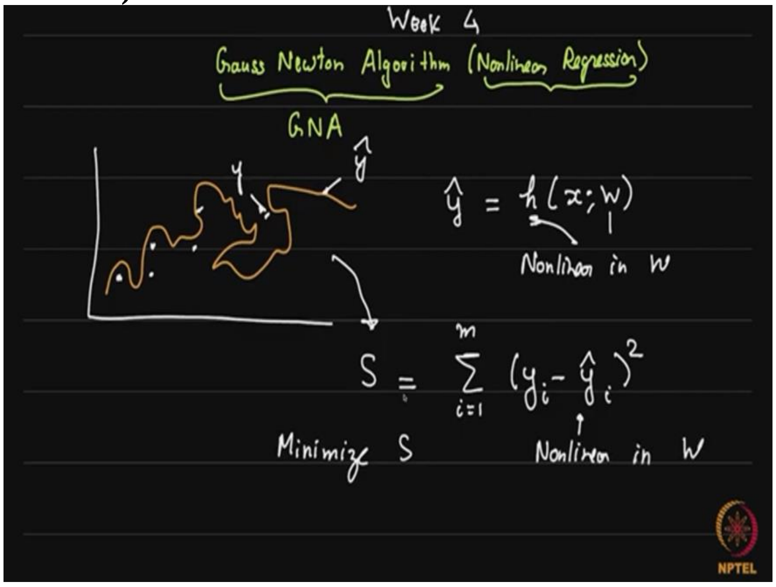
Inverse Methods in Heat Transfer

Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology - Madras

Lecture - 26
Gauss Newton Algorithm Derivation and Code

(Refer Slide Time: 00:19)


Week 4
Gauss Newton Algorithm (Nonlinear Regression)
GNA
y
$\hat{y}$
$\hat{y} = h(x;w)$
Nonlinear in W
$S = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$
Minimize S
Nonlinear in W

Welcome back, in this final main video for this week, in week 4, we are going to look at sort of the crowning jewel of what we have been discussing till now this is known as the gauss newton algorithm and use the short form GNA. The purpose of this is nonlinear regression. Remember what nonlinear regression does, it is we have a bunch of points. And we could have a multi parameter model, which is highly nonlinear in the parameters.

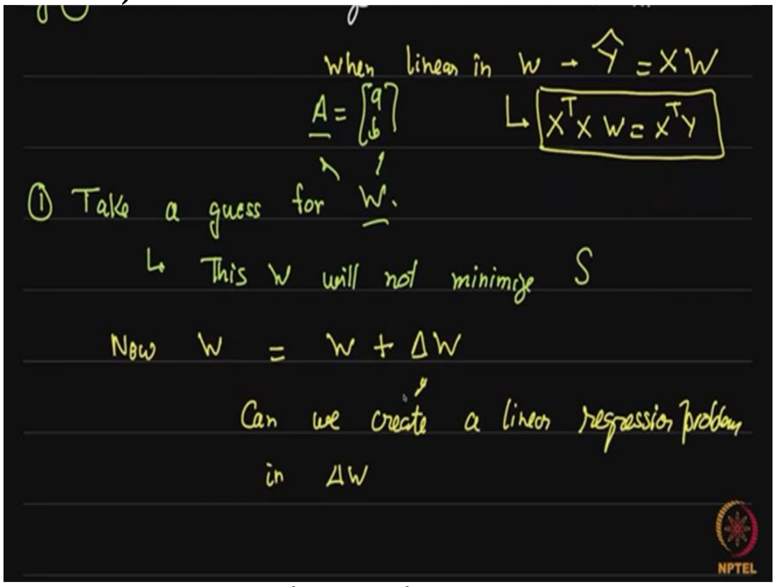
So, this kind of behaviour you will not get from linear models, and you want to fit it here. So, we basically have mathematically $\hat{y}$ equal to some function. So, some function let us call it $h(w)$, or we call it $h(x;w)$, and this h is nonlinear in w. We do not care what it is in x, but we care about what it is in w. So that is what we are trying to solve for.

Ultimately, however, having said that, we want to basically solve the least square problem now I am instead of calling it J, I will call it S, which we also use, which is,

$$S = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

If you wish, you can put a 1 over m here or not. I am not going to put it for now, i = 1 to m. This is what we want to minimize, we want to minimize S, that is what the regression problem boils down to $\hat{y}$ is given by this curve, y is given by these points, and we want to minimize the gap between the points and the curve. So, what is the issue? The issue is as usual, what we discussed before. The issue is this, now is nonlinear in w then it was linear in w.

