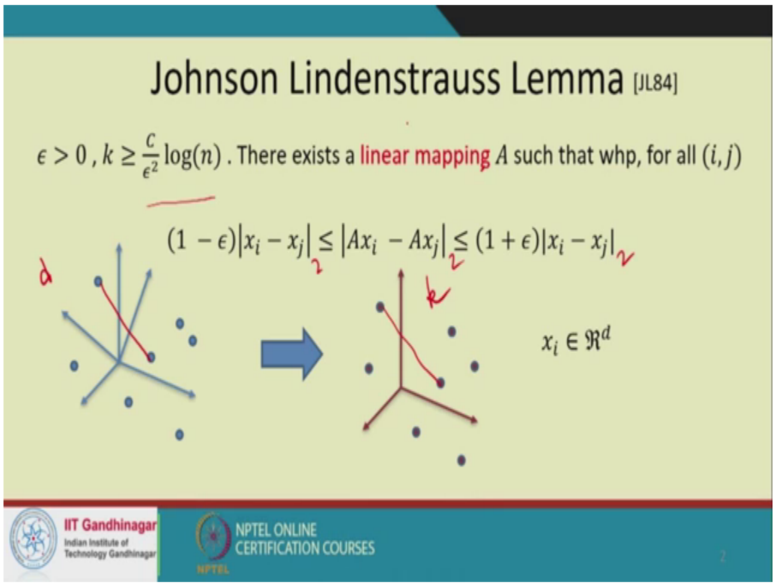
Scalable Data Science

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

Lecture - 14b
Random Projection Applications and Fast JL

Welcome to the course on Scalable Data Science. My name is Anirban I am from IIT, Gandhinagar. Today's lecture is going to be on the ups looking at small application for random projections and look and then looking at some other algorithms for improving the type of random projections that we were looking, that we had looked at before, right.

(Refer Slide Time: 00:39)



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)

$$(1-\epsilon)|x_i - x_j|_2 \leq |Ax_i - Ax_j|_2 \leq (1+\epsilon)|x_i - x_j|_2$$

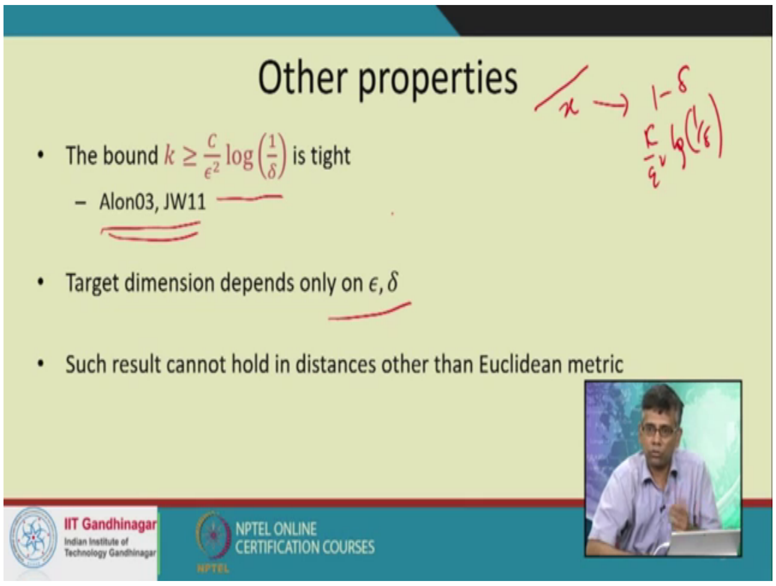
$x_i \in \Re^d$

d
k
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, just to remind you we were studying this Johnson Lindenstrauss Lemma, right what it says is that if you are given a set of points x 1 to x n in some d dimensional space and it does not really matter how big this d is. And what you are interested in is in creating a mapping that creates the images of these points in some lower dimensional space let me call that k, such that the pairwise distance between any two points is more or less preserved. So, it interested in coming upwards with representation with the representation of all the points such that the pairwise distances are more or less preserved. And the factor to which you want them reserved is given by this value epsilon, right.

So, every distance needs to be preserved to a factor 1 plus minus epsilon to a multiplicative factor 1 plus minus epsilon, right. And when we talk about distances we always talk about l 2 distances here. So, Johnson and Lindenstrauss says is that there exists a linear mapping, right such that the target dimension a is C over epsilon square log n and this mapping is created by the Gaussian by sort of filling it up by filling up the matrix with Gaussian, independent Gaussian random variables, ok. And what we will guarantee is that with very high probability for every pair ij the distance x i the distance A x i minus A x j will be within a 1 plus minus epsilon factor of the distance x i minus x j, ok.

(Refer Slide Time: 02:22)



Other properties

- The bound $k \geq \frac{C}{\epsilon^2}\log\left(\frac{1}{\delta}\right)$ is tight
  - Alon03, JW11
- Target dimension depends only on ϵ, δ
- Such result cannot hold in distances other than Euclidean metric

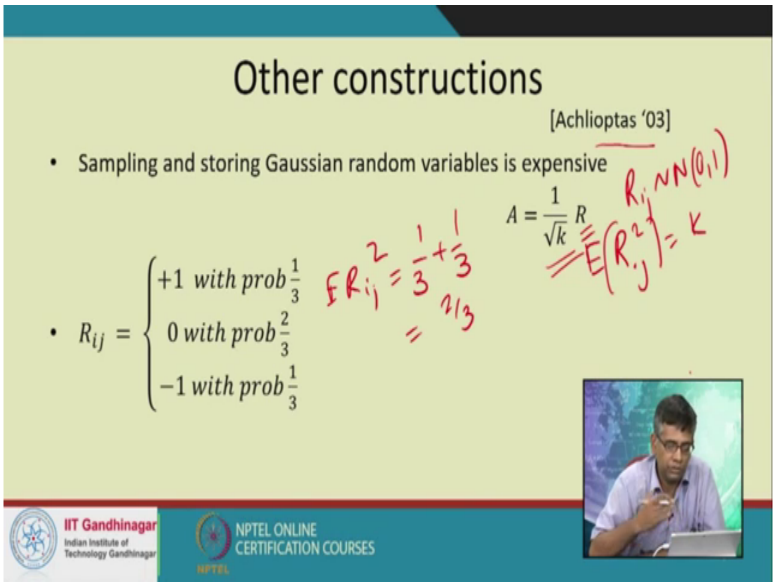
$x \to 1-\delta$
$\frac{C}{\epsilon^2}\log(1/\delta)$
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So couple of things before we move on into looking at an application first of all it is known that the bound on the bound on k, that this theorem achieves which is this 1 by epsilon square log 1 by delta. Remember why this log 1 by delta came, because improving the Johnson Lindenstrauss theorem the final result about the about n choose two pairs we were proving a core Lemma the core Johnson which is actually known as the j n Lemma in which we said that for a for a given for a given distance x or for a for a given point x its length is preserved with probability 1 minus delta if the target dimension is C over epsilon square log 1 by delta, right. And then we were applying a union bound to preserve the distances of all n choose two pairs, ok.

So, and this what this theorem says well what these results show this result by Alon and the result by Jayaram and Woodruff is that this bound is tight, right that if we want such a mapping you need to it has to be has to take at least 1 over epsilon, the target dimension has to be at least 1 over epsilon square it also has to be at least log 1 by delta and these are information theoretic bounds. You cannot really do much better however, smart your algorithm is, right.

It is also very interesting to know that the target dimension does not depend on the original dimension as we have mentioned before another very interesting and non trivial result is to show that that such a result cannot really hold in l 1 for instance, right. Then in some sense in a very strong sense that such a result more or less characterizes the Euclidean metric, ok. So, these are all pretty theoretical results that we are not going to go into the details, but they are I mean they are very insightful very good to know, ok.

