

**Applied Linear Algebra**  
**Prof. Andrew Thangaraj**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**

**Week 08**

**Linear equations, Least squares solutions and Linear regression**

Hello and welcome to this lecture. This and the next lecture we will see a couple of applications of projections and this distance minimization which is very popular in engineering, some areas. And they've sort of taken a life of their own. There's the terminology associated with those ideas and it's a, it's a pretty big area by itself. But at the root of it, at the base of it are simple projections onto an orthonormal basis using an orthonormal basis onto a subspace, okay? So I want to show both those examples and just bring out how linear algebra connects all of those together, okay? So the first one revolves around solving linear equations, least squares solutions specific, slightly variant, I mean different from actually solving the equation. When you cannot solve, this is sort of the best you can do, the least square solutions. And how there is this idea of linear regression which is very popular in machine learning today. And we will talk about how that has roots on the least square solutions idea, okay? So let us get started.

(Refer Slide Time: 01:38)

Linear equations, Least squares solutions and Linear regression

**Recap**

- Vector space  $V$  over a scalar field  $F$ 
  - $F$ : real field  $\mathbb{R}$  or complex field  $\mathbb{C}$  in this course
- $m \times n$  matrix  $A$  represents a linear map  $T : F^n \rightarrow F^m$ 
  - $\dim \text{null } T + \dim \text{range } T = \dim V$
  - Solution to  $Ax = b$  (if it exists):  $u + \text{null}(A)$
- Four fundamental subspaces of a matrix
  - Column space, row space, null space, left null space
- Eigenvalue  $\lambda$  and Eigenvector  $v: Tv = \lambda v$ 
  - Some linear maps are diagonalizable
- Inner products, norms, orthogonality and orthonormal basis
  - Upper triangular matrix for a linear map over an orthonormal basis
  - $V = U \oplus U^\perp$  for any subspace  $U$
  - Orthogonal projection gives closest vector in the subspace

1:38 / 26:21

Okay. Quick recap. We have been talking about, you know, operators of late and particular operators in inner product spaces. And when you have an inner product and a norm, there is this

notion of orthogonality. And subspaces have orthonormal bases. And you can project any vector onto that subspace using the orthonormal basis and you get, you know, the closest vector in the subspace. And so that anytime you solve an optimization like that in an elegant way, it has a lot of applications, right? So and then there are applications that come out of it and let's see one of those primary applications which is this notion of regression or least square solutions to a linear equation. Okay. So let's remind ourselves what linear equations are. You have, so we will predominantly stick with real valued, you know, real field. Vector spaces over real field. And the applications. But, you know, many of the ideas can be extended quite easily. But let us begin with linear equations. A matrix is given to you.  $m \times n$ . And a vector,  $m \times 1$  vector, column vector is given to you. And you have to find an  $x$  such that  $Ax = b$ . So this is the standard linear equation. And you know that there is a solution if  $b$  is in the range of  $A$ . Range is the column space of  $A$ .  $Ax$  is simply an element in the range. So if  $b$  is not in the range, there is no chance that you will have a solution. Solution exists only if  $b$  is in the range of  $A$ , okay?

(Refer Slide Time: 03:38)

Linear equations, Least squares solutions and Linear regression

**Linear equations**

$A: m \times n$  matrix and  $b: m \times 1$  vector

System of linear equations: Find  $n \times 1$  vector  $x$  s.t.

$$Ax = b$$

Solution exists, if  $b \in \text{range } A$

When solution exists

$$\|Ax - b\| = 0$$

So, solution to  $Ax = b$  is  $\arg \min_x \|Ax - b\|^2$

What if  $b \notin \text{range } A$  and there is no solution to  $Ax = b$ ?

3:37 / 26:21

So you notice one interesting observation particularly when you have inner products and norms and all that. When the solution exists, the  $\|Ax - b\|$  is equal to zero, right? It is exactly equal to zero. So when, so maybe you can find, if there are infinitely many solutions there may be many  $x$ 's, but still  $Ax$  is always  $b$ , right? So  $Ax$  equals  $b$  and  $Ax - b$  has norm zero, okay? So in a way, when solution exists, the solution to  $Ax = b$  is actually the minimizer for the norm, right? So, I mean, it's just when you have a solution, it all seems very redundant to think of it this way. But you can think of the solution in that fashion, okay? So what is the solution to  $Ax = b$ ? It's that  $x$

which minimizes  $\|Ax - b\|$ , okay? When solution exists... Of course norm is non-negative. So norm goes to zero, right? So that's the least possible value it can take. So that seems to fit. So the idea is when there is no solution, what do you do, okay? When  $b$  is not in the range of  $A$ , you can extend this minimization definition and get what's popularly called the least squares solution to linear equations, okay?