(Refer Slide Time: 02:34)



When linear in W → $\hat{Y} = XW$

$\underline{A} = \begin{bmatrix} a \\ b \end{bmatrix}$ ↳ $\boxed{X^T X W = X^T Y}$

① Take a guess for $\underline{W}$.

↳ This W will not minimize S

New W = W + ΔW

Can we create a linear regression problem in ΔW

When linear in w, what is linear in w? $\hat{Y}$. So, if $\hat{Y}$ was if you remember, the matrix was,

$$\hat{Y} = XW$$

then we knew how to solve it, then the solution was we would write it as,

$$X^T XW = X^T Y$$

So, this was the solution. Now, the question is can we do the same trick now, in order to whatever we did in the last video, which is if we know the solution to the linear can we now change it into multiple steps of the nonlinear problem as a multiple step of linear problems.

So, we will try the same trick again. And we will see if we can derive the Gauss Newton algorithm now, so, the way we are going to do is the same which is take a guess. As usual, the first step is taking a guess for W, so, if you have a nonlinear regression problem, for example, we had the case where we had,

$$\hat{y} = a(1 - e^{-bx})$$

Then that means take a guess for a take a guess for b.

So, W in that case would be a, b, another set of people would call this a, so, I will use this notation interchangeably within this video I both use A as well as W depends on what people find more convenient people like to use A because it is more indicative of these variables can be. So, take a guess for A W. Now what will happen is this W will not minimize our S. So that is kind of obvious if you have taken a random guess for S.

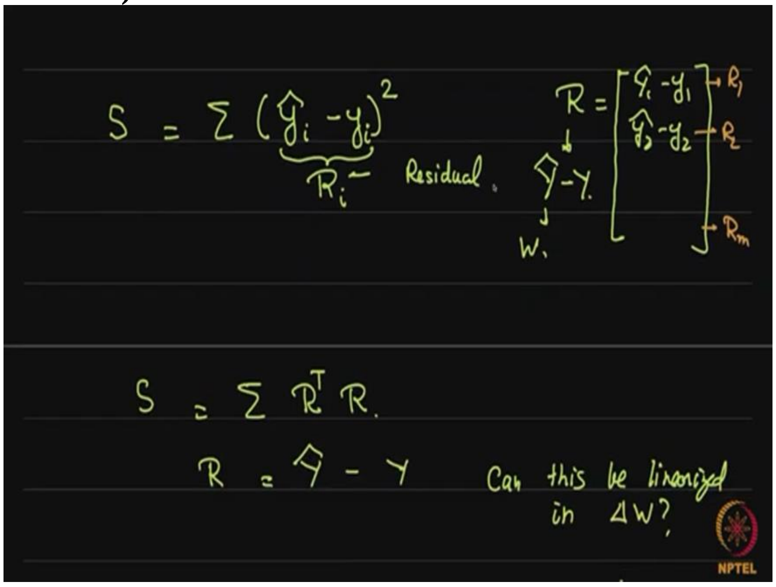
Now what we will do is now let us say that the new W is,

$$W = W + \Delta W$$

And what we want to do is can we create a linear regression problem in ΔW, we could not create a linear regression problem in W. But perhaps we could create a linear regression problem in ΔW, we will use the same trick that we did before. So here it is, this is the problem we wish to minimize, and see if we can create a linear regression problem there.

So let us try to do that. Please pay careful attention, this can get slightly confusing. That is my experience.

(Refer Slide Time: 05:38)



$$S = \sum \underbrace{(\hat{y}_i - y_i)}_{R_i\text{ - Residual.}}{}^2$$

$$R = \begin{bmatrix} \hat{y}_1 - y_1 \\ \hat{y}_2 - y_2 \\ \\ \end{bmatrix} \begin{matrix} \to R_1 \\ \to R_2 \\ \to R_m \end{matrix}$$

$R \downarrow \hat{Y} - Y. \downarrow W.$

$$S = \sum R^T R.$$

$$R = \hat{Y} - Y$$

Can this be linearized in ΔW?

So S is sigma, I am going to switch other signs because it is usually easier to work with,

$$S = \sum (\hat{y_i} - y_i)^2$$

Let us call this R_i, R stands for residual. You can call it small r_i also, which I did earlier, but I am going to call it capital R_i. capital R is the vector,

$$R = \begin{bmatrix} \widehat{y_1} - y_1 \\ \widehat{y_2} - y_2 \\ . \\ . \end{bmatrix}$$

Basically, R is the vector $\hat{Y} - Y$. All right, So, now we have R as a vector, which is a function of Y and $\hat{Y}$ Now, $\hat{Y}$ itself is a function of W.