(Refer Slide Time: 04:09)



Other constructions

[Achlioptas '03]

- Sampling and storing Gaussian random variables is expensive

$$A = \frac{1}{\sqrt{k}} R \qquad R_{ij} \sim N(0,1) \qquad E(R_{ij}^2) = k$$

- $R_{ij} = \begin{cases} +1 \ with \ prob \frac{1}{3} \\ 0 \ with \ prob \frac{2}{3} \\ -1 \ with \ prob \frac{1}{3} \end{cases}$ $\quad E R_{ij}^2 = \frac{1}{3} + \frac{1}{3} = 2/3$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

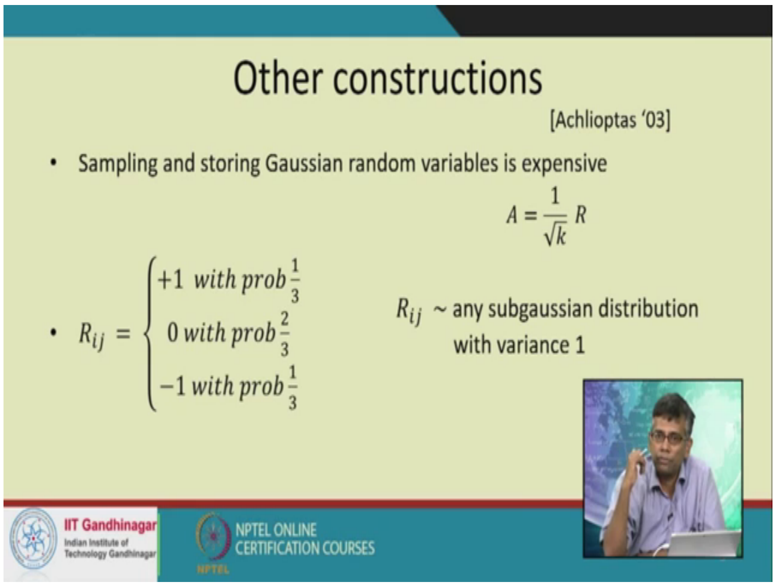
So, now let us talk about other ways of creating this matrix remember the way we were creating the matrix is by is by creating this matrix R such that R ij is N 0 1, ok. And then we made sure that every column of R is normally is has one norm that is more or less equal to 1, right.

So, the one, so has two norm that is more or less equal to that is more or less equal to 1 because the this norm of every column the expected norm of let us say the column 1 is equal to k, right. And if I divide by square root 1 over square root k if I take a equal to 1 over square root k times R then the expected norm of the of the first column of A becomes 1 and that is what I really wanted, right. So, therefore, I need to normalize it by 1 over square root in order to in order to actually get the JL matrix.

So, another way another interesting way and this was given by Achlioptas in 2003 of creating this matrix is to say that oh do not bother with normal random variables, because I need expensive to sort of sample from the expensive to create they are expensive to store because I need to store floating point numbers and so on, right. Instead create this R in the following way. For every entry independently toss a coin and now this is a 3 headed coin, right with probability two-third it gives you 0 with probability one-third it gives you plus 1, probability one-third it gives minus 1.

So, again it is unbiased. So, just like the original R ij's which say Gaussian is unbiased every R ij is unbiased it has a variance that is bounded by expectation of R i that is expectation of R ij square is one-third plus one-third which is equal to two-third, ok. And this is also fine we can show exactly the same kind of current is for this R as we can do for the for the original R, which is created out of normal random variables.

(Refer Slide Time: 06:10)



Other constructions

[Achlioptas '03]

- Sampling and storing Gaussian random variables is expensive

$$A = \frac{1}{\sqrt{k}} R$$

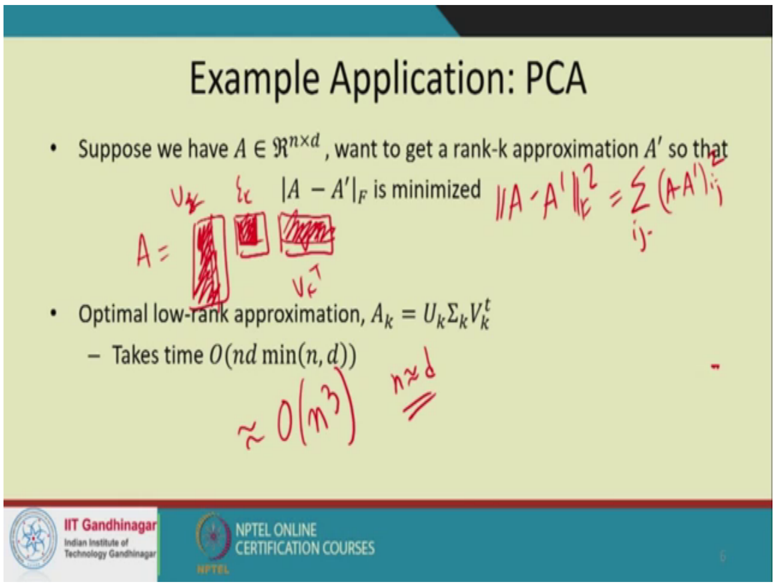
- $R_{ij} = \begin{cases} +1 \ with\ prob\ \frac{1}{3} \\ 0 \ with\ prob\ \frac{2}{3} \\ -1 \ with\ prob\ \frac{1}{3} \end{cases}$

R_{ij} ~ any subgaussian distribution with variance 1

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

Furthermore, any R ij that is created from what is known as a sub Gaussian distribution with variance 1 would work. So, intuitively as a Gaussian distribution is 1 who still decays faster than the normal distribution, right that is that is the intuition of the Gaussian distribution. So, you could sample R ij independently from any such distribution you need to normalize it so that it has variance 1 and then and then you are fine, ok. You can prove the same kind of inequalities.

(Refer Slide Time: 06:40)



Example Application: PCA

- Suppose we have $A \in \Re^{n \times d}$, want to get a rank-k approximation A' so that $|A - A'|_F$ is minimized
- Optimal low-rank approximation, $A_k = U_k \Sigma_k V_k^t$
  - Takes time $O(nd \min(n, d))$

$A =$
U_k
Σ_k
V_k^T
$\|A - A'\|_F^2 = \sum_{ij} (A-A')_{ij}^2$
$\approx O(n^3)$
$n \approx d$
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

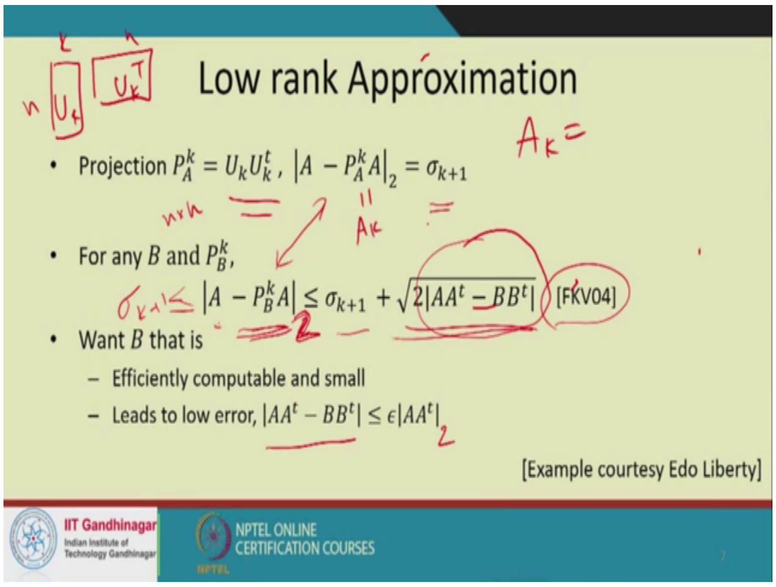
So, let us say a specific application of this random projection. So, suppose we have a matrix A in R n by d, right and we want to get a low rank approximation. Let us say a rank k approximation for a specific k. So, notice that I am sort of overloading k this k is disjoint is not the same as a k in the random projection that we talked about here we are using is it as a local variable.

We want to get a rank k approximation A prime so that A minus A prime the frobenius norm is minimized, if you remember if you do not remember what the frobenius norm is, so A minus A prime frobenius norm square is defined as the summation over all the entries A minus A prime ij square, you take the ijth entry you sort of square it and then you sum up the squares, ok.

So, we know what the optimal such A prime is if you have already seen it in previous lectures. The optimal low rank approximation is given by what is known as singular value decomposition. So, what you do is that you first do a singular value decomposition single singular vector singular value decomposition of A which is given by U sigma V transpose, where U has orthogonal columns V and so does V, and sigma is the diagonal matrix. And then if you want a rank a approximation of the optimal rank a approximation if you want A k if U prime to be of rank k, right then you keep only k singular values in sigma the corresponding singular vectors in U and the corresponding singular vectors in V transpose, right and this is denoted by U k sigma k and V k transpose, ok so, this is how we solve the problem.