(Refer Slide Time: 04:57)

Linear equations, Least squares solutions and Linear regression

NPTEL

## Least squares solution to linear equations

Least squares solution to  $Ax = b$

$$\arg \min_x \|Ax - b\|^2$$

Properties

- If  $b \in \text{range } A$ , least squares solution coincides with usual solution
- If  $b \notin \text{range } A$ , the solution is still well-defined. Why?

4:57 / 26:21

So given a linear equation  $Ax = b$ , the least squares solution is argument of the minimizer of, minimization of  $Ax = b$  over  $x$ , okay? That  $x$  which minimizes  $\|Ax - b\|$  is your least squares solution, okay? So now the advantage of this is: you don't have to bother whether  $b$  is in the range of  $A$  or not, okay? If  $b$  is in the range of  $A$ , yeah, well and good. I will give you the best solution for the minimizer. If  $b$  is not in the range of  $A$ , I can still get something with the least squares solution, right? So I have hope of getting something, okay? So that's the nice thing about this least squares solution. It not only gives you a linear equation, so given that you have an inner product also in this space, it uses that to find the closest vector in the column space of  $A$ , okay? So that's what. So your  $b$  is not in the column space, so you can't find an exact  $x$ , so you find an  $x$  which takes you closest to the  $b$ , right? So that sort of makes sense. So that's least squares solution, okay?

What're the properties? Some very quick properties to see. If  $b$  is in the range of  $A$ , least squares solution coincides with the usual solution. That's easy to see. If  $b$  is not in the range of  $A$ , the solution to this problem is well defined, okay? Why is that? Because it is very easily connected to orthogonal projection, okay? So you might say is the  $\min_x \|Ax - b\|^2$ , is that well defined? You

may want to use some calculus etc. to prove it. But we can use this notion of projection and elegantly conclude that the solution will always exist, okay? So why is that? Supposing you have an  $m \times n$  matrix  $A$  and  $b$  is an  $m \times 1$  vector.  $Ax = b$  the equation is given to you. You can define a subspace  $U$  as the range of  $A$ , okay? And then clearly you see that  $Ax$  belongs to range of  $A$ , okay? And  $b$  is some vector. So the  $\arg \min_x \|Ax - b\|^2$  is the same problem as finding the closest vector in the subspace  $U$ , okay? As you vary  $x$ ,  $Ax$  goes over all possible vectors in the range of  $A$ , so you can redefine that as argument of  $\min_u \|u - b\|^2$ . So all you are doing is finding the vector closest to  $b$  in the subspace  $U$ . And we are in an inner product space, so you know that's all well defined. And you will get the closest vector in the subspace as the least squares solution, okay?

(Refer Slide Time: 06:57)

Linear equations, Least squares solutions and Linear regression  
NPTEL

### Least squares solution and orthogonal projection

$A: m \times n$  matrix,  $b: m \times 1$  vector

Let  $U = \text{range } A$ . Then,  $Ax \in \text{range } A$  and  $b$ : vector

$$\arg \min_x \|Ax - b\|^2 \leftrightarrow \arg \min_{u \in U} \|u - b\|^2$$

Least squares solution is essentially the same as closest vector in a subspace!

Let  $P_U$ : orthogonal projection operator onto  $U$

Least squares solution to  $Ax = b$  is the same as solving

$$Ax = P_U b$$

6:57 / 26:21

So notice what is going on. So when  $b$  is in the range of  $A$ , you may have many solutions for  $x$ . When  $b$  is not in the range of  $A$ , you simply look at the projection of  $b$  onto  $U$  and solve for  $Ax$  equals that projection, okay? So that is the idea here. So you take the projection operator onto  $U$ . You know how to do that. And then the least square solution to  $Ax = b$  is the same as solving  $Ax = P_U b$ , okay? Instead of solving for  $b$ , you solve for the closest point of  $b$  inside the subspace that you want, okay? So it's sort of a natural extension. But we know that this has a solution, right? So there is no problem with worrying about does it have a solution or not. This definitely has a solution, okay?  $P_U b$  is definitely in the range of  $A$ , right? It is a projection on to  $U$ , so it's in  $U$ , so you will have a solution, okay? You may have even infinitely many solutions for this, okay?

