

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

Lecture - 16 a
Modified QB + Linear Regression

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 a modification that we look at for the QB algorithm, for the QB decomposition algorithm plus we look at applications of this randomized techniques, the random projection kind of techniques to linear regression solve problems.

(Refer Slide Time: 00:37)

QB decomposition

$A \approx QB$

Can we find orthogonal $Q \in \mathbb{R}^{n \times (k+p)}$ and B such that
 $|A - QB|_F \approx |A - A_k|_F$ and/or $|A - QB|_2 \approx |A - A_k|_2$

Also want $k + p = O(k)$

The slide features a diagram with three matrices: a blue matrix A of size $n \times d$, a green matrix Q of size $n \times (k+p)$, and a red matrix B of size $(k+p) \times d$. Red arrows indicate the dimensions of each matrix. Below the diagram, the text asks if we can find orthogonal Q and B such that the Frobenius and spectral norms of the error matrix $A - QB$ are approximately equal to those of $A - A_k$. A red arrow points from the Q matrix to the Q term in the equations. At the bottom right, there is a small video inset of the professor.

So, just refresh a memory, we talked about how important getting the lower rank composition of the matrixes for certain machine learning applications. We did not talk about the fact that while the single value decomposition is a pretty useful lower rank approximation of the matrix, it is very expensive. And, therefore we want to get something much more cheaply right and we are for instance try to build something that we call QB decomposition.

So, what we looking to solve a solve is the following, that suppose we have a target rank k in mind right, that is we really want to get down to the we really want to get down a low ranker approximation that approximates that captures the error A minus A_k either in

the Frobenius norm or in the spectral norm right. But for that we give a little more leaving right, in approximations terms this is also known as by criteria approximation that is just terminology. So, we so instead of using exactly ranking matrix we use k plus p matrix or rank k plus p matrix right Q right.

So, Q is of A is of size n by d Q is of size n by k plus P and B is of size k plus B by d and B and Q is orthogonal matrix. And, B therefore is the projection of the matrix A onto the column space of Q and not we really want is that the Frobenius error A minus QB or spectral error or maybe both right is they are really sort of close approximations of the optimal error A minus A_k Frobenius. And, or the A minus A_k 2 norm right and we further more we also want k plus B to be more or less of the order k right because, if k plus p is very large then achieving this is easy.

(Refer Slide Time: 02:31)



QB: prototype algorithm

Find $\Omega \in \mathbb{R}^{d \times (k+p)}$ a random matrix

$Y = A \Omega$

Find $Q =$ orthogonal basis for $\text{col}(Y)$

Return $(Q, Q^t A)$

And we one prototype algorithm in which we all we did was a multiply the matrix A on the right hand side with random matrix right. And, then we obtain the orthogonal basis for the column space of this projected of this projection Y right and this is my target Q and this is my t .

(Refer Slide Time: 02:51)

Modified Algorithm $|A - QB|_F \sim (1 + \frac{k}{P}) \text{OPT}$

In reality, the bound $|A - QB|_F$ depends on the singular values $\{\sigma_{k+1}, \dots\}$

Bound improves if $\sum_{i \leq k} \sigma_i^2 \gg \sum_{i > k} \sigma_i^2$

$\sigma_k > \sigma_{k+1}$
 $\sigma_k > \sigma_{k+1}$

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

So, we will also saw particular bound for the we also saw particular bound for this for instance we saw that the A minus QB Frobenius, A is can be bounded by 1 plus k by P minus 1 the optimum error right which is the A minus Ak Frobenius right ok. So, now the question I mean can we took better than this right? So in reality turns out that this bound really depends on the singular values from k plus 1 to n. So, in effect what did means is that if the singular values from 1 to k right, that is if the singular values and this is really in the signal that we under capture.

And this k plus 1 to n is really in terms of our model is really the noise or the perturbation right. So, if the signal is much bigger than noise right, then the bound actually improves right then the bound actually then we can then we should actually be able to improve this algorithm improve the performance of this algorithm right. In fact, it is also pretty clear that the bound improves as well as a performance empirical performance of this algorithm right.

