

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

Lecture – 15 a
Introduction to Randomized NLA Matrix Multiplication

Welcome to the course on Scalable Data Science. Today's lecture is on Introduction to Randomized Numerical Linear Algebra. I am Anirban, I am from IIT Gandhinagar.

(Refer Slide Time: 00:26)

Numerical Linear Algebra

- Large fraction of ML problems boil down to solving a linear algebraic optimization
 - recommendation systems → matrix factorization
 - ranking → eigenvector calculation
 - supervised models → regression with appropriate loss
 - ...
- Such matrices are typically “big”, or the method is computationally expensive

$O(nd \min(n, d))$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, numerical linear algebra ok; if you are familiar with the machine learning, you probably know that a large fraction of machine learning problems essentially boil down to solving a linear algebraic optimization problem right. Any kind of model that you are learning right, whether it be a very simple logistic regression model or it be a very complicated deep neural network model, it is you are essentially solving a problem an optimization problem in some linear algebra space.

So, in fact there are specific types of linear algebraic optimizations that are not to be very useful in specific branches of machine learning. The entire branch of recommendation systems right is really all about matrix factorization ok. Under certain constraints, under assuming certain priors, by sort of redefining the last function, it is all about matrix factorization in various forms right.

Ranking computing page rank is really just an eigenvector calculation right. You redefine the transition matrix in various forms, and then you calculate the corresponding eigenvectors. And this is also true for specific kinds of let us say community detection or clustering right. These are these all boil down to certain types of eigenvector calculations eigenvalue calculations right. And as I mentioned that a law I mean basically any supervised model is doing regression with an appropriate loss it is (Refer Time: 01:59) regression with an appropriate loss function.

So, such matrices the matrices that, we sort of solve the linear algebra problems in are typically big right, because of the I mean, and this is what the big of the big data really comes in right. If you imagine that, you are solving a recommendation system at the Amazon prime or at the Amazon prime level or at the Netflix level, we have like millions of movies, and we have millions of users, and we have hundreds and thousands of movies at least millions of users ok. So, you have a huge matrix, which is also very sparse.

And furthermore these methods are also often computation expensive, you have seen SVD the singular validity decomposition right that takes time order $n \times d$ times n plus d , which is at least quadratic right. And, so I mean $n \times d$ times in $n \times d$ order $n \times d$ times mean of $n \times d$ right something like this ok, which is fairly large. And it is not really realistic for any for any practical data set. There in hundreds and thousands, and these are the of the order of millions let us say ok.

(Refer Slide Time: 03:08)

Randomized Numerical Linear Algebra

- Randomization (sampling, sketching) allows us to build algorithms that can
 - return approximate solutions (with a quality control parameter)
 - succeed with high confidence
 - scale to massive data
 - have improved computational complexity

[Slides courtesy Michael Mahoney]

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, the idea behind randomized numerical linear algebra is this essential sort of intuition that, the concepts in ran the certain concepts in randomization for instance sampling being able to sample from the data, being able to sketch from the data allows us to build algorithms that, can be efficient much more efficient, and can actually scale to massive data. But in order to do so you have to give in certain things.

Number 1, you cannot hope for exact solutions right. Because an and I mean, because once you sample right I mean, and you are trying to solve the problem on the resulting data, you cannot hope for a solution that is exactly I mean the solution, that is exactly optimal for the original data right. You have to be you have to deal with approximate solutions right. But what will give you is a specific quality control parameters, for instance, we will be able to tell you that ok, this is not more than 1 percent far off from the actual solution.

We will also tell you right I mean a confidence parameter in the sense, that given the algorithm right. We will be able to tell you ok, that this algorithm is supposed to succeed with only 90 percent chance, which means the 10 percent of the cases I might return you something bad, but if it is a fact if it is I mean mostly in reality it is not going to be really bad right. It is going to be but the theoretical guarantee is only hold with the high confidence. And in a lot of sense is fine, because once you are in machine learning applications, it is not really I mean we are not really looking for the actual optimum in any way right, because I mean if it is a classifier, and we were doing I mean fine it is we do not really need to get to an actual optimal ok.