(Refer Slide Time: 09:06)

Linear equations, Least squares solutions and Linear regression

NPTEL

### Linear regression with IPL data

[Open in Colab](#)

```
In [1]: import numpy as np
```

```
In [3]: A=np.load('/content/drive/My Drive/Colab Notebooks/A.npy')
```

**IPL 1st innings score data**

A: 20 x 747 matrix

$A[i, j]$ : Runs scored in Over  $i$  in Match  $j$

The file A.npy above is [here](#). Copy it to your local disk or mounted Google drive and change the above PATH suitably.

```
In [7]: A10 = A[:10,:].transpose()
A15t = np.sum(A[:15,:],axis=0).transpose()
```

**Linear regression setup**

A10: 747 x 10 matrix

Row  $j$  of A10: runs scored in first 10 overs of Match  $j$

linearregressionipl.ipynb hosted with by GitHub

9:06 / 26:21

So this notion of a least squares solution is used in linear regression and other such applications inside machine learning and other areas. So what I have done to explain that is created a Colab sheet where I have taken some data in the spirit of machine learning from the real world. In fact given that we are in the IPL season currently, I have taken data from IPL first innings score. So you can go to some cricket websites, they'll give you this data. You might have to write some scripts to modify them etc. But I've created this matrix  $A$  here which is a  $20 \times 747$  matrix, okay? So you start seeing these kinds of applications, right? So a lot of linear algebra that we study: orthogonal projection, least square solutions have applications in the real world. And when you go there, the sizes of these matrices are big and you need these kinds of tools to deal with them. It's difficult to, you know, just write them down like  $20 \times 747$ . Why would I write that? In fact, I looked at the data. There's been about 760 odd IPL matches so far. Some of them have no result. I didn't take the no result matches. I don't know why, I just wasn't sure about it, so I dropped all of those. I only took the matches which had a result. And that's 747 so far. And what is this 20? What is this  $20 \times 747$ ? I'm going to take the number of runs scored in each over, okay? So each column of  $A$  is one match, okay? And the  $i^{\text{th}}$  entry in the column is the number of runs scored in that over, okay? There are 20 overs in an IPL match in the first innings. So you look at the  $i^{\text{th}}$  entry in a particular column, that would be the number of runs scored in the  $i^{\text{th}}$  over, okay? So this file A.npy is a numpy saved file. I have given a link for you and there you can go download that file and put it in your local disk or some Google drive or something and access it from your own version of Colab. So that'll, you might have to change the link there. So that will give you the data, okay?

(Refer Slide Time: 10:45)

Linear equations, Least squares solutions and Linear regression

## Linear regression with IPL data

The file A.npy above is [here](#). Copy it to your local disk or mounted Google drive and change the above PATH suitably.

```
In [7]: A10 = A[:10,:].transpose()
A15t = np.sum(A[:15,:],axis=0).transpose()
```

### Linear regression setup

A10: 747 x 10 matrix  
Row  $i$  of A10: runs scored in first 10 overs of Match  $i$   
A15t: 747 x 1 matrix  
A15t[ $i$ ]: Total score at the end of 15 overs

### Linear model

We hypothesize that the total score  $y_i$  at the end of 15 overs in Match  $i$  is a linear function of the vector  $x_i = [x_{i1} x_{i2} \dots x_{i10}]$ , where  $x_{ij}$  is the runs scored in the  $j$ -th over in Match  $i$ . In other words,

$$y_i = x_i a^T,$$

where  $a = [a_1 a_2 \dots a_{10}]$  specifies the linear function.