The problem is that it takes time that is almost cubic in the data in the that is almost like n d times n times d mean of n d, right. So, if n and d are the same it takes time n cubed more or less if n is close to d, right. So, that is cubic in the number of data points that is not acceptable, ok. So, you want to do this much faster, at least approximate this much faster.

(Refer Slide Time: 08:59)



Low rank Approximation

- Projection $P_A^k = U_k U_k^t$, $|A - P_A^k A|_2 = \sigma_{k+1}$
- For any B and P_B^k,

  $\sigma_{k+1} \le |A - P_B^k A| \le \sigma_{k+1} + \sqrt{2|AA^t - BB^t|}$ [FKV04]
- Want B that is
  - Efficiently computable and small
  - Leads to low error, $|AA^t - BB^t| \le \epsilon |AA^t|_2$

[Example courtesy Edo Liberty]

n
k
U_k
U_k^T
$A_k =$
$n \times n$
A_k
2
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

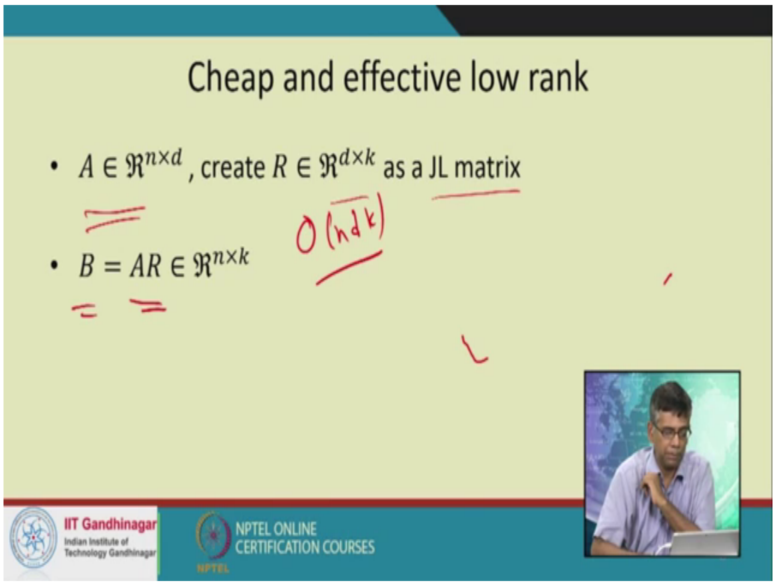
So, how do we do this? So, before we do this we need to look a little bit into in into some linear algebra. So, recall that this I mean we can write down the matrix A k also in terms of a projection of the columns of A onto some space. So, what is this space? You create a projection matrix out of the first k singular vectors of A. So, U k are the first k singular vectors of A and you create a projection matrix of rank k. So, U k is of size n. So, U k is of size n by k so, you create a project so, this projection matrix is of size n by n. So, it projects the columns of a onto the space spanned by the first k singular vectors of A, right. So, this is equal to A k actually and we know that A minus this P A k A two norm of this equal to sigma k plus 1. So, notice this is the matrix two norm right.

We also know using results from Frieze Kannan Vempala that for any matrix B, right. That suppose we look at any other matrix P and instead of taking it instead of taking the top k singular vectors of the matrix A, let me look at the top k singular vectors of the matrix B and project the columns of A onto that space, right. So, how close do we get from the error that we were getting when we are using A k, right.

So, justifiably if B is I mean you can imagine that if B is close to A, right then this error the error on the left hand side should be similar to this error, right. In fact, if it is exactly equal to k then the two errors are obviously, equal, right. So, this result by Frieze Kannan Vempala shows that if B is close to A then the error that you see out here, while it is atmost sigma k plus 1 because this is the sigma k plus 1 is the optimal error, right and we are looking at the 2 norm of the matrix here, while this error is at least sigma k plus 1 it is at most sigma k plus 1 plus term that depends on the difference between A and B. So, this difference is given by square root of the 2 norm of the matrix A transpose minus P B transpose, ok.

So, if I can make B be close to A, I mean if I can make B such that this error is small, right then I get that the error obtained by A minus P B A minus P B k of A is also small, right it is not very far off from the optimal error which is sigma k plus 1. So, I want to create a B that is efficiently computable and small in size, right and it should lead to small low error. So, concretely what I want is that I want this quantity the term that is computing to the error to be at most epsilon times the error the two norm of the A transpose, right why am I choosing this quantity because it is going to come up in my analysis, ok.

(Refer Slide Time: 12:12)



Cheap and effective low rank

- $A \in \Re^{n \times d}$, create $R \in \Re^{d \times k}$ as a JL matrix
- $B = AR \in \Re^{n \times k}$

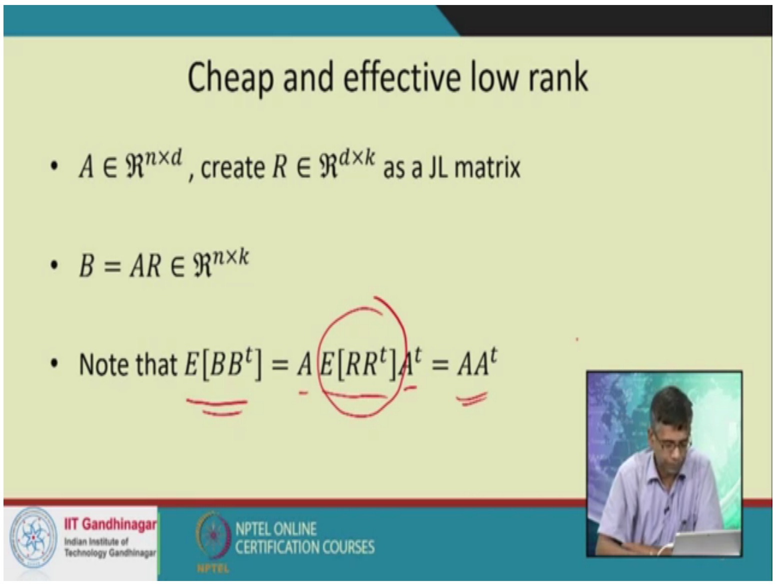
$O(ndk)$
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, how do I create such a B efficiently? It turns out that creating such a B is very easy, right and this is what I call cheap and effective low rank approximation. Given a matrix A which is which lies in n by d, right we just sample a d by k matrix JL matrix R. Remember what a JL matrix is, a JL matrix is something that that satisfies the conditions of the JL that satisfies the sort of statement of the JL theorem which means that it sort of preserves the distances to of a certain point set to a factor 1 plus minus epsilon that is what we call a JL matrix, ok.

And specifically I can create a JL matrix by sort of taking a d by k frame and then and then filling it up with Gaussians and 0 1 entries or with plus minus 1s sample according to the algorithm (Refer Time: 13:00) and then I could normalize it. So, that the length of each column, so that the expected length of each the expected squared length of each column is 1, ok.

And now which is we obtain B by just multiplying A with R that is it that is my B, ok. So, you can imagine that this is pretty efficient, right this takes time order n d k, this multiplication.

(Refer Slide Time: 13:29)



Cheap and effective low rank

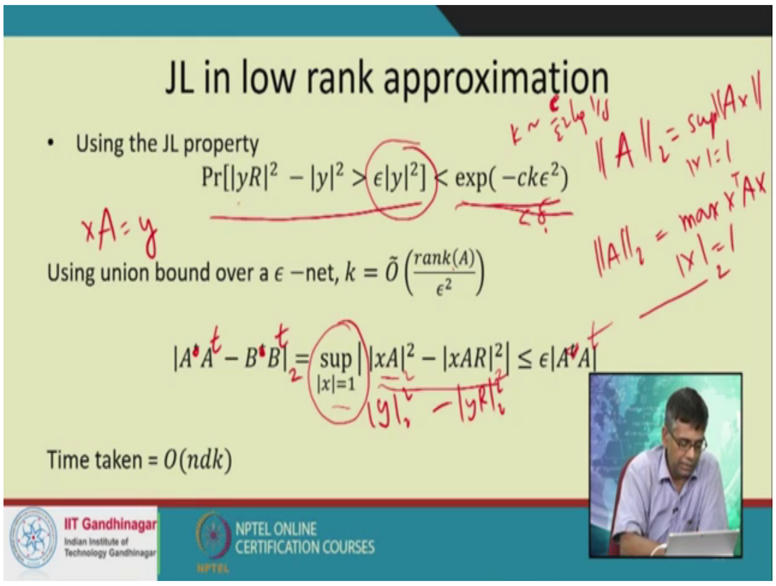
- $A \in \Re^{n \times d}$, create $R \in \Re^{d \times k}$ as a JL matrix
- $B = AR \in \Re^{n \times k}$
- Note that $\underline{E[BB^t]} = A\,E[RR^t]\,A^t = \underline{AA^t}$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

And also notice that just for sanity check if I calculate expectation of B B transpose that is equal to A times if I replace B by A times R, then this comes out to be expectation of E RR transpose A transpose. And it is not very hard for you to see that expectation of RR transpose under the condition is identity because every entry has unit variance every R ij has unit variance and because the R ijs are independent of each other therefore, the covariance of any two entries is 0, right.