(Refer Slide Time: 04:48)

The slide is titled "Basic Theme" and contains two main bullet points. The first bullet point states: "By sampling rows/columns/entries of matrices, we will create a smaller representative matrix" with a sub-point: "– solve the problem on the smaller matrix". The second bullet point states: "Alternately, take random combinations of rows/columns of original matrix" with a sub-point: "– random projections". To the right of the text, there is a hand-drawn diagram in red ink showing a large grid with several rows and columns, and a smaller grid below it, illustrating the concept of sampling or projection. At the bottom right of the slide, there is a small inset video frame showing a man speaking. The slide footer includes the logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES.

So, what are the basic thing in randomized numerical linear algebra? The basic theme is that is a following that, we think about representing the data as a matrix, and now we are doing operations on this matrix. Maybe we are doing a matrix factorization, maybe we are sort of solving a regression problem using this matrix, whatever right. What will see is that ok, we will sort of to choose specific rows or specific columns of this matrix or maybe a combination of both. And this will give me a smaller matrix somehow it will give me a smaller matrix right. And then, we will sort of then we will solve the problem on the smaller matrix right.

So, we have to be able to solve it in a way, so that we can say that there is a resulting solution is a good solution with a original problem ok. So, sometimes we live in of course, step (Refer Time: 05:35) and said that ok. We are not even choosing individual rows and columns of the matrix. What we are doing is that, maybe we are sort of maybe we are summarizing a bunch of rows, using a single row or maybe by summarizing a bunch of columns using a single column right. Rating a random combinations and then, we are creating a new matrix. So, we are creating a shorter matrix (Refer Time: 05:55) batch is sampling one by doing random combinations right. This is essentially what random projections just does right; if you remember, if you sort of think about it, so and then, we solve the problem on the resulting matrix ok.

(Refer Slide Time: 06:05)

The slide is titled "Matrix Multiplication". It contains the text: "Given $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, find out AB ". Below this, there are two diagrams: a rectangle labeled 'A' with dimensions m (height) and n (width), and a vertical rectangle labeled 'B' with dimensions n (height) and p (width). To the right of these diagrams, there is handwritten red text: $O(n)$, $O(mnp)$, and $(AB)_{ij} = \sum_k A_{ik} B_{kj}$. Below the diagrams, it asks "Can we find an approximate solution faster?". In the bottom right corner, there is a small video inset showing a man speaking. At the bottom of the slide, there are logos for "IIT Gandhinagar Indian Institute of Technology Gandhinagar" and "NPTEL ONLINE CERTIFICATION COURSES".

So, in this particular lecture, we will see a very simple, but very sort of clever example of such a algorithm right. So, we have all seen the we have all seen in our basic algorithms course that, the problem of matrix multiplication right. We even do it before we done the algorithm course. For instance, that suppose we that suppose, we are we given 2 matrices A which is of size m by n right and B which is of size n by p, and we have to find out the product of the of these 2 matrices A times B right.

So, the naive algorithm takes time order $m n p$ right, because just two I mean for every if you take every row of A, you take every column of B and that takes and that takes order n right. And you have to consider m times p such combinations right, because every row goes with every column right. So, and thus no cheating around this, unless the matrices have specific structure right. So, here is a question can I find, if we are not interested in the exact solution, but in an approximate solution, can I find an approximate solution faster ok. So, for this we have to define that what do we mean by an approximate solution first of all. And Secondly, how can you even think about doing this faster right. And this is where our first introduction into using randomization will be ok.

(Refer Slide Time: 07:42)

Matrix multiplication

• View the product as a sum of rank-one matrices

$$A B = A_{*1} b_{1*} + A_{*2} b_{2*} + A_{*3} b_{3*} + \dots$$

A_{*i} = i -th column
 A_{i*} = i -th row

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, here is a title different way of looking at the of looking at the matrix multiplication, and that will come to use right. Remember until, now that we have been looking at matrix multiplication as sort of taking every row of A and choosing a column of B right and multiplying it to get an entry right. So, basically AB_{ij} is summation over k $A_{ik} B_{kj}$ right. This is a row of A this is the under odd product of the i th row with the j th column. So, but this view is not very useful for us. The view that is useful for us is a little unintuitive, but once you get used to it is surprisingly useful right. What we see is that right that, we can also view the product AB as a sum of rank one matrices.