So here is the trick so I am going to say S,

$$S = \sum R^T R$$

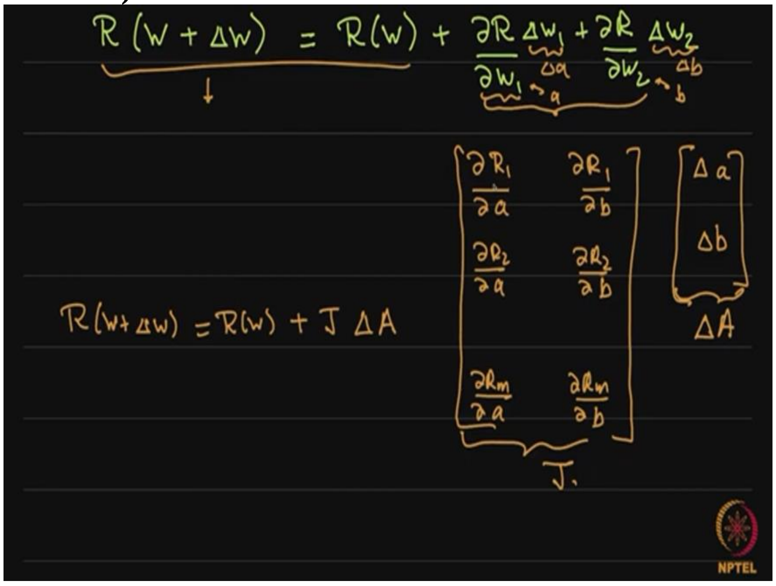
This R, which is $\hat{Y} - Y$, can this be linearized in ΔW? That is the question. Now we already have R at W that is our current residual. So, we can say,

$$R(W + \Delta W) = R(W) + \frac{\partial R}{\partial w_1}\Delta w_1 + \frac{\partial R}{\partial w_2}\Delta w_2$$

$R(W + \Delta W)$ is going to be $R(W)$ plus the same thing that we did last time. If you wish, you can write this as $\frac{\partial R}{\partial w_1}\Delta w_1$ + $\frac{\partial R}{\partial w_2}\Delta w_2$, you have to be a little bit careful.

Similarly, R will have multiple components. And we can use that here to our advantage. What are the multiple components of R multiple components of R we see, this is R_1, this is R_2. And you will go until how many other equations are there this is R_m.

(Refer Slide Time: 08:01)



$$\underbrace{R(W+\Delta W)}_{\downarrow} = R(W) + \underbrace{\frac{\partial R}{\partial \underset{a}{w_1}} \underset{\Delta a}{\Delta w_1} + \frac{\partial R}{\partial \underset{b}{w_2}} \underset{\Delta b}{\Delta w_2}}$$

$$R(W+\Delta W) = R(W) + J\,\Delta A$$

$$\underbrace{\begin{bmatrix} \frac{\partial R_1}{\partial a} & \frac{\partial R_1}{\partial b} \\ \frac{\partial R_2}{\partial a} & \frac{\partial R_2}{\partial b} \\ \vdots & \vdots \\ \frac{\partial R_m}{\partial a} & \frac{\partial R_m}{\partial b} \end{bmatrix}}_{J.} \underbrace{\begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix}}_{\Delta A}$$

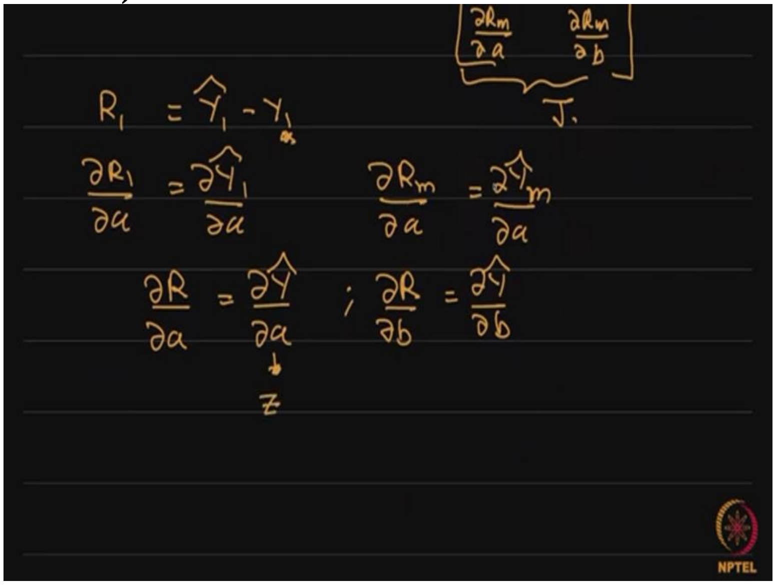
So now, this is going to look like for each row, you will have something like $\frac{\partial R_1}{\partial w_1}$, $\frac{\partial R_1}{\partial w_2}$, multiplied by Δw_1, Δw_2. If I open out the set of equations for every row, then this will look like $\frac{\partial R_2}{\partial w_1}$, $\frac{\partial R_2}{\partial w_2}$, you keep on going till del $\frac{\partial R_m}{\partial w_1}$, $\frac{\partial R_m}{\partial w_2}$, in case you have only 2 W's, which are unknowns, or you can call this for our case, I am going to make this a and b.

