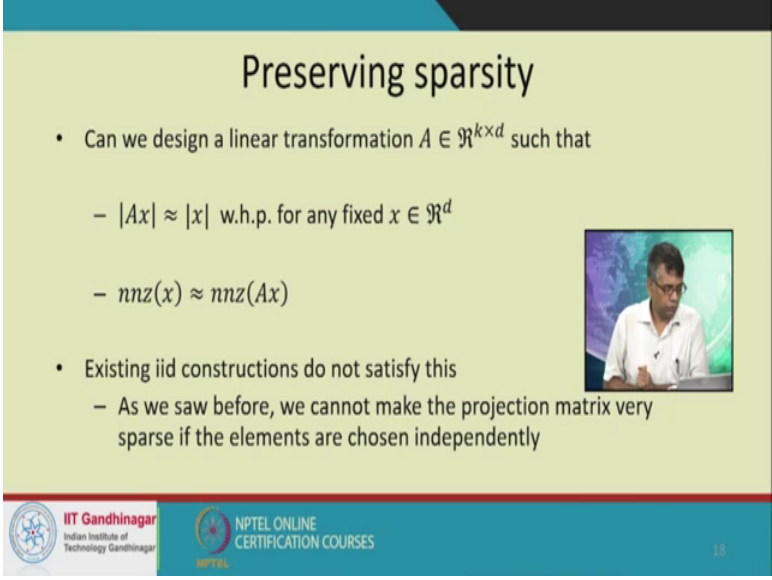
- Sparse vectors are prevalent in large scale ML
 - e.g. document representations
- What happens when a sparse vector is
 - multiplied by a dense Gaussian matrix
 - multiplied by *PHD* of FJLT
- Both result in dense vector!
 - much more expensive in terms of storage and computation

At the bottom of the slide, there are logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES. The number 17 is visible in the bottom right corner.

What happens is that in both cases it is becoming dense right because Hadamard transformation makes it dense right and multiplying it by p again makes it I mean increase I mean does not decrease the sparsity at least and definitely multiplying it as a sparse vector by a by a dense Gaussian matrix may makes it dense right.

So, in both the constructions at of random projections that we have seen we are converting a sparse vector to a very dense vector. In fact, even if the original vector had a single nonzero entry single nonzero entry the final vector is likely to have all d nonzero entries right basically full density. Soit results in its much more expensive and this can become much more expensive in terms of both storage and computation ok.

(Refer Slide Time: 26:50)



The slide is titled "Preserving sparsity" and contains the following text:

- Can we design a linear transformation $A \in \mathbb{R}^{k \times d}$ such that
 - $|Ax| \approx |x|$ w.h.p. for any fixed $x \in \mathbb{R}^d$
 - $\text{nnz}(x) \approx \text{nnz}(Ax)$
- Existing iid constructions do not satisfy this
 - As we saw before, we cannot make the projection matrix very sparse if the elements are chosen independently

A small video inset on the right side of the slide shows a man in a white shirt speaking. The bottom of the slide features logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES, along with the number 18.

So, here is the question can we design a linear transformation such that it preserves the norm just like before and it does not increase the density by so, much right maybe it has to increase the density by what, but maybe it does not increase the density by. So, much is it possible to sort of design such a random construction.

It is not hard to see that the existing iid constructions do not satisfy this and in some sense we have already seen a an example before right when we were sort of talking about sparsifying the random projection matrix right that its not that that if I make the random projection matrix independently and if I sort of make it very sparse right if I mostly try to put in 0s in every in every in every entry then i will end up with columns at a 0 and then I will be in trouble right because it is not going to preserve that the norm of sum vector.

(Refer Slide Time: 27:45)

Hashing as projection

Input $x \in \mathcal{R}^d$
Target $y \in \mathcal{R}^k$

Hash function $h: [d] \rightarrow [k]$
Sign hash: $s: [h] \rightarrow \{-1, +1\}$

$$y[j] = \sum_{i:h(i)=j} s(i) x_i$$

The diagram shows a vertical column of boxes representing the input vector x with entries a , b , and c . To the right is a vertical column of boxes representing the target vector y . Arrows indicate the mapping from x to y : a maps to the second bucket of y with a sign of $+1$; b maps to the same bucket with a sign of -1 ; c maps to the same bucket with a sign of $+1$. The resulting value in the second bucket is $a-b+c$. The target vector y is circled in red, and the equation $y[j] = \sum_{i:h(i)=j} s(i) x_i$ is also circled in red.