But in real matrixes this does not always hold right, what you see is if you really plot the singular value is that the kind of follow more sort of I mean slow dk rather than a very sharp of rather than the ideal case which is pretty sharp right, so that does not happening reality. But can I pre process matrix, so that is something like this is becomes more true right and this is clever way of doing this right which is as follows. That suppose imagine that the similar value of linear dk, so now imagine the right certain squad the similar

values right; then the new sort of plot for this would be let me change the colour of the pen, the new plot for this would be something like as follows right. If I take it to the power, if I take it to the power 4 then it would be even steeper right, it would be even steeper, I mean it would not go could down to 0.

So, but basically it would be even steeper ok, so what it means that if I happened really power the matrix right. Then if sigma 1 is sigma k is bigger than sigma k plus 1 right, then sigma k to the power p is even bigger than sigma k plus 1 to the power p right and so on and so forth ok.

(Refer Slide Time: 05:28)

Power scheme

In reality, the bound $|A - QB|_F$ depends on the singular values $\{\sigma_{k+1}, \dots\}$

Bound improves if $\sum_{i \leq k} \sigma_i^2 \gg \sum_{i > k} \sigma_i^2$

To achieve this, we use $(AA^t)^P A = U \Sigma^{2P+1} V^t$

Handwritten notes on slide:
 $\frac{\sum_{i=1}^k \sigma_i^{2P+1}}{\sum_{i=1}^k \sigma_i^{2P+1}}$ (circled)
 $\uparrow P$ (arrow pointing to the exponent)

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

So, then what we are looking to do is that we basically instead of working with original matrix we work with the power of the matrix right. So, if you for on the particular power you work for the matrix is this particular expression A transpose to the power capital P times A. So, if you just do a matrix decomposition singular value decomposition of the matrix A and just plays it here it would not be hard for you to sort convince yourself that the singular value decomposition of this matrix is really U sigma to the power 2 P plus 1 V transpose.

So, essentially what is happened is that we have kept the same singular vectors both on the left and right, we have just taken the singular values and taken each of them to the power 2 P plus 1. And therefore right therefore, what is happened is that the top I mean the top k singular values now, which are i equal to 1 to I mean k 2 P plus 1 right has

grown much bigger with respect to the entire set of singular values this ratio all i right. This ratio has increases with increasing P right. So, therefore I mean in and this is what will do right.

(Refer Slide Time: 06:41)

Modified Algorithm [HMK10]

```
function [Q, B] = randQB_p(A, l, P)
    Omega = randn(n, l);
    Q = orth(A*Omega);
    for j = 1 : P
        Q = orth(A'*Q);
        Q = orth(AQ);
    end for
    B = Q'*A
```

Total runtime = $O(n\ell + n\ell^2)$

$\ell = k + p \approx k + 5$

$P = 1 \text{ or } 2 \text{ is sufficient}$

Handwritten notes: $A^T = A^T$, $Q = \begin{bmatrix} AA^T & P \\ & A\Omega \end{bmatrix}$

IIT Gandhinagar Indian Institute of Technology Gandhinagar

NPTEL ONLINE CERTIFICATION COURSES

And this is under and this is a modification of that this is a corresponding modification of the algorithm right. That we what we do is that you given a target rank l and A P right and this capital P think of this as 1 or 2 that sufficient and then we generate the omega as we had said right and in this case we are using this is really MATLAB code. So, we are using omega to be a normal I mean each omega each and every omega to be n 0 1, then we multiply A by omega and take it is orthogonalization that is my first Q right and then I start of keep on doing this for P iteration right.

I multiply Q by A star right A star is really A transpose A star is the is A transpose of A and then I orthogonalize it and then I multiply A by I mean Q by A again orthogonalize it right. So, if you if you forget the 2 orthogonalization, then what we are doing is really doing is really A transpose to the power P right times A omega and Q is really orthogonal basis of this right. So, what this look does a it does this repeater orthogonalization in order to maintain numerical stability right and just so just to prevent overflowing and so on and so forth right.

So, these orthogonalization and more practical reasons, but in reality this is what is happening that I that really I find A transfer to the power P times A omega right and then

I take the orthogonal basis of this particular of this particular of this particular matrix right and that is my Q right and in I mean remember that in reality we have to take l to be slightly larger than k I mean something like a plus 5 is enough and while I will not show the exact bound that we get from here because, that is a little complicated expression I will sort of what I will tell you is that it is sort of you something like this. That remembered we had A minus QB and I am just writing down the bound approximately, we had a minus QB bounded by 1 plus k by small P minus 1.