So, based on this you can say that that this expectation comes out to be a transpose. So, so the matrix transpose certainly has a right expectation, right.

(Refer Slide Time: 14:13)



JL in low rank approximation

- Using the JL property

$$\Pr[|yR|^2 - |y|^2 > \epsilon|y|^2] < \exp(-ck\epsilon^2)$$

$k \sim \frac{\epsilon}{\epsilon^2}\log 1/\delta$

$< \delta$

$xA = y$

$\|A\|_2 = \sup_{|x|=1} \|Ax\|$

$\|A\|_2 = \max_{|x|_2=1} x^T A x$

Using union bound over a ϵ −net, $k = \tilde{O}\left(\frac{rank(A)}{\epsilon^2}\right)$

$$|AA^t - BB^t|_2 = \sup_{|x|=1} \left| |xA|^2 - |xAR|^2 \right| \le \epsilon |AA^t|$$

$|y|_2^2 - |yR|_2^2$

Time taken = $O(ndk)$

IIT Gandhinagar Indian Institute of Technology Gandhinagar

NPTEL ONLINE CERTIFICATION COURSES

So, now we know that the matrix R. So, now I want to make sure that I want to make sure that the error A transpose minus B B transpose is small, ok. So in fact, this should be A A transpose minus B B transpose, ok.

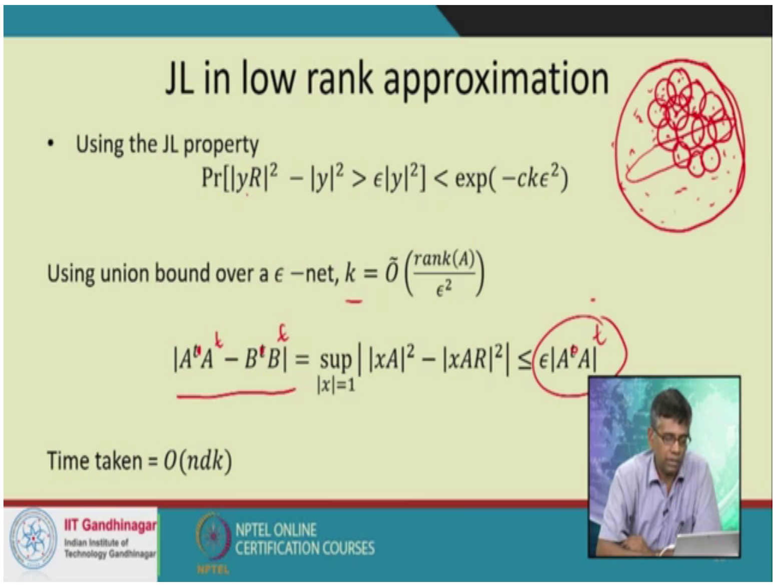
And this is really again just a simple application of the of the JL property, what we can show is that. So, remember what is the two norm of A transpose minus B B transpose. It is really the supremum, so the two norm of a matrix A is really the supremum of A x over all x that is two norm 1, right. So, therefore, with only a little bit of algebra the two norm of this of this of this particular matrix A A transpose minus B B transpose is given by you have to multiply it by x, let us say I mean an equivalent way would be to sort of, an equivalent way would be to just take maximum of x transpose x such that the two norm of such that the two norm of x is 1, right. And once you apply this inequality you see this once you apply this equality you see this that what I need to bound is the supremum of A x of this quantity under the condition of x A square the length of x A square minus the length of x A R square, right under the condition that x has unique norm, and.

So, now, how do we bound this? Now, for a fixed x, right, so imagine x times A is y, x times A is y, right. So, then this is nothing, but bounding y the length of y minus the projected length of y minus the length of the projection y R, right. And by using the Johnson Lindenstrauss property this is exactly what we know to be small, right. We know that the probability that the length y R the length of y R lies between I mean crosses one plus epsilon times the length of y is exponentially small, right is smaller than exp of minus c k epsilon square.

So, you have not written known exactly this form, right, but this is what the form that came out when we were actually doing the proof of Johnson Lindenstrauss and then in this form we plugged in k to be 1 by epsilon square log 1 by delta and then we get this quantity the, right hand side quantity to be less than delta k to be. I mean there has to be some hidden constants in there, right. So, now, we know. So, now, using the random projection the length preserving property of R we know that for a specific x this quantity is small, right. So, now, what do we do for the I mean small as in it lies it is of this order it is of the order of the length of y, it is of the order of the squared epsilon times the squared length of y.

So, but this is a supremum over a infinite set of vectors. So, what do we do? We have to take union bound, but we cannot really just take union down over infinite set of vectors, right. So, what a standard trick here is to do something like something called an epsilon net argument that is what we do is to say that, ok.

(Refer Slide Time: 17:50)



JL in low rank approximation

- Using the JL property

$$\Pr[|yR|^2 - |y|^2 > \epsilon|y|^2] < \exp(-ck\epsilon^2)$$

Using union bound over a ϵ −net, $k = \tilde{O}\left(\frac{rank(A)}{\epsilon^2}\right)$

$$\underline{|A^T\overset{k}{A} - B^T\overset{k}{B}|} = \sup_{|x|=1} \left| |xA|^2 - |xAR|^2 \right| \leq \overset{k}{\epsilon|A^T A|}$$

Time taken = $O(ndk)$

IIT Gandhinagar Indian Institute of Technology Gandhinagar

NPTEL ONLINE CERTIFICATION COURSES

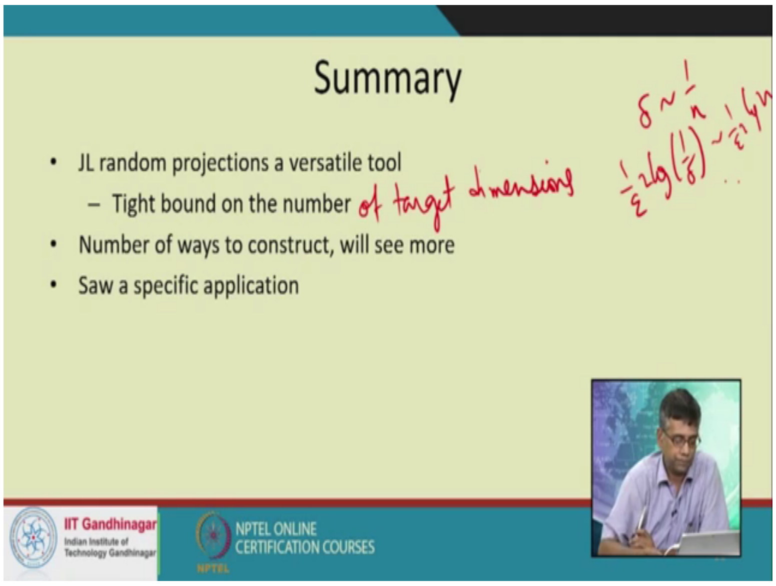
It is actually enough to show this, right for all y that is unit norm, right. And so what we want to show is that let us take the unit ball and let us put points on it and I want to put points on it such that the distance I want to put enough points on it. So, that the distance between any two of them is not more than epsilon, right.