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

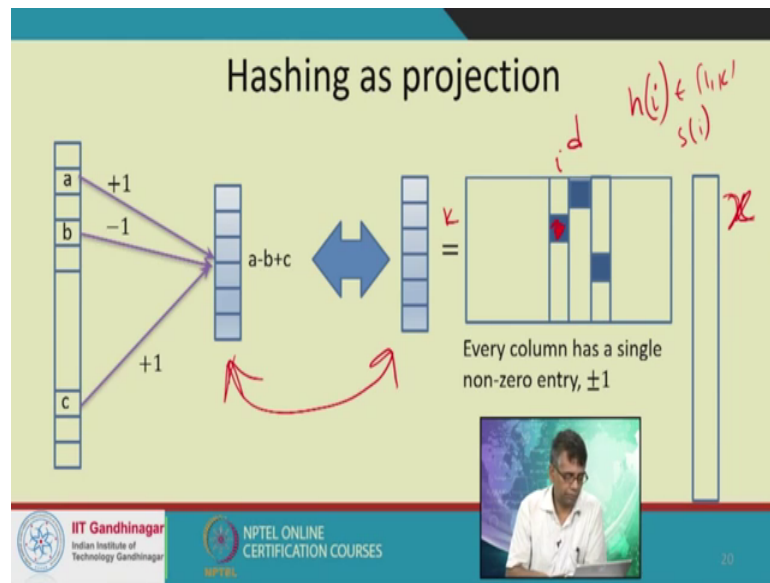
19

Here is another way of thinking about creating a random projection matrix. So, think about creating a I mean think about a hash function first of all.

So, what does this hash function do? So, let us say that the input vector x is of dimension d right. So, this is x and y is a dimensional k . So, y is my target vector. So what so we have a hash function that goes from d to k right. So, basically it takes an a it takes a coordinate id takes a coordinate id in a dimensions and maps it to one of the coordinate ids one of the buckets in k dimensions right we also have a sign hash function that does nothing, but sort of I mean adds I mean that assigns a random sign to each of the to each of the dimensions and so on and so forth maybe the center plus here minus here minus here plus here and. So, on maybe the sense plus here and now we obtain the vector y by basically taking all the I mean all the entries that are being mapped into this bucket multiplying them with the correct signs with the appropriate signs.

So, in this case for instance a , b and c are mapping into the second bucket of y therefore, the value that you put in the second bucket of y will be a minus b plus c because s assigns plus one here minus one here and plus one here right. So, this is what and if you write it down formally this is the definition of y right and y of j is the summation over all i such that m such that $h(i) = j$ $s(i) x_i$.

(Refer Slide Time: 29:07)




So, how does this correspond to projection it is not very hard to sort of see that you can capture this operation exactly by the by sort of designing a projection matrix what is this projection matrix hash this projection matrix is again of dimension k by d right and what it has is that you look at every column you sort of take the function h and then for the i th column you see where what is h of i right. So, h of i is a number between one and k . So, I go to that column you go to that particular row and then you see what is s of i right if s of i is plus 1 you put n plus 1 in that entry phase is minus 1 you put minus 1 in that entry right.


So, basically every column has a single nonzero entry that nonzero entry is defined by the position of the nonzero entry is defined by h of i and what you put in that nonzero entry plus 1 and minus 1 is defined by s of i and now it is not again very hard to see that sort of the operation that I just mentioned is basically the same as multiplying the vector x right with this with this particular projection matrix it is you get exactly the same y right there. So, these two these two y s are the same right it is exactly the same operation just convince yourself of this.


(Refer Slide Time: 30:23)

Sparsity

- $\text{nnz}(y) \leq \text{nnz}(x)$
- Norm preservation whp does not happen with only one hash function
 - Repeat the construction



 IIT Gandhinagar
Indian Institute of Technology Gandhinagar

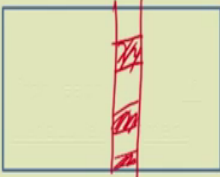
 NPTEL ONLINE
CERTIFICATION COURSES

21


So, a good thing has happened right because now if you again think of it the number of non zeros of y is that not is not more than the number of non zeros of x because I mean if x has I mean right because every entry of x I mean falls into exactly one of the buckets so. So, therefore, it cannot increase the number of non zeros of y . So, now, the question is that does the norm of y equal to norm of x I mean again we are not going to go into naught, but it is not I mean just convince yourself or think about this a bit, but what we can claim is a norm preservation does not really happen it there is only one hash function. So, we need to repeat this construction right and what we will see is that.