(Refer Slide Time: 08:41)

Modified Algorithm [HMK10]

```

function [Q, B] = randQB_p(A, l, P)
    Ω = randn(n, l);
    Q = orth(AΩ);
    for j = 1 : P
        Q = orth(A'Q);
        Q = orth(AQ);
    end for
    B = Q'A
        
```

Total runtime = $O(n\ell + n\ell^2)$

$\ell = k + p \approx k + 5$

$P = 1 \text{ or } 2 \text{ is sufficient}$

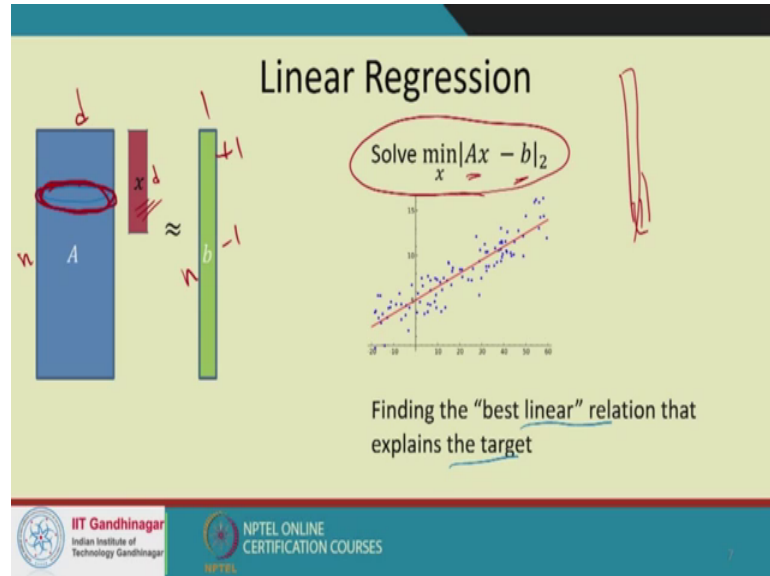
$\|A - QB\|_F \leq \left(1 + \frac{k}{P}\right) \|A - A_k\|_F$

So, remember this is the difference between small P and capital P A minus Ak Frobenius right and now if I instead run this algorithm so this was a bound had before. So, now, if I instant run this algorithm this expression will be raised to the power 1 over P something like one over P or 1 over 2 P 1 over capital P. So, this is capital P right and this is small p, so this small P is really the I mean what we are I mean something like 5 which is an extension of the that. I mean the number of which is giving me the rank the of the size of this random matrix and it is capital P is giving me the number of iterations here right.

So, I get to sort of do 1 over 1 by capital P of this of this of this particular error factor right, which means that make it closer to one ok. So, this is really a practical algorithm and it is been implemented in mat lab and in and in and also and several libraries are available, this algorithm you can find very nice expression in exposition of this by in

Haikou Marlene syndrome ok. So, that ends our discussion our discussion of the QB algorithm of the QB decomposition.

(Refer Slide Time: 10:02)



Now, we will move into the application of the application of this randomised techniques, random projection and sampling techniques to another common problem in machine learning or a linear regression. So, what is linear regression right, so intuitively linear regression is nothing but finding the best linear expression that explains a particular target right. That suppose we have a set of data points, imagine the data points are given in the, imagine that the data points are given in the column space of A in the in the sorry.

The data point rows of A rather not column of space A I am sorry and now b is the target right imagine b maybe a plus minus 1 or b have some set of values and then what you want to find out you want to find out the I mean x is a variable. So, you find out you want to find out A times you want to find out the best x right, such that Ax is closed as closed to B as possible right and 1 very popular version of this is taking the l 2 difference between Ax and b.

So, Ax is this is now sort of size d vector ok. So, let me give number here let us say A is of size n by d b is of size d by 1 and so x is of size sorry b is of size n by 1 and x is of size d by 1 right and can be so the question is that can I find out x right such that Ax is very close to b ok. So, and this is the l 2 error version of this problem the linear regression. So, if it is pretty clear that if d is much larger than n right, if the number of

variables is much larger than n then I can make this error to be a 0 right because, this in that case it is under constraint problem right.