So, what are these rank one matrices? So suppose, so think of the columns of A right and think of the rows of B right, the columns A and the rows of B. So, now, if you think about it, right I mean A is nothing really, but A collection of its columns and B is a collection of its rows right. So, the dot product AB can also be written as you take the outer product of the first column of A, let me write that at A_{*1} and the first row of B write as a b_{1*} . So, I am using MATLAB type notation right.

So, A_{*i} will be the i th column and A_{i*} will be the i th row. This is what MATLAB uses and I find this clear enough. So, I can write I can also write AB as the product of the as a sum of as a sum of rank 1 matrices each of them is formed by the outer product of a column of A and that row of B.

(Refer Slide Time: 09:49)

Matrix multiplication

A

B

=

p_1

+

p_2

+

p_3

+

\dots

Design $p_1, p_2 \dots p_n, \sum_i p_i = 1$
 For $t = 1 \dots c$
 pick the i^{th} rank one matrix in the summation with probability p_i , with replacement
 normalize this matrix by $\frac{1}{p_i}$
 Approximate this sum by the c terms

$\frac{1}{c} (| \text{---} + | \text{---} + | \text{---} + | \text{---})$

IIT Gandhinagar
 Indian Institute of
 Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

So, this is A star 1 with B 1 star A star 2 with B 2 star, and A star 3 with B 3 star, so basically as a sum of n rank one matrices. Each of them is an outer product of 2 of a row and of a column of n row of B ok. So, now, our sampling idea comes into picture. Now, what we say is that, if I were calculating this exactly, I would have to calculate the sum of n terms ok. Suppose I am not interested in calculating suppose I am interested only in approximating. It not interested in calculating it exactly. How about we do this, suppose we assign a probability to every into each of these terms right.

Suppose p 1 is the probability of the first one, p 2 is the probability of the second one p p 2 is a let us I mean let us consider these weights for now. Weights are probabilities that ripping having assign p 3 is the probability there that is going to assign the third one and so on and so forth. So, summation p i equals to 1. We will see how to define this p i's, but suppose you had this pi's, then here is what I want to do right. So, I am going to sample (Refer Time: 10:43) times ok. And each time I sample I am going to pick one of these rank 1 matrices ok.

So, the so for each of the c samples we pick the ith rank 1 matrix with probability p i right with replacement. And then, you normalize the matrix with the lower p i right. So, now, I have a sum of c terms right. And then we basically just average that by 1 by c average that, I mean divide that by c and this is my estimator right. So, basically this is my this is my this is the estimate of of A times B, that I will return ok.

(Refer Slide Time: 11:34)

Example

$$AB = \sum_{i \in [1, n]} A_{*i} B_{i*} \approx \frac{1}{c} \sum_{t \in [1, c]} A_{*j_t} B_{j_t*}$$

$a_i = \text{col vec.}$
 $b_i^T = \text{row vec.}$
 $p_1 \quad p_4$
 $c=1$
 $c=2$
 $c=3$

So, we will sort of see a little more I mean example of this, that supposing let us kind of go over it in a little bit details that, supposing a was a 1, a 2, a 3, a 4, right. These are the columns of a and b was let us say b 1, b 2 b 3, b 4 right. So, now actually let me stick to the same notation. So, let me write b as b 1 transpose, b 2 transpose, b 4 transpose. So, when I write I mean a i a i that is a column that is a column vector right, which is what which is like when I am writing down A row of B I am I am writing it as b i transpose, this is a row vector ok. So, this is now equal to summation a i b i transpose ok.

This is this is what it is basically a i is A star i and b is bi bi transpose is B i star ok,. So, what is happening here what is happening here is, now is now suppose we are choosing three terms c equal to 3 ok. Suppose, suppose we define p 1, p 2, p 3, p 4 right. So, now, what we will do is that for c equal to 1 right. I mean I will take a 4 headed dice right. So, the 4 headed dice it will turn out to be it will sort of evaluate to, so the random variable can take value 1 with probability p 1 with value it can take value 2 with probability p 2, value 3 with probability p 3, value 4 with probability p 4. So, I try toss this dice right. And let us say that the index that comes up is 2. So, for c equal to 1 I take a 2 b 2 transpose right. And I divide that by 1 by p 2.