(Refer Slide Time: 31:07)

Sparse random projection matrix



For every column, choose a fixed number, ℓ , positions
For each position chosen, fill up with uar ± 1 random variable

 IIT Gandhinagar
Indian Institute of Technology Gandhinagar

 NPTEL ONLINE
CERTIFICATION COURSES

22

I mean I will just tell you what the construction is and then we will sort of talk about this we will sort of refresh this in the in the next class.

What we need to do is that for every column we were choosing only one nonzero entry we were choosing only one nonzero entry here we will have l hash function here what we will do is choose l non zero entries l positions instead of one. So, we will make sure that every column has l positions more or less and for each position fill it up I mean toss a random coin again and fill it up with a plus one or minus one. So, similar to what as I was doing ok.

(Refer Slide Time: 31:47)

Formalization

$A \in \mathbb{R}^{k \times d}$

[DKS10] Choose l positions from each column with replacement

[KN11] Choose l positions without replacement ✓

For each nonzero position $A_{ij} = \begin{cases} +1 w. p. \frac{1}{2} \\ -1 w. p. \frac{1}{2} \end{cases}$

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

So, and we will sort of go over the proof we will sort of talk about let me actually finish this then there are basically only two different ways of I mean two different ways of choosing these positions right one is that you could choose the positions with replacement right and since if I mean i choose this position and then I could again choose back this position. So, I choose l positions with replacement which means that in effect there could be less than l nonzero entries right because the nonzero entries could clash the choice of the nonzero entries and this was I mean again work done in one of our papers better construction that was suggested by king and nelson is that you choose l positions without replacement.

So, in every column you make sure that you choose l positions independently right without replacement right and every set of l positions is equally likely to come and then for each nonzero position you put in a plus 1 or minus 1 with probability half.


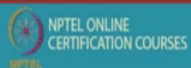
(Refer Slide Time: 32:46)

Guarantee

Claim: For $l = \tilde{O}\left(\frac{1}{\epsilon}\right)$, $k = O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$,

$$\Pr[(1 - \epsilon) \leq |Ax| \leq (1 + \epsilon)] \geq 1 - \delta$$

So a vector that initially has $\text{nnz}(x)$ nonzeros, now will have at most $\frac{\text{nnz}(x)}{\epsilon}$ non-zeros

So, it turns out that you can make the same kind of guarantees you can set k to be the same quantity one over epsilon square log one over delta some c by epsilon square, but this is order notation anyway and if you choose one to be basically 1 by epsilon right you can choose l to be 1 by epsilon and then have the same Johnson Lindenstrauss kind of guarantees.

So, now, the number of what you can then guarantee is that now my projection matrix is much sparser right and what you can guarantee is that if a vector x has n nonzero entries now it will have at most n by epsilon nonzero entries. So, we have increased the number of nonzero entries by a factor here 1 by epsilon, but still in effect it might have it might actually have much less nonzero entries it might not be entirely nonzero it might not be it might not be completely dense.

(Refer Slide Time: 33:38)



References:

- Primary references for this lecture
 - Fast Johnson Lindenstrauss Transform, Ailon and Chazelle, SIAM J Computing 2009.
 - Sparse Johnson Lindenstrauss Transformation, Dasgupta, Kumar, Sarlos, STOC 2010.
 - Fast Locality Sensitive Hashing, Dasgupta, Kumar, Sarlos, KDD 2011.

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Anirban Dasgupta
Computer Science and Engg.

25

So, just some references for this for this particular lecture the first Johnson Lindenstrauss transformation as you have seen was the was the paper from Ailon and Chazelle the two other result that I talked about the application from locality sensitive hashing comes from this paper in KDD and the sparse Johnson Lindenstrauss transformation comes from this paper in stock and we will see some applications of this later again.

Thank you.