(Refer Slide Time: 11:55)

The slide is titled "Linear Regression" and features a handwritten diagram in red ink showing a 3D coordinate system with axes labeled x_1, x_2, x_3 and a plane representing the span of the columns of matrix A . The text on the slide reads:

Overconstrained setting, $n \gg d$
→ there is no x that satisfies $Ax = b$
Want to find "best" x such that $Ax \approx b$
Statistical interpretation: Find the best "unbiased estimator"
Geometric interpretation: orthogonal projection of b onto $\text{span}(A)$

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

So, what we are interested in is it over constrain setting when n is much n is larger than d , which means there is no x that satisfy Ax equal to b right. So, what we looking to find out is the best x such that Ax is as close to this possible according to the 2 norm right and geometrically this is really the picture. That what we have is let us say we have the we have the space span by the by the columns of A and we have B right and so this particular space is a rank d space right and it lies in R^n . So, B also lies in R^n B is a point in R^n and what we are looking to find is what is the how can I best explain b right by taking linear combinations of these vectors in this in this rank d space ok, this is what we looking to find ok.

Because and we might not be able to explain b completely because b actually lies outside of this space that is entirely possible. So, this is geometric interpretations, so in terms of the statistical interpretation what we are saying is that we are find out for the linear unbiased estimator of b ok.

(Refer Slide Time: 13:20)

Solving Linear Regression

Cholesky decomposition:

- find out upper triangular R such that $R^t R = A^t A$
- solve normal equations $R^t R x = A^t b$

Normal equations:
 $(A^t A)x = A^t b$
 $\frac{1}{\sqrt{d}} \frac{1}{\sqrt{d}}$

QR decomposition: Time taken = $O(nd^2)$

- $A = QR$. Solve $Rx = Q^t b$

SVD:

- $A = U\Sigma V^t$, solve $x = V\Sigma^{-1}U^t b$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, so how do we solve this problem? So the many different ways of many established ways of solving the problem some. I mean one of these settings one of these is for instance by is by looking at the most popular way by looking at the normal equations right. So, the normal equation sort of says that we can look at the equation A transpose Ax equal to A transpose b , that the optimal solution x that we find will satisfied such a equation right. Remember the A transpose A is now of size d by d and A transpose b also rectifies d by d so that optimal x the A star right.

So just 2 sort of said the I mean we call x will call x star with optimal x throughout the optimal x will satisfies this normal equation and this we get by differentiating the objective function and so on and so forth. But once we know this we just need to solve this normal equation and we can do it in very various different ways.

We can do it in Cholesky decomposition we could do it which really works when an a is very well condition, we could do it using QR decomposition of A which works in a is not so well condition, we can also do it using an SVD decomposition of a similar value decomposition of a right. We really works for all A but it is also pretty expensive right. So, the time taken for each of them order nd square theoretically all other constant really differ in I mean depending on the composition that we are using.

(Refer Slide Time: 14:48)

Prototype Randomized Algorithm

Input: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^d$

Create $\Omega \in \mathbb{R}^{s \times n}$, random

Solve $\tilde{x} = \min_x \|\Omega A x - \Omega b\|_2$

The diagram illustrates the transformation of the system $Ax = b$ into a smaller system $\Omega A x = \Omega b$. The matrix A is of size $n \times d$, and b is of size d . A random matrix Ω of size $s \times n$ is multiplied by A and b to produce ΩA (size $s \times d$) and Ωb (size s). The resulting system is $\Omega A x = \Omega b$, where x is of size d . The slide also includes logos for IIT Gandhinagar and NPTEL.

So, can I speed this up this is a question, the here is again the prototype randomized algorithm right an then let us try to understand why this potentially works and then we will try to see this is really efficient. What we want to say is that instead of I do solve this long thin Ax is equal to b Ax equal to b , let us try to reduce this the size of A . So, how do I reduce the size of A I multiplied by a random matrix ω right so and let us say that the size of ω is s π n right.

So, I multiply both A and b by this by this random matrix ω , so therefore now I have sort of a problem of size s by d which is which is a which is ω times A right and then we are still solve A , I try to find the x and we try to make it close to a vector of size. Now a vector of size s right this is ω times b this is the new regression problems that we are solving. This is what we are says taking that suppose if multiply both A and b by the random matrix ω and then we try to solve the new regression problem.