So, I sort of do this index choosing again for c equal to 3 for c equal to 2 right. So, let us say it turns out to be, now the random variable says is that I mean based on the probabilities p 1, p 2, p 3, p 4, I will give you the index 1. So, I take a 1, b 1 transpose I

normalize that, by p_1 by p_1 right. And let us say I do it one more time, and it again turns out to $b_1 c$ equal to 3. So, $b_1 b_1^T$ transpose 1 by p_1 right. So, this is my final sum. So, this is what I returned, and I normalize this by 1 by c ok. This is my estimator of a times b . So, it happened with replay with replacement, and then, I sort of did it c times.

(Refer Slide Time: 14:20)

Algorithm

1. Pick c columns of A with replacement to form the matrix C and the corresponding rows of B to create R

2. Calculate CR and return

Notes: The columns (rows) are picked with non-uniform probabilities and scaled appropriately

Handwritten notes on slide: p_1 , p_n , $t=1 \dots c$

So, an equivalent way of looking at the algorithm is as follows right that from the matrix A and the matrix B we create 2 matrices C in R . So, what is C in R , I mean a column of C right is really picked from a column of A ok. So, think of the choice that we were doing right that for the for t equal to 1 to c right. We were choosing an index from 1 to n with probability p_1 to p_n right. So, now that I mean what you can also says that that choice of index let us say for t equal to 1.

We choose column number 10, we choose we choose we choose i equal to 10, which means that you take the column number 10 of A and the row number 10 of B right. And put it in position 1 of C and position 1 of R . Put it as a first column of C and the first row of R right. Similarly, for t equal to 2 right. If the random variable say is that you should be choosing column number 100, then you take the hundredth column 100 of A , then you take the 100 column number 100 of A , and the column number and the row number 100 of B right.

And then, and then put it in the in the second column I mean put the put the column of A in the column of C , and put the row of B in the row of R ok. So, basically and then and

then, it is not very hard to see that the I mean and of course, do appropriate normalizations right. And we will see how will do the normalizations in a little bit more, but, but once you define C in R and you add in the normalization add in you do the scaling correctly right. And then, you sort of do a 1 by c right, then the product C R is what we will is what we will sort of is what we have been looking at the, so the summation that we are looking at is really nothing but this product of the 2 matrices t times R. Now, let us going little bit details about how C is being normalized R is being normalized and so on right.

(Refer Slide Time: 16:19)

How to choose the probabilities

Choice 1: Uniform, $p_i = \frac{1}{n} \forall i$

What we want is $AB \approx CR$

Imagine we had numbers $a_1 b_1, a_2 b_2 \dots a_n b_n$ and we want $\sum_i a_i b_i$

Handwritten notes:
 $C =$
 $x_1 = x_2 = \dots = x_n =$
 $\sum_i = \sum_i b_i$

IIT Gandhinagar Indian Institute of Technology Gandhinagar
 NPTEL ONLINE CERTIFICATION COURSES

So, the choice 1 is, so first let us sort of see is that how do we choose the probabilities right sort of so here is the first obvious choice. Then do not try to do anything smart right. Make all the p_i is 1 by n right. So, that means, that at that at every point for every for every for every t equal 1 to C, I mean you choose you I mean uniformly one of the columns of A, and the corresponding row of B. So, remember that the choice in A and the choice in B have to be locked right. In the sense that if you if you choose the ith column of A, you have to choose the ith row of B we make these two choices together, and then you put it and then, you put it in put them in C in R ok.

So, we will sort of see how to do this in little more details, now. So, what we really want right. So, what we want is that the product A B should approximately equal the product C R. And we will again define what do we mean by approximate right. So, before we even

before we even sort of a go to matrices, let us imagine that we had n numbers right. And just to keep sort of I mean similarity with the with a matrix setting that we have sort of written, I call the first number to be to be a 1 .