So, every point, so marking these red points and I want to make sure, I want to make sure that any point on this ball is has a red point not far off I mean a any point on this ball, right has a red point to it that is closer than epsilon, ok. So, just to frame it in a different way that if I were drawing epsilon radius ball around every red point, right there these balls will of course, overlap, but the union of these balls will be the entire surface of this sphere, ok. So, I will cover the entire surface of the spheres by drawing balls of radius epsilon around these red points, ok.

So, this is known as epsilon net, right and it is I mean using such an epsilon net argument, right. So, we need to draw an epsilon net in for a ball, right that has a dimension which is more or less the rank of A. We can actually do better later in later a later class we will see a better bound on that on the target dimension that we need, but for now it is enough for you to note, right that that if you take k to be let us say rank of A by epsilon square, right with some log factors then using this epsilon net argument we can show that that this quantity, we can show that this quantity A transpose minus B B transpose is less than epsilon times A transpose, right. Using basically the Johnson Lindenstrauss theorem and some union bound some clever union bound arguments, ok.

So, this would then allow us to use B, this particular B in finding out a low rank approximation of A, and that is and that is very useful and we will see repetitions of this of this particular thing.

(Refer Slide Time: 20:11)



Summary

- JL random projections a versatile tool
  - Tight bound on the number of target dimensions
- Number of ways to construct, will see more
- Saw a specific application

$\delta \sim \frac{1}{n}$

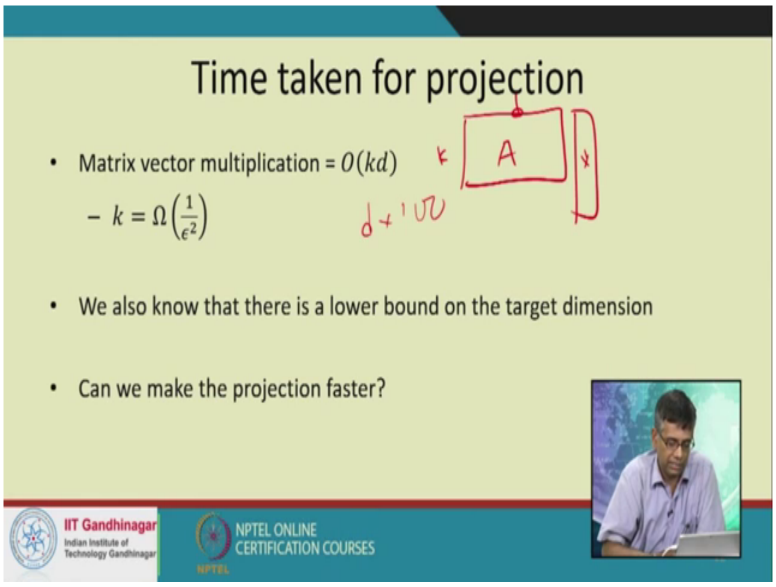
$\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right) \sim \frac{1}{\varepsilon^2}\log n$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, in summary Johnson Lindenstrauss projections are a very versatile pool. There is a there is a typed bound on the number of target dimensions, right there are a number of ways to construct these that we need for Johnson Lindenstrauss and this bound depends on both the error parameter, right. As well as I mean on the confidence value delta that you want to have, right, in the yeah. If you want delta to be let us say like one by n or you wanted to hold for n square pairs of points, right you want to set the I mean you need to make a delta to be to be of the order of 1 by n then the target dimension needs to depend on 1 over epsilon square log 1 by delta which is 1 over epsilon square log n, ok.

We saw a very specific application of this random projection in creating low dimensional sort of factorizations of a matrix, and we will see a little more of these.

(Refer Slide Time: 21:20)

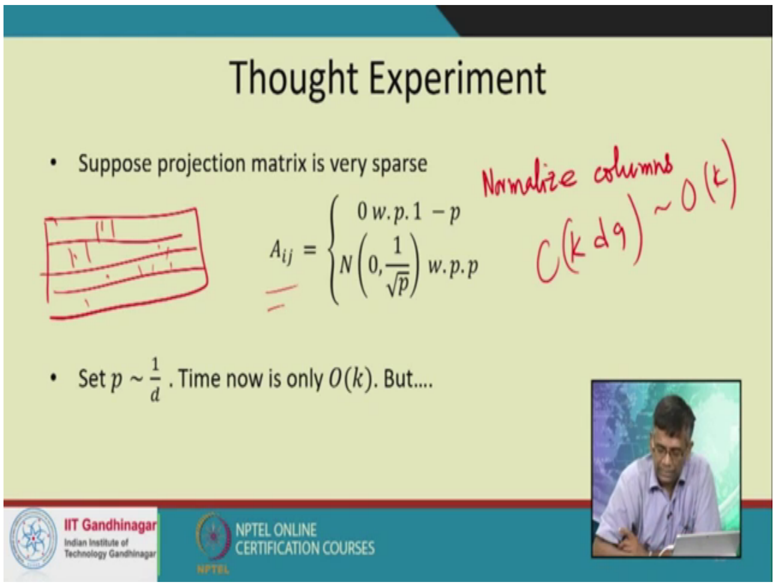


Time taken for projection

- Matrix vector multiplication = $O(kd)$
  - $k = \Omega\left(\frac{1}{\epsilon^2}\right)$
- We also know that there is a lower bound on the target dimension
- Can we make the projection faster?

So, the time taken, so then let us look at a little more about how to create more efficient random projections, the time what is the time taken for creating a random projection of a vector of dimension d, ok. The matrix is of is a size k by d the vector as I mentioned d and therefore, the matrix multiplication takes time order k d. Because this k and, but this k is 1 over epsilon square. So, this is not a very nice quantity, right because if you set epsilon to be let us say like 0.1, ok. There are we already have d times 100 d itself was fairly big. So, this, so can I sort of make it faster can I create a random projection faster well we could have if we if the target dimension was smaller, but we know that it is not very small it is it cannot be smaller than this. So, can we make it faster while keeping the target dimension to be the same this is a question that we investigate in this in the rest of this lecture ok. So, let us do a thought experiment.

(Refer Slide Time: 22:26)



Thought Experiment

- Suppose projection matrix is very sparse

$$A_{ij} = \begin{cases} 0 \; w.p. \; 1-p \\ N\left(0, \frac{1}{\sqrt{p}}\right) \; w.p. \; p \end{cases}$$

Normalize columns

$C(k\,d\,q) \sim O(k)$

- Set $p \sim \frac{1}{d}$. Time now is only $O(k)$. But....

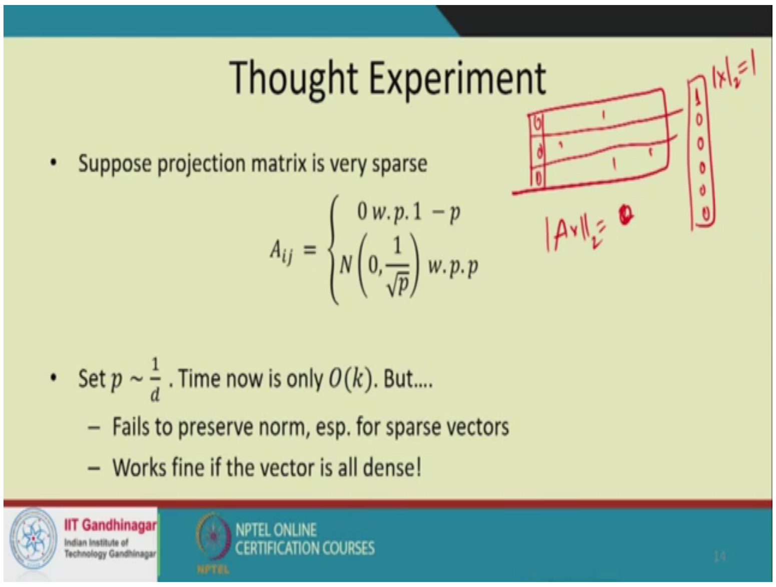
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

And suppose we are not reducing the target dimension, but we are making the matrix very sparse, right. In the sense that suppose in the extreme case, right if the matrix has only one nonzero per a row, then matrix vector multiplication is very easy, right. It would take only time order k, because I need to store I mean if I happen to store in a matrix I need to be a little clever I cannot store the 0s I have to store a sparse representation of the matrix, right. And therefore, write up the matrix vector multiplication a little more cleverly and I can do the matrix I mean exploit the sparsity and to reduce this time.