So now, this is much smaller regression problem, therefore it is more coefficient right. And therefore, I mean if I solve this I mean the question is that do I really get a good approximation of the original solution and so that the tentative claims that we want something like this right.

(Refer Slide Time: 16:17)

Prototype Randomized Algorithm

Input: $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^d$

Create $\Omega \in \mathbb{R}^{s \times n}$, random

Solve $\tilde{x} = \min_x \|\Omega Ax - \Omega b\|_2$

Tentative claim: $\|A\tilde{x} - b\|_2 \leq (1 + \epsilon) \|Ax^* - b\|_2$

Running time: If $\Omega \sim N(0,1)$, then $O(nd^2)$

Handwritten red notes: $\|Ax^* - b\|_2 \leq \|A\tilde{x} - b\|_2$

Logos: IIT Gandhinagar, NPTEL ONLINE CERTIFICATION COURSES

Supposing \tilde{x} is the is my is my is my optimal solution here and of the of the smaller problem right and x^* is my is my solution of the original problem. So, that what I really want is that if I take \tilde{x} and if I plug it in the original problem right and see what the error is, remember that this error is upper bounded by $\|Ax^* - b\|_2$ right because by definition of optimality ok.

But what we say is that is $\|A\tilde{x} - b\|_2$ and all of these are 2 norm whatever I am not written it is the vector 1 2 norm, the tentative claim that we are going to say that claim that we going to make that $\|A\tilde{x} - b\|_2$ is not much more than $\|Ax^* - b\|_2$ to the rank 1 plus epsilon and this epsilon will of course of course control the size of s , the size of Ω s the size of Ω .

What is the running time of this algorithm well that comes a bummer right because, now if Ω really is $n \times 0.1$ right. If any if entry of Ω comes from normal 0 1 then size then doing matrix vector multiplication Ω times n takes time nd^2 and so our algorithm is not any not any faster. So, is it really useful right it is not faster is it useful.

(Refer Slide Time: 17:36)

Why would it work?

Geometrically, solution depends on projection of b on the $\text{span}(A)$

$$\underline{\underline{x = (A^t A)^{-1} A^t b = A^+ b}}$$

Similarly, the optimal solution for the sampled problem

$$\underline{\underline{\tilde{x} = (A^t \Omega^t \Omega A)^{-1} A^t \Omega^t \Omega b = (\Omega A)^+ \Omega b}}$$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

13

Let us let us understand why it is potential useful right, in order to see this we need to understand the geometrical picture a little well right. So, remember the pitcher that we do the that we have this span of A star which is really a rank d space in R^n . So, this is a subset of R^n but it is it is a rank d space and here we have b which is a point in R^n and what we looking to find is really the projection of b onto the column space of A right and this error and this is and this is my error or the optimal error right; which is Ax star minus b which is basically the norm of Ax star minus b .

So, now what we did was that we projected each of these A star i each of these vectors A star i into a smaller dimensional space ΩA star i right. So, this is now a subset of R^S right depending on the S R b of and similarly b has been projected from R^n to R^S . So, now this picture gets (Refer Time: 18:37) little right because the because of the random position because of the random sort of projection, I mean if the if sort of the position of if this was if this is really the projection of b on A right and this is really the Ω times the projection of b on A rights. The optimal might be the optimal projection might be somewhere a little different right than the previous point right.

But the point is that they all these plotation are really small right, I mean if the this is the new projection of Ωb onto the onto the column space of ΩA star i it is what we are going to claim, is that it is not very far of from this previous projection right. So, therefore the linear I mean the components x that sort of created that is create this new

projection is not a not very different from the components x that that get this from the new projection ok.

So, mathematically here is what is going to be that if you look at the I mean look at the normal equations this is this is how we get the optimal x, that than I take A transpose A see imagine for now that is rank d, I mean A transpose A is a full rank d matrix we can extend all of this to the to the rank diffusion case easily. So, then x star is really A transpose A inverse time A transpose b.

Which is the, which also be the witness the pseudo inverse A rank b right. The optimal solution for the for the sample problem is now instead of A I replaced it by A pi omega times A right and therefore it is a it is this just particular quantity right. So, this A should not be here right, so I mean it this particular expression just by looking at the normal x the normal equation of the of the sub sample of the projected problem.