So, let us say x_1 equal to $a_1 b_1$, x_2 equal to $a_2 b_2$, and x_n equal to $a_n b_n$ right, I mean a_n, b_n is a single number. And we want to estimate the sum of this summation $a_i b_i$. So, summation x_i is equal to summation $a_i b_i$ this is what we on estimate. Now, we want to say is that that suppose we want to do a uniform choice right that uniformly choose I mean so choose C numbers out of out of 1 to n right.

(Refer Slide Time: 18:04)

The slide is titled "How to choose the probabilities". It contains the following text:

- Choice 1: Uniform, $p_i = \frac{1}{n} \forall i$
- What we want is $AB \approx CR$
- Imagine we had numbers $a_1 b_1, a_2 b_2 \dots a_n b_n$ and we want $\sum_i a_i b_i$
- Choosing uniformly has high variance, some $a_i b_i$ could be very large

Handwritten red ink notes on the slide include:

- $x_i \sim 1$ (with a tilde symbol)
- i (with a tilde symbol)
- $a_i b_i$ (with a tilde symbol)
- n (with a tilde symbol)
- $\frac{\sum x_i}{c}$ (with a tilde symbol)
- 0 (with a tilde symbol)
- 0 (with a tilde symbol)
- 0 (with a tilde symbol)
- 1 (with a tilde symbol)

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

And for each number choose 1 from 1 of the numbers I mean x_i , which is a_i, b_i . And then and then just normalize it by n multiplied I mean normalize it I mean basically just choose summation x_i and then divide by c multiplied by n ok, where x belongs to the i belongs to the sample. x the j th sample let me write it this way the j th sample is j_i for j belonging to the sample ok. So, now, this is something that we have seen.

So, how can I mean does any (Refer Time: 18:42) choice work well here? Well, you could see that not really right, because it could happen that one of the exercise is very big that supposing one of the exercise is 1 and everything else is basically 0 . So, then summation a_i, b_i is 1 right, but because we are doing uniformly random choice, because you are not even looking at the values, we could be ending up we could end up choosing the 0 value over and over again. And therefore, what we will get is is 0 right. I mean at

the end, because all our sample is going to be 0 right. And so and so that is it is not it is we can take this intuition you can formalize it and say that choosing uniformly essentially has very high variance, because some of the a_i , b_i could be very large. So, what is the better way of choosing it.

Well, at least in this case a better way of choosing it is according to the value right, but if you happen to choose the i mean the i th value with the i th i mean, if you happen to choose i with probability that is proportional to x_i right, then we will actually get a good estimate of the sum right, that is if the probability is proportional to $a_i b_i$, then we will get a good estimate of the sum. Now, we are starting to get intuition about what we should do in the matrix case right.

(Refer Slide Time: 19:50)

How to choose the probabilities $\|A_{*i}\|_2 =$

Weighted by row and column norm, $p_i = \frac{|A_{*i}||B_{i*}|}{\sum_j |A_{*j}||B_{j*}|}$ ← normalization

For $t = 1 \dots c$
 choose j_t column of A and row of B w.p. p_{j_t}

include $\frac{A_{*j_t}}{\sqrt{c p_{j_t}}}$ in C and $\frac{B_{j_t*}}{\sqrt{c p_{j_t}}}$ in R

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

12

So, in the matrix case right the right sort of probabilities happen to be happen to be this right, that you take the k so, A_{*i} is really the l_2 norm of the of the of the i th column of A and B_{i*} is the l_2 norm of the i th row of B right. So, we define p_i to be proportional to the product of the of the of the l_2 norm of A_{*i} and the l_2 norm of B_{i*} right. And then this denominator is really is essentially just normalization right, because we because we normalize we have to ensure that that summation p_i equal to 1 ok. And here is what we do right that. For t equal to 1 to C , we choose the j th column of A and j th row of B with probability p_j .

And if I happen to choose this right, then I normalize it by this quantity $A_{j,t}$ is what my job A is what I choose. And then I normalize it by square root c times $p_{j,t}$. And then I sort of so I include this normalized column in C and then, I include this normalized column in R right. So, why did the square root come in, because if you remember that, C in R , I will going to be multiplied to each other right. And therefore, and I mean I essentially wanted to normalize the sum only by 1 over $p_{j,t}$ and then outside by 1 over C right, but now, because I am distributing the normalization and putting it some in C and some in R right. I am normalizing each of them by the square root of the final normalizer. Basically, it is that simple ok.