So that we can easily apply it to the specific case that we looked at delta a delta b. Same thing, this I am calling delta a, this I am calling delta b. Some people find this notation easy this is a, this is b. So, these are the 2 parameters you can think w_1, w_2 are even. Now, what does this look like? So, this looks like the following. This, of course, is once again, the Jacobian matrix so, we can say that,

$$R(W + \Delta W) = R(W) + J\Delta A$$

In order to not get confused with what the Jacobian is, let us work this out a little bit further.

(Refer Slide Time: 09:53)



$$\underbrace{\begin{bmatrix} \frac{\partial R_m}{\partial a} & \frac{\partial R_m}{\partial b} \end{bmatrix}}_{J.}$$

$$R_1 = \hat{Y}_1 - Y_1$$

$$\frac{\partial R_1}{\partial a} = \frac{\partial \hat{Y}_1}{\partial a} \qquad \frac{\partial R_m}{\partial a} = \frac{\partial \hat{Y}_m}{\partial a}$$

$$\frac{\partial R}{\partial a} = \frac{\partial \hat{Y}}{\partial a} \; ; \; \frac{\partial R}{\partial b} = \frac{\partial \hat{Y}}{\partial b}$$

$$\downarrow$$

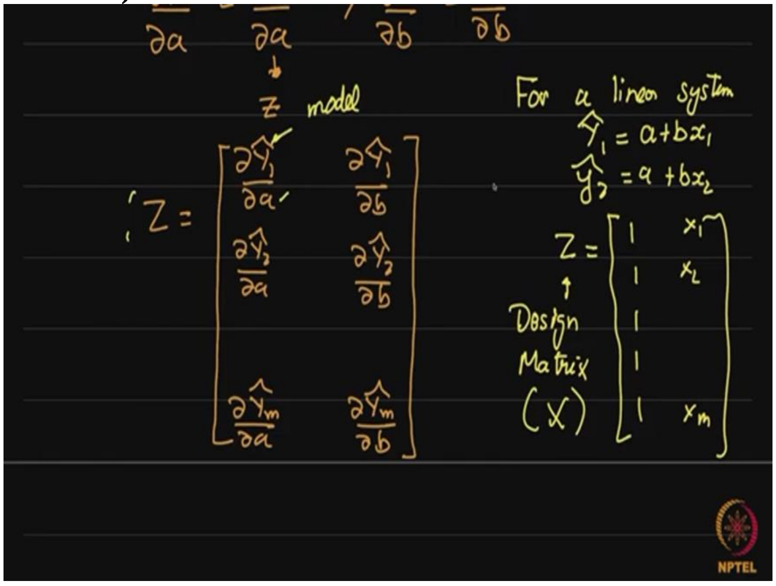
$$z$$

What is R_1?

$$R_1 = \widehat{Y_1} - Y_1$$

So, what is $\frac{\partial R_1}{\partial a}$, this is simply a $\frac{\partial \widehat{y_1}}{\partial a}$ because this part is 0. Similarly, is $\frac{\partial R_m}{\partial a}$ is simply going to be $\frac{\partial \widehat{y_m}}{\partial a}$, and in general, $\frac{\partial R}{\partial a}$ will be $\frac{\partial \hat{Y}}{\partial a}$. Similarly, $\frac{\partial R}{\partial b}$, is going to be $\frac{\partial \hat{Y}}{\partial b}$. So, we can now put this together and let us call these as z. Okay So this is the standard notation that is used.