linearregressionipl.ipynb hosted with by GitHub

So once you have this matrix  $A$  with you, you have all the data, right? And now you can start thinking of sending up some linear regression type problem. And I am going to show you one example. It actually may be a very bad example, I am just giving you one flavour. There are so many other problems where you can use the similar idea, but I'm just picking up some very simple thing, okay? So what am I going to do? So I've written a couple of commands here. I'll come to this later. But let's look at this linear regression setup as I have explained here in the text, okay? So I'm going to first define A10 which is a  $747 \times 10$  matrix. I'm first transposing that. I had  $20 \times 747$ , I'm making it  $747 \times 20$ . It's just a transpose. It's easier to deal with. Row  $i$  of A10 is the run scored in the first 10 overs of match  $i$ , okay? So hopefully that is clear to you. So every row is a match now, because I have transposed it, every row is a match and I'm keeping only the first 10 overs, okay? So why am I keeping only the first 10 overs? I'll tell you. That's what I want to do. So that's what I've done in the command here, right? So  $A[:10,:]$ . And then I've transposed it. So I'm keeping only the first 10 overs. I'll tell you why this learning problem is involving something like that. And then I have a number here which is A15t, okay? So I have a vector here of length 10. For every match, it is the run scored in each over for each of the first 10 overs. That's my vector. And then I also have A15t which is the total score of at the end of 15 overs of the  $i^{\text{th}}$  match, okay? So this will be a  $747 \times 1$  matrix and that is what I have done here. A15t I am summing up to the 15<sup>th</sup> row of the matrix A and then transposing, okay? So this axis 0 tells me that you do the column sum, okay? So this is A15t is the total score at the end of 15 overs, okay?

So maybe you can sort of smell where I am going here. I have the runs scored in the first 10 overs. And I'm also keeping the total score at the end of 15 overs, okay? So now we think of linear models. So what is this linear model that I'm thinking of? I'm going to hypothesize that the total score  $y_i$  at the end of 15 overs in match  $i$ , okay, is a linear function of the vector  $x_i$  which is, you know, the run scored in each of the first 10 overs. Is that okay? So I take this number of runs scored in each of the first 10 overs and then I want to project or I want to guess or predict what will be the total score at the end of 15 overs, okay? So I don't know what's going to happen in the next five overs, but still can I, using this data, come up with some estimate for what will be the total score at the end of 15 overs, okay? So I'm going to hypothesize that that will be a linear function, okay? So why linear function? Because, you know, that's the easiest I can deal with. So let's start with the linear function. So once I hypothesize it's a linear function, I only need these coefficients, right?  $[a_1 \dots a_{10}]$ . And I will multiply  $x_i$  by  $a^T$ , okay?  $a$  I am thinking of as a row vector. So if you do  $a^T \dots x$  is a row vector,  $a^T$  is a column vector. I am simply doing a linear combination to give  $y_i$ .

So my hypothesis is a linear model, right? So  $y_i$  is a linear combination. I do not know what the  $a$ 's are, okay? So in any machine learning problem typically you try to find these  $a$ 's, okay? I have to now use the data that I have, this 747 data that I have collected and then I try to find the value of  $A$ , okay? So what is the idea? So we will solve the linear equation, okay? So this gives you equations, right? We will do that with about half of the data, okay? I do not want to use all the data. So this is very typical in machine learning. This is how you work. You take half the data. 747 maybe half is some 400 or something, okay? I'm just taking 400 of the data points and I'm going to solve the equation that I get using the least square solution because I know least squares solution will always be there. I mean, I may not get exact solution for these linear equations, the model may not fit. But, you know, I can always do a least square solution. So I will get some answer, okay? How do I know if that answer is any good? I can test the solution on the remaining data, right? So I have, I take half the data, I compute the  $a$  and then keep that same  $a$  and actually compute the projected thing for the remaining things and see what my error is. Am I doing good on the remaining data. So this is a method that machine learning normally uses. You have taken all the data, split into two, you know, you compute the model parameters using the what's called the training set. The first training data. And then use that and test it on the testing set which is what you have kept away. You use that in the, use that model, evaluate your prediction on that model and then test it out, okay?