(Refer Slide Time: 21:43)

Algo in matrix notation

We can write this in terms of a sampling matrix $S \in \mathbb{R}^{n \times c}$ as

Each column $t \in [1, c]$ has single nonzero entry

$$S_{j_t, t} = 1 / \sqrt{c p_{j_t}}$$

$$CR = (AS)(S^t B)$$

$\begin{matrix} c \\ \hline \begin{matrix} \uparrow \\ \downarrow \end{matrix} \\ n \end{matrix}$
 $\begin{matrix} p_1 \\ \vdots \\ p_n \end{matrix}$

So, then so this is what it is right and then here I mean here is either way of looking at it from the matrix notation right. What if what we could also say is that that let us create a sampling matrix, sampling matrix of size n by c . So, this sampling matrix what it will have it is of size n by c . What it will have is that for every column, it has a single nonzero entry, and that nonzero entry will turn out to be j_t at the position j_t, t right. And it will have 1 by square roots C times $p_{j,t}$. So, basically what you could say is that in order to decide where to put the nonzero.

We have the probabilities we have the probability is p_1 to p_n , in order to decide the where to put the nonzero entry in the in the in the t th in the t th column right. You toss this n headed dice right, so that you get i with probability p_i and if you get or rather, so

that you get the j t with probability p_{jt} . And if you happen to get j t right, then you sort of put in a normalizer of 1 over square root put in 1 over the value, 1 over square root C times p_{jt} in the in the position j t comma t right. And then put in zeros everywhere else in the in this particular column.

So, this is my matrix S , now again it is not very hard for you to solve sort of C is that the C that, we defined in the algorithm before is really the matrix A times S . And R that we defined is really the matrix S transpose times p . And therefore, we what we can also look at is that we create this matrix S and then, we calculate we essentially calculate our project the 2 matrices A and B using this matrix S . We only have to make sure that the matrix S is really the sampling a projection matrix is the same for both A and B .

(Refer Slide Time: 23:30)

Simple guarantees

$$(CR)_{ij} = \frac{1}{c} \sum_{t \in [1,c]} \frac{A_{it} B_{jt}}{p_{jt}}$$

Easy to see: $E[(CR)_{ij}] = (AB)_{ij}$

$$\text{var}((CR)_{ij}) = \frac{1}{c} \sum_k (A_{ik}^2 B_{kj}^2) / p_k - \frac{1}{c} (AB)_{ij}^2$$

Handwritten red annotations on the slide include: $t - j_t$, $E[(CR)_{ij}]$, $E\left[\frac{A_{it} B_{jt}}{p_{jt}}\right]$, and circles around the expectation and variance formulas.

So, now what are some guarantees ok. So, let us look at the ij th entry of CR right. So, the ij th entry of CR right is really a sum right. So, it is really A sum of C terms right. And remember that for the t th term we had chosen the j th the j th column of A and j th row of B . So, therefore the t th term right will really be a i j t times b j t j divided p j t right.

And this the entire thing 1 get divided by 1 over c ok. So, this is and here j t are the random variables of course, ok. So, it is easy to see that the expectation of CR i j is really a b i j right, because once we start taking I mean once we, because if I take the expectation of expectation of CR i j right, because each of the t choices is i i d right I mean all these have the same expectation. And therefore, I could as well calculate the



expectation of $A_{i,t} B_{j,t}$ and $b_{j,t}$ divided by $p_{j,t}$ right. And then there is a 1 by c and then, there is a c times this and this c and this c cancels out. So, this the second c comes, because the summation every term in the summation has the same expectation ok.