(Refer Slide Time: 10:45)



z model

$$Z = \begin{bmatrix} \frac{\partial \hat{Y}_1}{\partial a} & \frac{\partial \hat{Y}_1}{\partial b} \\ \frac{\partial \hat{Y}_2}{\partial a} & \frac{\partial \hat{Y}_2}{\partial b} \\ \vdots & \vdots \\ \frac{\partial \hat{Y}_m}{\partial a} & \frac{\partial \hat{Y}_m}{\partial b} \end{bmatrix}$$

For a linear system

$$\hat{Y}_1 = a + bx_1$$

$$\hat{Y}_2 = a + bx_2$$

$$Z = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \\ 1 & \\ 1 & x_m \end{bmatrix}$$

$\uparrow$ Design Matrix (X)

So, I am going to say Z is the matrix,

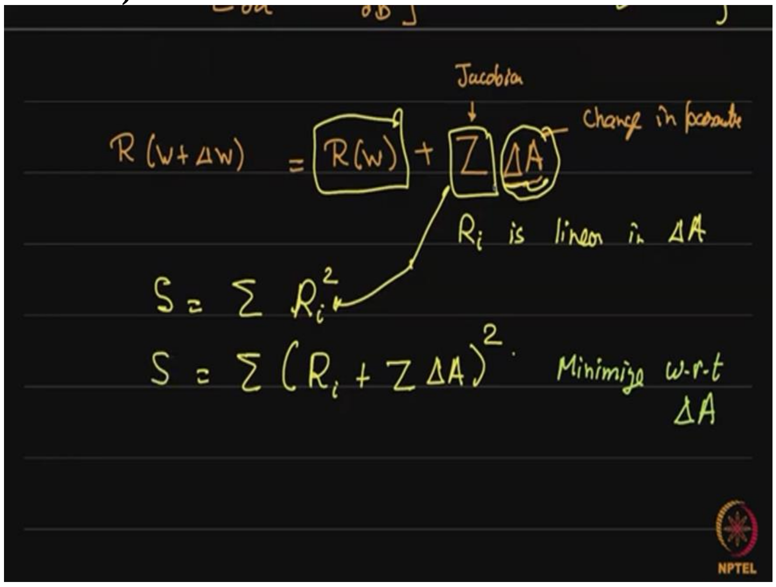
$$Z = \begin{bmatrix} \frac{\partial \widehat{y_1}}{\partial a} & \frac{\partial \widehat{y_1}}{\partial b} \\ \frac{\partial \widehat{y_2}}{\partial a} & \frac{\partial \widehat{y_2}}{\partial b} \\ \cdot & \cdot \\ \frac{\partial \widehat{y_m}}{\partial a} & \frac{\partial \widehat{y_m}}{\partial b} \end{bmatrix}$$

Now, just before we move a little bit further, in case some of you are confused, remember, for a linear system. So let us say we have a linear system, and in that case, $\hat{y}$ could be or let us say $\widehat{y_1}$ would be $a + bx_1$ since I am using a and b.

Similarly, $\widehat{y_2}$ would be $a + bx_2$ and then Z therefore, would be 1, 1, 1, 1, 1 and you would have x_1, x_2 up till x_m, which is exactly the design matrix. Remember the capital matrix X that we used for our normal equations, it sort of gets resurrected here, okay. So, in fact, the most general derivation is the one that we are doing. This works both for linear systems as well as nonlinear systems.

So, gauss Newton, when you apply to linear systems would converge in 1 step and it will give you exactly the normal equation method, okay. So, this is just to tell you that there is nothing strange, all we are doing is we are differentiating the model with respect to the parameters. So, it might look strange while we are doing the derivation, but that is exactly what is going on. So, let us go back to this equation, what were we trying to do, we were trying to write the nonlinear regression as a linear regression problem in the changes.

(Refer Slide Time: 12:46)



Jacobian

$$R(w+\Delta w) = \boxed{R(w)} + \boxed{Z}\,\Delta A$$

Change in parameter

R_i is linear in ΔA

$$S = \sum R_i^2$$

$$S = \sum (R_i + Z\Delta A)^2.$$

Minimize w.r.t ΔA

So, remember, now,

$$R(W + \Delta W) = R(W) + Z\Delta A$$

So, this is the Jacobian this is the change in parameters and this is what we are trying to solve for. Now, what we are saying is S is,