(Refer Slide Time: 20:31)

Guarantee

At its core, uses a matrix concentration type inequality to show that $|U(I - \Omega \Omega^t)U^t| \leq \epsilon^2$, U = left singular vectors of A

Holds for \mathcal{G} satisfying the JL lemma with (ϵ, δ) . Not always more efficient.

Handwritten notes:

- $U \approx \frac{1}{\sqrt{r}} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 & & & 0 & 0 & 0 & 0 \end{bmatrix}$
- $U \approx \frac{1}{\sqrt{r}} U U^t \approx \frac{1}{\sqrt{r}} U U^t \epsilon n r \nu$
- $\text{rank}(U) \approx 1 \pm \epsilon$
- $\downarrow \approx \frac{1}{\sqrt{r}}$
- $\downarrow \approx \frac{1}{\sqrt{r}}$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Now, at it is core what we are going to mean what this proves says is that we can prove matrix concentration type inequality, to show that this particular matrix has a 2 norm which is less than epsilon ok. So, remember what this I mean just to un make you understand what this what this matrix is, this matrix has I mean this particular matrix has a 2 norm which basically I mean not less than epsilon which is less than epsilon right ok.

So, what it means that $U \Omega \Omega^T U^T$ is more or less similar to UU^T . So, remember that UU^T is the projection matrix right, so therefore it has a singular set of singular values 1 and then and then 0 right. So, it is a similar value that are 1 and everything else is 0 right. So, it is a n by n projection matrix and this is again n by n projection matrix right. But it is an n by n projection matrix of much smaller rank of rank s right and what you gone to say is that, is that if you can look at singular values all of them. In the non 0 singular values all of them are going to be about 1 plus minus epsilon in the range $1 - \epsilon$ to $1 + \epsilon$ this is what this particular bound says and we can prove this using matrix concentration inequalities.

I mean even using the matrix multiplication theorem that we did earlier and the randomized matrix multiplication theorem that we did earlier and sort of plugging in suitable values, but it is not right. So, then so then I mean it is so if I mean again then what is the s or other what is Ω that we need to choose. We need to choose Ω that essentially satisfies JL lemma with factor with a error bound ϵ and with δ to be like δ is said to be like $1/d$ or $1/\text{poly } d$ right.

(Refer Slide Time: 22:45)

Guarantee

At its core, uses a matrix concentration type inequality to show that $|U(I - \Omega\Omega^T)U^T| \leq \delta^2$, U = left singular vectors of A

Holds for Ω satisfying the JL lemma with (ϵ, δ) . Not always more efficient.

Handwritten notes:
 $\text{rank}(s)$
 $1 \pm \epsilon$
 $\|1\|_1 \ 000000$
 $s \sim \frac{d \log d}{\epsilon^2}$

Which means that if you take Ω if you certainly, if you take the size of Ω to be if you take size of Ω to be something like $d \log d$ by ϵ square it is sufficient ok. So, now comes the question that this is given with s to be something like d right. We

show that we show that the random I mean even if s was d then multiplying omega times a takes time nd square right. So, if s is d log d epsilon square then it even slower right which is slower than solving the linear regression problem itself. So, what we are doing anything useful right.

(Refer Slide Time: 23:23)

Randomized Hadamard based projection

$\Omega = PHD \in \mathbb{R}^{s \times n}$

$\tilde{H} = n \times n$ Hadamard matrix *normalized*

$D =$ diagonal random ± 1

$P =$ can be just a sampling matrix now, i.e.

For each $t \in [1, s]$

$P_{t*} = e_i \cdot \frac{\sqrt{n/s}}{n}$ with prob $1/n$

Handwritten notes:

- $\tilde{H}_n = \frac{1}{\sqrt{n}} H_n$
- $H_n^2 = \begin{pmatrix} H_n & +I_n \\ H_n & -I_n \end{pmatrix}$

Diagram: A matrix P of size $s \times n$ with a single non-zero element in each row.

Logos: IIT Gandhinagar, NPTEL ONLINE CERTIFICATION COURSES

It done for the solution to the this is something we have already seen right, we have seen this randomized Hadamard based random projection. This randomize Hadamard transformation it is going to say right and just recall for the randomize Hadamard transformation was we say that we take the Hadamard matrix, which is really define by this Hadamard square root $n H n$ and $H n$ is to find in this in this recursive manner ok.