So that is what I have coded here. You can see what I have done. I have taken `A_train` as the `A10[:400, :]`. So only the first 400 matches I am keeping. And `y_train` is the actual output. So I will take the actual output for the first 400 and then numpy has this implementation called `lstsq` which is least square solution, okay? So if I give you `A_train` and `y_train`, it will give you the solution for  $a$ , okay? Well it will give you, it will give you many more things, so I am just keeping `a[0]`. `a[0]` is the solution and you can see the answer. And I think if you look at it most of you probably are not going to be surprised that each of these coefficients are around 1.5, right? I mean



if, without any further, you know, looking very closely at the data, if you had to guess from 10 overs what will be the score in 15, you're going to multiply by 1.5, right? You take the total and then multiply by 1.5. So that's the machine learning algorithm sort of learning that. But, you know, there are some coefficients which are a bit off, you know? I mean this 1.7 in the last one, there's some 1.4 in the third one, 1.7 odd in the fourth one. I mean, sort of around 1.5. But little bit off here there, I don't know, I mean this is what this data is telling you. God knows what happened in the first 400 matches. You can think of so many other factors here we're ignoring when we find just this. But this is okay, you know? You got some coefficients.

(Refer Slide Time: 15:33)

Linear equations, Least squares solutions and Linear regression

NPTEL

### Linear regression with IPL data

A15[i]: Total score at the end of 15 overs

**Linear model**

We hypothesize that the total score  $y_i$  at the end of 15 overs in Match  $i$  is a linear function of the vector  $x_i = [x_{i1} x_{i2} \dots x_{i10}]$ , where  $x_{ij}$  is the runs scored in the  $i$ -th over in Match  $i$ . In other words,

$$y_i = x_i a^T,$$

where  $a = [a_1 a_2 \dots a_{10}]$  specifies the linear function.

**How to find  $a$ ? Is the linear model any good?**

Solve the linear equation to find a least squares solution with about half of the data. Test the solution on the remaining data.

```
In [12]: A_train = A10[:400,:]
         y_train = A15[:400]

         a = np.linalg.lstsq(A_train, y_train, rcond=None)

In [14]: print(a[0])

[ 1.45144125  1.52879231  1.38912004  1.67933154  1.47458773  1.48360096
  1.56418163  1.52481581  1.52770619  1.70746336]
```