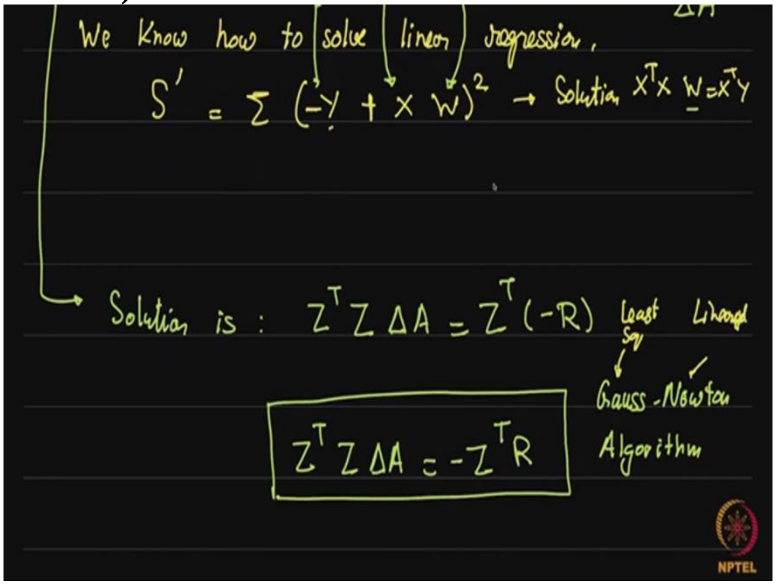
$$S = \sum {R_i}^2$$

but R_i is this and R_i is linear in delta A, I hope you can notice that okay. So, this term is known, this term is known at the particular W that we have chosen all the variable there is, is delta A.

So, remember, now, R of W plus delta W is R at whatever the W is, plus, I am now going to call it the Jacobian multiplied by the change. So, this is the Jacobian this is the change in parameters and this is what we are trying to solve for. Now, what we are saying is S is sigma of R_i-square, but R_i is this and R_i is linear in delta A, I hope you can notice that okay. So, this term is known, this term is known at the particular W that we have chosen all the variable there is, is delta A.

Then update the guess just like what we were doing for our nonlinear goods so far. So, it works in exactly the same way as before. This becomes clearer when you see the code.

(Refer Slide Time: 14:39)



ΔA

We Know how to solve linear regression,

$S' = \sum(-Y + XW)^2 \rightarrow$ Solution $X^TX\underline{W} = X^TY$

Solution is : $Z^TZ\Delta A = Z^T(-R)$ Least Sq Linearized

Gauss-Newton Algorithm

$\boxed{Z^TZ\Delta A = -Z^TR}$

Now, we already know how to solve linear regression products. So, for example if S' was,

$$S' = \sum(-Y + XW)^2$$

sigma of in our case, it was Y. Let me write it this way, $(-Y + XW)^2$, then the solution was,

$$X^TXW = X^TY$$

Now, let us compare apples to apples here. In fact, let us make this minus Y and this plus just so that we have, we do not have to do too many sign changes.

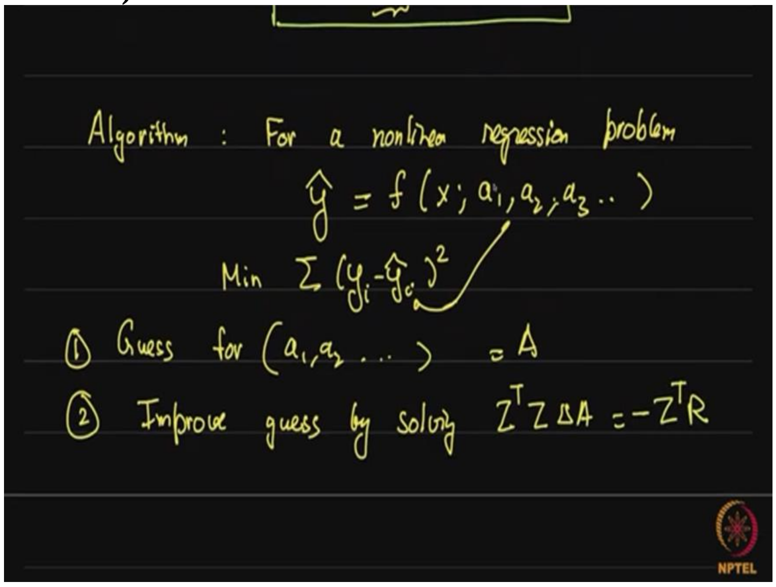
So now we can see the correspondences, the correspondences are X, Z has become X, as I just told you, W has become delta a, this is just like the linear cases that I showed you earlier and R has become minus Y. So now we can write our equation. The solution to this is you can derive it but it is easier to use our earlier results instead of X transpose X, it is going to be Z transpose Z instead of W it is going to be delta A = X transpose which is Z transpose multiplied by Y.

Now we are going to get minus R Okay. So, another way of saying it is,

$$Z^T Z \Delta A = -Z^T R$$

This is called the gauss Newton algorithm. The gauss part comes from using the least square and the Newton part is because we linearized just like we do with previous nonlinear linear systems.