So, specifically let us create a matrix like this, right. That suppose I put in a 0 with probability 1 minus P, and with probability P I sample I create a sample from this distribution n 0 1 over square root P, right and then and then I put in the n value in any ij. So, why am I normalizing this? It is again to make each entry need variance that is why I have this 1 over square root P out here, right. So, now, most of the so 1 minus P fraction of the entries are 0s and the rest have some Gaussian value put in them, right.

So, now imagine in the extreme that P is more or less like 1 by d, right and what I was saying that that is the case that happens before, that most rows have a constant number of entries. So, the time taken to do the matrix vector multiplication is now k d q, order k d q, right which is order k only but is this really I mean a JL matrix, that does it satisfy the conditions of the Johnson Lindenstrauss projection? Well it might not, right.

(Refer Slide Time: 24:11)



Thought Experiment

- Suppose projection matrix is very sparse

$$A_{ij} = \begin{cases} 0 \ w.p.\ 1-p \\ N\left(0, \frac{1}{\sqrt{p}}\right) \ w.p.\ p \end{cases}$$

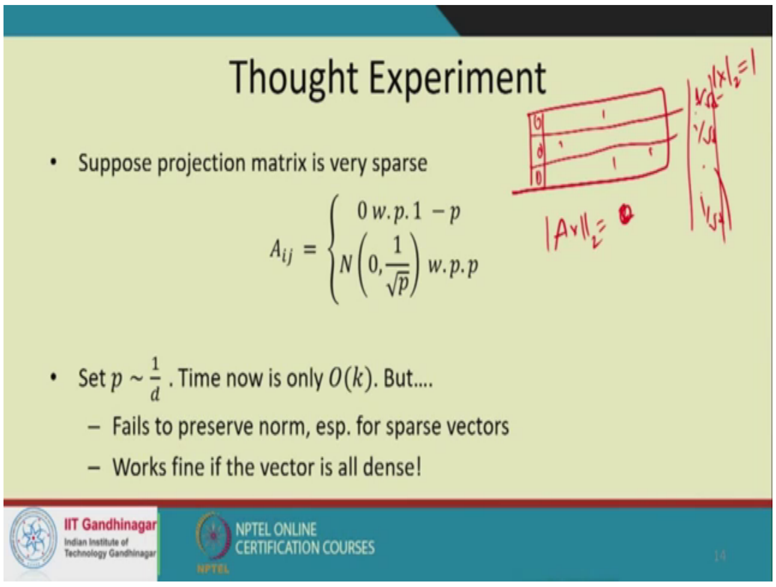
- Set $p \sim \frac{1}{d}$. Time now is only $O(k)$. But....
  - Fails to preserve norm, esp. for sparse vectors
  - Works fine if the vector is all dense!

$\|x\|_2 = 1$
$\|Ax\|_2 = 0$
IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

And this is and this is really the, this is really the problem. The problem is that that suppose you have a particular I mean because every row has only constant number of constant number of 1s there might be a column that is all 0.And imagine that if there is a column in a that is all 0 imagine just giving it a vector that has one only for that particular column and 0s everywhere else. So, then the norm of this vector is 1, but the norm of A x is 0, right. So, therefore, A x cannot be within a 1 plus minus epsilon approximation I mean factor of the input vector x, right.

So, then this does not work as a I mean we cannot hope to prove a Johnson Lindenstrauss bound for this matrix. However, imagine that if the matrix if the if the original vector was dense, if instead of this I had one over square root d for every entry, right then such a condition cannot happen and then and then we would be fine, right.

(Refer Slide Time: 25:22)



Thought Experiment

- Suppose projection matrix is very sparse

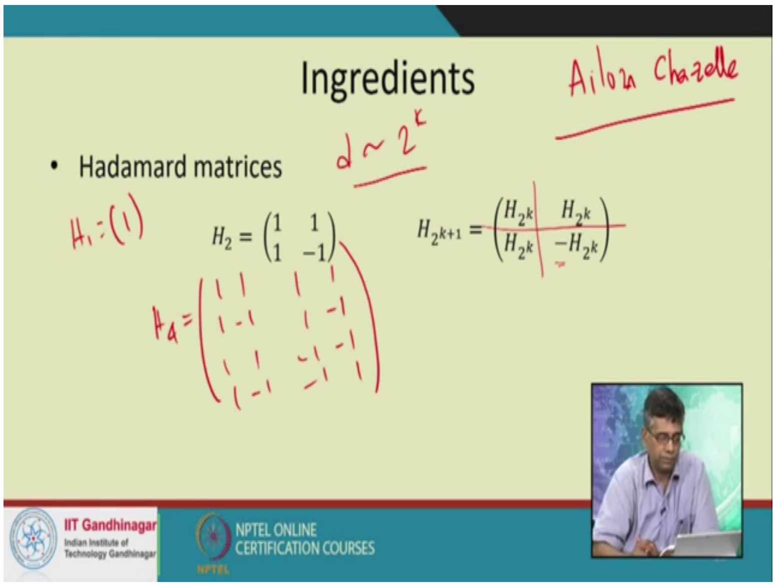
$$A_{ij} = \begin{cases} 0 \ w.p. 1-p \\ N\left(0, \frac{1}{\sqrt{p}}\right) \ w.p. p \end{cases}$$

- Set $p \sim \frac{1}{d}$. Time now is only $O(k)$. But....
  - Fails to preserve norm, esp. for sparse vectors
  - Works fine if the vector is all dense!

$\|x\|_2 = 1$
$\|Av\|_2 = 0$
$1/\sqrt{d}$

So, can I only work with dense vectors? Well, I have to give the Johnson Lindenstrauss bound for all vectors. So, what can we do?

(Refer Slide Time: 25:41)



Ingredients

- Hadamard matrices

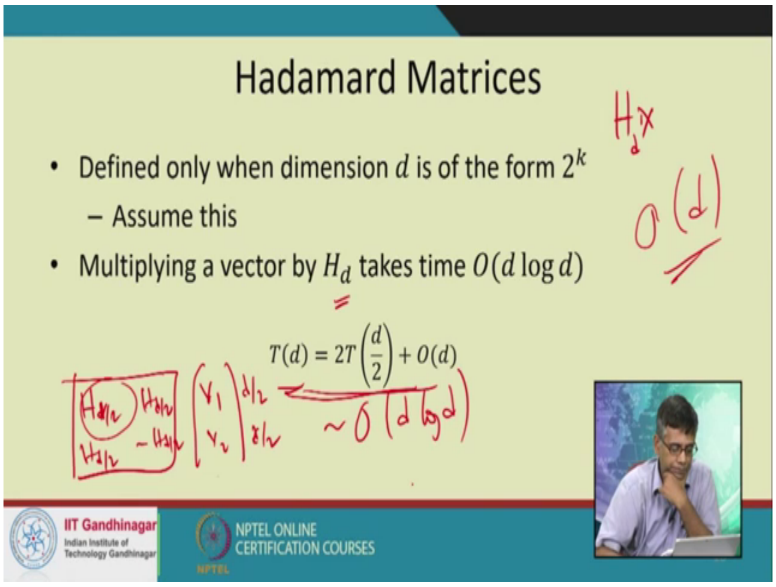
$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad H_{2^{k+1}} = \begin{pmatrix} H_{2^k} & H_{2^k} \\ H_{2^k} & -H_{2^k} \end{pmatrix}$$

Ailon Chazelle
$d \sim 2^k$
$H_1 = (1)$
$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$

So, can I pre process my vector to make it dense, right. But I have to make sure that the time taken for pre processing is not more than the time taken for to do the projection itself, right and this is the algorithm that was proposed by Ailon and Chazelle called as fast Johnson Lindenstrauss transformation.

The main ingredient here is something on the Hadamard matrix. So, what is the Hadamard matrix? Well, a Hadamard matrix is a of dimension d, right it is a d by d matrix, right its only defined when d is 2 to the k and its defined in this recursive way you must have seen this before that the Hadamard matrix H 1 is nothing, but the matrix that contains only one, H 2 is the matrix 1 1 minus 1 1 minus 1, right H, H 2 to the k plus 1 is defined recursively that you get 4 copies of H 2, H 2 to the k, right and you place the copies side by side, right except that you put in a negative in one of the copies. So, basically H 4 would be you put in 1 1 1 minus 1, 1 1 1 minus 1. So, I am just putting copies of H 2 1 minus 1 and then minus 1 minus 1, minus 1 1 this is H 4 and then you keep on going to H 8, H 16 and so on, ok.