linearregressionipl.ipynb hosted with ❤️ by GitHub

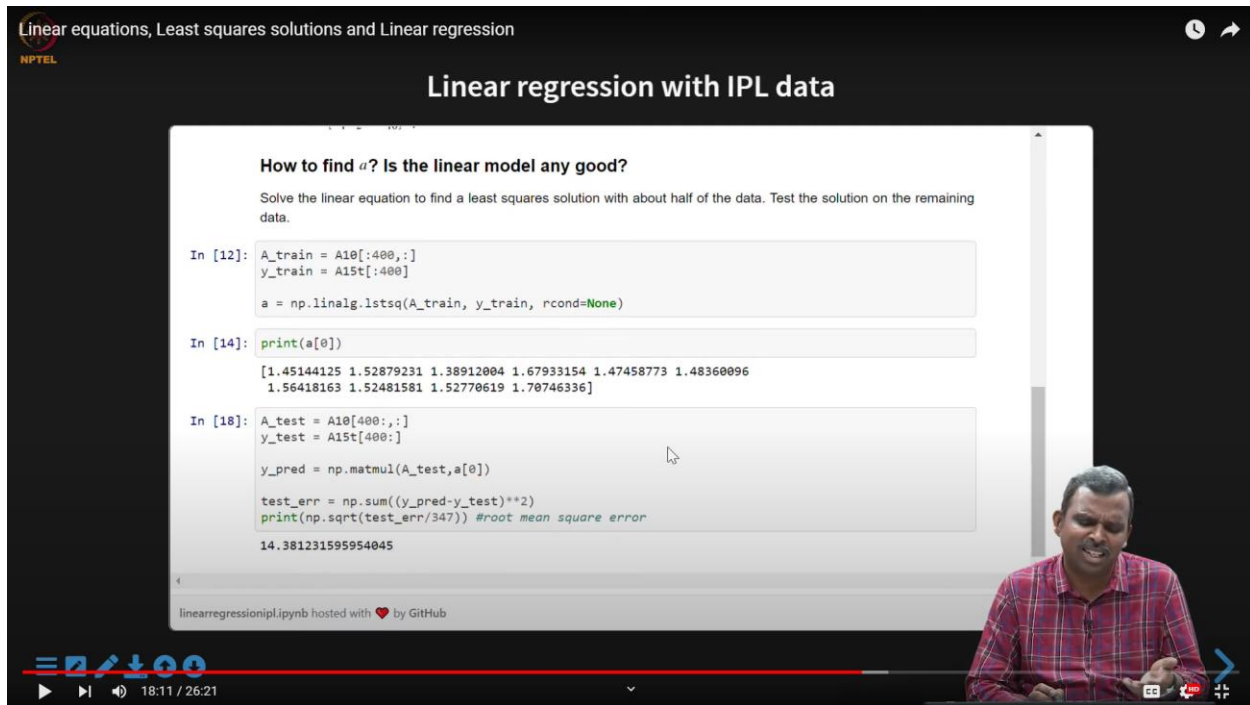
15:33 / 26:21

Now how good is your prediction? You have to test, right? So you take your  $A_{\text{test}}$  and then  $y_{\text{test}}$  and then you find your prediction. What's your prediction? You do the multiplication of  $A_{\text{test}}$  by  $a[0]$ , right? So that's your prediction. Your  $y_{\text{test}}$  is the actual score and you compute your squared error, okay?  $y_{\text{pred}}$  minus  $y_{\text{test}}$  squared. The  $**2$  just squares and  $\text{np.sum}$  sums. And then you can take root mean square, right? So you can divide the error by 347 which is the total number of test data. And then take the square root. So the answer comes out to be something like 14, okay? So this learning method at least for the data that I've given you seems to be off by 14. 14 runs in the IPL thing. So you may want to test it with other type of things. I mean, of course, the least squares with any linear model, this will be the best. You can't beat it with the linear model, right? So if you, for instance, if you just multiply the total score after 10 overs by 1.5, that will definitely be poorer than this, right? So because it can be, this is optimized with this kind of test data, okay? Now hopefully this showed you what is going on here, how these equations are formed and why



this linear regression sort of relies on least squares in a very interesting way. So this is the idea, okay? So you have data that you collect and then you have, you want to project or predict something. You assume a linear model and every data gives you, you know, one column so to speak or one row in your matrix then you keep building it up. When you have enough data for testing, enough data for training, on the training, you find the coefficient. On the testing you test how well you've done. So this is pretty good.

(Refer Slide Time: 18:11)



Linear equations, Least squares solutions and Linear regression

### Linear regression with IPL data

**How to find  $a$ ? Is the linear model any good?**

Solve the linear equation to find a least squares solution with about half of the data. Test the solution on the remaining data.

```
In [12]: A_train = A10[:400,:]
y_train = A15[:400]

a = np.linalg.lstsq(A_train, y_train, rcond=None)

In [14]: print(a[0])

[1.45144125 1.52879231 1.38912004 1.67933154 1.47458773 1.48360096
 1.56418163 1.52481581 1.52770619 1.70746336]

In [18]: A_test = A10[400:,:]
y_test = A15[400:]

y_pred = np.matmul(A_test, a[0])

test_err = np.sum((y_pred - y_test)**2)
print(np.sqrt(test_err/347)) #root mean square error

14.381231595954045
```

linearregressionipl.ipynb hosted with ❤️ by GitHub

18:11 / 26:21

So for instance one, I mean people who know cricket very well will tell you I am ignoring some very important things when I look at my  $x$ , right? So maybe I have to add one more vector, one more data point to  $x$  which could be maybe the number of wickets that have fallen up to the 10<sup>th</sup> over, isn't it? That could be a useful thing, you know? I mean, so maybe that number has to be added to the  $x$ , that data, if you can pick up and then maybe then you will improve this model, right? So even with the linear model with that added data, maybe this 14 can come down further, right? So this is the sort of thing that people do in machine learning, okay? So they think of what else to do, how else to do. Maybe linear is not a good function. If linear is not a good function, you go out of the realm of linear algebra. So that's not a course for us. But here is an example of this, okay? So this Colab workbook and all of this is shared with you. You can click on this I've embedded here. This file is with you. You can play around with this and see what I mean for sure and understand this. So this is very typical of how, this kind of idea is used in practice in machine learning applications. So hopefully that was interesting. So let's move on and try to look at this notion of orthogonal complement and projection, okay?