And this is really that normalized the normalized version then we take the diagonal matrix plus minus 1, that that randomised transformation right and further more instead of the I mean after that we had a sparse go symmetric, we do not really need a sparse go symmetric right now what we can do is that we can just take a sampling matrix. So, what is really a sampling matrix sampling matrix is a something very simple we say that for it is a s by n matrix right and for generating every row of this matrix we toss a we toss a n sided coin right and the coin says that for this matrix for this a for this row I am going to put a non 0 at the i th position right and everywhere else will be 0.

So, the i th position you just normalize I mean there is a normalisation factor which is this square root n by s x and you put in square root n by s are the i th position and you put in

the everywhere else 0 everywhere. Else say every row puts a select only one position try to put in a nonzero entry that is it. So, this is and this is and this happens and happens with equal probability for all the rows and all the and all for all the positions and all the rows adjust IID adjust IID random variable from the same distribution ok. So, this is my matrix this is my omega P H D.

(Refer Slide Time: 25:27)

Running time

To ensure $|A\tilde{x} - b| \leq (1 + \epsilon)|Ax^* - b|$, need $s \geq Cd \log(d) / \epsilon$

Time for projection = $O(nd \log n + sn)$

Time to solve = $O(sd^2)$

Overall time = $O\left(nd \log n + \frac{d^3 \log(n)}{\epsilon}\right)$, ignoring lower terms

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, now it should feel is simple to say that that calculating this P H D takes time is much faster right, that calculating this P H D takes time only in $d \log n$ right plus sn this is the time for protection and s and as I said before it is enough to take s to be at least $c \log d$. I mean C to be $d \log d$ by $d \log d$ by ϵ $d \log d$ by ϵ and then we can sort of ensure that we get an ϵ approximation $1 + \epsilon$ approximation right. That the x still other I find out from the smaller problem is really one plus ϵ approximation to the x star to the to the original problem right.

So, the time taken to solve the final problem the smaller problem right so, first so the total time taken is the taken for the projection for the time taken to solve this problem right. So, time taken for projection we are already mentioned, the time taken to solve smaller problem is that sd^2 because. Now I have a s by d s by d matrix s by d linear regression problem, so therefore the overall times. So, pugging in the value of s the overall time something like $nd \log n$ plus d plus $d^3 \log n$ by ϵ plus some other

terms right and this happens be faster than solving the linear regression for if n is very large and these are the an these are the guarantees.

(Refer Slide Time: 26:36)


Guarantees



1. $|A\tilde{x} - b| \leq (1 + \epsilon)|Ax^* - b|$
2. $|\tilde{x} - x^*| \leq \sqrt{\epsilon} \kappa \left(\frac{1}{\gamma^2} - 1\right)^{\frac{1}{2}} |x|$

$\kappa = \text{condition number of } A$

$\gamma = \frac{|U^T b|}{|b|} = \text{captures how much of } b \text{ is present in span}(A)$