(Refer Slide Time: 27:06)



Hadamard Matrices

- Defined only when dimension d is of the form 2^k
  - Assume this
- Multiplying a vector by H_d takes time $O(d \log d)$

$$T(d) = 2T\left(\frac{d}{2}\right) + O(d)$$

$H_d x$

$O(d)$

$$\begin{bmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \begin{matrix} d/2 \\ d/2 \end{matrix} \quad \sim O(d \log d)$$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

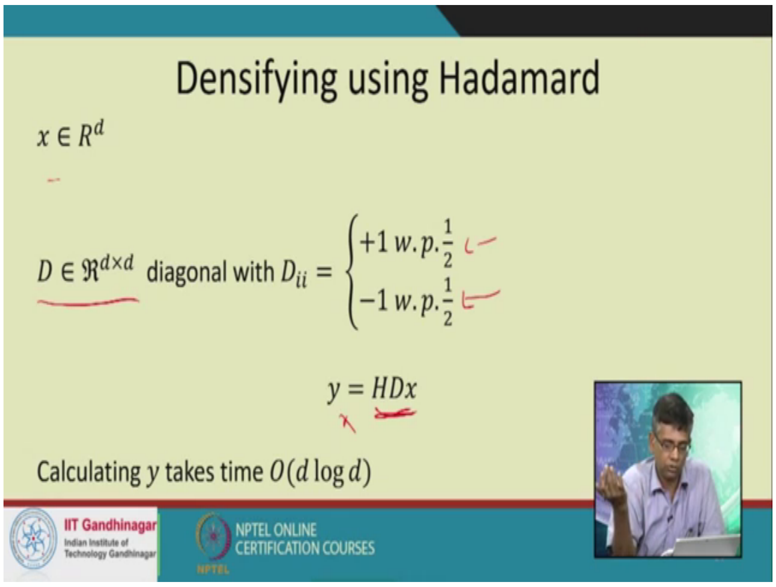
So, why is this nice? So, first of all it is defined only when d the dimensions is of the form 2 to the k. So, let us assume this and this is not to this is not such a bad assumption because I mean if the dimension is not of size 2 to the k then by at most doubling it I can reach it a power 2 to the k, ok.

So, now, it is easy to see that multiplying H d times x H d. I mean H d is let us say the d by d Hadamard matrix. So, I will refer to both H d and H. So, multiplying H d times x takes only instead of taking d squared time it takes only time d log d. And why is that? Because of this of this recursive way, right because in order to multiply I needed to do H d by 2 H d by 2, H d by 2 minus H d by 2 and then x 1 and x 2, where x 1 is of dimension d by 2 and x 2 is of dimension d by 2.

So, then I need to multiply H d by 2 with x 1 H d by 2 with x 2, but again in order to fill up the next one I need to multiply H d by 2 again with x 1 and minus H d by x 2. So, I need to do only two multiplications, right that is of a matrix of size d by 2 by d by 2 with a vector of size d by 2, right and I can fill up the entire and I can do the multiplication of H d with x using only two multiplications of the form, H d by 2 with some vector of dimension d by 2. Therefore, if you write if you happen to, right there on the recursion it turns out to be like this the T of d which is the time taken to do the d by d matrix vector multiplication is here is 2 T d by 2 plus order d which if you roll off the recursion turns out to be order d log T, right.

So, instead of d square for a general d by d matrix I mean d by d matrix vector multiplication its now order d log d, ok.

(Refer Slide Time: 29:11)



Densifying using Hadamard

$x \in R^d$

$D \in \Re^{d \times d}$ diagonal with $D_{ii} = \begin{cases} +1 \ w.p. \frac{1}{2} \\ -1 \ w.p. \frac{1}{2} \end{cases}$

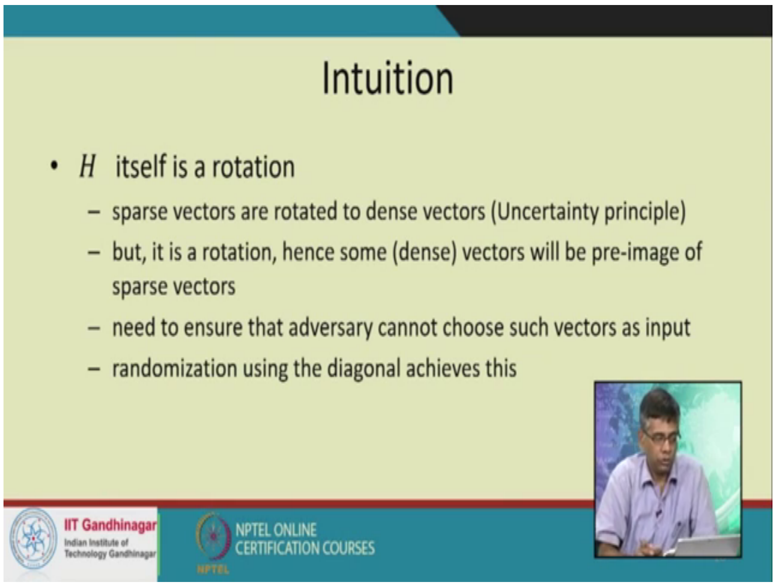
$y = HDx$

Calculating y takes time $O(d \log d)$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, now the next step what we will do is a densification using Hadamard. So, before I explain what it is I mean why it works, let me just show it show it to you what it is that we have a diagonal matrix D such that D ii is plus 1 with probability half and minus 1 probability half. And then given m given a vector x i first multiply it with the diagonal matrix d then I multiplied to the Hadamard matrix H and I get y. So, this y we will call it as the densified version of x. So, couple of things to note first is that this also takes time d log T because multiplying by diagonal is only linear time. Secondly, H times d is an orthonormal matrix because H is orthonormal matrix and this particular d that we have defined is also an orthonormal matrix, ok.

(Refer Slide Time: 29:59)



Intuition

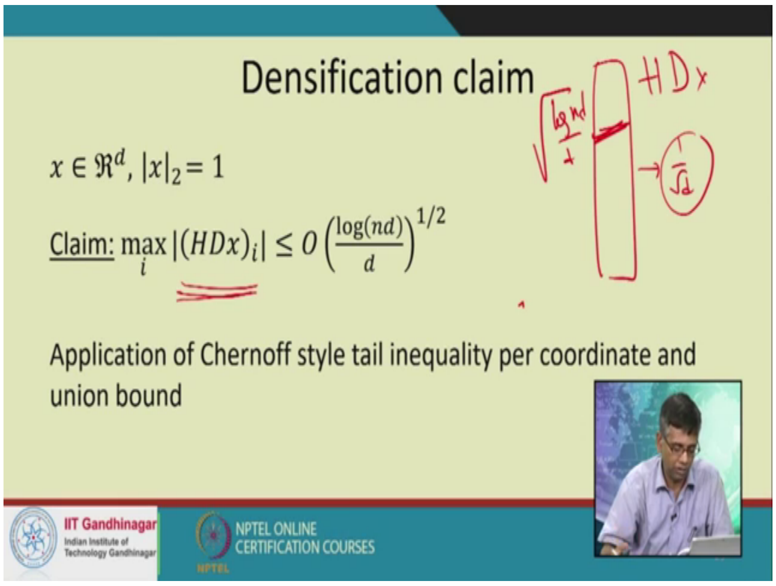
- H itself is a rotation
  - sparse vectors are rotated to dense vectors (Uncertainty principle)
  - but, it is a rotation, hence some (dense) vectors will be pre-image of sparse vectors
  - need to ensure that adversary cannot choose such vectors as input
  - randomization using the diagonal achieves this

IIT Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

So, the intuition behind this is that H itself is a rotation and by using something with uncertainty principle, right which basically says that H is a kind of Fourier transform. So, the uncertainty principle says that a signal cannot be sparse both in original domain as well as in the Fourier domain, which means that if I apply H to a sparse vector it becomes a dense vector. However, because H is a full rank matrix it is and because it is a rotation itself I mean some dense vector also be the pre-image of some sparse vector.