So we've been studying orthogonal complement and we know that they are connected, right? So it is quite easy to see. If you have a subspace and if you have a projection operator and  $u$  becomes the projection of a vector  $v$ , then this  $(u - v)$  actually belongs to the complement, right? So it is a very standard thing. So  $v$  minus what you projected onto is orthogonal to the entire subspace, okay? So that is a very standard result in this projection. So now you notice this slight twist on this, okay?  $u$  is the closest vector to  $v$  if and only if  $(u - v)$  belongs to  $U^\perp$ , okay? So this is also a statement. I didn't quite put it like this maybe when we studied orthogonal projection properly, but you can see that this is true, right? Out of all the vectors that are closest to  $v$ , right, I mean of all the vectors in  $U$ , the vector that is closest to  $v$  is that one which will satisfy this.  $(u - v)$  has to be in the orthogonal complement, okay? Only then this will work out. The same orthogonality condition. So notice how this will subtly transform itself into something else, okay? So now how do you do this testing? How do you find if  $(u - v)$  is in  $U^\perp$  or not? It's not very hard, right? So you take any spanning set for  $U$ . I'm going to say spanning set, it could be a basis, but maybe not, I mean I don't care. Any spanning set for  $U$ . Then  $(u - v)$  is in  $U^\perp$  if and only if this inner product is 0, isn't it? So each of these spanning vectors inner product with  $(u - v)$  has to be zero. If that is true, then clearly you know this is in this  $U^\perp$ . If this is in  $U^\perp$ , this will also be zero. So this is an if and only if, okay?

(Refer Slide Time: 21:16)

Linear equations, Least squares solutions and Linear regression  
NPTEL

### Orthogonal complement and projection

$U$ : subspace of  $V$ ,  $P_U$ : orthogonal projection

For  $v \in V$ , suppose  $u = P_U v$ . Then,

$$u - v \in U^\perp$$

$u \in U$  is the closest vector to  $v$  iff  $u - v \in U^\perp$

Spanning set for  $U$ :  $\{u_1, \dots, u_m\}$

Then,  $u - v \in U^\perp$  iff  $\langle u_i, u - v \rangle = 0$

In other words,  $u = P_U v$  is the unique solution to

$$\langle u_i, u \rangle = \langle u_i, v \rangle \text{ for } i = 1, \dots, m$$

21:16 / 26:21

Notice where we are going here. You will see this. We are going into an interesting direction as far as orthogonal complement is concerned. We had one way of doing the orthogonal complement. I will give you another way of thinking about the orthogonal complement. In other words, this  $u$

which was the orthogonal projection of  $v$  onto the subspace  $U$  is also the unique solution to these set of equations. You take any spanning set for  $U$ , it is the unique solution to  $\langle u_i, u \rangle$  being equal to  $\langle u_i, v \rangle$ , this inner product being zero for all of the vectors in the spanning set, okay? So this is another way of doing the projection. Instead of doing the projection, you know, you find the orthonormal basis and take the inner product and do that, you know? That I think is quite an easy process. You can also do something like this, but this is also the same, it's not anything different, right? So you go to an equation of this form, okay? So hopefully this was clear. It is just, we are just using the orthogonality in a very interesting way with a spanning set for  $U$ , okay? Because we know that you know  $(u - v)$  is in the complement, so you take a spanning set for  $U$ , then you get a set of equations and these are linear, you know? I mean this is all inner product so it's all linear. So it's very nice to write down, okay?

(Refer Slide Time: 24:24)

Linear equations, Least squares solutions and Linear regression

### Solving linear regression

$A: m \times n$  real matrix,  $b: m \times 1$  real vector

Let  $U = \text{range } A$ . Let  $a_i$  be the  $i$ -th column of  $A$ .

Spanning set for  $U: \{a_1, \dots, a_n\}$

Closest  $Ax$  to  $b$  satisfies  $Ax - b \in U^\perp$

or

$$\langle a_i, Ax - b \rangle = 0$$

In other words,

$$A^T(Ax - b) = 0$$

or

$$A^T Ax = A^T b$$

The diagram shows a matrix  $A^T$  with rows  $a_1, a_2, \dots, a_n$  and an equation  $(Ax - b) = 0$ .