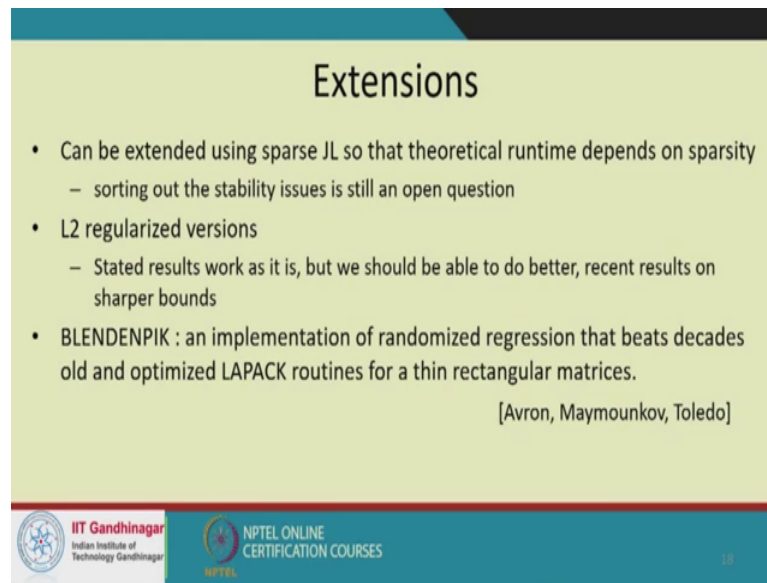
$\frac{\sigma_{\max}}{\sigma_{\min}}$



So, this is the guarantee that we have seen before, further more we can also guarantee that the I mean under certain cases we can also guarantee that the solution \tilde{x} itself is closed to x^* right. But, for the this we need to make sum assumptions we need to because this bound is depended on the condition number of A , remember condition number of A is the ratio of the maximum singular value to the minimum singular value right. So, the $x - x^*$ depends on condition number of A , it also depends on how much of b is present in span of A , it is dependent on this ratio which is depend which is the data dependent data dependent bound of course.

(Refer Slide Time: 27:19)



The slide is titled "Extensions" and contains the following content:

- Can be extended using sparse JL so that theoretical runtime depends on sparsity
 - sorting out the stability issues is still an open question
- L2 regularized versions
 - Stated results work as is, but we should be able to do better, recent results on sharper bounds
- BLENDENPIK : an implementation of randomized regression that beats decades old and optimized LAPACK routines for a thin rectangular matrices.

[Avron, Maymounkov, Toledo]

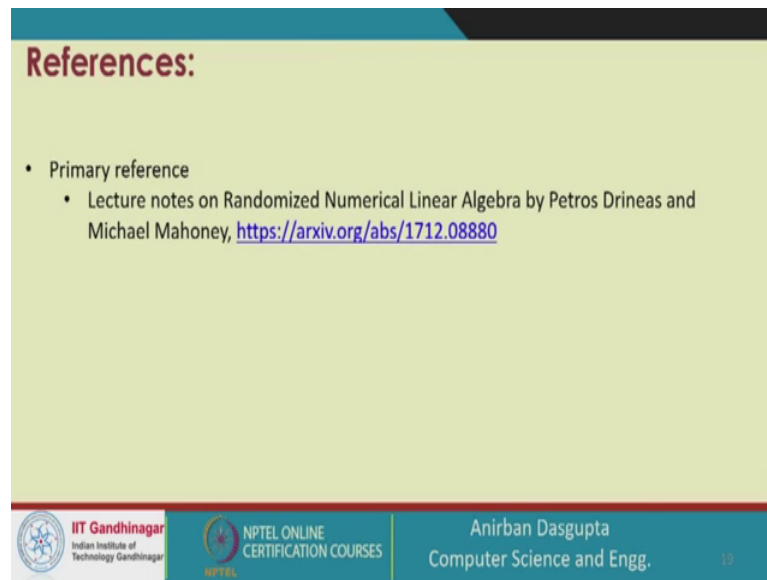
At the bottom of the slide, there are logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES. The slide number 18 is visible in the bottom right corner.

So, just to mention the extensions that I mean we have shown the this particular random projection we can extended it to take care of the Sparsity of the problem right. We can make sure that we I meant once we use the sparse Jonson Linerar Shros or it is variant I mean there is a more interesting variant also called as optimize Jonson linear transformation subscription projection, then the theoretical run time matrix depend on Sparsity of the problem.

However, I mean it is not clear the that is entirely I mean a stable algorithm in practice, I mean the questions of numerical stability are still yield to be resolved is not been any good experimentation there as far as I know. We can also look at there is also be results about regularized version of this problem and specifically about the of the l 2 regularized versions.

There has been there has been some results that says that if we using regularization, we should actually be able to project although much smaller on number of dimensions right. There is a very nice implementation of it call Blend and pick which shows that that at least for a class of matrixes thin rectangular matrixes. I mean there is implementation of this randomize regression that can beat the decades old optimized LAPACK routines for a in terms of performance for this class of matrixes right.

(Refer Slide Time: 28:37)



References:

- Primary reference
 - Lecture notes on Randomized Numerical Linear Algebra by Petros Drineas and Michael Mahoney, <https://arxiv.org/abs/1712.08880>

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Anirban Dasgupta
Computer Science and Engg.

19

And the so just an most of the lecture just to give the reference, most of the lecture was from these lecture notes by Petros and Michael. And, if you are interested in looking at the papers you should look at the paper by blend and pick by hibernate all, as well as the other references that will be put on the webpage right.

Thank you.