(Refer Slide Time: 25:26)

Derivation

$$E \left[\frac{A_{i,t} B_{j,t}}{p_k} \right] = \sum_{k=1}^n \left(\frac{A_{i,k} B_{k,j}}{p_k} \right) p_k$$

$$= \sum_{k=1}^n A_{i,k} B_{k,j} = (AB)_{i,j}$$

So, now what is this quantity let me just do a sort of quick calculation of this $A_{i,t} B_{j,t}$ divided by $p_{j,t}$ right. So, we know that this value can take one of n possible. This random variable can take n of impossible values right. And the n possible values are i equal to 1 to n. It can take value let me actually call it k equal to 1 to n $A_{i,k} B_{k,j}$ divided by p_k . So, this value happens when the column k of A and the and the corresponding row of B is chosen, and that is chosen with probability p_k right.

So, this is the value chose taken by there is the value that the random variable assumes is the probability of that right. And this and this cancels out which is the exact, which is exactly the point of this normalization and equal to summation k equal to 1 to n $A_{i,k}$ times $B_{k,j}$ right, which is equal to $AB_{i,j}$ right. So, then what we then see is that coming back here, is that the $C R_{i,j}$ has exactly the right expectation right.

Again you can do this calculation not very hard to say not only hard to do is that, the variance of $C R_{i,j}$ is this quantity which is one over C times the summation of the of the product of the squares of $A_{i,k}$ and $B_{k,j}$ divided by p_k minus the minus this quantity this is one by $C B_{i,j}$ square. So, one thing to notice that I mean if nothing else what you see is that if c increases by the variance decreases so, just to be expected.

(Refer Slide Time: 27:02)

Error in Frobenius norm

We want to bound $\|AB - CR\|_F$

$E\|X\|_F^2 = \sum x_{ij}^2$

$$E[\|AB - CR\|_F^2] = E[\|AB - (AS)(S^tB)\|_F^2] = \sum \text{var}(CR)_{ij}$$

Using the discussed values for p_i allows us to bound

$$E[\|AB - CR\|_F^2] \leq \frac{1}{c} \|A\|_F^2 \|B\|_F^2$$

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar
 NPTEL ONLINE
 CERTIFICATION COURSES

So, then we have been talking about AB being approximately equal to C R right. What we really want is that one way of measuring that is Frobenius norm. There are other ways of measuring it like the spectral norm of the of A B minus C R, but for, now let us concentrate on the Frobenius norm, and we want to bound A B minus C R Frobenius norm ok. So, again I mean A B minus C R Frobenius norm squared the expectation of that is the same as writing the expectation of A B minus I replaced C by A I replace R by S transpose B Frobenius norm square right.

So, again I mean coming back from here, because Frobenius norm Frobenius norm of a matrix is really the summation of x_{ij}^2 square right. So, therefore, the expectation of this is really the expectation of this and, the each x_{ij} , because x_{ij} is a random variable, and the expectation of the I mean in this case x equal to A B minus C R in this case the expectation of I mean x_{ij}^2 is really just the variance of C R C R x_{ij} the variance of the of the ij th entry of C R, which is what we have calculated right. And just plugging that that back in if once you finish the calculation what you can say is that expected value of A B minus C R Frobenius square is at most $\frac{1}{c}$ by C Frobenius square of a times Frobenius square of B. And we have a bound on the expectation of the expected Frobenian the expected error.

(Refer Slide Time: 28:27)

Error in Frobenius norm

We want to bound $\|AB - CR\|_F$

$$E\|AB - CR\|_F \leq \sqrt{E\|AB - CR\|_F^2}$$

$$E\|AB - CR\|_F^2 = \sum \text{var}(CR)_{ij}$$

Using the discussed values for p_i allows us to bound

$$E\|AB - CR\|_F^2 \leq \frac{1}{c} \|A\|_F^2 \|B\|_F^2$$

We can now use Markov's inequality to bound in probability
 Can also improve bound using Chernoff style inequalities

Now, now we can do multiple, now what we can do is that we can use the I mean we know that this is nonnegative (Refer Time: 28:36) variable. Therefore, we can use Markov's inequality to say that the expectation is small. And if the expectation is small, the actual random variable is not going to be very much larger than its expectation with at least some probability with constant probability. Let us, say it is not going to cross 10 times expectation with at most 1 I mean with probability more than one-tenth ok. We can also prove better bounds using Chernoff style inequalities right.