So now when you want to solve linear regression, there is an equation that you can write using this. This, the previous property that we studied, okay? Take a spanning set and do that, okay? So you have a  $m \times n$ , in the linear regression setting you have  $m \times n$  matrix  $A$ . I am picking it as real for this purpose but you can even, extending to complex is also very easy.  $b$  could be a real vector,  $b$  is a real vector. I am going to define  $U$  as the range of  $A$ , okay? It's the range of  $A$ . This is the subspace  $U$  which we have been trying to project on to, right? So let us say  $a_i$  is the  $i^{\text{th}}$  column of  $A$ . So notice, why am I worried about the columns of  $A$ ? I am looking at range of  $A$ , so because this will give me a spanning set for  $U$ , okay? What is the spanning set for  $U$ ?  $a_1$  to  $a_n$ , the columns of  $A$ , okay? So notice how I will move into this inner product thing. So the closest  $Ax$  to  $b$ , okay,

the closest  $Ax$  to  $b$ . Remember  $Ax$  is an element of  $U$ , right? Closest  $Ax$  to  $b$  satisfies  $(Ax - b)$  is in  $U^\perp$ , right? This is a very simple little relationship that we have seen before, okay? Or, what did we see about how to check for  $(Ax - b)$  belonging to  $U^\perp$ ? The inner product of each of these columns of  $A$  with  $(Ax - b)$  has to be 0, okay? These are all equivalent conditions. We know that all this has solutions, right? This projection exists, we know they can project. So all of these have solutions, there's no need to check anymore. So this inner product of  $a_i$  with  $(Ax - b)$  has to be equal to 0 for every  $i$ . This is the same condition as projecting and solving, right? So there's no difference here. So in other words you write this for every  $i$ , which means every column inner product with  $(Ax - b)$  goes to zero. But how do I write every column? So I can simply write it as  $A^T$  multiplying  $(Ax - b)$ . Notice what happens if I do  $A^T$ . If you want I can write out what  $A^T$  will be.  $A^T$  will simply be  $a_1$  here,  $a_2$  here, so on till  $a_n$  here, isn't it? Right? Now I want to take inner product with  $(Ax - b)$ . What is  $a_1$  inner product with  $(Ax - b)$ ? So now I can write  $(Ax - b)$  here, right? This will be just a column vector. So this into this will give you every value being inner product and that has to be zero throughout, okay? So that's the same condition here.  $A^T(Ax - b) = 0$ . You see I keep writing zero on this side. Remember zero is, you know, you put as many zeros as you need on the column, okay? So that's how you think about it. So this other thing, the same thing now you know. You just multiply it out, you get this very interesting looking equation  $A^T Ax = A^T b$ , okay?

(Refer Slide Time: 26:21)

Linear equations, Least squares solutions and Linear regression

### Solving linear regression

$A$ :  $m \times n$  real matrix,  $b$ :  $m \times 1$  real vector

Let  $U = \text{range } A$ . Let  $a_i$  be the  $i$ -th column of  $A$ .

Spanning set for  $U$ :  $\{a_1, \dots, a_n\}$

Closest  $Ax$  to  $b$  satisfies  $Ax - b \in U^\perp$

or

$$\langle a_i, Ax - b \rangle = 0$$

In other words,

$$A^T(Ax - b) = 0$$

or

$$A^T Ax = A^T b$$

Solution always exists for above equation!

Why? By orthogonal projection!

*Handwritten notes:*  $\min_x \|Ax - b\|^2$ , calculus, diff. w.r.t.  $x_i$ ,  $x = \text{equal to zero}$

25:51 / 26:21

And how do I know that this equation always has a solution? Of course it has to, right? It is the same as the orthogonal projection. Orthogonal projection exists, so definitely this equation will

have solutions, okay? So a nice little, nice looking equation which you can directly look at from these. So a solution always exists. There are various ways to show that a solution always exists even if you do not like the inner product approach. We will do that later on for the equations like this. But when you have inner products and when you have projections, a solution will always exist for this by the orthogonal projection that we are looking at, okay? So this is another way of thinking of linear regression. You'll see a lot of... So in fact if you start with, you know, so if you start with  $\|Ax - b\|^2$  and minimizing over  $x$  and if you do calculus... What is calculus? You differentiate with respect to  $x$ ,  $x_i$  and equate to zero, you will get the same equation, okay? So that is another way in which people come to this  $A^T Ax = A^T b$ . We came to it using our orthogonal projection but all of them are the same, okay? So there can be very very many ways to solve the problem, but at the end of the day they solve the same problem in one way or the other, okay? So thank you very much for listening up to this point. We will look at another application in the next lecture which is quite interesting in its own right but it is very different from this application, okay? See you then.