So, therefore, if an adversity gives me fines gives me such a dense vector my output of the H of x will be a dense vector and I cannot allow that to happen, I always want my output to be dense, I always want my y to be dense. So, therefore, I have to come up with this random diagonal matrix which sort of says that that adversary cannot without sort of knowing the random bits the adversary it does not really know how to come up with a pre image of a sparse vector, right and the way to formally sort of state this is as follows.

(Refer Slide Time: 31:04)



Densification claim

$x \in \Re^d, |x|_2 = 1$

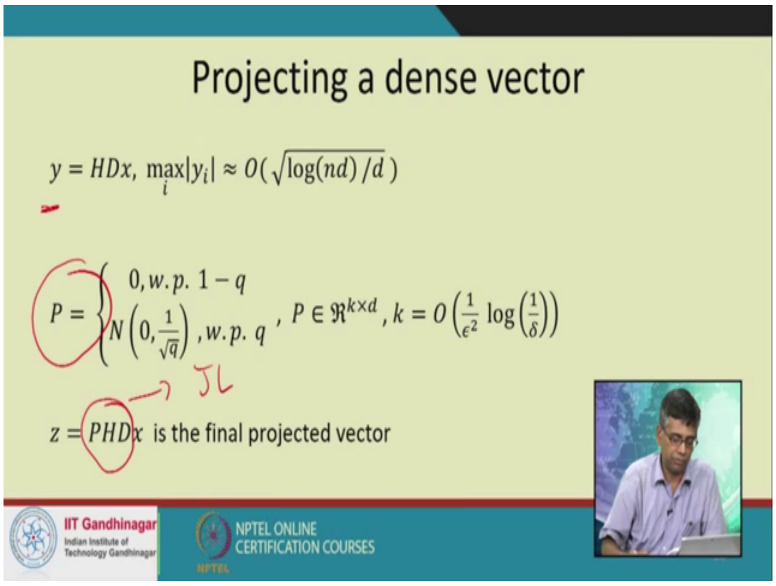
Claim: $\max_i |(HDx)_i| \leq O\left(\frac{\log(nd)}{d}\right)^{1/2}$

Application of Chernoff style tail inequality per coordinate and union bound

HDx
$\sqrt{\frac{\log nd}{d}}$
$\frac{1}{\sqrt{d}}$

Is that what you can say is that for any x that is that has 2 norm 1. If you look at the maximum entry in HDx, right. So, HDx is also has 2 norm 1. So, therefore, the average entry is 1 over square root d, if it was completely dense all the entries would be 1 over square root d. What it says? What this says is that the maximum entry is not much more than this average entry. So, the maximum entry is something like square root of log n d by square root d the maximum entry which means that HDx is a pretty dense vector, right and this the getting I mean getting this bound is not I mean conceptually it is not very hard just application of Chernoff style tail inequality.

(Refer Slide Time: 31:51)



Projecting a dense vector

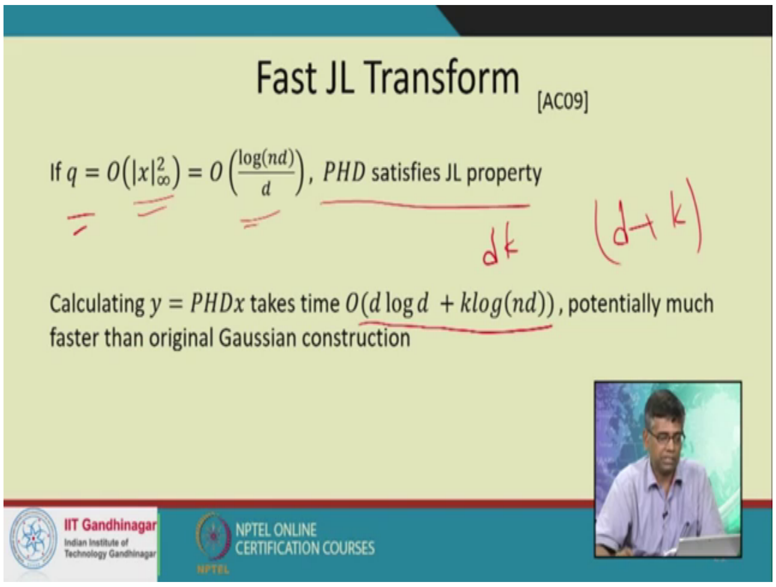
$y = HDx, \max_i |y_i| \approx O(\sqrt{\log(nd)/d})$

$P = \begin{cases} 0, w.p.\ 1-q \\ N\left(0, \frac{1}{\sqrt{q}}\right), w.p.\ q \end{cases}, P \in \Re^{k \times d}, k = O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$

$z = PHDx$ is the final projected vector

So, now, we are almost done because we because now that we have we have a dense vector y we can apply our previous sparse projection matrix, right on this dense vector y and my final matrix is this matrix P times H times d is my final projected vector, ok. So, and this is my JL matrix. So, first I will apply the diagonal matrix on x, then I will apply the Hadamard transformation, then I left like the sparse projection matrix, right.

(Refer Slide Time: 32:21)



Fast JL Transform [AC09]

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

dk $(d+k)$

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

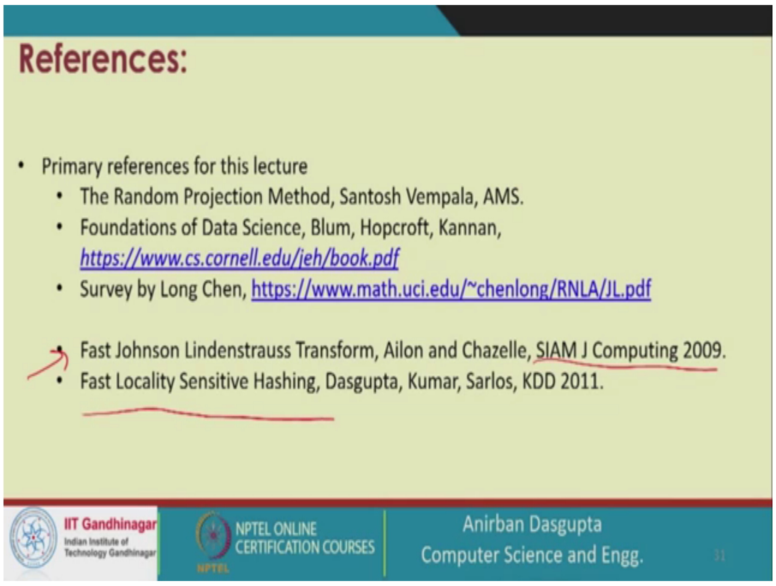
IIT Gandhinagar
Indian Institute of Technology Gandhinagar
NPTEL ONLINE CERTIFICATION COURSES

And this is the fast Johnson Lindenstrauss transformation given by Ailon on and Chezelle, it says that if I happen to choose q. So, remember q was the how sparse I made the matrix q is that probability, right only I expect only I mean only q fraction of the entries to be nonzero.

So, if I choose q to be of the order of the of the maximum entry of maximum entry square of x which in this case turns out to be log n d by d then PHD satisfies the JL property which is the 1 plus minus epsilon length preservation. And setting this value of q makes that calculation of the of the PHD x takes time order d log t plus k log nd. Remember that the original time was order d times k and now we have boiled it down to something like d plus k with some log factors thrown in and potentially therefore, this is much faster then the Gaussian concentration, ok.

So, we will talk about the application of this locality sensitive hashing later. But just some references.

(Refer Slide Time: 33:19)



References:

- Primary references for this lecture
  - The Random Projection Method, Santosh Vempala, AMS.
  - Foundations of Data Science, Blum, Hopcroft, Kannan, https://www.cs.cornell.edu/jeh/book.pdf
  - Survey by Long Chen, https://www.math.uci.edu/~chenlong/RNLA/JL.pdf
  - Fast Johnson Lindenstrauss Transform, Ailon and Chazelle, SIAM J Computing 2009.
  - 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.

There is a very nice book on The Random Projection Method by Santosh Vempala. You can also find this in chapter two, of the foundations of data science which is a free book by Blum, Hopcroft and Kannan. There is a Survey by Long Chen at this link, and the fast Johnson Lindenstrauss transformation is obtained from this paper by in the science journal of computing. In the next lecture we will talk about an application on fast locality sensitive hashing that is by that is from a paper of us and.

Thank you.