(Refer Slide Time: 17:15)



Algorithm : For a nonlinear regression problem

$$\hat{y} = f(x; a_1, a_2, a_3 ..)$$

$$\text{Min } \Sigma (y_i - \hat{y}_i)^2$$

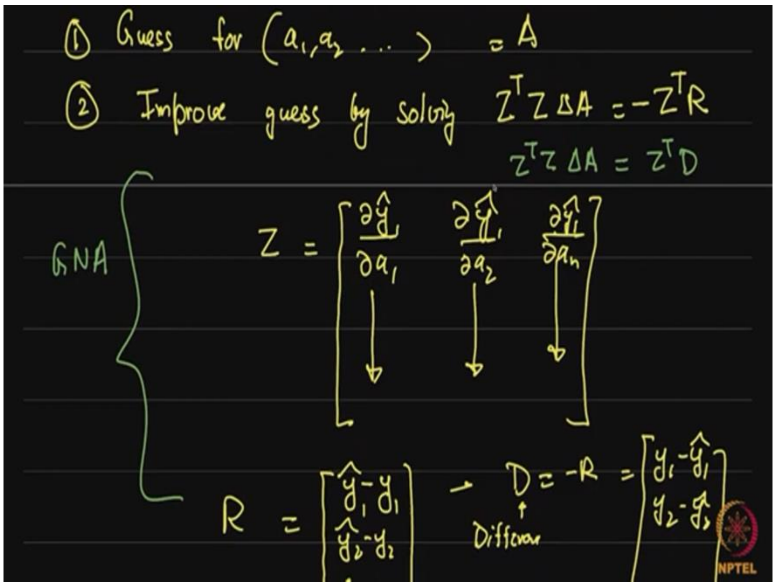
① Guess for $(a_1, a_2 \ldots) = A$

② Improve guess by solving $Z^T Z \Delta A = -Z^T R$

So, what is the algorithm? The algorithm is like this. For a nonlinear regression problem, so, let us say you have y-hat is some function of x but more importantly, the parameters are a1, a2, a3, or W1, W2, W3 whichever way you wanted, I will use a because I have been using a, b, c right now, if we want to minimize sigma of yi - y-hat i square and y-hat i depends on these parameters instead I guess, so, the first step is guess for a1, a2 which we will call A initially, then improve guess by solving this equation,

$$Z^T Z \Delta A = -Z^T R$$

(Refer Slide Time: 18:30)



① Guess for $(a_1, a_2 \ldots) = A$

② Improve guess by solving $Z^T Z \Delta A = -Z^T R$

$$Z^T Z \Delta A = Z^T D$$

GNA

$$Z = \begin{bmatrix} \frac{\partial \hat{y}}{\partial a_1} & \frac{\partial \hat{y}_1}{\partial a_2} & \frac{\partial \hat{y}_1}{\partial a_n} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$R = \begin{bmatrix} \hat{y}_1 - y_1 \\ \hat{y}_2 - y_2 \end{bmatrix} \rightarrow \underset{\text{Difference}}{D} = -R = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \end{bmatrix}$$

Where Z is the Jacobian matrix to calculate del y-hat del a1 del y-hat del a2 etcetera, these entire columns for each data point so, for example, del y1 del y1 del y1 hat del an and R is simply the residual is,

$$R = \begin{bmatrix} \widehat{y_1} - y_1 \\ \widehat{y_2} - y_2 \\ . \\ \widehat{y_m} - y_m \end{bmatrix}$$

Some people also call the negative of this as D for the difference. So, this would be,

$$D = \begin{bmatrix} y_1 - \widehat{y_1} \\ y_2 - \widehat{y_2} \\ . \\ y_m - \widehat{y_m} \end{bmatrix}$$

And that way you can avoid the minus.

So, another way to write it is,

$$Z^T Z \Delta A = Z^T D$$

either of these are fine. So, this in essence is our gauss newton algorithm. It is messy I will confess it is messy in the sense not in terms of programming, which I will show you, it is actually a very simple program to write. When you calculate it by hand, it is messy. So again, I am going to recommend that the code that I am putting up, you at least try a few steps by hand.

Typically, within IIT, where we teach this course, we do insist that people do solve this equation by hand in all the occurrences of inverse heat transfer that we have done here, insist that people do that for the exams. So anyway, it is good practice for you to do it. Let me however, show you a short code, which goes back to our problem which we tried to solve by gradient descent and does it by the gauss newton algorithm. So let us see that code now.