In fact, what we can also say is that, expected value of a we can take away the square this square is not that important Frobenius is less than equal to using Jensen's inequality expectation of $\|AB - CR\|_F^2$ is less than equal to using Jensen's inequality. This is Jensen and then, you plug in this quality right, the quantity that you have been here ok.

(Refer Slide Time: 29:23)


Special case $B = A^t$


Sampling probabilities are $p_i = \frac{|A_{*i}|^2}{\sum_j |A_{*j}|^2} = \frac{|A_{*i}|^2}{|A|_F^2}$


Bound becomes

$$\|AA^t - CC^t\|_F \leq \frac{1}{\sqrt{c}} |A|_F^2$$

Can get improved bound to spectral norm in this case




IIT Gandhinagar
Indian Institute of
Technology Gandhinagar


 NPTEL ONLINE
CERTIFICATION COURSES

So, a special I will just before ending I just mentioned a special case that going to be useful for us. So, one special case is, when B equal to A transpose, in this case the sampling probabilities turned out to be very simple, it is just an it is just the squared norm of the of the ith column right. And the denominator turns out to be just the Frobenius norm square of A right.

And the and the bound becomes A transpose minus C C transpose Frobenius is less than 1 by square root C Frobenius norm of A square right, because the Frobenius norm of A and Frobenius norm of B are the same right. And in fact, it turns out that in this case this Frobenius instead of the Frobenius norm bound, we can actually get a spectral bound right, which is which is a better bound.

(Refer Slide Time: 30:02)

Using a dense S

Instead of sampling matrix S , can also use a JL matrix \rightarrow data oblivious

E.g. $S \in \mathbb{R}^{n \times c}$ $S_{ij} = N(0,1)$ or ± 1

S could also be FJLT

Easier to get bound on $\|AA^t - CC^t\|_F = \|AA^t - (AS)(S^tA^t)\|_F$



Another side variant that, we will see is that that, we were talking about this the sampling matrix S right. We were designing this sampling matrix using the I mean using the we will designed this matrix S using this sampling idea. We could also design this matrix S using a sketching idea right. In the sense that, we I mean instead of saying that S is this represents a sampling notion that every column has only one only nonnegative entry. We could say that S represents $A_j A_j^t$ matrix, which is data oblivious actually.

Now, right, and it could be either the dense JL, it could be the a it could have $N(0,1)$ and so on and so on and so. And in that case also I mean it is easy to get a bound on a transpose minus CC^t transpose Frobenius ok.

(Refer Slide Time: 30:47)

Running time

Using a sampling matrix: $O(mn + np) + O(mcp) + nc$

Using FJLT: $O((mn + np + c) \log n) + O(mcp)$

(assume $n > m, p$)

$O(mnp)$



So, let us just look at the running time using a sampling matrix right. Being a I mean doing the sampling takes time $m n$ plus $n p$ right, because we have to do a linear scan for each of the for each of this for each of the and time C actually. Well, $m n$ plus I mean just to calculate the probabilities you have to do a linear scan over the two matrices, then we have to I mean to choose C samples. And then, so maybe that takes time $n c$ and plus $m c$ p is the time for doing the smaller matrix multiplications. And you get a similar boundary using the FJLT, please think about it yourself. You should note that this is much less than the order $m n p$ that we were saying initially right which is for line matrix multiplication.

(Refer Slide Time: 31:35)

Summary

- Randomization and approximation a powerful tool in numerical linear algebra
- Saw two applications
 - Approximating PCA using random projections
 - Approximating matrix multiplication (basis of many results)
- Interchangeable role of sampling and sketching

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES 21

So, just to summarize what we will see in the next couple of lectures is that including this is a randomization approximation is a very powerful tool in numerical linear algebra. We have seen two applications of this already. If you remember, we have seen the this question of approximating the principal component analysis using random projections. We have also in this lecture, we also saw approximating matrix multiplication, and this will be the basis of many results. You also pointed out the interchangeable role of sampling and sketching.

(Refer Slide Time: 32:04)

References:

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

So, the reference for this is this lecture notes by Michael Mahoney and Petros Drineas on that is an archive freely available, please look at this.

Thank you.