(Video Starts: 20:46)

So here is the problem once again, I have taken the same data set, notice this thing of theta being $a(1 - e^{-bt})$, again, I am going to make the change from t to x and x to y, etcetera, I will make all those changes, I will retain the a and b. Same initial guesses, we saw that with gradient descent, it took a large number of guesses to get somewhere to convergence.

And even then, it turned out that the final answer was not physical, though I did not show you why it is not physical but our aim remains the same. To find out the optimal coefficients, we are going to start in the same way, you can see that I have input the same t and theta I have done the same thing of x being t and y being theta. And up until here is just a = 35, b = 0.004. And I am just taking 5 iterations, which I have called epochs here.

Now let us come to the derivative again, you would have seen the same expression within gradient descent too, notice y is $a(1 - e^{-bx})$, dyda, which we also calculated before so I did not do it once again in the previous video is simply $(1 - e^{-bx})$ and dydb is I told you to be cautious here a times x times e^{-bx}. So, till that time is the same I have called it the gradient descent step.

Really speaking, I should say that iterations start here. Okay. So, this is our initial guess. And we calculate the gradients after the step has started. You should not get confused by the t here, I could probably call this is. As actually let me retain the t so as to not confuse the program right now. Okay So y is calculated, or dyad is calculated dydb is calculated and I have just used that to calculate Z. So let us just run it so that you have an intuition for what Z looks like.

So, you come here and take a step. Now what is the size of Z? Notice dyda is a vector of size 6, it is all at every single point at every single t point, okay, so the point at time you actually calculate whatever y is. So, whatever dyda is, similarly for B by dydb, okay, so you calculate all of them. Now you put them together, and that is when you have Z. Z is now you can see a 6 cross 2 vector why 6?

6 is the number of data points we took in time and 2 2 is the number of a and b, the number of parameters that we have. So, the derivative of how y-hat our prediction changes the first variable, and how y-hat changes with the second variable. That is basically what we are trying to guess in Z, the analogy is the design vector as I told you 1 and x, so this is what is known as the feature vector. Now we calculate the left-hand side of this matrix.

So, notice the matrix we are solving Z transpose Z delta A is Z transpose D, if you wish, we can call this minus the Z transpose R. But I will stick with the D because that is the way I have programmed it. So, I will change the negative sign back. So, nothing minus of R is simply D. So, you can see that here, the left-hand side is simply Z transpose Z the right-hand side is Z transpose multiplied by D, D is y - yh. I did not specify it separately when you write a program in Excel or if you do it by hand

you will have to actually calculate each one of these differences separately, what is y, and what is y-hat, you can see, the value of y and the value of y-hat are vastly different. So, you can see that within this screen itself. So now here we solve for delta a, again, what I am doing is the same as solving inverse of the left-hand side multiplied by the right-hand side, as I told you during the normal equations, case, also, is the same thing.

So now once we do that, we actually have, let me just take a quick step, you have a solution. Now notice, since the left-hand side is now a 2 cross 2 matrix, the right-hand side is simply a 2 cross 1 vector, because it is the difference, pre multiplied by is Z prime. Now Z prime has the right size to reduce this just to a 2 cross 1. So really speaking, we are only solving a 2 cross 2 system each time again, I highly recommend you do this by hand, both for the sake of exam and for the sake of your own understanding.

So now you calculate a new ‘a’, and you calculate a new b. Okay, this actually turns out to be much, much, much more stable. And I ran this for 5 epochs. And only the final values displayed for some reason, just rerun this again. That is because of the breakpoint. Let me run this again. So, you can now see, see at the bottom, there is some amount of convergence and this is actually a physical value.

So, the value of D which has a time constant actually needs to be which we derived, if you might remember, it actually needs to be fairly small for this problem, based on how we did it, if you wish, we can run this for a few more epochs. Or let us say I run this for 10 epochs or 10 iterations. And you can see that it has more or less converge see the last final 2 iterations, you can see both the ‘a’ variable as well as the ‘b’ variable have actually converged just like the usual Newton methods and these turned out to be physical too.

So, this actually converges just within an iteration. So of course, each iteration is slightly painful, but this is far better than gradient descent. So, this is the major advantage of the gauss newton algorithm and it is a powerful algorithm for nonlinear regression. Typically, almost all practical nonlinear regressions are done on some variant of this, what these variants are, we will discuss next week, because even gauss Newton has some problems with stability

And this is connected to the air conditioning problem, which I had discussed with you in the first week. So, what we will see next week is some minor modifications of the gauss newton algorithm and several other algorithms that we have seen so far in the course. So, thank you, I hope to see you in the next week. Thanks.

(Video Ends: 28:17)